STPA HANDBOOK

영문/한글판 v1.1

NANCY G. LEVESON
JOHN P. THOMAS

MARCH 2018

이 핸드북은 STPA를 실 시스템에 적용하고자 하는 사람을 대상으로 한다. STPA의 이론적 기초는 다른 자료를 참고할 수 있으므로 여기서는 논하지 않는다. 여기서의 우리의 목표는 실제 프로젝트에서 STPA를 시작하는 사람들에게 방향을 제시하거나 STPA를 가르치기 위해 필요한 자료를 제공하는 것이다.

Translated by TTA (supported by Ministry of Science and ICT)

과학기술정보통신부 Ministry of Science and ICT

TTA 한국정보통신기술협회 Telecommunications Technology Association

COPYRIGHT © 2018 BY NANCY LEVESON AND JOHN THOMAS. ALL RIGHTS RESERVED. THE UNALTERED VERSION OF THIS HANDBOOK AND ITS CONTENTS MAY BE USED FOR NON-PROFIT CLASSES AND OTHER NON-COMMERCIAL PURPOSES BUT MAY NOT BE SOLD.

본 핸드북은 과학기술정보통신부의 “SW공학경쟁력강화” 사업의 결과물입니다.

과학기술정보통신부
Ministry of Science and ICT

한국정보통신기술협회
Telecommunications Technology Association

TABLE OF CONTENTS

Preface	3
1. Introduction	4
2. How to do Basic STPA	14
3. Integrating STPA into Your System Engineering Process	54
4. Workplace Safety	72
5. Organizational and Social Analysis	86
6. Identifying Leading Indicators of Risk	101
7. Designing an Effective Safety Management System	116
8. Integrating STPA into a Large Organization	142
Appendix A: Examples of hazards	146
Appendix B: Examples of control structures	148
Appendix C: Examples of UCA tables	156
Appendix D: Guidelines for designing and evaluating a safety management system	163
Appendix E: More about why decomposition does not work for complex, computer-based systems	167
Appendix F: Basic engineering and system engineering concepts for non- engineers	169
Appendix G: A new model to assist in causal scenario generation	178

Acknowledgements: Many people provided comments on early drafts of this handbook. We want to thank them for their help, including (alphabetically) Matthew Boesch (Ford), Diogo Silva Castilho (MIT and Brazilian Air Force), Christopher Degni (Akamai), Mikela Chatzimichailidou (Imperial College), Rashmi Hedge (Ford), Stephen Johnson (Fluor), Ioana Koglbauer (University of Graz), Galvani Lacerda (Embraer), Simon Lucchini (Fluor), Shem Malquist (FedEx), Maj. Dan Montes (U.S. Air Force), Sandro Nuesch (Ford), Felipe Oliveira (Embraer), Todd Pawlicki (U.C. San Diego Medical Center), Kep Peterson (Akamai), Martin Rezjek (Zurich University of Applied Sciences), Lori Smith (Boeing), Michael Stone (Akamai), Maj. Sarah Summers (U.S. Air Force), Mark Vernacchia (General Motors), Sarra Yako (Ford), Col. William Young (U.S. Air Force), and members of the U.K. National Cyber Security Centre.

목차

머리말 3

1. 소개 4

2. 기본적인 STPA 분석 수행 방법 14

3. STPA를 시스템 엔지니어링 프로세스에 통합 54

4. STPA를 사용한 작업장 안전 72

5. 조직 및 사회 분석 86

6. STPA를 사용한 선행지표 식별 101

7. 안전 관리 시스템 116

8. 조직에 STPA 도입 142

부록 A: 위험(Hazard)의 예 146

부록 B: 컨트롤 스트럭처의 예 148

부록 C: UCA 테이블의 추가 예 156

부록 D: 안전 컨트롤 스트럭처에 포함될 책임 163

부록 E: 소프트웨어 집약적인 시스템에 대한 분석적 분해의 한계(항공전자 사례) 167

부록 F: 비-엔지니어를 위한 기본적인 공학 및 시스템 공학 개념 169

부록 G: 원인 시나리오 생성을 도와주는 컨트롤 모델 178

감사의 말: 많은 분들이 핸드북의 초안에 대한 의견을 주었습니다. 도움에 감사드립니다(알파벳순). Matthew Boesch(Ford), Diogo Silva Castilho(MIT and Brazilian Air Force), Christopher Degn(Akamai), Mikela Chatzimichailidou (Imperial College), Rashmi Hedge(Ford), Stephen Johnson(Fluor), Ioana Koglbauer(University of Graz), Galvani Lacerda(Embraer), Simon Lucchini(Fluor), Shem Malquist(FedEx), Dan Montes(U.S. Air Force), Sandro Nuesch(Ford), Felipe Oliveira Embraer), Todd Pawlicki(UC San Diego Medical Center), Kep Peterson(Akamai), Martin Rezjek(University of Applied Sciences), Lori Smith(Boeing), Michael Stone(Akamai), Sarah Summers(U.S. Air Force), Mark Vernacchia(General Motors), Sarra Yako(Ford), Col William Young(U.S. Air Force), and members of the U.K. National Cyber Security Centre.

Preface

This handbook is intended to assist those who want to start using STPA or who want to try using it for more than simple hazard analysis. As such, we tried to provide examples and have included more examples in the appendices. An appendix is also provided to explain some basic engineering concepts needed to understand and use the handbook to those who are not trained in engineering. Other new concepts are introduced in the main chapters of the handbook itself.

The introduction provides a brief introduction to STAMP, the accident causality model underlying STPA. It also shows examples accidents and explains why STPA is needed for today's complex, software-intensive systems.

The next chapter is an in-depth tutorial on how to perform STPA. It will be useful for the beginner as well as for those who have tried STPA and want to improve their results.

The rest of the handbook describes uses for STPA, including how to integrate it into a standard system engineering process, its use in workplace safety, using STPA for organizational analysis and emergent system properties other than safety, using STPA to provide leading indicators of increasing risk, designing an effective safety management system, and cyber security.

The final chapter describes what we have learned about integrating STPA into a large organization and how to structure the STPA process to make it most effective but also least disruptive to the enterprise.

Our goal in writing this handbook is to provide a guide for those who are actually using STPA rather than writing an academic primer. Therefore, we have omitted references to other work, etc. There are many other sources that provide this type of information but few that focus on instruction and ways to use STPA. To find many examples, published papers, theses, etc., see http://psas.scripts.mit.edu/home/

We encourage users of the handbook to work on examples relevant to their industry as they go through the handbook.

머리말

이 핸드북은 STPA를 시작하거나 단순 위험 분석 이상의 용도로 사용하려는 사용자를 돕기 위한 것이다. 이를 위해 예제를 제시하고, 부록에 더 많은 예제를 포함하였다. 또한 부록에는 공학 교육을 받지 않은 사람들이 핸드북을 이해하고 사용하는 데 필요한 몇 가지 기본 공학 개념을 설명하였다. 다른 새로운 개념들은 핸드북의 주요 장에서 소개되고 있다.

소개(1장)에서는 STPA의 기초가 되는 사고 원인(accident causality) 모델인 STAMP에 대해 간략히 소개한다. 또한 사고 사례를 보여주고 오늘날의 복잡한 소프트웨어 집약적인 시스템에 STPA가 필요한 이유를 설명한다.

다음 장에서는 STPA를 수행하는 방법에 대해 자세히 설명한다. 초보자뿐만 아니라 STPA를 적용하여 분석 결과를 개선시키고자 하는 사람들에게 유용할 것이다.

핸드북의 나머지 부분에는 STPA를 표준 시스템 엔지니어링 프로세스에 통합, 작업장 안전에 활용, 조직 분석 및 새로운 시스템 속성에 적용, 증가하는 리스크의 선행지표 제공, 효과적인 안전 관리 시스템 설계, 사이버 보안 등에 적용하는 사례를 기술하였다.

마지막 장에서는 STPA를 대규모 조직에 통합하는 과정에서 도출한 교훈에 대해 소개하고, 가장 효과적이면서도 기업에 미치는 영향을 최소화하여 STPA 프로세스를 구조화하는 방법을 설명한다.

이 핸드북은 학술적 입문서가 아니라, STPA를 실제로 사용하는 사람들을 위한 안내서이다. 따라서 기타 참조 자료는 생략했다. 이러한 유형의 정보를 제공하는 다른 출처는 많이 있지만 STPA를 사용하는 지침과 방법에 중점을 두는 출처는 거의 없다. 다른 많은 예제나 논문 등은 http://psas.scripts.mit.edu/home/을 참조할 수 있다.

핸드북을 활용하는 사람들이 이 핸드북을 따라가면서, 각자의 산업에 관련된 사례에 적용해 보기를 권한다.

Chapter 1: Introduction

Nancy Leveson

STPA (System-Theoretic Process Analysis) is a relatively new hazard analysis technique based on an extended model of accident causation. In addition to component failures, STPA assumes that accidents can also be caused by unsafe interactions of system components, none of which may have failed. Some of the advantages of STPA over traditional hazard/risk analysis techniques are that:

- Very complex systems can be analyzed. "Unknown unknowns" that were previously only found in operations can be identified early in the development process and either eliminated or mitigated. Both intended and unintended functionality are handled.
- Unlike the traditional hazard analysis methods, STPA can be started in early concept analysis to assist in identifying safety requirements and constraints. These can then be used to design safety (and security) into the system architecture and design, eliminating the costly rework involved when design flaws are identified late in development or during operations. As the design is refined and more detailed design decisions are made, the STPA analysis is also refined to help make more and more detailed design decisions. Complete traceability from requirements to all system artifacts can be easily maintained, enhancing system maintainability and evolution.
- STPA includes software and human operators in the analysis, ensuring that the hazard analysis includes all potential causal factors in losses.
- STPA provides documentation of system functionality that is often missing or difficult to find in large, complex systems.
- STPA can be easily integrated into your system engineering process and into model-based system engineering.

Many evaluations and comparisons of STPA to more traditional hazard analysis methods, such as fault tree analysis (FTA), failure modes and effects criticality analysis (FMECA), event tree analysis (ETA), and hazard and operability analysis (HAZOP) have been done.[1] In all of these evaluations, STPA found all the causal scenarios found by the more traditional analyses but it also identified many more, often software-related and non-failure, scenarios that the traditional methods did not find. In some cases, where there had been an accident that the analysts had not been told about, only STPA found the cause of the accident. In addition, STPA turned out to be much less costly in terms of time and resources than the traditional methods.

Theoretical Foundation

To use STPA requires little understanding of the underlying theoretical, mathematical foundation, but an intuitive understanding is helpful. If this topic does not interest you, we suggest you skip to the next section.

STPA is based on System Theory. Systems theory was developed after World War II to cope with the increasingly complex systems with advanced technology that were being created. First, it is important to understand what it was replacing.

[1] Information about some of these were presented at past MIT STAMP/STPA workshops. Presentations can be found at http://psas.scripts.mit.edu/home/

1장: 소개

Nancy Leveson

STPA(System-Theoretic Process Analysis)는 사고 원인의 확장된 모델을 기반으로 하는 비교적 새로운 위험(hazard) 분석 기술이다. STPA는 컴포넌트의 오류 이외에도 시스템 컴포넌트 중 어느 것도 실패하지 않았더라도, 시스템 컴포넌트의 안전하지 않은 상호작용으로 인해 사고가 발생할 수 있다고 본다. 전통적인 위험/리스크 분석 기법에 비해 STPA는 다음과 같은 장점을 가진다.

- 매우 복잡한 시스템을 분석할 수 있다. 이전에는 시스템 운영 시에만 발견되었던 "알 수 없는 무언가"를 개발 프로세스 초기에 식별하여 제거하거나 완화할 수 있다. 의도한 기능과 의도하지 않은 기능을 모두 다룰 수 있다.
- STPA는 전통적인 위험 분석 방법과는 달리 초기 개념 분석에서 시작하여 안전 요구사항 및 안전 제약사항을 식별하는 데 도움을 준다. 이는 시스템 아키텍처 및 시스템 설계에 안전(및 보안)을 설계하고, 개발 후반 또는 운영 중에 설계 결함이 식별되어 발생하는 비용이 많이 드는 재작업을 제거할 수 있다. 설계가 상세화되고 세부적인 설계가 결정되면 STPA 분석 또한 보다 상세한 설계 결정을 내릴 수 있도록 세분화된다. 요구사항에서 모든 시스템 결과물(artifacts)에 이르기까지 완벽한 추적성을 쉽게 유지할 수 있으므로 시스템 유지보수(maintainability) 및 진화(evolution)에 용이하다.
- STPA는 분석 과정에 소프트웨어 및 운영자가 포함되어 있으므로 위험 분석 시 손실의 모든 잠재적 요인을 포함하여 분석하였음을 보장한다.
- STPA는 크고 복잡한 시스템에서 자주 누락되거나 찾기 어려운 시스템 기능(functionality)에 대한 문서를 제공한다.
- STPA는 시스템 엔지니어링 프로세스 및 모델 기반 시스템 엔지니어링에 쉽게 통합될 수 있다.

FTA(Fault Tree Analysis), FMECA(Failure Mode and Effects Criticality Analysis), ETA(Event Tree Analysis), HAZOP(Hazard and operability analysis)와 같은 기존의 위험 분석 방법과 STPA 간의 많은 평가와 비교가 이루어져 왔다.[1] 이러한 모든 평가에서 STPA는 전통적인 분석 방법에서 발견된 모든 원인 시나리오를 도출하였을 뿐만 아니라, 더 많은 시나리오를 식별하였고, 때로는 전통적 방법에서는 도출하지 못한 소프트웨어 관련 원인 시나리오, 고장이 아니더라도 사고로 이어지는(non-failure) 시나리오도 도출하였다. 몇몇 경우에는 분석가들이 들은 바가 없었던 사고에 대해 오직 STPA만이 사고의 원인을 발견했다. 추가적으로 STPA는 기존 방법보다 시간과 자원 측면에서 훨씬 적은 비용이 드는 것으로 나타났다.

이론적 기초

STPA를 사용할 때 기본적인 이론적, 수학적 기초를 거의 이해할 필요가 없으며, 직관적인 이해력이 필요하다. 만약 이 주제에 관심이 없다면 다음 섹션으로 건너뛰어도 좋다.

STPA는 시스템 이론을 기반으로 한다. 시스템 이론은 제2차 세계대전 이후 첨단기술로 점차 복잡해지는 시스템에 대응하기 위해 개발되었다. 우선, 무엇을 대체하고 있는지 이해하는 것이 중요하다.

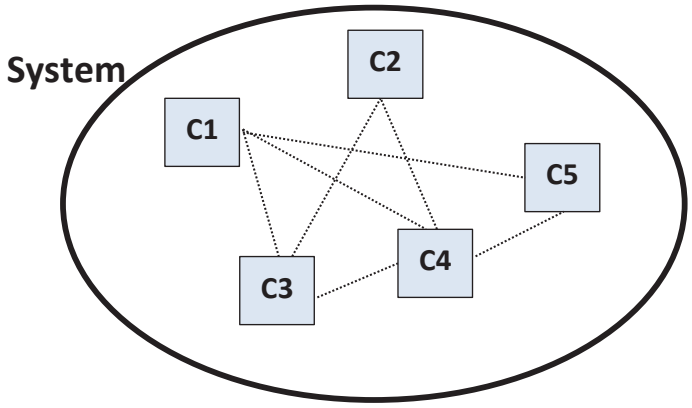
---

[1] 이 중 일부 정보는 과거의 MIT STAMP/STPA 워크샵에서 발표되었다. 프리젠테이션은 http://psas.scripts.mit.edu/home/에서 찾을 수 있다.

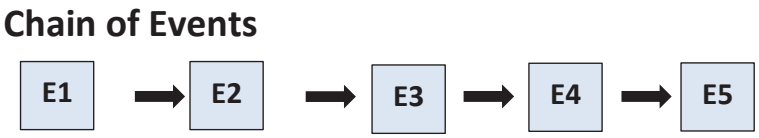
Analytical Decomposition (the traditional approach)

For centuries complexity has been handled by breaking the system into smaller components, examining and analyzing each component in isolation, and then combining the results in order to understand the behavior of the composed components.

The physical or functional components are broken into distinct components that are assumed to interact in direct and known ways. The functional components of an aircraft, for example, might be propulsion, navigation, attitude control, braking, cabin environment control, and so on. Aircraft may also be decomposed into physical components, such as fuselage, engines, wings, stabilizers, nozzles, and so on. Notice that mappings may be made between the functional components and the physical ones but the mappings are assumed to be direct and known.



Behavior, in contrast, is traditionally modeled as separate events over time, where each event is the direct result of the preceding event(s). There may be multiple preceding or following events combined through the use of AND/OR logic but that simply results in multiple chains (with the AND/OR logic used to reduce the number of separate chains needing to be specified by putting them together into trees).



The events may be grouped into operational stages such as for aircraft, pushback, taxi, take off, climb, cruise, approach, and touchdown.

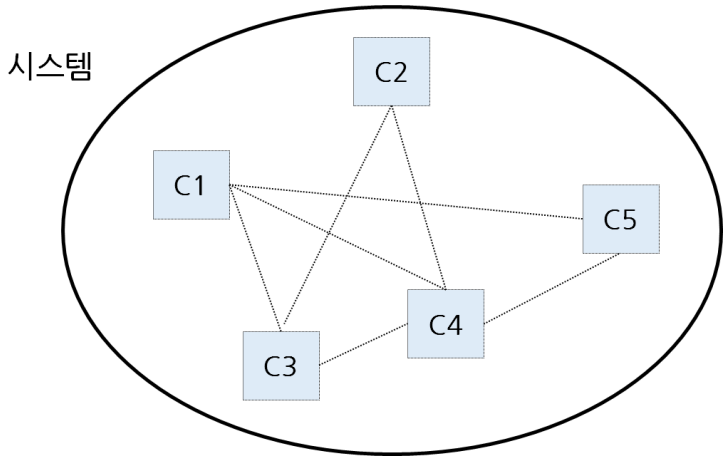
Once the system has been broken into components (pieces), the components are analyzed separately and the results of the separate analyses are combined to create a system analysis. For example, if the weight of an complex object is the analysis goal, the separate pieces may be weighed and the result combined to get the system weight. As another common example, the system reliability is usually evaluated by evaluating the reliability of the individual components and then the component reliabilities are combined mathematically to evaluate the system reliability.

The success of this type of decompositional or reductionist approach relies on the assumption that the separation and individual analysis does not distort the phenomenon or property of interest. More specifically, the approach works if:

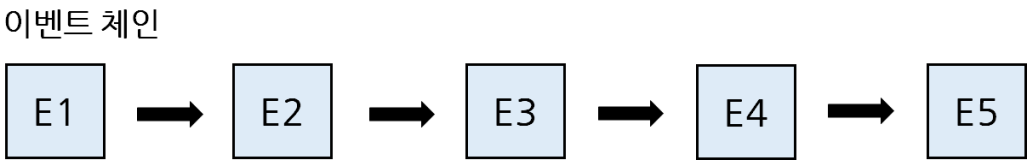
분석적 분해(전통적인 접근법)

수세기 동안 복잡도(complexity)를 다루기 위해 시스템을 더 작은 컴포넌트로 분해하고 각 컴포넌트를 개별적으로 관찰하고 분석한 다음, 그 결과를 결합하여 컴포넌트의 동작을 이해해 왔다.

물리적 또는 기능적 컴포넌트는 직접적이고 잘 알려진 방식으로 상호작용한다고 가정되는 개별적인 컴포넌트로 나누어졌다. 예를 들어 항공기의 기능적 컴포넌트는 추진력, 항법, 자세 제어, 제동, 기내 환경 제어 등이 될 수 있다. 또한, 항공기는 동체, 엔진, 날개, 안정 장치, 노즐과 같은 물리적 컴포넌트로 분해될 수 있다. 기능적 컴포넌트와 물리적 컴포넌트 간에 매핑이 이루어질 수도 있지만 매핑은 직접적이며 잘 알려진 것으로 가정한다.



동작은, 반대로, 시간 경과에 따른 별개의 이벤트로 모델링 되었으며, 여기서의 각 이벤트는 앞선 이벤트의 직접적인 결과이다. AND/OR 로직을 사용하여 다수의 선행 또는 후행 이벤트를 결합할 수 있는데, 트리에 배치해야 하는 개별 체인 수를 AND/OR 로직을 사용하여 줄임으로써 멀티플 체인으로 단순하게 만들 수 있다.



이벤트는 운영 단계에서 그룹으로 구분할 수 있는데 항공기의 경우, 푸시백(pushback), 지상활주(taxi), 이륙(takeoff), 상승(climb), 운항(cruise), 어프로치(approach) 및 터치다운(touchdown)으로 구분할 수 있다.

시스템이 컴포넌트(조각)로 분해되면 컴포넌트가 개별적으로 분석되고, 개별 분석 결과가 결합되어 시스템 분석 결과를 만든다. 예를 들어 분석 목표가 복잡한 객체의 무게인 경우, 분리된 조각의 무게를 측정하고 그 결과를 합하여 시스템 무게를 구할 수 있다. 또 다른 흔한 예로 시스템 신뢰도는 보통 개별 컴포넌트의 신뢰도를 평가한 다음 컴포넌트 신뢰도를 수학적으로 결합하여 시스템 신뢰도를 평가한다.

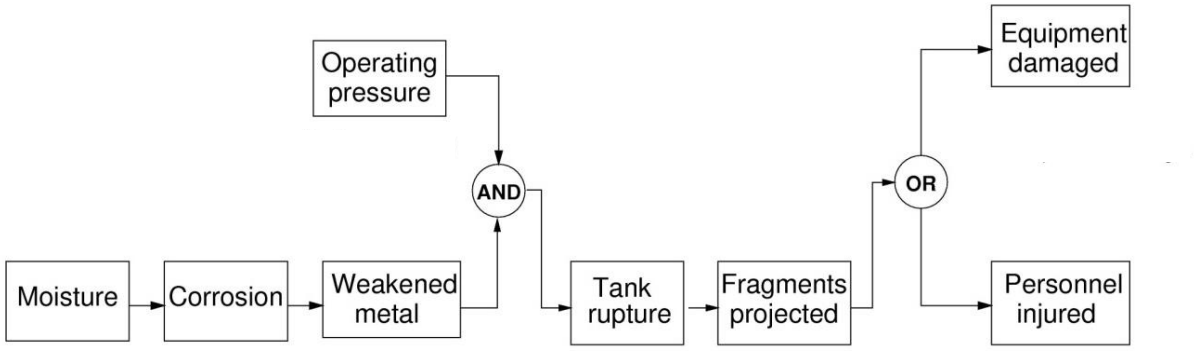
이러한 유형의 분해 또는 환원법적(reductionist) 접근 방식의 성공 여부는 컴포넌트를 분리하여 개별적으로 분석하는 것이 관심 대상의 현상이나 속성을 왜곡하지 않는다는 가정에 기반한다. 보다 구체적으로 이러한 접근법은 다음과 같은 경우에 적용 가능하다.

- Each component or subsystem operates independently. If events are modeled, then they are independent except for the immediately preceding and following events.
- Components act the same when examined separately (singly) as when they are playing their part in the whole.
- Components and events are not subject to feedback loops and other indirect interactions.
- The interactions among the components or events can be examined pairwise and combined into a composite value.

If these assumptions are not true, then the simple composition of the separate analyses will not accurately reflect the system value.

Why is all of this important? Because traditional hazard analysis is based on decomposition and therefore on these assumptions. The basic approach involved is to divide the system into *components*, assume that accidents are caused by component failure, calculate the probability of failure of each component separately, and later combine the analysis results (based on assumptions about the types of interactions among components that can occur) into a system reliability figure, which is assumed to be a measure of safety or risk. Examples are Failure Modes and Effects Analysis (FMEA) and Failure Modes and Effects Criticality Analysis. The latter considers only failures that can lead to a critical loss rather than all failures.

Alternatively, chains of directly related physical or logical (functional) failure *events* that can lead to a loss are identified and the probability of each is combined into a probability of the event chain occurring. The following is a chain of events example for a tank explosion.



In this example, moisture gets into the tank, causing corrosion. The corrosion leads to weakened metal, which, along with a particular operating pressure, can cause the tank to rupture. As a result, fragments are projected and equipment is damaged or personnel are injured.

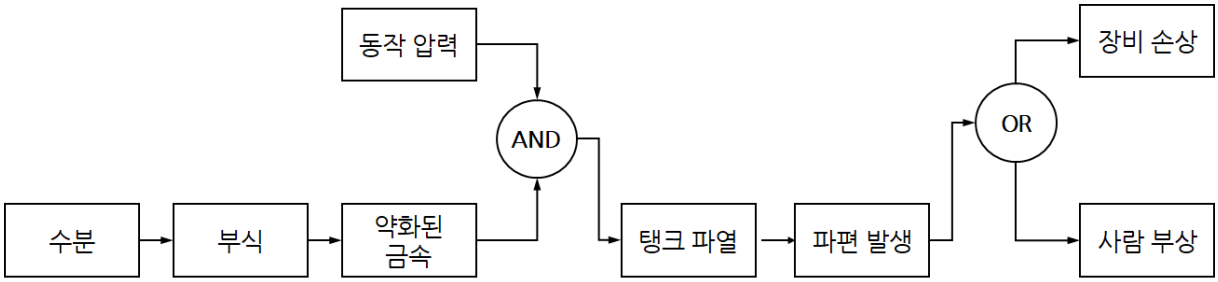
If accidents are assumed to be caused by failure events, then it makes sense that the approach to preventing accidents is to eliminate the events or to put barriers between the events so that one failure does not cause another event down the chain. The tank rupture model of events is annotated below with the typical types of accident prevention design techniques used, including redundancy, barriers, high component integrity and overdesign, fail safe design, and, for humans, the use of operational procedures, checklists, and training. These design features are used to reduce the probability of the failure events occurring or of them propagating.

- 각 컴포넌트 또는 서브시스템은 독립적으로 동작한다. 이벤트가 모델링되는 경우, 이벤트는 바로 앞뒤의 이벤트를 제외하고는 독립적이다.
- 컴포넌트를 별도로(단독으로) 분석할 때의 컴포넌트 동작이 컴포넌트가 전체에서 부분으로서 동작할 때와 동일하다.
- 컴포넌트와 이벤트는 피드백 루프 및 기타 간접적인 상호작용에 영향을 받지 않는다.
- 컴포넌트 간 또는 이벤트 간의 상호작용은 쌍(pairwise) 단위로 관찰하여 복합 값(composite value)으로 결합할 수 있다.

이러한 가정이 성립하지 않는다면 개별 분석 결과를 단순히 결합하는 것은 시스템을 정확히 반영하지 못할 것이다.

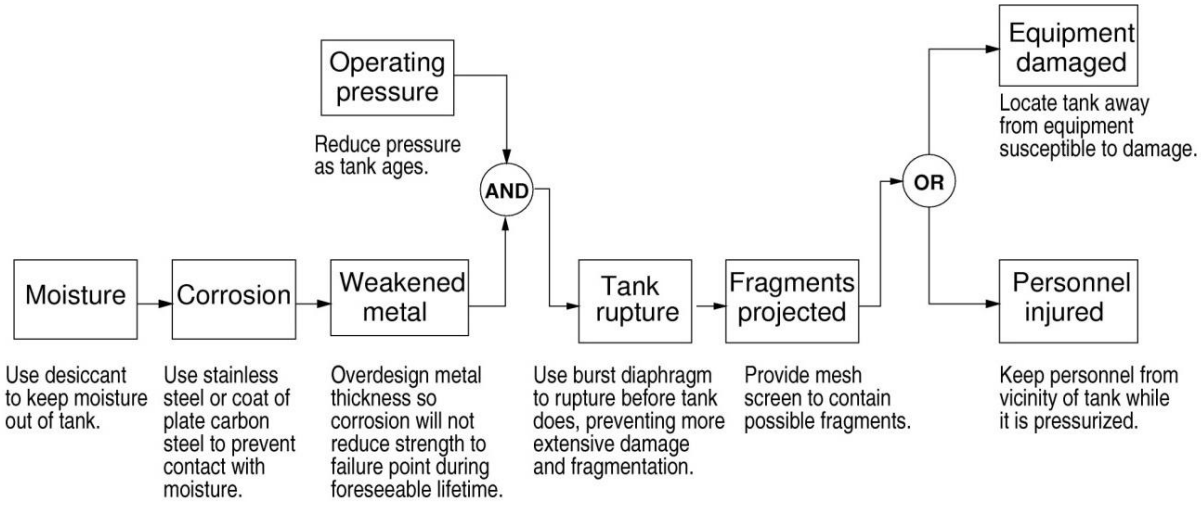
왜 이러한 것들이 중요할까? 전통적인 위험 분석은 분해에 기반을 두고 있으므로 이러한 가정에 기반하기 때문이다. 기본적인 접근법은 시스템을 *컴포넌트*로 나누고, 사고(accident)는 컴포넌트의 실패로 인해 발생한다는 가정 하에 각 컴포넌트의 실패 확률을 개별적으로 계산한 다음, 시스템 수준의 신뢰도 수치를 계산할 때 개별 분석 결과를 결합한다(컴포넌트 간 발생할 수 있는 상호작용 유형에 대한 가정에 기반함). FMEA 기법과 FMECA 기법이 이러한 접근법의 예시이다. FMECA의 경우 모든 실패를 고려하지 않고 심각한 손실로 이어질 수 있는 실패만을 고려한다.

다른 대안으로, 손실로 이어질 수 있는 직접 연관된 물리적 또는 논리적(기능적) 결함 이벤트 체인을 식별하고, 이벤트 체인의 발생 확률을 각각의 발생 확률의 합으로 계산한다. 다음은 탱크 폭발에 대한 이벤트 체인의 예이다.



이 예에서는 수분이 탱크로 들어가 부식을 일으킨다. 그 부식은 금속을 약화시키고 특정 압력에 도달하면 탱크 파열을 야기할 수 있다. 결과적으로 파편으로 인해 장비가 손상되거나 사람이 다치게 된다.

사고가 이벤트에 의해 야기되었다고 가정하면, 사고를 예방하기 위해서는 이벤트를 제거하거나 이벤트들 사이에 장벽을 두어 하나의 실패로 인해 다른 이벤트가 연쇄적으로 발생하지 않도록 하는 것이 타당한 방법이다. 탱크 파열 이벤트 모델의 하단에 다중화, 장벽, 컴포넌트의 높은 무결성과 오버디자인(overdesign), 페일세이프(fail-safe) 설계, 사람의 측면에서의 운영 절차, 체크리스트, 훈련과 같은 사고 예방 설계 기법의 전형적인 유형들을 표시하였다. 이러한 설계 특성들은 실패 이벤트 발생 가능성 및 전파 확률을 줄이기 위해 사용된다.



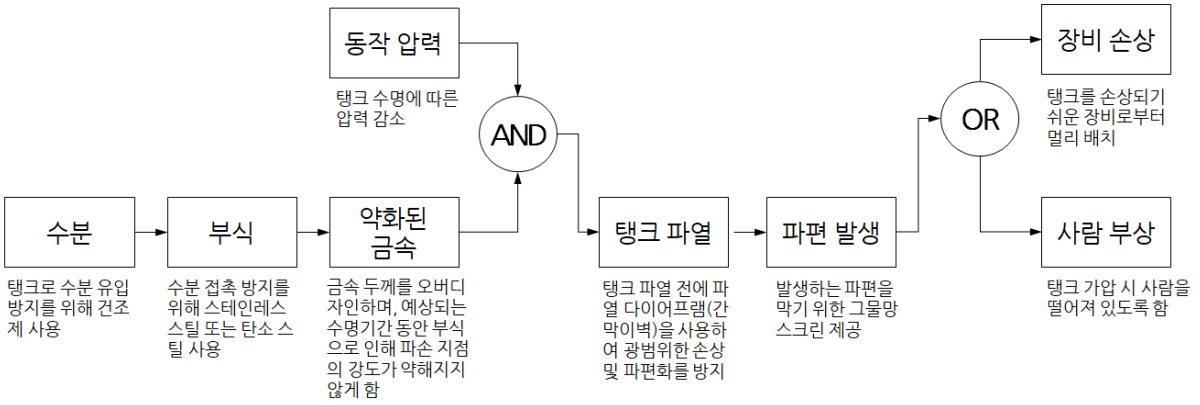
The failure events, of course, must be assumed to be stochastic[2] for probabilities or likelihood to be determined. Unfortunately, software and humans do not satisfy this assumption. This second, event-chain approach to hazard analysis is the basis for fault tree analysis (FTA), event tree analysis (ETA), fault hazard analysis (FHA), and hazard and operability analysis (HAZOP, which uses a deviation rather than a failure as the event or condition to be considered).

The assumptions underlying the decompositional approach to traditional hazard analysis are true (or close enough) for the type of electromechanical systems we used to build, and they are still true for certain properties in our new high-tech, software-intensive systems. However, the greatly increased complexity in our systems today (which is made possible primarily by the use of software) is creating systems where the approach is no longer as effective. It is much more difficult today to anticipate, understand, plan, and guard against all potential system behavior before operational use of our systems. Complexity is creating "unknowns" that cannot be identified by breaking the system behavior into chains of events. In addition, complexity is leading to important system properties (such as safety) not being related to the behavior of individual system components but rather to the interactions among the components. Accidents can occur due to unsafe interactions among components that have not failed and, in fact, satisfy their requirements.

Some examples may be helpful.

Some Navy aircraft were ferrying missiles from one point to another. One pilot executed a planned test by aiming at the aircraft in front (as he had been told to do) and firing a dummy missile. Apparently nobody knew that the "smart" software was designed to substitute a different missile if the one that was commanded to be fired was not in a good position. In this case, there was an antenna between the dummy missile and the target, so the software decided to fire a live missile located in a different (better) position instead. What aircraft component(s) failed here?

[2] Something is stochastic if it can be described by a probability distribution or pattern that may be analyzed statistically to understand average behavior or an expected range of behavior but may not be predicted precisely



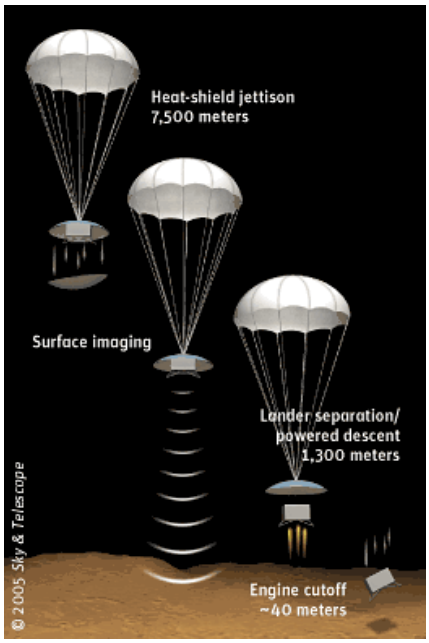
물론 실패 이벤트는 발생확률(probabilities) 또는 발생 가능성(likelihood)에 의해 결정되며 확률론적[2]임이 가정되어야 한다. 불행하게도 소프트웨어와 사람에 대해서는 이 가정이 만족되지 않는다. 이러한 위험 분석을 위한 두 번째 이벤트 체인 접근법은 FTA(fault tree analysis), ETA(event tree analysis), FHA(fault hazard analysis), 그리고, HAZOP(hazard and operability analysis, 고려되어야 할 이벤트 또는 조건으로서 실패보다는 일탈(deviation)을 사용하는 방법)의 기반이 되는 접근법이다.

기존 위험 분석에서 분해 접근법의 기본 가정은 이미 구축된 전기 기계 시스템 유형에 대해서는 성립하며(또는 충분히 그러함), 또한 새로운 첨단 기술, 소프트웨어 집약적 시스템의 특정 속성에 대해서도 여전히 적합하다. 그러나 오늘날 시스템 복잡도의 급격한 증가로(주로 소프트웨어 사용에 의함) 이러한 접근법은 더 이상 효과적이지 않다. 최근에는 시스템을 운영하기 전에 모든 잠재된 시스템 동작을 예측, 이해, 계획 및 보호하는 것이 훨씬 더 어렵게 되었다. 복잡도는 시스템 동작을 이벤트 체인으로 분할하는 경우, 식별할 수 없는 "알 수 없는 것(unknown)"을 발생시킨다. 또한 복잡도는 개별 시스템 컴포넌트의 동작보다는 컴포넌트 간의 상호작용에 관련된 중요한 시스템 속성(안전 등)으로 이어진다. 컴포넌트가 실패하지 않았고 실제로 요구 사항을 만족하는 경우에도 컴포넌트 간의 안전하지 않은 상호작용에 인해 사고는 발생할 수 있다.

위와 관련된 사례를 몇 가지 살펴보자.

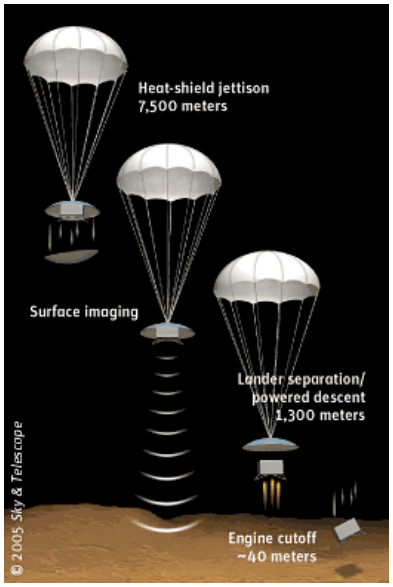
어떤 해군 항공기가 한 지점에서 다른 지점으로 미사일을 운반하고 있었다. 한 조종사는 (지시대로) 항공기 앞을 조준하고 더미 미사일을 발사하는 계획된 테스트를 실시했다. 하지만 발사되는 미사일의 위치가 적절하지 않은 경우, 발사가 명령된 미사일을 다른 미사일로 대체하도록 하는 "스마트" 소프트웨어가 설계되었다는 사실을 아무도 몰랐다. 이 상황에서 더미 미사일과 표적 사이에 안테나가 있었기 때문에 소프트웨어는 다른(더 나은) 위치에 있는 실제 미사일을 발사하기로 결정하였다. 여기서 항공기의 어떤 컴포넌트가 실패한 것인가?

[2] 확률론적(stochastic)이란 확률분포(probability distribution)로 설명되거나, 정확하지는 않지만 평균적인 행동이나 예상되는 행동반경을 이해할 수 있도록 통계적으로 분석되는 패턴으로 설명되는 것을 말함



This loss involved the Mars Polar Lander. It is necessary to slow the spacecraft down to land safely. Ways to do this include using the Martian atmosphere, a parachute and descent engines (controlled by software). As soon as the spacecraft lands, the software must immediately shut down the descent engines to avoid damage to the spacecraft. Some very sensitive sensors on the landing legs provide this information. But it turned out that noise (false sensor signals) are generated when the legs are deployed. This expected behavior was not in the software requirements. Perhaps it was not included because the software was not supposed to be operating at this time, but the software engineers decided to start early to even out the load on the processor. The software thought the spacecraft had landed and shut down the descent engines while the spacecraft was still 40 meters about the planet surface. Which spacecraft components failed here?

It is dangerous for an aircraft's thrust reversers (which are used to slow the aircraft after it has touched down) to be activated when the aircraft is still in the air. Protection is designed into the software to prevent a human pilot from erroneously activating the thrust reversers when the aircraft is not on the ground. Without going into the details, some of the clues for the software to determine the plane has landed are weight on wheels and wheel spinning rate, which for a variety of reasons did not hold in this case. For example, the runway was very wet and the wheels hydroplaned. As a result, the pilots could not activate the thrust reversers and the aircraft ran off the end of the runway into a small hill. What aircraft components failed here?



이 손실(loss)은 화성 극지 착륙선에 대한 것이다. 이 착륙선이 안전하게 착륙하기 위해서는 우주선의 속도를 낮추어야 하는데, 이를 위한 방법으로 화성의 대기(atmosphere), 낙하산, 하강 엔진(소프트웨어에 의해 제어됨)을 사용한다. 우주선이 착륙하자마자 우주선의 손상을 방지하기 위해 소프트웨어는 즉시 하강 엔진을 종료해야 한다. 착륙용 다리(landing leg)에 있는 매우 민감한 센서가 이러한 정보를 제공한다. 그러나 다리가 펼쳐질 때 노이즈(잘못된 센서 신호)가 발생된다는 것이 밝혀졌다. 하지만 이런 예상된 동작이 소프트웨어 요구사항에는 명시되어 있지 않았다. 아마도 다리가 펼쳐지고 이 시점에 소프트웨어가 작동하지 않는 것으로 되어 있었기 때문에 요구사항에 포함되지 않았겠지만, 소프트웨어 엔지니어는 프로세서 부하를 균등하게 만들기 위해 소프트웨어를 일찍 작동시키기로 결정했다. 소프트웨어는 우주선이 행성 표면에서 40미터의 거리에 있음에도 불구하고, 우주선이 착륙하였다고 판단하여 하강 엔진을 종료하였다. 이 경우 우주선의 어떤 컴포넌트가 실패한 것인가?

항공기가 상공에 있을 때 역방향 추진 장치(항공기 터치다운 후 항공기를 감속시키는 데 사용)의 활성화는 위험하다. 항공기가 지상에 있지 않을 때 조종사가 역방향 추진력을 잘못 활성화하는 것을 방지하기 위해 소프트웨어에 보호 기능이 설계되었다. 세부 사항을 제외하고, 소프트웨어가 항공기가 착륙했는지를 판단하는 데 필요한 몇 가지 단서들로는 착륙감지(weight on wheel)나 바퀴회전률(wheel spinning rate)이 있는데, 이 경우에는 여러 가지 이유로 착륙의 정황이 감지되지 않았다. 그 예로, 활주로가 많이 젖어 있었고 바퀴에 수막현상(Hydroplaned)이 발생하였다. 그 결과, 조종사는 역방향 추진 장치를 활성화시킬 수 없었고, 결국 항공기는 활주로를 넘어가 활주로 끝의 작은 언덕에 충돌하였다. 이 경우 항공기의 어떤 컴포넌트가 실패한 것인가?

This accident involved a Tupelov aircraft landing in Moscow in 2012. A soft touchdown made runway contact a little later than usual. With the crosswind at the time, the weight-on-wheels switches did not activate and the thrust reverse system would not deploy. The pilots thought that the thrust reversers would deploy as they always do. So with limited runway space to stop the aircraft, they quickly engaged high engine power to stop quicker. Instead, this accelerated the aircraft forward, eventually colliding with a highway embankment. What aircraft components failed here?

The coastal area of Washington state has lots of water and islands. Car ferries are necessary to move people around. One day, the ferry system was brought to its knees when rental cars could not be driven off the ferries after arriving in port. It turned out that a local rental car company had installed a security device to prevent theft by disabling cars if the car moved when the engine was stopped. When the ferries moved and the cars were not running, the cars all disabled themselves and the ferry system was brought to a standstill until enough tow trucks could be found to tow the cars off the ferries. What car component(s) failed here?

The manufacturer of the lithium-ion batteries involved in this incident determined that battery cell venting would occur once in 10 million flight hours, but two batteries malfunctioned in just two weeks in 2013. While there were many factors involved, one of the factors makes an interesting example here. In the event of a battery malfunction, an environmental control system was designed to remove smoke through cooling duct fans by actuating certain valves. However, the unit providing power to the environmental control system simultaneously shut down due to the battery malfunction. As a result, the valves could not be actuated and smoke being generated by the APU battery could not be effectively directed outside the passenger cabin and battery compartment.

이 사고는 2012년 모스크바에 착륙한 Tupelov 항공기 사고이다. 소프트 터치다운(soft touchdown)으로 인해 항공기가 평상시보다 조금 더 늦게 활주로에 닿았다. 당시 측풍(crosswind)이 불어 착륙감지(weight-on-wheel) 스위치가 활성화되지 않았으며 역방향 추진 시스템이 동작하지 않았다. 하지만 조종사는 늘 그랬듯이 역방향 추진 장치가 동작하는 것으로 생각하였다. 항공기가 멈추기 위한 활주로 공간이 부족했기 때문에 조종사는 항공기를 빨리 멈추기 위해 높은 엔진 출력을 사용하였으나, 이것이 오히려 항공기를 앞쪽으로 가속시켰으며, 결국 고속도로 제방과 충돌하였다. 이 경우 어떤 컴포넌트가 실패한 것일까?

워싱턴 주 해안 지역에는 많은 물과 섬이 있다. 따라서 사람들을 수송하기 위해 자동차 페리가 필요하다. 어느 날, 페리가 항구에 도착한 후 특정 렌터카를 페리에서 꺼낼 수 없었고, 이 때문에 페리시스템이 마비되었다. 지역 렌터카 회사에서 자동차 도난 방지를 위해 자동차 엔진이 꺼진 상태에서 자동차가 움직이면 자동차를 동작하지 않게 하는 보안 장치를 설치한 것이 원인이었다. 페리가 움직였으나 자동차는 운영 상태가 아니었으므로, 자동차는 스스로 불능 상태가 되었고, 페리 시스템은 견인 트럭이 올 때까지 멈춰있었다. 여기서 어떤 자동차 컴포넌트가 실패한 것일까?

이 사건과 관련된 리튬-이온 배터리 제조업체에서는 배터리 셀 폭발이 1천만 비행 시간당 한번 발생한다고 결정하였지만, 2013년에는 2주 만에 2개의 배터리가 오작동하였다. 많은 요인이 관련되어 있지만 그 중 한 가지 요인이 흥미로운 예이다. 배터리가 오작동하는 경우, 환경 제어 시스템은 특정 밸브를 작동시켜 냉각 덕트 팬을 통해 연기를 내보내도록 설계되어 있었다. 그러나, 환경 제어 시스템에 전원을 공급하는 장치가 배터리 오작동으로 인해 동시에 종료되었다. 그 결과, 밸브는 작동될 수 없었으며 APU(보조동력장치) 배터리에 의해 생성된 연기는 객실 및 배터리 실 외부로 효과적으로 배출될 수 없었다.

A more complex example in aircraft avionics systems can be found in Appendix A. In addition, Dr. Ioana Koglbauer at the University of Graz (Austria) provided an example that emphasizes the usefulness of a systems approach. A certified fixed-base flight simulator at their university has the same production cockpit components (e.g. rudder-pedals) as the certified aircraft, but the pedals in the simulator often break and must be replaced. If you try to understand why and solve the problem at the component level, you could blame and punish the pilot who produced the damage, train the pilot on how to handle the simulator, or invest in improving the pedals. If you look at interactions among components (people, hardware, software), and try to understand why the pilots use the pedals differently in the simulator than in a real aircraft , you come to a different problem definition and solution. It turns out that when the pilots brake in the simulator, the simulator software is not programmed to show the typical small pitch down movement when the aircraft comes to a stop. This movement can also be felt when parking a car. In the absence of this feedback in the aircraft simulator, the pilots apply the brakes longer and stronger than needed. For this reason, the pedals break in the simulator, but not in the real aircraft. The best solution is to improve the software and give pilots the feedback they need.

In all of these cases (and in hundreds of others), the problems did not stem from individual component failures but from flaws in the system engineering process that resulted in flawed system designs. Decompositional analysis cannot identify these causes of accidents including human error, software requirements errors, and system design flaws. We need something different. That different theoretical foundation, which underlies STPA, is called System Theory.

System Theory

As stated above, system theory as used in engineering was created after World War II to deal with the increased complexity of the systems being built after the war.[3] It was also created for biology in order to successfully understand the complexity of biological systems. In these systems, separation and analysis of separate, interacting components (subsystems) distorts the results for the system as a whole because the component behaviors are coupled in non-obvious ways. The first engineering uses of these new ideas were in the missile and early warning systems of the 1950s and 1960s.

Some unique aspects of System Theory are:

- The system is treated as a whole, not as the sum of its parts. You have probably heard the common statement: “the whole is more than the sum of its parts.”
- A primary concern is *emergent properties*, which are properties that are not in the summation of the individual components but “emerge” when the components interact. Emergent properties can only be treated adequately by taking into account all their technical and social aspects.
- Emergent properties arise from relationships among the parts of the system, that is, by how they interact and fit together.

[3] If you are interested in learning more, we recommend Peter Checkland, *System Thinking, System Practice*, John Wiley & Sons (1981), Gerald Weinberg, *An Introduction to General Systems Thinking*, John Wiley & Sons (1975), and perhaps for historical interest, W. Ross Ashby, *An Introduction to Cybernetics*, Chapman and Hall (1956).

항공 전자 공학 시스템의 보다 복잡한 예는 부록 A에 나와 있다. 또한 (오스트리아) 그라즈(Graz) 대학교의 코글바우어(Ioana Koglbauer) 박사는 시스템 접근법(System approach)의 유용성을 강조한 사례도 제시하였다. Graz 대학에서 인증된 고정 기지 비행 시뮬레이터는 인증된 항공기와 동일한 제품의 조종실 컴포넌트(예: 방향타 페달)를 사용하지만, 시뮬레이터 페달이 자주 고장나서 교체가 필요했다. 만약 당신이 컴포넌트 수준에서 원인을 이해하고 문제를 해결한다면 손상을 일으킨 조종사를 비난하거나 처벌하고, 조종사에게 시뮬레이터를 다루는 방법을 교육시키거나 페달을 개선하는 데 투자할 수 있다. 만약 당신이 컴포넌트(사람, 하드웨어, 소프트웨어)간의 상호작용을 살펴보고, 조종사가 실제 항공기에서와 시뮬레이터에서 페달을 다르게 사용하는 이유를 이해한다면 문제의 새로운 정의와 해결 방법을 발견할 수 있다. 이 경우, 일반적으로 조종사가 항공기에서 제동을 할 때 표시되는 스몰 피치 다운 동작이 시뮬레이터 소프트웨어에는 프로그래밍 되지 않은 것으로 밝혀졌다. 이 동작은 자동차의 주차 시에도 느낄 수 있다. 항공기 시뮬레이터에서 이 피드백이 없으면 조종사는 제동을 필요 이상으로 더 길고 강하게 건다. 이러한 이유로 시뮬레이터에서는 페달이 더 자주 파손되지만 실제 항공기에서는 그렇지 않다. 이를 위한 가장 좋은 해결책은 소프트웨어를 개선하여 조종사가 필요로 하는 피드백을 제공하는 것이다.

이러한 모든 경우(그리고 수 백 가지의 다른 경우)에서 문제는 개별 컴포넌트의 오류로 인한 것이 아니라 시스템 설계 결함을 야기하는 시스템 엔지니어링 프로세스의 결함 때문이라고 할 수 있다. 분해 분석(decompositional anlaysis)은 인적 오류, 소프트웨어 요구사항 오류 및 시스템 설계 결함을 포함하는 이러한 사고의 원인을 식별할 수 없다. 그러므로 우리는 다른 이론적 기반이 필요한데, 이러한 이론적 기반은 STPA의 근간을 이루는 것으로 시스템 이론(System theory)이라고 한다.

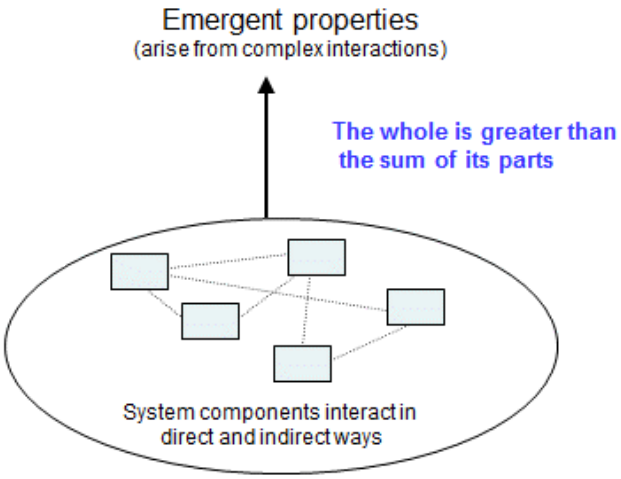
시스템 이론(System Theory)

위에서 언급했듯이 엔지니어링에 사용된 시스템 이론은 제2차 세계 대전 이후에 만들어졌으며 전쟁 이후에 생겨난 시스템들의 복잡도를 다루기 위해 사용되었다.[3] 또한 생물학적 시스템의 복잡도를 성공적으로 이해하기 위해 만들어졌다. 이러한 시스템에서는 컴포넌트의 동작이 명확하지 않은 방식으로 결합되기 때문에 상호작용하는 컴포넌트(서브시스템)를 분리하여 분석하면 시스템 전체의 결과가 왜곡된다. 이러한 새로운 아이디어가 최초로 공학적으로 사용된 것은 1950년대와 1960년대의 미사일 및 조기 경보 시스템이었다.

시스템 이론의 몇 가지 특징은 다음과 같다.

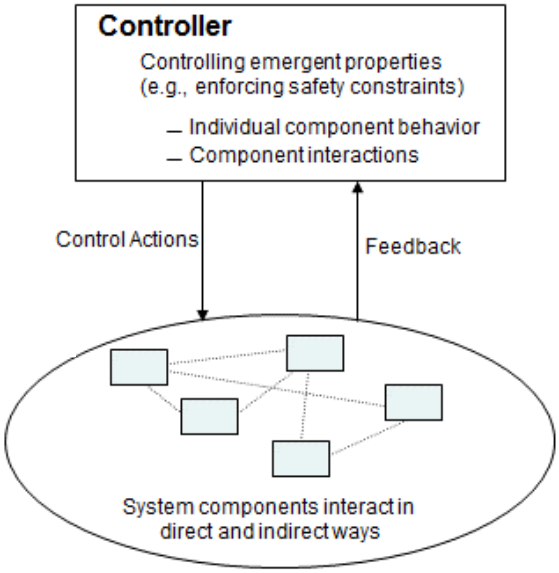
- 시스템을 부분들의 합으로서가 아닌 전체로서 취급한다. 당신은 아마 “전체가 그 부분의 합보다 더 크다”라는 통설을 알 것이다.
- 주요 관심사는 개별 컴포넌트의 합이 아니고 컴포넌트가 상호작용할 때 “발현(emerge)”하는 특성인 발현 속성이다. 발현 속성은 모든 기술적 측면과 사회적 측면을 고려하여야만 적절하게 처리될 수 있다.
- 발현 속성은 시스템의 각 부분간의 관계, 즉, 어떻게 상호작용하고 조화하는지에 따라 발생한다.

[3] 더 많은 것을 알고 싶다면 Peter Checkland, Systems Thinking, System Practice, JohnWilley&Sons(1981), Gerald Weinberg, An Introduction to General Systems Thinking, John Wiley & Sons(1975) and perhaps for historical interest, W. Ross Ashby, An Introduction to Cybernetics, Chapman and Hall(1956) 등을 추천한다.

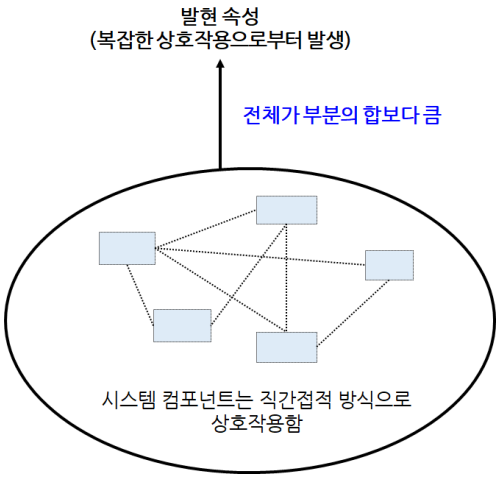


If emergent properties arise from individual component behavior and from the interactions among components, then it makes sense that controlling emergent properties, such as safety, security, maintainability, and operability, requires controlling the behavior of the individual components and the interactions among the components. We can add a controller to the figure. The controller provides control actions on the system and gets feedback to determine the impact of the control actions. If this looks like a standard feedback control loop to you, you are absolutely correct. That is what it is.

The controller enforces constraints on the behavior of the system. Example safety constraints might be that aircraft or automobiles must remain a minimum distance apart, pressure in deep water wells must be kept below a safe level, aircraft must maintain sufficient lift to remain airborne unless landing, accidental detonation of weapons must be prevented, and toxic substances must never be released from a plant.

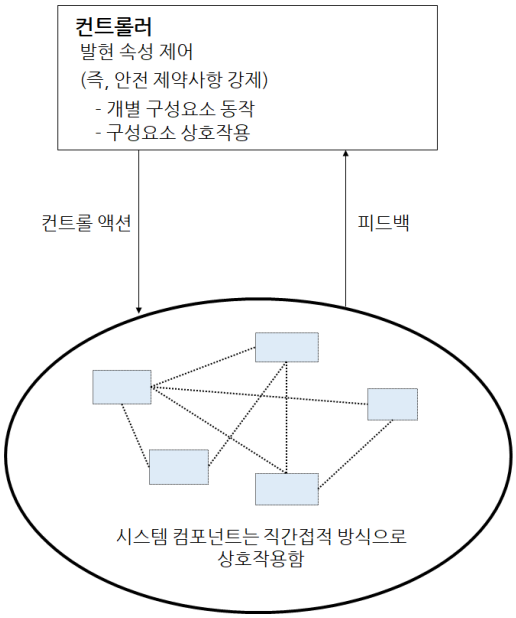


To explain the model above using a simple example, consider the national or international airspace. If each airline were allowed to optimize its schedule by flying at any time and using any route, chaos might result if everyone tried to land at Chicago, New York, or Heathrow at 5 pm. To avoid such conflicts, system-level air traffic control (ATC) is introduced that controls two emergent properties: throughput and safety.



발현 속성이 개별 컴포넌트 동작과 컴포넌트 간의 상호작용에서 발생한다면 안전(safety), 보안(security), 유지보수가능성(maintainability), 운영가능성(operability)과 같은 발현 속성을 제어하기 위해서는 개별 컴포넌트의 동작과 컴포넌트 간의 상호작용을 제어해야 한다. 우리는 그림에 컨트롤러를 추가할 수 있다. 컨트롤러는 시스템에 컨트롤 액션을 제공하고 피드백을 받아 컨트롤 액션에 따른 영향을 판별한다. 이것이 표준 피드백 컨트롤 루프처럼 보인다면 절대적으로 맞게 본 것이다.

컨트롤러는 시스템의 동작에 안전 제약사항을 가한다. 안전 제약사항(Safety Contraints)의 예로는 '항공기 또는 자동차가 최소한의 거리를 유지해야 한다', '깊은 우물의 압력은 안전한 수준 이하로 유지해야 한다', '항공기는 착륙하지 않는 한 충분한 양력을 유지해야 한다', '무기의 우발적인 폭발은 막아야 한다', '독성 물질은 공장에서 결코 유출되면 안 된다' 등이 있다.



국내 또는 국제 공역(airspace)을 고려한 간단한 예제를 사용하여 위의 모델을 설명하고자 한다. 만약, 각 항공사가 스케줄 최적화를 위해 임의의 시간과 경로를 사용하는 것이 허용된다면, 모든 비행기가 시카고, 뉴욕 또는 히드로 공항에 오후 5시에 착륙하려 하여 혼란이 초래될 수 있다. 이러한 충돌을 피하기 위해 시스템 수준의 ATC(Air Traffic Control, 항공교통관제)가 도입되었는데, ATC는 처리량

ATC is responsible for optimizing the overall throughput in the system (which may not optimize the route for every aircraft) and for maintaining adequate separation among the aircraft.

This model includes everything we now do in safety engineering. Control is interpreted broadly. For example, component failures and unsafe interactions may be controlled through design, such as using redundancy, interlocks, barriers, or fail-safe design. Safety may also be controlled through process, such as development processes, manufacturing processes and procedures, maintenance processes, and general system operating processes. Finally, safety may be controlled using social controls including government regulation, culture, insurance, law and the courts, or individual self-interest. Human behavior can be partially controlled through the design of the societal or organizational incentive structure.

What is STAMP?

STAMP[4] (System-Theoretic Accident Model and Processes) is the name of the new accident causality model based on systems theory,[5] as described in the previous section, which provides the theoretical foundation for STPA. It expands the traditional model of causality beyond a chain of directly-related failure events or component failures to include more complex processes and unsafe interactions among system components, and it underlies STPA and other tools.

In STAMP, safety is treated as a dynamic control problem rather than a failure prevention problem. No causes are omitted from the STAMP model, but more are included and the emphasis changes from preventing failures to enforcing constraints on system behavior.

Some advantages of using STAMP are that:

- It works on very complex systems because it works top-down rather than bottom up.
- It includes software, humans, organizations, safety culture, etc. as causal factors in accidents and other types of losses without having to treat them differently or separately.
- It allows creating more powerful tools, such as STPA, accident analysis (CAST), identification and management of leading indicators of increasing risk, organizational risk analysis, etc.

Because STAMP applies to any emergent property, STPA can be used for any system property, including cybersecurity.

STAMP is not an analysis method. Instead it is a model or set of assumptions about how accidents occur. STAMP is an alternative to the chain-of-failure-events (or dominos or Swiss cheese slices, all of which are essentially equivalent) that underlies the traditional safety analysis techniques (such as Fault Tree Analysis, Event Tree Analysis, HAZOP, FMECA, and HFACS). Just as the traditional analysis methods are constructed on the assumptions about why accidents occur in a chain-of-failure-events model, new analysis methods can be constructed using STAMP as a basis. Note that because the chain-of-failure events model is a subset of STAMP, tools built on STAMP can include as a subset all the results derived using the older safety analysis techniques.

The two most widely used STAMP-based tools today are STPA (System Theoretic Process Analysis) and CAST (Causal Analysis based on Systems Theory). STPA is a proactive analysis method that analyzes the potential cause of accidents during development so that hazards can be eliminated or controlled.

[4] Nancy G. Leveson, *Engineering a Safer World*, MIT Press (2012)

[5] The basic system theory that underlies STAMP should not be confused with Complexity Theory, which later grew out of basic system theory to explain the adaptive, non-linear behavior common in nature and social organizations. For engineering, even the engineering or design of organizations, basic system theory suffices. For more, see Appendix F.

(throughput)과 안전(safety)이라는 두 가지 발현 속성을 제어한다. ATC는 시스템의 전체 처리량을 최적화(모든 항공기의 경로를 최적화하지 못할 수도 있음)하고 항공기 간의 적절한 간격을 유지하는 역할을 한다.

이 모델에는 현재 안전 공학에서 수행하는 모든 작업을 포함한다. 컨트롤(제어)은 광범위하게 해석된다. 예를 들어, 다중화, 인터락(interlock, 연동장치), 배리어(barrier) 또는 페일세이프(fail-safe)와 같은 설계를 통해 컴포넌트 실패와 안전하지 않은 상호작용을 컨트롤 할 수 있다. 또한 안전은 개발 프로세스, 제조 프로세스 및 절차, 유지보수 프로세스 및 일반적인 시스템 운영 프로세스와 같은 프로세스를 통해 컨트롤 될 수도 있다. 마지막으로 안전은 정부 규제, 문화, 보험, 법 및 법원 또는 개인의 사적 이익을 포함한 사회적 제어를 사용하여 컨트롤 될 수 있다. 사람의 행동은 사회적 또는 조직적 인센티브 구조의 설계를 통해 부분적으로 컨트롤 될 수 있다.

STAMP란?

STAMP[4](System-Theoretic Accident Model and Processes)는 시스템 이론[5]에 기초한 새로운 사고(accident) 인과관계 모델의 이름으로 앞 섹션에서 설명한대로 STPA에 대한 이론적 토대를 제공한다. STAMP는 전통적인 인과관계의 모델을 직접 관련된 실패 이벤트 체인이나 컴포넌트 실패로 확장하여 시스템 컴포넌트 간의 복잡한 프로세스 및 안전하지 않은 상호작용을 포함시키며, 이것은 STPA 및 다른 도구(tool)의 근간이 되고 있다.

STAMP에서 안전은 실패(failure) 예방 문제가 아닌 동적 제어 문제로 취급된다. STAMP 모델은 실패 방지에서 시스템 동작에 제약을 가하는 것으로 초점이 바뀌므로, 어떤 원인도 누락되지 않고 더 많은 원인들이 포함되어 있다.

STAMP를 사용하면 다음과 같은 이점이 있다.

- 매우 복잡한 시스템에 적용 가능하다. 왜냐하면 상향식보다 하향식이기 때문이다.
- STAMP는 사고나 다른 유형의 손실에서의 원인 요소가 되는 소프트웨어, 사람, 조직, 안전 문화 등을 모두 포함하고 있어, 이를 다른 방법으로 또는 별도로 다루지 않아도 된다.
- STPA, 사고 분석(CAST), 증가하는 리스크 선행지표 식별 및 관리, 조직 리스크 분석 등과 같은 보다 강력한 도구를 만들 수 있다.

STAMP는 모든 발현 속성에 적용되므로 STPA는 사이버 보안을 포함한 모든 시스템 속성에 사용할 수 있다. STAMP는 분석 방법이 아니고 사고가 어떻게 발생하는지에 대한 모델 또는 가정들의 집합(set of assumptions)이다. STAMP는 기존의 안전 분석 기법(Fault Tree Analysis, Event Tree Analysis, HAZOP, FMECA, HFACS)의 근간을 이루는 chain-of-failure-events(또는 도미노 모델, 스위스 치즈 슬라이스 모델, 모두 본질적으로 동일)의 대안이 될 수 있다. 기존의 분석 방법이 chain-of-failure-events 모델에서 사고가 발생하는 이유에 대한 가정을 기반으로 구축된 것과 같이, 새로운 분석 방법은 STAMP를 기반으로 구축할 수 있다. chain-of-failure-events 모델은 STAMP의 일부분이기 때문에 STAMP에 기반한 도구는 기존의 안전 분석 기법을 사용하여 도출하는 모든 결과를 포

[4] Nancy G. Leveson, Engineering a Safer World, MIT Press (2012)

[5] STAMP의 기초가 되는 기본 시스템 이론은 복잡도 이론(Complexity Theory)과 혼동해서는 안된다. 복잡도 이론은 기본 시스템 이론에서 추후 파생된 것으로 자연 및 사회 조직에서 흔히 볼 수 있는 적응 행동(adaptive behavior) 및 비선형적 행동(non-linear behavior)을 설명하기 위해 사용된다. 엔지니어링, 조직의 설계 또는 엔지니어링도 기본 시스템 이론으로 설명하기에 충분하다. 자세한 내용은 부록 F를 참조한다.

CAST is a retroactive analysis method that examines an accident/incident that has occurred and identifies the causal factors that were involved. This handbook concentrates on the use of STPA. A future, similar handbook is planned for CAST.

Organization of this Handbook

Chapter 2 of this handbook explains the basics of STPA and how to do it. An aerospace example is used, but examples from other industries are included in the appendices.

Chapters 3 through 7 describe how to use STPA in various common types of engineering activities. Because STPA can be applied to any emergent (system) property, Chapter 5 applies STPA to something other than safety, in this case, system engineering effectiveness. While most of the applications of STPA in industry so far have involved safety, companies are now starting to use it for other system properties such as quality, security, and production engineering.

Finally, Chapter 8 provides some advice about how to integrate STPA into a large organization. The appendices provide additional examples, a guide for analyzing an organizational safety management system, some additional explanation (including an example) about why decomposition does not work for today's software-intensive, complex systems, and an introduction to some basic concepts for non-engineers.

함할 수 있다.

오늘날 가장 널리 사용되는 STAMP 기반의 2가지 도구는 STPA(System Theoretic Process Analysis)와 CAST(Causal Analysis based on System Theory)이다. STPA는 개발 단계에서 사고의 잠재적 원인을 분석하여 위험을 제거하거나 제어하기 위한 사전적인(proactive) 분석 방법이다. CAST는 발생한 사고/사건을 조사하고 관련된 요인을 식별하는 사후적인(retroactive) 분석 방법이다. 이 핸드북은 STPA에 중점을 두며, 향후 CAST에 대한 유사한 핸드북이 제작될 예정이다.

핸드북 구성

이 핸드북의 2장에서는 STPA의 기본사항과 수행 방법에 대해 설명한다. 2장에서는 항공 우주의 예를 소개하였고, 부록에 다른 산업의 사례를 포함하였다.

3장부터 7장까지는 다양한 일반적인 유형의 엔지니어링 활동에서 STPA를 사용하는 방법을 설명한다. STPA는 모든 발현(시스템) 속성에 적용될 수 있으므로 5장에서는 STPA를 안전 이외의 다른 것에 적용하는 것, 즉, 조직이나 사회에의 시스템 엔지니어링 효과에 대해 언급한다. 지금까지 산업계에서 STPA를 적용한 대상은 대부분 안전과 관련된 것이었지만 최근에는 품질, 보안 및 제품 공학과 같은 다른 시스템 속성에도 사용하기 시작했다.

마지막으로 8장에서는 STPA를 대규모 조직에 통합하는 방법에 대해 조언한다. 부록에서는 추가적인 예시, 조직의 안전 관리 시스템 분석 지침을 제공하며, 왜 최근의 소프트웨어 집약적이고 복잡한 시스템에서 분해적(decomposition) 접근법이 동작하지 않는지에 대한 추가적인 설명(예시포함), 엔지니어가 아닌 이들을 위한 기본 개념을 소개한다.

Chapter 2: How to Do a Basic STPA Analysis

John Thomas

STPA Method Overview

The steps in basic STPA are shown in *Figure 2.1* along with a graphical representation of these steps.

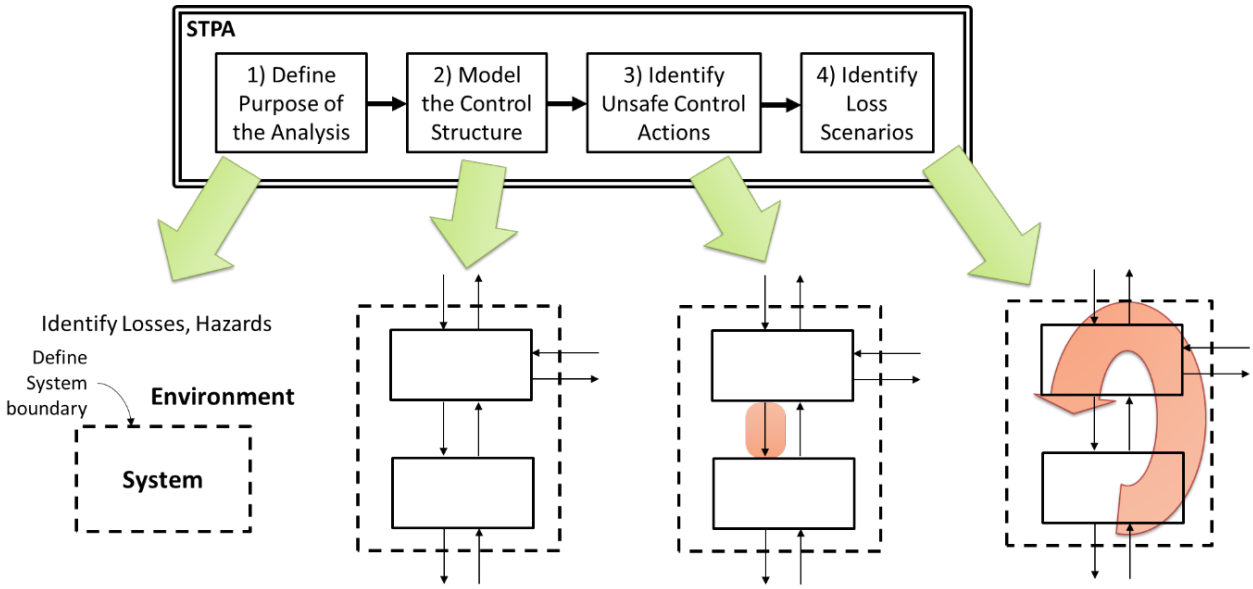


Figure 2.1: Overview of the basic STPA Method

Defining the purpose of the analysis is the first step with any analysis method. What kinds of losses will the analysis aim to prevent? Will STPA be applied only to traditional safety goals like preventing loss of human life or will it be applied more broadly to security, privacy, performance, and other system properties? What is the system to be analyzed and what is the system boundary? These and other fundamental questions are addressed during this step.

The second step is to build a model of the system called a control structure. A control structure captures functional relationships and interactions by modeling the system as a set of feedback control loops. The control structure usually begins at a very abstract level and is iteratively refined to capture more detail about the system. This step does not change regardless of whether STPA is being applied to safety, security, privacy, or other properties.

The third step is to analyze control actions in the control structure to examine how they could lead to the losses defined in the first step. These *unsafe control actions* are used to create functional requirements and constraints for the system. This step also does not change regardless of whether STPA is being applied to safety, security, privacy, or other properties.

The fourth step identifies the reasons why unsafe control might occur in the system. Scenarios are created to explain:

1. How incorrect feedback, inadequate requirements, design errors, component failures, and other factors could cause unsafe control actions and ultimately lead to losses.
2. How safe control actions might be provided but not followed or executed properly, leading to a loss.

2장: 기본적인 STPA 분석 수행 방법

John Thomas

STPA 방법 개요

*그림 2.1*에 기본적인 STPA의 단계별 절차를 도식화하였다.

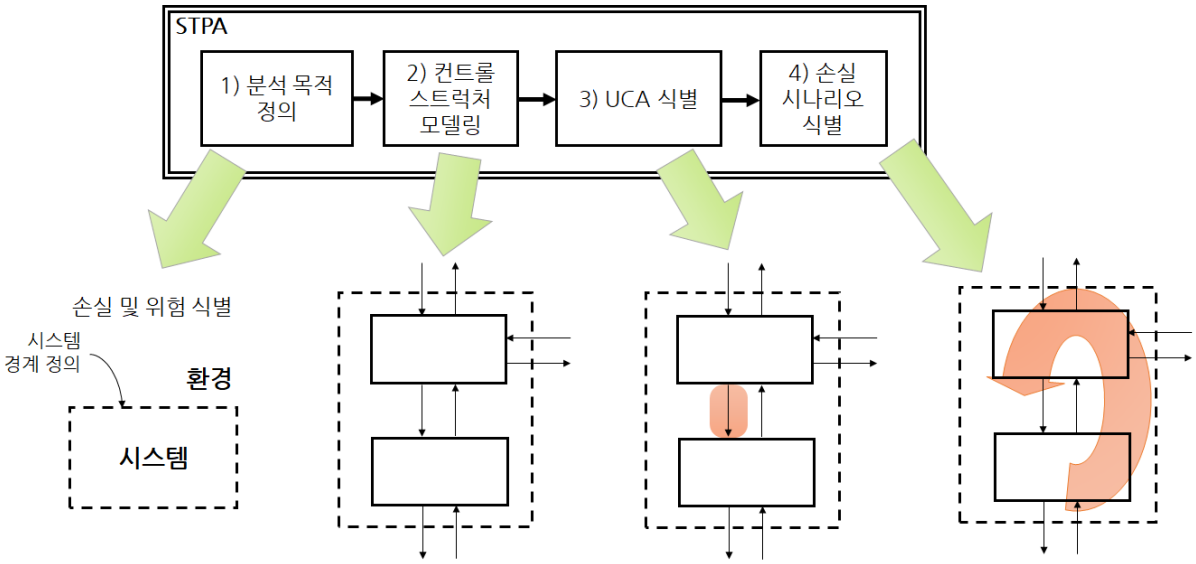


그림 2.1: STPA 방법 개요

분석 목적을 정의하는 것은 모든 분석에서 첫 번째 단계이다. 어떤 종류의 손실을 예방하기 위해 분석하는가? STPA를 인명 손실(loss) 방지와 같은 전통적인 안전 목표에만 적용할 것인가? 아니면 보다 광범위하게 보안, 개인 정보 보호, 성능 등의 다른 시스템 속성에까지 적용할 것인가? 분석할 시스템은 무엇이며 시스템 경계는 어디까지인가? 이 단계에서는 이러한 기본적인 질문을 논의한다.

두 번째 단계는 컨트롤 스트럭처(Control Structure)라고 불리는 시스템 모델을 구축하는 것이다. 컨트롤 스트럭처는 시스템을 일련의 피드백 컨트롤 루프(Feedback Control Loops)로 모델링하여 기능적 관계와 상호작용을 표현한다. 컨트롤 스트럭처는 일반적으로 매우 추상적인 수준에서 시작하여 시스템의 세부 정보를 표현하는 과정을 반복적으로 수행한다. 이 단계는 STPA를 안전, 보안, 개인 정보 또는 다른 속성에 적용하는지 여부에 관계없이 달라지지 않는다.

세 번째 단계는 컨트롤 스트럭처에서 컨트롤 액션(Control Action)을 분석하여 컨트롤 액션이 어떻게 첫 번째 단계에서 정의한 손실로 이어질 수 있는지를 조사하는 것이다. 이러한 *언세이프 컨트롤 액션(Unsafe Control Action, UCA)*은 시스템에 대한 기능 요구사항 및 안전 제약사항을 작성하는 데 사용된다. 이 단계 역시 STPA를 안전이나 보안, 개인 정보, 다른 속성에 적용하는지 여부에 관계없이 달라지지 않는다.

네 번째 단계는 시스템에서 안전하지 않은 컨트롤이 발생할 수 있는 원인을 식별하는 것이다. 여기서는 아래 항목을 설명하기 위한 시나리오를 만든다.

1. 부정확한 피드백(Feedback), 부적절한 요구사항, 설계 오류, 컴포넌트 실패 또는 다른 요인이 어떻게 언세이프 컨트롤 액션을 초래하여 궁극적으로는 손실로 이어질 수 있는가?

Once scenarios are identified, they can be used to create additional requirements, identify mitigations, drive the architecture, make design recommendations and new design decisions (if STPA is used during development), evaluate/revisit existing design decisions and identify gaps (if STPA is used after the design is finished), define test cases and create test plans, develop leading indicators of risk, and for other uses as described in later chapters of this handbook.

Defining the purpose of the analysis

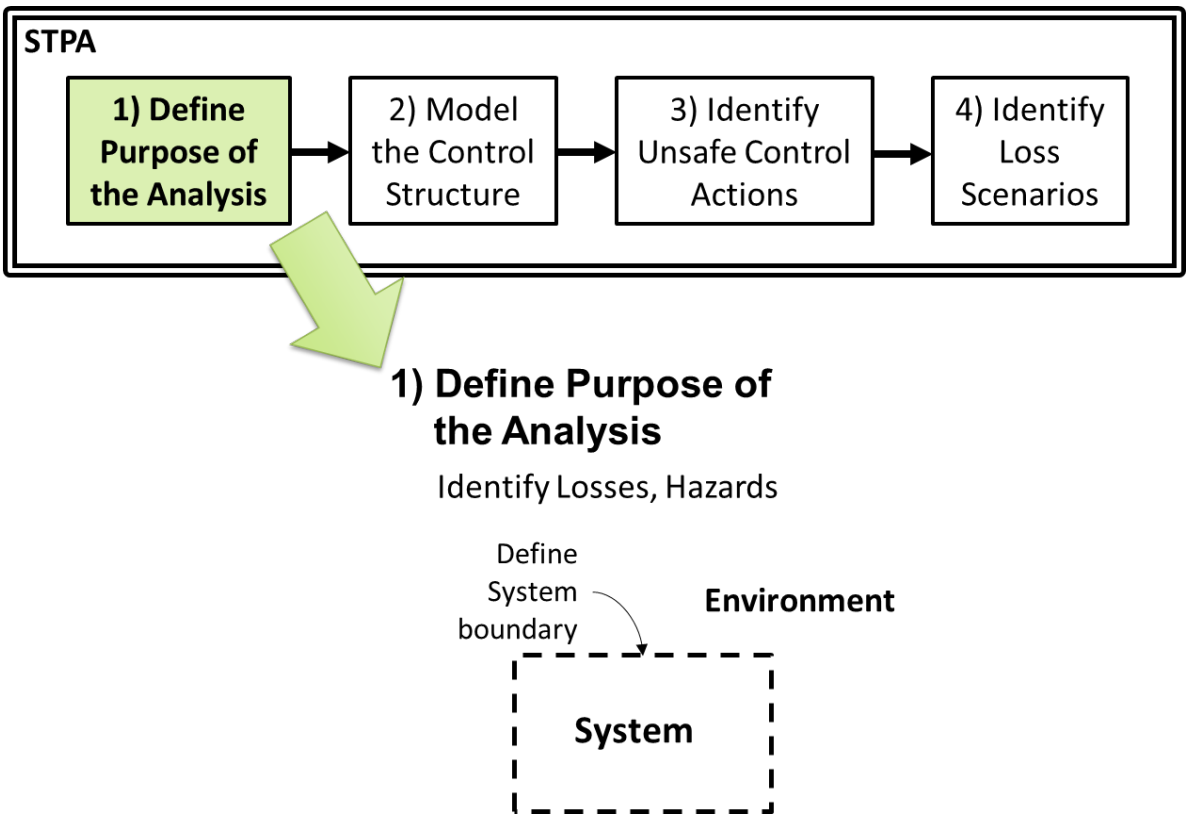


Figure 2.2: Defining the Purpose of the Analysis

As *Figure 2.2* shows, the first step in applying STPA is to define the purpose of the analysis. Defining the purpose of the analysis has four parts:

1. Identify losses
2. Identify system-level hazards
3. Identify system-level safety constraints
4. Refine hazards (optional)

Figure 2.3 summarizes the four parts described in this section.

2. 안전한 컨트롤 액션이 제공되었음에도 불구하고 어떤 이유로 실행되지 않거나 적절히 수행되지 않아 손실로 이어질 수 있는가?

시나리오는 추가 요구사항 작성, 완화조치 식별, 아키텍처 도출, 설계 권장 사항 및 새로운 설계 결정(개발 중에 STPA가 사용되는 경우), 기존 설계 결정 평가/재검토 및 갭 식별(STPA가 설계가 완료된 후 사용되는 경우), 테스트케이스 정의 및 테스트 계획 수립, 리스크 선행지표의 개발, 그리고 이 핸드북에서 이후 설명할 다른 용도 등에 활용할 수 있다.

분석 목적 정의

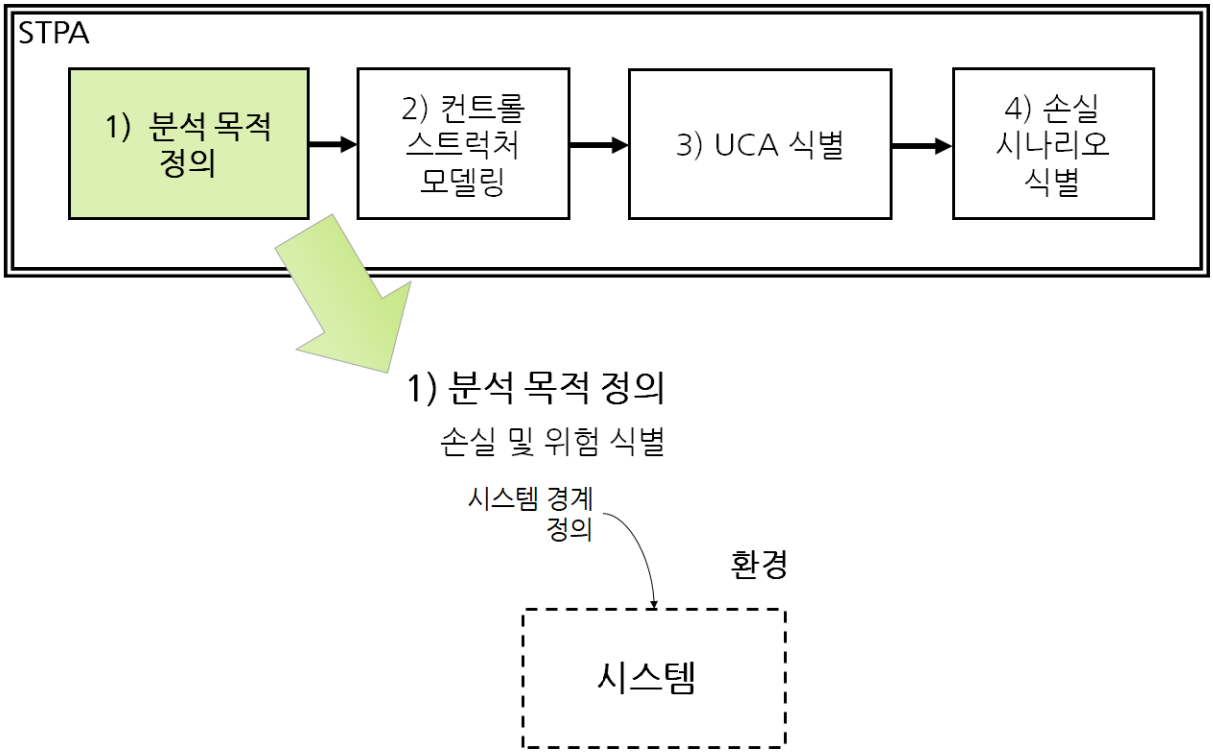


그림2.2: 분석의 목적 정의

*그림 2.2*와 같이 STPA 적용의 첫 번째 단계는 분석의 목적을 정의하는 것이다. 분석의 목적을 정의하는 것은 네 파트로 구성된다.

1. 손실(loss) 식별
2. 시스템 수준의 위험(hazards) 식별
3. 시스템 수준의 안전 제약사항(safety constraints) 식별
4. 위험 상세화(선택적)

이 절에서 설명하는 네 파트를 *그림 2.3*으로 요약하였다.

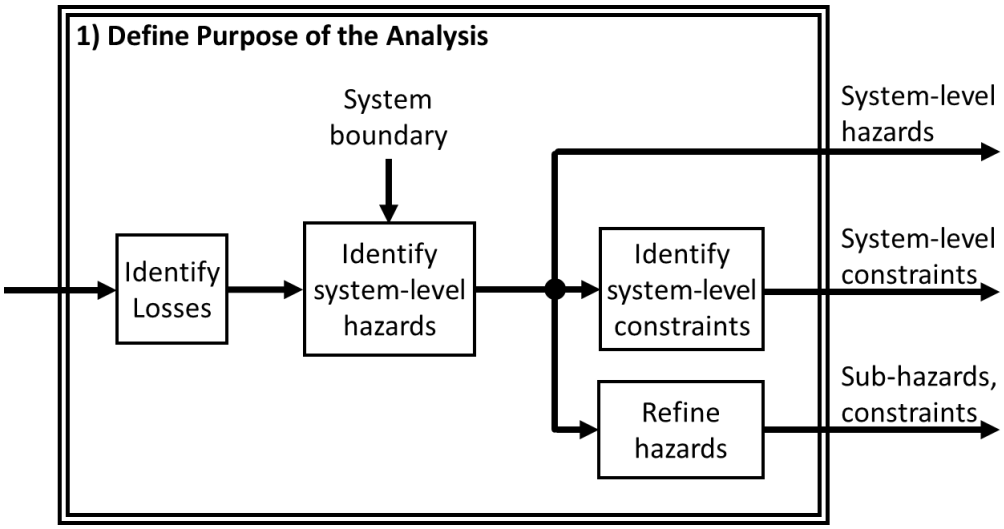


Figure 2.3: Overview of defining the analysis purpose

Identifying losses

> Definition: A *loss* involves something of value to stakeholders. Losses may include a loss of human life or human injury, property damage, environmental pollution, loss of mission, loss of reputation, loss or leak of sensitive information, or any other loss that is unacceptable to the stakeholders.

Different words are used to identify the goal of hazard analysis in different industries, for example, preventing an accident, a mishap, or an adverse event. To eliminate confusion about multiple terms used by practitioners in various industries, "losses" will be the term used in this chapter and the goal of STPA is to prevent losses.

STPA can be used to target any loss that is unacceptable to stakeholders. If more than one loss is included, they can be ranked and prioritized. Because every STPA result will be traceable to one or more losses, the analysis results can be easily ranked and prioritized based on the losses to which they refer.

Before any analysis begins, the stakeholders must identify the losses on which they want the analysis to focus. The losses to be considered may be defined by management, by government regulations, or by customers. A general approach to identifying losses may involve:

1. Identify the stakeholders, e.g. Users, producers, customers, operators, etc.
2. Stakeholders identify their "stake" in the system. What do they value? For example, human life, fleet of useable aircraft, electrical power generation, transportation, etc. What are their goals? For example, to maintain a fleet of useable aircraft, to provide transportation, to provide medical treatment, to provide electrical power generation, etc.
3. Translate each value or goal into a loss, e.g. loss of life, loss of aircraft, loss of electrical power generation, loss of transportation, etc.

The following list provides some examples of losses that users of the analysis often want to avoid:

L-1: Loss of life or injury to people
L-2: Loss of or damage to vehicle
L-3: Loss of or damage to objects outside the vehicle
L-4: Loss of mission (e.g. transportation mission, surveillance mission, scientific mission, defense mission, etc.)
L-5: Loss of customer satisfaction

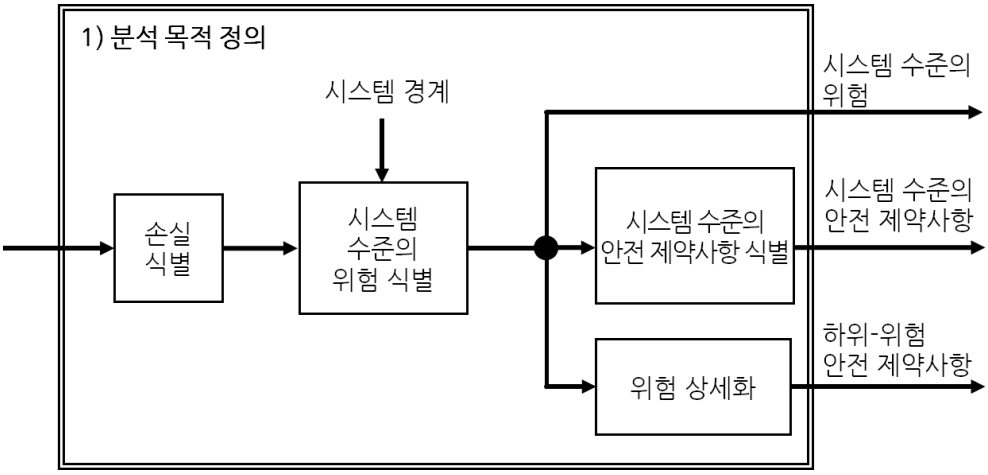


그림2.3: 분석 목적 정의의 개요

손실 식별

정의: *손실(loss)*은 이해관계자의 가치와 관련되어 있다. 손실은 인명 손실 또는 부상, 재산상의 손해, 환경 오염, 미션 실패, 평판의 손실, 민감한 정보의 손실이나 유출 또는 이해관계자가 수용 불가한 다른 손실을 포함할 수 있다.

위험분석의 목표를 식별하기 위해 여러 산업분야에서는 사고(accident), 미스햅(mishap), 부정적 이벤트(adverse event) 예방과 같은 각기 다른 단어들을 사용한다. 다양한 산업에서 사용하는 여러 용어들의 혼동을 없애기 위해 본 장에서는 "손실"이라는 용어를 사용하며, STPA의 목표는 손실을 방지하는 것이다.

STPA는 이해관계자가 수용할 수 없는 모든 형태의 손실을 대상으로 한다. 손실이 둘 이상인 경우 순위를 매기거나 우선순위를 부여할 수 있다. 모든 STPA 결과는 하나 이상의 손실에 대해 추적 가능(traceable)하기 때문에 분석 결과는 해당 손실을 기준으로 쉽게 순위를 지정하고 우선순위를 매길 수 있다.

분석을 시작하기 전에 이해관계자는 집중적으로 분석할 손실을 식별해야 한다. 고려해야 할 손실은 경영진, 정부 규정 또는 고객이 정의할 수 있다. 손실을 식별하기 위한 일반적인 접근법에는 다음과 같은 것들이 있을 수 있다.

1. 사용자, 생산자, 고객, 운영자 등 이해관계자(stakeholders)를 식별한다.
2. 이해관계자는 시스템에서 자신의 "이해관계(stake)"를 식별한다. 무엇이 소중한 것인가? 예를 들어 사람의 생명, 사용 가능한 항공기 함대, 전력생산, 운송 등. 목표는 무엇인가? 예를 들어 항공기 함대를 유지하는 것, 운송을 제공하는 것, 의료 서비스를 제공하는 것, 전력 발전을 제공하는 것 등.
3. 각 가치 또는 목표를 손실로 변환한다. 예를 들어, 생명의 손실, 항공기의 손실, 전력생산의 손실, 운송의 손실 등.

다음 목록은 분석 사용자가 종종 피하고자 하는 몇 가지 손실의 예이다.

L-1: 사람의 생명 손실이나 부상
L-2: 차량의 손실 또는 손상
L-3: 차량 외부 대상의 손실 또는 손상
L-4: 임무의 손실(예를 들어, 수송 임무, 감시 임무, 과학 임무, 국방 임무 등)

L-6: Loss of sensitive information
L-7: Environmental loss
L-8: Loss of power generation

Some tips to prevent common mistakes when identifying losses:

1. Losses can include any loss that is unacceptable to any stakeholder
2. Losses should not reference individual components or specific causes like “human error” and “brake failure”
3. Losses may involve aspects of the environment that are not directly controlled by the system designer
4. Document any special considerations or assumptions made, such as losses that are explicitly excluded

After the losses of concern in the analysis are identified, the next step is to define the hazards related to these losses.

Identifying system-level hazards

Definition: A *hazard* is a system state or set of conditions that, together with a particular set of worst-case environmental conditions, will lead to a loss.

Definition: A *system* is a set of components that act together as a whole to achieve some common goal, objective, or end. A system may contain subsystems and may also be part of a larger system.

To identify the system-level hazards, you will first need to identify the system to be analyzed and the system boundary. A system is an abstraction conceived by the analyst. A decision must be made about what is included in the system and what the system boundary is. With respect to engineering, the most useful way to define the system boundary for analysis purposes is to include the parts of the system over which the system designers have some control. This is the primary reason for distinguishing between hazards and losses—losses may involve aspects of the environment over which the system designer or operator has only partial control or no control at all. The goal of safety engineering is to eliminate or mitigate the effects of hazards in the system under analysis so some level of control is necessary. *Figure 2.4* illustrates the system boundary as an abstraction that separates a system from its environment.

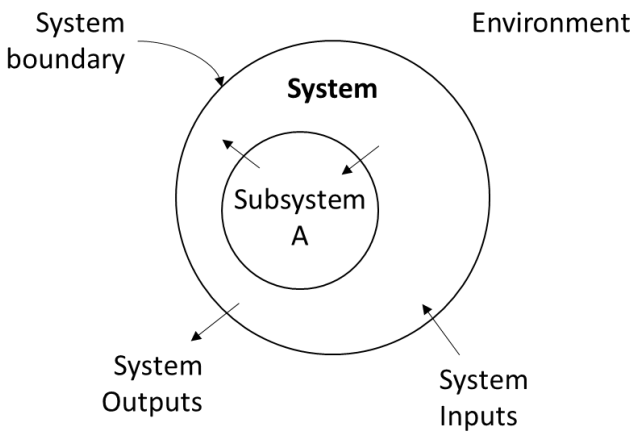


Figure 2.4: The relationship between a system, system boundary, and the environment

L-5: 고객 만족의 손실
L-6: 민감한 정보 손실
L-7: 환경 손실
L-8: 전력 발전 손실

손실을 식별할 때 일반적인 실수를 방지하기 위한 몇 가지 팁:

1. 손실에는 어떤 이해관계자에게든 수용할 수 없는 손실이면 포함될 수 있다.
2. 손실은 개별 컴포넌트 또는 "인적 오류" 및 "브레이크 오류"와 같은 특정 원인을 참조해서는 안 된다.
3. 손실은 시스템 설계자가 직접 제어하지 않는 환경 측면을 포함할 수 있다.
4. 명시적으로 제외된 손실과 같은 특별한 고려 사항이나 가정을 문서화해야 한다.

분석 단계에서 관심 있는 손실을 식별한 후에는 다음 단계에서 이러한 손실과 관련된 위험을 정의한다.

시스템 수준 위험 식별

정의: *위험*은 특정한 최악의 환경 조건들의 집합 하에서 손실로 이어질 수 있는 시스템 상태 또는 조건들의 집합이다.

정의: *시스템(System)*은 몇 가지 공통의 목표 또는 목적, 결과를 달성하기 위하여 전체적으로 함께 동작하는 컴포넌트의 집합이다. 시스템은 서브시스템을 포함할 수 있으며 더 큰 시스템의 일부일 수도 있다.

시스템 수준 위험을 식별하려면 먼저 분석할 시스템과 시스템 경계를 식별해야 한다. 시스템은 분석가가 생각하는 추상적인 개념이다. 시스템에 포함된 것이 무엇인지, 시스템 경계는 무엇인지에 대한 결정이 반드시 이루어져야 한다. 공학적 측면에서 시스템 경계를 정의하는 가장 유용한 방법은 시스템 설계자가 제어할 수 있는 시스템 영역을 포함시키는 것이다. 이것이 위험과 손실을 구별하는 가장 중요한 이유이다. 손실은 시스템 설계자 또는 운영자가 부분적으로 제어하거나 전혀 제어하지 않는 환경 측면까지도 포함할 수 있는데, 안전 공학의 목표는 분석중인 시스템에서 위험의 영향을 제거하거나 완화하는 것이며, 이를 위해서는 어느 정도의 제어가 필요하다. *그림 2.4*는 시스템 경계를 시스템과 환경으로 구분하여 추상적으로 보여준다.

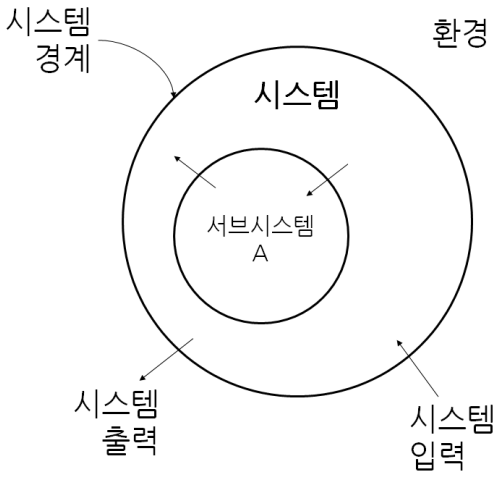


그림 2.4: 시스템과 시스템 경계, 환경 사이의 관계

As an example, consider a nuclear power plant. A release of radioactive materials, the proximity of a nearby population or city, and the direction of the wind may all be important factors that lead to a potential loss of life. However, as engineers we cannot control the wind and we may not be able to control the city location, but we *can* control the release of radioactive materials in or outside the plant (a system-level hazard).

Once the system and system boundary is identified, the next step is to define the system-level hazards by identifying system states or conditions that will lead to a loss in worst-case environmental conditions. The following list provides some examples of system-level hazards:

H-1: Aircraft violate minimum separation standards in flight [L-1, L-2, L-4, L-5]
H-2: Aircraft airframe integrity is lost [L-1, L-2, L-4, L-5]
H-3: Aircraft leaves designated taxiway, runway, or apron on ground [L-1, L-2, L-5]
H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]
H-5: Satellite is unable to collect scientific data [L-4]
H-6: Vehicle does not maintain safe distance from terrain and other obstacles [L-1, L-2, L-3, L-4]
H-7: UAV does not complete surveillance mission [L-4]
H-8: Nuclear power plant releases dangerous materials [L-1, L-4, L-7, L-8]

In general, a hazard can lead to one or more losses and each hazard should be traced to the resulting losses. This traceability is typically documented in brackets after the hazard description. The examples above show the traceability to the examples of losses on the previous page.

There are three basic criteria for defining system-level hazards:

- Hazards are *system* states or conditions (not component-level causes or environmental states)
- Hazards *will* lead to a loss in some worst-case environment
- Hazards must describe states or conditions to be prevented

First, the system-level hazards are system states or conditions, not external environmental states that are outside the designer's control. In addition, system-level hazards should not describe detailed component-level causes like a physical component failure (e.g. a hydraulic leak, insufficient brake fluid, etc.). Referencing component-level causes during this step will overly restrict the analysis, making it easy to overlook other less obvious causes during later steps. Instead, identify the system-level states or conditions to be prevented (the hazards) and allow the later STPA steps to systematically identify component-level causes of the hazards.

Second, there must be a worst-case environment in which hazards will lead to a loss. This requirement does not necessarily guarantee that a hazard will *always* result in a loss. For example, a chemical plant may release toxic materials but wind and weather conditions prevent the toxic materials from impacting nearby personnel and populated regions. However, in a worst-case environment, the toxic materials can be carried to populated areas and lead to losses.

Finally, hazards are states or conditions to be prevented. "Aircraft in flight" is a system state that could arguably lead to a loss in a worst-case environment, but it is not a condition to be eliminated or prevented (otherwise we would not be building aircraft). Hazards should be states to be prevented and states that we never want the system to get into—not states that the system must normally be in to accomplish its goals.

Common mistakes when identifying system-level hazards

Confusing hazards with causes of hazards

예를 들어, 원자력 발전소를 생각해보자. 방사성 물질의 방출, 인근 인구 또는 도시에의 근접도 및 바람의 방향은 모두 잠재적인 생명 손실로 이어질 수 있는 중요한 요소일 수 있다. 그러나 엔지니어는 바람을 컨트롤 할 수 없고 도시의 위치도 컨트롤 할 수 없지만, 시스템 내·외부의 방사성 물질 방출(시스템 수준의 위험)을 컨트롤 할 수는 있다.

시스템 및 시스템 경계가 식별되면 다음 단계는 최악의 환경 조건에서 손실을 초래할 수 있는 시스템 상태나 조건을 식별함으로써 시스템 수준의 위험을 정의하는 것이다. 다음은 시스템 수준의 위험에 대한 몇 가지 예시이다.

H-1: 항공기가 비행 중 최소 간격 기준을 위반한다 [L-1, L-2, L-4, L-5]
H-2: 항공기가 기체 무결성을 상실한다 [L-1, L-2, L-4, L-5]
H-3: 항공기가 지상의 지정된 유도로(taxiway), 활주로(runway) 또는 계류장 경로(apron path)를 벗어난다 [L-1, L-2, L-5]
H-4: 항공기가 지상의 다른 물체에 너무 가까이 접근한다 [L-1, L-2, L-5]
H-5: 위성이 과학적 데이터를 수집할 수 없다 [L-4]
H-6: 차량이 지형 및 기타 장애물로부터 안전한 거리를 유지하지 못한다 [L-1, L-2, L-3, L-4]
H-7: UAV(무인항공기)가 감시 임무를 완수하지 못한다 [L-4]
H-8: 원자력 발전소가 위험한 물질을 방출한다 [L-1, L-4, L-7, L-8]

일반적으로 위험은 하나 또는 그 이상의 손실로 이어질 수 있으며 각 위험은 결과로 발생한 손실로 추적할 수 있어야 한다. 이 추적성은 일반적으로 위험 설명 뒤에 괄호로 표기된다. 위의 예는 이전 페이지의 손실 사례에 대한 추적성을 보여준다.

시스템 수준의 위험을 정의하는 세 가지 기본 기준이 있다.

- 위험은 시스템 상태 또는 조건이다(컴포넌트 수준의 원인 또는 환경 상태가 아님)
- 위험은 최악의 환경에서 손실로 이어질 것이다.
- 위험은 예방되어야 하는 상태 또는 조건을 반드시 기술해야 한다.

첫째, 시스템 수준의 위험은 시스템 상태 또는 조건이며 설계자의 제어 범위를 벗어난 외부 환경 상태는 해당되지 않는다. 또한 시스템 수준의 위험은 물리적 컴포넌트 실패(예: 유압 누출, 브레이크 오일 부족 등)와 같은 상세한 컴포넌트 수준의 원인을 나타내서는 안 된다. 이 단계에서 컴포넌트 수준의 원인을 참조하면 분석이 지나치게 제한되고, 이후 단계에서 불분명한 다른 원인을 간과하기 쉽다. 대신 예방할 시스템 수준의 상태 또는 조건(위험)을 식별하고 이후의 STPA 단계에서 위험 원인을 체계적으로 식별하도록 한다.

둘째, 위험이 손실로 이어지게 만드는 최악의 환경 조건들이 반드시 기술되어야 한다. 이러한 환경 조건들이 위험이 항상 손실로 이어지는 것을 의미하지는 않는다. 예를 들어 화학 공장에서 유독 물질을 방출할 수 있지만, 유독 물질이 인근 주민 및 인구 밀집 지역에 영향을 미치는 것을 바람과 기상 조건이 막을 수 있다. 그러나 최악의 환경에서는 독성 물질이 사람이 거주하는 지역으로 운반되어 손실을 초래할 수 있다.

마지막으로 위험은 예방되어야 할 상태 또는 조건이다. “비행중인 항공기”는 최악의 환경에서는 손실을 초래할 수 있는 시스템 상태이지만, 제거하거나 예방할 수 있는 조건은 아니다(그렇지 않으면 항공기를 만들지 말아야 한다). 위험은 예방되어야 할 상태로서 시스템이 절대로 진입하지 않았으면 하는 상태이며, 목표 달성을 위해 시스템이 정상적인 상황에서 진입해서는 안 되는 상태이다.

A common mistake in defining hazards is to confuse hazards with causes of hazards. For example, “brake failure”, “brake failure not annunciated”, “operator is distracted”, “engine failure”, and “hydraulic leak” are not system-level hazards but potential *causes* of hazards. To avoid this mistake, make sure the identified hazards do not refer to individual components of the system, like brakes, engines, hydraulic lines, etc. Instead, the hazards should refer to the overall system and system states. In other words, check that each hazard contains:

<Hazard specification> = <System> & <Unsafe Condition> & <Link to Losses>

E.g. H-1 = Aircraft violate minimum separation standards in flight [L-1, L-2, L-4, L-5]

The exact sequence is not important—you could just as easily write “Minimum separation standards for aircraft are violated [L-1, L-2, L-4, L-5]”. What is important is that the system-level hazards contain these elements.

Too many hazards containing unnecessary detail

Like losses, there are no hard limits on the number of system-level hazards to include. As a rule of thumb, if you have more than about seven to ten system-level hazards, consider grouping or combining hazards to create a more manageable list. You may be including unnecessary detail and making the list unmanageable, difficult to review, and harder to identify things that are missing. Instead, begin with a more abstract and manageable set of system-level hazards and refine them into sub-hazards later if needed (as explained in the section on refining hazards below).

Ambiguous or recursive wording

The system-level hazards define exactly what “unsafe” means at the system level. A common mistake is to use the word “unsafe” in the hazards themselves. Doing so creates a recursive definition and does not add information or value to the analysis. For example, it might be tempting to write “H-1: Aircraft experiences unsafe flight [L-1]”. It can be tempting because it certainly sounds dangerous—an unsafe flight by definition must be hazardous, right? The problem is that it is too vague and does not help specify the actual condition that is unsafe. Some have fallen into the same trap with statements like “H-1: Aircraft experiences an unsafe state [L-1]”, only to struggle with the rest of the analysis and miss important cases. A simple solution is to avoid using the word “unsafe” in the hazard itself and instead specify exactly what is meant by “unsafe”—what system states or conditions would make it unsafe? For example, an aircraft that is uncontrolled or that is too close to other aircraft would be unsafe. As you will see, specifying actual conditions like this is extremely useful during later STPA steps.

Confusing hazards with failures

Professionals who are experienced in other hazard analysis methods sometimes fall into the trap of writing STPA hazards describing potential deviations from specified technical functions or describing physical component failures. You may be familiar with traditional techniques that begin by searching for a set of deviations, faults, or functional failures in the technical system. To identify a broader set of causes in STPA, we cannot assume that the defined and specified functions are safe and correct, that human operators will perform as expected, that automated behaviors will not induce human error or confusion, that off-nominal cases will not occur, or that the technical design, specification, and requirements are correct. For example, the hazard “Controlled flight of aircraft into terrain” can be included in STPA while it may be omitted by efforts to examine only purely technical functional failures.

Hazard identification in STPA is about system states and conditions that are inherently unsafe—regardless of the cause. In fact, the system hazards should be specified at a high-enough level that does not distinguish between causes related to technical failures, design errors, flawed requirements, or human procedures and interactions.

시스템 수준의 위험을 식별할 때 흔히 저지르는 실수

위험과 위험의 원인(Causes of hazards)을 혼동

위험을 정의할 때 흔히 저지르는 실수는 위험과 위험의 원인을 혼동하는 것이다. 예를 들어 "브레이크 고장", "브레이크 고장 미경고", "운영자 산만함", "엔진 고장" 및 "유압 누출"은 시스템 수준의 위험이 아니라 위험의 잠재적인 원인이다. 이러한 실수를 피하려면 식별된 위험이 브레이크, 엔진, 유압 라인 등과 같은 시스템의 개별 컴포넌트를 언급하지 않도록 해야 한다. 대신 위험은 전체 시스템 및 시스템 상태를 언급해야 한다. 다시 말해 각 위험이 다음 사항을 포함하는지 확인해야 한다.

〈위험 명세〉 = 〈시스템〉 & 〈안전하지 않은 상태〉 & 〈손실에 대한 링크〉
예 H-1 = 항공기가 최소 이격기준을 위반함[L-1, L-2, L-4, L-5]

정확한 순서는 중요하지 않다. "항공기의 최소 이격기준이 위반됐다.[L-1, L-2, L-4, L-5]"라고 편하게 쓸 수도 있다. 중요한 것은 시스템 수준의 위험에는 이러한 요소를 포함해야 한다는 것이다.

불필요한 세부 사항이 포함된 위험이 너무 많음

손실과 마찬가지로, 시스템 수준의 위험 개수에 대한 엄격한 제한은 없다. 일반적으로 시스템 수준 위험이 약 7~10개 이상인 경우 위험을 그룹핑하거나 결합하여 보다 관리하기 쉬운 목록을 만드는 것이 좋다. 불필요한 세부 사항을 포함하면 목록을 관리하기 어렵거나 검토하기 어렵고 누락된 항목을 식별하기가 어려울 수 있다. 때문에, 보다 추상적이고 관리가 용이한 시스템 수준 위험 집합에서 시작하고 필요할 경우 나중에 하위 위험(sub-hazards)으로 상세화한다(아래의 위험 상세화 절에서 설명).

모호하거나 재귀적인 표현

시스템 수준의 위험은 시스템 수준에서 "안전하지 않은(unsafe)"의 의미를 정확히 정의하는 것이다. 일반적인 실수는 위험 자체를 기술할 때 "안전하지 않은" 이라는 단어를 사용하는 것이다. 이런 기술은 재귀적인 정의(recursive definition)를 만들게 되고, 분석 시에 필요한 정보나 값을 제공하지 않는다. 예를 들어 "H-1: 항공기가 안전하지 않게 비행함[L-1]" 이라고 쓰고 싶을 수도 있다. 확실히 위험해 보이기 때문에 이렇게 쓰고 싶은 유혹을 받을 수 있다. 안전하지 않은 비행은 정의로 보자면 위험해야 한다. 그렇지 않은가? 문제는 너무 애매하며 안전하지 않은 실제 조건을 명세하는 데 도움이 되지 않는다는 것이다. 일부는 "H-1: 항공기가 안전하지 않은 상태를 경험함[L-1]"과 같은 문장을 사용하여 동일한 함정에 빠지는데, 이는 나머지 분석을 엉망으로 만들고 중요한 것을 놓치게 만들 뿐이다. 간단한 해결책으로는 위험 자체에서 "안전하지 않은(unsafe)"이라는 단어를 사용하지 말고 대신 "안전하지 않은"이 의미하는 바를 정확하게 명시하는 것이다 – 시스템의 어떤 상태 또는 어떤 조건으로 인해 안전하지 않은가? 예를 들어, 통제 불능이거나 다른 항공기와 너무 가까운 항공기는 안전하지 않을 것이다. 앞으로 보게 되겠지만 이와 같이 실제 조건을 명시하는 것은 이후 STPA 단계에서 매우 유용하다.

위험(Hazards)과 실패(Failure)를 혼동

다른 위험 분석 방법에 경험이 있는 전문가는 STPA 위험을 작성할 때 명세된 기술적 기능에서 가능한 일탈상황(deviation)을 기술하거나, 물리적 컴포넌트의 실패를 기술하는 등의 함정에 빠질 수 있다. 그러한 기술적 시스템에서 일탈상황 또는 결함, 기능 실패를 찾는 것에서 시작하는 기존 방법에 익숙할 수 있다. STPA에서는 더 광범위한 원인을 식별하기 위해, 정의되거나 명세된 기능이 안전하고 정확하며, 운영자(사람)가 예상대로 행동하고, 자동화된 동작이 사람의 실수나 혼동을 유발하지 않는다고 가정하지 않는다. 또한, 비정상(off-nominal) 상황이 발생하지 않거나, 기술 설계, 사양 및 요구사항이

What should I look for when reviewing hazards?

Tips to prevent common mistakes when identifying hazards

- Hazards should not refer to individual components of the system
- All hazards should refer to the overall system and system state
- Hazards should refer to factors that can be controlled or managed by the system designers and operators
- All hazards should describe system-level conditions to be prevented
- The number of hazards should be relatively small, usually no more than 7 to 10
- Hazards should not include ambiguous or recursive words like “unsafe”, “unintended”, “accidental”, etc.

Note that STPA is an iterative method and the hazards need not be set in stone at this point. Later STPA steps may uncover new hazards and this list can be revisited and revised as needed.

Defining system-level constraints

Definition: A system-level constraint specifies system conditions or behaviors that need to be satisfied to prevent hazards (and ultimately prevent losses)

Once the system-level hazards are identified, it is straightforward to identify system-level constraints that must be enforced: simply invert the condition.

<Hazard> = <System> & <Unsafe Condition> & <Link to Losses>
<Safety Constraint> = <System> & <Condition to Enforce> & <Link to Hazards>

H-1: Aircraft violate minimum separation standards [L-1, L-2, L-4, L-5]
SC-1: Aircraft must satisfy minimum separation standards from other aircraft and objects [H-1]

H-2: Aircraft airframe integrity is lost [L-1, L-2, L-4, L-5]
SC-2: Aircraft airframe integrity must be maintained under worst-case conditions [H-2]

Each constraint can be traceable to one or more hazards, and each hazard is traceable to one or more losses. In general, the traceability need not be one-to-one; a single constraint might be used to prevent more than one hazard, multiple constraints may be related to a single hazard, and each hazard could lead to one or more losses.

Constraints can also define how the system must minimize losses in case the hazards do occur. For example, if aircraft do violate minimum separation then the violation must be detected and measures must be taken to keep the aircraft from colliding. If a chemical plant does release toxic chemicals, then the toxic environment must be detected and appropriate measures taken. These constraints can generally be written as:

<Safety Constraint> = If <hazard> occurs, then <what needs to be done to prevent or minimize a loss>

SC-3: If aircraft violate minimum separation, then the violation must be detected and measures taken to prevent collision

The safety constraints should not specify a particular solution or implementation. For example, instead of specifying solutions like installing a collision avoidance system, SC-3 simply states that the

올바르다고 가정하지 않는다. 예를 들어 "조종상태에서의 지형충돌(CFIT, Controlled flight into terrain)"이라는 위험은 STPA 분석 과정에 포함될 수 있지만, 단지 기술적인 기능 결함만을 검토하는 경우에는 위험이 누락될 수 있다.

STPA의 위험 식별은 위험의 원인에 관계없이 본질적으로 안전하지 않은 시스템의 상태와 조건에 대한 것이다. 실제로 시스템 위험은 기술적인 실패, 설계 오류, 잘못된 요구사항, 운영자 절차 및 상호작용과 관련된 원인을 구분하지 못하는, 충분히 높은 수준으로 명세되어야 한다.

위험을 검토할 때 무엇을 확인해야 할까?

위험을 식별할 때 흔한 실수를 방지하는 팁

- 위험은 시스템의 개별 컴포넌트를 언급하지 않아야 한다.
- 모든 위험은 전체 시스템 및 시스템 상태에 대해 언급해야 한다.
- 위험은 시스템 설계자와 운영자가 제어하거나 관리할 수 있는 요소를 언급해야 한다.
- 모든 위험은 방지되어야 할 시스템 수준의 조건(condition)을 기술해야 한다.
- 위험의 수는 비교적 적어야 하며, 보통 7-10개 보다 많지 않아야 한다.
- 위험은 "안전하지 않은(unsafe)", "의도하지 않은(unintended)", "우연한(accidental)" 등과 같은 모호하거나 재귀적인 단어를 포함해서는 안 된다.

STPA는 반복적인 방법이므로 이 시점에서 위험을 확정할 필요는 없다. STPA의 이후 단계에서 새로운 위험을 발견할 수 있으며 이 목록은 필요에 따라 다시 검토하고 수정할 수 있다.

시스템 수준의 안전 제약사항 정의

정의: 시스템 수준의 안전 제약사항은 위험을 방지(궁극적으로는 손실을 방지)하기 위해 충족되어야 할 시스템의 조건(condition) 또는 동작(behavior)을 기술한다.

시스템 수준의 위험이 식별되면 반드시 지켜져야 하는 시스템 수준의 안전 제약사항을 바로 파악할 수 있다: 간단하게 조건을 뒤집으면 된다.

〈위험〉 = 〈시스템〉 & 〈안전하지 않은 조건〉 & 〈손실에 대한 링크〉
〈안전 제약사항〉 = 〈시스템〉 & 〈지켜져야 하는 조건〉 & 〈손실에 대한 링크〉

H-1: 항공기가 최소이격기준을 위반한다[L-1, L-2, L-4, L-5]
SC-1: 항공기는 다른 항공기 및 물체와의 최소이격기준을 충족해야 한다[H-1]

H-2: 항공기가 기체 무결성을 상실한다[L-1, L-2, L-4, L-5]
SC-2: 항공기의 기체 무결성은 최악의 조건에서도 반드시 유지되어야 한다[H-2]

각 안전 제약사항은 하나 이상의 위험으로부터 추적될 수 있으며 각 위험은 하나 이상의 손실로부터 추적될 수 있다. 일반적으로 추적성(traceability)은 일대일(1:1)일 필요는 없다; 하나의 안전 제약사항은 하나 이상의 위험을 방지하기 위해 사용될 수 있으며, 여러 안전 제약사항이 하나의 위험과 관련될 수 있고, 각 위험은 하나 이상의 손실을 초래할 수 있다.

안전 제약사항은 위험이 발생할 경우 시스템이 손실을 어떻게 최소화해야 하는지도 정의할 수 있다. 예를 들어 항공기가 최소이격을 위반하면, 위반 사항을 탐지해야 하며 항공기가 충돌하지 않도록 조치해야 한다. 화학 공장에서 독성 화학 물질이 방출되면, 독성 환경을 탐지하고 적절한 조치를 취해야 한다. 이러한 안전 제약사항은 일반적으로 다음과 같이 작성할 수 있다.

violation must be detected and there must be some way to prevent collision. Specifying a particular solution is usually premature at this early stage and can result in alternative and potentially better solutions being overlooked.

The rest of the STPA analysis will systematically identify scenarios that can violate these constraints, leading to system-level hazards and losses.

Refining the system-level hazards (optional)

Once the list of system-level hazards has been identified and reviewed, these hazards can be refined into sub-hazards if appropriate. Sub-hazards are not necessary for many STPA applications[6], but they can be useful for large analysis efforts and complex applications because they can guide future steps like modeling the control structure (see the next section about control structures).

The first step in refining the system-level hazards is to identify basic system processes or activities that need to be controlled to prevent system hazards. For example, consider the system-level hazard we identified earlier for commercial aviation:

H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]

One way to derive sub-hazards is to ask: What do we need to control to prevent this hazard? To control the aircraft on the ground, we will need some way to control aircraft deceleration, acceleration, and steering. If these are not controlled adequately (for example if the deceleration is insufficient), it could lead to a system-level hazard.

The following sub-hazards can be derived for H-4:

H-4: Aircraft comes too close to other objects on the ground [L-1, L-2, L-5]

Deceleration
H-4.1: Deceleration is insufficient upon landing, rejected takeoff, or during taxiing
H-4.2: Asymmetric deceleration maneuvers aircraft toward other objects
H-4.3: Deceleration occurs after V1 point during takeoff
Acceleration
H-4.4: Excessive acceleration provided while taxiing
H-4.5: Asymmetric acceleration maneuvers aircraft toward other objects
H-4.6: Acceleration is insufficient during takeoff
H-4.7: Acceleration is provided during landing or when parked
H-4.8: Acceleration continues to be applied during rejected takeoff
Steering
H-4.9: Insufficient steering to turn along taxiway, runway, or apron path
H-4.10: Steering maneuvers aircraft off the taxiway, runway, or apron path

Each of these sub-hazards can be used to produce more specific constraints. *Table 2.1* shows the constraints derived for deceleration based on the sub-hazards above.

Table 2.1: Deriving specific process constraints from the sub-hazards

Sub-hazards derived from H-4	Example constraints
H-4.1: Deceleration is insufficient upon landing, rejected takeoff, or during taxiing	SC-6.1: Deceleration must occur within TBD seconds of landing or rejected takeoff at a rate of at least TBD m/s^2

[6] Those new to STPA may find it helpful to begin with smaller applications that do not require this step.

〈안전 제약사항〉 = 만약 〈위험〉이 발생하면 〈손실을 방지하거나 최소화하기 위해 해야 할 일〉

SC-3: 만약 항공기가 최소이격을 위반하면 위반을 탐지하고 충돌 방지 조치를 취해야 한다

안전 제약사항은 특정 솔루션이나 구현을 명시해서는 안 된다. 예를 들어 SC-3은 충돌 회피 시스템 설치와 같은 솔루션을 명시하는 대신, '위반 사항을 탐지해야 하며 충돌을 방지할 수 있는 방법이 있어야 한다'와 같이 단순하게 기술하고 있다. 일반적으로 초기 단계에서 특정 솔루션을 명시하는 것은 시기상조이며 대안이나 잠재적으로 더 나은 솔루션을 간과하게 하는 결과를 초래할 수 있다.

나머지 STPA 분석에서는 이러한 안전 제약사항 위반으로 시스템 수준의 위험과 손실을 초래할 수 있는 시나리오를 체계적으로 식별할 것이다.

시스템 수준의 위험 상세화(선택사항)

시스템 수준의 위험들이 식별되어 검토된 후, 필요한 경우 이러한 위험을 하위 위험(sub-hazards)으로 상세화 할 수 있다. 하위 위험은 대부분의 STPA 적용 시에는 필요하지 않지만[6] 많은 분석이 필요한 복잡한 경우에는 유용할 수 있다. 왜냐하면 컨트롤 스트럭처를 모델링하는 것(다음 섹션 컨트롤 스트럭처 참조)과 같은 다음 단계를 가이드 할 수 있기 때문이다.

시스템 수준 위험을 상세화하는 첫 번째 단계는 시스템 위험을 방지하기 위해 제어해야 하는 기본적인 시스템 프로세스 또는 활동을 식별하는 것이다. 예를 들어 앞서 식별한 상업용 항공기에 대한 시스템 수준의 위험을 생각해 보자:

H-4: 항공기가 지상의 다른 물체에 너무 가까이 접근한다 [L-1, L-2, L-5]

하위 위험을 이끌어 내는 한 가지 방법은 다음과 같은 질문을 하는 것이다: 이 위험을 방지하려면 무엇을 제어해야 할까? 지상에서 항공기를 제어하려면 항공기의 감속, 가속 및 스티어링(steering)을 제어할 수 있는 방법이 필요하다. 이들이 적절히 제어되지 않으면(예를 들어 감속이 불충분한 경우) 시스템 수준의 위험이 초래될 수 있다.

H-4에 대해 다음과 같은 하위 위험이 도출될 수 있다.

H-4: 항공기가 지상의 다른 물체에 너무 가까이 접근한다 [L-1, L-2, L-5]

감속(Deceleration)

H-4.1: 착륙, 이륙중지, 지상활주 중(taxiing) 감속이 불충분하다.
H-4.2: 비대칭적 감속으로 항공기가 다른 물체를 향하여 기동(maneuvers)한다.
H-4.3: 이륙 중 V1 포인트를 지나서 감속이 발생한다.

가속(Acceleration)

H-4.4: 지상활주 중에(taxiing) 과도하게 가속된다.
H-4.5: 비대칭적 감속으로 항공기가 다른 물체를 향하여 기동한다.
H-4.6: 이륙 중 가속이 충분하지 않다.
H-4.7: 착륙 또는 주차 중 가속된다.
H-4.8: 이륙중지(Rejected take-off) 동안 계속 가속된다.

스티어링(steering)

[6] STPA를 처음 접하는 사용자는, 이 단계가 필요하지 않은 보다 작은 어플리케이션에 적용을 시작해보는 것이 도움이 될 수 있다.

H-4.2: Asymmetric deceleration maneuvers aircraft toward other objects	SC-6.2: Asymmetric deceleration must not lead to loss of directional control or cause aircraft to depart taxiway, runway, or apron
H-4.3: Deceleration occurs after V1 point during takeoff	SC-6.3: Deceleration must not be provided after V1 point during takeoff

Modeling the control structure

The next step in STPA is to model the hierarchical control structure, as *Figure 2.5* shows.

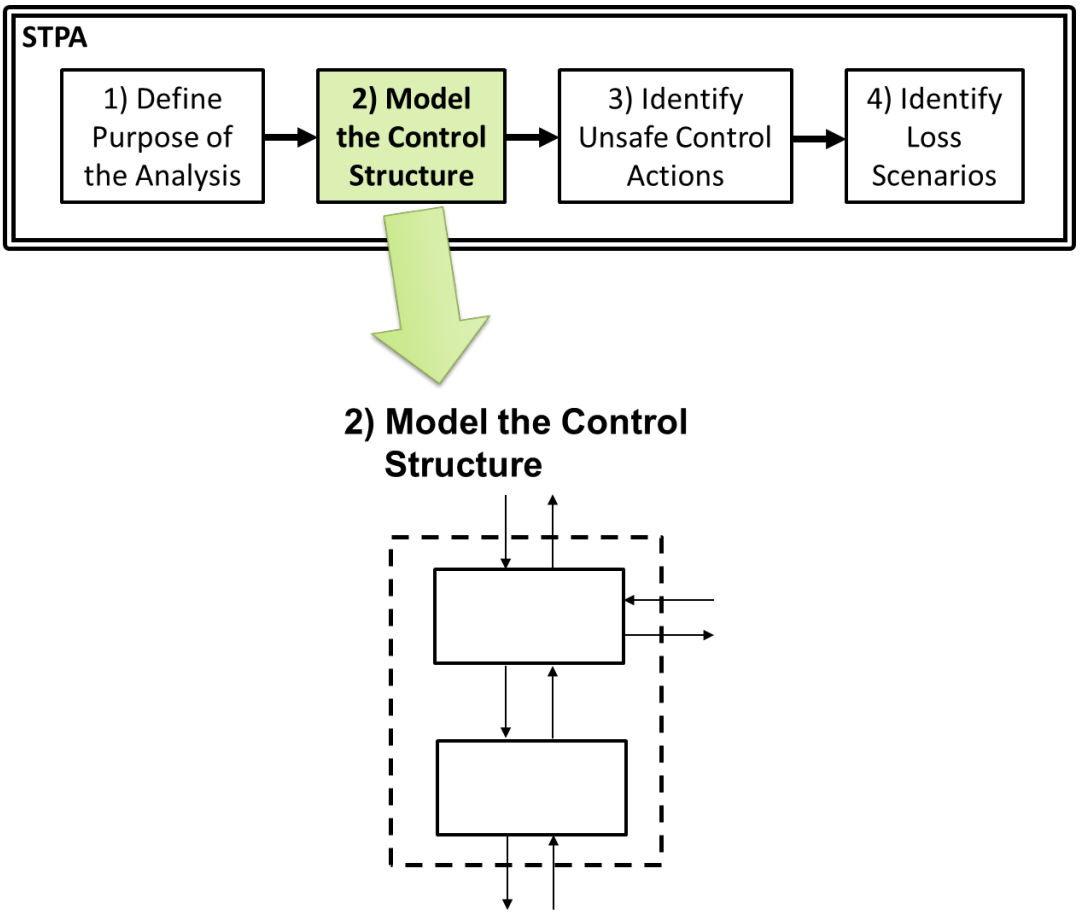


Figure 2.5: Modeling the control structure

What is a control structure?

> Definition: A hierarchical control structure is a system model that is composed of feedback control loops. An effective control structure will enforce constraints on the behavior of the overall system.

A hierarchical control structure is composed of control loops like the one shown in *Figure 2.6*. In general, a controller may provide control actions to control some process and to enforce constraints on the behavior of the controlled process. The control algorithm represents the controller's decision-making process—it determines the control actions to provide. Controllers also have process models that represent the controller's internal beliefs used to make decisions. Process models may include beliefs

H-4.9: 유도로(taxiway), 활주로(runway), 계류장 경로(apron path)를 따라 회전하기에 스티어링이 부족하다.

H-4.10: 스티어링이 항공기가 유도로, 활주로, 계류장 경로를 벗어나도록 기동된다.

각 하위 위험은 보다 구체적인 안전 제약사항을 생성하는 데 사용될 수 있다. *표 2.1*은 위의 하위 위험을 기반으로 감속에 대해 도출된 안전 제약사항을 보여준다.

표 2.1: 하위 위험에서 도출한 특정 프로세스 안전 제약사항

H-4에서 도출된 하위 위험	안전 제약사항의 예
H-4.1: 착륙, 이륙중지, 지상활주 중에 감속이 불충분하다.	SC-6.1: 착륙 또는 이륙중지 TBD초 이내에 적어도 TBD m/s^2 속도로 감속이 이루어져야 한다.
H-4.2: 비대칭적 감속으로 항공기가 다른 물체를 향하여 기동된다.	SC-6.2: 비대칭 감속으로 방향 제어가 불가능하거나, 항공기가 유도로, 활주로, 계류장에서 절대로 이탈하지 않아야 한다.
H-4.3: 이륙 중 V1 포인트를 지나서 감속이 발생한다.	SC-6.3: 이륙 중 V1 포인트 이후 절대로 감속되지 않아야 한다.

컨트롤 스트럭처(Control Structure) 모델링

STPA의 다음 단계는 그림 2.5와 같이 계층적 컨트롤 스트럭처를 모델링하는 것이다.

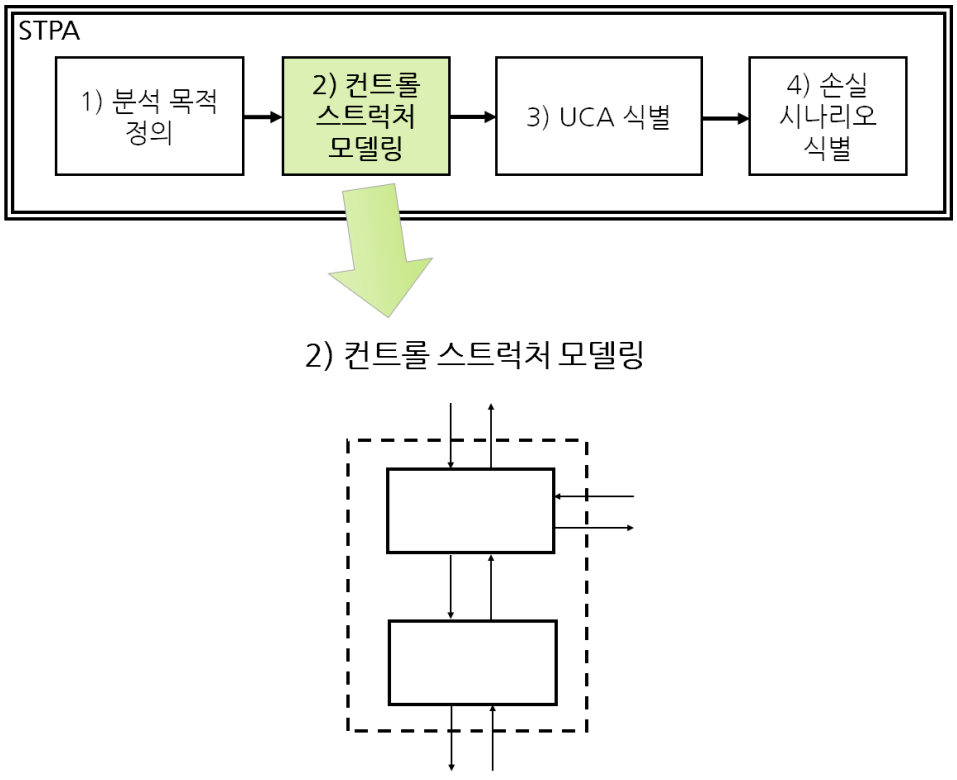


그림 2.5: 컨트롤 스트럭처 모델링

컨트롤 스트럭처란 무엇인가?

정의: 계층적 컨트롤 스트럭처는 피드백 컨트롤 루프로 구성된 시스템 모델이다. 효과적인 컨트롤

about the process being controlled or other relevant aspects of the system or the environment. Process models may be updated in part by feedback used to observe the controlled process.

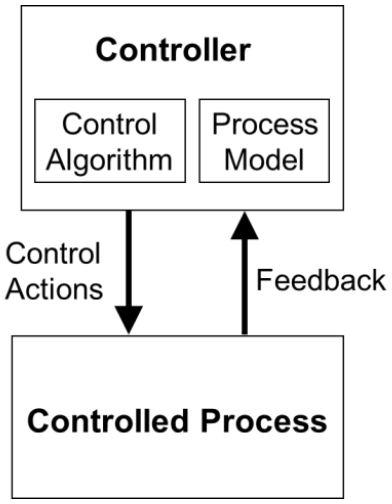


Figure 2.6: Generic control loop

Problems can occur at any point in *Figure 2.6*. For example, a process model may not be consistent with reality (e.g. a controller believes an aircraft is descending when it is really ascending or believes a car is in Park when it is really in Reverse, the airport runway is empty, etc.) can lead to control actions that are unsafe. A sensor failure may cause incorrect feedback and lead to unsafe behavior. A design may be missing the necessary feedback or may provide delayed feedback that results in a process model flaw and unsafe behavior. As you'll see, STPA provides a way to systematically identify these and other scenarios that can lead to a loss.

The generic control loop in *Figure 2.6* can be used to explain and anticipate complex software and human interactions that can lead to losses—two of the biggest challenges in modern engineering. For humans, the process model is usually called a mental model and the control algorithms may be called operating procedures or decision-making rules, but the basic concept is the same.

Of course, most systems typically have several overlapping and interacting control loops. Multiple interacting control loops can be modeled in a hierarchical control structure, as shown in *Figure 2.7*. A simple example for aviation is shown in *Figure 2.8*.

스트럭처는 전체 시스템의 동작에 안전 제약사항을 강제할 것이다.

계층적 컨트롤 스트럭처는 *그림 2.6*에 나타낸 것과 같이 컨트롤 루프로 구성된다. 일반적으로 컨트롤러(Controller)는 일부 프로세스를 제어하고 컨트롤드 프로세스(controlled Process)의 동작에 안전 제약사항을 강제하기 위해 컨트롤 액션을 제공할 것이다. 컨트롤 알고리즘(control algorithm)은 컨트롤러의 의사 결정 절차를 나타낸다 — 컨트롤 액션을 제공할지 여부를 결정한다. 컨트롤러에는 의사 결정에 사용되는 컨트롤러의 내부적인 믿음(internal belief)을 나타내는 프로세스 모델(process Model)도 있다. 프로세스 모델에는 제어되는 프로세스, 시스템 또는 환경과 관련된 측면에 대한 믿음이 포함될 수 있다. 프로세스 모델은 컨트롤드 프로세스를 관찰하는 데 사용되는 피드백(feedback)에 의해 부분적으로 업데이트 될 수 있다.

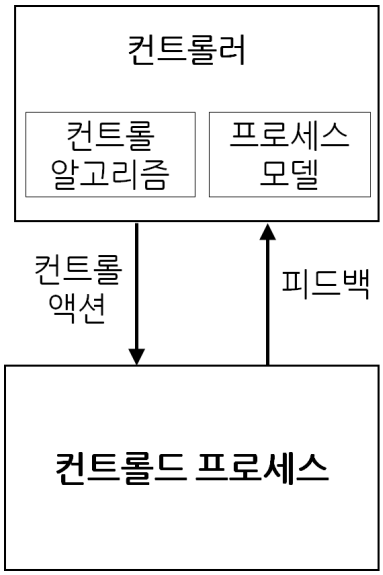


그림 2.6: 일반적인 컨트롤 루프

문제는 *그림 2.6*의 어느 지점에서든 발생할 수 있다. 예를 들어 프로세스 모델이 현실과 일치하지 않은 경우(즉, 컨트롤러가 실제로는 상승 중인 항공기를 하강한다고 믿는 경우, 자동차 기어가 실제로는 R(후진)에 있지만 P(주차)에 있다고 믿는 경우, 공항 활주로가 비어 있지 않은데 비어있다고 믿는 경우 등)는 안전하지 않은 컨트롤 액션으로 이어질 수 있다. 센서(sensor)의 실패는 잘못된 피드백을 야기할 수 있고 안전하지 않은 동작(unsafe behavior)으로 이어질 수 있다. 필요한 피드백이 설계에서 누락되거나 지연되도록 설계될 수 있는데, 이는 프로세스 모델의 결함이나 안전하지 않은 동작을 초래한다. 앞으로 보겠지만, STPA는 손실로 이어질 수 있는 이러한 시나리오를 체계적으로 식별할 수 있는 방법을 제공한다.

*그림 2.6*의 일반적인 컨트롤 루프는 손실로 이어질 수 있는 복잡한 소프트웨어와 사람의 상호작용을 설명하고 예상하는 데에도 사용될 수 있다 — 이 둘은 현대의 엔지니어링에서 가장 중요한 두 가지 숙제이다. 사람의 경우 프로세스 모델을 일반적으로 멘탈 모델(mental model)이라고 부르며 컨트롤 알고리즘은 운영 절차 또는 의사 결정 규칙이라고 할 수 있으며 기본 개념은 동일하다.

물론 대부분의 시스템은 일반적으로 몇 가지 오버랩 및 상호작용하는 컨트롤 루프를 가지고 있다. 다수의 상호작용하는 컨트롤 루프는 *그림 2.7*에 나타난 바와 같이 계층적 컨트롤 스트럭처로 모델링 될 수 있다. 간단한 항공기 사례를 *그림 2.8*에 나타내었다.

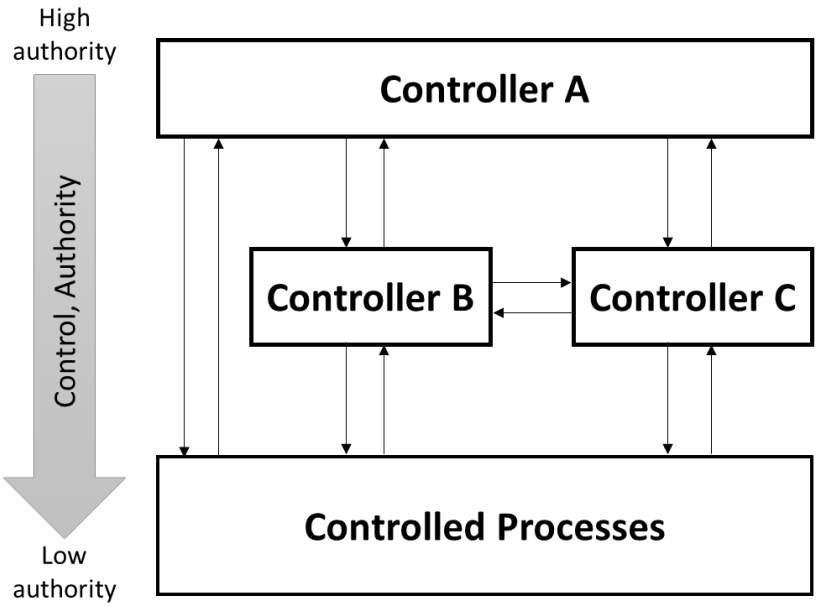


Figure 2.7: Generic hierarchical control structure

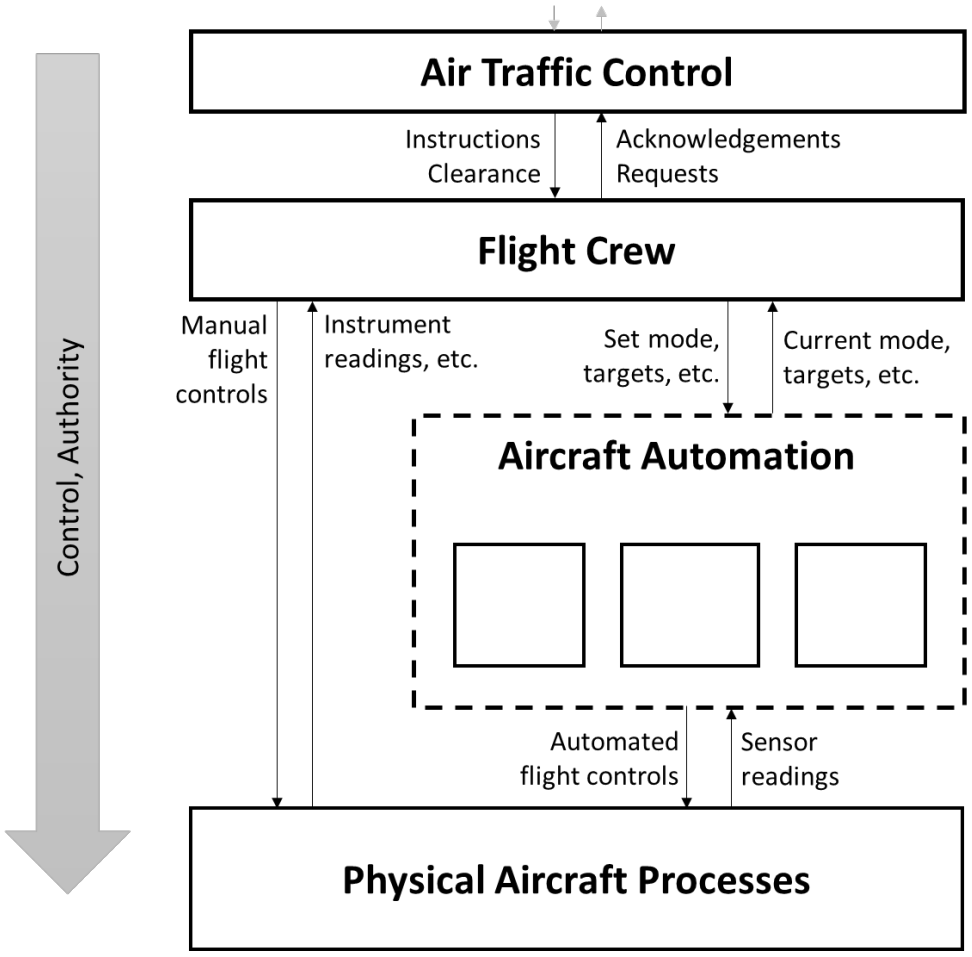


Figure 2.8: Simple example of a hierarchical control structure for aviation

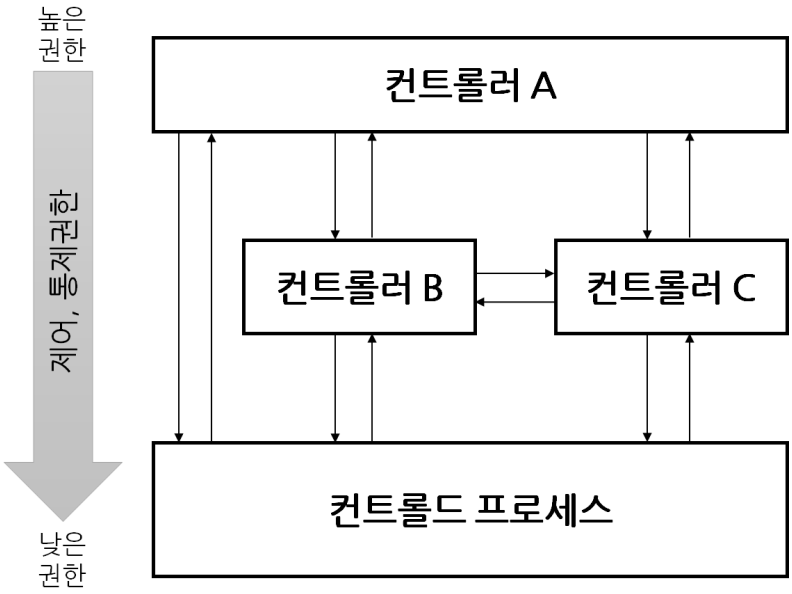


그림 2.7: 일반적인 계층적 컨트롤 스트럭처

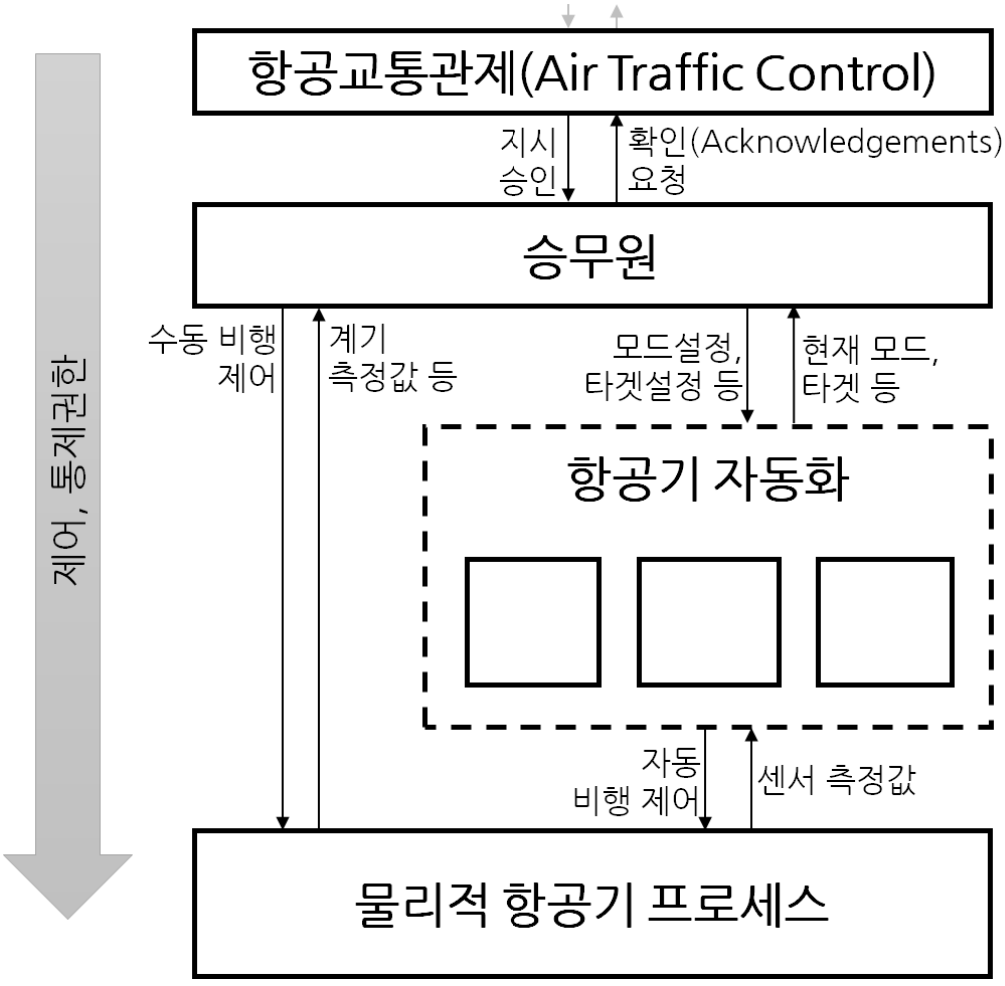


그림 2.8: 항공(aviation)의 계층적 컨트롤 스트럭처의 간단한 예시

In general, a hierarchical control structure contains at least five types of elements[7]:

- Controllers
- Control Actions
- Feedback
- Other inputs to and outputs from components (neither control nor feedback)
- Controlled processes

The vertical axis in a hierarchical control structure is meaningful: it indicates control and authority within the system. The vertical placement represents the hierarchy of control from high-level controllers at the top to the lowest-level entities at the bottom. Each entity has control and authority over the entities immediately below it, and each entity is likewise subject to control and authority from the entities immediately above. For example, the aircraft automation in *Figure 2.8* can act as a controller by sending control actions to physical aircraft systems and monitoring feedback. At the same time, the aircraft automation is also a controlled process that receives and executes control actions from the flight crew and sends feedback to the crew.

In other words, all downward arrows represent control actions (commands) while the upward arrows represent feedback. These conventions help to manage complexity and make control relationships and feedback loops easier to recognize.[8] In some cases, simply drawing a control structure diagram can make previously undiscovered flaws painfully obvious. For example, control actions may be provided by entities that don't have the necessary feedback to select safe control actions, feedback may be provided to entities with no ability to do anything about it, multiple controllers may be able to provide conflicting commands to the same component with no ability to detect or resolve the conflict, etc. If you do not catch flaws at this stage though, don't worry—the rest of the STPA method will systematically identify these and other issues.

Common Points of Confusion

A control structure is not a physical model

The hierarchical control structure used in STPA is a functional model, not a physical model like a physical block diagram, a schematic, or a piping and instrumentation diagram. The connections show information that can be sent, such as commands and feedback—they do not necessarily correspond to physical connections. For example, the interactions between the flight crew and air traffic control are not of a physical nature, but they are modeled in a functional control structure.

A control structure is not an executable model

The control structure is not an executable model or a simulation model. In fact, control structures often include components for which executable models do not exist (such as humans). Instead, STPA can be used to carefully derive the necessary behavioral constraints, requirements, and specifications needed enforce the desired system properties. For example, the control structure in *Figure 2.8* does not assume that air traffic control will always sent instructions to the flight crew when needed, that they will always have the capability to send instructions (e.g. that radios will always be operational), that the correct instructions will always be sent, or that the instructions will always be followed by the pilots. It simply indicates that a system was/will be created to allow air traffic control to send instructions to the

[7] A control structure can also contain actuators and sensors, but these are usually added later during the scenario identification step

[8] Control structures can also be drawn left to right if needed, but we are purposely simplifying the discussion here.

일반적으로 계층적 컨트롤 스트럭처는 적어도 5가지 유형의 요소들을 포함한다[7]:

- 컨트롤러
- 컨트롤 액션
- 피드백
- 그 외 다른 컴포넌트로의 입력 및 다른 컴포넌트로부터의 출력(컨트롤이나 피드백 아님)
- 컨트롤드 프로세스

계층적 컨트롤 스트럭처의 세로축은 시스템 내에서의 제어(control)와 통제권한(authority)을 의미한다. 수직 배치는 맨 위의 상위 수준 컨트롤러에서 맨 아래의 하위 수준 개체(entity)로의 제어 계층 구조를 나타낸다. 각 개체는 그 바로 아래 개체에 대한 제어와 통제권한을 가지며, 같은 의미로 각 개체는 바로 위에 있는 개체의 제어 및 통제권한에 따른다. 예를 들어 *그림2.8*의 항공기 자동화는 물리적 항공기 시스템에 컨트롤 액션을 보내고 피드백을 감시함으로써 컨트롤러 역할을 수행할 수 있다. 동시에 항공기 자동화는 승무원으로부터 컨트롤 액션을 받아서 실행하고 승무원에게 피드백을 보내는 컨트롤드 프로세스이기도 하다.

즉, 아래로 향하는 모든 화살표는 컨트롤 액션(명령)을 나타내고, 위로 향하는 화살표는 피드백을 나타낸다. 이러한 규칙은 복잡성을 관리하고 제어 관계 및 피드백 루프를 보다 쉽게 인지하도록 도와준다[8]. 어떤 경우에는 컨트롤 스트럭처 다이어그램을 그리는 것만으로도 이전에 발견되지 않은 너무나 명백한 결함을 발견할 수도 있다. 예를 들어 안전한 컨트롤 액션을 선택하기 위해 필요한 피드백이 전달되지 않는 개체가 컨트롤 액션을 제공할 수 있거나, 제공된 피드백에 대하여 아무것도 할 수 없는 개체에 피드백이 제공될 수 있고, 제어 명령의 충돌을 탐지하거나 해결할 능력이 없는 컴포넌트에 여러 컨트롤러가 충돌하는 제어 명령을 제공하는 것 등이 있다. 만일 이 단계에서 결함을 발견하지 못했다 하더라도 걱정할 필요는 없다. STPA 방법의 나머지 부분에서 이러한 문제 및 다른 이슈를 체계적으로 식별한다.

흔히 혼동하는 포인트

컨트롤 스트럭처는 물리적 모델이 아니다.

STPA의 계층적 컨트롤 스트럭처는 물리적 블록 다이어그램, 계통도(schematic) 또는 배관 및 계기도면(piping and instrumentation diagram)과 같은 물리적 모델이 아닌, 기능적 모델(functional model)이다. 연결은 컨트롤이나 피드백과 같이 전송할 수 있는 정보를 나타내며 반드시 물리적 연결과 일치하는 것은 아니다. 예를 들어 승무원과 ATC(Air Traffic Control, 항공교통관제) 간의 상호작용은 물리적인 것은 아니지만 기능적인 컨트롤 스트럭처로 모델링된다.

컨트롤 스트럭처는 실행 가능한 모델이 아니다.

컨트롤 스트럭처는 실행 가능한 모델 또는 시뮬레이션 모델이 아니다. 실제로 컨트롤 스트럭처에는 실행 가능한 모델이 존재하지 않는 컴포넌트(예: 사람)가 포함되는 경우가 있다. 대신 STPA를 사용하여 원하는 시스템 속성을 적용하기 위해 반드시 필요한 동작 제약사항, 요구사항 및 명세를 면밀히 도출할 수 있다. 예를 들어 *그림2.8*의 컨트롤 스트럭처는 ATC가 필요한 경우에 항상 승무원에게 지시(instruction)를 전송하거나, 지시를 보낼 수 있는 기능을 항상 갖추고 있다고(예: 라디오가 항상 작동

[7] 컨트롤 스트럭처는 액추에이터 및 센서를 포함할 수도 있지만 일반적으로 나중에 시나리오 식별 단계에서 추가된다.

[8] 필요에 따라 컨트롤 스트럭처를 왼쪽에서 오른쪽으로 그릴 수도 있지만 여기에서는 의도적으로 설명을 단순화하고 있다.

flight crew. The next steps in STPA will carefully examine how unsafe behaviors may occur, including instructions that are sent but not received, unsafe instructions that may be sent, etc. Although the control structure itself is not an executable model, the STPA method will produce precise requirements and other outputs that can be used to generate executable models and specifications.

A control structure does not assume obedience

Do not confuse controllers and control actions with obedience. Just because a controller sends a control action does not mean that in practice it will always be followed. Likewise, just because a feedback path is included in a control structure does not mean that in practice the feedback will always be sent when needed or that it will be accurate. The control actions and feedback in a control structure simply indicate that a mechanism will be created to send this information (that is, it will be in the system design). It does not imply or assume anything about how controllers and processes will actually behave in practice. In fact, a major goal of STPA is to analyze the control structure and anticipate how each element might behave in unsafe and potentially unexpected ways.

Use abstraction to manage complexity

One of the biggest challenges in any hazard analysis is managing system complexity. Control structures use abstraction in several ways to help manage complexity. For example, in commercial flights the flight crew might consist of two or three individual pilots. Rather than clutter the control structure right from the beginning with three separate pilots, we can group them together as a flight crew that collectively provides control actions and collects feedback. Similarly, rather than explicitly listing every individual aircraft subsystem, we could begin at a more abstract level by modeling aircraft automation and the physical processes they control as two levels in the control hierarchy.

The principle of abstraction can also be applied to the command and feedback paths in the control structure. Rather than listing each individual button, switch, and lever in the cockpit, we might begin with much broader actions like a climb maneuver. Later, we could refine these broad actions into pitch, thrust, or other commands as appropriate. This principle is especially useful during early development phases when individual commands and sensors are not yet known.

The control action path may contain mechanisms by which the controller acts upon a controlled process (referred to as actuators) and mechanisms by which the controller senses feedback from a controlled process (referred to as sensors). These details are usually abstracted away when initially creating the control structure, but the control structure will be refined to include actuators and sensors later during the scenario creation step.

One additional type of abstraction is used in control structures. Consider the flight commands sent from the flight crew to the aircraft automation. In a remotely piloted UAV application, those commands may need to pass through many different components—from a physical button on the command console, through an embedded system that encodes the command within a digital packet, to a network switch, a radio transmitter, a satellite, and a radio receiver on the UAV. It is not necessary to show all of these detailed steps along the control path in the initial control structure—what matters is that the remote pilot will have some way to send flight commands to the UAV.

In fact, the most efficient way to apply STPA is to begin before those design decisions have been made and before such details are known. The abstract control structure above can be used to begin STPA and identify the requirements and constraints for the communication path and other parts of the system. Then, STPA results can be used to drive the architecture, preliminary and detailed design, make implementation decisions, and refine the control structure. Even if details are known and design decisions have been made, it can be helpful to first apply STPA at a higher abstract level first to provide quicker results and identify broader issues before analyzing more detailed control structure models.

할 수 있음) 가정하지 않는다. 또한, 정확한 지시가 항상 보내지거나, 조종사가 항상 지시를 지키는 것을 가정하지 않는다. 이것은 ATC가 단순히 승무원에게 지시를 보낼 수 있도록 시스템이 만들어졌거나 만들어질 것을 나타낼 뿐이다. STPA의 다음 단계에서는, 전송되었지만 수신되지 않은 지시, 안전하지 않은 지시 전송 등을 포함하여 안전하지 않은 동작이 어떻게 발생할 수 있는지를 신중하게 검토한다. 컨트롤 스트럭처 자체가 실행 가능한 모델은 아니지만 STPA 기법은 실행 가능한 모델과 명세를 생성할 수 있는 정확한 요구사항을 비롯한 여러 결과를 만들 수 있다.

컨트롤 스트럭처는 항상 지켜지는 것(obedience)이라고 가정하지 않는다.

컨트롤러와 컨트롤 액션이 항상 지켜지는 것이라고 혼동해서는 안 된다. 즉, 컨트롤러가 컨트롤 액션을 전송하는 것이 실제로 그것이 항상 지켜진다는 것을 의미하지는 않는다. 마찬가지로 피드백 경로가 컨트롤 스트럭처에 포함되어 있다고 해서 실제로 필요할 때마다 항상 피드백이 보내지거나 정확하게 전달한다는 의미는 아니다. 컨트롤 스트럭처의 컨트롤 액션 및 피드백은 단순히 이 정보를 전송하는 메커니즘이 생성될 것임을 나타내는 것이다(즉, 시스템 설계에 포함됨). 이것은 컨트롤러와 프로세스가 실제로 어떻게 작동할 것인가에 대해서는 암시하거나 가정하지 않는다. 실제 STPA의 주요 목표는 컨트롤 스트럭처를 분석하여 각 요소가 안전하지 않고 잠재적으로 예상치 못한 방식으로 동작하는 것을 예측하는 것이다.

복잡성을 관리하기 위해 추상화(抽象化, abstraction)를 사용해라.

어떤 위험분석이든지 가장 큰 과제 중 하나는 시스템 복잡성을 관리하는 것이다. 컨트롤 스트럭처는 복잡성을 관리하기 위해 여러 가지 방법으로 추상화를 사용한다. 예를 들어, 상업 비행에서 승무원은 2~3명의 개별 조종사로 구성될 수 있다. 컨트롤 스트럭처를 처음부터 3명의 개별 조종사로 급히 정의하기 보다는, 여러 컨트롤 액션을 제공하고 여러 피드백을 수집하는 하나의 승무원으로 그룹화할 수 있다. 마찬가지로, 모든 개별 항공기의 서브시스템을 명시적으로 나열하는 대신, 컨트롤 계층에서 두 단계로 컨트롤하는 항공기 자동화 및 물리적 프로세스를 모델링하여 보다 추상적인 수준에서 시작할 수 있다.

추상화 원리는 컨트롤 스트럭처의 컨트롤 및 피드백 경로에도 적용할 수 있다. 조종석의 각 버튼, 스위치 및 레버를 나열하는 대신, '상승 조작'과 같은 훨씬 더 광범위한 액션에서 시작할 수 있다. 이후, 이 광범위한 액션을 피치(pitch), 추진력, 또는 다른 적절한 명령으로 상세화할 수 있다. 이 원리는 개별 명령과 센서가 아직 구체화되지 않은, 초기 개발 단계에서 특히 유용하다.

컨트롤 액션 경로는 컨트롤러가 컨트롤드 프로세스에 작용하는 메커니즘(액추에이터라고 함) 및 컨트롤러가 컨트롤드 프로세스로부터 피드백을 감지하는 메커니즘(센서라고 함)을 포함할 수 있다. 이러한 세부 사항은 일반적으로 컨트롤 스트럭처를 처음 생성할 때에는 추상화되지만 나중에 시나리오 작성 단계에서 액추에이터 및 센서를 포함하도록 상세화된다.

컨트롤 스트럭처에는 한 유형의 추상화가 추가적으로 사용된다. 승무원이 항공기 자동화에 보낸 비행 명령을 고려해보자. 원격 조종되는 UAV 어플리케이션에서 이러한 명령은 명령 콘솔의 물리적 단추, 디지털 패킷 내 명령을 인코딩하는 임베디드 시스템, 네트워크 스위치, 무선 송신기, 위성 및 UAV 라디오 수신기와 같은 서로 다른 여러 컴포넌트를 통해 전달될 수 있다. 초기 컨트롤 스트럭처에서 컨트롤 경로에 존재하는 이러한 세부적인 단계를 모두 표시할 필요는 없다 — 중요한 것은 원격 조종사가 UAV에 비행 명령을 보낼 수 있는 방법이 있다는 것이다.

실제로 STPA를 적용하는 가장 효율적인 방법은 위와 같은 설계 결정이 내려지기 전에 그리고 그러한 세부 사항을 알기 전에 STPA 적용을 시작하는 것이다. 위의 추상적인 컨트롤 스트럭처는 STPA를 시작하는 데 사용할 수 있고, 커뮤니케이션 경로 및 시스템의 다른 부분에 대한 요구사항 및 안전

Modeling the control structure

Control structure modeling begins with an abstract control structure and iteratively adds detail. In many cases, the control structure and the control loops within the system may be obvious or can be reused from previous applications. If the control structure is not obvious, the following guidance explains one way to model a functional control structure and add detail. Appendix B contains examples of control structures that have been used in different industries.

One way to begin is to identify basic subsystems needed to enforce the constraints and prevent the hazards identified earlier. For example, we derived hazards and constraints related to insufficient deceleration. These constraints could be enforced by using a wheel braking subsystem, reverse thrust, and other subsystems. *Figure 2.9* shows an initial control structure with these subsystems.

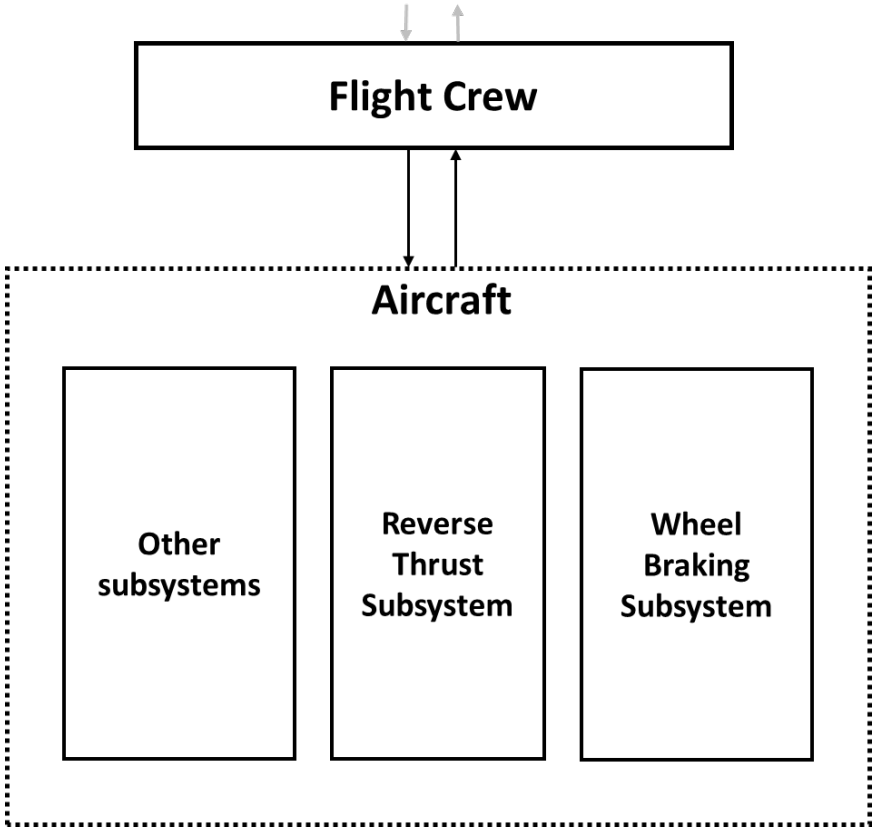


Figure 2.9: Example of a refined control structure with subsystems

Once subsystems have been identified, we can refine the control structure by defining how they will be controlled. For the purposes of this example, let's refine the wheel braking subsystem. Will the wheel braking subsystem be controlled directly and manually by the flight crew only? Will control units, automated controllers, or other humans exist that can also control the brakes? If STPA is being applied during later development stages, these answers may be known and obvious. If STPA is being applied during early concept development, then the hazards and the constraints above can be used to guide these decisions. For example, we may include an automated brake controller in the wheel braking subsystem to automatically brake upon landing or rejected takeoff.

Figure 2.10 shows the refined control structure including a Brake System Control Unit (BSCU) within the wheel braking subsystem. Notice that we are carefully "zooming in" to add more detail in the control structure.

제약사항을 식별하는 데 사용될 수 있다. 그런 다음 STPA 결과를 사용하여 아키텍처, 예비 설계 및 세부 설계를 도출하고 구현 결정을 내리며, 컨트롤 스트럭처를 상세화 할 수 있다. 세부 사항을 알고 설계를 결정한 경우라고 하더라도 상세한 컨트롤 스트럭처 모델을 분석하기 전에, 먼저 보다 높은 추상 수준에서 STPA를 적용하는 것이 더 빠른 결과를 도출하고 광범위한 이슈를 도출하는 데 도움이 될 수 있다.

컨트롤 스트럭처 모델링

컨트롤 스트럭처 모델링은 추상 컨트롤 스트럭처에서 시작하여 반복적으로 세부 사항을 추가한다. 대부분의 경우 시스템 내의 컨트롤 스트럭처와 컨트롤 루프가 명백하거나, 이전 어플리케이션을 재사용할 수 있다. 여기서는 컨트롤 스트럭처가 명확하지 않은 경우에 기능적인 컨트롤 스트럭처를 모델링하고 세부 사항을 추가하는 한 가지 방법을 설명한다. 부록 B는 다른 산업에서 사용된 컨트롤 스트럭처의 예를 다룬다.

시작하는 한 가지 방법은 안전 제약사항을 부여하고 이전에 식별된 위험을 방지하기 위해 필요한 기본 서브시스템을 식별하는 것이다. 예를 들어 우리는 불충분한 감속과 관련된 위험과 안전 제약사항을 도출했다. 이러한 안전 제약사항은 휠 브레이크 서브시스템, 역방향 추진력(reverse thrust) 및 다른 서브시스템을 사용하여 적용할 수 있다. *그림 2.9*는 이들 서브시스템을 사용한 초기 컨트롤 스트럭처를 나타낸다.

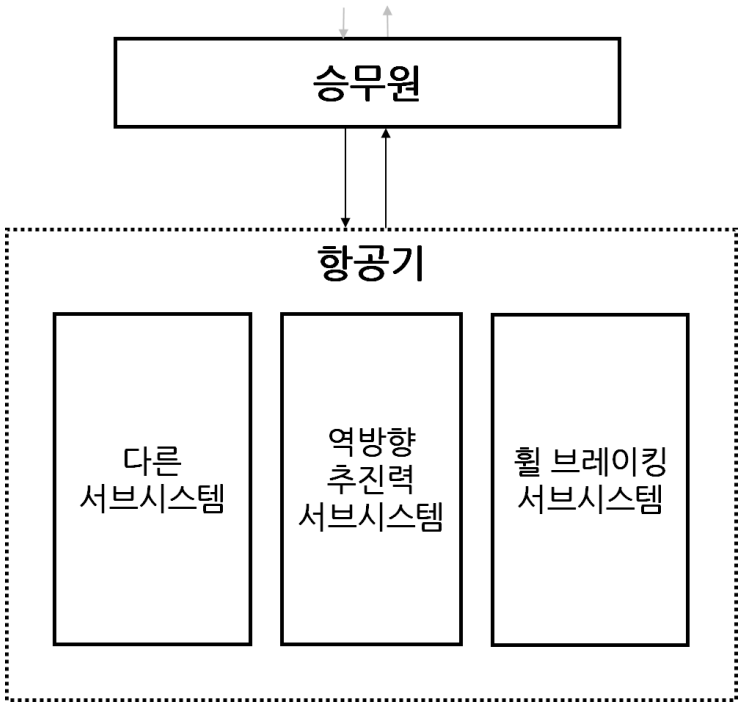


그림 2.9: 서브시스템이 포함된 컨트롤 스트럭처 상세화 예시

일단 서브시스템이 식별되면 그들이 어떻게 제어되는지 정의하여 컨트롤 스트럭처를 상세화할 수 있다. 이 예제를 대상으로 휠 브레이킹 서브시스템(Wheel Braking Subsystem, WBS)을 상세화해 보자. 휠 브레이킹 서브시스템은 승무원에 의해서만 직접 수동으로 제어되는가? 컨트롤 유닛(control unit), 자동화 컨트롤러(automated controller) 또는 다른 사람도 브레이크를 제어할 수 있는가? STPA가 개발

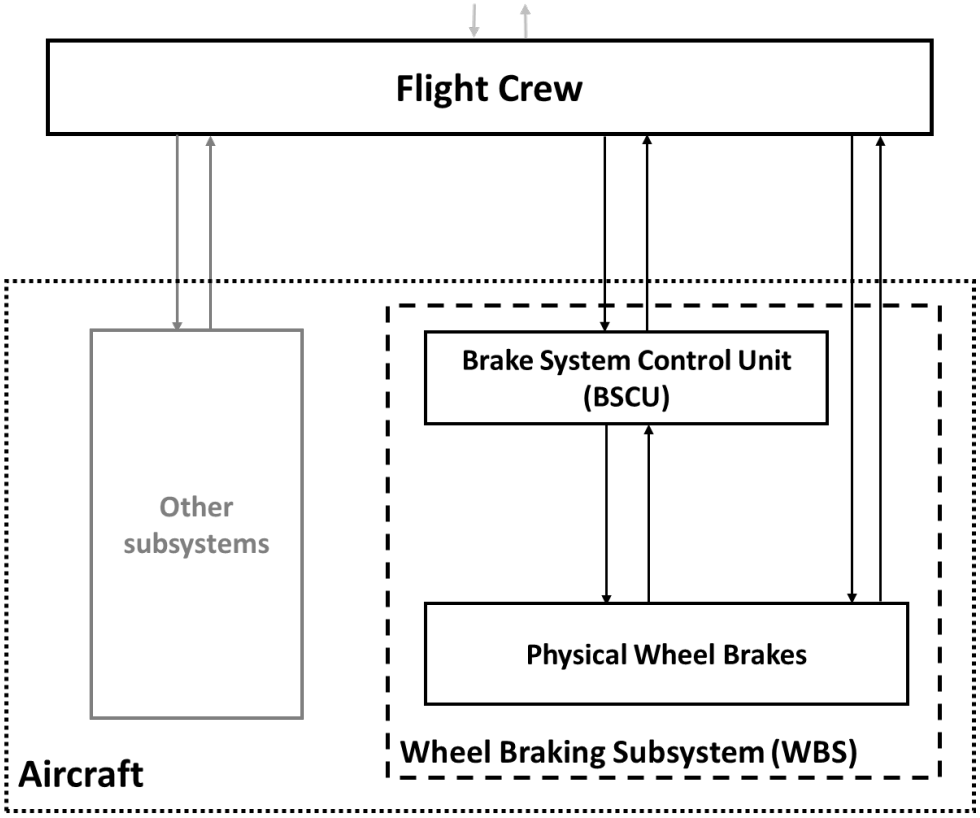


Figure 2.10: Refined control structure with subsystem controllers

Once the controllers have been identified, system engineers can assign responsibilities. These responsibilities are a refinement of the safety constraints—what does each entity need to do so that *together* the safety constraints will be enforced? For example, the BSCU might be responsible for automatically pulsing the brakes when braking causes a skid (anti-skid function) while the flight crew may be responsible for deciding when the brakes need to be applied.

Example responsibilities related to wheel braking:

Physical Wheel Brakes
- R-1: Decelerate wheels when commanded by BSCU or Flight Crew [SC-6.1]

BSCU
- R-2: Actuate brakes when requested by flight crew [SC-6.1]
- R-3: Pulse brakes in case of a skid (Anti-skid) [SC-6.2]
- R-4: Automatically engage brakes on landing or rejected takeoff (Autobrake) [SC-6.1]

Flight crew
- R-5: Decide when braking is needed [SC-6.1, SC-6.3]
- R-6: Decide how braking will be done: Autobrake, normal braking, or manual braking [SC-6.1]
- R-7: Configure BSCU and Autobrake to prepare for braking [SC-6.1]
- R-8: Monitor braking and disable BSCU, manually brake in case of malfunction [SC-6.1, SC-6.2]

Next, control actions for each controller can be defined based on these responsibilities. For example, the flight crew will need the capability to send manual braking control actions to satisfy R-5 and R-6. They will need a way to arm and set the BSCU to satisfy R-6 and R-7. They may need to disarm the BSCU

단계 중 후반부에 적용될 경우 이러한 질문의 답은 이미 나와 있으며 명백할 수도 있다. 초기 개념 개발 단계에 STPA가 적용되는 경우에는 위의 위험과 안전 제약사항을 사용하여 이러한 것들에 대한 결정을 유도할 수 있다. 예를 들어 착륙이나 이륙중지 상황에서 자동으로 브레이크를 걸어 주기 위하여 자동 브레이크 컨트롤러를 휠 브레이킹 서브시스템에 포함시킬 수 있다.

*그림2.10*은 휠 브레이킹 서브시스템 내에 브레이크 시스템 컨트롤 유닛(BSCU)을 포함한 상세화된 컨트롤 스트럭처를 보여준다. 컨트롤 스트럭처에 세부 사항을 추가하려면 조심스럽게 "파고들어야(zooming in)" 한다.

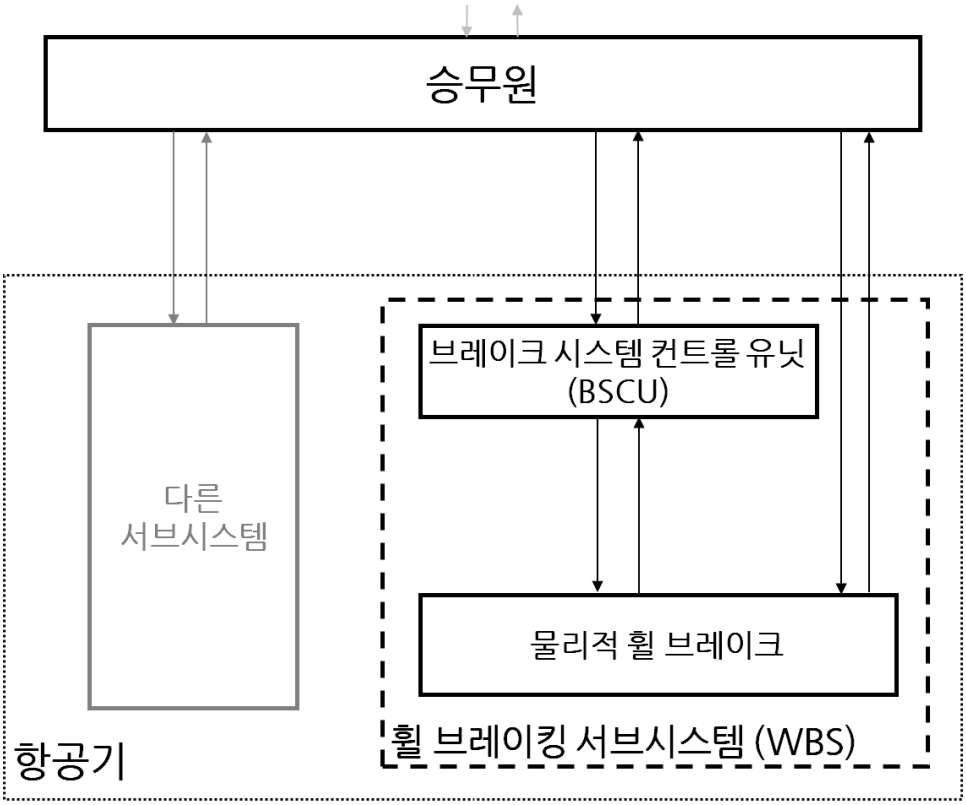


그림2.10: 서브시스템 컨트롤러를 추가하여 상세화한 컨트롤 스트럭처

컨트롤러가 식별되면 시스템 엔지니어는 책임(responsibility)을 부여할 수 있다. 이러한 책임은 안전 제약사항을 상세히 하는 것이다. — 모든 안전 제약사항이 어우러져 적용될 수 있도록 각 개체들은 무엇을 해야 하는가? 예를 들어, 브레이크로 인해 미끄럼(skid)이 발생했을 때 BSCU는 자동으로 브레이크를 펄싱(pulsing)할 책임이 있으며(미끄럼 방지 기능), 승무원은 브레이크 적용 시기를 결정할 책임이 있다.

휠 제동과 관련된 책임의 예:

물리적 휠 브레이크

- R-1: BSCU 또는 승무원이 명령할 때 바퀴를 감속시킨다 [SC-6.1]

BSCU

- R-2: 승무원의 요청에 따라 브레이크 작동시킨다 [SC-6.1]
- R-3: 미끄럼 발생 시 브레이크를 펄스(purse)한다(미끄럼 방지) [SC-6.2]
- R-4: 착륙 또는 이륙중지 시 자동으로 브레이크를 작동시킨다(오토브레이크) [SC-6.1]

to satisfy R-8. *Figure 2.11* shows a revised control structure with labeled control actions based on the responsibilities.

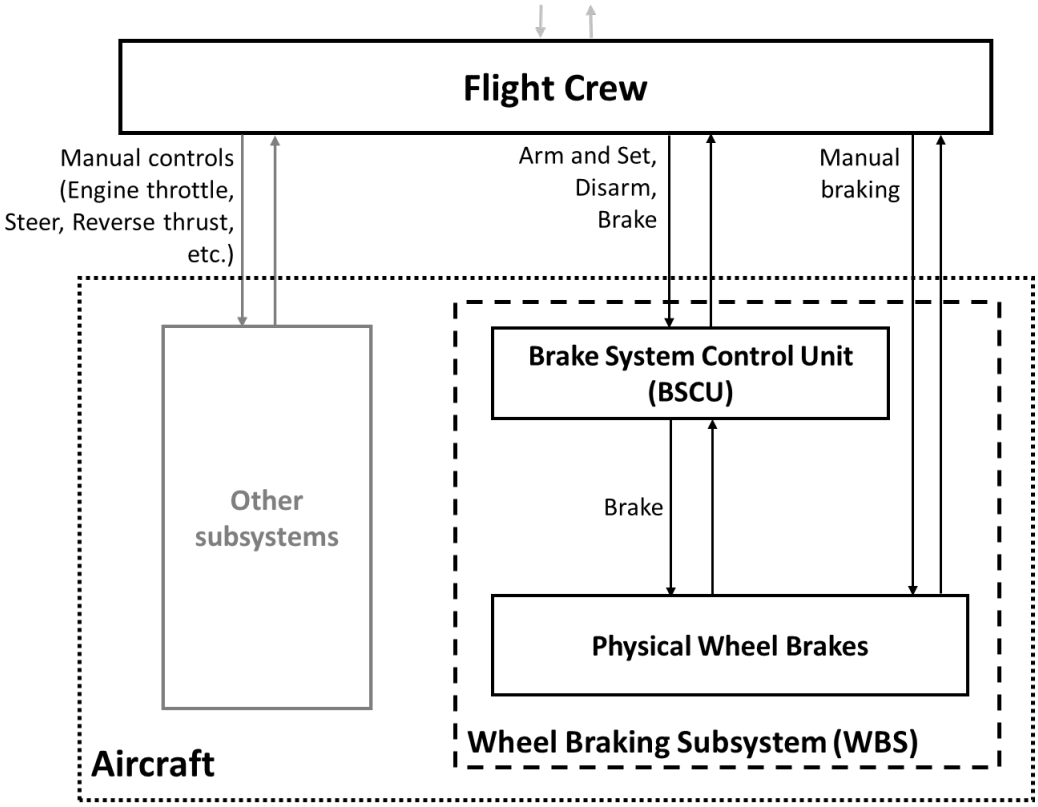


Figure 2.11: Refined control structure after allocation of processes to subsystems

At this point, the controllers and control actions have been identified and labeled but what about feedback? Feedback can be derived from the control actions and responsibilities by first identifying the process models that controllers will need to make decisions. Then, feedback and other information needed to form accurate process models can be identified. For example, R-3 specifies that the BSCU will need to pulse the brakes in case of a skid. To do this, the BSCU will need to know that a skid is occurring (information that should be included in the BSCU's process model). What feedback is needed to detect a skid? Wheel speed feedback could be used. Similarly, R-8 specifies that the flight crew may need to disable the BSCU in case of a malfunction. To do this, the flight crew will need to know that the BSCU is malfunctioning (again, information that should be in the process model). What feedback can the flight crew use to detect a malfunction? Feedback about BSCU faults could be provided. To automatically engage the brakes upon landing or RTO (R-4), the BSCU will need to know when the aircraft has landed or when RTO is occurring (should be in the process model). Weight on wheels switches and other inputs could be used to detect landing and RTO conditions. *Table 2.2* below shows how feedback can be derived from the responsibilities.

승무원

- R-5: 브레이크가 필요할 때를 결정한다 [SC-6.1, SC-6.3]
- R-6: 브레이킹 방식을 결정한다: 오토브레이크, 노멀 브레이킹, 매뉴얼 브레이킹 [SC-6.1]
- R-7: 브레이킹을 준비하기 위해 BSCU 및 오토브레이크를 설정한다 [SC-6.1]
- R-8: 브레이크와 BSCU의 비활성화(disable)를 모니터하고, 오작동 시 수동으로 브레이크를 작동시킨다 [SC-6.1, SC-6.2]

그 다음, 이러한 책임을 기반으로 각 컨트롤러에 대한 컨트롤 액션을 정의할 수 있다. 예를 들어 승무원은 R-5 및 R-6을 충족시키기 위해 매뉴얼 브레이킹 컨트롤 액션을 보낼 수 있는 능력이 필요하다. 승무원은 R-6 및 R-7을 충족시키기 위해 BSCU를 작동개시 및 설정하는(arm and set) 방법이 필요하다. 승무원은 R-8을 충족시키기 위해 BSCU를 작동해제해야(disarm) 할 수도 있다. *그림 2.11*은 책임에 기반한 컨트롤 액션을 표기하여 수정한 컨트롤 스트럭처를 보여준다.

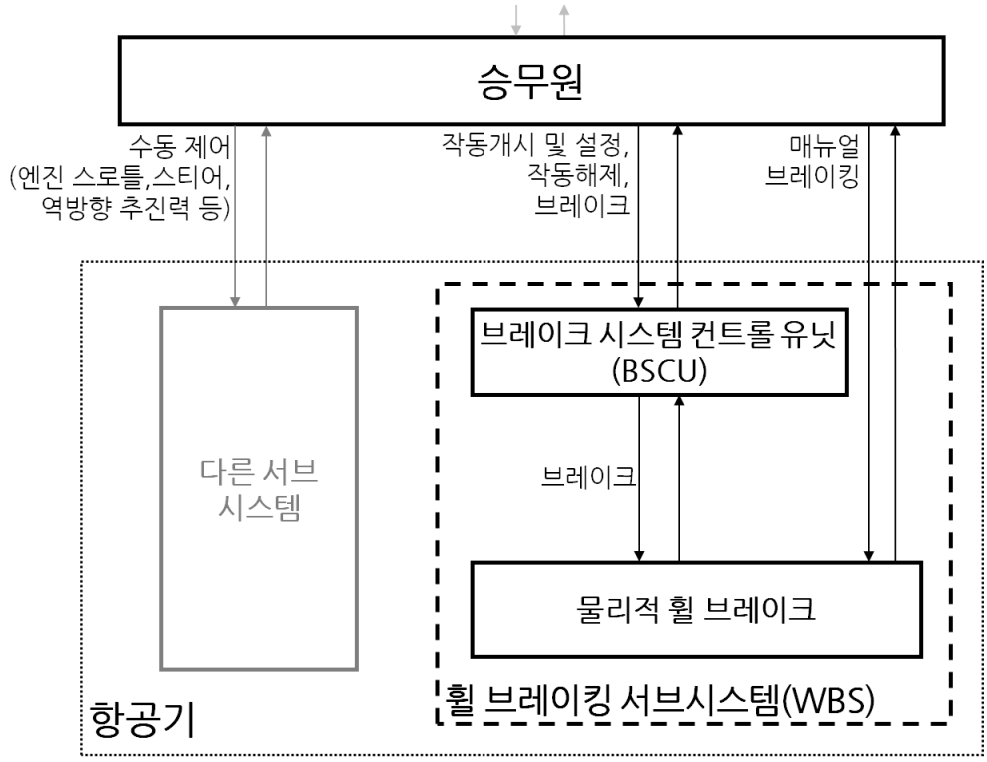


그림 2.11: 서브시스템에 프로세스를 할당한 후 상세화한 컨트롤 스트럭처

이 시점에서 컨트롤러와 컨트롤 액션을 식별 및 표기하였지만, 피드백은 어떻게 식별하고 표기하여야 할까? 컨트롤러가 의사 결정을 내리는 데 필요한 프로세스 모델을 먼저 식별한 후 컨트롤 액션 및 책임으로부터 피드백을 도출할 수 있다. 그런 다음, 정확한 프로세스 모델을 형성하는 데 필요한 피드백 및 다른 정보(information)가 식별될 수 있다. 예를 들어 R-3은 미끄럼이 발생했을 때 BSCU가 브레이크를 펄스할 필요가 있음을 나타낸다. 이를 위해 BSCU는 미끄럼이 발생하고 있음(BSCU의 프로세스 모델에 포함되어야 하는 정보임)을 알아야 한다. 미끄럼을 탐지하려면 어떤 피드백이 필요할까? 휠 속도 피드백이 사용될 수 있다. 마찬가지로 R-8은 고장이 났을 때 승무원이 BSCU를 비활성화해야 할 수도 있음을 나타낸다. 이를 위해 승무원은 BSCU가 오작동하고 있음(마찬가지로 프로세스 모델 내에 있어야 하는 정보임)을 알아야 한다. 승무원이 오작동을 탐지하기 위해 사용할 수 있는 피드백은 무엇일까?

Table 2.2: Examples showing how feedback can be derived from responsibilities

BSCU Responsibility	Process Model	Feedback
Actuate brakes when requested by flight crew [SC-6.1]	Braking is requested by flight crew	Brake pedal applied
Pulse brakes in case of a skid (Anti-skid) [SC-6.2]	Aircraft is skidding	Wheel speeds Inertial reference unit
Automatically engage brakes on landing or RTO (Autobrake) [SC-6.1]	Aircraft landed Takeoff is rejected	Weight on wheels Throttle lever angle

The control structure can be refined further by using the responsibilities to "zoom in" again and add additional details. For example, the physical wheel brakes are responsible for decelerating the wheels on command (R-1). This could be done with hydraulics. R-6 indicates that the BSCU will need to execute both normal and automatic braking (Autobrake). Two controllers within the BSCU could be used to control these two behaviors: an autobrake controller and a hydraulic controller.

Figure 2.12 shows a refined control structure with feedback labeled and with BSCU internal controllers identified.

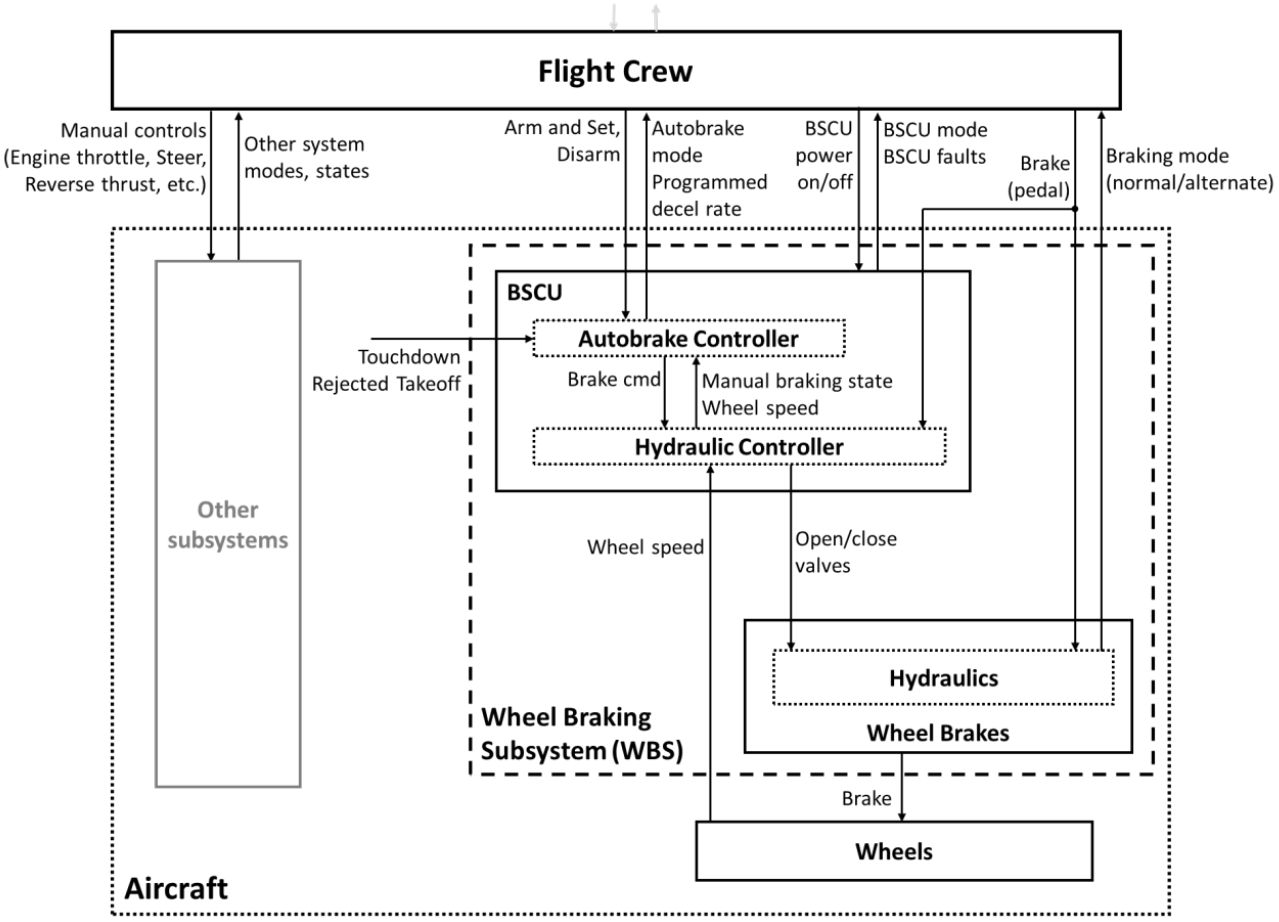


Figure 2.12: Control structure after refinement based on the responsibilities

BSCU 결함에 대한 피드백이 있을 수 있다. 착륙 또는 이륙중지 시(R-4) 자동으로 브레이크를 작동시키기 위해서 BSCU는 항공기가 착륙했을 때 또는 이륙중지가 발생한 시점(프로세스 모델에 포함되어야 함)을 알아야 한다. 착륙 감지(Weight-on-wheels) 스위치나 다른 입력값으로 착륙이나 이륙중지 상태를 탐지할 수 있다. 아래 *표 2.2*는 피드백이 책임으로부터 어떻게 도출될 수 있는 지를 보여준다.

표 2.2: 피드백이 책임으로부터 도출되는 방법에 대한 사례

BSCU 책임	프로세스 모델	피드백
승무원 요청에 따라 브레이크를 작동시킨다 [SC-6.1]	승무원에 의해 브레이크가 요청된다	브레이크 페달 적용
미끄럼 발생 시 브레이크를 펄스한다(미끄럼 방지) [SC-6.2]	항공기가 미끄러지고 있다	바퀴속도 관성참조장치(Internal Reference Unit)
착륙 또는 이륙중지 시 자동으로 브레이크를 건다(Autobrake) [SC-6.1]	항공기가 착륙했다 이륙이 중지되었다	착륙감지(Weight on wheels) 스로틀 레버 각도

컨트롤 스트럭처를 책임을 사용하여 다시 “확대(zoom in)”하고 세부 사항을 추가하여 더욱 상세화할 수 있다. 예를 들어 물리적인 휠 브레이크는 명령(R-1)에 따라 바퀴를 감속시킬 책임이 있다. 이것은 유압 장치로 할 수 있다. R-6은 BSCU가 일반적 및 자동적 브레이킹(오토브레이크)을 모두 실행해야 함을 나타낸다. 이러한 두 가지 동작을 제어하기 위하여 BSCU 내의 두 컨트롤러(오토브레이크 컨트롤러, 유압 컨트롤러)를 사용할 수 있다.

*그림 2.12*는 BSCU 내부 컨트롤러를 식별하고 피드백을 표기한 상세화된 컨트롤 스트럭처를 보여준다.

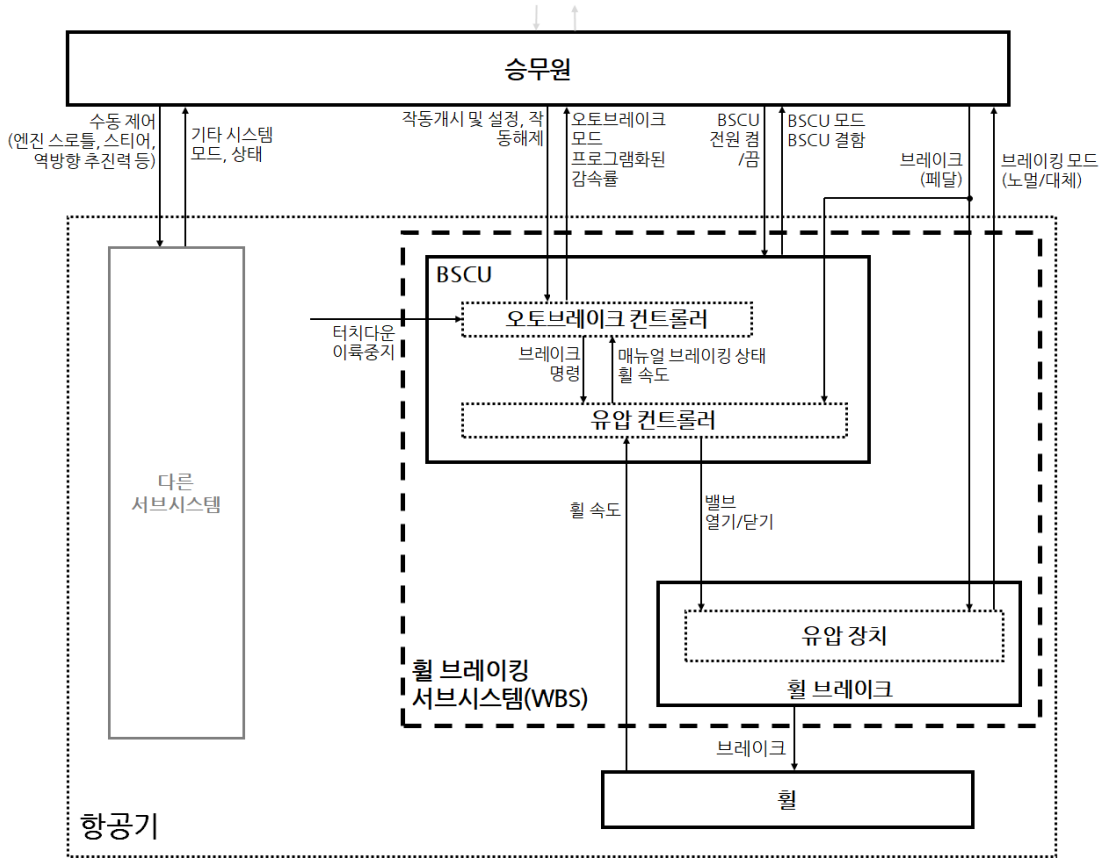


그림 2.12: 책임을 기반으로 상세화한 컨트롤 스트럭처

Common questions when modeling the control structure

Does the control structure need to be complete before proceeding?

If STPA is being applied early before development is finished, some information may not be known and the control structure may be incomplete. The analysis can begin with an incomplete control structure and STPA will help identify potentially missing feedback, controls, and other gaps so the control structure can be refined in parallel with development. The bare minimum needed to begin the next steps is at least one controller, control action, and controlled process. However, in general STPA will be easier and more efficient if relevant information is not intentionally missing from the control structure.

If STPA is being applied during later development stages after decisions about controllers, control actions, and feedback have already been made, then there is no reason to intentionally omit high-level information from the control structure. STPA can be applied to the design as it exists to identify potential flaws like critical feedback that was overlooked.

How specific should the arrow labels be?

Avoid using ambiguous and vague labels in the control structure, such as labeling all control actions as simply "Commands", all feedback as simply "Feedback" or "Status", and controllers as simply "Computer" or "Controller." Labels for control actions should indicate the type of command (if known), like "Open/close valves" and feedback should indicate the type of information being sent (if known), like "Wheel speed". The specific physical medium used to send control actions and feedback is irrelevant at this point—what matters is the *functional* information that can be sent. For example, use "BSCU power on/off" rather than "BSCU button", "Shared bus", or "Encoded digital packets." Similarly, labels for controllers should indicate the functional type of behavior or the role of the controller, not the physical implementation. For example, use controller labels like "Autobrake Controller" rather than "Single Board Computer," and use labels like "Flight Crew" or "Pilot" whether the piloting function is accomplished on board or on the ground.

Should the control structure include all actuators and sensors?

Specific actuators and sensors are not needed to begin STPA and do not need to be included yet (they will be included in a later step). To help manage complexity, it can be helpful to wait until these details need to be considered during the scenario creation step later in STPA. For example, the wheel braking subsystem control actions and feedback in *Figure 2.12* may be implemented with different kinds of electromechanical valves or by using some other completely different implementation. What matters at this phase are the types of commands and feedback that might be provided, not the specific implementation (which may or may not be known).

How do physical processes and physical interactions fit into the control structure?

Like any model, a control structure model emphasizes certain aspects of the real world while abstracting or de-emphasizing others. A control structure will emphasize functional relationships and functional interactions, which is very useful for identifying problems like design flaws, requirements flaws, human error, software errors, and even traditional physical component failures. A control structure model does not typically capture purely physical or geometric relationships like physical proximity between components or fire propagation.

The physical processes being controlled are typically specified at the lowest level of the control structure while every level above specifies functional controllers that make decisions and directly or indirectly control the physical processes.

컨트롤 스트럭처를 모델링할 때 자주 나오는 질문

진행하기 전에 컨트롤 스트럭처를 완성해야 하는가?

개발이 완료되기 전에 STPA가 일찍 적용되는 경우, 일부 정보를 알 수 없으며 컨트롤 스트럭처가 불완전할 수 있지만, 불완전한 컨트롤 스트럭처에서도 분석을 시작할 수 있다. STPA는 잠재적으로 누락된 피드백, 컨트롤 및 다른 차이점들을 식별하여, 개발과정과 병행하여 컨트롤 스트럭처를 상세화하는데 도움을 준다. 다음 단계를 시작하기 위해서는 최소한 하나의 컨트롤러와 컨트롤 액션 그리고 컨트롤드 프로세스가 필요하다. 그러나 일반적으로 관련 정보가 컨트롤 스트럭처에서 의도적으로 누락되지 않았다면 STPA는 더 쉽고 효율적일 것이다.

만약 컨트롤러, 컨트롤 액션 및 피드백에 대한 결정이 이루어진 개발 단계 후기에 STPA가 적용되는 경우에는 컨트롤 스트럭처에 의도적으로 상위 수준의 정보를 누락시킬 이유가 없다. 간과되었던 중요 피드백과 같은 잠재적 결함을 식별하기 위해 STPA를 설계에 적용할 수 있다.

화살표의 명칭은 얼마나 구체적이어야 할까?

컨트롤 스트럭처에서 모호하고 애매한 명칭의 사용은 피해야 한다. 예를 들어 모든 컨트롤 액션을 단순히 "명령"으로 표시하거나, 모든 피드백을 간단히 "피드백" 또는 "상태"로 표시하고 컨트롤러는 단순히 "컴퓨터" 또는 "컨트롤러"로 지정하는 경우 등이다. 컨트롤 액션의 명칭은(알고 있다면) "밸브 열기/닫기"와 같은 명령 유형을 나타내야 하고, 피드백은(알고 있다면) "휠 속도"와 같이 전송되는 정보 유형을 나타내야 한다. 컨트롤 액션과 피드백을 전송하는 데 사용되는 특정 물리적 매체는 이 시점에서 중요하지 않다 — 중요한 것은 전송될 수 있는 기능적 정보이다. 예를 들어 "BSCU 버튼", "공유 버스" 또는 "인코딩된 디지털 패킷" 보다는 "BSCU 전원 켜기/끄기"를 사용해야 한다. 마찬가지로 컨트롤러의 명칭은 물리적으로 구현된 것이 아니라 기능적 작동 유형이나 컨트롤러의 역할을 나타내야 한다. 예를 들어 "Single Board Computer" 보다는 "Autobrake Controller"와 같은 컨트롤러 명칭을 사용하고, "승무원" 또는 "조종사"와 같은 명칭을 사용하여 조종 기능이 기내(on board) 또는 지상(on the ground) 중 어디에서 수행되는지 알 수 있도록 한다.

컨트롤 스트럭처에 모든 액추에이터와 센서가 포함되어야 할까?

특정 액추에이터 및 센서는 STPA를 시작하는 데 필요하지 않으며 아직 포함할 필요가 없다(이후 단계에서 포함될 예정임). 복잡도를 관리하는 데 도움이 되도록 이후 STPA의 시나리오 생성 단계에서 이러한 세부 사항을 고려해야 할 때까지 기다리는 것이 좋다. 예를 들면, *그림2.12* 휠 브레이킹 서브시스템의 컨트롤 액션 및 피드백은 다른 종류의 전기 기계 밸브나 전혀 다른 것을 사용하여 구현될 수 있다. 이 단계에서 중요한 것은 제공될 수 있는 명령 및 피드백의 형태이며(알려지지 않았거나 알 수 없는), 구체적인 구현물이 아니다.

물리적인 프로세스와 물리적인 상호작용이 어떻게 컨트롤 스트럭처에 들어 맞는가?

다른 모델과 마찬가지로 컨트롤 스트럭처 모델은 실제 세계에 있는 특정 측면을 강조하고, 그 외의 것들은 추상화하거나 강조하지 않는다. 컨트롤 스트럭처는 기능적 관계와 기능적 상호작용을 강조하며 이는 설계 결함, 요구사항 결함, 사람의 실수, 소프트웨어 오류 및 기존의 물리적 컴포넌트 오류와 같은 문제를 식별하는 데 매우 유용하다. 컨트롤 스트럭처 모델은 일반적으로 컴포넌트 간의 물리적 인접성 또는 화재 확산과 같은 오직 물리적 또는 기하학적 관계만을 담지는 않는다.

제어되는 물리적 프로세스는 일반적으로 컨트롤 스트럭처의 최하위 레벨에서 명세되는 반면, 최하위 이상의 모든 레벨에서는 물리적인 프로세스를 직·간접적으로 제어하고 의사 결정을 하는 기능 컨트롤러

Does the control structure require a linear hierarchy?

No. For some systems, the control structure may appear similar to a ladder with a clear vertical linear hierarchy. Control structures for other systems may include multiple controllers at the same vertical level that do not control each other. Controllers may all control the same physical process, or they may control different processes.

Controllers at the same level may also communicate with each other outside of a control/feedback relationship. Such communication may be represented by horizontal arrows in the control structure diagram. In general, the interactions (arrows) in a control structure can represent any of three categories: Control actions, feedback, and other information.

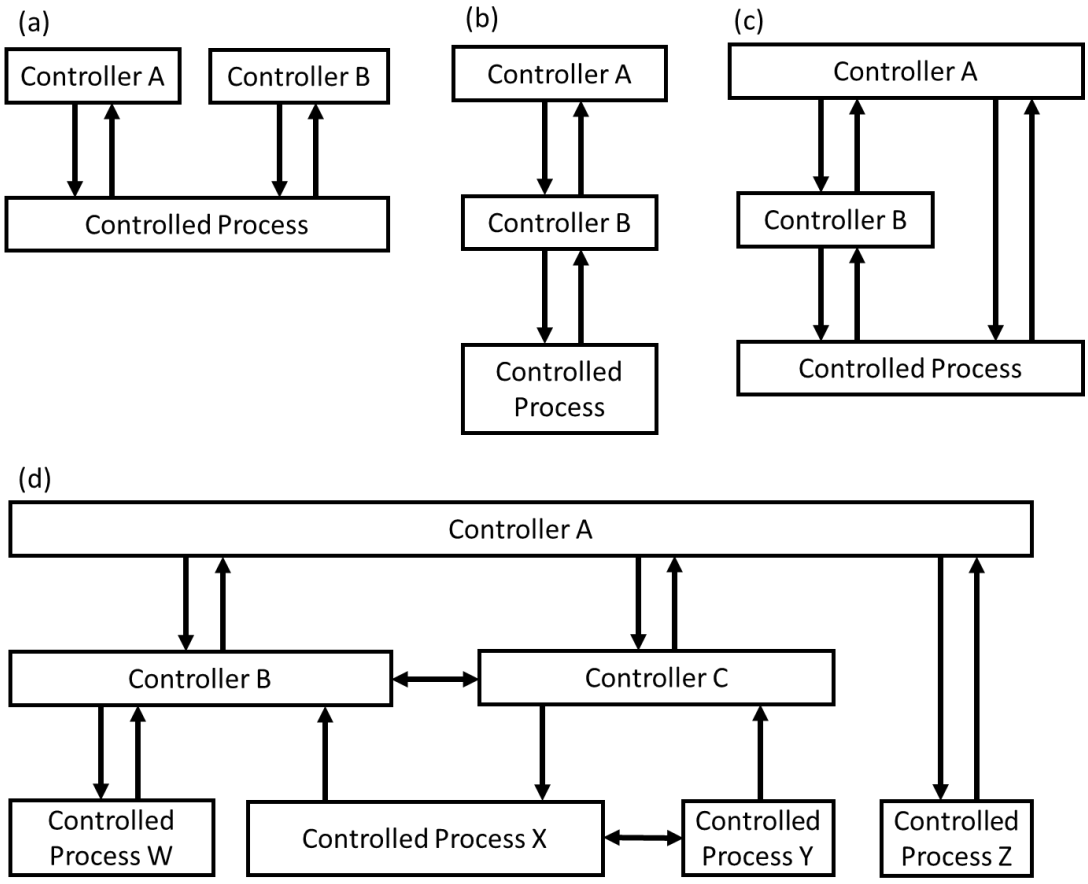


Figure 2.13: Different types of control structures

Figure 2.13 shows a few different types of control structure models, each with different advantages and tradeoffs to consider when designing or analyzing a system. The control structure in (a) shows two controllers operating in parallel to control and monitor a shared process. The controllers do not interact with each other and neither one can control the other (which may lead to loss scenarios that STPA will identify). This type of structure could be advantageous if controllers will need to act quickly or independently of each other—for example, it might be advantageous to provide several operators with the capability to directly abort an operation whenever an unsafe condition is observed without going through a chain of command. However, this structure can present coordination challenges like diffusion of responsibility and assumptions about the actions of other controllers (later steps will analyze these and other problems in more detail).

The control structure in (b) shows a linear control hierarchy such as a clear chain of command. This structure may help address the coordination issue and clarify the difference in responsibilities for each

를 명세한다.

컨트롤 스트럭처가 선형 계층 구조일 필요가 있는가?

아니다. 일부 시스템에서는 컨트롤 스트럭처가 명확한 수직 선형 계층구조를 가진 사다리와 비슷하게 보일 수 있다. 하지만, 다른 시스템의 컨트롤 스트럭처에는 서로를 제어하지 않는, 수직적으로 동일한 수준의 다중 컨트롤러가 있을 수 있다. 또한, 컨트롤러는 동일한 물리적 프로세스를 모두 제어하거나 서로 다른 프로세스를 제어할 수 있다.

동일한 레벨의 컨트롤러는 컨트롤/피드백 관계를 벗어나서 서로 통신할 수도 있다. 이러한 통신은 컨트롤 스트럭처 다이어그램에서 수평 화살표로 표시될 수 있다. 일반적으로 컨트롤 스트럭처의 상호작용(화살표)은 컨트롤 액션, 피드백 및 기타 정보의 세 가지 범주 중 하나를 나타낸다.

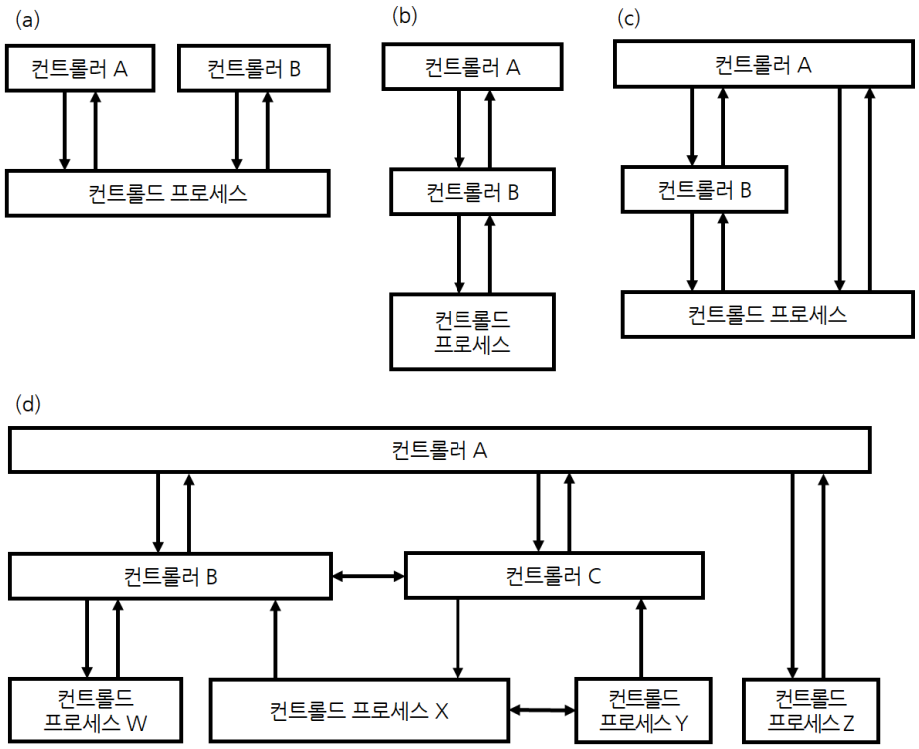


그림 2.13: 컨트롤 스트럭처의 다양한 종류

*그림 2.13*은 시스템을 설계하거나 분석할 때 고려해야 할 서로 다른 장점과 트레이드오프(tradeoff)를 가진 몇 가지 유형의 컨트롤 스트럭처 모델을 보여준다. (a)의 컨트롤 스트럭처는 두 개의 컨트롤러가 공유 프로세스를 제어하고 모니터하기 위해 병렬로 작동하는 구조를 보여준다. 컨트롤러는 서로 상호작용하지 않으며 어떤 컨트롤러도 다른 컨트롤러를 제어할 수 없다(이로 인해 STPA가 식별하게 될 손실 시나리오가 발생할 수 있음). 이러한 유형의 구조는 컨트롤러가 서로 신속하게 또는 독립적으로 작동해야 하는 경우에 유용할 수 있다 — 예를 들면 명령 체계를 거치는 것이 아니라, 안전하지 않은(unsafe) 상태가 관찰될 때마다 여러 운영자가 직접 작업을 중단할 수 있는 능력을 제공하는 경우에 유용할 수 있다. 하지만 이 구조는 책임 배분과 다른 컨트롤러의 동작에 대한 가정과 같은 조정(coordination) 측면의 과제가 생길 수 있다(이후 단계에서는 이러한 문제와 기타 문제를 보다 자세히 분석한다).

(b)의 컨트롤 스트럭처는 명확한 명령 체계와 같은 선형 제어 계층을 보여준다. 이 구조는 조정 이

controller, but may present challenges in terms of reaction time for Controller A's feedback and control actions and is more dependent on the operation of Controller B.

Control actions and feedback may "skip" over levels—there is no model limitation that prevents controllers from interacting beyond one level above and below. For example, high-level controllers may be able bypass other controllers and directly control or monitor lower-level process as in (c). Controller A could represent pilots that normally use automation in Controller B to actuate the brakes. However, in an emergency the pilots may be able to bypass Controller B and directly (manually) actuate the brakes.

More complex structures like (d) are also possible. There is no requirement for a 1-to-1 mapping between controllers and processes. A controller may control one or more processes, and a process may be controlled by zero or more controllers. Typically, each control action path will be paired with a parallel feedback path but not always. For example, Controller C may not be able to control Process Y directly. Instead, it may control Process X and monitor the effect of X on Process Y.

How do I figure out who controls who?

In many cases the control relationship is simple and obvious, such as a supervisor giving instructions to a subordinate or a computer sending commands to open/close a valve. Control relationships can also be less direct, such as managers assigning priorities, airlines providing standard operating procedures, etc. These are all different forms of control.

As mentioned earlier, control is different from obedience. Just because a control relationship exists does not mean that the control actions will always be executed and performed adequately. In fact, we may not even want all control actions to be obeyed all the time; control actions may potentially be disregarded for good reasons when required to prevent a loss in some unexpected situation. STPA does not assume obedience and these problems will be carefully examined in later steps.

Similarly, the ability to trigger a response or influence behavior of another component does not automatically imply control. A control action to open a valve may trigger the valve to open, but conversely a feedback signal indicating high temperature may trigger a controller to turn on a fan. The ability to influence or respond is not sufficient to distinguish control actions from feedback.

Control involves making purposeful decisions to achieve goals. Feedback can be provided without being aware of (or responsible for) a particular higher-level goal, but control actions are always provided to achieve a goal. Control also typically involves supervision—monitoring lower-level entities, having better capability or information to evaluate their behaviors or impacts, and guiding them or intervening to ensure higher-level goals are achieved.

The control hierarchy is closely related to the hierarchy of goals and responsibilities. The automated BSCU controller above may have very narrow responsibilities related to triggering brakes when a landing is detected. The flight crew has higher-level responsibilities like deciding when to land and higher-level goals like avoiding a rough landing. The BSCU may be a necessary but insufficient element in achieving these higher-level flight crew goals—the flight crew must supervise the BSCU and many other elements to achieve the higher-level goals. Similarly, air traffic control has even higher-level responsibilities like ensuring minimum separation between multiple aircraft and higher-level goals like maximizing throughput across many flights. A single flight crew is only one element used by air traffic control to achieve the goals, and the flight crew might not even be aware of how they are contributing to those higher-level goals or if those higher-level goals are being achieved by air traffic control.

Finally, control is closely related to authority. Consider the relationship between air traffic control and pilots. Air traffic control has authority over pilots and provides instructions and clearances that pilots generally must obey. Pilots do not have authority over air traffic control, and they cannot simply order air traffic control to obey. Pilots *can* declare an emergency and air traffic control will need to respond appropriately to that feedback—just because pilots can trigger a response does not imply

슈를 해결하고 각 컨트롤러에 대한 책임의 차이를 명확히 하는 데 도움이 되지만, 컨트롤러 A의 피드백 및 컨트롤 액션에 대한 반응 시간 측면의 어려움이 있을 수 있으며 컨트롤러 B의 동작에 더 의존하게 된다.

컨트롤 액션과 피드백은 레벨을 "건너뛸 수도 있다" — 모델에서 컨트롤러가 위와 아래의 한 레벨을 넘어가며 상호작용하지 못하게 하는 제한은 없다. 예를 들어 상위 컨트롤러는 다른 컨트롤러를 우회(bypass)하여 (c)에서와 같이 하위 단계 프로세스를 직접 제어하거나 감시할 수 있다. 컨트롤러 A는 조종사를 나타내며 일반적으로 컨트롤러 B의 자동화 장치를 사용하여 브레이크를 작동시킬 수 있다. 그러나 비상 시 조종사는 컨트롤러 B를 우회하여 직접(수동으로) 브레이크를 작동시킬 수 있다.

(d)와 같은 더 복잡한 구조도 가능하다. 컨트롤러와 프로세스 간에 일대일 매핑이 되어야 하는 것은 아니다. 하나의 컨트롤러는 하나 이상의 프로세스를 제어할 수 있으며 프로세스는 0개 또는 그 이상의 컨트롤러에 의해 제어될 수 있다. 일반적으로 각 컨트롤 액션 경로는 피드백 경로와 평행하게 쌍을 이루지만 항상 그런 것은 아니다. 예를 들어 컨트롤러 C는 프로세스 Y를 직접 제어하지 못할 수 있다. 대신 프로세스 X를 제어하고 X가 Y에게 준 영향을 모니터 할 수 있다.

누가 누구를 컨트롤하는지 어떻게 알 수 있을까?

대부분의 경우 컨트롤 관계는 관리자가 밸브를 열거나 닫는 명령을 직원 또는 컴퓨터에 제공하는 것처럼 단순하고 분명하다. 컨트롤 관계는 관리자가 우선순위를 정하거나 항공사가 표준 운영 절차를 제공하는 것과 같이 덜 직접적인 경우도 있다. 이들은 모두 다른 형태의 제어이다.

앞서 언급했듯이 컨트롤이 항상 실행되는 것은 아니다. 즉, 컨트롤 관계가 있다고 해서 컨트롤 액션이 항상 실행되고 적절하게 수행된다는 것을 의미하지는 않는다. 사실 우리는 모든 컨트롤 액션이 항상 실행되기를 원하지 않을 수도 있다; 일부 예상치 못한 상황에서 손실을 예방해야 하는 경우 컨트롤 액션은 타당한 이유로 무시될 수 있다. STPA는 컨트롤 액션이 항상 실행되는 것을 가정하지 않으며 이러한 문제는 이후 단계에서 면밀히 검토하도록 하겠다.

마찬가지로 응답을 트리거하거나 다른 컴포넌트의 동작에 영향을 미치는 기능이 자동적으로 컨트롤을 의미하지는 않는다. 밸브를 열기 위한 컨트롤 액션이 밸브를 열도록 트리거할 수 있지만 고온을 나타내는 피드백 신호에 의해 컨트롤러가 팬을 켜도록 트리거할 수도 있다. 영향을 미치거나 응답하는 기능으로 컨트롤 액션과 피드백을 구별하기는 어렵다(충분하지 않다).

컨트롤은 목표 달성을 위해 목적 있는 결정을 하는 것을 포함한다. 피드백은 특정 상위 수준의 목표를 알지 못해도 제공될 수 있지만 컨트롤 액션은 항상 목표를 달성하기 위해 제공된다. 컨트롤은 전형적으로 감독(supervision)과 관련되어 있다. — 감독은 하위 수준의 개체(entities)를 모니터링하고 개체의 행동이나 영향을 평가하기 위한 더 나은 능력이나 정보를 가지며, 더 높은 수준의 목표를 달성할 수 있도록 가이드하거나 개입하는 것 등을 말한다.

컨트롤 계층은 목표와 책임의 계층구조와 밀접하게 관련되어 있다. 위의 자동화된 BSCU 컨트롤러는 착륙이 탐지될 때 브레이크를 작동시키는 것과 관련하여 매우 제한된 책임을 가질 수 있다. 승무원은 착륙시기를 결정하는 것과 같은 높은 수준의 책임과 매끄럽지 못한 착륙을 막는 것과 같은 높은 수준의 목표를 가진다. BSCU는 필요한 요소이기는 하나 이렇게 높은 수준의 승무원 목표를 달성하기에는 불충분한 요소일 수 있다. 승무원은 더 높은 수준의 목표를 달성하기 위해 BSCU 및 기타 여러 요소를 감독해야 한다. 마찬가지로 ATC(Air Traffic Control)는 항공기 간의 최소 간격을 보장하는 것과 같은 훨씬 높은 수준의 책임과, 여러 항공편에서 처리량을 극대화하는 것과 같은 높은 수준의 목표를 가진다. 한 명의 승무원은 ATC의 목표 달성을 위해 사용되는 하나의 요소일 뿐이며 승무원은 그

control. Pilots can also disobey air traffic control instructions when necessary to ensure safety of the flight, but obedience is also different from control. Pilots do have final and direct authority over their aircraft, but they do not have final authority over ATC and they do not manage the overall airspace. ATC generally controls aircraft indirectly by issuing instructions and clearances, not the other way around. Chapter 5 contains further discussion about control structures and control relationships as they relate to organizational and social analysis.

Although beginners sometimes struggle with these distinctions, with more experience it quickly becomes easy and natural. It can also be helpful to realize that mistakes about who controls whom in the control structure usually do not have a significant impact on the results of the analysis. For example, suppose control action X is mischaracterized as feedback X. Because it is characterized as feedback, the step that identifies unsafe control actions will not consider how a missing or delayed control action X might lead to a hazard. However, the next step examines potential feedback problems and will identify the same scenarios when considering how missing or delayed feedback X might lead to a hazard.

Do I need to document anything other than the control structure diagram?

It is good practice to document any additional information that is helpful to understand the control structure. Additional information about controllers such as a basic description, purpose, special functionality, controller responsibilities, process models, etc. should be documented for a completely specified control structure. Clarifying information about control actions and feedback can also be helpful, particularly when important aspects may not be clear from short labels. The same is true for controlled processes. In general, any important clarifying information or assumptions should be clearly documented along with the control structure diagram.

What should I look for when reviewing a control structure?

The following tips can help find common mistakes in a control structure:

Tips to prevent common mistakes in a control structure

- Ensure labels describe functional information that is sent, not a specific physical implementation.
- Avoid ambiguous and vague labels like simply "Command" or "Feedback" when the type of information is known.
- Check that every controlled physical process is controlled by one or more controllers (not always required, but often indicates a mistake).
- Review responsibilities (including traceability) for conflicts and gaps.
- Check that control actions needed to satisfy the responsibilities are included.
- Check that feedback needed to satisfy the responsibilities is included. (optional if applied early in concept development when feedback is unknown; later steps can identify missing feedback)

Identifying Unsafe Control Actions

Once the control structure has been modeled, the next step (shown in *Figure 2.14*) is to identify Unsafe Control Actions.

들이 어떻게 더 높은 수준의 목표에 기여하고 있는지 또는 ATC에 의해 높은 목표가 달성되고 있는지 알지 못할 수도 있다.

마지막으로 컨트롤은 통제권한(authority)과 밀접한 관련이 있다. ATC와 조종사 간의 관계를 고려해 보자. ATC는 조종사에 대한 통제권한을 가지며 조종사가 일반적으로 반드시 준수해야 하는 지시(instruction)와 승인(clearance)을 내린다. 조종사는 ATC에 대한 통제권한이 없으며 ATC에 명령을 내릴 수 없다. 조종사는 비상 사태를 선언할 수 있고 ATC는 해당 피드백에 적절하게 응답해야 한다. — 왜냐하면 조종사가 응답을 트리거할 수 있다는 것이 제어를 의미하지는 않기 때문이다. 조종사는 비행 안전을 위해서 필요한 경우 ATC의 지시에 불복할 수 있지만 복종과 제어는 다르다. 조종사는 자신의 항공기에 대한 최종적이고 직접적인 통제권한을 가지고 있지만 ATC에 대한 최종 통제권한은 없으며 전체 공역을 관리하지는 않는다. ATC는 일반적으로 지시와 승인을 발행하여 간접적으로 항공기를 제어하며 그 외 다른 제어 방법은 없다. 5장에서는 조직 및 사회 분석과 관련된 컨트롤 스트럭처와 컨트롤 관계에 대해 추가적으로 논의한다.

입문자에게는 이런 구별이 어려울 수도 있지만 더 많은 경험을 통해 곧 쉽고 자연스러워질 것이다. 또한 컨트롤 스트럭처에서 누가 누구를 컨트롤 하는지에 대한 잘못된 이해가 일반적으로 분석 결과에 중요한 영향을 주지 않는다는 것도 깨닫게 된다. 예를 들어 컨트롤 액션 X가 피드백 X로 잘못 판단되었다고 가정해보자. 컨트롤 X가 피드백으로 표시됐기 때문에 언세이프 컨트롤 액션을 식별하는 단계에서 어떻게 컨트롤 액션 X가 누락되거나 지연되어 위험을 일으킬 수 있는지에 대해서는 고려하지 않는다. 그러나 다음 단계에서 잠재적인 피드백 문제를 검토하고 피드백 X가 누락되거나 지연되면 어떻게 위험이 초래될 수 있는지 고려하는 과정에서 동일한 시나리오를 식별하게 될 것이다.

컨트롤 스트럭처 다이어그램 이외의 것을 문서화할 필요가 있을까?

컨트롤 스트럭처를 이해하는 데 도움이 되는 추가 정보를 문서화하는 것이 좋다. 컨트롤 스트럭처를 완전하게 명세하기 위해서는 기본 설명, 목적, 특수한 기능, 컨트롤러 책임, 프로세스 모델 등과 같은 컨트롤러에 대한 추가 정보를 문서화해야 한다. 컨트롤 액션 및 피드백에 대한 정보를 명확히 하는 것이 유용한데, 특히 짧은 명칭을 사용하면 중요한 측면이 명확히 드러나지 않는 경우가 그러하며, 컨트롤드 프로세스에서도 마찬가지이다. 일반적으로 컨트롤 스트럭처 다이어그램과 함께 모든 중요하고 명확한 정보 또는 가정을 분명하게 문서화해야 한다.

컨트롤 스트럭처를 검토할 때 무엇을 봐야 할까?

다음 팁은 컨트롤 스트럭처에서 발생하는 일반적인 실수를 찾는 데 도움이 될 수 있다.

컨트롤 스트럭처의 일반적인 실수를 방지하는 팁

- 명칭(label)이 특정 물리적 구현(implementation)이 아닌, 전송된 기능적 정보를 설명하는지 확인한다.
- 정보의 종류를 알고 있는 경우에는 단순히 “명령” 또는 “피드백”과 같이 모호하고 애매한 명칭을 사용해서는 안 된다.
- 모든 물리적인 컨트롤드 프로세스가 하나 이상의 컨트롤러에 의해 제어되는지 확인한다(항상 필요한 것은 아니지만 종종 실수일 수 있음).
- 충돌(conflicts)이나 차이점(gaps)에 대해 책임(추적성 포함)을 검토한다.
- 책임을 충족시키기 위해 필요한 컨트롤 액션이 포함되어 있는지 확인한다.
- 책임을 충족시키기 위해 필요한 피드백이 포함되어 있는지 확인한다(만일 피드백을 모르는 개념 개발 단계 초기에 적용한 경우, 이후 단계에서 누락된 피드백을 식별할 수 있음).

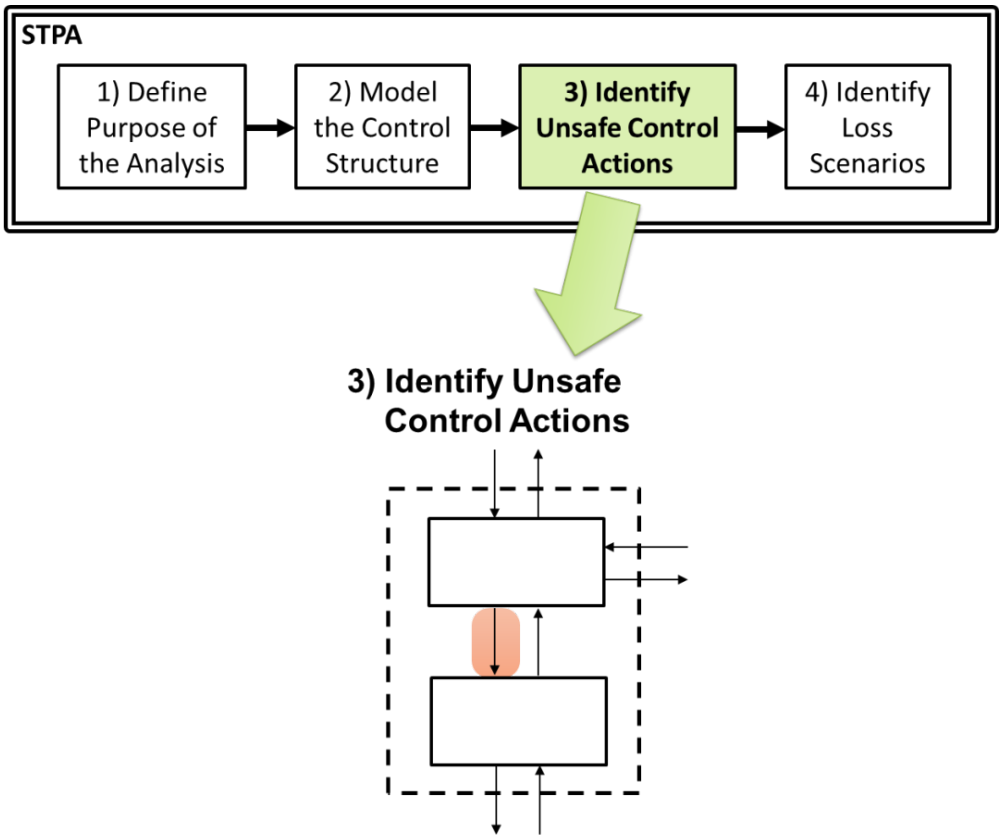


Figure 2.14: Identify Unsafe Control Actions

Definition: An Unsafe Control Action (UCA) is a control action that, in a particular context and worst-case environment, will lead to a hazard.[9]

Table 2.3 below shows examples of UCAs for the BSCU controller. A larger set of UCAs for the wheel braking subsystem can be found in Appendix C.

[9] The term "unsafe" refers to the hazards identified in STPA. As discussed earlier, hazards can include issues related to loss of human life or injury (traditional safety) but they can also be defined much more broadly to include other losses like a mission loss, loss of performance, environmental losses, etc.

언세이프 컨트롤 액션(Unsafe Control Actions) 식별

컨트롤 스트럭처가 모델링 되면 다음 단계(*그림 2.14*)는 언세이프 컨트롤 액션(UCA)을 식별하는 것이다.

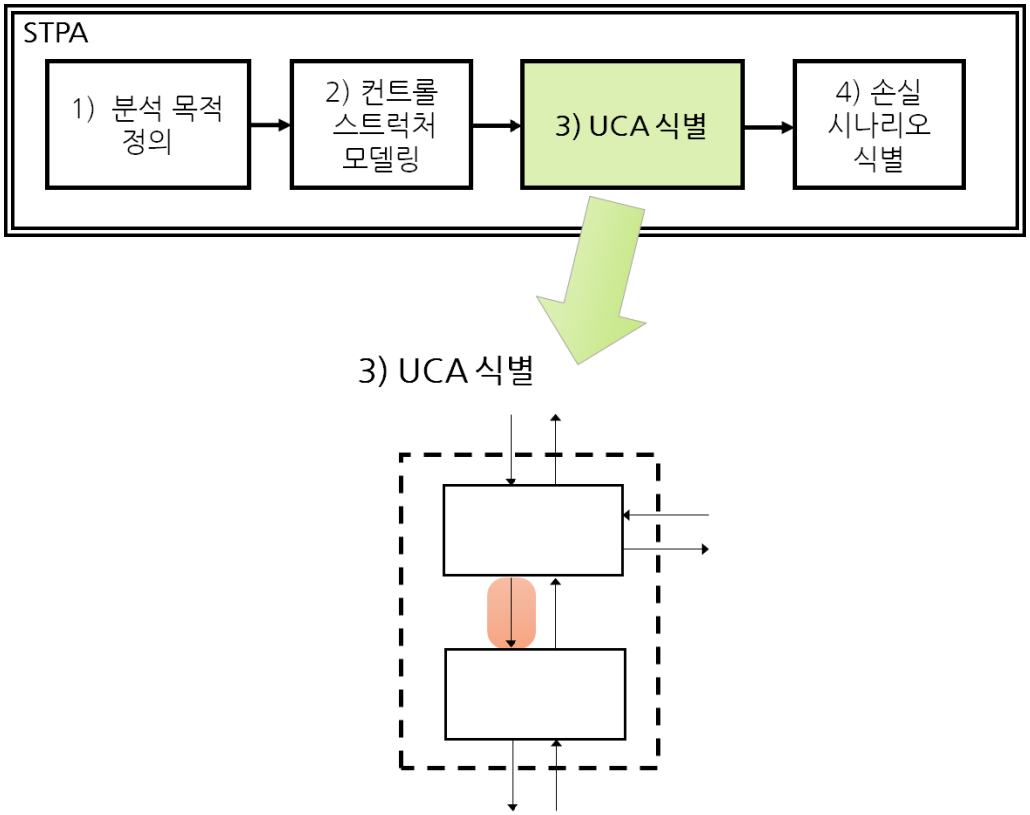


그림 2.14: UCA 식별

정의: 언세이프 컨트롤 액션(Unsafe Control Action, UCA)은 특정 상황과 최악의 환경에서 위험을 초래할 수 있는 컨트롤 액션이다.[9]

아래 *표 2.3*은 BSCU 컨트롤러에 대한 UCA의 사례이다. 휠 브레이킹 서브시스템에 대한 더 많은 UCA는 부록 C에 있다.

9 용어 "언세이프"는 STPA에서 식별된 위험을 의미한다. 앞에서 설명한 것처럼 위험은 사람의 생명 손실이나 신체 상해와 관련된 이슈(전통적인 안전)를 포함하며, 임무실패, 성능저하, 환경파괴 등의 다른 손실을 포함하여 훨씬 더 광범위하게 정의할 수 있다.

Table 2.3: Examples of Unsafe Control Actions for the BSCU (partial example)

Control Action	Not providing causes hazard	Providing causes hazard	Too early, too late, out of order	Stopped too soon, applied too long
Brake	UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]	UCA-2: BSCU Autobrake provides Brake control action during a normal takeoff [H-4.3, H-4.6] UCA-5: BSCU Autobrake provides Brake control action with an insufficient level of braking during landing roll [H-4.1] UCA-6: BSCU Autobrake provides Brake control action with directional or asymmetrical braking during landing roll [H-4.1, H-4.2]	UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]	UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

There are four ways a control action can be unsafe (represented in the columns above):

1. Not providing the control action leads to a hazard.
2. Providing the control action leads to a hazard.
3. Providing a potentially safe control action but too early, too late, or in the wrong order
4. The control action lasts too long or is stopped too soon (for continuous control actions, not discrete ones).

Consider UCA-2:

> UCA-2: BSCU Autobrake provides Brake command during a normal takeoff [H-4.3, H-4.5]

This UCA is unsafe because it can lead to H-4.3: Deceleration occurs after the V1 point during takeoff and H-4.5: Acceleration is insufficient during takeoff. Every UCA can be traced to one or more hazards (or sub-hazards), and it is good practice to document the traceability in brackets at the end of every UCA.

UCAs should specify the context in which the control action is unsafe. The context is critical. Suppose you know that the BSCU on an aircraft can provide a Brake command. Could that command be unsafe? It's impossible to assess without considering the context. UCA-2 contains the context "during a normal takeoff", which is what makes the control action unsafe.

If a control action were always unsafe, then the engineers would never include it in the system design. Every UCA must specify under what conditions (in what context) the control action is unsafe. Then we can eliminate those instances from the system design or find ways to mitigate them. Any relevant context can be referenced in a UCA, including environmental conditions, controlled process states, controller states, previous actions, and parameters (e.g. a particular descent rate being

표 2.3: BSCU의 UCA 사례(일부)

컨트롤 액션	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	너무 일찍, 너무 늦게, 잘못된 순서	너무 빨리 중지됨, 너무 오래 적용됨
브레이크	UCA-1: BSCU 오토브레이크가 BSCU 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 제공하지 않는다. [H-4.1]	UCA-2: BSCU 오토브레이크가 정상적인 이륙 중 브레이크 컨트롤 액션을 제공한다. [H-4.3, H-4.6] UCA-5: BSCU 오토브레이크가 착륙활주 동안 제동 수준이 불충분한 브레이크 컨트롤 액션을 제공한다. [H-4.1] UCA-6: BSCU 오토브레이크가 착륙활주 동안 방향성 있거나 비대칭적인 브레이크 컨트롤 액션을 제공한다. [H-4.1, H-4.2]	UCA-3: BSCU 오토브레이크가 터치다운 후 너무 늦게(>TBD초) 브레이크 컨트롤 액션을 제공한다. [H-4.1]	UCA-4: BSCU 오토브레이크가 항공기 착륙 시 너무 일찍 (지상활주 속도가 TBD에 도달하기 전에) 브레이크 컨트롤 액션 제공을 중지한다. [H-4.1]

컨트롤 액션이 안전하지 않을 수 있는 경우에는 네 가지가 있다(위 열에 표시).

1. 컨트롤 액션을 제공하지 않는 것이 위험을 유발한다.
2. 컨트롤 액션을 제공하는 것이 위험을 유발한다.
3. 잠재적으로 안전한 컨트롤 액션을 제공하지만 너무 일찍, 너무 늦게, 또는 잘못된 순서로 제공한다.
4. 컨트롤 액션이 너무 오래 지속되거나 너무 빨리 중지된다(불연속 컨트롤 액션이 아닌 연속적인 컨트롤 액션의 경우).

UCA-2의 경우를 살펴보자:

UCA-2: BSCU 오토브레이크가 정상적인 이륙 중 브레이크 명령을 제공한다[H-4.3, H-4.6]

이 UCA는 H-4.3(이륙 중 V1 포인트 이후에 감속이 발생한다)와 H-4.6(이륙 중 가속이 충분하지 않다)로 이어질 수 있기 때문에 안전하지 않다. 모든 UCA는 하나 이상의 위험(또는 하위 위험)을 추적할 수 있으므로 모든 UCA 끝에 대괄호로 추적성을 표시하는 것이 좋다.

UCA는 컨트롤 액션이 안전하지 않은 컨텍스트를 명시해야 한다. 컨텍스트는 필수적이다. 항공기의 BSCU가 브레이크 명령을 제공할 수 있다는 것을 알고 있다고 가정해보자. 그 명령은 안전하지 않을 수 있을까? 그것에 대해 컨텍스트를 고려하지 않고 평가하는 것은 불가능하다. UCA-2에 "정상적인 이륙 중"이라는 컨텍스트가 포함되어 있는데 이것이 컨트롤 액션을 안전하지 않게 만든다.

만일 컨트롤 액션이 항상 안전하지 않다면 엔지니어는 그것을 절대로 시스템 설계에 반영하지 않는다. 모든 UCA는 어떤 조건에서(어떤 컨텍스트에서) 컨트롤 액션이 안전하지 않은지 명시해야 한다. 그런 다음 시스템 설계에서 그러한 경우를 제거하거나 완화할 수 있는 방법을 찾을 수 있다. 환경 조건, 컨트롤드 프로세스의 상태, 컨트롤러 상태, 이전의 행동 및 파라미터(예: 프로그래밍된 특정 강하율(descent rate))를 포함하여 관련된 모든 컨텍스트를 UCA에서 언급할 수 있다. UCA 구성에서 "~일 때(when)",

programmed). Using words like “when”, “while”, or “during” in UCA construction is often helpful in developing the context.

Every UCA contains five parts:

UCA-2: BSCU Autobrake provides Brake command during a normal takeoff [H-4.3]
<Source> <Type> <Control Action> <Context> <Link to Hazards>

The first part is the controller that can provide the control action. The second part is the type of unsafe control action (provided, not provided, too early or too late, stopped too soon or applied too long). The third part is the control action or command itself (from the control structure). The fourth part is the context discussed above, and the last part is the link to hazards (or sub-hazards).

UCAs are often written with each part in the same order shown above, but in some cases it may be clearer or more natural to use a different ordering. The ordering is not critical. The key point is that UCAs contain these four parts.

Common questions about UCAs

Does a UCA guarantee that a hazard will always result?

No. In a best-case scenario, UCA-2 might occur early during takeoff, the pilots recognize that UCA-2 has happened, they immediately disable the BSCU, and there is no accident. In a worst-case scenario, the pilots might not be able to react and recover in time, their actions to disable the BSCU might be ineffective, there is a significant tail-wind and UCA-2 occurs right after the V1 decision speed, or other factors arise and the aircraft runs off the runway (H-4). The goal of STPA is not to argue whether the best-case or worst-case scenarios are more likely or to make assumptions about pilot capabilities and response. When engineering the BSCU we’d still like to prevent it from sending brake commands during normal takeoffs regardless of whether the flight happens to occur in a best-case or a worst-case environment. The goal of this step is simply to identify the behaviors that should be prevented. STPA is a worst-case analysis method not a best-case, average-case, or most-likely-case method.

Can I identify UCAs when we already have safeguards in place?

Systems may include safeguards like protective features, redundancies, and backup systems specifically meant to prevent unsafe control actions from causing hazards. For example, some might argue that UCA-1: “BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed” cannot lead to a hazard because an independent alternate braking system is included in the design that effectively bypasses the BSCU and allows manual braking at any time. In a best-case scenario, the safeguards will operate as intended, will be effective and sufficient, and the hazard may be avoided. However, in a worst-case scenario the safeguards may not operate as intended, may not be sufficient, or they may not be effective for the situation at hand. Similar to the reasoning above, STPA is a worst-case analysis method and we cannot omit UCAs when a safeguard exists.

Another way to understand this reasoning is to realize that we’d like to prevent UCAs *even if safeguards do exist*. For example, even though an alternate braking system may be included in the design we wouldn’t intentionally design the BSCU to provide UCAs. We’d like to make sure the BSCU Autobrake will provide the proper brake control actions (e.g. to prevent UCA-1, etc.) even when safeguards like an alternate braking system are available.

In fact, STPA is ideally applied early before safeguards are known and incorporated into the design. STPA identifies UCAs that must be prevented, and the UCAs are then used to derive functional requirements and make design decisions to prevent or mitigate the UCAs. After potential unsafe behavior is identified, then the specific design features can be created and safeguards added (if the

"~하는 동안(while)" 또는 "~하는 중에(during)"와 같은 단어를 사용하면 컨텍스트를 개발하는 데 종종 도움이 된다.

모든 UCA는 5개의 파트로 구성된다.

UCA-2: BSCU 오토브레이크는 〈소스〉 제공한다 〈유형〉 브레이크 명령을 〈컨트롤 액션〉 정상적인 이륙 중 〈컨텍스트〉 [H-4.3] 〈위험과의 연결〉

첫 번째 파트는 컨트롤 액션을 제공할 수 있는 컨트롤러이다. 두 번째 파트는 UCA의 유형(제공됨, 제공되지 않음, 너무 일찍 또는 너무 늦었음, 너무 빨리 중지됨 또는 너무 오래 적용됨)이다. 세 번째 파트는 컨트롤 액션 또는 명령 자체(컨트롤 스트럭처에 있는)이다. 네 번째 파트는 위에 설명된 컨텍스트이고, 마지막 파트는 위험으로의 연결(또는 하위 위험)이다.

UCA는 각 파트가 위에 열거한 순서대로 작성되는 경우가 있지만 경우에 따라 다른 순서를 사용하는 것이 더 명확하거나 자연스러울 수도 있다. 〈소스〉, 〈유형〉, 〈컨트롤 액션〉, 〈컨텍스트〉의 순서는 중요하지 않다. 요점은 UCA가 이 4개 파트를 포함한다는 것이다.

UCA에 관한 일반적인 질문

UCA로 인해 항상 위험이 발생하는가?

아니다. 최선의 시나리오에서는 UCA-2가 이륙 초기에 발생할 수 있고, 조종사가 UCA-2가 일어난 것을 인식한 후 즉시 BSCU를 비활성화(disable)하면 사고는 일어나지 않는다. 최악의 시나리오에서는 조종사가 시간 내에 반응 및 복구하지 못할 수 있고 BSCU를 해제하는 행동이 효과가 없을 수도 있으며 배풍(tail-wind)이 심하고 UCA-2가 V1 결정속도 직후에 발생하거나 다른 요인들이 발생하여 항공기가 활주로 밖으로 이탈할 수도 있다(H-4). STPA의 목표는 최선의 시나리오 또는 최악의 시나리오 중 무엇의 가능성이 높은 지를 논하거나 조종사의 능력 및 대응에 대해 가정하는 것이 아니다. BSCU를 제작할 때 비행이 최선의 상황에서 이루어지는지 최악의 상황에서 이루어지는지와 무관하게 정상적인 이륙 중 브레이크 명령을 보내는 것을 막고 싶은 것이다. 이 단계의 목표는 예방되어야 할 행동을 식별하는 것이다. STPA는 최악의 상황을 분석하는 기법(worst-case analysis method)이며 최선의 상황이나 평균적인 상황, 또는 가장 발생확률이 높은 상황을 분석하는 방법이 아니다.

이미 보호장치(safeguards)가 갖춰져 있을 때 UCA를 도출할 수 있을까?

시스템은 언세이프 컨트롤 액션으로 인한 위험을 방지하기 위한 예방적 기능, 다중화 및 백업 시스템과 같은 보호장치를 갖추고 있을 수 있다. 예를 들면 독립적인 대체 브레이킹 시스템이 설계에 반영되어 있으면 BSCU를 효과적으로 우회(bypass)할 수 있고 언제든지 매뉴얼 브레이킹이 가능하다. 이 때, UCA-1: "BSCU 오토브레이크가 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 제공하지 않는다"는 위험을 초래할 수 없다고 주장할 수도 있다. 최선의 시나리오에서는 보호장치가 의도한대로 작동하고, 효과적이고 충분한 수준으로 작동하여 위험을 피할 수 있을 것이다. 그러나 최악의 시나리오에서는 보호장치가 의도한대로 작동하지 않을 수도 있고 충분하지 않을 수도 있으며, 현재의 상황에 효과적이지 않을 수도 있다. 위의 추론과 유사하게 STPA는 최악의 경우에 대한 분석 방법이며 보호장치가 있더라도 UCA를 생략할 수는 없다.

이러한 추론을 이해하는 또 다른 방법은 보호장치가 존재하는 경우에서도 UCA를 방지하고자 하는 것임을 깨닫는 것이다. 예를 들어 또 다른 브레이킹 시스템이 설계에 포함되어 있더라도 의도적으로 UCA가 발생하도록 BSCU를 설계하지는 않는다. 대체 브레이킹 시스템과 같은 보호장치가 있더라도

design does not already exist) or the adequacy of existing design decisions and safeguards can be determined (if the design already exists).

The last two types of unsafe control actions are both about timing. What's the difference?

The third type of UCA is about control actions provided at the wrong time—too early, too late, or out of order. The fourth type of unsafe control action only applies to control actions with a duration, that is, continuous or non-discrete control actions.

First, suppose the Brake command provided by BSCU Autobrake is implemented as a continuous control action. In other words, the brakes are applied when Autobrake starts providing the Brake control action, and the brakes continue to be applied until Autobrake stops providing the Brake control action. The initial timing of the Brake control action can cause a hazard:

> UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]

The Brake control action could also be provided correctly during a landing and on-time without delay (within TBD seconds), but Autobrake then immediately stops providing the control action and the brakes immediately cease to be applied. The braking was stopped too soon to be effective. This behavior could cause H-4.1 even though the control action was originally provided in the right situation and on time, as reflected in UCA-4:

> UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

Now let's consider an alternative situation: the BSCU Autobrake provides two separate discrete control actions to Start Braking and to Stop Braking. The individual control actions do not have a meaningful duration, so "stopped too soon / applied too long" is not applicable. Instead, we'd consider how the either control action could be provided too early or too late:

> UCA-3: BSCU Autobrake provides the Start Brake control action too late (>TBD seconds) after touchdown [H-4.1]
>
> UCA-4: BSCU Autobrake provides the Stop Brake control action too early (before TBD taxi speed attained) when aircraft lands [H-4.1]

Notice that the same issues are covered by the analysis whether control actions are continuous or discrete.

Do I need to identify exactly one UCA for each type of unsafe control action?

No. Although all four UCA types should be considered, they may not all be applicable in every case. It is also possible to identify several UCAs of a single type, as demonstrated by UCA-2, UCA-5, and UCA-6 above. In general, each category may contain 0, 1, 2, or more UCAs.

Are there more than four types of unsafe control? Is another category needed?

These four categories are provably complete—there is no other category of unsafe control action. However, there are subcategories. For example, there are three different ways the second type of UCA can occur:

2. An unsafe control action is provided
    a. Consider contexts in which the control action may never be safe
    b. Consider contexts in which an insufficient or excessive control action may be unsafe
    c. Consider contexts in which the direction of the control action may be unsafe

BSCU 오토브레이크가 적절한 브레이크 컨트롤 액션(예: UCA-1을 방지하기 위한)을 제공하는지 확인하고 싶은 것이다.

사실, STPA는 보호장치가 정해지고 설계에 통합되기 전에 일찍 적용하는 것이 이상적이다. STPA는 반드시 방지해야 하는 UCA를 식별한 다음 UCA를 사용하여 기능 요구사항을 도출하고 UCA를 방지하거나 완화하기 위한 설계를 결정한다. 잠재적으로 안전하지 않은 행동이 식별되면 특정 설계 기능(feature)을 생성하고 보호장치를 추가하거나(설계가 존재하지 않을 경우), 기존 설계 결정 및 보호장치의 적절성을 판단할 수 있다(이미 설계된 경우).

마지막 두 가지 유형의 UCA은 모두 타이밍에 관한 것이다. 차이는 무엇인가?

세 번째 유형의 UCA는 잘못된 시간(너무 일찍, 너무 늦거나 순서가 맞지 않는)에 제공되는컨트롤 액션에 관한 것이다. 네 번째 유형의 UCA는 지속 시간이 있는, 즉, 연속적 또는 개별적이지 않은 컨트롤 액션에만 적용된다.

먼저, BSCU 오토브레이크가 제공하는 브레이크 명령이 연속적인 컨트롤 액션으로 구현되었다고 가정하자. 바꿔 말하면 브레이크는 오토브레이크가 브레이크 컨트롤 액션을 제공하기 시작할 때 적용되고 오토브레이크가 브레이크 컨트롤 액션을 중단할 때까지 계속 적용되며, 브레이크 컨트롤 액션의 첫 타이밍(initial timing)이 위험을 초래할 수 있다.

> UCA-3: BSCU 오토브레이크가 터치다운 후 너무 늦게(>TBD초) 브레이크 컨트롤 액션을 제공한다 [H-4.1]

브레이크 컨트롤 액션은 착륙 중 정확하게 그리고 지연 없이 정시에 제공될 수 있지만(TBD 초 이내), 오토브레이크가 컨트롤 액션 제공을 즉시 중단하면 브레이크는 바로 중단된다. 브레이크가 효력을 발생시키기에는 너무 일찍 중단되었다. UCA-4에 반영된 바와 같이 컨트롤 액션이 처음부터 적절한 상황 및 시점에 제공되었더라도 이 동작으로 인해 H-4.1이 발생할 수 있다.

> UCA-4: BSCU 오토브레이크가 항공기 착륙 시 너무 일찍(지상활주 속도가 TBD에 도달하기 전에) 브레이크 컨트롤 액션 제공을 중단한다 [H-4.1]

이제는 다른 상황을 고려해 보자. BSCU 오토브레이크는 '브레이킹 시작'과 '브레이킹 중지'에 대한 두 개의 별도 컨트롤 액션을 제공한다고 하자. 개별 컨트롤 액션에는 지속시간(duration)이 의미 없으므로 "너무 빨리 중지/너무 오래 적용"이 적용되지 않는다. 대신 어떻게 각각의 컨트롤 액션이 너무 일찍 또는 너무 늦게 제공될 수 있는지에 대해 고려해 볼 수 있다.

> UCA-3: BSCU 오토브레이크가 터치다운 후 너무 늦게(>TBD초) '브레이크 시작' 컨트롤 액션을 제공한다 [H-4.1]
>
> UCA-4: BSCU 오토브레이크가 항공기 착륙 시 너무 빨리(지상활주 속도가 TBD에 도달하기 전에) '브레이크 중지' 컨트롤 액션을 제공한다 [H-4.1]

동일한 이슈는 위와 같이 컨트롤 액션이 연속적인지 또는 불연속적인지를 분석하여 처리할 수 있다.

각 유형의 언세이프 컨트롤 액션에 대해 정확히 하나의 UCA를 도출해야하는가?

아니다. 네 가지 UCA 유형 모두 고려해야 하지만 모든 경우가 적용되지 않을 수도 있다. 위의 UCA-2, UCA-5 및 UCA-6에서 알 수 있듯이 한 가지 유형에 여러 UCA가 식별될 수도 있다. 일반적으로 각 유형에 따라 0개, 1개, 2개 또는 그 이상의 UCA가 있을 수 있다.

The last two cases are only relevant for control actions that include one or more parameters. For example, a BSCU Brake control action is not typically a simple binary on/off control action. The BSCU controls braking by commanding the specific amount of braking to be applied. UCA-2 specifies that providing the Brake command during normal takeoff (regardless of the amount of braking the BSCU specifies) is unsafe. UCA-5 exists because even though the BSCU may correctly provide a brake command during landing roll, it can still be unsafe if the BSCU commands an insufficient level of braking. UCA-6 exists because the braking commands could be sent asymmetrically causing the aircraft to turn in a wrong direction. Whenever control actions can specify one or more parameters, it is important to consider how the parameters may be insufficient, excessive, or in a wrong direction for a given context.

I just identified an important UCA that isn't related to any system-level hazards. What do I do?

Every UCA must be traceable to one or more system-level hazards. If you identify a UCA that does not relate to one of the identified hazards, you could be missing a hazard. Identify the system-level state or condition caused by the UCA and consider adding a new hazard or revising the existing set of hazards to include the new state or condition. STPA is iterative and need not be performed in a strictly linear fashion—earlier results can be updated as the analysis progresses and as more information becomes available.

Where is the outcome or result of the UCA described?

Every UCA should reference the hazards or sub-hazards it can lead to, which captures system-level effects and outcomes of the UCA. For many systems, this link between a UCA and hazards is clear and obvious. For example, the BSCU providing insufficient braking during landing roll is clearly related to H-4.1: Insufficient deceleration upon landing.

However, in some cases the link between a UCA and the hazards may be more complex or unintuitive. It is good practice to document any special reasoning behind UCAs and their relationship to the hazards, particularly in more complex applications where it is not obvious. The reasoning can sometimes be documented by adding a few words to the UCA, but this can be difficult if the reasoning is not simple. A general solution is to document comments for each UCA as needed. UCA comments may describe the rationale for the UCA, how the UCA can lead to hazards, any assumptions the UCA is based on, the effects or results of the UCA, or other information needed to understand the UCA and how it can lead to hazards.

Be careful not to confuse the UCA context with UCA outcomes, results, or other reasoning. One common mistake is to omit the UCA context and instead include the UCA result. For example:

> Correct UCA: BSCU Autobrake provides Brake command during normal takeoff [H-4.3]
>
> Incorrect UCA: BSCU Autobrake provides Brake command resulting in a collision

If the UCA context (the current state or condition) is not identified, the next steps that create requirements and identify scenarios will be difficult or impossible to perform. Every UCA must contain the UCA context. For clarity, it may also contain the result.

Should I specify process model flaws in the UCA context?

Be careful not to confuse a UCA with a cause of a UCA. The UCA context should specify the actual (true) state or condition that would make the control action unsafe, not a particular controller process model or belief (which may or may not be true). The next step in STPA will identify causes of UCAs, such as process model flaws and other factors. For example:

> Correct UCA: BSCU Autobrake provides Brake command during normal takeoff
>
> Incorrect UCA: BSCU Autobrake provides Brake command when it incorrectly believes the aircraft is landing

언세이프 컨트롤에는 네 가지 유형보다 더 많은 것이 있을까? 다른 유형이 필요할까?

이 네 가지 유형이면 아마도 충분하며, 언세이프 컨트롤 액션의 다른 유형은 없다. 하지만 하위 유형은 있다. 예를 들어 두 번째 유형의 UCA가 발생할 수 있는 세 가지 형태가 있다.

2. 언세이프 컨트롤 액션이 제공된다.
   a. '컨트롤 액션이 절대로 안전하지 않은 컨텍스트'를 고려하라.
   b. '컨트롤 액션이 부족하거나 지나치면 안전하지 않은 컨텍스트'를 고려하라.
   c. '컨트롤 액션의 방향성에 따라 안전하지 않게 되는 컨텍스트'를 고려하라.

마지막 두 경우는 하나 이상의 파라미터를 포함하는 컨트롤 액션에만 해당된다. 예를 들어 BSCU 브레이크 컨트롤 액션은 일반적으로 단순한 이진(binary) 형태의 on/off 컨트롤 액션이 아니다. BSCU는 특정 제동량을 적용하여 제동을 수행한다. UCA-2는 정상적인 이륙 중에 브레이크 명령을 제공하는 것은(BSCU가 제동하는 양에 관계없이) 안전하지 않다고 명시하고 있다. 하지만, BSCU가 착륙활주 동안 브레이크 명령을 올바르게 제공하더라도 BSCU가 불충분한 수준으로 브레이크 한다면 안전하지 않으므로 UCA-5가 명시되었다. 또한, 브레이크 명령이 비대칭적으로 전송되면 항공기가 잘못된 방향으로 회전할 수 있기 때문에 UCA-6가 명시되었다. 컨트롤 액션이 하나 이상의 파라미터를 명시할 수 있다면, 주어진 컨텍스트에 대하여 파라미터가 불충분한지, 과도한지, 잘못된 방향일 수 있는지를 고려하는 것이 중요하다.

시스템 수준의 위험과 관련이 없는 중요한 UCA를 발견했다. 어떻게 해야 하는가?

모든 UCA는 반드시 하나 이상의 시스템 수준 위험을 추적할 수 있어야 한다. 만약 식별된 위험 중 관련 있는 위험이 하나도 없는 UCA가 있다면 위험이 누락된 것일 수 있다. UCA로 인해 발생할 수 있는 시스템 수준의 상태 또는 조건을 식별하고, 새로운 위험을 추가하거나 기존의 위험 집합을 수정하여 새로운 상태 또는 조건을 포함할 것을 고려한다. STPA는 반복적인 것으로, 엄격하게 선형적인 방식으로 수행할 필요가 없으므로, 분석이 진행되고 더 많은 정보가 가용하게 되면 이전 결과를 갱신할 수 있다.

UCA가 설명된 결과나 결과물(result 또는 outcome)은 어디에 있는가?

모든 UCA는 시스템 수준의 영향과 UCA의 결과물을 기술할 수 있는 위험 또는 하위 위험을 참조해야 한다. 많은 시스템에서 UCA와 위험 간의 연관성은 분명하고 명백하다. 예를 들어 착륙활주 동안 브레이크가 불충분한 BSCU는 H-4.1(착륙 시 불충분한 감속)과 명확히 관련이 있다.

그러나 어떤 경우에는 UCA와 위험 사이의 연관성이 더 복잡하거나 직관적이지 않을 수 있다. 특히, 명확하지 않은 복잡한 어플리케이션의 경우에는 UCA에 대한 특별한 추론(이유)과, 그러한 추론과 위험과의 관계를 문서로 기록하는 것이 좋다. UCA에 몇 개의 단어를 추가하여 추론을 포함하여 작성할 수도 있지만 추론이 단순하지 않은 경우에는 이 방법은 어려울 수 있다. 일반적인 해법은 UCA마다 필요한 만큼 코멘트를 작성하는 것이다. UCA 코멘트에는 UCA의 근거, UCA가 위험을 초래할 수 있는 방법, UCA의 기반이 되는 가정, UCA의 효과나 결과 또는 UCA를 이해하고 UCA가 위험에 이르는 방법을 이해하기 위한 다른 정보를 설명한다.

UCA 컨텍스트가 UCA 결과물(outcomes), 결과(results) 또는 다른 추론과 혼동되지 않도록 주의해야 한다. 흔한 실수는 UCA 컨텍스트를 누락하고, 대신에 UCA 결과를 쓰는 것이다. 예를 들면:

올바른 UCA: BSCU 오토브레이크가 정상적인 이륙 중 브레이크 명령을 제공한다 [H-4.3]
잘못된 UCA: BSCU 오토브레이크가 브레이크 명령을 제공하여 충돌을 일으킨다

만일 UCA 컨텍스트(현재 상태 또는 조건)가 식별되지 않으면 요구사항을 만들고 시나리오를 식별하

Identifying Human UCAs

The same approach can be applied to human controllers to identify UCAs. For example, consider the flight crew control action to power off the BSCU. *Table 2.4* shows some examples of the corresponding flight crew UCAs for this command. Additional example UCAs for the flight crew related to wheel braking can be found in Appendix C.

Table 2.4: Example Unsafe Control Actions for the Flight Crew (partial example)

Control Action	**Not providing causes hazard**	**Providing causes hazard**	**Too early, too late, out of order**	**Stopped too soon, applied too long**
Power Off BSCU	UCA-1: Crew does not provide BSCU Power Off when abnormal WBS behavior occurs [H-4.1, H-4.4, H-7]	UCA-2: Crew provides BSCU Power Off when Anti-Skid functionality is needed and WBS is functioning normally [H-4.1, H-7]	Crew powers off BSCU too early before Autobrake or Anti-Skid behavior is completed when it is needed [H-4.1, H-7]	N/A

The last column is considered N/A (not applicable) because the Power Off control action does not have a duration in this case. The crew simply provides Power On and Power Off commands to toggle the state of the BSCU. Issues related to the BSCU remaining powered off for too long will be captured when considering the Power On command not provided or provided too late. Note that humans are treated in the same way as other types of system components and can be easily integrated into the overall analysis.

Tips to prevent common mistakes

The following tips will help prevent common mistakes when identifying UCAs:

Tips to prevent common mistakes when identifying UCAs

- Ensure every UCA specifies the context that makes the control action unsafe.
- Ensure UCA contexts specify the actual states or conditions that would make the control action unsafe, not potential beliefs about the actual states.
- Ensure the UCA contexts are defined clearly.
- Ensure the UCA contexts are included and not replaced by future effects or outcomes.
- Ensure traceability is documented to link every UCA with one or more hazards.
- Review any control action types assumed to be N/A, and verify they are not applicable.
- For any continuous control actions with a parameter, ensure that excessive, insufficient, and wrong direction of the parameters are considered.
- Ensure any assumptions or special reasoning behind the UCAs are documented

Defining Controller Constraints

Definition: A controller constraint specifies the controller behaviors that need to be satisfied to prevent UCAs

는 다음 단계를 수행하기 어렵거나 수행이 불가능하다. 모든 UCA는 UCA 컨텍스트를 포함해야 한다. UCA를 명확히 하기 위해 UCA에 결과를 포함시킬 수는 있다.

UCA 컨텍스트에서 프로세스 모델 결함을 명시해야 하는가?

UCA와 UCA의 원인을 혼동하지 않도록 주의해라. UCA 컨텍스트에는 특정 프로세스 모델이나 믿음(사실일 수도 있고 아닐 수도 있음)이 아니라 컨트롤 액션을 안전하지 않게 만드는 실제(사실) 상태 또는 조건을 명시해야 한다. STPA의 다음 단계에서는 프로세스 모델 결함(flaws)이나 다른 요인과 같은 UCA의 원인을 식별한다. 예를 들어:

올바른 UCA: BSCU 오토브레이크가 정상적인 이륙 중 브레이크 명령을 제공한다.
잘못된 UCA: BSCU 오토브레이크가 항공기가 착륙 중이라고 잘못 판단하여 브레이크 명령을 제공한다.

휴먼 UCA 식별

휴먼 컨트롤러에 대한 UCA를 식별하기 위해서도 동일한 방법을 사용할 수 있다. 예를 들어 BSCU의 전원을 끄는 승무원의 컨트롤 액션을 고려해보자. *표 2.4*는 이 명령에 대한 해당 승무원 UCA의 몇 가지 예를 보여준다. 휠 브레이킹(WBS)과 관련된 승무원 추가 예제 UCA는 부록 C에 있다.

표 2.4: 승무원에 대한 UCA 예제(일부예제)

컨트롤 액션	**제공하지 않음이 위험을 유발함**	**제공함이 위험을 유발함**	**너무 일찍, 너무 늦게, 잘못된 순서**	**너무 빨리 중지됨, 너무 오래 적용됨**
BSCU 전원끄기	Crew-UCA-1: 비정상적인 WBS 동작이 발생할 때 승무원이 BSCU 전원끄기를 제공하지 않는다. [H-4.1, H-4.4, H-7]	Crew-UCA-2: 미끄럼 방지 기능이 필요하고 WBS가 정상 작동할 때 승무원이 BSCU 전원끄기를 제공한다. [H-4.1, H-7] [H-4.1, H-4.2]	Crew-UCA-3: 승무원이 오토브레이크 또는 미끄럼 방지 동작이 필요할 때, 해당 동작이 완료되기 전에 너무 일찍 BSCU를 종료한다. [H-4.1, H-7]	N/A

위의 전원끄기 컨트롤 액션은 승무원이 전원을 켜고 끄는 명령으로, BSCU의 상태를 전환(toggle)하므로 지속시간(duration)이 없어 마지막 열은 N/A(적용할 수 없음)가 된다. 너무 오랫동안 전원이 꺼진 상태로 유지되는 BSCU와 관련된 문제는 전원켜기 명령이 제공되지 않거나 너무 늦게 제공되는 상황을 고려할 때 다뤄진다. 사람은 다른 유형의 시스템 컴포넌트와 동일한 방식으로 다룰 수 있으며 전체 분석에 쉽게 통합될 수 있다.

일반적인 실수 방지 팁

다음 팁은 UCA를 식별할 때 일반적인 실수를 예방하는 데 도움이 되는 사항들이다.

UCA를 식별할 때 흔한 실수를 방지하는 팁

- 모든 UCA가 컨트롤 액션을 안전하지 않게 하는 컨텍스트를 명시하는지 확인한다.
- UCA 컨텍스트가 실제 상황에 대한 잠재적 가능성이 아니라 컨트롤 액션을 안전하지 않게 만드는 실제 상태 또는 조건을 명시하는지 확인한다.
- UCA 컨텍스트가 명확하게 정의되었는지 확인한다.
- UCA 컨텍스트가 포함되었는지 확인하고 미래의 영향이나 결과로 대체되지 않았는지 확인한다.
- 모든 UCA에 대해 UCA와 관련된 하나 이상의 위험을 문서화 하였는지 확인한다.

Once UCAs have been identified, they can be translated into constraints on the behavior of each controller. For example, when analyzing BSCU control actions we determined that the BSCU providing a Brake control action during a normal takeoff could lead to a hazard. Therefore, the BSCU must not provide the Brake control action in that context. In general, each UCA can be inverted to define constraints for each controller. Table 2.5 shows the BSCU behavioral constraints that can be derived from the UCAs in *Table 2.4*.

Table 2.5: Example Unsafe Control Actions for the Flight Crew (incomplete)

Unsafe Control Actions	**Controller Constraints**
UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]	C-1: BSCU Autobrake must provide the Brake control action during landing roll when the BSCU is armed [UCA-1]
UCA-2: BSCU Autobrake provides Brake control action during a normal takeoff [H-4.3, H-4.5]	C-2: BSCU Autobrake must not provide Brake control action during a normal takeoff [UCA-2]
UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]	C-3: BSCU Autobrake must provide the Brake control action within TBD seconds after touchdown [UCA-3]
UCA-4: BSCU Autobrake stops providing the Brake control action too early (before TBD taxi speed attained) during landing roll [H-4.1]	C-4: BSCU Autobrake must not stop providing the Brake control action before TBD taxi speed is attained during landing roll [UCA-4]
UCA-5: BSCU Autobrake provides Brake control action with an insufficient level of braking during landing roll [H-4.1]	C-5: BSCU Autobrake must not provide less than TBD level of braking during landing roll [UCA-5]
UCA-6: BSCU Autobrake provides Brake control action with directional or asymmetrical braking during landing roll [H-4.1, H-4.2]	C-6: BSCU Autobrake must not provide directional or asymmetrical braking during landing roll [UCA-6]

Inputs and Outputs for Control Action Analysis

Figure 2.15 summarizes the inputs and outputs for the control action analysis.

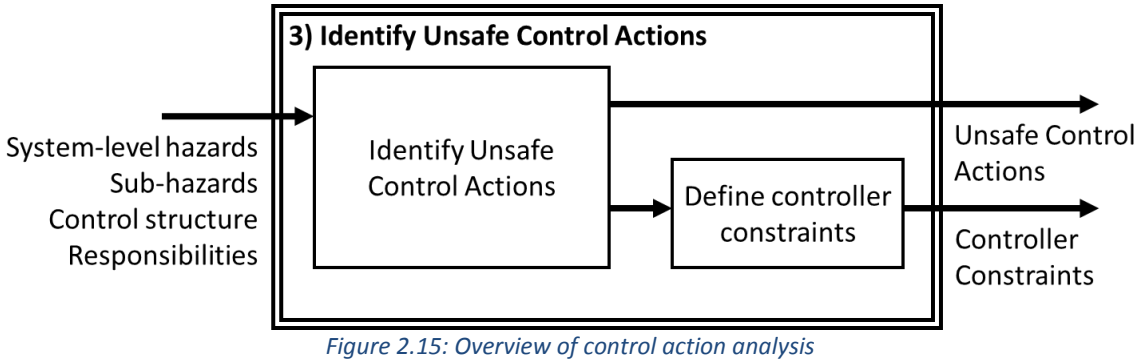


Figure 2.15: Overview of control action analysis

- N/A로 간주된 모든 컨트롤 액션 유형을 검토하고 적용할 수 없는 것인지 확실히 한다.
- 파라미터를 사용한 연속적인 컨트롤 액션의 경우, 파라미터가 과도하거나 불충분하거나, 방향이 잘못되지 않았는지 확인한다.
- UCA 배후에 있는 가정이나 특별한 추론이 문서화되어 있는지 확인한다.

컨트롤러 안전 제약사항 정의

정의: 컨트롤러 안전 제약사항은 UCA를 예방하기 위해 충족되어야 하는 컨트롤러의 동작을 기술한다.

UCA가 식별되면 UCA는 각 컨트롤러의 동작에 대한 안전 제약사항으로 변환될 수 있다. 예를 들어 BSCU 컨트롤 액션을 분석할 때, 정상적인 이륙 중 BSCU가 브레이크 컨트롤 액션을 제공하면 위험을 초래할 수 있다고 판단했다. 따라서 BSCU는 그 상황에서 브레이크 컨트롤 액션을 제공하지 말아야 한다. 일반적으로 각 UCA를 역으로 기술하여 각각의 컨트롤러에 대한 안전 제약사항을 정의할 수 있다. 표 2.5는 *표 2.3*의 UCA로부터 유도될 수 있는 BSCU의 동작 안전 제약사항을 나타낸다.

표 2.5: BSCU의 안전 제약사항 사례(불완전)

UCA	컨트롤러 안전 제약사항
UCA-1: BSCU 오토브레이크가 BSCU 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 제공하지 않는다. [H-4.1]	C-1: BSCU 오토브레이크는 BSCU가 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 반드시 제공해야 한다. [UCA-1]
UCA-2: BSCU 오토브레이크가 정상적인 이륙 중 브레이크 컨트롤 액션을 제공한다. [H-4.3, H-4.6]	C-2: BSCU 오토브레이크는 정상적인 이륙 중 브레이크 컨트롤 액션을 절대로 제공하면 안 된다. [UCA-2]
UCA-3: BSCU 오토브레이크가 터치다운 후 너무 늦게 (>TBD초) 브레이크 컨트롤 액션을 제공한다. [H-4.1]	C-3: BSCU 오토브레이크는 터치다운 후 TBD초 내에 브레이크 컨트롤 액션을 반드시 제공해야 한다. [UCA-3]
UCA-4: BSCU 오토브레이크가 착륙활주 동안 너무 일찍 (지상활주 속도가 TBD에 도달하기 전에) 브레이크 컨트롤 액션 제공을 중지한다. [H-4.1]	C-4: BSCU 오토브레이크는 착륙활주 동안 지상활주 속도가 TBD에 도달하기 전에 브레이크 컨트롤 액션을 절대로 중지하면 안 된다. [UCA-4]
UCA-5: BSCU 오토브레이크가 착륙활주 동안 제동 수준이 불충분한 브레이크 컨트롤 액션을 제공한다. [H-4.1]	C-5: BSCU 오토브레이크는 착륙활주 동안 절대 TBD 이하로 브레이크 해서는 안 된다. [UCA-5]
UCA-6: BSCU 오토브레이크가 착륙활주 동안 방향성 있거나 비대칭적인 브레이크 컨트롤 액션을 제공한다. [H-4.1, H-4.2]	C-6: BSCU 오토브레이크는 착륙활주 동안 방향성 있거나 비대칭적인 브레이크를 제공해서는 안 된다. [UCA-6]

컨트롤 액션 분석을 위한 입력과 출력

*그림 2.15*는 컨트롤 액션 분석을 위한 입력과 출력을 요약한 것이다.

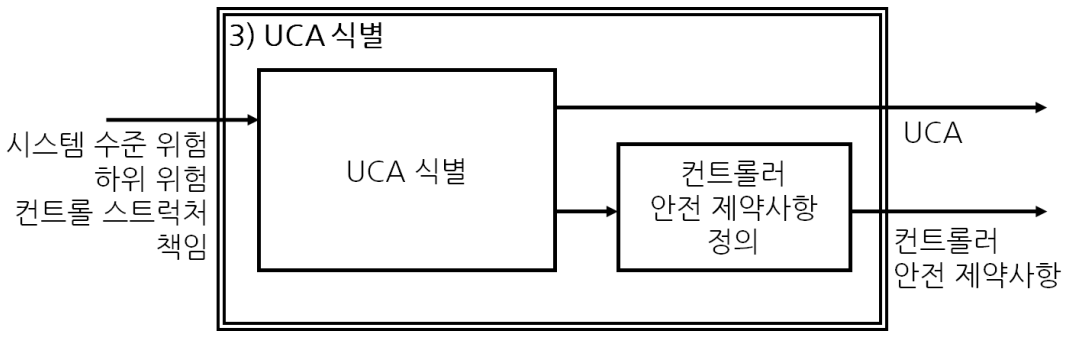


그림 2.15: 컨트롤 액션 분석의 개요

Identifying loss scenarios

Once unsafe control actions have been identified, the next step (shown in *Figure 2.16*) is to identify loss scenarios.

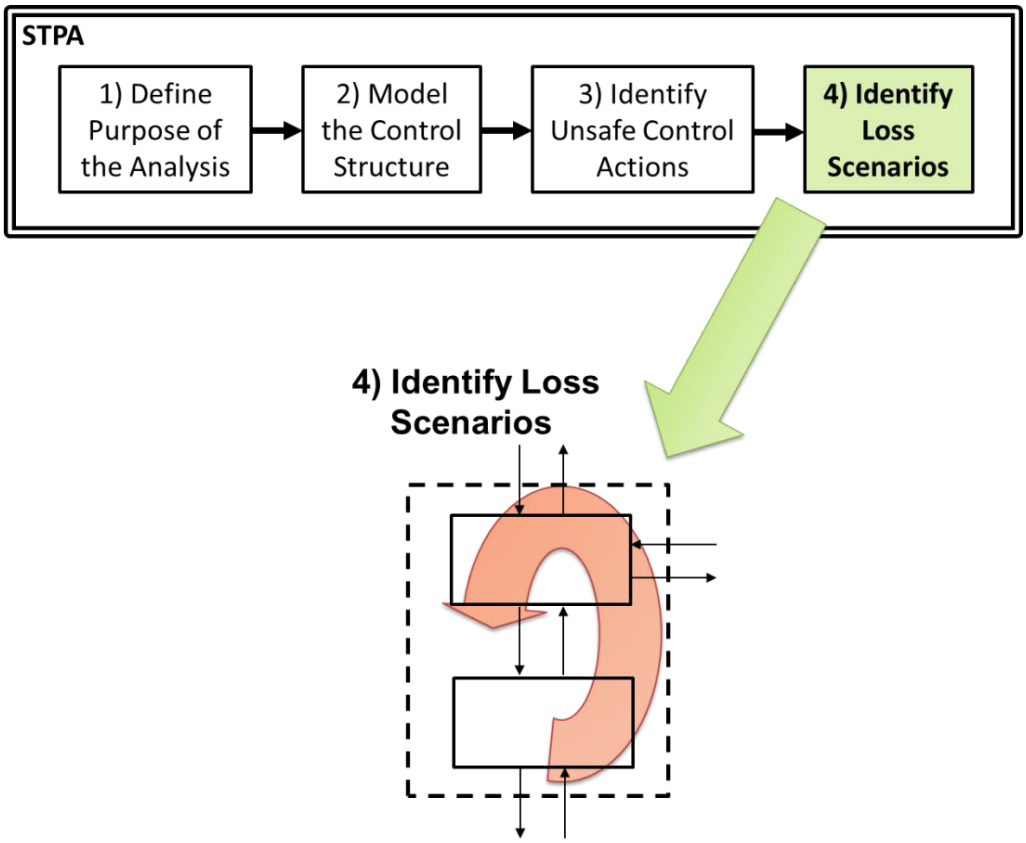


Figure 2.16: Identify loss scenarios

Definition: A loss scenario describes the causal factors that can lead to the unsafe control actions and to hazards.

Two types of loss scenarios must be considered, as shown in *Figure 2.17*:

a) Why would Unsafe Control Actions occur?
b) Why would control actions be improperly executed or not executed, leading to hazards?

손실(loss) 시나리오 식별

UCA를 식별하면 그 다음 단계는 손실 시나리오를 식별하는 것이다. (*그림 2.16에 보임*)

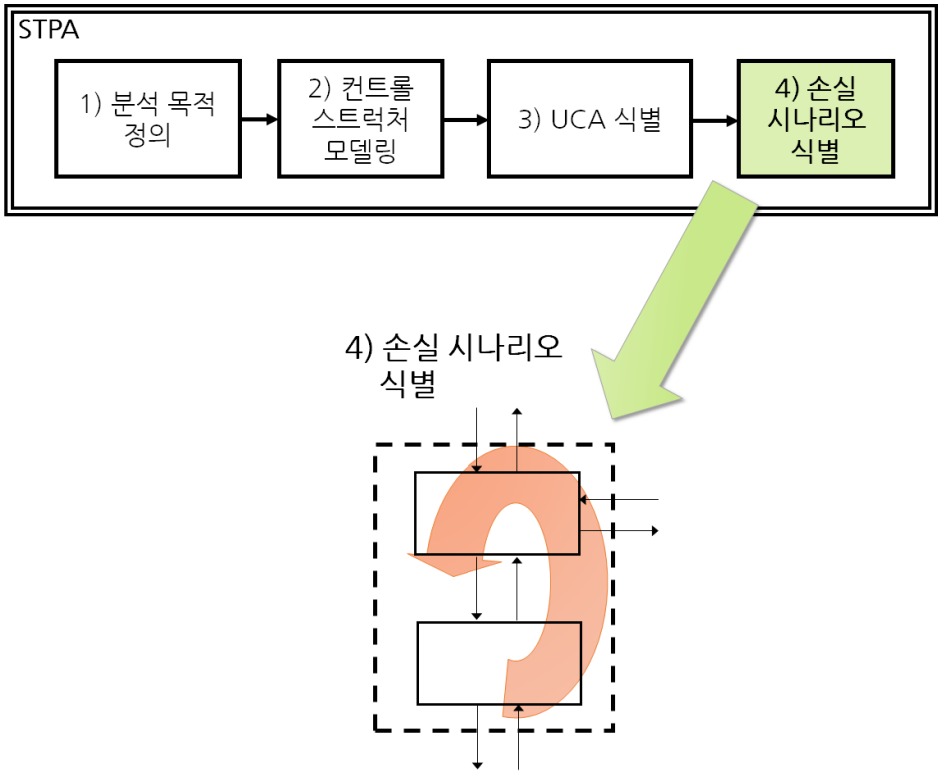


그림 2.16: 손실 시나리오 식별

정의: 손실(loss) 시나리오는 UCA 및 위험을 초래할 수 있는 원인 요소(causal factor)를 기술한다.

*그림 2.17*에 보이는 것과 같이 두 가지 유형의 손실 시나리오를 반드시 고려한다.

a) 왜 언세이프 컨트롤 액션(UCA)이 발생하는가?

b) 왜 컨트롤 액션이 부적절하게 실행되거나 실행되지 않아 위험을 유발하는가?

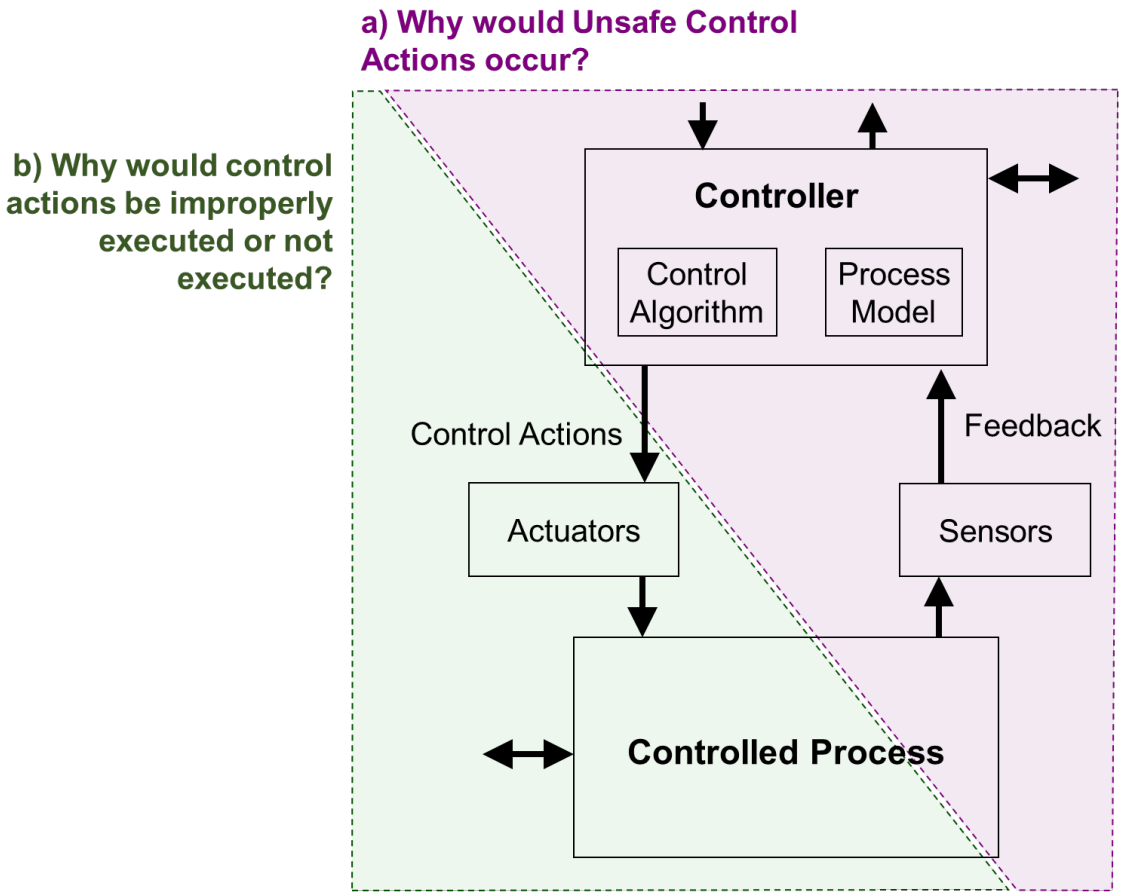


Figure 2.17: Two types of scenarios that must be considered

Notice that *Figure 2.17* includes sensors and actuators. Up to this point in the analysis, we have considered the control actions and feedback that may exist but we have not yet examined how the feedback is measured or detected (e.g. with sensors) or how the control actions are executed (e.g. with actuators). Because scenarios identify the specific causes of unsafe control and feedback, it is helpful to refine the control structure to include sensors and actuators.

a) Identifying scenarios that lead to Unsafe Control Actions

This type of scenario can be created by starting with a UCA and working backward to explain what could cause the controller to provide (or not provide) that control action. In general, scenarios that lead to UCAs may involve:

- Failures related to the controller (for physical controllers)
  - Physical failure of the controller itself
  - Power failure
  - Etc.
- Inadequate control algorithm
  - Flawed implementation of the specified control algorithm
  - The specified control algorithm is flawed
  - The specified control algorithm becomes inadequate over time due to changes or degradation

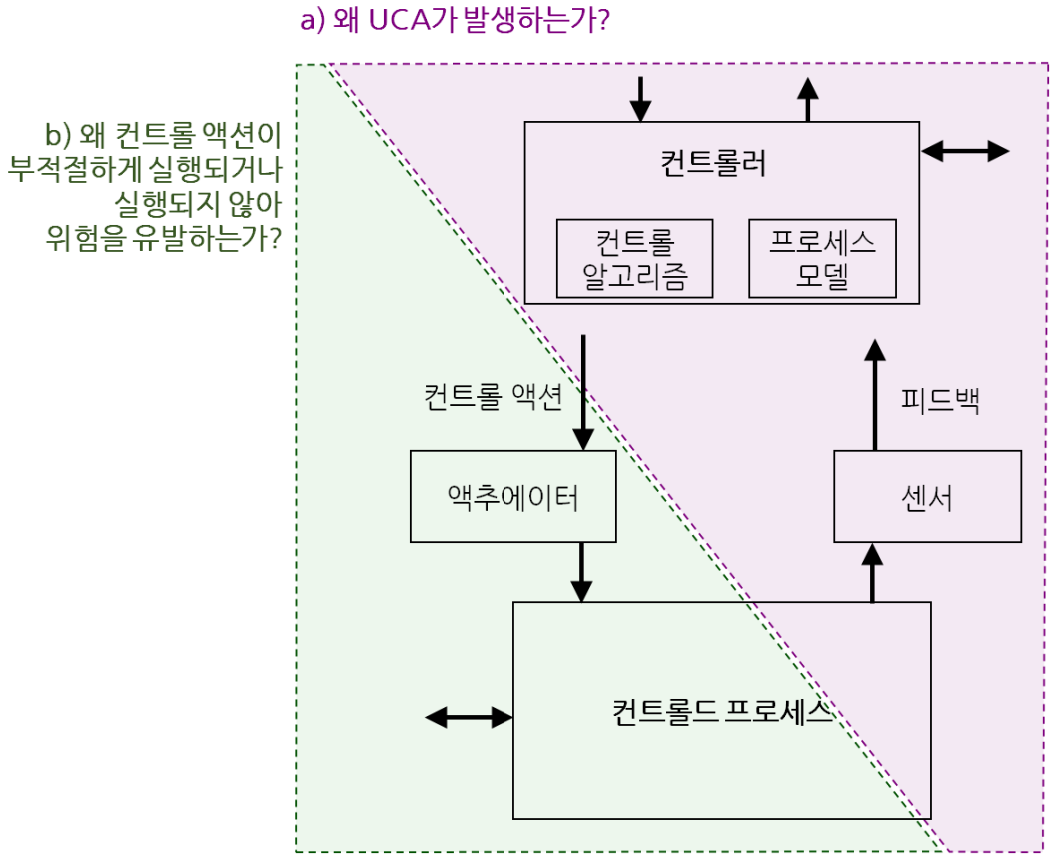


그림 2.17: 반드시 고려해야 하는 두 종류의 시나리오

*그림 2.17*에 센서와 액추에이터가 포함되어 있음을 주목하자. 우리는 지금까지의 분석에서는 존재할 수 있는 컨트롤 액션과 피드백에 대하여 고민했지만 피드백이 어떻게 측정되고 검출되는지(예: 센서를 통해) 또는 컨트롤 액션이 어떻게 실행되는지(예: 액추에이터를 통해)는 아직 검토하지 않았다. 시나리오에는 안전하지 않은 컨트롤과 피드백의 구체적인 원인이 식별되어야 하므로 센서와 액추에이터를 포함하도록 컨트롤 스트럭처를 상세화하는 것이 좋다.

a) 언세이프 컨트롤 액션을 유발하는 시나리오 식별

이러한 유형의 시나리오를 만들 때는 무엇이 컨트롤러로 하여금 컨트롤 액션을 제공하도록(혹은 제공하지 않도록) 만들었는지를 설명하기 위해 UCA에서 시작하여 역으로 작업한다. 일반적으로 UCA를 유발하는 시나리오에는 다음과 같은 내용이 포함될 수 있다.

- 컨트롤러와 관련된 고장(failure)(물리적 컨트롤러의 경우)
  - 컨트롤러 자체의 물리적 고장
  - 전원 장애
  - 기타
- 부적절한 컨트롤 알고리즘
  - 특정 컨트롤 알고리즘의 구현의 결함
  - 특정 컨트롤 알고리즘 자체의 결함
  - 특정 컨트롤 알고리즘이 시간이 지나면서 변경되거나 성능저하로 부적합해짐

- Unsafe control input
  - UCA received from another controller (already addressed when considering UCAs from other controllers)
- Inadequate process model
  - Controller receives incorrect feedback/information
  - Controller receives correct feedback/information but interprets it incorrectly or ignores it
  - Controller does not receive feedback/information when needed (delayed or never received)
  - Necessary controller feedback/information does not exist

To create scenarios that involve unsafe control actions, we must consider factors shown in *Figure 2.18* starting with the unsafe controller behavior that caused the UCA.

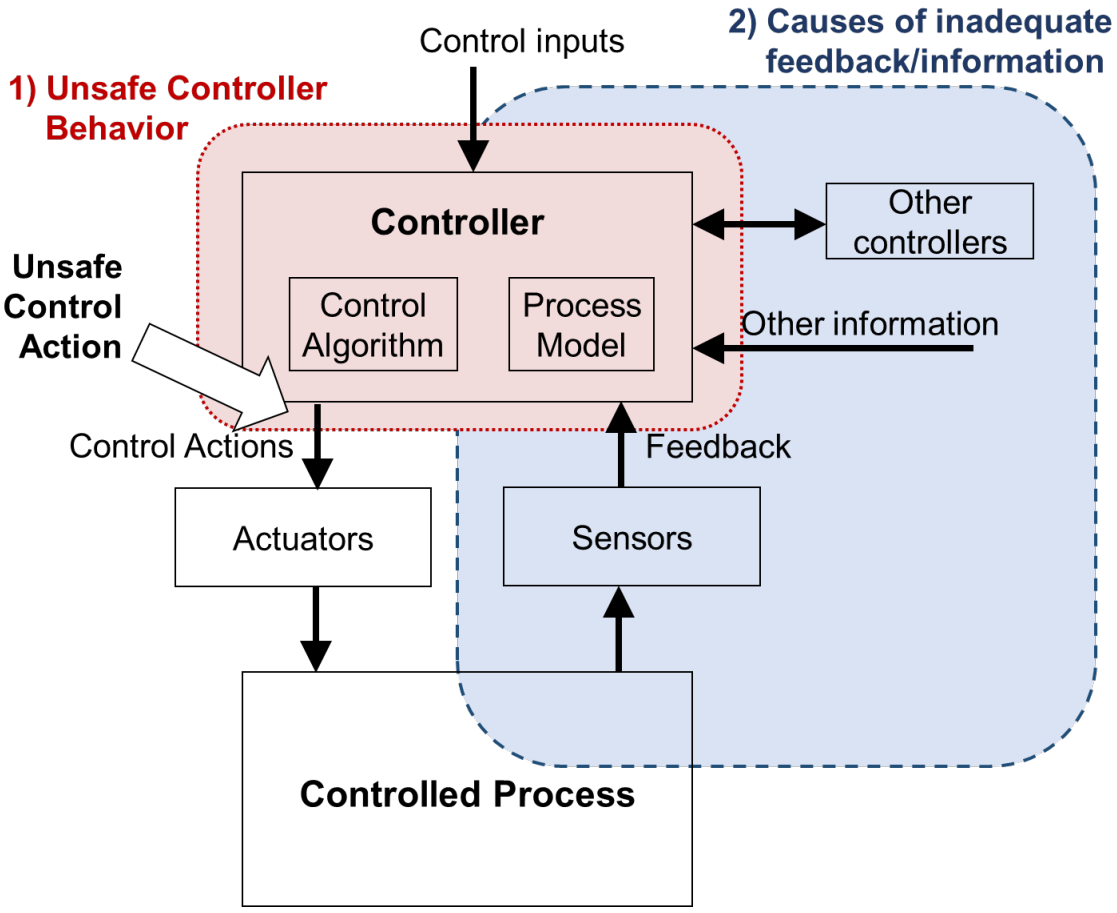


Figure 2.18: Unsafe Control Actions can be caused by (1) unsafe controller behavior and (2) inadequate feedback and other inputs

- 안전하지 않은 컨트롤 입력
  - 다른 컨트롤러에서 받은 UCA(다른 컨트롤러의 UCA를 고려할 때 이미 작성됨)
- 부적절한 프로세스 모델
  - 컨트롤러가 잘못된 피드백/정보를 수신함
  - 컨트롤러가 올바른 피드백/정보를 받았지만 잘못 해석하거나 무시함
  - 피드백/정보가 필요한 상황이지만 컨트롤러가 이를 수신하지 못함(지연되거나 수신되지 않음)
  - 필요한 컨트롤러 피드백/정보가 존재하지 않음

UCA를 유발하는 시나리오를 만들기 위해서는 UCA를 야기하는 안전하지 않은 컨트롤러 동작에서 시작하여 *그림 2.18*에 표시된 요소들을 반드시 고려해야 한다.

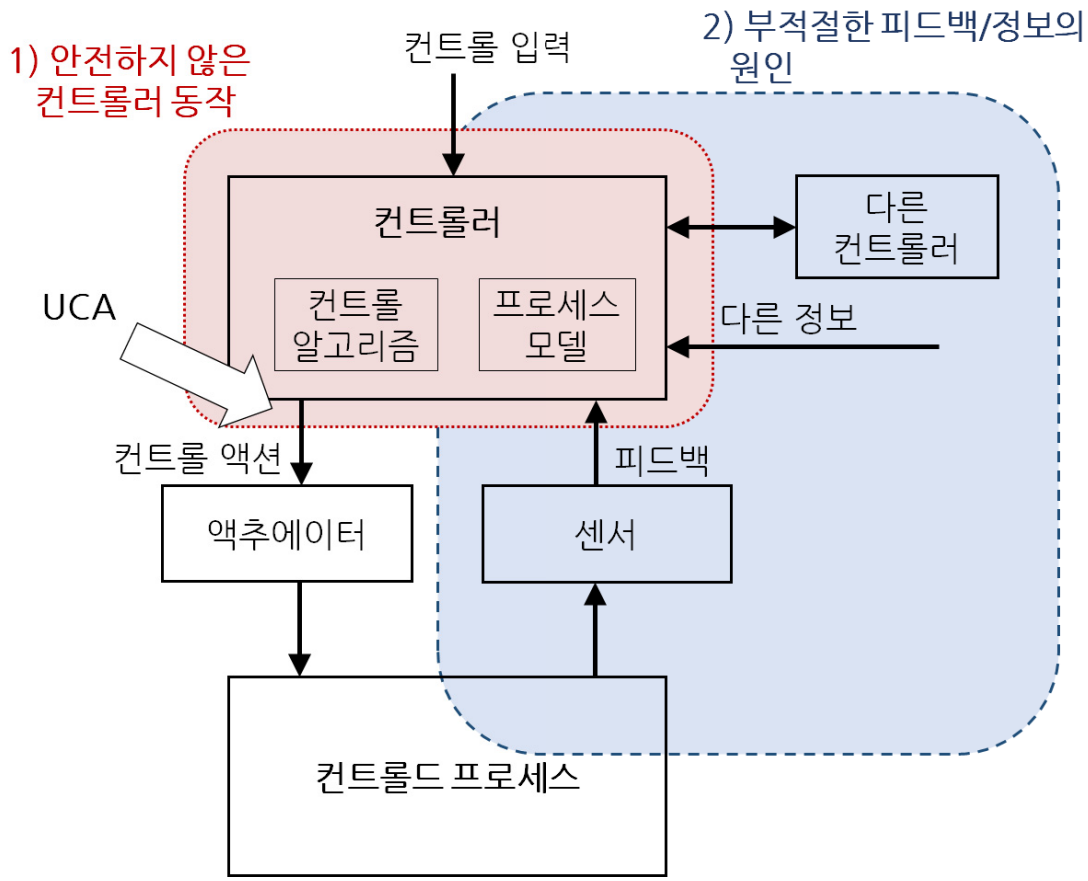


그림 2.18: (1) 안전하지 않은 컨트롤러의 동작, (2) 부적절한 피드백과 다른 입력에 의해 유발될 수 있는 UCA

1) Unsafe controller behavior

There are four general reasons why a controller might provide (or not provide) a control action that is unsafe:

- Failures involving the controller (for physical controllers)
- Inadequate control algorithm
- Insafe control input (from another controller)
- Inadequate process model

For physical controllers, a UCA may occur due to a failure related the controller. For example, the BSCU may not provide the brake command because the BSCU controller fails, because the BSCU power fails, etc. To identify these scenarios, start with a UCA, identify the controller, and identify physical failures related to the controller that can explain the UCA. For example:

> UCA-1: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]
>
> Scenario 1 for UCA-1: The BSCU Autobrake physical controller fails during landing roll when BSCU is armed, causing the Brake control action to not be provided [UCA-1]. As a result, insufficient deceleration may be provided upon landing [H-4.1]

An inadequate control algorithm can also cause a UCA. A control algorithm specifies how control actions are selected based on the controller's process model, previous control inputs and outputs, and other factors. For human controllers, the control algorithm is sometimes called decision-making and it may be shaped by different factors like training, procedures, and past experience.

To identify these scenarios, start with a UCA and identify how the control algorithm may cause the UCA. For example:

> BSCU Autobrake example:
>
> UCA-3: BSCU Autobrake provides the Brake control action too late (>TBD seconds) after touchdown [H-4.1]
>
> Scenario 1 for UCA-3: The aircraft lands, but processing delays within the BSCU result in the Brake control action being provided too late [UCA-3]. As a result, insufficient deceleration may be provided upon landing [H-4.1]
>
> Human Crew example:
>
> Crew-UCA-1: Crew does not provide BSCU Power Off when abnormal WBS behavior occurs [H-4.1, H-4.4]
>
> Scenario 1 for Crew-UCA-1: Abnormal WBS behavior occurs and a BSCU fault indication is provided to the crew. The crew does not power off the BSCU [Crew-UCA-1] because the operating procedures did not specify that the crew must power off the BSCU upon receiving a BSCU fault indication.

In general, flaws in the control algorithm can stem from:

- Flawed implementation of the specified control algorithm
- The specified control algorithm is flawed
- The specified control algorithm becomes inadequate over time due to changes or degradation

The specific reasons for each of these types of flaws may be dependent on the application being studied.

1) 안전하지 않은 컨트롤러 동작

컨트롤러가 안전하지 않은 컨트롤 액션을 제공할(또는 제공하지 않을) 이유에는 크게 네 가지가 있다.

- 컨트롤러와 관련된 실패(물리적 컨트롤러의 경우)
- 부적절한 컨트롤 알고리즘
- 안전하지 않은 컨트롤 입력(다른 컨트롤러로부터)
- 부적절한 프로세스 모델

물리적 컨트롤러의 경우, 컨트롤러와 관련된 실패(failure)로 인해 UCA가 발생할 수 있다. 예를 들어 BSCU 전원 장애 또는 BSCU 컨트롤러 고장(fail) 때문에 BSCU가 브레이크 명령을 제공하지 않을 수 있다. 이러한 시나리오를 식별하려면 UCA에서 시작하여, 컨트롤러를 식별하고 UCA를 설명할 수 있는 컨트롤러와 관련된 물리적 실패를 식별해야 한다. 예를 들어:

> UCA-1: BSCU 오토브레이크가 BSCU 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 제공하지 않는다 [H-4.1]
>
> UCA-1에 대한 시나리오1: BSCU가 작동개시 중일 때 착륙활주 중에 BSCU 오토브레이크의 물리적 컨트롤러 고장으로 인해 브레이크 컨트롤 액션이 제공되지 않는다[UCA-1]. 이로 인해 착륙 시 감속이 불충분할 수 있다 [H-4.1]

부적절한(inadequate) 컨트롤 알고리즘도 UCA를 유발할 수 있다. 컨트롤 알고리즘은 컨트롤러의 프로세스 모델, 이전 컨트롤 입·출력 및 다른 요인에 기반하여 어떤 컨트롤 액션이 선택될지를 결정한다. 휴먼 컨트롤러의 경우 이런 컨트롤 알고리즘을 의사 결정이라고 부르기도 하며, 교육, 절차, 과거의 경험과 같은 서로 다른 요소에 의해 형성될 수 있다.

이러한 시나리오를 식별하려면 UCA에서 시작하여 컨트롤 알고리즘이 어떻게 UCA를 야기하는지 식별해야 한다. 예를 들면:

> BSCU 오토브레이크의 예제:
>
> UCA-3: BSCU 오토브레이크가 터치다운 후 너무 늦게(>TBD초) 브레이크 컨트롤 액션을 제공한다 [H-4.1]
>
> UCA-3에 대한 시나리오1: 항공기가 착륙하지만 BSCU 내의 처리 지연으로 인해 브레이크 컨트롤 액션이 너무 늦게 제공된다 [UCA-3]. 이로 인해 착륙 시 감속이 불충분할 수 있다 [H-4.1]
>
> 승무원의 예:
>
> Crew-UCA-1: 비정상적인 WBS 동작이 발생할 때 승무원이 BSCU 전원끄기를 제공하지 않는다 [H-4.1, H-4.4, H-7]
>
> Crew-UCA-1에 대한 시나리오1: 비정상적인 WBS 동작이 발생하고 승무원에게 BSCU 결함 표시(fault indication)가 제공된다. 승무원이 BSCU 결함 표시를 수신하면 반드시BSCU의 전원을 꺼야 한다고 운영 절차에 명시하지 않았기 때문에 승무원은 BSCU 전원을 끄지 않는다 [Crew-UCA-1]

일반적으로 컨트롤 알고리즘의 결함(flaw)은 다음에서 기인할 수 있다.

- 특정 컨트롤 알고리즘 구현의 결함
- 특정 컨트롤 알고리즘 자체의 결함
- 특정 컨트롤 알고리즘이 시간이 지나면서 변경되거나 성능저하로 부적합해짐

One common control algorithm flaw occurs when the control algorithm assumes that previous control actions have been executed properly. This flaw is especially relevant when there is no feedback to indicate whether the control action was successful. For example, the BSCU Autobrake may not provide the Brake control action because it incorrectly assumes that a previous Brake control action was successful and that the aircraft was already braking.

To include scenarios related to security, one additional possibility needs to be considered here: identify if and how the control algorithm flaw could be introduced by an adversary.

Unsafe control inputs from other controllers can also cause UCAs. These can be found during the previous step when identifying Unsafe Control Actions for other controllers.

Finally, inadequate process models can cause Unsafe Control Actions. As explained above, process models represent the controller's internal beliefs that are used by the control algorithm to determine control actions. Process model flaws occur when a controller's process model does not match reality. Process model flaws may occur because:

- Controller receives incorrect feedback/information
- Controller receives correct feedback/information but interprets it incorrectly or ignores it
- Controller does not receive feedback/information when needed (delayed or never received)
- Necessary controller feedback/information does not exist

These problems could arise in many different ways depending on the application. Incorrect feedback/information might be received by the controller, including conflicting information that cannot be resolved or conflicts that may be resolved incorrectly. Correct feedback/information can be received but ignored because the controller is disabled, turned off, busy with another task, is missing a process model with the necessary conditions to update, or some other reason. Controller interpretation problems may occur if the process model is updated incorrectly, the wrong process model is updated, the feedback/information was thought to represent something else, or other errors. Feedback/information may not be received when needed if the feedback is never received—especially if the controller may assume a default value in lieu of feedback—or if feedback is delayed, including information received out of order. Finally, the necessary feedback/information may not exist in the control structure or in the design, resulting in an inadequate process model.

To identify these scenarios, start with a UCA and identify the controller process models that could cause the UCA. Consider beliefs about current states or modes, previous states, capabilities, dynamic behavior[10], previous behaviors or actions, future states or behaviors (predictions), beliefs about the process currently being controlled, other controlled processes, other controllers in the system (especially beliefs needed for coordination), actuators, sensors, or other relevant aspects of the system or environment.

Once the relevant process models that can cause a UCA have been identified, identify how the process models might occur due to received feedback or other information (or lack thereof).

[10] Tuning-related issues in control theory can be captured here. Process models may include controller beliefs about dynamic characteristics of a controlled process, and tuning problems can occur when those beliefs are incorrect. For example, proportional, integral, and derivative terms in PID controller represent beliefs about the dynamics of the controlled process. If those beliefs are incorrect, stability or other issues may occur and the control algorithm may produce unsafe control actions as a result.

이러한 유형의 각 결함에 대한 구체적인 이유는 적용 분야에 따라 다를 수 있다.

한 가지 흔한 컨트롤 알고리즘 결함은 컨트롤 알고리즘에서 이전 컨트롤 액션이 적절히 실행되었다고 가정하여 발생한다. 이 결함은 컨트롤 액션이 성공적으로 수행되었는지 여부를 나타내는 피드백이 없는 경우에 특히 발생한다. 예를 들면, BSCU 오토브레이크는 이전의 브레이크 컨트롤 액션이 성공적으로 수행되어 항공기에 이미 브레이크가 걸렸다고 잘못 가정함에 따라 브레이크 컨트롤 액션을 제공하지 않을 수 있다.

보안과 관련된 시나리오를 포함시키려면 한 가지 가능성만 추가적으로 고려하면 된다. 컨트롤 알고리즘의 결함이 공격자(adversary)에 의해 유입된 것인지와 그 방법을 식별하는 것이다.

다른 컨트롤러로부터의 안전하지 않은 컨트롤 입력으로 인해 UCA가 발생할 수도 있다. 이는 이전 단계에서 다른 컨트롤러의 UCA를 식별할 때 찾아질 수도 있다.

마지막으로 부적절한 프로세스 모델로 인해 언세이프 컨트롤 액션이 발생할 수도 있다. 위에서 설명한 것처럼 프로세스 모델은 컨트롤 알고리즘이 컨트롤 액션을 결정하기 위한 컨트롤러 내부의 믿음(belief)을 나타낸다. 프로세스 모델의 결함(flaw)은 컨트롤러의 프로세스 모델이 실제와 일치하지 않을 때 발생한다. 프로세스 모델의 결함은 다음과 같은 이유로 발생할 수 있다.

- 컨트롤러가 잘못된 피드백/정보를 수신함
- 컨트롤러가 올바른 피드백/정보를 받았지만 잘못 해석하거나 무시함
- 피드백/정보가 필요한 상황이나 컨트롤러가 이를 수신하지 못함(지연되거나 수신되지 않음)
- 필요한 컨트롤러의 피드백/정보가 존재하지 않음

이러한 문제는 적용 분야에 따라 다양한 방법으로 발생할 수 있다. 컨트롤러가 잘못된 피드백/정보를 수신할 수 있는데, 모순되는 정보를 수신하여 해결할 수 없거나 모순된 정보를 잘못 해결할 수도 있다. 컨트롤러가 올바른 피드백/정보를 받았음에도 불구하고 컨트롤러가 비활성화되었거나, 전원이 꺼져 있거나, 다른 작업으로 바쁘거나, 프로세스 모델이 누락되었거나(프로세스 모델의 업데이트가 필요한 상황), 그 밖의 다른 이유로 인하여 피드백/정보가 무시될 수도 있다. 만일 프로세스 모델이 잘못 업데이트 되거나, 잘못된 프로세스 모델이 업데이트 되거나, 피드백/정보가 다른 무언가로 간주되거나, 다른 오류로 인해 컨트롤러의 해석 문제가 발생할 수도 있다. 피드백이 수신되지 않거나(특히 컨트롤러가 피드백 대신 기본값을 가질 수 있는 경우) 피드백이 지연되는 경우, 정보가 순서에 어긋나게 수신되는 경우 등 피드백/정보가 필요하지만 수신하지 못할 수도 있다. 마지막으로 필요한 피드백/정보가 컨트롤 스트럭처 또는 설계에 존재하지 않아서 부적절한 프로세스 모델이 발생할 수도 있다.

이러한 시나리오를 식별하려면 UCA에서 시작하여 UCA를 유발할 수 있는 컨트롤러 프로세스 모델을 식별해야 한다. 현재 상태 또는 모드, 이전 상태, 기능, 동적 행동(dynamic behavior)[10], 이전 행동이나 동작, 미래 상태나 행동(예측)에 대한 믿음을 고려해야 하며 현재 제어되는 프로세스, 다른 컨트롤드 프로세스, 시스템의 다른 컨트롤러(특히 조정을 위해 필요한 믿음), 액추에이터, 센서 또는 시스템이나 환경과 관련이 있는 다른 측면에 대한 믿음도 고려해야 한다.

UCA를 유발할 수 있는 관련 프로세스 모델이 식별되면, 수신된 피드백 또는 다른 정보(또는 피드백

10 컨트롤 이론(control theory)에서 튜닝(tuning) 관련 사항이 포함될 수 있다. 프로세스 모델에는 컨트롤드 프로세스의 동적 특성에 대한 컨트롤러 믿음이 포함될 수 있으며, 이러한 믿음이 올바르지 않을 때 튜닝 문제가 발생할 수 있다. 예를 들어 PID 컨트롤러의 비례, 적분 및 미분항은 컨트롤드 프로세스의 역학 관계에 대한 믿음을 나타낸다. 만일 이러한 믿음이 올바르지 않으면 안정성이나 다른 문제가 발생할 수 있으며 결과적으로 컨트롤 알고리즘이 UCA를 만들 수 있다.

Example:

UCA-2: BSCU Autobrake does not provide the Brake control action during landing roll when the BSCU is armed [H-4.1]

Controller process model (belief) that could cause the UCA: Controller believes the aircraft has already stopped on ground

Controller receives correct feedback but interprets it incorrectly: Wheel speed signals may momentarily reach zero during anti-skid operation, causing flawed process model

Scenario 1 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft has already come to a stop. This flawed process model will occur if the received feedback momentarily indicates zero speed during landing roll. The received feedback may momentarily indicate zero speed during anti-skid operation, even though the aircraft is not stopped.

Controller process model (belief) that could cause the UCA: Aircraft is in flight

Controller does not receive information when needed: Touchdown indication is not received

Scenario 2 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft is in the air and has not touched down. This flawed process model will occur if the touchdown indication is not received upon touchdown. <Why? This scenario needs to be finished.>

Any scenarios that involve inadequate feedback/information must be completed to explain why the feedback/information might be inadequate. Inadequate feedback and information cannot be prevented without understanding the reasons for it.

2) Causes of inadequate feedback and information

Whenever a scenario identifies feedback or information (or lack thereof) that can cause a UCA, we need to examine where the feedback/information comes from to explain what could cause those problems. Feedback comes from the controlled process (usually via sensors) and other information may come from other processes, other controllers, or other sources in the system or the environment.

In general, scenarios related to inadequate feedback and information might involve:

- Feedback or information not received
  - Feedback/info sent by sensor(s) but not received by controller
  - Feedback/info is not sent by sensor(s) but is received or applied to sensor(s)
  - Feedback/info is not received or applied to sensor(s)
  - Feedback/info does not exist in control structure or sensor(s) do not exist
- Inadequate feedback is received
  - Sensor(s) respond adequately but controller receives inadequate feedback/info
  - Sensor(s) respond inadequately to feedback/info that is received or applied to sensor(s)
  - Sensor(s) are not capable or not designed to provide necessary feedback/info

Scenarios involving feedback/info that is sent but not received or received inadequately may be caused by transmission errors, lost communication, delays in communication (including feedback/info sent but received in a different order), and other problems. Scenarios with an inadequate sensor response or no response may be caused by sensor failures, loss of power to the sensor, inaccuracies in sensor operation or measurement, sensor errors or misbehaviors, delays in sensor response, incorrect

/정보의 부족)로 인해 프로세스 모델이 어떻게 발생할 수 있는지 식별한다.

예:

UCA-1: BSCU 오토브레이크가 BSCU 작동개시 중일 때 착륙활주 동안 브레이크 컨트롤 액션을 제공하지 않는다 [H-4.1]

UCA를 일으킬 수 있는 컨트롤러 프로세스 모델(믿음): 컨트롤러는 항공기가 이미 지상에 정지해 있다고 생각한다.

컨트롤러가 정확한 피드백을 받았지만 잘못 해석한다: 미끄럼 방지 작동 중에 휠 속도 신호가 일시적으로 제로가 되어 프로세스 모델의 결함을 유발한다.

UCA-1에 대한 시나리오 2: BSCU가 작동개시 중일 때 항공기가 착륙활주를 시작한다. BSCU가 항공기가 이미 정지했다고 잘못 믿기 때문에 BSCU가 브레이크 컨트롤 액션을 제공하지 않는다 [UCA-1]. 이러한 결함이 있는 프로세스 모델은 착륙활주 중 순간적으로 속도가 "0"라는 피드백을 수신하면 발생하게 된다. 수신된 피드백은 항공기가 정지되지 않아도 미끄럼 방지 동작(anti-skid operation) 중에 순간적으로 속도가 0으로 나타날 수 있다.

UCA를 일으킬 수 있는 컨트롤러 프로세스 모델(믿음): 항공기가 비행중임

컨트롤러가 정보를 필요한 시점에 받지 못함: 터치다운 알림(indication)이 수신되지 않음

UCA-1에 대한 시나리오 3: BSCU가 작동개시 중일 때 항공기가 착륙한다. BSCU는 항공기가 공중에 있으며 아직 터치다운 하지 않았다고 잘못 믿기 때문에 BSCU가 브레이크 컨트롤 액션을 제공하지 않는다 [UCA-1]. 만일, 터치다운 알림이 터치다운 시 수신되지 않으면 프로세스 모델에 결함이 발생한다. 〈왜 그런가? 이 시나리오는 더 마무리 되어야 한다.〉

피드백/정보가 부적절한 이유를 추가 설명하여 모든 시나리오를 완성해야 한다. 피드백과 정보에 대한 부적절한 이유를 이해하지 않고는 이를 예방할 수 없다.

2) 부적절한 피드백 및 정보의 원인

UCA를 야기할 수 있는 피드백/정보(또는 피드백/정보 부족)를 식별할 때마다 이러한 문제를 일으킬 수 있는 원인을 설명하기 위해 피드백/정보의 출처를 조사할 필요가 있다. 피드백은 컨트롤드 프로세스로부터(일반적으로 센서를 통해) 발생하며 다른 정보는 다른 프로세스, 다른 컨트롤러 또는 시스템 또는 환경 안에 있는 다른 위치에서 발생한다.

일반적으로 부적절한 피드백 및 정보와 관련된 시나리오에는 다음이 포함될 수 있다.

- 수신되지 않는 피드백 또는 정보
  - 센서가 피드백/정보를 보냈지만 컨트롤러가 수신하지 않음
  - 센서에 피드백/정보가 수신되거나 적용되었지만, 센서가 이를 보내지 않음
  - 센서가 피드백/정보를 수신하지 않거나 적용하지 않음
  - 피드백/정보가 컨트롤 스트럭처에 존재하지 않거나 센서가 존재하지 않음
- 부적절한 피드백 수신
  - 센서가 적절하게 응답하였지만 컨트롤러가 부적절한 피드백/정보를 수신함
  - 센서에 수신되거나 적용된 피드백/정보에 대하여 센서가 부적절하게 반응함
  - 센서가 필요한 피드백/정보를 제공하지 못하거나 설계에 반영되지 않음

configuration, degradation or changes to the sensor over time, unanticipated conditions in the sensor environment, or other problems. In addition, sensors may not be capable of providing the necessary feedback due to design errors, specification flaws, incorrect assumptions about the controlled process, sensors measuring the wrong conditions, sensors that report correct but misleading information (such as zero wheel speed when wheels are locked but the aircraft is moving), or other problems.

We can finish scenarios that involve feedback/info by identifying why that feedback/info might be received given the actual true state of the system (specified in the UCA context).

For example, let's finish Scenario 2 for UCA-2 above:

True state from UCA context: Aircraft is in landing roll (See Scenario 2 above)

Information received: Touchdown indication is not received upon touchdown (see Scenario 2 above)

How this could happen given the true state: Reported wheel speed is insufficient, reported weight on wheels is insufficient, wheel speed or weight on wheels indications are delayed, etc.

Scenario 2 for UCA-2: The BSCU is armed and the aircraft begins landing roll. The BSCU does not provide the Brake control action [UCA-2] because the BSCU incorrectly believes the aircraft is in the air and has not touched down. This flawed process model will occur if the touchdown indication is not received upon touchdown. The touchdown indication may not be received when needed if any of the following occur:

- Wheels hydroplane due to a wet runway (insufficient wheel speed)
- Wheel speed feedback is delayed due to filtering used
- Conflicting air/ground indications due to crosswind landing
- Failure of wheel speed sensors
- Failure of air/ground switches
- Etc.

As a result, insufficient deceleration may be provided upon landing [H-4.1]

To include causes related to security, only one additional possibility needs to be considered here: identify how the specified feedback and other information could be affected by an adversary. More specifically: how could they be injected, spoofed, tampered, intercepted, or disclosed to an adversary? For example, the following causes might be added to Scenario 2 above if security is included:

- Adversary spoofs feedback indicating insufficient wheel speed
- Wheel speed feedback is delayed due to adversary performing DoS attack
- Correct wheel speed feedback is intercepted and blocked by an adversary
- Adversary disables power to the wheel speed sensors

b) Identifying scenarios in which control actions are improperly executed or not executed

Hazards can be caused by UCAs, but they can also be caused without a UCA if control actions are improperly executed or not executed. To create these scenarios, we must consider factors that affect the control path as well as factors that affect the controlled process, as shown in *Figure 2.19*.

피드백/정보를 보냈지만 수신하지 못하거나 부적절하게 수신한 시나리오는 전송 오류, 통신 두절, 통신 지연(보낸 순서와 다른 순서로 받은 피드백/정보 포함) 및 다른 문제로 인하여 발생할 수 있다. 센서의 부적절한 응답이나 응답이 없는 시나리오는 센서 실패, 센서 전원 손실, 부정확한 센서의 작동이나 측정, 센서 오류 또는 오작동, 센서 응답 지연, 잘못된 구성, 시간이 지나면서 발생하는 센서의 성능저하나 변경, 예상치 못한 센서 환경 조건 또는 기타 다른 문제로 인하여 발생할 수 있다. 그 뿐만 아니라, 센서는 설계 오류, 사양 결함, 컨트롤드 프로세스에 대한 잘못된 가정, 잘못된 상태를 측정하는 센서, 정확하지만 오해를 일으킬 수 있는 정보를 주는 센서(항공기가 움직이고 있지만 휠이 잠겨 휠 속도가 0인 것처럼) 또는 다른 문제로 인하여 필요한 피드백을 제공하지 못할 수도 있다.

시스템의 실제 상태(UCA 컨텍스트에 명세된)에서 왜 그러한 피드백/정보를 수신하는지 이유를 식별하여 피드백/정보와 관련된 시나리오를 마무리 지을 수 있다.

예를 들어, 위의 UCA-1에 대한 시나리오 3을 완성해 보자.

UCA 컨텍스트에서의 실제 상태: 항공기가 착륙활주하고 있음(위의 시나리오 3 참고)

수신 정보: 터치다운할 때 터치다운 알림이 수신되지 않음(위의 시나리오 3 참고)

이것이 실제 상황에서 일어날 수 있는 경우: 보고된 휠 속도가 충분하지 않거나, 착륙감지(weight on wheels)가 불충분하거나, 휠 속도 또는 착륙감지 표시 장치가 지연되는 등

UCA-1에 대한 시나리오 3: BSCU가 작동개시 중일 때 항공기가 착륙한다. BSCU는 항공기가 공중에 있으며 아직 터치다운 하지 않았다고 잘못 믿기 때문에 BSCU가 브레이크 컨트롤 액션을 제공하지 않는다[UCA-1]. 만일 터치다운 알림이 터치다운 시 수신되지 않으면 프로세스 모델에 결함이 발생한다. 터치다운 알림은 다음 중 어떤 하나라도 발생하면 수신되지 않을 수 있다.

- 젖은 활주로로 인해서 휠 수막현상(hydroplaning) 발생(휠 속도 불충분)
- 필터링으로 인해 휠 속도 피드백 지연
- 측풍착륙(crosswind landing)으로 인해 상충되는 에어/그라운드 알림
- 휠 속도 센서의 고장
- 에어/그라운드 스위치의 고장
- 기타

결과적으로 착륙 시 불충분하게 감속될 수 있다 [H-4.1]

보안과 관련된 원인을 포함하려면 여기에 다음과 같은 한 가지 가능성만 추가적으로 고려하면 된다. 공격자가 어떻게 특정 피드백과 다른 정보에 영향을 줄 수 있는지를 식별하는 것이다. 더 구체적으로는 공격자가 어떻게 인젝션(injection), 스푸핑(spoofing), 변조, 도청 또는 감청할 수 있는가이다. 예를 들어 위의 시나리오 3에 보안을 고려한 경우, 다음과 같은 원인이 추가될 수 있다.

- 불충분한 휠 속도가 표시되도록 공격자가 피드백을 스푸핑(spoofing)한다.
- 공격자의 DoS 공격으로 인해 휠 속도 피드백이 지연된다.
- 공격자가 올바른 휠 속도 피드백을 가로채고 차단한다.
- 공격자가 휠 속도 센서의 전원 공급을 차단한다.

b) 컨트롤 액션이 부적절하게 실행되거나 실행되지 않는 시나리오의 식별

위험은 UCA로 인해서 발생할 수도 있지만, 만약 컨트롤 액션이 부적절하게 실행되거나 실행되지 않으면 UCA 없이도 발생할 수 있다. 이러한 시나리오를 만들기 위해서는 *그림 2.19*와 같이 컨트롤 경로에 영향을

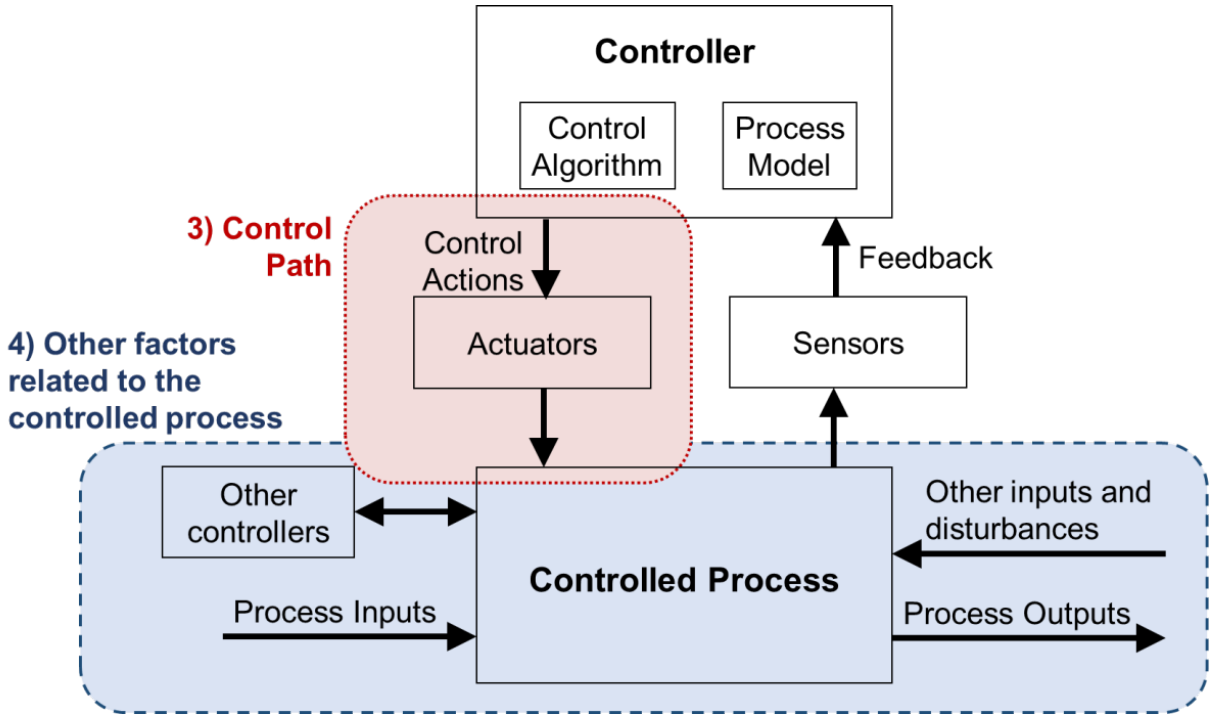


Figure 2.19: Generic control loop illustrating 1) the control path and 2) other factors that can affect the controlled process

1) Scenarios involving the control path

The control path transfers control actions to the controlled process. The control path may consist of a simple actuator, may involve a series of actuators, or it may transfer control actions through a complex network with switches, routers, satellites, or other equipment. Regardless of how it is implemented, we must identify how problems along this path can cause control actions to be improperly executed or not executed.

In general, scenarios involving the control path might include:

- Control action not executed
  - Control action is sent by controller but not received by actuator(s)
  - Control action is received by actuator(s) but actuator(s) do not respond
  - Actuator(s) responds but the control action is not applied to or received by the controlled process
- Control action improperly executed
  - Control action is sent by controller but received improperly by actuator(s)
  - Control action is received correctly by actuator(s) but actuator(s) respond inadequately
  - Actuator(s) respond adequately, but the control action is applied or received improperly at the controlled process
  - Control action is not sent by controller, but actuators or other elements respond as if it had been sent

Scenarios with control actions that are sent but improperly received or not received may be caused by delays in communication (including control actions sent but received in a different order), transmission errors, lost communication, and other problems. Scenarios with an improper actuator response or no response may be caused by actuator failures, loss of power to the actuator, inaccuracies in actuator operation, actuator errors or misbehaviors, delays in actuator response, other commands received by the actuator (including potentially conflicting commands from other controllers), incorrect

미치는 요인뿐만 아니라 컨트롤드 프로세스에 영향을 미치는 요인도 고려해야 한다.

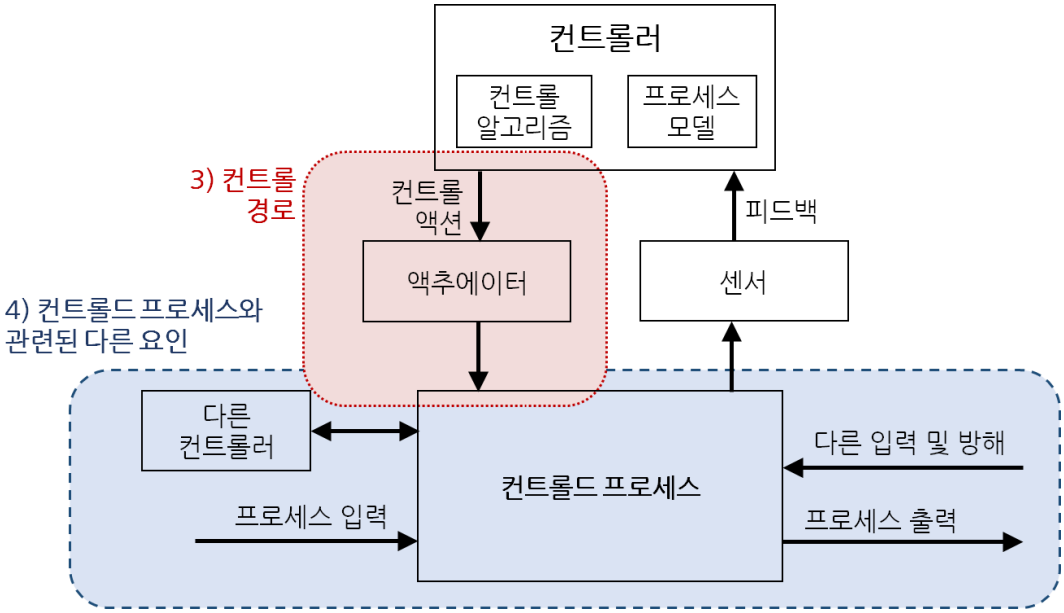


그림 2.19: 1) 컨트롤 경로와 2) 컨트롤드 프로세스에 영향을 줄 수 있는 다른 요인들을 표현한 일반적인 컨트롤 루프

1) 컨트롤 경로와 관련된 시나리오

컨트롤 경로는 컨트롤 액션을 컨트롤드 프로세스로 전송한다. 컨트롤 경로는 단순한 액추에이터 하나로 구성되거나, 일련의 액추에이터들을 포함하거나, 스위치, 라우터, 위성 또는 여러 장비를 거치는 복잡한 네트워크를 통해 컨트롤 액션을 전송할 수 있다. 구현 방법에 관계없이 우리는 경로를 따라 발생하는 문제로 인해 컨트롤 액션이 부적절하게 실행되거나 실행되지 않게 되는지를 식별해야 한다.

일반적으로 컨트롤 경로와 관련된 시나리오에는 다음이 포함될 수 있다.

- 컨트롤 액션이 실행되지 않음
  - 컨트롤러가 컨트롤 액션을 전송하였으나, 액추에이터가 수신하지 못함
  - 액추에이터가 컨트롤 액션을 수신했지만, 액추에이터가 반응하지 않음
  - 액추에이터가 반응하였으나, 컨트롤 액션이 컨트롤드 프로세스에 적용되지 않거나 수신되지 않음
- 컨트롤 액션이 부적절하게 실행됨
  - 컨트롤러가 컨트롤 액션을 전송하였으나, 액추에이터가 부적절하게 수신함
  - 액추에이터가 컨트롤 액션을 정확히 수신하였으나, 액추에이터가 부적절하게 반응함
  - 액추에이터가 적절하게 반응하였지만, 컨트롤 액션이 컨트롤드 프로세스에 부적절하게 적용되거나 부적절하게 수신됨
  - 컨트롤러가 컨트롤 액션을 전송하지 않았지만, 액추에이터나 다른 요소(element)가 마치 컨트롤 액션이 전송된 것처럼 반응함

컨트롤 액션이 전송되었지만 부적절하게 수신되었거나 수신되지 않는 시나리오는 전송 지연(컨트롤 액션이 전송됐으나 다른 순서로 수신한 경우 포함), 전송 오류, 통신 두절 및 기타 다른 문제들로 인하여 발생할 수 있다. 액추에이터의 부적절한 반응 또는 반응이 없는 시나리오는 액추에이터 고장, 액추

priority scheme used by the actuator, incorrect configuration, degradation or changes to the actuator over time, unanticipated conditions in the actuator environment, or other problems.

To create these scenarios, start with a control action, identify what improper execution or no execution means for your application, and identify how the control path could contribute to that behavior.

Example:

Control action: BSCU sends Brake command

No execution: Brakes not applied

Improper execution: Insufficient braking applied

Scenario 1: The BSCU sends the Brake command upon landing, but the brakes are not applied due to actuator failure. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 2: The BSCU sends the Brake command upon landing, but insufficient braking is applied due to slow actuator response. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 3: The BSCU sends the Brake command upon landing, but it is not received by the actuator due to a wiring error. As a result, insufficient deceleration may be provided upon landing [H-4.1]

It is also important to consider how control actions may not be sent but actuators or other elements respond as if a control action was sent. These scenarios will be similar to UCAs for control actions that are provided.

Example:

Control action: BSCU does not send Brake command

Improper execution: Brakes are applied during normal takeoff (similar to UCA-2)

Scenario 4: The BSCU does not send Brake command, but the brakes are applied due to hydraulic valve failure. As a result, acceleration may be insufficient during takeoff [H-4.6]

To include security problems, one additional possibility needs to be considered here: identify how the specified control action could be affected by an adversary. More specifically: how could it be injected, spoofed, tampered, intercepted, or disclosed to an adversary? For example:

Scenario 5: The BSCU sends the Brake command, but the brakes are not applied because an adversary executes a denial of service attack that blocks the Brake command. As a result, insufficient deceleration may be provided upon landing [H-4.1]

2) Scenarios related to the controlled process

Even if control actions are transferred or applied to the controlled process, they may not be effective or they may be overridden by other controllers.

In general, scenarios related to the controlled process might include:

- Control action not executed
    - Control action is applied or received by the controlled process but the controlled process does not respond
- Control action improperly executed
    - Control action is applied or received by the controlled process but the controlled process responds improperly

에이터의 전력 손실, 액추에이터 동작의 부정확성, 액추에이터 오류 또는 오작동, 액추에이터 응답 지연, 액추에이터가 수신한 다른 명령(다른 컨트롤러로부터 수신한 모순 가능성이 있는 명령 포함), 액추에이터가 사용하는 잘못된 우선순위 구조, 부정확한 구성, 시간 경과에 따른 액추에이터의 성능저하 또는 변경, 액추에이터 환경의 예기치 않은 조건 또는 기타 다른 문제로 인하여 발생할 수 있다.

이러한 시나리오를 작성하려면 컨트롤 액션에서 시작하여, 부적절한 실행 또는 실행하지 않는 것이 어플리케이션에서 어떤 의미인지를 식별하고, 컨트롤 경로가 해당 동작에 어떻게 기여할 수 있는지 식별해야 한다.

예:

컨트롤 액션: BSCU가 브레이크 명령을 전송한다.

실행하지 않음: 브레이크가 적용되지 않는다.

부적절한 실행: 브레이크가 불충분하다.

시나리오 1: BSCU가 착륙 시 브레이크 명령을 전송하지만, 액추에이터 고장(failure)으로 인해 브레이크가 적용되지 않는다. 결과적으로 착륙 시 감속이 불충분할 수 있다 [H-4.1]

시나리오 2: BSCU가 착륙 시 브레이크 명령을 전송하지만, 액추에이터가 느리게 반응하여 브레이크가 불충분하게 적용된다. 결과적으로 착륙 시 감속이 불충분할 수 있다 [H-4.1]

시나리오 3: BSCU는 착륙 시 브레이크 명령을 전송하지만, 배선 오류로 인해 액추에이터가 수신하지 못한다. 결과적으로 착륙 시 감속이 불충분할 수 있다 [H-4.1]

컨트롤 액션이 전송되지 않았지만 액추에이터나 다른 요소들이 마치 컨트롤 액션을 수신한 것처럼 반응하는 상황도 중요하게 고려해야 한다. 이러한 시나리오는 컨트롤 액션이 제공된 경우의 UCA와 유사한 방식으로 작성할 수 있다.

예:

컨트롤 액션: BSCU는 브레이크 명령을 보내지 않는다.

부적절한 실행: 정상적인 이륙 중 브레이크가 적용된다(UCA-2와 유사).

시나리오 4: BSCU는 브레이크 명령을 보내지 않지만 유압 밸브 고장으로 인하여 브레이크가 적용된다. 그 결과, 이륙 시 가속이 불충분할 수 있다 [H-4.6]

보안 문제를 포함하려면 여기에 한 가지 가능성을 추가적으로 고려하면 된다. 공격자가 어떻게 특정 컨트롤 액션에 영향을 줄 수 있는지를 식별하는 것이다. 더 구체적으로는 어떻게 인젝션(injection), 스푸핑(spoofing), 변조, 도청 또는 감청할 수 있는지를 식별하는 것이다. 예를 들면:

시나리오5: BSCU는 브레이크 명령을 전송하지만, 브레이크가 적용되지 않는다. 왜냐하면 공격자가 브레이크 명령을 차단하는 DoS(Denial of Service) 공격을 실행하기 때문이다. 그 결과, 착륙 시 감속이 불충분할 수 있다 [H-4.1]

2) 컨트롤드 프로세스와 관련된 시나리오

컨트롤 액션이 컨트롤드 프로세스로 전송되거나 적용되더라도 컨트롤 액션이 효과를 발휘하지 못하거나 다른 컨트롤러에 의해 무시될 수(overridden) 있다.

일반적으로 컨트롤드 프로세스와 관련된 시나리오에 다음이 포함될 수 있다.

- 컨트롤 액션이 실행되지 않음

- Control action is not applied or received by the controlled process but the process responds as if the control action had been applied or received

These scenarios may be caused by process inputs that are missing or inadequate (such as inadequate hydraulic pressure, etc.), external or environmental disturbances, component failures, delayed response by the process, errors or misbehaviors in the process, potentially conflicting commands received from other controllers, previous control actions that were received or applied to the controlled process, incorrect priority scheme used by the process, incorrect configuration, degradation or changes to the process or the environment over time, unanticipated or unhandled conditions in the process environment, or other problems.

To create these scenarios, select a control action and identify what factors can affect the controlled process to make the control action ineffective.

Example:

Control action: BSCU sends Brake command

Scenario 6: The BSCU sends Brake command, but the brakes are not applied because the wheel braking system was previously commanded into alternate braking mode (bypassing the BSCU). As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 7: The BSCU sends Brake command, but the brakes are not applied due to insufficient hydraulic pressure (pump failure, hydraulic leak, etc.). As a result, insufficient deceleration may be provided upon landing [H-4.1]

Scenario 8: The BSCU sends Brake command, the brakes are applied, but the aircraft does not decelerate due to a wet runway (wheels hydroplane). As a result, insufficient deceleration may be provided upon landing [H-4.1]

To include security issues, the same additional possibility needs to be considered again: identify how adversaries can interact with the control process to cause the same issues. For example:

Scenario 9: The BSCU sends Brake command, but the brakes are not applied because an adversary injected a command that put the wheel braking system into alternate braking mode. As a result, insufficient deceleration may be provided upon landing [H-4.1]

Tips to prevent common mistakes

The most common mistake is to identify individual causal factors rather than a scenario. For example, you may be tempted to create list of factors like “wheel speed sensor failure”, “wheel speed feedback is delayed”, “loss of power”, etc. The problem with listing individual factors outside the context of a scenario is that it’s easy to overlook how several factors interact with each other, you can overlook non-trivial and non-obvious factors that indirectly lead to UCAs and hazards, and you may not consider how combinations of factors can lead to a hazard. Considering single factors essentially reduces to a FMEA where only single component failures are considered.

- ◦ 컨트롤 액션이 컨트롤드 프로세스에 적용되었거나 수신되었지만 컨트롤드 프로세스가 반응(respond)하지 않음

- 컨트롤 액션이 부적절하게 실행됨
  - ◦ 컨트롤 액션이 컨트롤드 프로세스에 적용되었거나 수신되었지만, 컨트롤드 프로세스가 부적절하게 반응함
  - ◦ 컨트롤 액션이 컨트롤드 프로세스에 적용되지 않았거나 수신되지 않았지만, 컨트롤드 프로세스가 마치 컨트롤 액션이 적용되었거나 수신된 것처럼 반응함

이러한 시나리오들은 누락되거나 부적절한 프로세스 입력(부적절한 유압 압력 등), 외부 또는 환경적 방해(disturbance), 컴포넌트 장애, 프로세스의 응답 지연, 프로세스의 오류 또는 오동작, 다른 컨트롤러로부터 받은 모순 가능성이 있는 명령, 컨트롤드 프로세스에 수신되거나 적용된 이전 컨트롤 액션, 프로세스에 사용된 잘못된 우선순위 구조, 잘못된 구성, 시간 경과에 따른 프로세스나 환경의 저하 또는 변경, 프로세스의 환경에서 예기치 않거나 핸들링되지 못한 조건 또는 기타 다른 문제로 인하여 발생할 수 있다.

이러한 시나리오를 작성하기 위해서는 컨트롤 액션을 선택하고 어떤 요소가 컨트롤드 프로세스에 영향을 주어 컨트롤 액션을 무력하게 만들 수 있는지 파악해야 한다.

예:

컨트롤 액션: BSCU가 브레이크 명령을 전송한다.

시나리오 6: BSCU는 브레이크 명령을 전송하지만, 브레이크가 적용되지 않는다. 왜냐하면 휠 브레이킹 시스템이 이전 명령에 의해 '대체 브레이킹 모드(BSCU를 우회함)'로 변경되었기 때문이다. 그 결과, 착륙 시 감속이 불충분할 수 있다 [H-4.1]

시나리오 7: BSCU가 브레이크 명령을 전송하지만, 브레이크가 적용되지 않는다. 왜냐하면 유압이 부족(펌프 고장, 유압 누수 등)하기 때문이다. 그 결과, 착륙 시 감속이 불충분할 수 있다 [H-4.1]

시나리오 8: BSCU는 브레이크 명령을 전송하고 브레이크가 적용되지만, 항공기가 감속하지 않는다. 왜냐하면 젖은 활주로(바퀴 수막 현상) 때문이다. 그 결과, 착륙 시 감속이 불충분할 수 있다 [H-4.1]

보안 문제를 포함시키기 위해서는 여기에 동일한 추가적인 가능성을 다시 고려해야 한다: 공격자가 어떻게 컨트롤 프로세스와 상호작용하여 동일한 이슈를 유발할 수 있는지를 식별해야 한다. 예를 들어:

시나리오 9: BSCU가 브레이크 명령을 전송하지만, 브레이크가 적용되지 않는다. 왜냐하면 공격자가 바퀴 제동 시스템을 '대체 브레이킹 모드'로 변환하는 명령을 주입(injection)했기 때문이다. 그 결과, 착륙 시 감속이 불충분할 수 있다 [H-4.1]

흔한 실수를 예방하는 팁

가장 흔한 실수는 시나리오가 아닌 개별적인 원인 요소를 도출하는 것이다. 예를 들어 "휠 속도 센서 고장", "바퀴 속도 피드백 지연", "전원 손실" 등과 같은 요인들의 목록을 만드는 유혹에 빠지는 것이다. 시나리오의 컨텍스트에서 벗어나 개별 요인을 나열하는 것의 문제는, 여러 요소가 어떻게 서로 상호작용하는지를 간과하기 쉽고 UCA와 위험을 간접적으로 야기하는 중요하지만 명확하지 않은 요인을 간과할 수 있으며 위험을 초래할 수 있는 요인들이 어떻게 조합되는지를 고려하지 않을 수 있다는 것이다. 한 가지 요소만을 고려하게 되면 필연적으로 단일 컴포넌트의 실패만 고려하는 FMEA로 축소되게 된다.

Inputs and Outputs for Identifying Scenarios

Figure 2.20 summarizes the inputs and outputs for the scenario identification step.

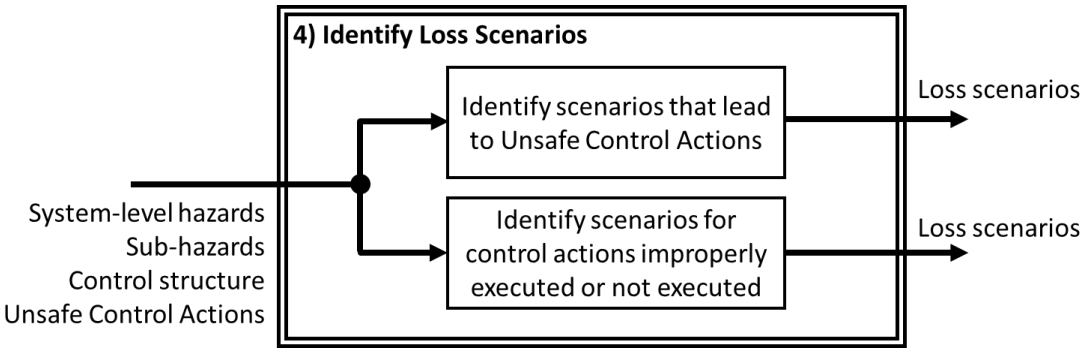


Figure 2.20: Overview of scenario identification

STPA Outputs and Traceability

Figure 2.21 shows the traceability that is maintained between various STPA outputs.

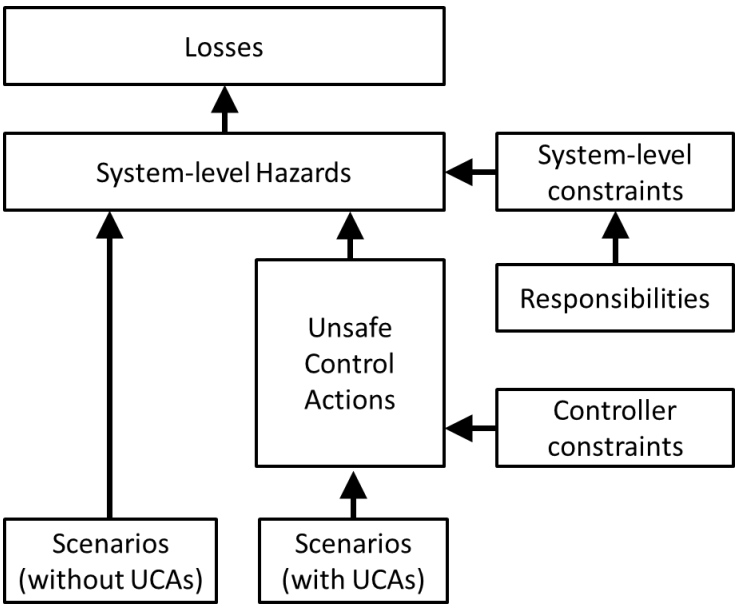


Figure 2.21: Traceability between STPA outputs

The control structure is closely related to every STPA output and therefore it is not explicitly shown in *Figure 2.21* for simplicity. These STPA outputs can be used in many different ways, including:

- Drive the system architecture
- Create executable requirements
- Identify design recommendations
- Identify mitigations and safeguards needed
- Define test cases and create test plans
- Drive new design decisions (if STPA is used during development)
- Evaluate existing design decisions and identify gaps and changes needed (if STPA is used after the design is finished)
- Develop leading indicators of risk
- Design more effective safety management systems

시나리오 식별을 위한 입력 및 출력

*그림 2.20*은 시나리오 식별 단계의 입력과 출력을 요약한 것이다.

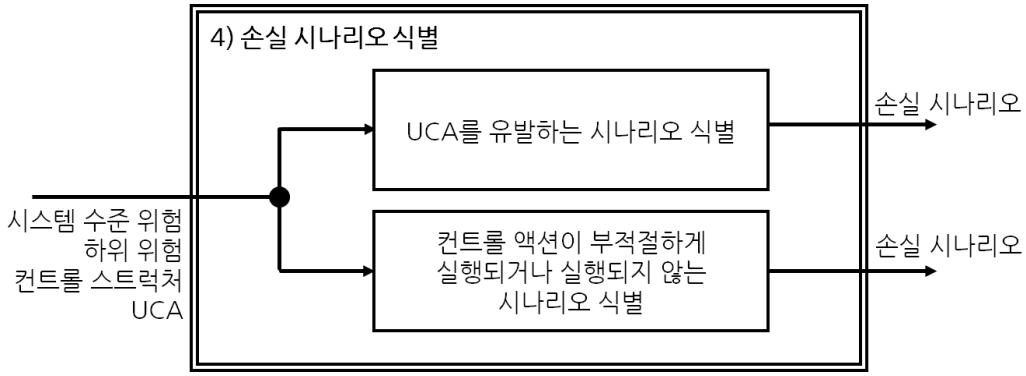


그림 2.20: 시나리오 식별 개요

STPA 결과물 및 추적성

*그림 2.21*은 다양한 STPA 결과물 간에 유지되는 추적성(traceability)을 보여준다.

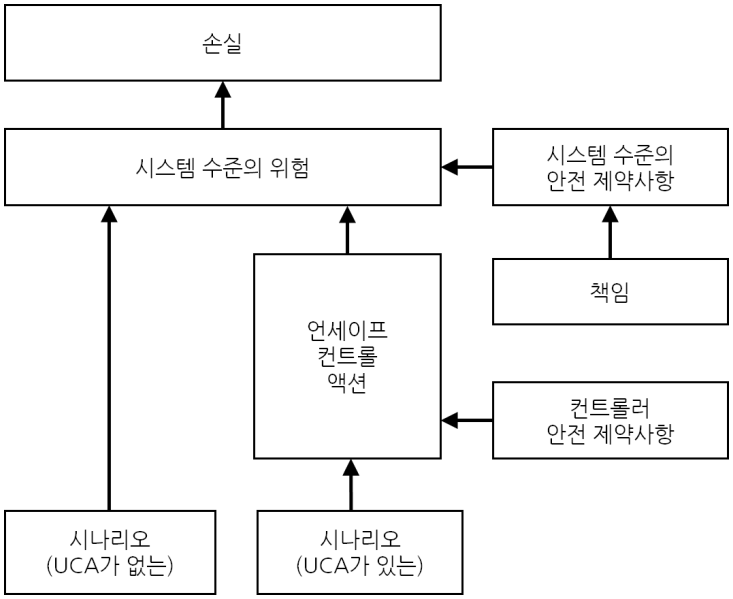


그림 2.21: STPA 결과물 간의 추적성

컨트롤 스트럭처는 모든 STPA 결과물과 밀접한 관련이 있으나 그림을 단순화하기 위하여 *그림 2.21*에는 구체적으로 나타내지 않았다. 이러한 STPA 결과물은 다음과 같은 여러 가지 방법으로 사용할 수 있다.

- 시스템 아키텍처의 추진(drive)
- 실행 가능한 요구사항의 생성
- 설계 권고사항의 식별
- 필요한 완화조치(mitigations) 및 안전장치(safeguards)의 식별

- Etc.

You should now have a basic understanding of how STPA is performed. We suggest that you try the analysis on some of your own engineering problems. The Appendices provide some additional examples that may be useful.

Summary and a Look Forward

This chapter has described the basic approach used in STPA. The next chapter explains how STPA can be applied to the various phases and activities involved in system engineering. Chapter 4 provides additional information and examples for using STPA in workplace safety. Up to this point, the examples all involve technical systems but STPA can be used for any sociotechnical system. Chapter 5 shows how to use STPA in organizational analysis and for emergent, system-wide properties other than safety. Chapter 6 shows how STPA can be used to identify leading indicators of risk. Chapter 7 describes the use of STAMP and STPA to design more effective safety management systems. Finally, Chapter 8 describes what we have learned so far about how to integrate STPA into large organizations and projects.

- 테스트케이스 정의 및 테스트 계획의 생성
- 새로운 설계 결정 추진(개발 과정에서 STPA가 사용되는 경우)
- 기존 설계 결정을 평가하고 차이점을 파악한 후 변경사항 식별(설계가 완료된 후 STPA가 사용되는 경우)
- 리스크 선행지표의 개발
- 보다 효과적인 안전 관리 시스템(safety management systems)의 설계
- 기타

이제 당신은 STPA가 수행되는 방법에 대하여 기본적으로 이해했을 것이다. 당신이 가진 엔지니어링 문제 중 몇 가지를 대상으로 STPA 분석을 시도해 보는 것을 추천한다. 부록에 유용한 추가 예제를 실었다.

요약 및 미리보기

본 장에서는 STPA에서 사용되는 기본적인 접근 방식을 설명하였다. 다음 장에서는 시스템 엔지니어링과 관련된 여러 단계 및 활동에 STPA를 적용하는 방법에 대하여 설명한다. 4장에서는 작업장 안전에 STPA를 사용하기 위한 추가 정보와 예제를 제공한다. 지금까지의 모든 예제는 기술적 시스템(technical systems)과 관련이 있지만 STPA는 어느 사회-기술적 시스템(Sociotechnical system)에서도 사용될 수 있다. 5장에서는 조직 분석과 안전성 이외의 시스템 측면의 발현 속성에 STPA를 사용하는 방법에 대해 설명한다. 6장에서는 STPA를 사용하여 리스크 선행지표를 식별하는 방법을 보여준다. 7장에서는 보다 효과적인 안전 관리 시스템을 설계하기 위한 STAMP 및 STPA의 사용에 대해 설명한다. 마지막으로 8장에서는 대규모 조직 및 프로젝트에 STPA를 통합하는 방법에 대해 지금까지 우리가 학습한 것을 설명한다.

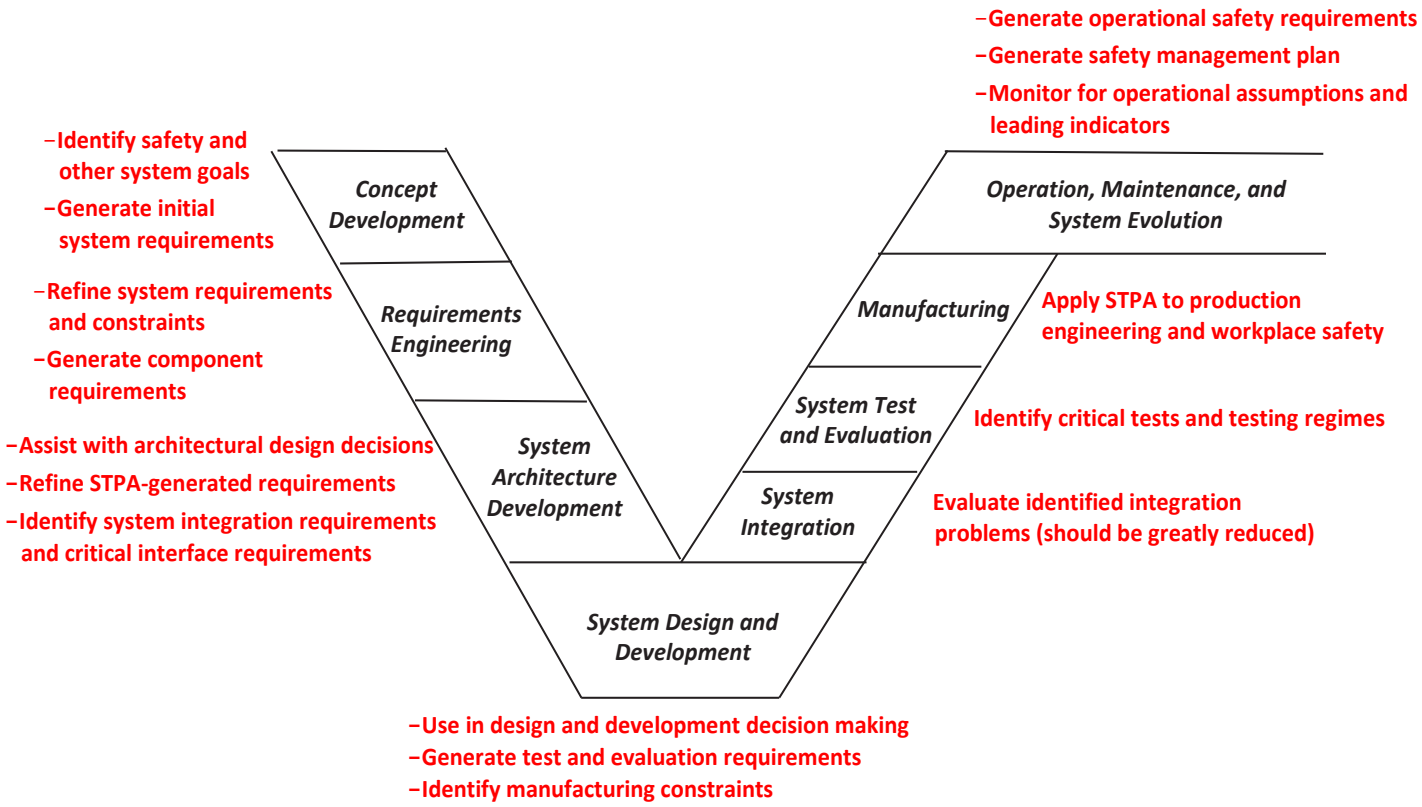
Chapter 3: Integrating STPA into the System Engineering Process

Nancy Leveson

Too often, system safety is isolated or separated in some way from the system engineering process. The most common result is that safety is treated as an after-the-fact assurance activity. Because safety cannot be assured into a system but must be designed in, safety-related design flaws are often found late, when they cannot be fixed. At that point, the effort then focuses on trying to find arguments that the identified flaws do not need to be fixed. When those arguments cannot be sustained, the efforts to deal with the safety flaws often devolve to making expensive and not very effective solutions, such as redundancy or expecting the operators of the system to detect and fix problems through far-from-ideal procedural solutions.

This chapter of the handbook describes how to tightly integrate system safety analysis into the entire system engineering process (see Appendix F for a basic introduction to system engineering). The result will be a significant decrease in the cost of engineering for safety as well as greatly increased effectiveness and, hopefully, fewer losses. It can also reduce rework, which reduces cost and schedule.

The figure below shows a simplified version of the standard system engineering V-model (or waterfall model drawn with a kink in it). The feedback loops are omitted simply to declutter the diagram. This figure is used to illustrate how to integrate STPA into the standard system engineering process. The potential roles for STPA are shown in red. If you use a variant of the V-model or a different development model, it should not be difficult to translate from the standard V-model to most any other model except, perhaps, for those that omit the early processes (the upper two activities on the left leg of the V). In that case, you should not be building safety-critical systems.



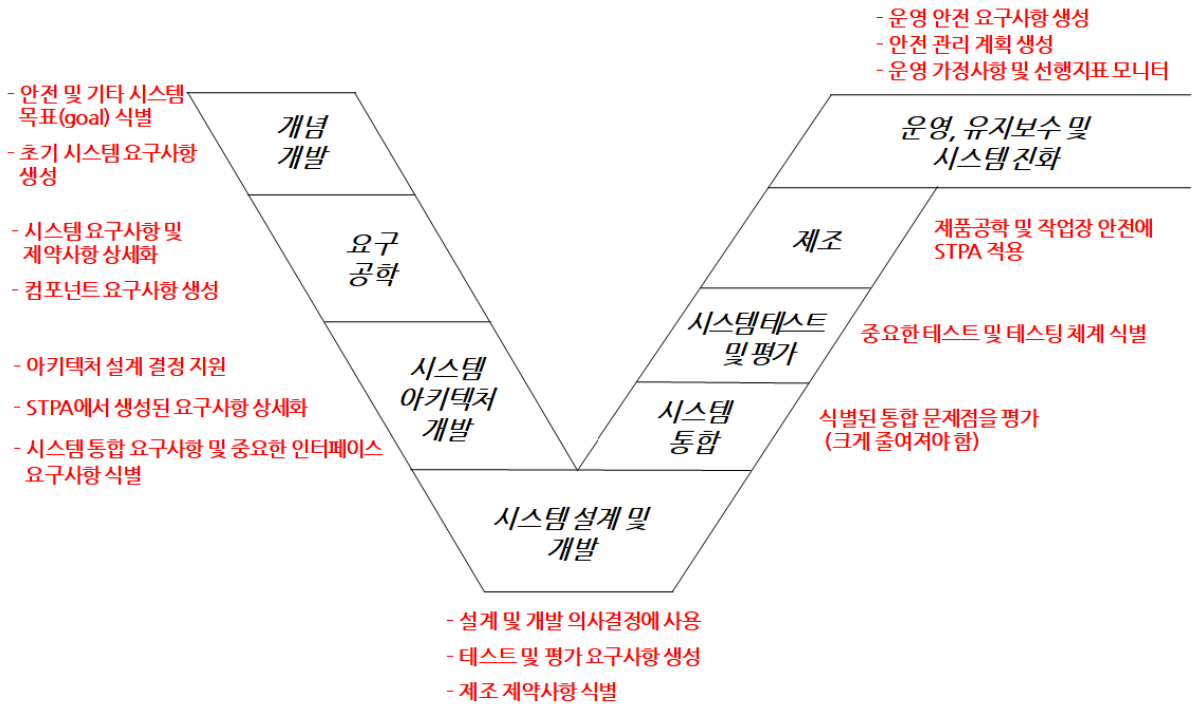
3장: STPA를 시스템 엔지니어링 프로세스에 통합

Nancy Leveson

시스템 안전이 시스템 엔지니어링 프로세스와 독립적으로 또는 분리되어 처리되는 경우는 매우 빈번하다. 가장 흔한 결과로는 안전이 사후보증 활동으로 취급된다는 것이다. 안전은 시스템 내에서 보장될 수 없기 때문에 반드시 설계에 반영되어야 한다. 안전-관련 설계 결함은 종종 고칠 수 없는 늦은 시점에 발견되곤 하는데, 그 시점에서는 식별된 결함이 고쳐질 필요가 없다는 근거를 찾는 데 집중하게 된다. 그러한 근거가 성립되지 않는 경우 중복(redundancy)되거나 시스템 운영자의 이상적인 절차적 방법과는 거리가 먼 방법으로 문제를 탐지하고 고치는 등의 고(高)비용 또는 매우 효과적이지 못한 방법으로 안전 결함을 처리하게 된다.

이 장에서는 시스템 안전 분석과 전체 시스템 엔지니어링 프로세스를 긴밀하게 통합하는 방법에 대해 설명한다(시스템 엔지니어링에 대한 기본적인 소개는 부록 F 참조). 그러한 통합은 효과를 크게 높이고 잘 되는 경우에는 손실을 줄일 수 있을 뿐만 아니라 안전을 위한 엔지니어링 비용 또한 크게 줄일 수 있다. 또한 재작업을 줄여 비용 및 기간도 단축될 것이다.

아래 그림은 표준 시스템 엔지니어링의 V-모델(또는 꺾인 모양의 폭포수 모델)의 간략화된 버전이다. 피드백 루프는 다이어그램을 단순하게 처리하기 위해 생략되었다. 이 그림은 STPA를 표준 시스템 엔지니어링 프로세스에 통합하는 방법을 보여준다. STPA의 역할은 빨간색으로 표시하였다. 변형된 V-모델 또는 다른 개발 모델을 사용하는 경우 초기 프로세스(V-모델 왼쪽 다리의 상위 2개의 활동)를 생략한 모델을 제외하고 표준 V-모델에서 다른 모델로 변환하는 것이 어렵지 않아야 한다. 만약 그 변환이 어렵다면 안전-필수(safety-critical) 시스템을 구축해서는 안 된다.



STPA can be used throughout the standard system engineering process, starting in the earliest concept development stage. In essence, STPA can be used to generate high-level safety requirements early in the concept development phase, refine them in the system requirements development phase. The system requirements and constraints can then assist in the design of the system architecture and more detailed system design and development. The STPA result process then goes hand in hand with design and development as the analysis can be used to inform decisions as they are made. STPA continues to be useful through assurance and manufacturing and provides important information for use during operations

STPA fits into a model-based engineering process as it works on a model of the system (which is refined as design decisions are made) although that model is different than the architectural models usually proposed for model-based system engineering today. STPA promotes traceability throughout the development process so decisions and designs can be changed with minimum requirements for redoing previous analyses.

Finally, as noted in many places in this handbook, STPA can be applied to any emergent system property in the system engineering and product lifecycle, not just safety.

Overall Process

STAMP and STPA contribute to all the activities in system engineering. The use of the analysis and its results for the following activities are described in the rest of this chapter:

1. Definition of losses to be handled during development and operation
2. Identification of external constraints (including market and regulatory requirements) on the system design
3. Identification of system-level hazards and the related requirements and constraints on system behavior
4. Modeling of the system control structure
5. Refinement of hazards and constraints and allocation of functions to system components
6. Assist with making architectural, design, and implementation decisions
7. System Integration assistance
8. Generation of system test requirements
9. Control of manufacturing (production engineering, workplace safety)
10. Generation of operational safety requirements (including leading indicators of increasing risk) and the safety management plan
11. Operational safety management including monitoring leading indicators

1. Decisions about Losses to be Considered

While early Concept Development may differ in specific variants of the V-model, this stage usually includes such activities as stakeholder and user analysis (needs analysis), customer requirements generation, regulatory requirements review, feasibility studies, concept and trade space exploration, and establishment of criteria for evaluation of the evolving and final design. Toward the end of this step, there may be creation of a Concept of Operations.

Too often, this early stage of system engineering is not given the attention and effort it deserves and development proceeds almost immediately with system architecture specification and high-level design. Inadequate concept development may, however, lead to systems that are not usable by the customer, only partially satisfy stakeholder needs, or are difficult to assure, maintain, and operate. While changes

STPA는 초기 개념 개발 단계부터 표준 시스템 엔지니어링 프로세스 전반에 걸쳐 사용될 수 있다. 본질적으로 STPA는 개념 개발 단계 초기에서 높은 수준의 안전 요구사항을 생성하고 시스템 요구사항 개발 단계에서 이를 상세화하는 데 사용될 수 있으며 이 시스템 요구사항과 제약사항은 시스템 아키텍처 설계 및 보다 상세한 시스템 설계·개발을 지원하기 위해 사용될 수 있다. STPA 분석 결과는 의사 결정을 내리는 데 사용될 수 있으므로 STPA 프로세스(result process)는 설계 및 개발 과정과 밀접하게 연관되어 진행된다. 또한, STPA는 일련의 보증활동 및 생산 과정에서 지속적으로 유용하게 사용되며 시스템 운영 중 사용할 중요한 정보를 제공한다.

STPA는 오늘날 모델 기반 시스템 엔지니어링에 대해 일반적으로 제안된 아키텍처 모델과는 다르지만, 시스템 모델(설계 결정이 진행됨에 따라 상세화되는 모델)에서 동작하므로 모델 기반 엔지니어링 프로세스에는 적합하다. STPA는 개발 프로세스 전반에 걸쳐 추적성(traceability)을 지향하므로 기존의 분석을 다시 수행하는 경우 최소 요구사항만으로 의사 결정이나 설계를 변경할 수 있다.

마지막으로, 이 핸드북의 여러 곳에서 언급했듯이 STPA는 안전뿐만 아니라 시스템 엔지니어링 및 제품 수명주기의 모든 발현 속성에 적용할 수 있다.

전체 프로세스

STAMP 및 STPA는 시스템 엔지니어링의 모든 활동에 기여한다. 이 장의 나머지 부분에서는 다음과 같은 활동에 대한 분석과 그 결과의 사용에 대해 설명한다.

1. 개발 및 운영과정에서 다루게 될 손실의 정의
2. 시스템 설계 시 외부 제약사항의 식별(시장 및 규제 요구사항 포함)
3. 시스템 동작에 대한 시스템 수준의 위험, 관련 요구사항 및 제약사항의 식별
4. 시스템 컨트롤 스트럭처 모델링
5. 위험과 제약사항의 상세화 및 시스템 컴포넌트에 기능 할당
6. 아키텍처, 설계 및 구현 시 의사 결정 지원
7. 시스템 통합 지원
8. 시스템 테스트 요구사항 생성
9. 제조(제조공학, 작업장 안전) 컨트롤
10. 운영 안전 요구사항(증가하는 위험에 대한 선행지표 포함) 및 안전 관리 계획의 생성
11. 선행지표 모니터링을 포함하는 운영 안전 관리

1. 고려해야 할 손실에 대한 결정

초기의 '개념(concept) 개발'은 V-모델의 변형 형태에 따라 다를 수 있지만 이 단계에서는 보통 이해관계자 및 사용자 분석(요구 분석), 고객 요구사항 생성, 규제 요구사항 검토, 타당성 조사, 개념 및 절충 여지(trade space) 탐색, 진화(evolving) 및 최종 설계에서의 평가 기준 수립과 같은 활동이 포함된다. 이 단계가 끝날 때까지 운영 개념을 수립할 수 있다.

이러한 시스템 엔지니어링의 초기 단계에 마땅히 해야 할 주의나 노력을 기울이지 않고 시스템 아키텍처 명세나 상위 수준 설계 단계로 즉시 들어가는 일이 매우 자주 발생한다. 그러나 개념 개발이 부적절하면 고객이 사용할 수 없거나 이해관계자의 요구를 부분적으로만 충족시킬 수 있을 수 있고 보증, 유지, 운영이 어려운 시스템이 될 수 있다. 초기 개념 개발 단계의 누락 사항을 보완하기 위해 개발 중에 변경이 이루어질 수도 있지만 이러한 늦은 변경은 개발이 진행됨에 따라 점점 더 많은 비용과 혼란을 야기할 것이다.

may be made later in development to make up for omissions in the early concept development stage, these later changes are increasingly expensive and disruptive as development proceeds.

Safety and security concerns, in particular, are often inadequately considered in early concept exploration. Figure 3.1 shows a typical approach to handling these emergent system properties [Young 2017]. Major emphasis is often placed on responding to a loss only after it occurs in operations. In addition, the major focus may be on adding "bolt-ons" (e.g., protection systems or intruder detection) after the bulk of design and system engineering has already been done. Making changes like these late in the development process is not only more expensive, the fixes are usually much less effective than if safety and security had been built into the system from the beginning.

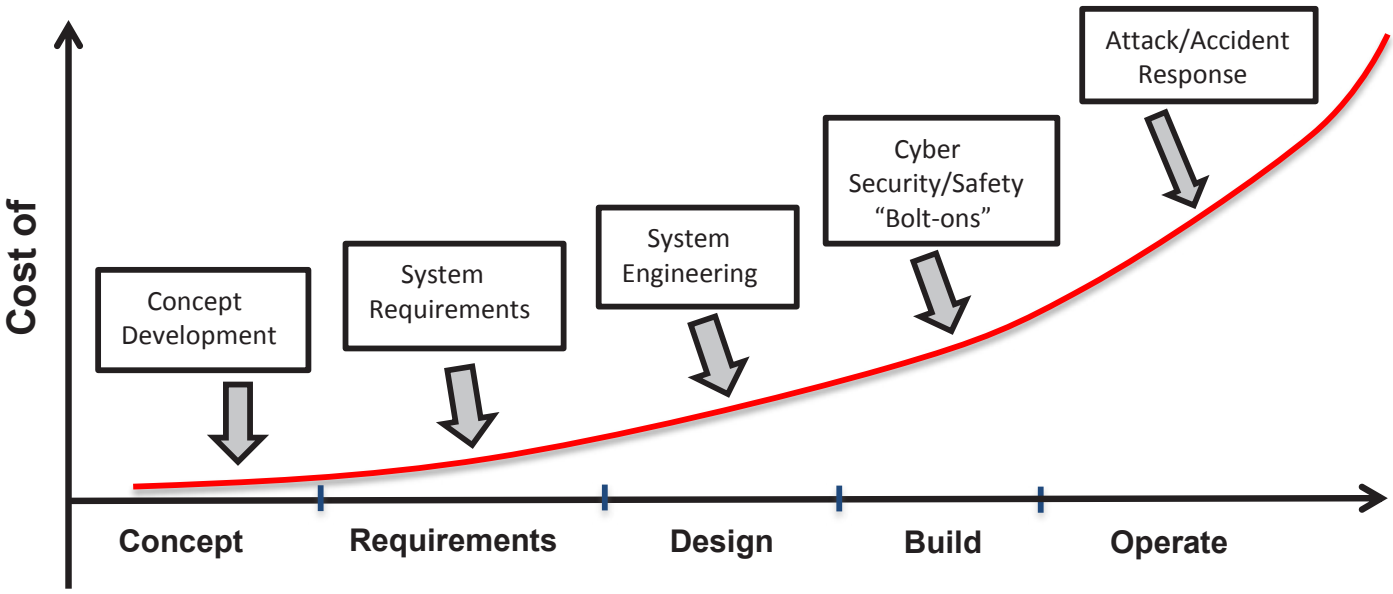


Figure 3.1: Safety and security need to be built into the system from the beginning of development[11]

Instead, safety and security considerations should be initiated in the concept development stage and the results used to generate the safety and security requirements for the overall system and its components. Frola and Miller (1984)[12] claimed that 70-90% of the design decisions related to safety are made in the concept development stage and changing these decisions later may be infeasible or enormously expensive. The same is true for security.

Safety may, in many system engineering processes, be considered at this stage, but usually in the form of a Preliminary Hazard Analysis, which attempts to provide a risk assessment to determine how much effort should be put into the identified hazards during development and operations and some preliminary recommendations on how to eliminate or control them.

[11] William Young, *A System-Theoretic Security Analysis Methodology for Assuring Complex Operations Against Cyber Disruptions*, Ph.D. dissertation, MIT, 2017.

[12] F. Ronald Frola and C.O. Miller, System Safety in Aircraft Acquisition, Technical Report, Logistics Management Institute, Washington D.C., January 1984.

안전과 보안의 문제는 특히, 초기 개념 검토 시 종종 부적절하게 처리되는데 그림 3.1은 이러한 시스템 발현(emergent) 속성을 다루기 위한 전형적인 접근법을 보여준다 [Young 2017]. 종종 운영 중 손실이 발생한 후에야 그 손실에 대응하는 데 중점을 둔다. 또한, 대부분의 설계 및 시스템 엔지니어링이 이미 완료된 후 "볼트 온(bolt-ons)"(예 : 보호 시스템 또는 침입자 탐지)을 추가하는 데 중점을 둘 수 있다. 개발 프로세스의 후반부에서의 이러한 변경은 비용이 많이 소요될 뿐만 아니라, 안전 및 보안이 처음부터 시스템에 구축된 경우보다 훨씬 효과적이지 않다.

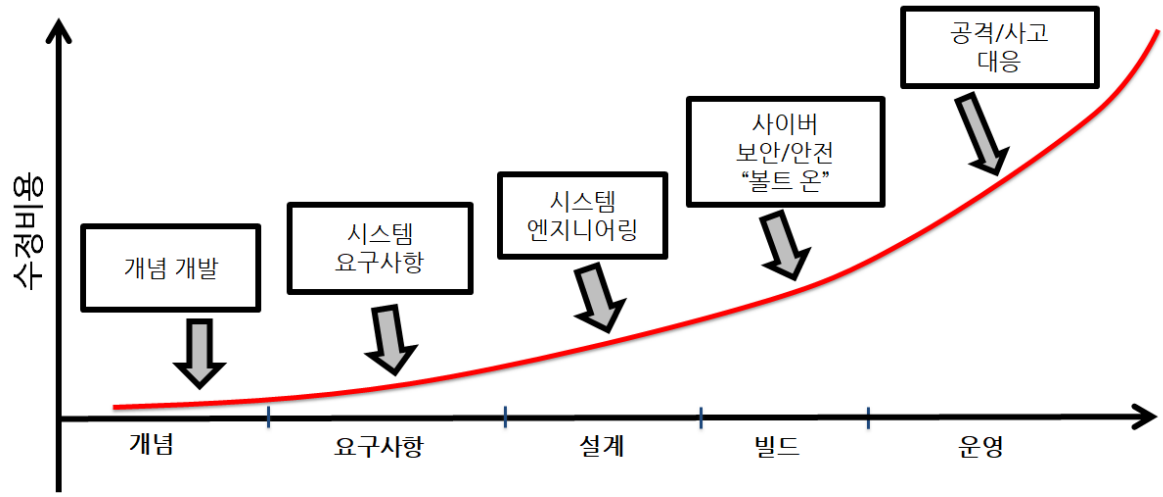


그림 3.1 : 개발 초기부터 안전과 보안을 시스템에 포함하여 구축(build)해야 함[11]

안전과 보안에 대한 고려는 개념 개발 단계에서부터 시작해야 하며 그 검토 결과는 전체 시스템과 컴포넌트에 대한 안전 및 보안 요구사항을 만드는 데 사용되어야 한다. 프롤라(Frola)와 밀러(Miller)(1984)[12]는 안전과 관련된 설계 결정의 70-90%가 개념 개발 단계에서 이루어지며, 나중에 이러한 결정을 변경하는 것은 불가능하거나 막대한 비용이 발생할 수 있다고 주장했다. 이는 보안에서도 마찬가지이다.

많은 시스템 엔지니어링 프로세스에서 안전은 초기 단계에서 고려되지만 일반적으로 예비 위험 분석(PHA, Preliminary Hazard Analysis)의 형태를 취한다. 예비 위험 분석은 식별된 위험에 대해 개발 및 운영단계에서 얼마만큼의 노력을 들여야 할지를 결정하기 위한 리스크 평가(risk assessment)와 그것을 어떻게 제거하거나 제어할지에 대한 예비 권고사항(preliminary recommendations)을 제공한다.

11 William Young, *A System-Theoretic Security Analysis Methodology for Assuring Complex Operations Against Cyber Disruptions*, Ph.D. dissertation, MIT, 2017.

12 F. Ronald Frola and C.O. Miller, System Safety in Aircraft Acquisition, Technical Report, Logistics Management Institute, Washington D.C., January 1984.

Table 2.1. A typical PHA format[13]

PROGRAM: _____ DATE: _____ ENGINEER: _____ PAGE: _____						
ITEM	HAZARD COND	CAUSE	EFFECTS	RAC	ASSESS-MENTS	RECOMM-ENDATIONS
Assigned number	List the nature of the condition	Describe what is causing the stated condition to exist	If allowed to go uncorrected, what will be the effect or effects of the hazardous condition	Hazard Level assign-ment	Probability, possibility of occurrence: -Likelihood -Exposure -Magnitude	Recommended actions to eliminate or control the hazard

[Vincoli, 2005]

While most of the information in the table is available in the early concept development stage, the probability or likelihood of the hazardous condition cannot be known before any detailed design has been done or even after that time if the system is software-intensive. Historical information is not useful when the system differs significantly from past systems. We have found using STPA that specific hazards on real projects that have perfectly reasonable causal scenarios are often incorrectly dismissed early in the system development process as "marginal" or extremely unlikely. Research by psychologists has shown that humans are very bad at estimating probability or likelihood of unusual events.[14]

The biggest problem is that this type of PHA does not provide the information necessary to identify the functional safety and security requirements for the system, which is the primary goal in early concept analysis. Traditional hazard analysis techniques, which theoretically might be used for this purpose, require a fairly detailed design and thus are not appropriate until much later in the development process, when it is too late to provide the functional safety requirements for the design effort.

An alternative often used in the aviation community is to generate probabilistic requirements to be satisfied by the designed system [SAE ARP 4761]. Examples include "Loss of all wheel braking during landing or rejected take off shall be less than 5E-7 per flight" and "Undetected inadvertent wheel braking on one wheel without locking during takeoff shall be less than 5E-9 per flight" [SAE ARP 4761]. While these types of requirements might have been reasonable in the era when braking systems were composed almost exclusively of hardware parts with known failure probabilities, the extensive use of software in almost all systems make it impossible to ensure that these requirements are satisfied in the software-intensive systems being engineered today.

STAMP and STPA provide the ability to generate the information needed in the concept development stage—identifying stakeholder and user needs (needs analysis), generating customer goals and requirements, reviewing regulatory requirements, evaluating feasibility, exploring system concepts and the conceptual trade space, and establishing criteria for evaluation of the evolving and final design. The

[13] Jeffrey Vincoli, Basic Guide to System Safety, John Wiley & Sons, 2005.

[14] See, for example, the work of Tversky and Kahneman on heuristic biases.

표 3.1. 일반적인 PHA 형식[13]

프로그램: _____ 엔지니어: _____					날짜: _____ 페이지: _____	
아이템	위험 조건	원인	영향	RAC	평가	권고사항
할당된 번호	조건의 특성(nature)을 열거	무엇이 명시된 조건을 발생 시키는지 기술	수정되지 않은 상태로 둘 경우, 발생하는 영향 또는 위험한 상태에 대한 영향은 무엇인가	위험 등급 지정	발생 확률, 발생 가능성: - 발생가능성 - 노출 - 규모	위험을 제거하거나 통제하기 위한 권장사항

[Vincoli, 2005]

표에 있는 정보 대부분은 초기 개념 개발 단계에서 알 수 있지만 위험한 조건의 발생확률 또는 발생 가능성은 세부 설계가 완료되기 전에는 알 수 없고, 특히, 소프트웨어 집약적인 시스템의 경우에는 이후 단계까지도 알 수 없다. 또한, 시스템이 이전 시스템과 크게 다른 경우에는 과거의 정보는 도움이 되지 않는다. 우리는 실제 프로젝트에서 매우 합리적인 원인 시나리오를 가진 특정 위험이 종종 시스템 개발 초기 단계에서 미미하거나(marginal) 극히 드물다는 이유로 무시되는 잘못된 상황을 STPA를 적용하며 발견하였다. 심리학자들의 연구에 따르면, 사람은 일반적이지 않은 이벤트에 대해서는 발생 확률이나 발생 가능성을 잘 예측하지 못한다고 한다.[14]

가장 큰 문제는 이러한 유형의 PHA가 초기 개념 분석의 주요 목표에 해당하는 시스템 기능 안전 및 보안 요구사항을 식별하는 데 필요한 정보를 제공하지 않는다는 것이다. 이론적으로 이러한 목적으로 사용될 수 있는 전통적인 위험 분석 기법은 매우 상세한 설계를 필요로 하므로 개발 단계의 훨씬 나중 시점까지도 사용이 적절하지 않으며 개발 단계 후반부는 기능 안전 요구사항을 설계에 반영하기에는 너무 늦은 시점이 된다.

항공 분야에서 흔히 사용되는 대안은 설계된 시스템이 만족할 수 있는 확률적 요구사항을 만드는 것이다 [SAE ARP 4761]. 예를 들어 "착륙 또는 이륙중지(rejected take off) 시 모든 휠 브레이킹의 손실 확률은 비행 당 5E-7 미만이어야 한다" 또는 "이륙 단계에서 락킹(locking)이 되지 않은 하나의 휠에, 감지되지 않고 의도치 않은 휠 브레이킹이 걸릴 확률은 비행 당 5E-9 미만이어야 한다" [SAE ARP 4761] 등이 있다. 브레이킹 시스템의 대부분이 고장 확률이 알려진 하드웨어 부품으로 구성된 시기에는 이러한 유형의 요구사항들이 합리적이었을지도 모르지만 거의 모든 시스템에 소프트웨어를 광범위하게 사용하고 있는 오늘날의 소프트웨어 집약적 시스템에서는 이러한 요구사항들이 충족되지 않을 수도 있다.

STAMP 및 STPA를 사용하여 개념 개발 단계에서 필요한 정보를 도출할 수 있다 – 이해관계자 및 사용자 요구(요구 분석) 식별, 고객 목표 및 요구사항 생성, 규제 요구사항 검토, 타당성 평가, 시스템 개념 및 개념적인 절충 여지(trade space) 탐색, 개선 및 최종 설계에 대한 평가 기준 수립 등. 실패나 문제가 있는 여러 시스템 엔지니어링에서의 주요한 문제점은 이해관계자의 니즈(needs)와 적절한 시스템 목표에 대한 이해의 한계로부터 발생한다.

---

13 Jeffrey Vincoli, Basic Guide to System Safety, John Wiley & Sons, 2005.
14 예를 들어, 휴리스틱 편견(heuristic biases)에 대한 Tversky와 Kahneman의 연구 참조

primary problems involved in many failed or problematic system engineering efforts stem from limited understanding of the stakeholder needs and the appropriate system goals.

2. Identification of external constraints (including market and regulatory requirements) on the system design

The social system safety control structure can be used to understand the external (environmental) social constraints on the system. Figure 3.2 shows an example sociotechnical control structure that is similar to that common to many regulated industries. There is both a control structure for development (on the left) and for operations on the right. Each industry will probably have its own sociotechnical control structure, and it may differ by country. Only an example national structure is shown here but in some industries (e.g., aviation and nuclear) there are overlying international control components.

Once you have modeled this structure, there will probably be few changes for each development project undertaken but some differences may exist. The structure can be used to identify and document regulatory and other external requirements for the system being developed.

3. Identification of Losses to be Considered

At the very early conceptual development stage, stakeholders need to specify unacceptable losses. While many hazard analysis and system safety techniques handle only human death or injury and property damage, STPA can handle any loss including environmental pollution, mission loss, monetary losses, and even damage to company reputation. In other words, the losses represent any emergent system property that the stakeholders want to avoid. Formally, an accident (or mishap, which is the term used by the military) is any undesired or unplanned event or condition that results in what stakeholders consider to be a loss. The aircraft example that has been used in this handbook starts from the following accidents (losses):

A1. Death or serious injury to aircraft passengers or people in the area of the aircraft

A2. Unacceptable damage to the aircraft or objects outside the aircraft

What is considered to be “unacceptable” must be defined by the stakeholders. Additional losses might be specified if they are considered to be worth putting in effort to avoid them such as:

A3: Financial losses resulting from delayed operations

A4: Reduced sales due to damage to aircraft or airline reputation.

2. 시스템 설계 시 외부 제약사항의 식별(시장 및 규제 요구사항 포함)

사회적인 시스템(social system) 안전 컨트롤 스트럭처를 사용하여 시스템 외부의(환경적) 사회적 제약사항을 이해해 볼 수 있다. 그림 3.2는 사회의 기술적 컨트롤 스트럭처 예를 보여주는데 이는 많은 규제 산업에서 공통적으로 유사하다. 이 컨트롤 스트럭처에는 개발을 위한 영역(왼쪽)과 운영을 위한 영역(오른쪽)이 있다. 각 산업은 아마 고유의 사회 공학적 컨트롤 스트럭처를 가질 것이며 이는 국가마다 다를 수도 있다. 여기에는 한 국가의 스트럭처가 예시로 소개되었지만 일부 산업(예: 항공 및 원자력)에서는 국제적인 컨트롤 컴포넌트가 존재하는 경우도 있다.

이러한 스트럭처를 한 번 모델링한 후에는 각 개발 프로젝트에서 변경할 사항이 거의 없겠지만 일부 차이가 있을 수는 있다. 이 스트럭처는 개발 중인 시스템에 대한 규제 및 기타 외부 요구사항을 식별하고 문서화하는 데 사용할 수 있다.

3. 고려되어야 할 손실의 식별

아주 초기 개념 개발 단계에서 이해관계자는 수용할 수 없는 손실을 명시해야 한다. 많은 위험 분석 및 시스템 안전 기법은 사람의 사망이나 부상, 재산의 피해만을 다루지만 STPA는 환경 오염, 임무 실패, 금전적 손실, 회사 평판의 손상까지 포함하여 모든 손실을 다룰 수 있다. 즉, 손실은 이해관계자가 피하고자 하는 모든 시스템 발현 속성(emergent system property)을 의미한다. 공식적으로 사고(또는 국방 용어로 미스햅)는 원하지 않거나 계획되지 않은, 이해관계자가 손실이라고 생각하는 사건 또는 조건을 의미한다. 이 핸드북에서의 항공기 예제는 다음과 같은 사고(손실)에서 출발한다.

A1: 항공기 승객이나 항공기 주변 사람들의 사망 또는 심각한 부상
A2: 항공기나 항공기 외부 물체의 수용할 수 없는 손상

이해관계자는 “수용할 수 없는(unacceptable)” 것으로 간주되는 것을 정의해야 한다. 다음과 같이 피하기 위해 노력할 필요가 있는 경우 추가적인 손실이 명시될 수 있다:

A3: 지연 운항으로 인한 재정적 손실
A4: 항공기 손상이나 항공사 평판 훼손으로 인한 매출 감소

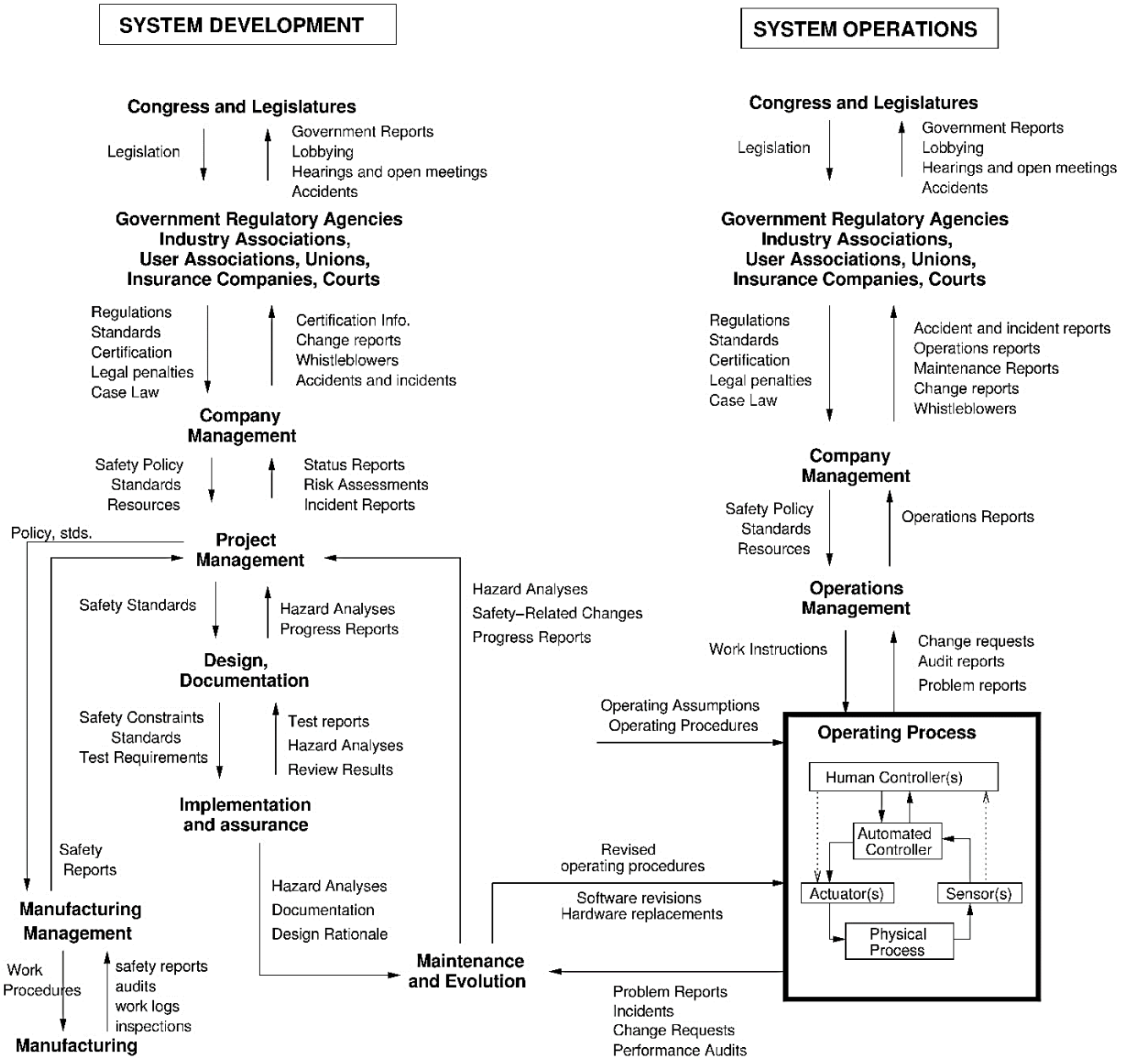


Figure 3.2: An Example Safety Control Structure

4. Identification of system-level hazards and the related constraints on system behavior

Once unacceptable losses are identified by the stakeholders, engineers can define the hazards that are associated with those losses. The ultimate goal is to eliminate or, if not possible, to control or reduce identified hazards during the system design activity.

The first problem is how to define the hazards. Although the term hazard is sometimes used to refer to things outside the system boundaries, such as inclement weather in aviation, hazards in STPA are limited to system states that are within the control of the system designer. For example, the hazard is not the inclement weather but rather it is the aircraft being negatively impacted by inclement weather. Constraints or controls may include staying clear of the weather or designing the aircraft to withstand

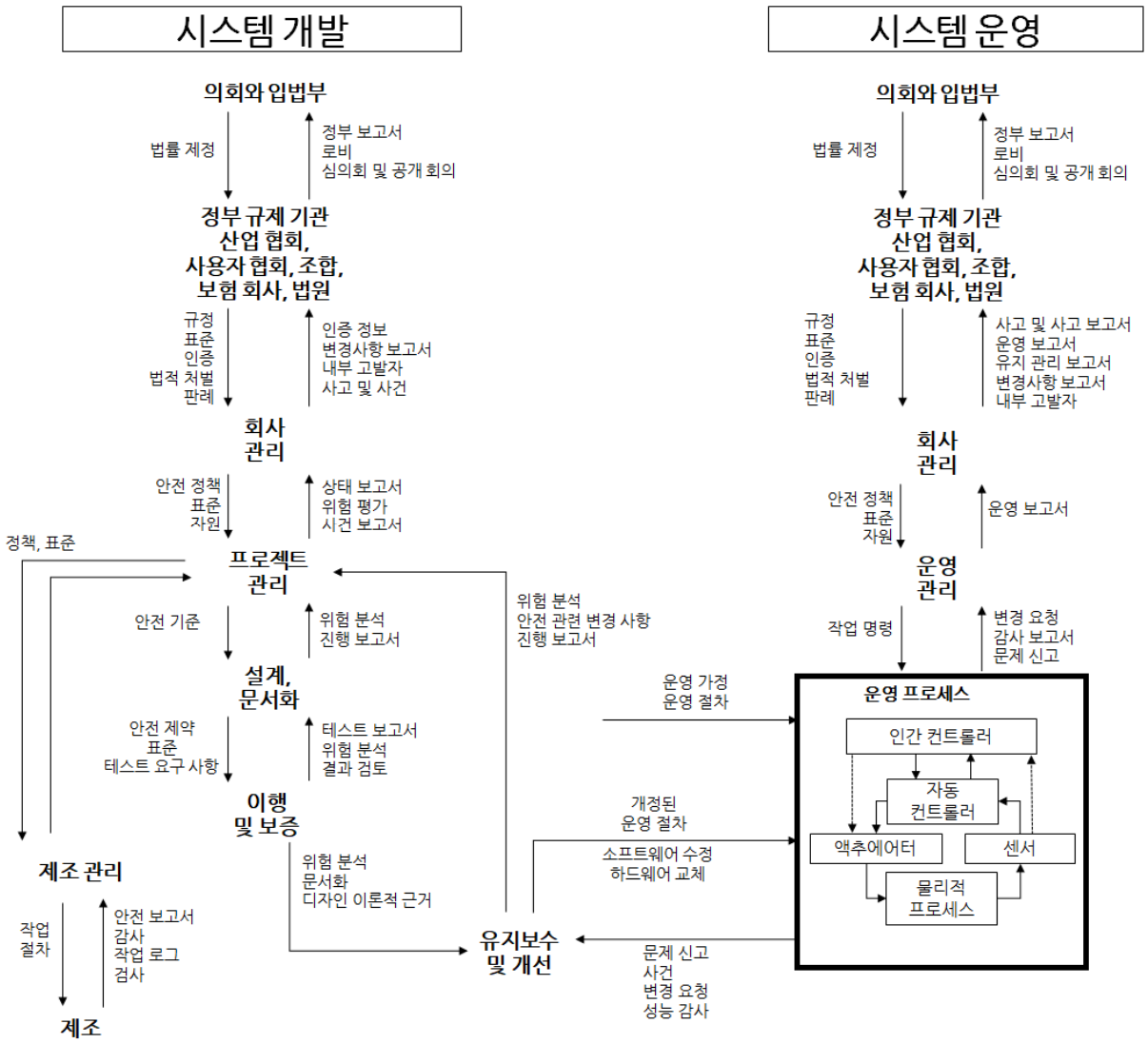


그림 3.2 : 안전 컨트롤 스트럭처의 예

4. 시스템 동작에 대한 시스템 수준 위험 및 관련 제약사항 식별

이해관계자가 수용할 수 없는 손실을 식별하면 엔지니어는 해당 손실과 관련된 위험을 정의할 수 있다. 궁극적인 목표는 시스템 설계 활동을 통해 식별된 위험을 제거하고 제거가 불가능한 경우에는 이를 통제하거나 줄이는 것이다.

우선 첫 번째 문제는 위험을 어떻게 정의하는 가이다. 위험이라는 용어는 때로는 항공 도메인에서의 궂은 날씨와 같이 시스템 경계 외부의 것을 가리키는 데 사용되기도 하지만 STPA의 위험은 시스템 설계자가 제어할 수 있는 시스템의 상태로 한정한다. 예를 들어, 위험은 궂은 날씨가 아니라 궂은 날씨로 인해 부정적인 영향을 받는 항공기이다. 제약사항이나 컨트롤에는 날씨의 영향을 받지 않거나 날씨의 영향을 견딜 수 있도록 항공기를 설계하는 것이 포함될 수 있다. 시스템 엔지니어링은 시스템 설계와 관련되기 때문에 시스템 자체의 동작에만 영향을 미칠 수 있으며 시스템 경계 외부(즉, 시스템 운영 환경)는 변경할 수 없다.

the impact of the weather. Because system engineering is concerned with the design of the system, it can have an impact only on the behavior of the system itself and cannot change things outside the system boundary (i.e., in the system operating environment).

Sometimes a hazard is defined as a condition that could lead to an accident or a precondition for an accident. But that definition includes almost everything, most of which cannot be eliminated or controlled if the system is to operate at all. For example, an aircraft that is landing could lead to an accident later on, but we cannot build a useful system where aircraft never land. As another example, the hazard is not deceleration on landing (which is required for safe operations) but inability to decelerate after landing.

So for practical reasons, hazards are defined as precursor states to an accident that the system designers never want to occur and thus should be eliminated or controlled in the design process. A formal definition was provided in the previous chapter.

Clearly, in order to eliminate the hazards in the system design, they need to be identified at the beginning of the system development process. The specified hazards are used to create the behavioral (functional but not probabilistic) safety requirements for the various system components, including the software and human operators.

For the accidents A1 and A2 above, the hazards include**:**

H1: Insufficient thrust to maintain controlled flight [A1, A2]

H2: Loss of airframe integrity [A1, A2]

H3: Violating minimum separation between aircraft and fixed or moving objects [A1, A2]

H4: An aircraft on the ground comes too close to moving or stationary objects or inadvertently leaves the taxiway [A1, A2]

H5: Aircraft unable to take off when scheduled [A3, A4]

H6: etc.

These hazards lead to very high-level system constraints[15] such as:

SC1: Sufficient thrust must be available to maintain controlled flight [H1]

SC2: Airframe integrity must be maintained under worst case conditions [H2]

SC3: Aircraft must satisfy minimum separation standards from fixed or moving objects [H3]

SC4: Aircraft on the ground must always maintain a safe distance from moving or stationary and objects and remain within safe regions such as taxiways.

SC5: Aircraft must be able to take off within TBD minutes of scheduled departure [H6]

SC6: etc.

5. Modeling the Functional System Control Structure

While the first control structure model developed can include the environment in which the system will operate (outside the system boundary), the system control structure in this step provides more detail about what is within the system design space. The scope of the model (which is the definition of

[15] Because negative requirements are often not allowed and changing safety constraints into positive statements is sometimes not possible without losing important information, requirements and constraints are distinguished here. Constraints are described using “must” or “must not” rather than “shall.” There is an additional advantage in that a distinction is made between requirements (system goals or mission) and constraints on how those goals can be achieved. The differentiation is useful when tradeoff decisions need to be made.

때때로 위험은 사고를 야기할 수 있는 조건 또는 그러한 사고의 전제조건으로 정의되기도 한다. 그러나, 이러한 정의는 거의 모든 것을 포함하기 때문에 만일 시스템이 전혀 동작하지 않으면 그러한 위험의 대부분은 제거되거나 통제될 수 없는 것이 되어 버린다. 예를 들어 착륙을 시도하는 항공기는 사고가 발생할 수 있지만 절대 착륙하지 않는 항공기 시스템을 구축할 수는 없다. 또 다른 예로, 착륙 시의 위험은 감속(안전한 운영을 위해 필요한 것)이 아니라 착륙 후 감속이 불가능한 경우이다.

따라서 현실적인 이유로 위험은 시스템 설계자가 결코 원하지 않는 사고의 전조(前兆, precursor) 상태로 정의하며 위험은 설계 프로세스에서 제거되거나 통제되어야 하는 것이다. 공식적인 정의는 이전 장에서 언급하였다.

시스템 설계에서 위험을 제거하기 위해서는 시스템 개발 프로세스의 초기 단계에서 위험을 분명히 식별해야 한다. 식별된 위험을 활용하여 소프트웨어와 운영자를 포함한 다양한 시스템 컴포넌트에 대한 동작(확률적이 아닌 기능적) 안전 요구사항을 작성한다.

위의 A1 및 A2 사고의 위험은 다음과 같다.

H1: 통제된 비행을 유지하기에 추진력이 불충분함 [A1, A2]
H2: 기체 무결성의 손실 [A1, A2]
H3: 항공기와 고정된 물체 또는 움직이는 물체 사이의 최소이격거리를 위반함 [A1, A2]
H4: 지상에 있는 항공기가 움직이거나 고정된 물체에 너무 가까이 접근하거나 유도로를 의도치 않게 이탈함 [A1, A2]
H5: 스케줄된 항공기가 이륙할 수 없음 [A3, A4]
H6: 기타

이러한 위험은 다음과 같은 상위 수준의 시스템 제약사항을 생성한다.[15]

SC1: 통제된 비행을 유지하기 위해 충분한 추진력이 있어야 한다 [H1]
SC2: 최악의 조건에서 기체 무결성을 유지해야 한다 [H2]
SC3: 항공기는 고정된 물체 또는 움직이는 물체로부터 최소 이격 기준을 만족해야 한다 [H3]
SC4: 지상에 있는 항공기는 움직이는 물체나 고정된 물체로부터 항상 안전한 거리를 유지해야 하며 유도로와 같은 안전한 지역 내에 있어야 한다 [H4]
SC5: 항공기는 스케줄 된 출발시간 TBD분 이내에 이륙할 수 있어야 한다 [H5]
SC6: 기타

5. 기능적 시스템 컨트롤 스트럭처의 모델링

앞서 개발된 첫 번째 컨트롤 스트럭처 모델에는 시스템이 작동하는 환경(시스템 경계 외부)을 포함시킬 수 있었지만 이 단계의 시스템 컨트롤 스트럭처는 시스템 설계 영역 안에 있는 것에 대하여 보다 상세한 정보를 제공한다. 모델링 범위(고려할 시스템의 경계로 정의됨)는 시스템 엔지니어링 프로세스의 목표에 따라 달라진다. 개념 설계 프로세스가 진행됨에 따라 세부 사항이 증가하면 다수의 서로 다른 모

15 부정적인 형태로 요구사항을 쓰는 것이 가끔 허용되지 않으며 중요한 정보를 잃지 않으면서 안전 제약사항을 긍정적인 문장으로 변경하는 것이 때로는 가능하지 않기 때문에 여기서는 요구사항과 제약사항을 구분한다. 제약사항은 "~하도록 한다(shall)"보다는 "~해야 한다(must)" 또는 "~해서는 안 된다(must not)"를 사용하여 기술한다. 또 다른 추가적인 장점은 요구사항(시스템 목표 또는 미션)과 제약사항은 그 목표를 달성하는 방법이 다르다는 점이다. 이러한 차별화는 트레이드오프(tradeoff) 결정을 내려야 할 때 유용하다.

the boundaries of the system to be considered) will depend on the goal of the system engineering process. Multiple different models may be useful, with increasing detail being provided as the conceptual design process proceeds. Details on building control structure models are provided in the previous chapter. Some repetition is included here to make the chapters free-standing.

Looking at the control structure in Figure 3.2, the system to be designed might be the functional system control structure within the lower right-hand box of the figure labeled “Operating Process.” STPA is usually first performed on this system control structure to identify the very high-level system requirements and constraints. During the conceptual design process, the control structure will be refined and STPA is used to refine the associated system requirements and constraints.

For example, Figure 3.3 shows a very high-level functional control model of an aircraft, with just three components: the pilot, the automated control system (which will probably consist of multiple computers, but that decision is left for a later design stage), and the physical aircraft components. For complex systems, such as aircraft, levels of abstraction can be used to zoom in on the pieces of the control structure currently being considered. This type of top-down refinement is also helpful in understanding the overall operation of the aircraft and in identifying interactions among the components.

Note that there are no design decisions in this model beyond the fact that the system will be an aircraft and the aircraft will include the general functions and design of any aircraft. Therefore, it is appropriate for use at the early concept development stage. If any of the parts of the model are too detailed for a particular application at this point in system engineering, then they can be removed. Note that the pilot may be a human (onboard or on the ground) or automation (for unmanned aircraft) and the aircraft could even be fixed wing, vertical lift, or VTOL (vertical takeoff and landing). No distinction is necessary at this point of conceptual development. This generality allows decisions to be put off until the more information is available and also allows reuse of the models and the analysis. Aircraft systems being designed today may include several or all of these types of piloting configurations.

The role of the pilot, as shown in the Figure 3.3 control structure, is to manage the aircraft automation and, depending on the design of the aircraft, directly or indirectly control takeoff, flight, landing, and maneuvering the aircraft on the ground. The pilot and the automated controllers contain a model of the system or subsystem that they are controlling. The automation is controlling the aircraft so it must contain a model of the current aircraft state. The pilots also need a model of the aircraft state, but in addition they need a model of the state of the automation and a model of the airport or airspace in which they are operating. Many pilot errors can be traced to flaws in their understanding of how the automation works or in their understanding of the current state of the automation.

델을 개발하는 것이 유용할 수도 있다. 컨트롤 스트럭처 모델 구축에 대한 자세한 내용은 이전 장에서 언급하고 있지만, 장을 독립적으로 만들기 위해 여기에서도 일부 내용을 반복해 언급한다.

그림 3.2의 컨트롤 스트럭처를 보면 설계되어야 할 시스템은 "운영 프로세스"라고 표시된, 그림 우측 하단 상자 내에 있는 기능적 시스템 컨트롤 스트럭처가 될 수 있다. 일반적으로 STPA의 시작은 매우 상위 수준의 시스템 요구사항 및 제약사항을 식별하기 위해 이 컨트롤 스트럭처에서 수행된다. 개념 설계 과정에서 컨트롤 스트럭처가 상세화되고 STPA를 사용하여 연관된 시스템 요구사항과 제약사항을 상세화한다.

예를 들어 그림 3.3은 매우 상위 수준의 항공기 기능적 컨트롤 모델을 보여주는데, 이 모델에서 항공기는 조종사, 자동 컨트롤 시스템(아마 여러 대의 컴퓨터로 이루어질 것이며 이는 이후 설계 단계에서 결정됨), 그리고 물리적 항공기라는 3개의 컴포넌트로 구성되어 있다. 항공기와 같은 복잡한 시스템의 경우 현재 검토 중인 컨트롤 스트럭처의 일부를 확대(zoom in)하기 위해 몇 단계의 추상화(abstraction)를 사용할 수 있다. 이러한 유형의 하향식 상세화는 항공기의 전반적인 운영을 이해하고 컴포넌트 간의 상호작용을 파악하는 데 도움이 된다.

이 모델에서, 시스템이 항공기이며 항공기의 일반적인 기능과 설계가 포함되는 것 외에는 다른 설계 결정이 없다는 점에 유의한다. 따라서 이것은 개념 개발 단계 초기부터 사용하기에 적합하다. 이 시점에서 모델의 어떤 부분이 시스템 엔지니어링의 특정 용도에 비해 너무 상세하다면 이런 상세한 내역은 제거할 수 있다. 예를 들어 조종사는 사람(기내 또는 지상) 또는 자동화(무인 항공기)일 수 있으며 항공기는 고정익(fixed wing), 수직이륙(vertical lift) 또는 수직 이착륙기(VTOL)일 수 있다. 개념 개발의 시점에서는 이런 구별이 필요하지 않다. 이러한 보편성은 더 많은 정보를 얻을 수 있을 때까지 의사 결정을 미룰 수 있고 이는 모델과 분석 결과의 재사용을 가능케 한다. 오늘날 설계되는 항공기 시스템은 이러한 유형의 파일럿 구성(piloting configuration) 중 일부 또는 모두를 포함할 수 있다.

그림 3.3의 컨트롤 스트럭처에 있는 조종사의 역할은 항공기 자동화를 관리하고 항공기의 설계에 따라 지상에서의 항공기의 이륙, 비행, 착륙 및 기동을 직접 또는 간접적으로 컨트롤하는 것이다. 조종사 및 자동화 컨트롤러에는 제어중인 시스템 또는 서브시스템 모델이 포함되어 있다. 자동화 컨트롤러(automation)는 항공기를 제어하므로 현재 항공기 상태 모델을 포함해야 한다. 조종사도 항공기 상태 모델이 필요하며 추가적으로 자동화 컨트롤러의 상태 모델과 운영 중인 공항 또는 영공의 모델도 필요하다. 조종사의 실수 중 많은 원인은 조종사가 자동화 컨트롤러의 작동 방식이나 현재의 자동화 컨트롤러의 상태를 잘못 이해하는 데 있다.

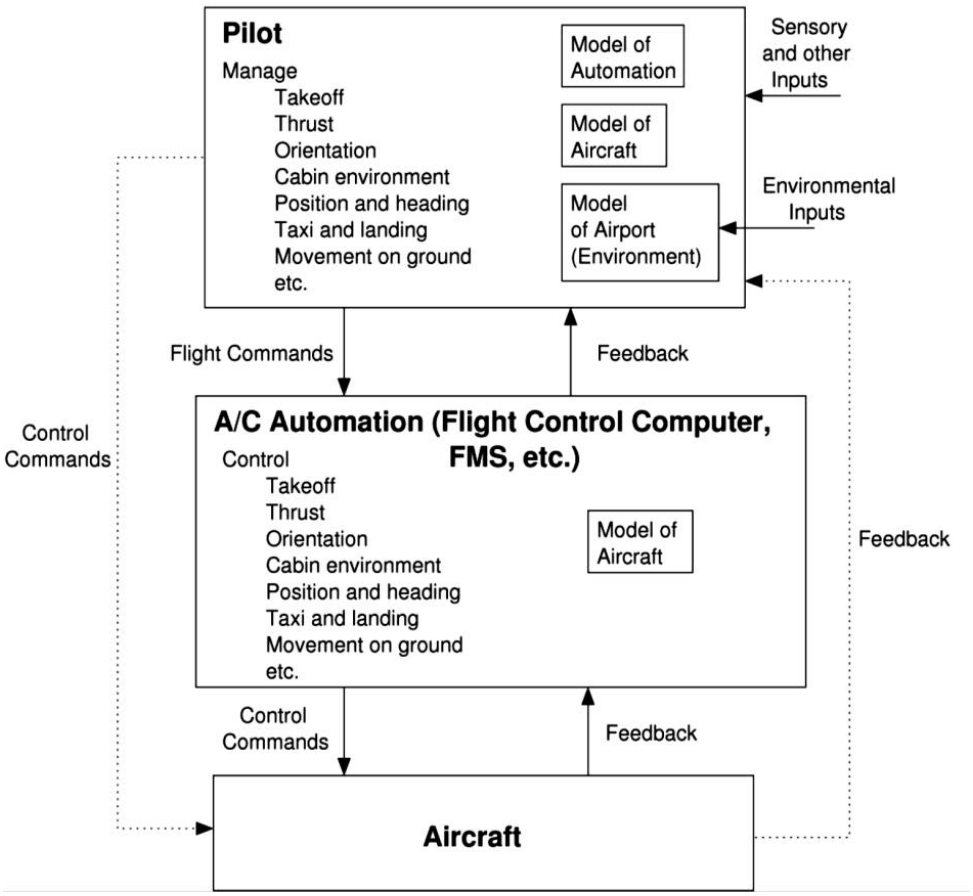


Figure 3.3: A High-Level Control Structure at the Aircraft Level

Pilots provide flight commands to the automation and receive feedback about the state of the automation and the aircraft. In some designs, the pilot can provide direct control actions to the aircraft hardware (i.e., not going through the automated system) and receive direct feedback. The dotted lines represent this direct control and feedback. As the design is refined and more detailed design decisions are made, these dotted line links may be eliminated or instantiated with specific content. The pilot always has *some* direct sensory feedback about the state of the aircraft and the environment (unless the pilot is on the ground, as in UAS) through various sensory modalities. At the same time, pilots often have limited direct sensory feedback about the state of the automation and the physical components the automaton is controlling; the information available to them will be limited to those items the engineers deemed important for them to know. STPA helps to identify what this critical information is. Some advanced flight controls today contain states and inputs that provide no feedback to the pilot at all because the engineers have assumed they do not need it or have omitted it for some other reason.

STPA is done using this control structure to refine the very general system-level safety requirements and constraints (examples shown above) developed from the system hazards. These new refined safety requirements will specify the responsibilities, authority, and accountability of the three components in the high-level control structure as well as the contents of the process models for each component and the feedback requirements in order to maintain safe operation of the aircraft. For example, one requirement may be that the pilot commands adequate thrust to ensure safe maneuvering of the aircraft at each stage of operation [**SC1**] and another is that the pilot shall provide safe commands to control movement on the ground [**SC4**]. Note that the system-level safety requirements are now starting to be allocated to individual components in the safety control structure.

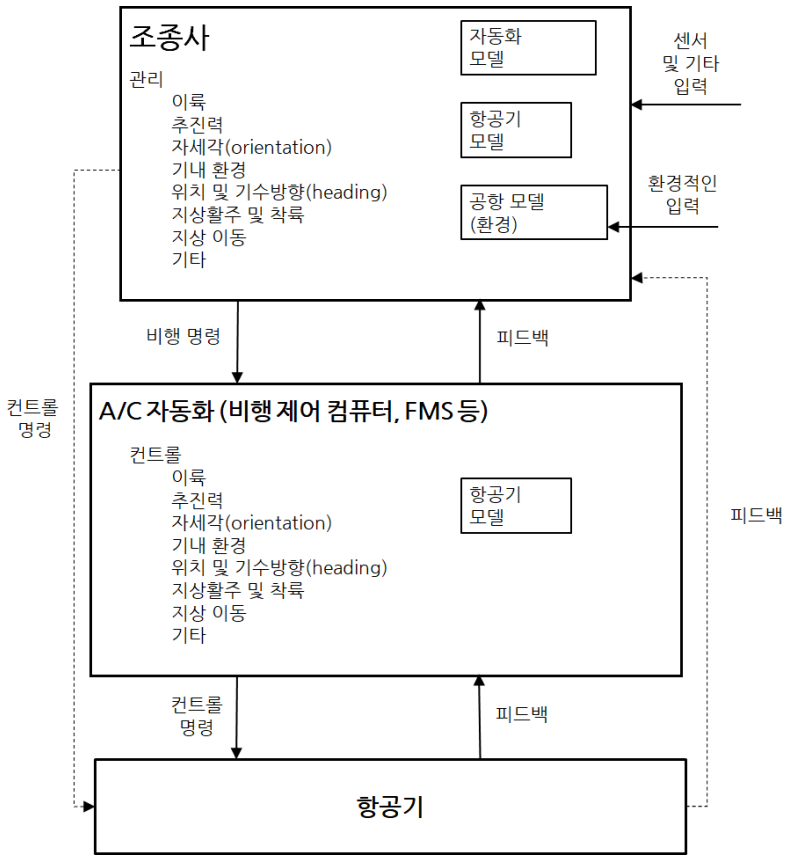


그림 3.3 : 항공기 레벨에서의 상위 수준 컨트롤 스트럭처

조종사는 자동화 시스템에 비행 명령을 지시하고 자동화 및 항공기 상태에 대한 피드백을 받는다. 일부 설계에서는 조종사가 항공기 하드웨어에 직접 컨트롤 액션을 제공할 수 있고(즉, 자동화 시스템을 거치지 않고) 직접 피드백을 수신할 수 있다. 위 그림에서 점선은 직접적인 제어 및 피드백을 나타낸다. 설계가 상세화되어 보다 상세한 설계 결정이 이루어지면 이러한 점선은 제거되거나 특정 내용으로 구체화될 수 있다. 조종사는(무인항공기처럼 조종사가 지상에 있지 않는 한) 다양한 감지(sensory) 수단을 통해 항공기나 환경 상태에 대한 몇 가지 직접적인 센서 피드백을 항상 받는다. 동시에, 조종사는 자동화 및 자동화가 제어하는 물리적 컴포넌트에 대한 직접적인 감각 피드백에 제한이 있는 경우가 있으므로 조종사가 접근 가능한 정보는 엔지니어가 조종사에게 중요하다고 판단하는 정보로 제한된다. STPA는 이러한 중요한 정보가 무엇인지 파악하는 데 도움을 준다. 오늘날 일부 발전된 비행 제어 장치에서는 조종사에게 제공하는 상태나 입력값에 피드백 정보가 전혀 없는데 이는 엔지니어가 이런 정보가 조종사에게 필요하지 않다고 판단하거나 다른 이유로 누락한 것이다.

STPA는 이 컨트롤 스트럭처를 사용하여 시스템 위험에서 도출된 매우 일반적인 시스템 수준의 안전 요구사항과 제약사항(위에서 보여준 예)을 상세화한다. 이렇게 새로 상세화된 안전 요구사항에 따라 상위 수준의 컨트롤 스트럭처의 세 가지 컴포넌트 각각에 대해 항공기의 안전한 운항 유지에 필요한 프로세스 모델 내용 및 피드백 요구사항뿐만 아니라 책임(responsibilities), 통제권한(authority) 및 의무(accountability)를 구체화한다. 예를 들어 조종사는 각 운영 단계에서 항공기의 안전한 기동을 보장하기 위해 적절한 추진력을 명령해야 한다는 요구사항이 있을 수 있으며 [SC1], 항공기의 지상 이동을 제어하기 위해 안전한 명령을 제공한다는 요구사항[SC4]이 있을 수 있다. 이제 시스템 수준의 안전 요구사항이 안전 컨트롤 스트럭처의 개별 컴포넌트에 할당되기 시작함에 주목한다.

6. Refinement of Hazards and Safety Constraints and Allocation to System Components

As the conceptual design process continues, the control structure will be refined as well as the requirements. This refinement process results from the identification of unsafe control actions and their causal scenarios appropriate to the current level of design detail. The result will be a set of safety-related requirements that can be refined and used for the rest of the development process.

The system and safety requirements should precede creation of the system architecture used to satisfy them. Of course, safety is not the only goal for any system and other requirements will need to be considered in the development of the architecture. But, in our experience, architectures are often developed before the safety (and often the system) requirements are identified. This inverted sequence raises the likelihood that the architecture will not be optimized for and sometimes not even appropriate for the system goals.

The architectural development will include the allocation of system requirements to the system components. This process will include, for STPA, generation of causal scenarios for the unsafe control actions generated at the system level. Deceleration is used as an example of this refinement process. The relevant high-level system hazard here is

H4: An aircraft on the ground comes too close to moving or stationary objects or inadvertently leaves the taxiway [A1, A2]

The specific accidents related to H4 occur when the aircraft operates on or near the ground and may involve the aircraft departing the runway or impacting object(s) on or near the runway. Such accidents may include hitting barriers, other aircraft, or other objects that lie on or beyond the end of the runway at a speed that causes unacceptable damage, injury or loss of life.

H4 can be refined into the following deceleration-related hazards:

H4-1: Inadequate aircraft deceleration upon landing, rejected takeoff, or taxiing

H4-2: Deceleration after the V1[16] point during takeoff

H4-3: Aircraft motion when the aircraft is parked

H4-4: Unintentional aircraft directional control (differential braking)

H4-5: Aircraft maneuvers out of safe regions (taxiways, runways, terminal gates, ramps, etc.)

H4-6: Main gear wheel rotation is not stopped when (continues after) the landing gear is retracted

The high-level system safety constraints associated with these hazards are a simple restatement of the hazards as constraints on the design.

SC1: Forward motion must be retarded within TBD seconds of a braking command upon landing, rejected takeoff, or taxiing (H4-1).

SC2: The aircraft must not decelerate after V1 absent direct pilot action (H4-2).

SC3: Uncommanded movement must not occur when the aircraft is parked (H4-3).

SC4: Differential braking must not lead to loss of or unintended aircraft directional control (H4-4)

[16] The V1 point is that point in the takeoff sequence where it is more dangerous to stop the takeoff than to continue. In some very rare circumstances, it is safer for the pilot to decelerate after the V1 point than to continue the takeoff procedure. These situations (and a few others) are ignored in this example analysis to allow the results to fit in a few pages.

6. 위험과 안전 제약사항의 상세화 및 시스템 컴포넌트에의 할당

개념 설계 프로세스가 계속됨에 따라 요구사항뿐만 아니라 컨트롤 스트럭처 또한 상세화 될 것이다. 이런 상세화 프로세스는 언세이프 컨트롤 액션(UCA)과 현재 수준의 설계 세부 사항에 부합하는 원인 시나리오를 식별한 결과이다. 그러한 결과는 안전-관련 요구사항(safety-related requirement)의 집합이며 이는 남아있는 개발 프로세스에서 상세화되고 활용될 수 있다.

시스템 및 안전 요구사항은 이를 충족시키기 위한 시스템 아키텍처를 설계하기 전에 먼저 만들어야 한다. 물론 모든 시스템에서 안전이 유일한 목표는 아니며 아키텍처 개발 시 다른 요구사항도 고려해야 한다. 그러나 우리의 경험에 따르면 종종 안전 요구사항(때로는 시스템 요구사항)이 정해지기 전에 아키텍처가 먼저 개발된다. 이 뒤바뀐 순서로 인해 아키텍처가 최적화되지 않고 때로는 시스템 목표에 적합하지 않을 가능성이 증가하게 된다.

아키텍처 개발 시 시스템 요구사항을 시스템 컴포넌트에 할당하는 작업을 수행하는데 STPA의 경우 이 프로세스에서 시스템 수준에서 생성된 UCA에 대한 원인 시나리오를 생성하게 된다. 이러한 상세화 프로세스의 예로 항공기 감속(deceleration)에서 관련된 상위 수준의 시스템 위험은 다음과 같다.

H4 : 지상에 있는 항공기가 움직이거나 고정된 물체에 너무 가깝게 접근하거나 유도로를 의도치 않게 이탈함 [A1, A2]

H4와 관련된 사고는 항공기가 지상 또는 지상 근처에서 운항할 때 발생하며 항공기가 활주로를 이탈하거나, 활주로나 그 근처에서 물체와 충돌하는 결과를 초래할 수 있다. 그러한 사고는 항공기가 수용할 수 없는 손상(데미지), 부상 및 인명 사망을 유발할 수 있는 속도로 장애물(barrier), 다른 항공기 또는 활주로나 활주로 끝 너머에 있는 다른 물체와 충돌하는 것일 수 있다.

H4는 다음과 같은 감속 관련 위험으로 상세화될 수 있다.

H4-1: 착륙, 이륙중지 또는 지상활주 시의 불충분한 항공기의 감속

H4-2: 이륙 중 V1[16] 포인트 이후의 감속

H4-3: 항공기가 주차되었을 때의 항공기 움직임

H4-4: 의도하지 않은 항공기 방향 제어(차등 브레이킹)

H4-5: 안전한 지역(유도로, 활주로, 터미널 게이트, 주기장 등)을 벗어난 항공기 기동

H4-6: 랜딩 기어를 접어 넣을 때 주 기어 휠 회전이 멈추지 않음

이러한 위험과 관련된 상위 수준의 시스템 안전 제약사항은 위험을 설계의 제약사항으로 간단하게 재작성한 것이다.

SC1: 착륙, 이륙중지 또는 지상활주 시 앞 방향으로의 움직임(forward motion)은 브레이크 명령 후 TBD초 이내로 감속되어야 한다 (H4-1)

SC2: 항공기는 V1 이후에는 조종사의 직접적인 명령 없이 감속해서는 안 된다 (H4-2)

SC3: 항공기가 주차했을 때는 명령하지 않은 이동이 발생하지 않아야 한다 (H4-3)

SC4: 차등 브레이킹이 방향 제어 상실 또는 의도치 않은 항공기 방향 제어를 야기해서는 안 된다

16 V1 포인트는 이륙 시퀀스 중에서 이륙을 계속 진행하는 것보다 멈추는 것이 더 위험한 시점을 말한다. 매우 드문 환경에서 조종사가 V1 포인트 이후에 이륙 절차를 계속 진행하는 것보다 감속하는 것이 안전할 수 있다. 본 사례 분석에서는 예시의 결과를 몇 페이지 분량으로 맞추기 위해 이러한 상황(그리고 다른 몇 가지 상황)을 고려하지 않았다.

SC5: Aircraft must not unintentionally maneuver out of safe regions (taxiways, runways, terminal gates and ramps, etc.) (H4-5)

SC6: Main gear rotation must stop when the gear is retracted (H4-6)

Note that all of these are very high-level, i.e., they start with consideration of the aircraft as a whole. As explained in the previous chapter, they will be refined through an iterative process into a more detailed set of functional safety requirements that are associated with specific system components, including the crew, the software, and the component interactions. System safety requirements are generated to prevent the causal scenarios identified by STPA. The process continues until the hazards have all been adequately analyzed and handled in the design. The important point is that the system behavior as a whole is considered first so that potential unsafe interactions among components can be identified at the beginning of the process without having to try to consider all interactions later and determine whether they are hazardous or not.

Figure 3.4 zooms in on the control structure model for the ground control function, which is where deceleration occurs. There are three basic physical components being controlled: the reverse thrusters, the spoilers, and the wheel braking system. Again, by including the larger functional control structure rather than simply one of these, STPA can consider all interactions (both intended and unintended) among the braking components related to the hazard being analyzed. There are clearly some design decisions shown here in that the braking system is determined to contain reverse thrusters, spoilers, and wheel brakes. This level of design detail, however, is still very general and high-level and satisfies most aircraft.

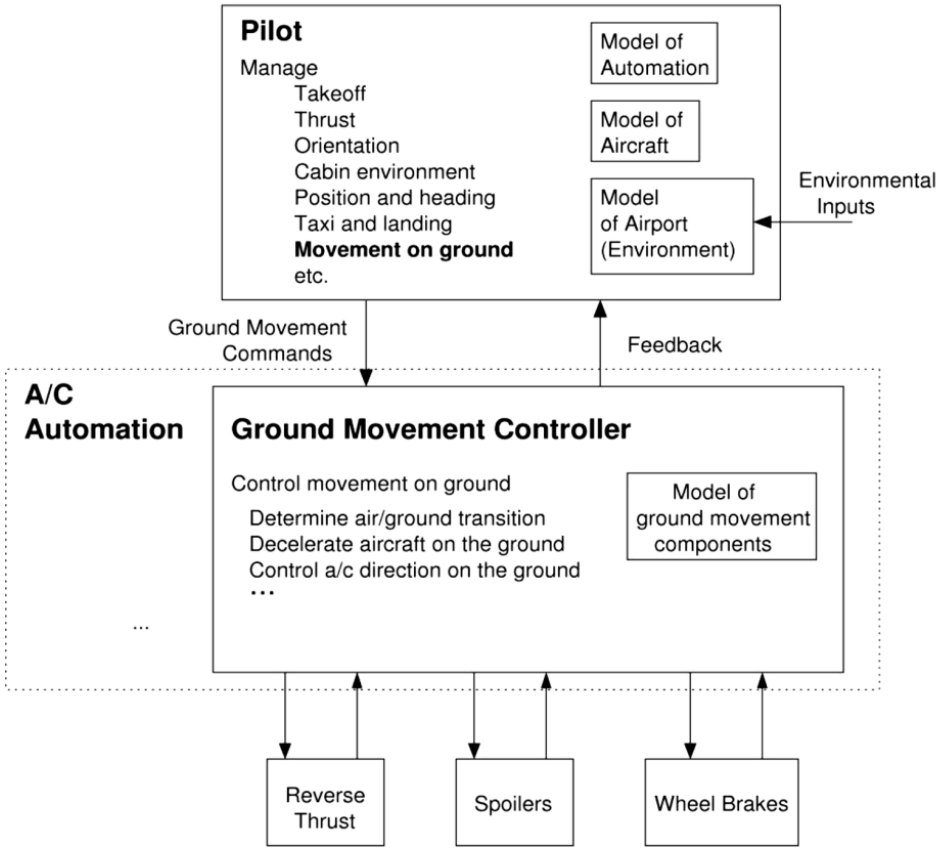


Figure 3.4: Deceleration Control Structure

(H4-4)

SC5: 항공기가 안전한 지역(유도로, 활주로, 터미널 게이트, 주기장 등) 밖으로 의도치 않게 기동되어서는 안 된다 (H4-5)

SC6: 랜딩 기어를 접어 넣었을 때, 주 기어 회전은 중지되어야 한다 (H4-6)

이것들은 모두 매우 상위 수준이며 다시 말해, 항공기 전체를 고려하는 것에서 시작한 것이다. 이전 장에서 설명한 것처럼 이 요구사항은 반복적인 과정을 통해 특정 시스템 컴포넌트(승무원, 소프트웨어, 컴포넌트 상호작용을 포함함)와 연관된 보다 세부적인 기능 안전 요구사항의 집합으로 상세화될 것이다. 시스템 안전 요구사항은 STPA에 의해 식별된 원인 시나리오를 방지하기 위해 생성되는데 이 과정은 설계 과정에서 위험이 모두 적절히 분석되고 다루어질 때까지 계속된다. 중요한 것은 처음부터 전체적인 관점에서 시스템 동작을 고려하므로 컴포넌트 간의 잠재적으로 안전하지 않은 상호작용이 프로세스 시작 시점에 식별 가능하다는 점이며, 나중에 모든 상호작용을 고려하여 위험 여부를 판단할 필요가 없다는 것이다.

그림 3.4는 감속이 이루어지는 지상 제어 기능의 컨트롤 스트럭처 모델을 확대한 것이다. 제어되는 세 가지 기본적인 물리적 컴포넌트는 다음과 같다: 역방향 추진력, 스포일러 및 휠 브레이크 시스템. 다시 말하지만, STPA는 단순히 이들 중 하나가 아닌, 보다 큰 컨트롤 스트럭처를 포함함으로써 분석중인 위험과 관련된 브레이킹 컴포넌트 간의 모든 상호작용(의도한 것과 의도하지 않은 것 모두)을 고려할 수 있다. 브레이크 시스템이 역방향 추진력, 스포일러 그리고 휠 브레이크를 포함한다고 결정한 점에서 설계에 대한 결정이 있었던 것은 분명하다. 하지만, 이러한 수준의 설계 세부 사항은 매우 일반적이고 상위 수준이며 대부분의 항공기에 적용되는 사항이라 하겠다.

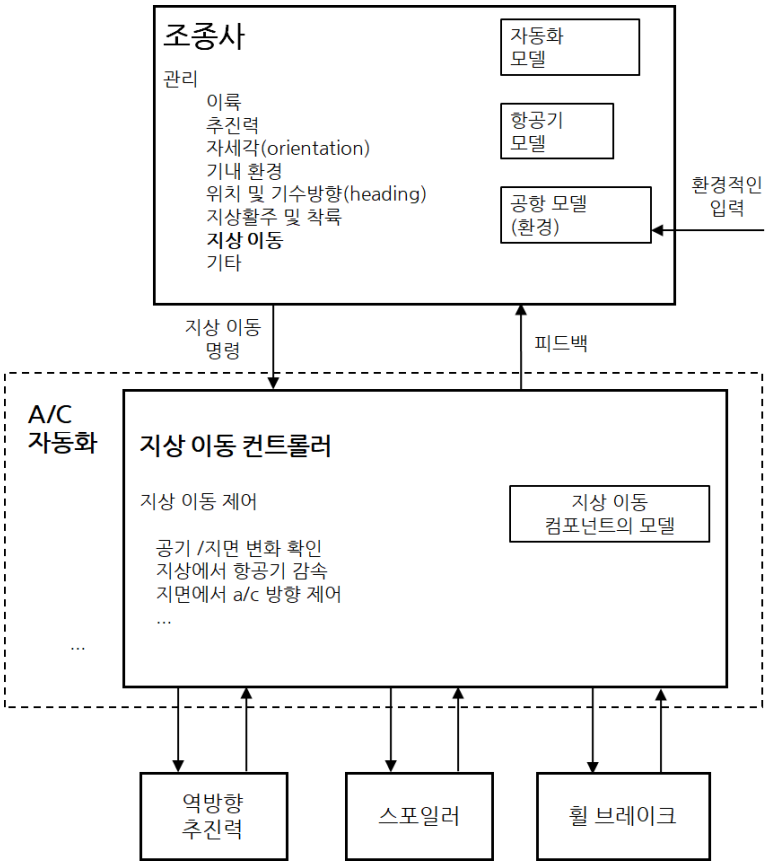


그림 3.4 : 감속 컨트롤 스트럭처

Examples of safety constraints on the interaction between the braking system components that can be generated by STPA at this point in development include:

SC-BS-1: Spoilers must deploy when the wheel brakes are activated manually or automatically above TBD speed.

SC-BS-2: Wheel brakes must activate upon retraction of landing gear.

SC-BS-3: Activation of ground spoilers must activate armed automatic braking (autobrake) system.

SC-BS-4: Automatic braking system must not activate wheel brakes with forward thrust applied.

SC-BS-5: Automatic spoiler system must retract the spoilers when forward thrust is applied.

Note that because STPA includes humans, such as operators, maintainers, and even managers, STPA provides a structured method to identify human errors influenced by the system design, such as mode confusion and loss of situational awareness, that can lead to hazards. These human errors can arise due to a loss of consistency (lack of synchronization) between the actual automation state and the operator's mental model of that state.

Figure 3.5 further zooms into the deceleration control structure by focusing on the functional structure of the wheel brake system. Again, although only one component of the braking system is considered in Figure 3.5, there are no assumptions about the physical design and implementation. STPA starts without a specific design solution to potential problems. Instead, it starts from the basic required functional behavior and identifies the conditions under which that behavior can be hazardous. Designers can later decide on particular design solutions, such as redundancy or redesign of the functions, necessary to satisfy the safety requirements derived through this analysis.

Figure 3.5 does include one important design decision to include an autobraking function, which is common in modern aircraft. Pilots can preset an autobrake before they land in order to ensure consistent braking action is applied. Autobraking can reduce pilot workload under gusty or icy conditions and greatly reduce brake wear. If set, the autobrake will engage the wheel brakes at the specified time.

이 시점에서 STPA로 만들 수 있는 브레이크 시스템 컴포넌트 간의 상호작용에 대한 안전 제약사항 예는 다음과 같다.

SC-BS-1: 속도가 TBD 이상인 상태에서 휠 브레이크가 수동 또는 자동으로 활성화되는 경우에는 스포일러를 전개해야 한다.

SC-BS-2: 랜딩 기어를 집어넣을 때 휠 브레이크가 반드시 활성화되어야 한다.

SC-BS-3: 지상 스포일러(ground spoiler)의 활성화는 작동개시된 자동 브레이킹(오토브레이크) 시스템을 활성화시켜야 한다.

SC-BS-4: 자동 브레이킹 시스템은 정방향 추진력이 적용된 휠 브레이크를 활성화시키지 않아야 한다.

SC-BS-5: 자동 스포일러 시스템은 정방향 추진력이 적용된 때에는 스포일러를 접어야 한다.

STPA는 운영자, 유지보수자, 관리자와 같은 사람을 포함하므로 위험을 초래할 수 있는 모드 혼동, 상황인식 실패와 같은 시스템 설계의 영향을 받는 인적 오류(human error)를 식별하기 위한 체계화된 방법을 제공한다. 이러한 인적 오류는 실제 자동화 상태와 그 상황에 대한 운영자의 멘탈 모델 간의 일관성 손실(동기화 부족)로 인해 발생할 수 있다.

그림 3.5는 휠 브레이크 시스템의 기능적 구조에 초점을 맞추어 감속 컨트롤 스트럭처를 더욱 확대한 것이다. 다시 말해, 그림 3.5에는 브레이크 시스템의 한 가지 컴포넌트만 고려되었으며, 물리적 설계 및 구현에 대한 가정이 없다. STPA는 잠재적 문제에 대한 구체적인 설계 솔루션 없이 시작할 수 있다. 대신, 기본적인 필수 기능 동작에서 분석을 시작하여 해당 동작이 위험할 수 있는 조건을 식별한다. 설계자는 이 분석을 통해 도출된 안전 요구사항을 충족시키는 데 필요한 다중화 또는 기능 재설계(redesign)와 같은 특정 설계 솔루션을 나중에 결정할 수 있다.

그림 3.5는 오토브레이킹 기능을 시스템에 포함하는 한 가지 중요한 설계 결정을 보여주는데, 이 기능은 오늘날 항공기에서 일반적인 기능이다. 조종사는 브레이킹 액션이 일관되게 적용되도록 착륙 전에 오토브레이크를 미리 설정할 수 있다. 오토브레이크는 바람이 불거나 얼음이 많은 곳에서 조종사의 작업 부하를 줄이고 브레이크 마모를 크게 줄일 수 있다. 설정한 경우, 오토브레이크가 지정된 시간에 휠 브레이크를 작동시킨다.

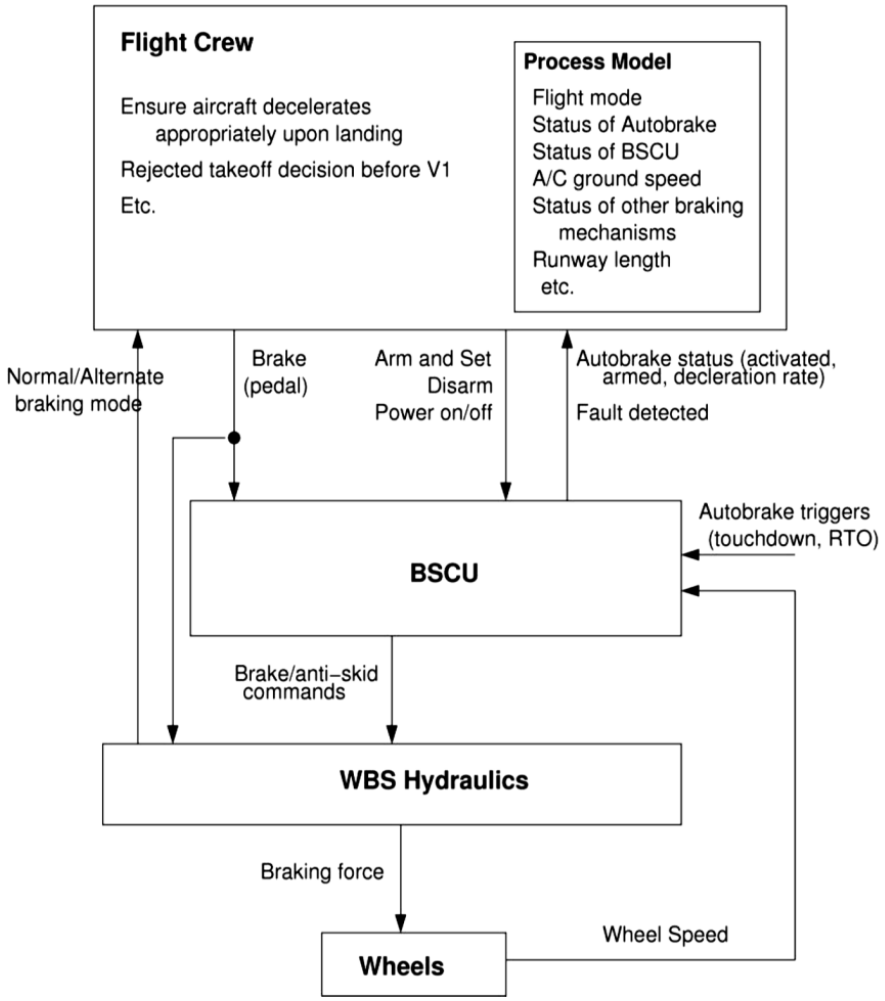


Figure 3.5: The Control Structure for the Wheel Braking System

Table 3.1 shows some of the safety constraints generated by STPA for Figure 2.6 for the flight crew (FC) and the Brake System Control Unit (BSCU). These are derived from the UCA Context Table generated as described in earlier chapters of this handbook. Rationale is included in Table 3.1 as is traceability information to assist in changing design decisions later without introducing unsafe features.

Table 3.1: Example STPA Generated System-Level Safety Constraints

Unsafe Control Action	Description	Rationale
FC-R1	Crew must not provide manual braking before touchdown [CREW.1c1]	Could cause wheel lockup, loss of control, or tire burst
FC-R2	Crew must not stop manual braking more than TBD seconds before safe taxi speed reached [CREW.1d1]	Could result in overspeed or runway overshoot
FC-R3	The crew must not power off the BSCU during autobraking [CREW.4b1]	Autobraking will be disarmed
BSCU-R1	A brake command must always be provided during RTO [BSCU.1a1]	Could result in not stopping within the available runway length

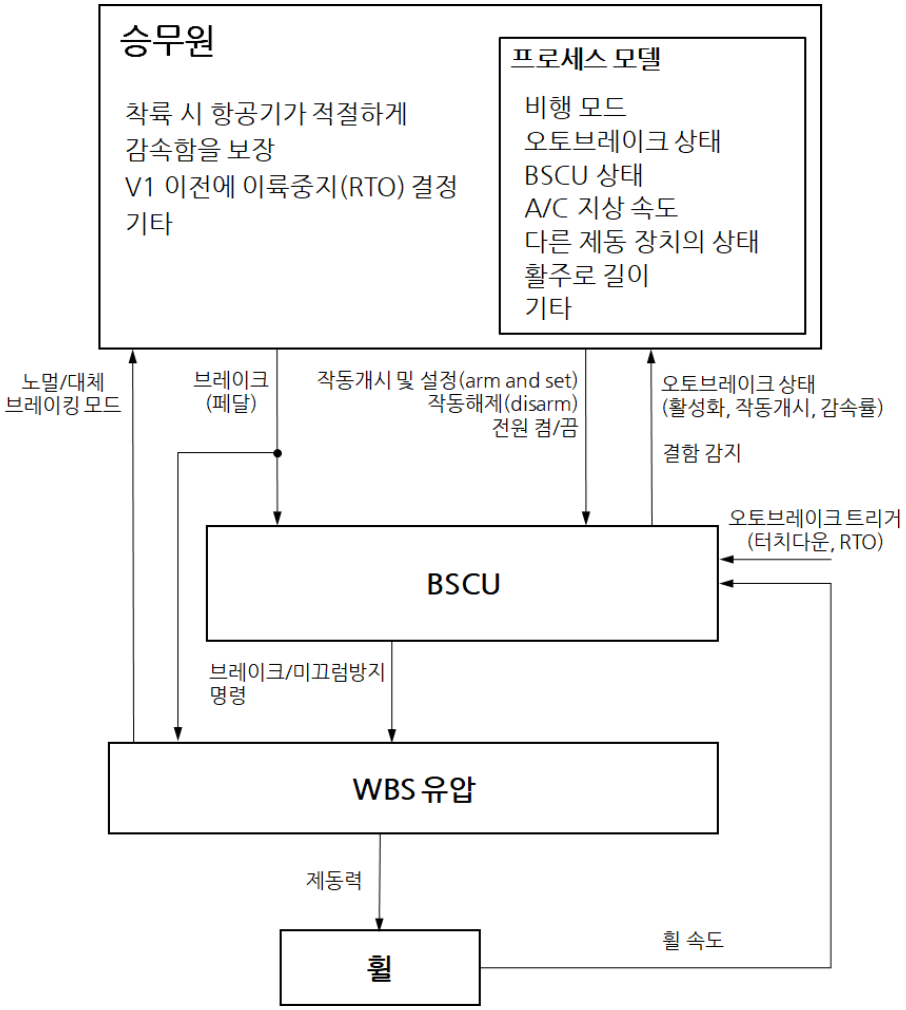


그림 3.5 : 휠 브레이크 시스템을 위한 컨트롤 스트럭처

표 3.2는 그림 2.6에 대해 승무원(FC, Flight Crew)과 브레이크 시스템 컨트롤 유닛(BSCU, Brake System Control Unit)의 안전 제약사항을 STPA를 사용하여 생성한 것의 일부이다. 이 핸드북의 앞장에서 설명한대로 이 제약사항들은 UCA 컨텍스트 테이블에서 도출된다. 표 3.2의 '근거'는 이후 설계 결정을 변경할 때 안전하지 않은 속성이 생기지 않도록 도와주는 추적성 정보로서 포함되었다.

표 3.2 : STPA를 통해 생성한 시스템 수준 안전 제약사항의 예

UCA	설명	근거
FC-R1	승무원은 터치다운 전에 매뉴얼 브레이킹을 해서는 안 된다 [CREW.1c1]	휠 잠김, 제어상실 또는 타이어 파열의 원인이 될 수 있음
FC-R2	승무원은 안전한 지상활주 속도에 도달하기 전에 TBD 초 이상 매뉴얼 브레이킹을 멈추지 않아야 한다 [CREW.1d1]	과속 또는 활주로를 벗어나는(overshoot) 결과를 초래할 수 있음
FC-R3	승무원은 오토브레이킹 중에는 BSCU의 전원을 끄면 안 된다 [CREW.4b1]	오토브레이킹이 작동해제(disarm)됨
BSCU-R1	이륙중지(RTO) 중에는 브레이크 명령이 항상 제공되어야 한다 [BSCU.1a1]	사용 가능한 활주로 길이 내에서 정지하지 않을 수 있음

BSCU-R2	Braking must never be commanded before touchdown [BSCU.1c1]	Could result in tire burst, loss of control, injury, or other damage
BSCU-R3	Wheels must be locked after takeoff and before landing gear retraction [BSCU.1a4]	Could result in reduced handling margins from wheel rotation in flight

The high-level system architecture could be defined using the system requirements and these safety constraints at this point in development. The STPA-generated causal scenarios for violation of these high-level safety constraints will provide more detailed design constraints and other information to assist in the detailed architectural and system design process.

As design decisions are made, the STPA analysis is continually iterated and refined to assist the designer in making tradeoffs and effective design decisions. For the braking example, a more detailed control structure that shows some design decisions (including the valves used by the hydraulic controller) is shown in Figure 3.6. The design decisions, involving the additional of valves and commands related to the hydraulic controller might be at least partially the result of resolving the hazardous scenarios already identified.

Continuing the STPA analysis at this more detailed level of design involves the identification of unsafe control actions and the associated safety constraints related to the BSCU hydraulic controller (HC) when controlling the three individual valves included in the refined design (Table 3.2):

Table 3.2: Example STPA Generated System-Level Safety Constraints On the Wheel Brake System Control Structure

Unsafe Control Action	Description	Rationale
HC-R1	The HC must not open the green hydraulics shutoff valve when there is a fault requiring alternate braking [HC.1b1]	Both normal and alternate braking would be disabled.
HC-R2	The HC must pulse the anti-skid valve in the event of a skid [HC.2a1]	Anti-skid capability is needed to avoid skidding and to achieve full stop in wet or icy conditions.
HC-R3	The HC must not provide a position command that opens the green meter valve when no brake command has been received [HC.3b1]	Crew would be unaware that uncommanded braking was being applied.
HC-R4	The HC anti-skid must not release brakes when on low speed taxi.	Anti-skid system senses skid by sensing low wheel rotation or a rapid change in wheel rotation speed. This could lead to inadvertent release of brakes during low-speed taxi.

Once again, traceability is maintained with the unsafe control actions and scenarios from which these functional design constraints were derived. Those control actions and scenarios are in turn traceable back to higher levels of design and analysis. The result is an elaborate tree structure of a tracing of the STPA analysis throughout the design process that will assist in making changes or in understanding where certain constraints came from.

UCA	설명	근거
BSCU-R2	터치다운 전에 브레이크 명령이 내려져서는 안 된다 [BSCU.1c1]	타이어 파열, 통제상실, 부상 또는 다른 손상을 초래할 수 있음
BSCU-R3	이륙 후 랜딩기어를 접기 전에 바퀴를 잠가야(lock) 한다 [BSCU.1a4]	비행 중 휠 회전으로 인한 핸들링 마진 감소를 초래할 수 있음

개발 단계의 이 시점에서 시스템 요구사항과 이러한 안전 제약사항을 사용하여 상위 수준의 시스템 아키텍처를 정의할 수 있다. STPA에서 도출된, 이러한 상위 수준의 안전 제약사항을 위반하는 원인 시나리오는 보다 상세한 설계 제약사항과 기타 다른 정보를 제공하며, 이는 상세 아키텍처 및 시스템 설계 프로세스에 도움이 된다.

설계 결정이 이루어짐에 따라 설계자가 트레이드오프와 효과적인 설계 결정을 내리는 것을 지원할 수 있도록 STPA 분석이 지속적으로 반복되고 상세화된다. 브레이킹 예제에 대해 몇 가지 설계 결정(유압 컨트롤러가 사용하는 밸브 포함)을 보여주는 보다 상세한 컨트롤 스트럭처를 그림 3.6에 보였다. 유압 컨트롤러와 관련된 추가 밸브와 명령에 대한 설계 결정은 적어도 이미 식별한 위험 시나리오를 부분적으로 해결한 결과일 수 있다.

이러한 보다 상세한 수준의 설계에서 STPA 분석을 계속하기 위해서는 상세화된 설계에 포함된 3개의 개별 밸브를 제어할 때, BSCU 유압 컨트롤러(HC)와 관련된 UCA 및 연관된 안전 제약사항을 식별해야 한다(표 3.3):

표 3.3 : 휠 브레이크 시스템 컨트롤 스트럭처에서 STPA를 통해 생성한 시스템 수준 안전 제약사항의 예

UCA	설명	근거
HC-R1	HC는 대체 브레이킹이 필요한 결함이 있는 경우, 녹색 유압차단 밸브를 열면 안 된다[HC.1b1]	노멀 브레이킹 및 대체 브레이킹 모두 작동하지 않을 것이다.
HC-R2	HC는 미끄럼 발생 시 미끄럼 방지 밸브를 작동시켜야 한다[HC.2a1]	미끄럼 방지 기능은 미끄러짐을 방지하고, 젖은 상태이거나 얼음이 많은 조건에서 완전히 정지하기 위해 필요하다.
HC-R3	HC는 브레이크 명령이 수신되지 않았으면 녹색 계기밸브를 여는 위치 명령을 제공해서는 안 된다[HC.3b1]	승무원은 명령하지 않은 브레이크가 적용되고 있음을 모른다.
HC-R4	저속으로 지상활주 하는 경우, HC 미끄럼 방지 브레이크가 해제되어서는 안 된다	미끄럼 방지 시스템은 적은 바퀴회전 또는 바퀴 회전속도의 빠른 변화를 감지하여 미끄럼을 감지한다. 이로 인해 저속 지상 활주 중 실수로 브레이크가 풀릴 수 있다.

아울러, 이러한 기능적 설계 제약사항이 도출된 UCA 및 시나리오에 대해서도 추적성이 유지된다. 그러한 컨트롤 액션 및 시나리오는 상위 수준의 설계 및 분석으로 순차적으로 다시 추적할 수 있다. 그 결과, 설계 프로세스 전반을 거쳐 STPA 분석을 추적할 수 있고, 이는 특정 제약사항이 어디서 발생했는지 이해하거나 변경하는 데 도움이 된다.

또한 UCA와 관련된 보다 자세한 원인 시나리오를 생성하고 새로운 시나리오를 제거하거나 완화하는 방법에 대한 더 많은 설계 결정을 내릴 수 있다. 대안이 될 수 있는 설계 사이의 트레이드오프(tradeoff)를 해결하기 위해 대체 설계들을 분석해 볼 수 있다. 이 프로세스는 전체 시스템 개발 프로세스에 걸쳐 계속된다.

Additionally, more detailed causal scenarios associated with the unsafe control actions can be generated along with more design decisions about how to eliminate or mitigate the new scenarios. Alternative designs might be analyzed to provide information about how to resolve tradeoffs between alternative design choices. The process continues throughout the entire system development process.

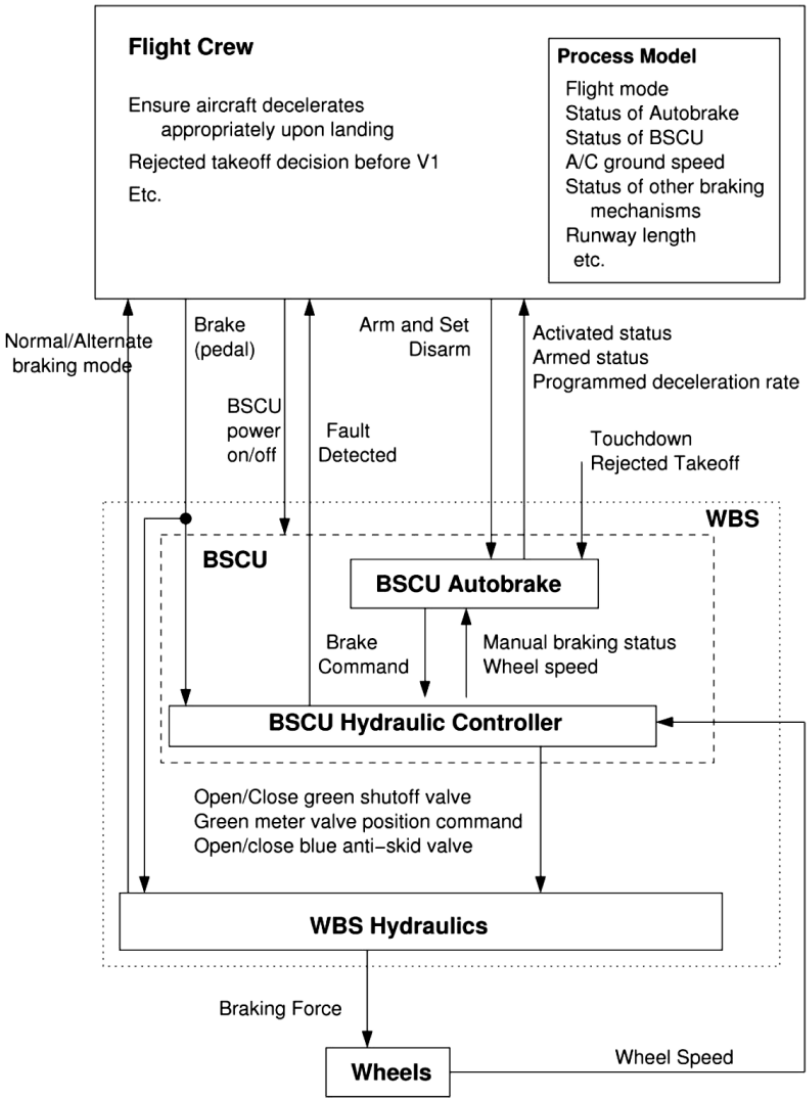


Figure 3.6: A More Detailed Version of the Wheel Brake System Control Structure

7. System Integration

System integration should be a lot less painful (at least from a safety standpoint) if STPA is used to identify the integrated system safety requirements early in the development process. Unsafe interactions among components, rather than being found now, should have been designed out of the system before this late development phase. Any safety-related integrated system flaws found at this point should be traced back to flaws in the development process (described above) and eliminated from future development projects. Of course, the flaws found will need to be incorporated into the STPA analysis to identify constraints to eliminate the flaws.

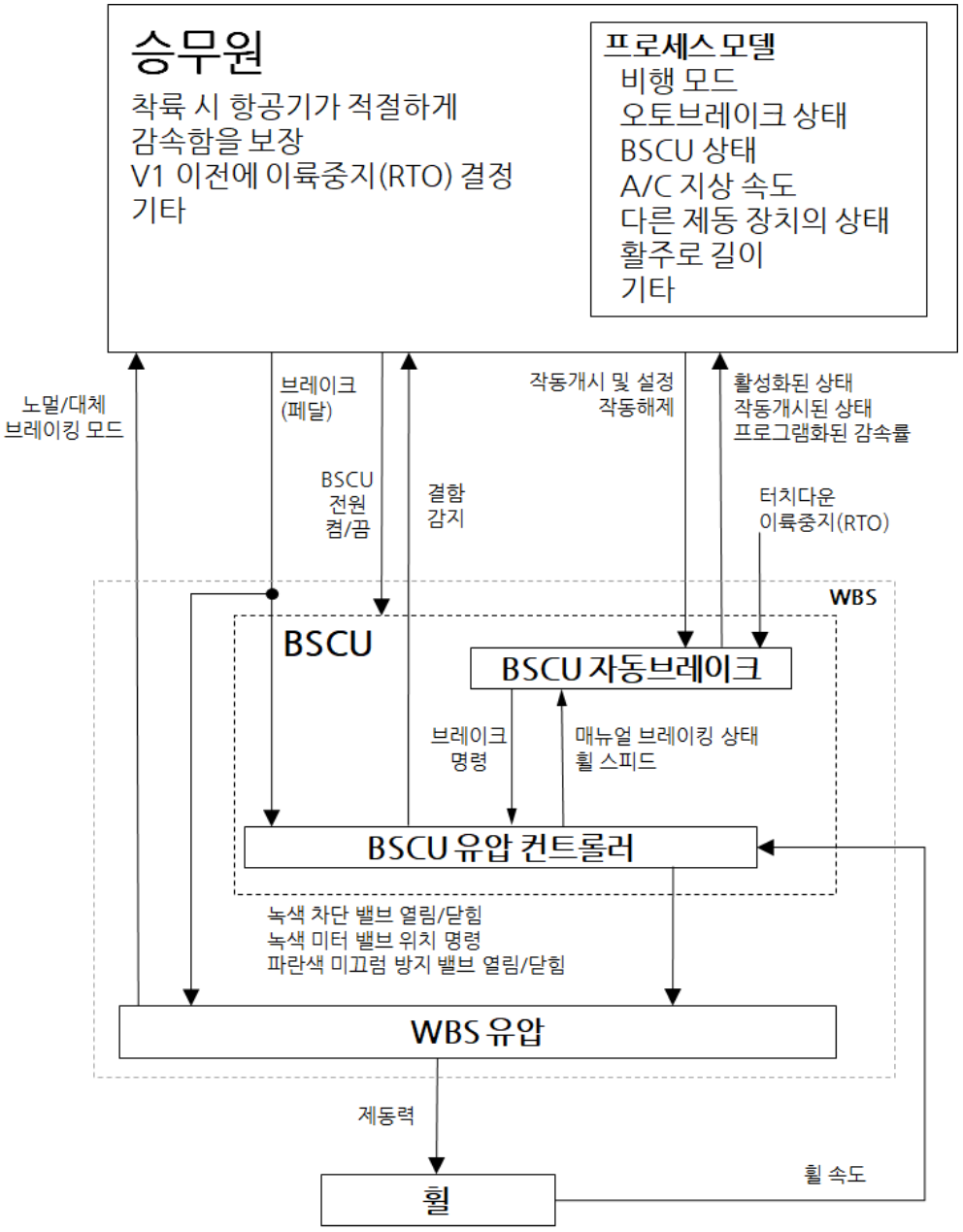


그림 3.6 : 휠 브레이크 시스템 컨트롤 스트럭처의 상세 버전

7. 시스템 통합

STPA를 사용하여 개발 프로세스 초기에 통합된 시스템의 안전 요구사항을 식별하게 되면 (적어도 안전 측면에서 볼 때) 시스템 통합은 훨씬 덜 힘들게 된다. 컴포넌트 간의 안전하지 않은 상호작용은 시스템 통합 단계에 이전에 발견되고 시스템에서 제외되도록 설계되는 것이 바람직하다. 이 시점에 발견된 모든 안전-관련 통합 시스템 결함은 (위에서 설명한) 개발 프로세스에서의 결함으로 추적되어야 하며, 향후 개발 프로젝트에서 제거해야 한다. 물론 발견된 결함은 결함 제거를 위한 제약사항을 식별할 수 있도록 STPA 분석에 포함되어야 한다.

8. Generation of System Test and Evaluation Requirements

The causal scenarios and other information generated in the STPA analysis during the activities on the left-side of the "V" can be used to generate test requirements and evaluation programs. The test requirements and planning will depend on the type of system and testing being done.

As an example, Dan Montes, a U.S. Air Force test pilot, in his MIT Ph.D. dissertation, describes how STPA can be used in aircraft flight testing. Figure 3.7 shows a version of the example safety control structure in Figure 3.2 augmented with a separate testing control structure embedded between the development and operations control structures. Note that new hazards, unsafe control actions, and causal scenarios may exist in the flight test program. For example, new contexts may need to be considered for the test flight environment that might not exist in the normal operational environment (e.g., testing the results of abnormal stress on the aircraft or applying extreme control actions to test for recoverability). STPA can be applied to this system testing control structure, just as it is on any control structure, and the results used for flight test safety planning and in generating flight test plans.

9. Control of Manufacturing (Production Engineering, Workplace Safety)

STPA has been applied successfully to both production engineering and to workplace safety. Considerations of producibility (particularly with respect to safety in manufacturing) should be included in the original STPA analysis if manufacturing and design for manufacturability is important for the particular system being designed. The application of STPA to workplace safety, as well as its use in organizational and managerial analysis, is described in separate chapters of this handbook.

10. Generation of Operational Safety Requirements (including leading indicators of increasing risk and the safety management plan)

The causal scenarios that cannot be eliminated or adequately controlled in the system design must be passed to the system operators so that they can implement operational controls. Leading indicators of increasing risk, described in Chapter 6 of this handbook, can be derived from the STPA analysis generated during the development process or during an operational safety analysis. The goal of these leading indicators is to identify flawed assumptions about the use environment during design, hopefully before any serious loss occurs, and also identify when the behavior of the system changes over time and violates the original design assumptions about usage.

Requirements on system operators and maintainers can be derived directly from the STPA analysis performed during development as both are included as components in the system design. The causal scenarios will provide input to the operational requirements.

8. 시스템 테스트 및 평가 요구사항의 생성

"V-모델" 왼쪽에 있는 활동 수행 중, STPA 분석을 통해 도출된 원인 시나리오 및 다른 정보는 테스트 요구사항 및 평가 프로그램을 생성하는 데 사용할 수 있다. 테스트 요구사항 및 계획은 시스템 종류 및 이미 실시한 테스트에 따라 달라진다.

예를 들어, 미 공군 파일럿인 댄 몬테스(Dan Montes)는 MIT 박사 학위 논문에서 STPA가 항공기의 비행 시험(flight test)에 어떻게 사용될 수 있는지에 대해 설명하고 있다. 그림 3.7은 그림 3.2의 안전 컨트롤 스트럭처에서 개발 및 운영 컨트롤 스트럭처 사이에 별도의 테스트 컨트롤 스트럭처를 추가한 것이다. 새로운 위험, 안전하지 않은 컨트롤 액션 및 원인 시나리오가 비행 시험 프로그램에 존재할 수도 있다는 점에 유의한다. 예를 들어, 정상 운영 환경이 아닌 테스트 비행 환경에서는 새로운 컨텍스트를 고려해야 할 수도 있다(예: 항공기의 비정상적인 상황에 대한 스트레스 테스트 또는 복구 가능성 테스트를 위한 극단적인 컨트롤 액션 적용). STPA는 모든 컨트롤 스트럭처에서와 마찬가지로 이러한 시스템 테스트 컨트롤 스트럭처에 적용할 수 있으며 그 결과는 비행 시험 안전 계획 및 비행 시험 계획 생성에 사용된다.

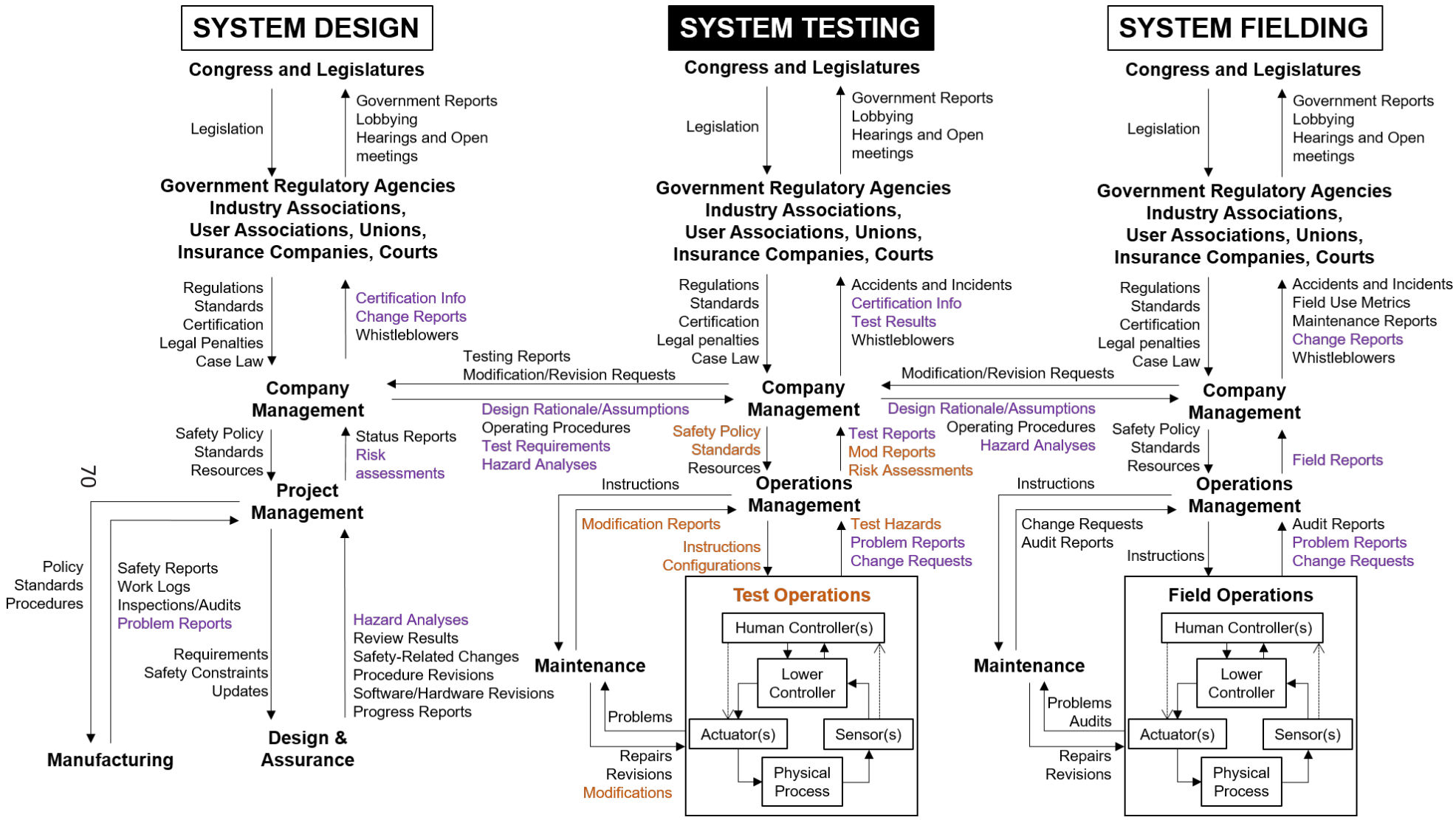
9. 제조(제조 공학, 작업장 안전) 컨트롤

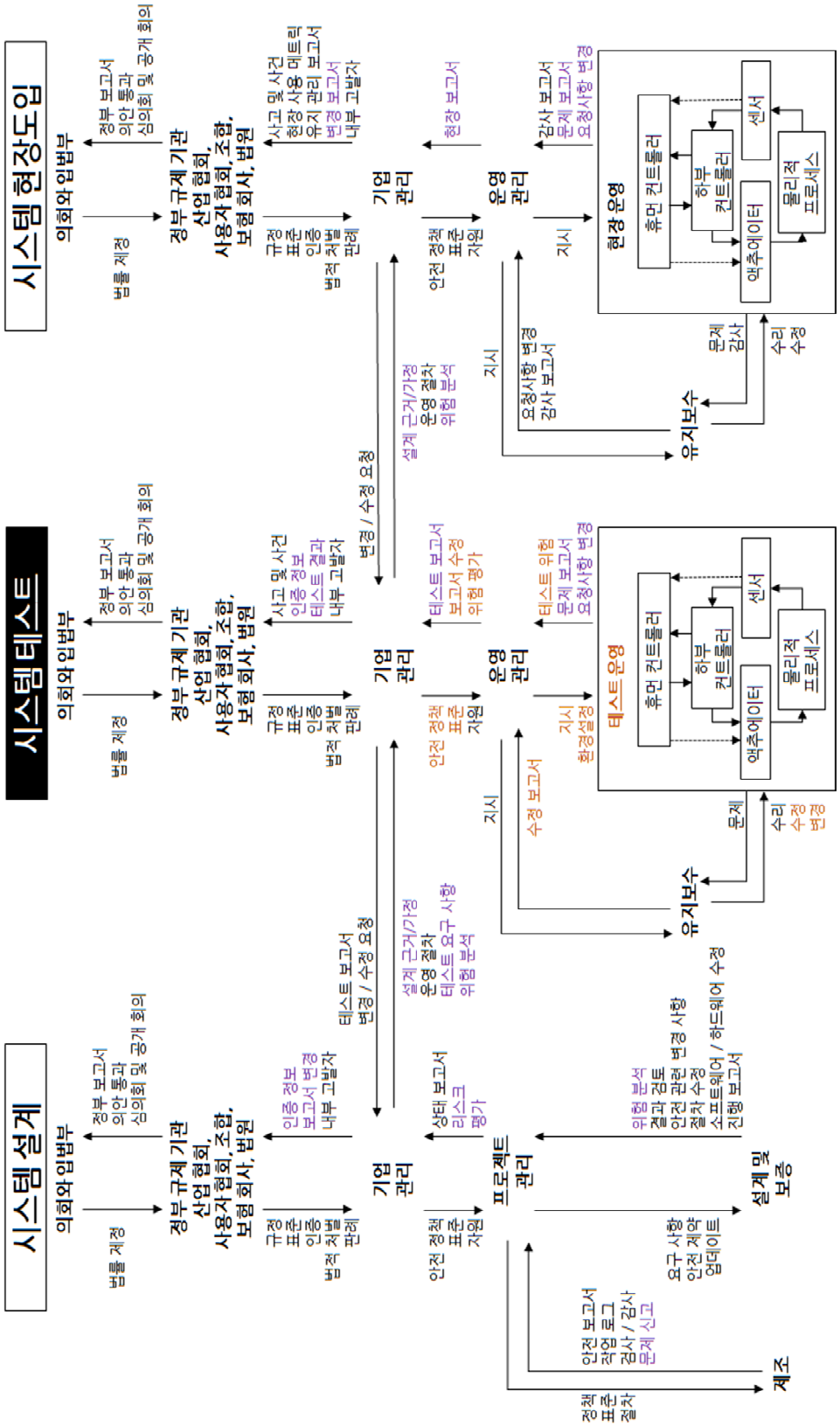
STPA는 지금까지 제조 공학(product engineering)과 작업장 안전(workplace safety)에도 성공적으로 적용되어 왔다. 설계중인 특정 시스템에서 제조 및 제조 능력(manufacturability)에 대한 설계가 중요한 경우, STPA 분석에 생산가능성(producibility)(특히 제조 안전성과 관련하여)을 포함해야 한다. STPA를 조직 및 경영 분석에서 사용하거나 작업장 안전에 적용하는 것은 이 핸드북에 별도로 설명되어 있다.

10. 운영 안전 요구사항 생성(증가하는 위험에 대한 선행지표 및 안전 관리 계획 포함)

시스템 설계에서 제거되거나 적절하게 제어할 수 없는 원인 시나리오는 시스템 운영자가 제어할 수 있도록 시스템 운영자에게 전달되어야 한다. 이 핸드북 6장에서 설명하고 있는 리스크 증가에 대한 선행지표는 개발 프로세스 또는 운영 안전 분석 중에 생성된 STPA 분석 결과에서 도출될 수 있다. 이러한 선행지표의 목표는 심각한 손실이 발생하기 전에 설계 과정에서의 사용 환경에 대한 잘못된 가정을 식별하는 것이다. 또한 시간이 지남에 따라 시스템의 동작이 변하고 사용(usage)에 대한 초기의 설계 가정이 위배되는 시점을 식별하는 것이다.

시스템 설계에 운영자와 유지보수자가 모두 컴포넌트로 포함되었으므로 시스템 운영자와 유지보수자에 대한 요구사항은 개발 중에 수행된 STPA 분석 결과에서 직접 도출할 수 있다. 원인 시나리오는 운영 요구사항에 대한 입력 정보(input)를 제공한다.


SYSTEM DESIGN
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal Penalties
Case Law
Certification Info
Change Reports
Whistleblowers
Company Management
Testing Reports
Modification/Revision Requests
Design Rationale/Assumptions
Operating Procedures
Test Requirements
Hazard Analyses
Safety Policy
Standards
Resources
Status Reports
Risk assessments
Project Management
Policy
Standards
Procedures
Safety Reports
Work Logs
Inspections/Audits
Problem Reports
Requirements
Safety Constraints
Updates
Hazard Analyses
Review Results
Safety-Related Changes
Procedure Revisions
Software/Hardware Revisions
Progress Reports
Manufacturing
Design & Assurance
SYSTEM TESTING
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal penalties
Case Law
Accidents and Incidents
Certification Info
Test Results
Whistleblowers
Company Management
Design Rationale/Assumptions
Operating Procedures
Hazard Analyses
Safety Policy
Standards
Resources
Test Reports
Mod Reports
Risk Assessments
Operations Management
Instructions
Modification Reports
Instructions
Configurations
Test Hazards
Problem Reports
Change Requests
Test Operations
Human Controller(s)
Lower Controller
Actuator(s)
Sensor(s)
Physical Process
Maintenance
Problems
Repairs
Revisions
Modifications
SYSTEM FIELDING
Congress and Legislatures
Legislation
Government Reports
Lobbying
Hearings and Open meetings
Government Regulatory Agencies
Industry Associations,
User Associations, Unions,
Insurance Companies, Courts
Regulations
Standards
Certification
Legal penalties
Case Law
Accidents and Incidents
Field Use Metrics
Maintenance Reports
Change Reports
Whistleblowers
Modification/Revision Requests
Company Management
Design Rationale/Assumptions
Operating Procedures
Hazard Analyses
Safety Policy
Standards
Resources
Field Reports
Operations Management
Instructions
Change Requests
Audit Reports
Instructions
Audit Reports
Problem Reports
Change Requests
Field Operations
Human Controller(s)
Lower Controller
Actuator(s)
Sensor(s)
Physical Process
Maintenance
Problems
Audits
Repairs
Revisions


시스템 현장도입
시스템 테스트
시스템 설계
의회와 입법부
법률 제정
정부 규제 기관
산업 협회,
사용자 협회, 조합,
보험 회사, 법원
규정
표준
인증
법적 처벌
관례
기업
관리
안전 정책
표준
자원
운영
관리
프로젝트
관리
설계 및
보증
제조
유지보수
현장 운영
휴먼 컨트롤러
하부
컨트롤러
액추에이터
센서
물리적
프로세스
테스트 운영
변경 / 수정 요청
설계 근거/가정
운영 절차
위험 분석
테스트 요구 사항
위험 분석
안전 보고서
작업 로그
검사 / 감사
문제 신고
요구 사항
안전 제약
업데이트
위험 분석
결과 검토
안전 관련 변경 사항
절차 수정
소프트웨어 / 하드웨어 수정
진행 보고서
정책 표준 절차
문제
수리
수정 변경
지시
수정 보고서
환경설정
테스트 위험
문제 보고서
요청사항 변경
요청사항 변경
감사 보고서
문제 감사
수리
수정

11. Operational safety management including monitoring leading indicators

An operational risk management or safety management system should exist in every operational environment. STPA results can be used to create the operational safety plan and safety requirements. In addition, it can play a major role in identifying leading indicators and monitoring the operation of the system to detect migration to states of higher risk (see the chapter in the handbook on this topic).

STPA can also play a role in incident and accident analysis and change management. Changes such as system design changes (upgrades or mods), context changes (using the system in a different environment or for a different purpose), and operations support changes (logistics supply chain or maintenance practices) need to be considered as well as others.

Planned changes must always be subjected to a hazard analysis. This process should be made easier with the traceability embedded into STPA so that change impact can be traced to specific system hazards or shown not to impact them. Unplanned changes are handled through leading indicators generated from the STPA analysis.

11. 선행지표 모니터링을 포함하는 운영 안전 관리

운영 리스크 관리 또는 안전 관리 시스템은 모든 운영 환경에 존재해야 한다. STPA 결과는 운영 안전 계획 및 안전 요구사항을 작성하는 데 사용될 수 있다. 또한 선행지표를 식별하고 시스템 운영을 모니터링하여 리스크가 높은 상태로 이전되는 것을 탐지하는 데 중요한 역할을 할 수 있다(본 주제에 대한 핸드북의 다른 장 내용 참조).

STPA는 사건, 사고 분석 및 변경 관리에서도 역할을 수행할 수 있다. 시스템 설계 변경(업그레이드 또는 개조), 컨텍스트 변경(다른 환경에서 또는 다른 목적으로 시스템 사용) 및 운영 지원 변경(물류 공급망 또는 유지보수 관행)과 같은 변경 사항도 고려해야 한다.

계획된 변경 사항에 대해서는 항상 위험 분석을 수행해야 한다. 이 프로세스는 STPA에 포함된 추적성으로 보다 쉽게 이루어질 수 있으므로 변경의 영향이 특정 시스템의 위험에 도달할 수 있는 것인지 그렇지 않은 것인지를 추적할 수 있다. 계획되지 않은 변경 사항은 STPA 분석 에서 생성된 선행지표를 통해 처리된다.

Chapter 4: Workplace Safety using STPA

Nancy Leveson

Workplace or occupational safety has often focused on the worker's behavior and not gone beyond that. A system's approach to workplace safety assumes that human behavior is affected by the system in which it is embedded. The goal is to identify how to design the work environment in order to reduce human error. The application of STPA in this environment is identical to the standard application of STPA on a more technological system, but there are not many examples in the open literature so we have included this chapter of the handbook.

Two examples are shown included: one involves people working with advanced automation and the other involves the less technological workplace hazard involving the use of lockout/tagout (LOTO). In both cases, the major difference between STPA and the standard approach to workplace safety today is that STPA focuses on more than just creating procedures for humans to follow but instead looks at the entire system to identify hazardous scenarios and how to eliminate or reduce them.

Because the first example below describes a real experimental application of STPA in a manufacturing environment, details about the actual assessment procedure used and the cost can be provided.

STPA Example for a Semi-Automated Manufacturing Process

The example here comes directly from the master's thesis of Nathaniel Peper,[17] in which he applied STPA to a real aircraft assembly process where advanced automation (robotics) was being introduced into the workplace. He describes the process as follows:

> *"… the process begins with sub-components for the aircraft entering the factory on the back of a vehicle operated by a worker. Within the factory, two more workers are each responsible for each driving an automated ground vehicle (AGV) by handheld control from an AGV docking and charging location within the factory to connection points on a Product Transportation Vehicle (PTV) for the incoming sub-component. After connecting to the PTV, the two drivers must collaboratively drive the combined Product Transportation System (PTS) around the factory to the location of the sub-component on the first mentioned vehicle. At this point, another group of workers will then work together to move the sub-component from the vehicle to the PTS. The AGV operators will then drive the PTS through the factory to a location at which the sub-component can be prepared for entry into an automated manufacturing cell. Inside the cell are multiple robots and associated support equipment that will perform the automated process of drilling holes in the sub-component and filling the holes with additional components. When the sub- component, the PTS, the robotic cell, and the operators are all ready, the cell operator and controller will assume control of the PTS and move it through the automated process to perform its work statement. During this time, there will also be concurrent work occurring to portions of the sub-component that are not inside the robotic cell perimeter. Once the entire automated process is complete, workers will re-assume control of the PTS and move it down the factory line to the next position for more manual work processes."*

The accident or loss here is:

[17] Nathaniel Peper, Systems Thinking Applied to Automation and Workplace Safety, MIT Masters Thesis (Leaders for Global Operations Program), June 2017.

4장 : STPA를 사용한 작업장 안전

Nancy Leveson

작업장(workplace) 또는 직업 안전(occupational safety)은 주로 근로자의 행동에만 초점을 맞춰왔고 그 이상으로는 수행되지 않았다. 작업장 안전에 대한 시스템 접근 방식은 사람의 행동이 사람이 포함되어 있는 시스템에 영향을 받는다고 가정한다. 이 접근법의 목표는 인적 오류(human error)를 줄이기 위해 작업 환경을 설계하는 방법을 찾는 것이다. 이러한 환경에서의 STPA 적용은 보다 기술적인 시스템에 STPA를 적용하는 방법과 동일하지만 공개된 문헌에 사례가 많지 않기 때문에 이 부분을 핸드북에 포함시켰다.

본 장에는 두 가지 예가 포함되어 있다. 하나는 고도로 자동화된 작업을 하는 사람들에 관한 것이고, 다른 하나는 LOTO(lockout/tagout, 역주: 허가되지 않은 작업자가 임의로 조작하는 것을 경고하는 태그와 함께 설치되는 잠금장치) 사용과 관련하여 비교적 덜 기술적인 작업장 위험에 관한 것이다. 두 경우 모두에서, 작업장 안전에 대한 STPA와 기존 접근법의 주요 차이점은 STPA는 단지 사람이 따르는 절차를 만드는 것에 초점을 맞추기보다 전체 시스템을 보며 위험한 시나리오를 식별하고 그것들을 완화 또는 제거하는 방법에 초점을 맞추고 있다는 것이다.

아래의 첫 번째 예제는 제조 환경에서의 STPA 실제 적용에 대한 내용이기 때문에 실제로 사용된 평가 절차와 비용에 대한 세부 정보가 포함되어 있다.

반자동 제조 프로세스에서의 STPA 적용 예제

이 예제는 Nathaniel Peper의 석사 학위 논문[17]에서 그대로 발췌한 것으로, 고도의 자동화(로보틱스)가 작업장에 도입된 실제 항공기 조립 프로세스에 STPA를 적용한 사례이다. 여기서 그는 다음과 같이 프로세스를 설명하고 있다.

"... 그 프로세스는 작업자가 운전하는 차량이 항공기 하위 컴포넌트를 탑재하고 공장에 들어올 때부터 시작된다. 공장 내에는 두 명의 작업자가 더 있는데, 각 작업자는 휴대용 제어 장치를 이용하여 AGV(Automated Ground Vehicle, 자동 지상차량)를 공장 내 AGV 도킹 및 충전 지점에서, 공장에 들어오는 하위 컴포넌트를 위한 PTV(Product Transportation Vehicle, 제품 운송 차량) 연결 지점까지 운전한다. PTV까지 이동한 후, 두 운전자는 공장 주변의 결합된 PTS(Product Transportation System, 제품 운송 시스템)를, 앞서 언급한 차량의 하위 컴포넌트 위치까지 협력하여 운전한다. 이 때, 다른 작업자 그룹이 함께 차량에 있는 하위 컴포넌트를 PTS로 옮긴다. 그런 다음 AGV 운영자는 PTS를 하위 컴포넌트가 자동 생산 셀에 투입되기를 기다리는 위치까지 공장을 가로질러 이동시킨다. 셀 내부에는 다양한 로봇 및 관련 지원 부품이 있는데, 하위 컴포넌트에 구멍을 뚫고 구멍에 다른 컴포넌트를 채우는 자동화된 프로세스를 수행한다. 하위 컴포넌트, PTS, 로보틱스 셀 및 운영자가 모두 준비되면 셀 운영자와 컨트롤러는 PTS의 제어를 가정하여 작업 지시서를 수행하는 자동화 프로세스에 따라 PTS를 이동한다. 이 과정동안 로보틱스 셀 경계 안쪽에 있지 않은 하위 컴포넌트에도 동시 작업이 발생한다. 이러한 전체 자동화 프로세스가 완료되면 작업자는 PTS의 제어를 다시 수행하고 공장 라인을 따라 다음번 수동 작업 처리를 위한 위치로 PTS를 옮길 것이다."

사고 또는 손실은 다음과 같다.

---

17 Nathaniel Peper, Systems Thinking Applied to Automation and Workplace Safety, MIT Masters Thesis (Leaders for Global Operations Program), June 2017.

A: Death, Injury, or Illness to Humans

General hazards in this workplace:

H1: Exposure to uncontrolled energy (or energy at a level that could lead to a loss)

H2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury

H3: Exposure to toxic materials above a safe level

H4: Exposure to noise levels that could affect hearing

H5: Extended exposure to an environment not providing basic human health requirements

The team refined this set of hazards to one related to the specific application:

H1: Exposure to uncontrolled energy (or energy at a level that could lead to a loss)

H1.1: Violation of minimum separation between AGVs and external objects

H1.2: Violation of minimum separation between PTV and external objects

H1.3: Violation of minimum separation between AGVs and PTV combination and external objects

H1.4: Violation of minimum separation between Robotics and external objects

Other energy sources present in the current design that must be controlled for H1 including electrical, thermal, pneumatic, hydraulic, gravitational, mechanical, and non-ionizing radiation (lasers).

H2 and **H3** can be refined to be:

H2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury

H2.1: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during routine operation

H2.2: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during maintenance, servicing, or troubleshooting

H2.3: Potentially injurious movement of the human body (or stress on the body) that could lead to injury during installation, repair, or overhaul

H3: Exposure to toxic materials above a safe level

H3.1: Workers are exposed to toxic chemicals above a safe level while operating the system

H3.2: Workers are exposed to toxic chemicals above a safe level while servicing the equipment

H3.3: Workers are exposed to toxic materials above a safe level from the manufacturing process

Changing these hazards into safety constraints is trivial and is not included here.

The eight-member cross-functional team that did the actual analysis was composed of members from integration, engineering, operations, and maintenance. All the people involved were considered to be leads or subject matter experts within their scope of responsibility for the system. The facilitator was an MIT master's student who was trained in STPA but had little familiarization with the new manufacturing process being introduced to the plant.

The analysis was completed over the course of a three-week period. A group setting for the assessment was only used twice during the process for less than three hours each, while the rest of the assessment was completed in one-on-one meetings between the facilitator and the respective expert for the portion of the system being discussed.

A: 사람의 사망, 상해 또는 질병

이 작업장에서의 일반적 위험:

H1: 제어되지 않은 에너지(또는 손실을 야기할 수 있는 수준의 에너지)에 노출됨

H2: 잠재적으로 인체에 상해를(또는 신체에 스트레스를) 야기할 수 있는 유해한 움직임

H3: 안전한 수준을 넘어선 독성 물질에 노출됨

H4: 청력에 영향을 줄 수 있는 수준의 소음에 노출됨

H5: 기본적인 사람의 건강 요구사항을 충족하지 않는 환경에 장시간 노출됨

팀에서는 위의 위험 집합을 특정 어플리케이션과 관련된 것으로 상세화하였다.

H1: 제어되지 않은 에너지(또는 손실을 야기할 수 있는 수준의 에너지)에 노출됨

H1.1: AGV와 외부 물체 간의 최소이격거리 위반

H1.2: PTV와 외부 물체 간의 최소이격거리 위반

H1.3: AGV, PTV 조합과 외부 물체 간의 최소이격거리 위반

H1.4: 로보틱스와 외부 물체 간의 최소이격거리 위반

H1을 위해 제어되어야 하는 전기, 열, 공압, 유압, 중력, 기계 및 비이온화 방사선(레이저)를 포함한 다른 에너지원이 현재 설계 내역에 기술되어 있다.

H2와 H3는 다음과 같이 상세화될 수 있다.

H2: 잠재적으로 인체에 상해(또는 신체에 스트레스)를 야기할 수 있는 유해한(injurious) 움직임

H2.1: 일상적인 작업 중 잠재적으로 인체에 상해(또는 신체에 스트레스)를 야기할 수 있는 유해한 움직임

H2.2: 유지보수, 서비스 또는 문제 해결 중 잠재적으로 인체에 상해(또는 신체에 스트레스)를 야기할 수 있는 유해한 움직임

H2.3: 설치, 수리 또는 분해점검(overhaul) 중 잠재적으로 인체에 상해(또는 신체에 스트레스)를 야기할 수 있는 유해한 움직임

H3: 안전한 수준을 넘어선 독성 물질에 노출됨

H3.1: 작업자가 시스템을 운영하는 동안 안전한 수준을 넘어선 독성 물질에 노출됨

H3.2: 작업자가 장비를 정비하는 동안 안전한 수준을 넘어선 독성 물질에 노출됨

H3.3: 작업자가 제조 프로세스에서 안전한 수준을 넘어선 독성 물질에 노출됨

이러한 위험을 안전 제약사항으로 변경하는 것은 어렵지 않으므로 여기에 명시하지 않았다.

실제 분석을 수행한 8명의 복합기능팀(cross-functional team, 역주: 프로젝트를 중심으로 서로 다른 기능 전문가로 별도 팀을 구성한 것으로 TFT와 유사)은 통합, 엔지니어링, 운영, 그리고 유지보수로 구성되었다. 관련된 모든 사람들은 시스템에 대해 그들의 책임 범위 내에서 리더 또는 전문가로 볼 수 있다. 퍼실리테이터인 MIT 석사과정 학생은 STPA 교육은 받았지만 공장에 도입되는 새로운 제조 프로세스에는 익숙하지 않았다.

분석은 3주간의 기간을 거쳐 완성되었다. 전체 평가 회의는 2회, 각 3시간 이내로 진행되었고, 나머지

Because STAMP and STPA are relatively new for the company and entirely new for the team, the first group meeting was used as an instruction period to provide an overview of the methodology being used and then to define the accidents (or losses) the team wanted to avoid, the hazards and system boundaries, and an initial draft of the safety control structure for this application. After developing the initial draft of the safety control structure, follow-on meetings were scheduled with each respective expert to add detail to specific portions of the control structure. Because STPA uses a model of the control structure, the assessment could be focused on specific control loops within the model after the complete model was constructed. This ability to divide up the modeling and analysis process provided for easy scheduling of meeting times and numerous meetings within the available time frames of everyone within the group. During these meetings, details such as control actions, process models, feedback, etc. were added and any conflicts or issues were resolved with the affected parties.

Once the appropriate level of detail was developed for the system safety control structure, the one-on-one or small group discussions continued with STPA Steps 1 and 2. The results of this portion of the analysis were then discussed in another group setting to confirm the individual findings of the group and discuss potential mitigations for the system. This process occurred over the course of three weeks, and the time spent in group, one-on-one meetings, and individual facilitator work totaled approximately 300 hours.

After the hazards were identified, control structures were developed for four different functions and possible configurations of the system, and the RAAs (Responsibilities, Accountability, and Authority) of each controller. The RAAs in the STAMP and STPA context are defined as:

Responsibilities: Basic control responsibilities, process model, and process model variables

Authority: Possible control actions controller could provide, how they are sent, when they are sent, and where the control action is sent.

Accountability: What feedback is provided, when it is provided, where the feedback goes in the control structure, how it is given.

Because this team was new to STPA, they began by developing the control structure as a group to ensure that all of the major details were captured. The models were then refined by individuals and small groups with the appropriate technical expertise on different parts of the system.

The four scenario-based models are shown in Figures 4.1, 4.2, 4.3, and 4.4.

평가는 퍼실리테이터와 시스템 부분(part)의 전문가 간 1:1 미팅을 통해 완료되었다.

STAMP와 STPA는 회사에게는 상대적으로, 팀에게는 완전히 생소한 것이었기 때문에 첫 그룹 미팅은 이론을 가르쳐주는 시간(instruction period)으로 사용되었는데, 방법론의 개요를 소개한 후 팀에서 피하고자 하는 사고(손실), 위험과 시스템 경계, 그리고 이번 적용에 대한 컨트롤 스트럭처의 초안을 정의하였다. 컨트롤 스트럭처 초안 작성 후, 컨트롤 스트럭처의 특정 부분에 대해 세부 사항을 추가하기 위해 각 전문가가 참석한 후속 회의가 계획되었다. STPA는 컨트롤 스트럭처의 모델을 사용하기 때문에 평가(assessment)는 전체 모델을 만든 이후에 모델의 특정 컨트롤 루프에 초점을 맞추어 수행할 수 있다. 모델링과 분석 프로세스를 분리할 수 있는 이러한 STPA의 특징으로 인해, 그룹 내 모든 사람들의 가능한 시간 범위 내에서 수많은 회의와 회의시간을 쉽게 스케줄링 할 수 있었다. 이러한 회의를 수행하는 과정에서 컨트롤 액션, 프로세스 모델, 피드백 등과 같은 세부 사항이 추가되었으며 충돌(conflict) 또는 이슈는 관련된 당사자 간 해결하였다.

시스템 안전 컨트롤 스트럭처에 대한 세부 사항이 적절한 수준으로 만들어지면, STPA 1단계 및 2단계에 대한 1:1 또는 소그룹 토론이 계속 진행되었다. 그런 다음 분석의 이 부분적인 결과는 또 다른 그룹이 논의하는데, 여기서는 그룹이 발견한 개별적 사항들을 확인하고 잠재적인 시스템 완화조치(mitigation)를 논의하였다. 이 과정은 3주에 걸쳐 이루어졌으며, 그룹, 1:1미팅 및 각 퍼실리테이터 작업에 소요된 시간은 약 300시간이었다.

위험이 식별된 후에는 4가지 다른 기능(function), 가능한 시스템 구성, 각 컨트롤러의 RAA(책임 Responsibilities, 의무 Accountability, 통제권한 Authority)에 대한 컨트롤 스트럭처가 개발되었다. STAMP와 STPA 컨텍스트에서 RAA는 다음과 같이 정의된다.

책임(Responsibility) : 기본적인 컨트롤 책임, 프로세스 모델 및 프로세스 모델 변수

통제권한(Authority) : 컨트롤러가 제공할 수 있는 컨트롤 액션이 무엇이고, 어떻게 전송되는지, 언제 전송되는지, 어디로 전달되는지에 대한 내용

의무(Accountability) : 어떤 피드백이 제공되는지, 언제 제공되는지, 피드백이 컨트롤 스트럭처의 어디로 가는지, 어떤 방식으로 전달되는지에 대한 내용

이 팀은 STPA를 처음 접했기 때문에 모든 주요한 세부 사항이 포함되었음을 확실히 할 수 있도록 하나의 그룹으로 컨트롤 스트럭처를 개발하기 시작했다. 그런 다음, 개인 또는 소그룹(시스템의 서로 다른 부분에 대한 적절한 기술 전문가로 구성됨)에서 그 모델을 상세화하였다.

4가지 시나리오 기반 모델을 그림 4.1, 4.2, 4.3 및 4.4에 보였다.

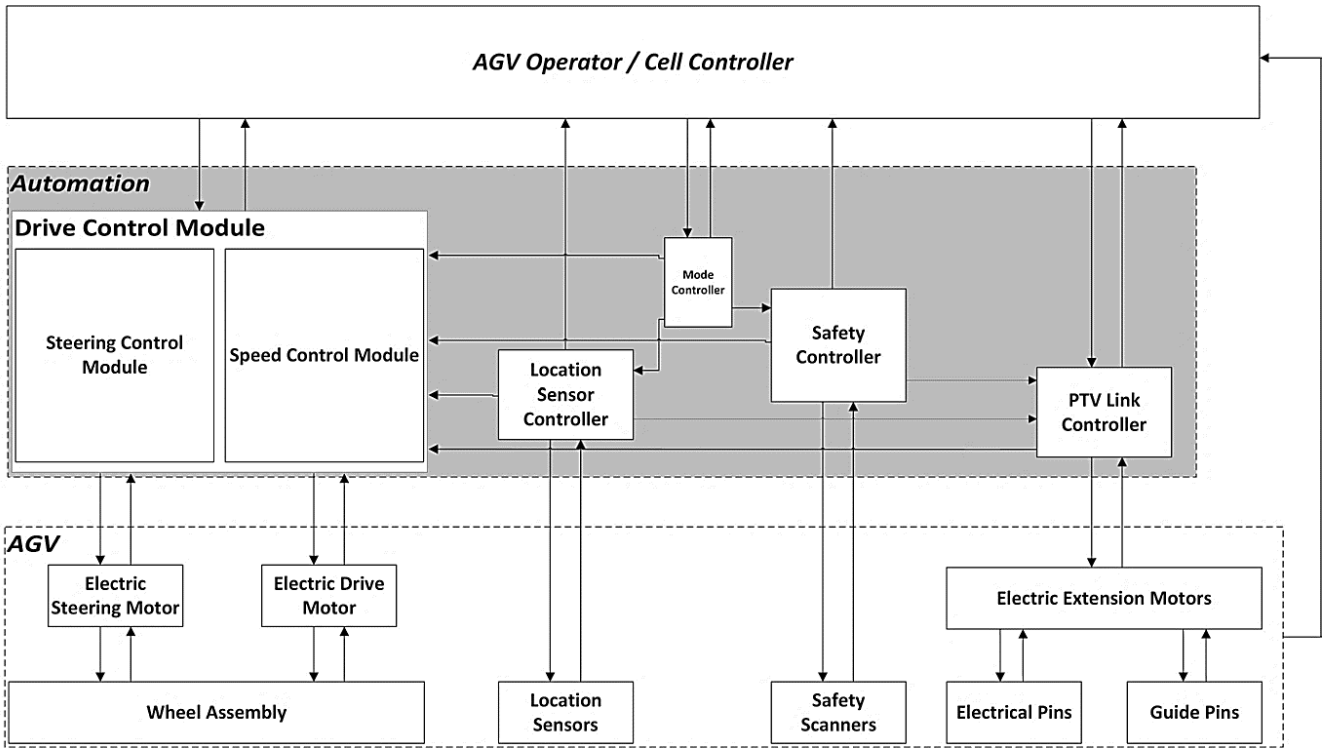


Figure 4.1: Automated Ground Vehicle (AGV) Safety Control Structure

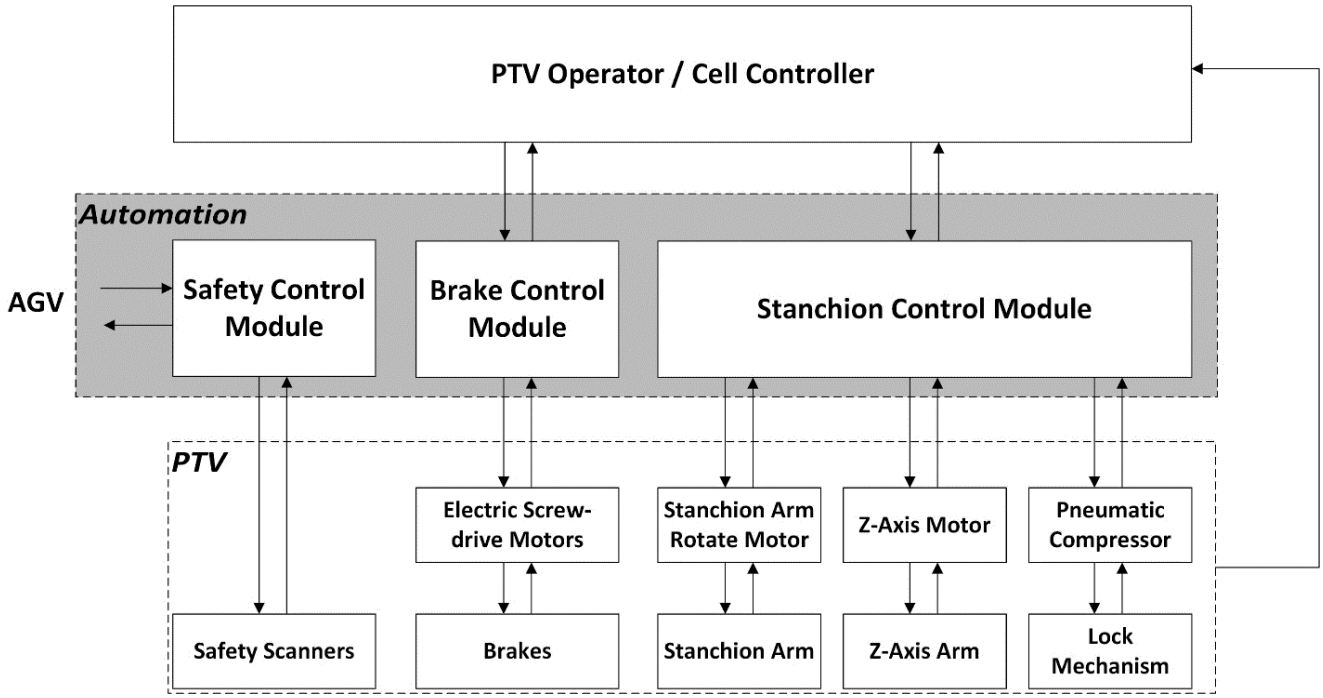


Figure 4.2: Product Transportation Vehicle (PTV) Control Structure

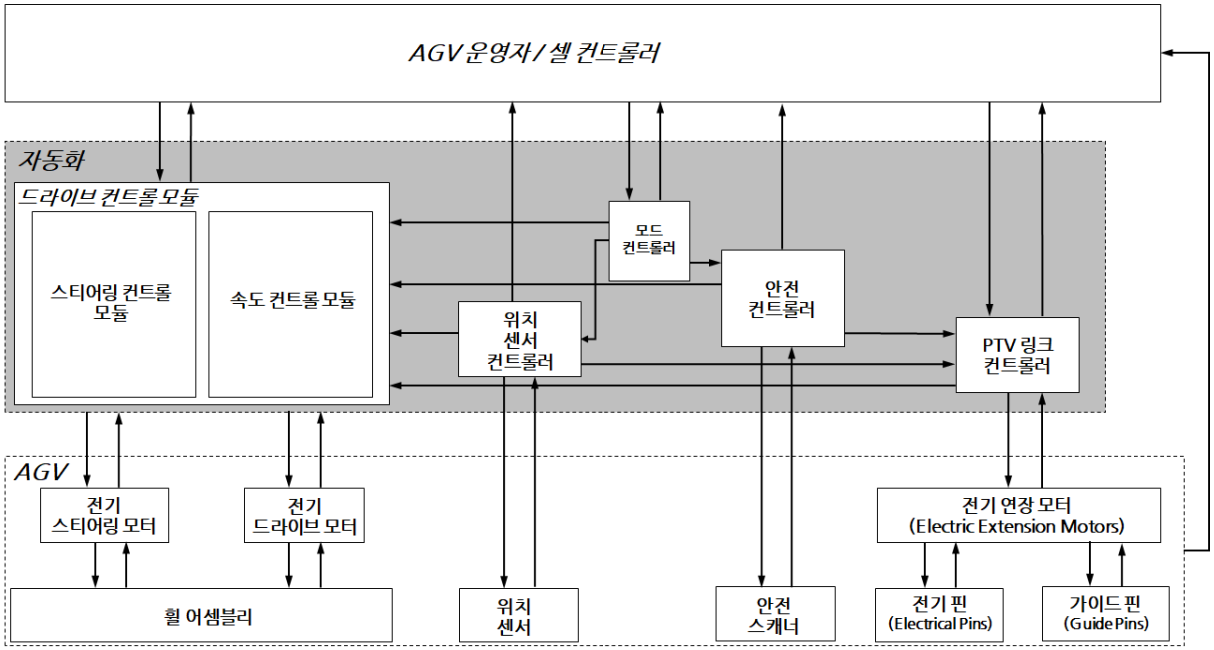


그림 4.1: AGV(Automated Ground Vehicle, 자동 지상차량) 안전 컨트롤 스트럭처

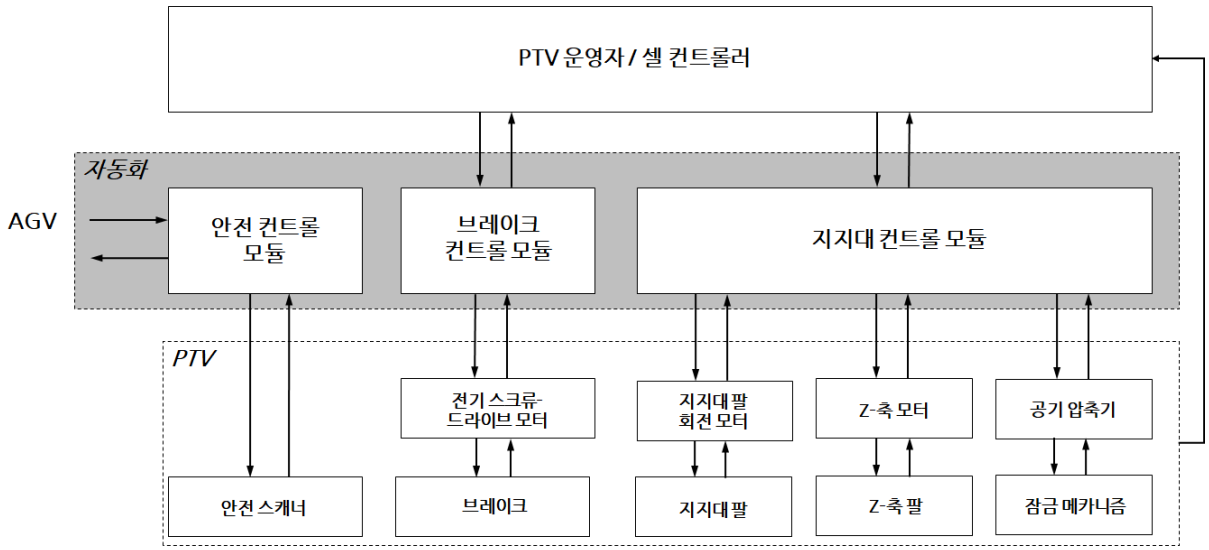


그림 4.2: PTV(Product Transportation Vehicle, 제품 운송 차량) 컨트롤 스트럭처

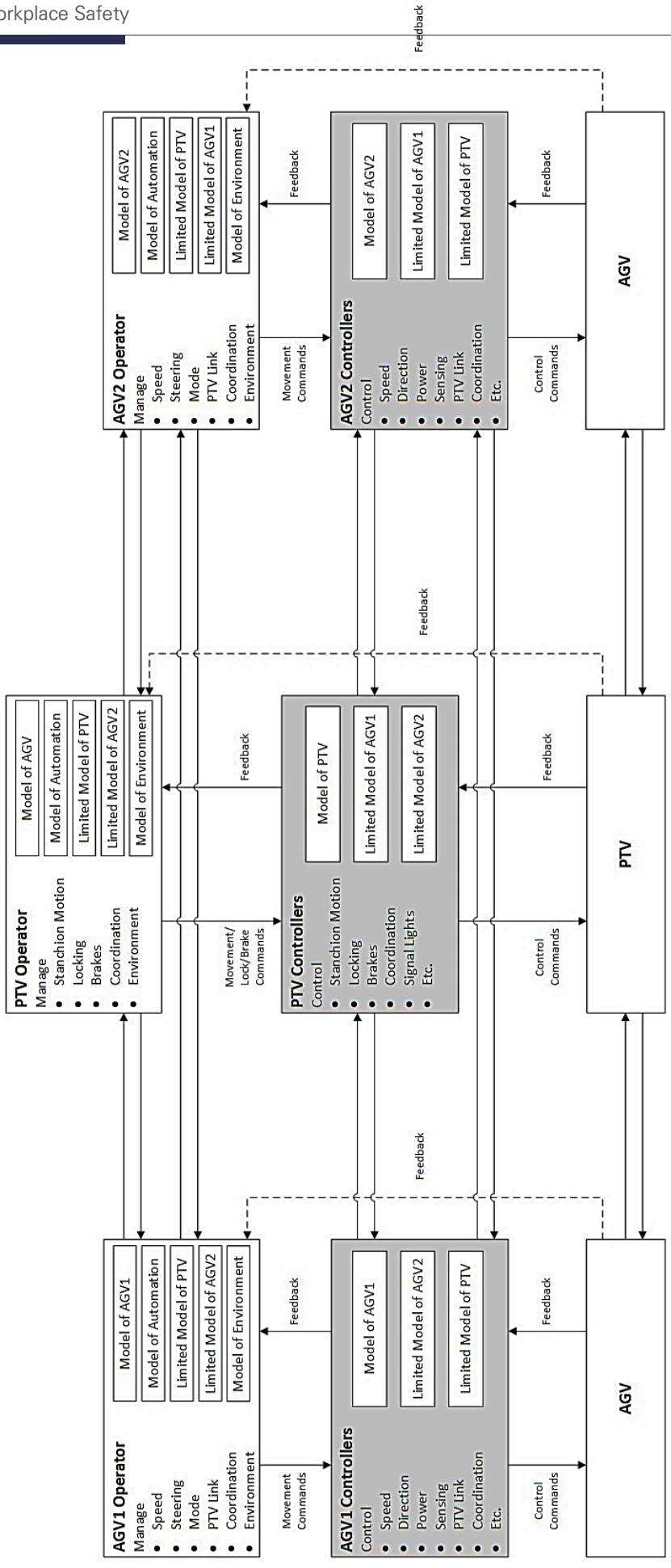


Figure 4.3: Combined Product Transportation System Safety Control Structure

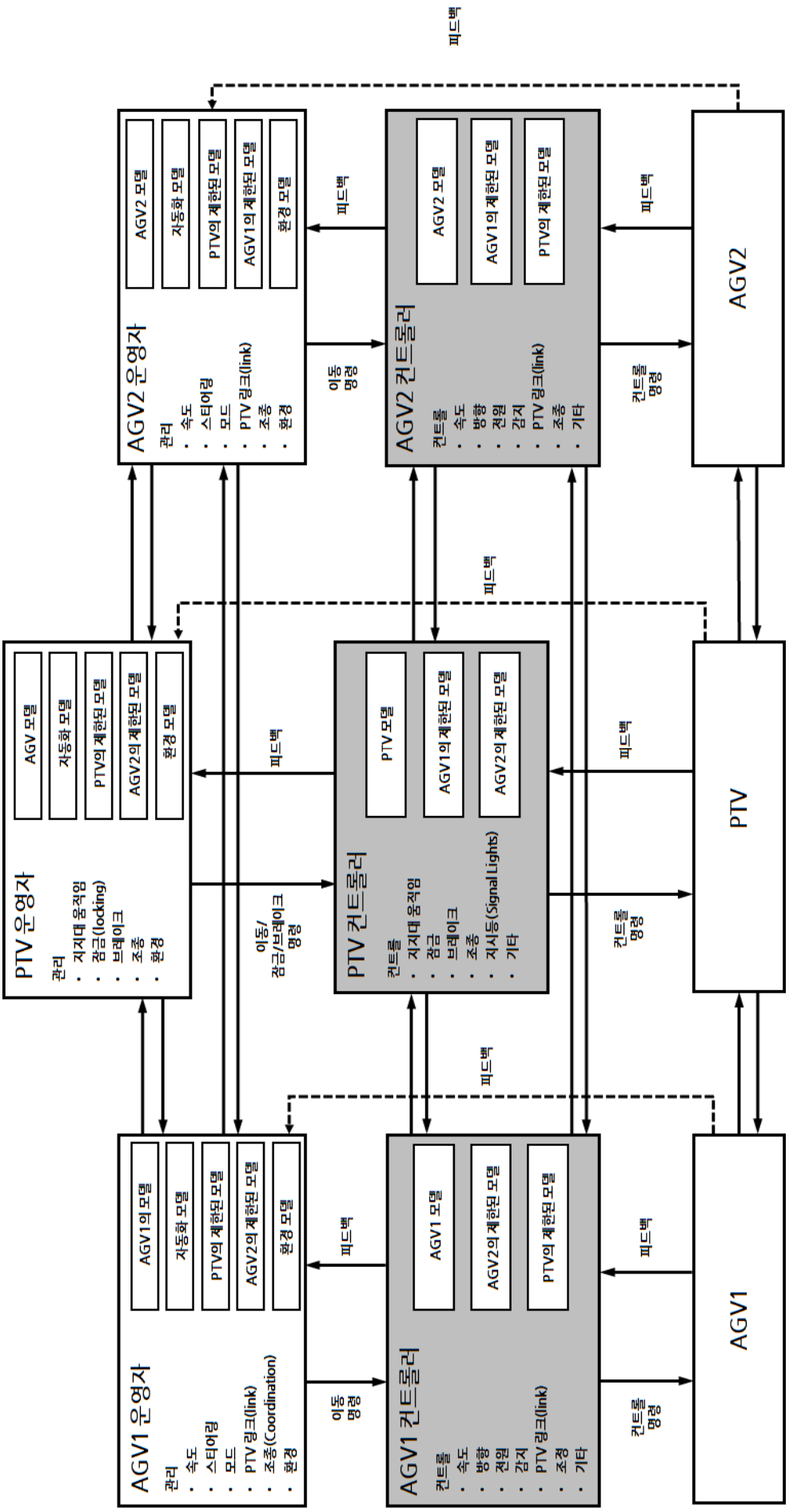


그림 4.3: 결합된 PTS(Product Transportation System, 제품 운송 시스템) 안전 컨트롤 스트럭처

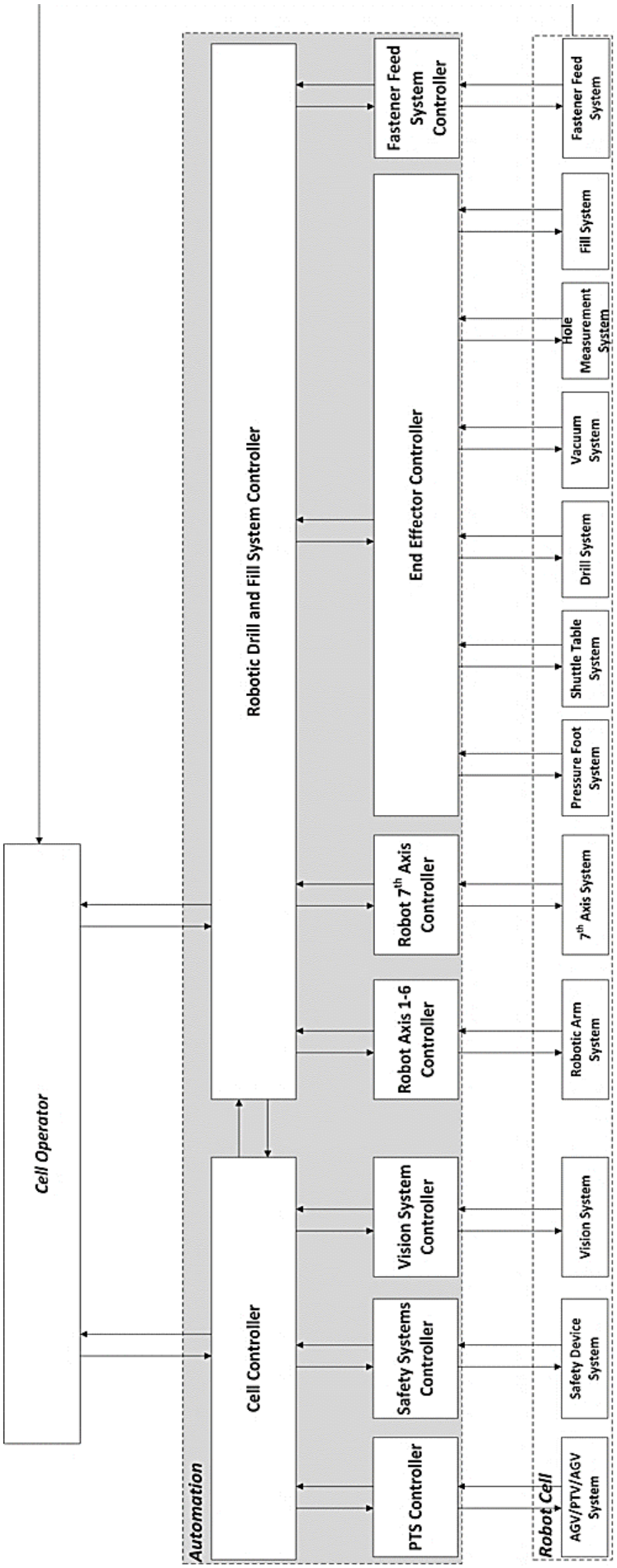


Figure 4.4: Robotic System Safety Control Structure

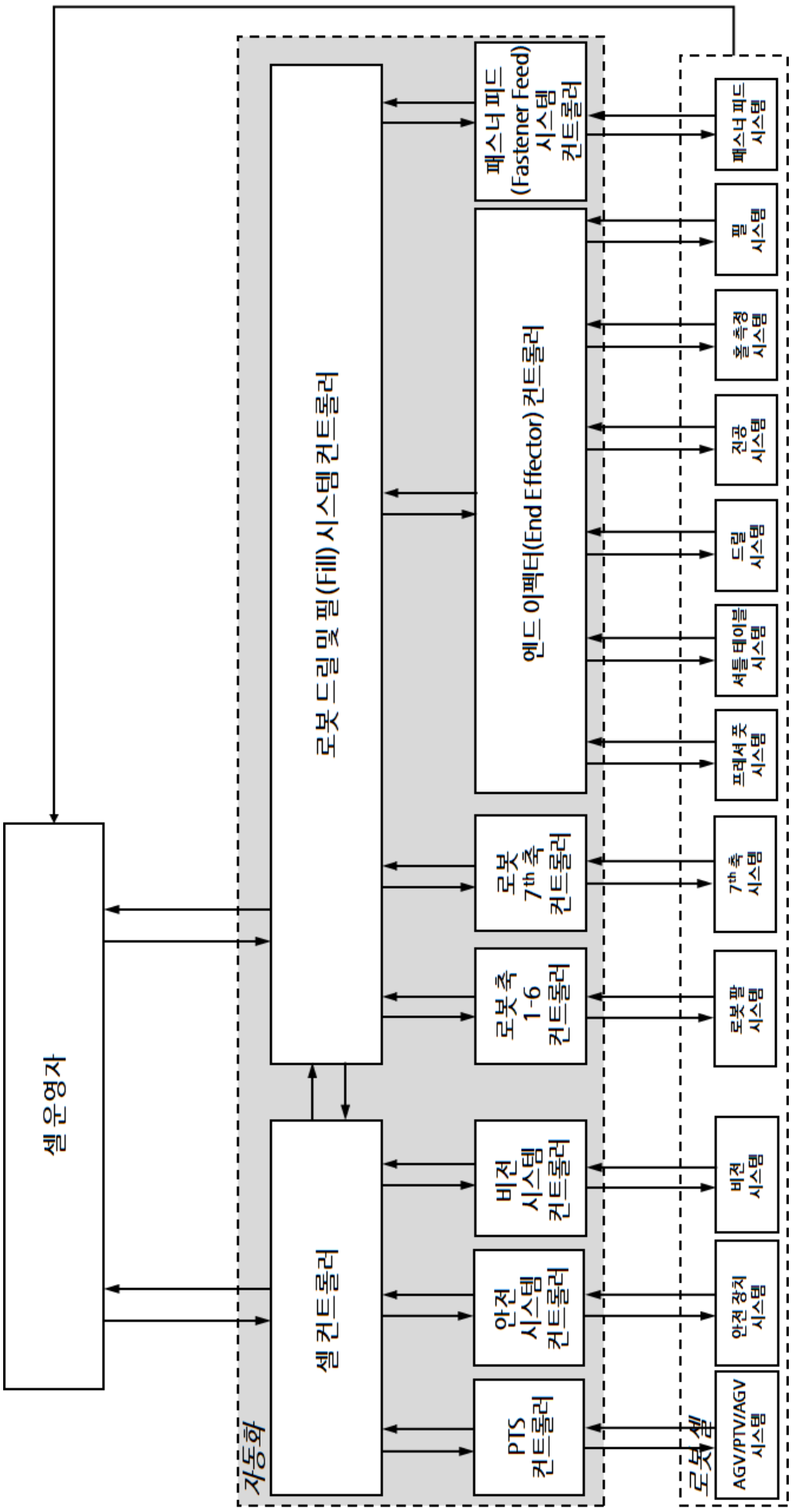


그림 4.4: 로보틱스 시스템 안전 컨트롤 스트럭처

Only a sample of the rest of the STPA analysis is shown here. The more detailed segment of the AFV safety control structure used in the example is shown in Figure 4.5.

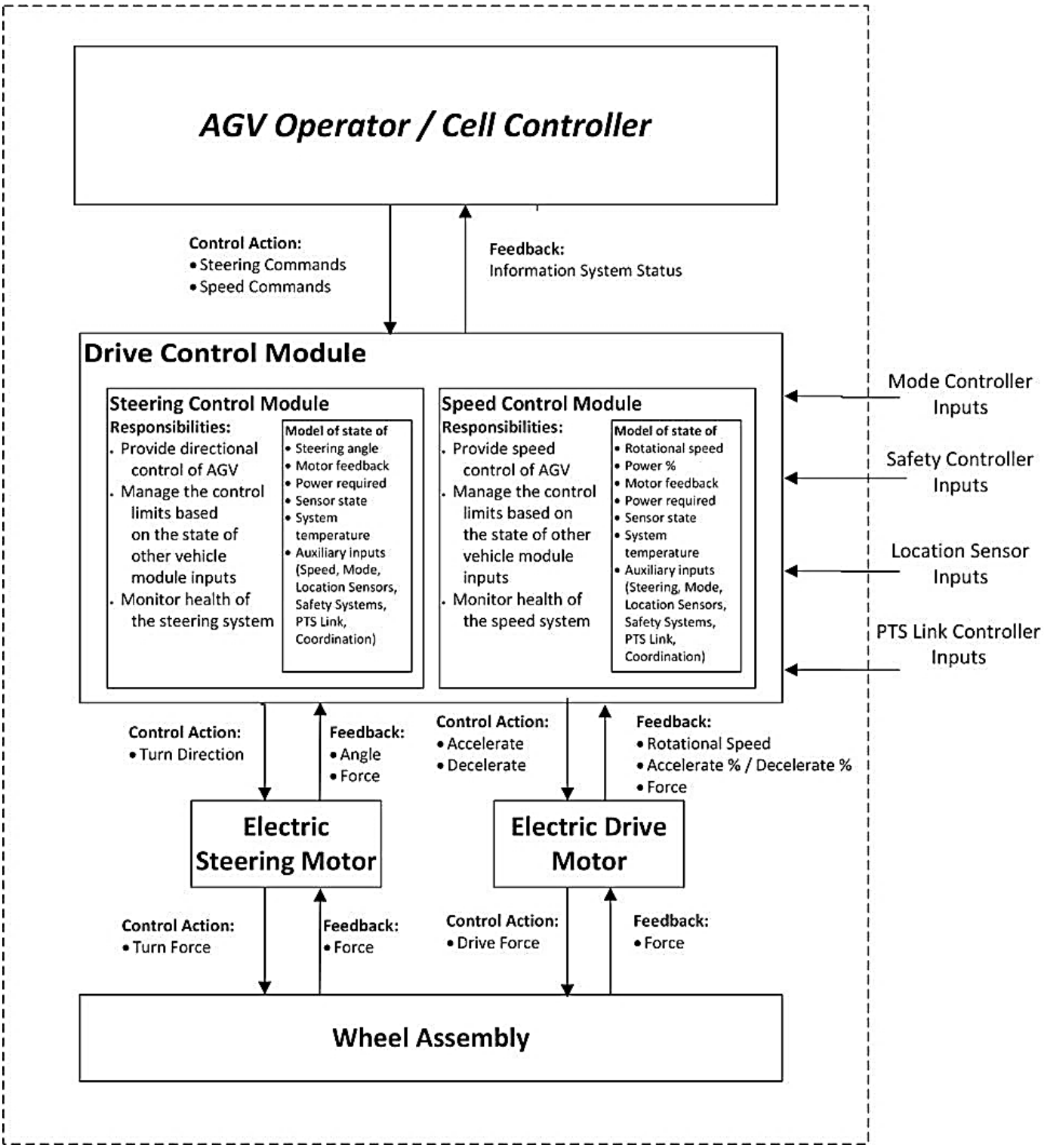


Figure 4.5: AGV Safety Control Structure Segment

여기에는 나머지 STPA 분석 결과의 일부만 명시하였다. 이 예에서 사용된 AGV 안전 컨트롤 스트럭처의 자세한 부분은 그림 4.5에서 볼 수 있다.

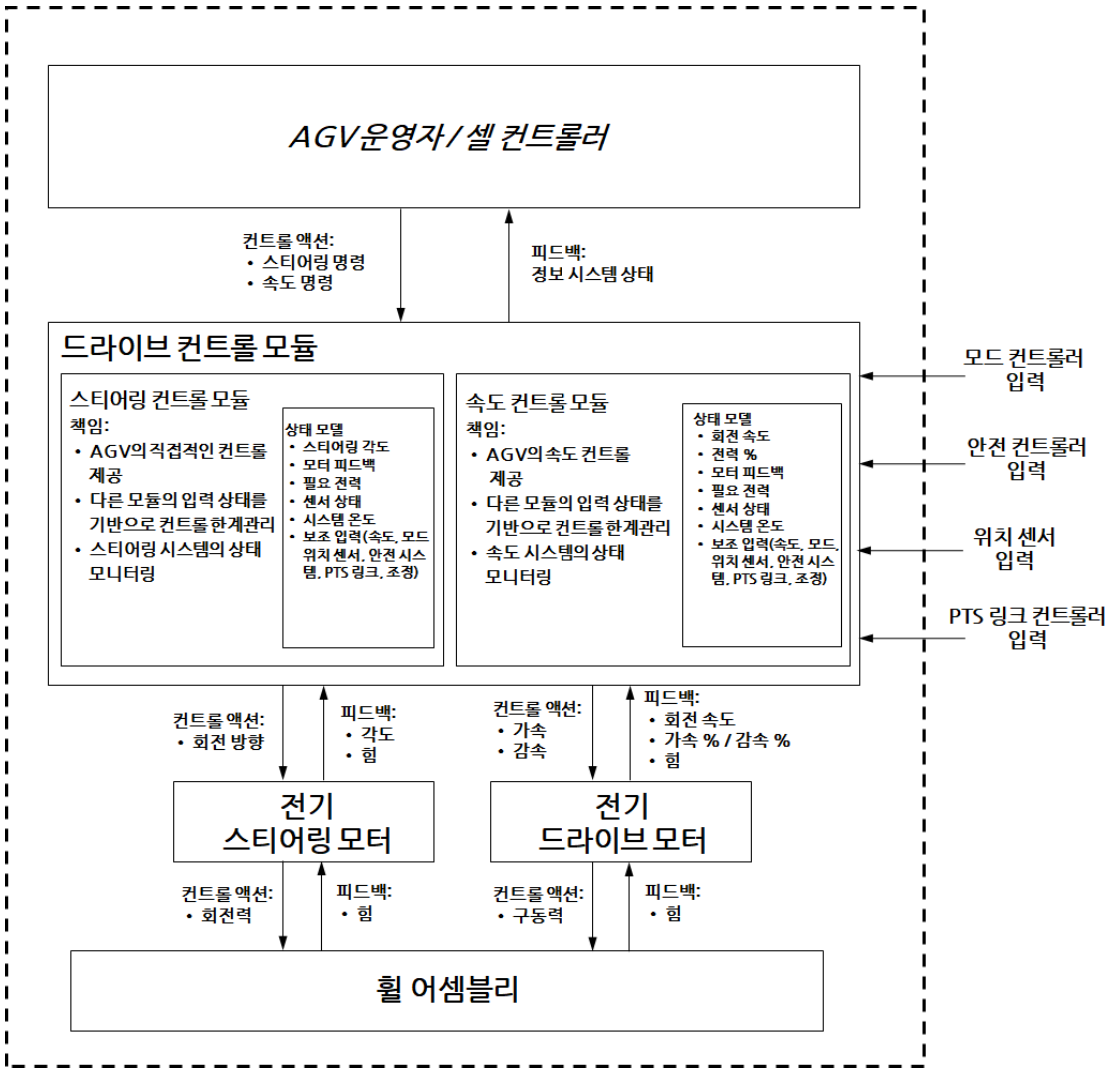


그림 4.5: AGV 안전 컨트롤 스트럭처 세그먼트

An example from the UCA table for the AGV follows:

Control Action	Providing Causes Hazard	Not Providing Causes Hazard	Incorrect Timing/Order	Stopped Too Soon/ Applied Too Long
Operator provides drive commands to drive control module	**UCA1:** Drive control module commanded to drive when the movement will violate minimum separation with an object [H1.1]	**UCA3:** Drive control module not commanded to drive when the movement will prevent a violation of minimum separation with an object [H1.1]	**UCA4:** Drive control module commanded to drive before or after a safe path direction [H1.1]	**UCA7:** Drive control module commanded to drive too long when the movement violates minimum separation with an object [H1.1]
	UCA2: Drive control module commanded to drive when a human is handling components that will move [H1]		**UCA5:** Drive control module commanded to drive before a human stops handling components that will move [H1]	
			UCA6: Drive control module commanded to drive after a human starts handling components that will move [H1]	

Causal scenarios for UCA1 (Drive control module commanded to drive when the movement will violate minimum separation with an object [H1]) include the following examples:

Causal Scenario 1: The AGV is driven inappropriately because the operator is not familiar with its operation.

- New operator that is not or inadequately trained

Causal Scenario 2: The AGV is driven toward an external object because the operator does not see the object or misjudges the safe path.

- Operator inattention due to task overload, changes to the environment, or other external factors.
- Operating in cluttered/restrictive areas

AGV에 대한 UCA 테이블에서의 예제는 아래와 같다:

컨트롤 액션	**제공함이 위험을 유발**	**제공하지 않음이 위험을 유발**	**잘못된 타이밍/순서**	**너무 빨리 중지/ 너무 오래 적용**
운영자가 드라이브 컨트롤 모듈에 드라이브 명령을 제공함	UCA1: 자동차의 움직임이 물체와의 최소이격거리를 위반할 예정일 때, 드라이브 컨트롤 모듈이 드라이브 명령을 제공함[H1.1]	UCA3: 자동차의 움직임이 물체와의 최소이격거리를 위반하지 않을 예정일 때, 드라이브 컨트롤 모듈이 드라이브 명령을 제공하지 않음[H1.1]	UCA4: 안전한 경로 방향 전(before) 또는 후에(after) 드라이브 컨트롤 모듈이 드라이브 명령을 제공함[H1.1]	UCA7: 자동차의 움직임이 물체와의 최소이격거리를 위반할 때, 드라이브 컨트롤 모듈이 드라이브 명령을 너무 길게 제공함[H1.1]
	UCA2: 사람이 움직이는 컴포넌트를 핸들링 할 때, 드라이브 컨트롤 모듈이 드라이브 명령을 제공함[H1]		UCA5: 사람이 움직이는 컴포넌트의 핸들링을 멈추기 전에 드라이브 컨트롤 모듈이 드라이브 명령을 제공함[H1]	
			UCA6: 사람이 움직이는 컴포넌트의 핸들링을 시작한 이후에 드라이브 컨트롤 모듈이 드라이브 명령을 제공함[H1]	

UCA1(자동차의 움직임이 물체와의 최소이격거리를 위반할 예정일 때, 드라이브 컨트롤 모듈이 드라이브 명령을 제공함)에 대한 원인 시나리오는 다음과 같다.

원인 시나리오1: 운전자가 운전에 익숙하지 않아서, AGV를 부적절하게 주행함

- 훈련을 받지 않거나 부적절하게 훈련받은 신규 운전자

원인 시나리오 2: 운전자가 물체를 보지 못하거나 안전한 경로를 잘못 판단하여, AGV가 외부 물체 방향으로 주행함

- 작업 과부하, 환경 변화 또는 기타 외부 요소로 인한 작업자의 부주의(inattention)
- 어수선하거나(cluttered) 제한적인(restrictive) 구역에서의 운영
- 다른 작업자, 차량 자체 또는 AGV/PTV/AGV컴비네이션 위에 실린 스파(spar, 역주: 항공기의 가로날개 뼈대)에 의해 시야가 가려져서 물체가 보이지 않음

원인 시나리오 3: AGV 속도/스티어링 설정이 변경되어 AGV가 외부 물체로 주행함

- 운전자가 모르고 있거나 익숙하지 않은 컨트롤러의 수정사항에 대해 기존의 이해도를 바탕

- Objects are blocked from view by other workers, the vehicle itself, or the spar load on the AGV/PTV/AGV combination

Causal Scenario 3: The AGV is driven into an external object because the AGV speed/steering settings have changed.

- Modifications to the controller that the operator does not know about or is not familiar with and operating under their previous understanding.
- Degradation to the controller that the operator does not know about or is not familiar with and operating under their previous understanding.
- Modifications to the AGV system that the operator does not know about or is not familiar with and operating under their previous understanding.
-

Causal Scenario 4: The AGV is driven into an external object because the controller has a hardware failure.

- Input is frozen and continues to be communicated even though the operator changes the command.

Causal Scenario 5: The AGV is driven into an external object because the operator holds the drive command for too long due to a delay in the command starting movement in the vehicle.

- Processing delays and computing overloads
- No timeout of control inputs to AGV

Causal Scenario 6: The AGV is driven into an obstacle because the operator is provided incorrect or no information on critical system statuses relied upon for navigation, such as safety scanners and location sensors.

- Operator misses any feedback because it is not easily accessible, small HMI on side of the AGV, next to the ground.
- No status of scanners being on or off
- No status of location sensors being on or off

Causal Scenario 7: The operator drives the AGV into an obstacle that is not detected by the scanners. Possible causes include:

- The object is outside of the safety scanners field of view.
- Obstacles in the factory at the PTV or spar level.
- PTV guide rails above the AGV
- AGV being used for unintended use, such as carrying objects that extend past the scanner FOV.
- The object is in the safety scanners field of view but below the detection threshold
- The object enters the field of view and is impacted by the AGV at a rate faster than the scan and reaction rate of the vehicle
- The scanner capabilities have degraded over time.
- The scanner fails into an unsafe state.

Causal Scenario 8: The operator drives the AGV into an obstacle that is detected by the scanners but does not stop the AGV. Possible causes include:

- Software coding errors do not command the vehicle to stop as intended
- There is a delay in the signal processing due to CPU overload

으로 운영함
- 운전자가 모르거나 익숙하지 않은 컨트롤러의 성능저하에 대해 기존의 이해도를 바탕으로 운영함
- 운전자가 모르거나 익숙하지 않은 AGV 시스템의 수정사항에 대해 기존의 이해도를 바탕으로 운영함

원인 시나리오 4: AGV 컨트롤러의 하드웨어 오류로 인해 AGV가 외부 물체로 주행함

- 운전자가 명령을 변경하더라도 입력이 고정되어 계속 전달됨

원인 시나리오 5: 차량 이동을 시작하는 명령의 지연으로 인해 운전자가 드라이브 명령을 너무 오랫동안 유지하여 AGV가 외부 물체로 주행함

- 프로세싱 지연 및 컴퓨팅 과부하
- AGV 제어 입력 시간의 제한이 없음

원인 시나리오 6: 안전 스캐너 및 위치 센서와 같은, 내비게이션에 필요한 중요한 시스템의 상태 정보가 운전자에게 제공되지 않거나 정확하지 않게 제공되어, AGV가 장애물로 주행함

- AGV 측면의 지면 옆에, 쉽게 접근할 수 없는 소형 HMI(Human Machine Interface)가 있어 운전자가 피드백을 놓침
- 스캐너의 켜짐 또는 꺼짐 상태 정보가 없음
- 위치 센서의 켜짐 또는 꺼짐 상태 정보가 없음

원인 시나리오 7: 운전자가 스캐너가 검출하지 못한 장애물로 AGV를 운전함. 가능한 원인들은 다음과 같음

- 물체가 안전 스캐너 시야 밖에 위치함
- PTV 또는 스파 높이의 공장 내부에 있는 장애물
- AGV보다 위에 있는 PTV 가이드 레일
- 스캐너 시야를 벗어나 물체를 옮기는 등 AGV가 의도하지 않은 용도로 사용됨
- 물체가 안전 스캐너 시야에 있지만 검출 임계값 이하에 있음
- 물체가 차량 스캔 및 반응 속도보다 빠른 속도로 시야에 들어와 AGV에 충돌함
- 스캐너 성능이 시간이 지남에 따라 저하됨
- 스캐너가 안전하지 않은 상태에 빠짐

원인 시나리오 8: 운전자가 스캐너가 검출한 물체로 AGV를 주행하지만 AGV를 멈추지 않음

- 소프트웨어 코딩 오류로 인해 소프트웨어가 의도한대로 차량 정지 명령을 내리지 않음
- CPU 과부하로 인해 신호 처리가 지연됨
- PTV 스캐너와의 인터페이스가 부적절하여 PTV 스캐너가 AGV 안전 제어 모듈에 신호를 보낼 수 없음

원인 시나리오 9: 스캐너가 의도적으로 꺼져 있을 때 운전자가 외부 물체로 AGV를 운전함.

- 스캐너가 꺼져 있지만(disabled) 운전자가 그것을 모르고 있어 AGV를 정지시키는 것을 스캐너에만 의존함
- PTV 스캐너가 AGV 운전자에게 피드백을 주지 않고 꺼져 있음(disable)

- The interface with the PTV scanners is inadequate and the PTV scanners cannot send the signal to the AGV Safety Control Modules

Causal Scenario 9: The AGV operator drives the AGV into an external object when the scanners are deliberately turned off.

- The scanners were disabled and the operator did not know it, relying on the scanners to stop the AGV.
- The scanners on the PTV were disabled with no feedback to the AGV operator
- The operator did not see the object.
- Operator is relying on the location sensing module to avoid possible obstacles
- The operator believes the vehicle is in a different mode and expects the vehicle to respond differently
- The AGV has been modified and the operator does not know or understand the updates.
- The commands are delayed and the operator provides the command for too long

In general, the scenarios included such causal factors as hardware failures, incorrect process models (stemming from missing or incorrect feedback), system component interactions, and the role of management and procedures combined with schedule pressures in the factory. After the team completed STPA Steps 1 and 2, they used the results to develop new controls for the system in order to eliminate or prevent entering hazardous states. A few examples follow.

One example comes from the system's heavy reliance on safety scanners to prevent the AGV and product transportation vehicle from colliding with surrounding objects, including workers. If an object is in the path of motion of the system and is not detected by the operator or scanners, there are no other controls designed into the system to prevent the hazard or mitigate the effects of an accident. Due to the massive size, length, and weight of the system and the force of the AGV drive motors, any unsafe control action could result in a severe accident under certain conditions. While operator and workers in the area staying alert to AGV movement around them and AGV safety scanners were the current design control, there are a number of causal scenarios that show them as insufficient.

First, as had already been demonstrated in other parts of the company, workers may bypass safety scanners if they malfunction or are not calibrated or designed properly, preventing the workers from completing their tasks and falling behind on the production schedule. The safety scanners are not integrated in a way that prevents the AGV from operating when the scanners are not in use and there is no notification to the operator or surrounding workers that the AGV is operating with a disabled safety system. While it may not be an issue for the user that turns the safety system off in order to get a job done, if they do not turn the safety system back on the next user will be operating the AGV without knowing that the scanners are disabled.

Another scenario is if an AGV were to encounter an obstacle that the scanners do not protect against. The drive motors for the vehicle do not provide any feedback to the drive controller about actual force being encountered. This would result in the AGV continuing to apply full force into the obstacle until the AGV drive controller receives a different drive command from the operator. An example of this could be something that is outside of the scan volume of the scanners, such as an obstacle above the vehicle but still in the path of the product on top of the transportation system, a scanner adjusted with a resulting gap in coverage around the vehicle, scanner control logic about clearing faults or obstacles when an obstacle is still in the scan volume, multiple controllers managing multiple scan systems with the integrated product transportation system, etc.

- 운전자가 물체를 보지 못함
- 운전자가 장애물을 피하는 것을 위치센서 모듈에 의존함
- 운전자가 차량이 다른 모드에 있어 차량이 다르게 반응할 것이라고 기대함
- AGV가 수정되었으나 운전자가 알지 못하거나 해당 업데이트를 이해하지 못함
- 명령이 지연되어 운전자가 명령을 너무 오랫동안 제공함

일반적으로, 시나리오에는 하드웨어 고장, 잘못된 프로세스 모델(누락되거나 잘못된 피드백으로 인한), 시스템 컴포넌트 상호작용, 공장에서의 일정 압박으로 인한 관리역할 및 절차 등의 원인 요소(causal factor)가 포함된다. 팀은 STPA 1단계 및 2단계를 완료한 후, 그 결과를 사용하여 위험한 상태로 빠지는 것을 예방하거나 제거할 수 있도록 시스템에 새로운 컨트롤을 개발하였다. 다음은 그에 대한 몇 가지 예이다.

한 가지 예로, AGV 및 PTV(제품운송차량)가 주변 물체나 작업자와 충돌하는 것을 방지하기 위해 시스템이 안전 스캐너에 과도하게 의존하는 것을 들 수 있다. 물체가 시스템 동작 경로상에 있고 운전자나 스캐너에 의해 검출되지 않았을 때 시스템에는 위험을 예방하거나 사고 영향을 줄이기 위해 설계된 다른 컨트롤이 없다. 시스템의 방대한 크기, 길이, 무게, AGV 구동 모터의 힘으로 인해 어떤 UCA라도 특정 조건에서 심각한 사고를 일으킬 수 있다. 구역내의 운전자와 작업자가 그들 주변의 AGV 움직임에 주의를 기울이고 있고 AGV 안전 스캐너가 현재 설계되어 있지만 이것으로는 충분치 않음을 보여주는 수많은 원인 시나리오가 있다.

첫째로, 회사의 다른 파트에서 이미 증명된 바와 같이, 안전 스캐너가 오작동하거나 교정되지 않았거나 적절하게 설계되지 않은 경우, 작업자는 작업을 완료하지 못하거나 생산 일정에 맞추지 못하는 상황을 방지하기 위해 안전 스캐너를 무시할 수 있다. 안전 스캐너는 스캐너가 사용 중이 아닐 때 AGV가 운영되지 못하게 하는 방식으로 통합되지(integrated) 않았고, 운전자나 주변 작업자에게 AGV가 안전 시스템이 동작하지 않는 상태에서 운영되고 있음을 알려주지 않는다. 작업을 수행하기 위한 사용자가 안전 시스템을 끄는 것은 문제가 되지 않지만, 만일 안전 시스템을 다시 켜놓지 않으면 다음 사용자는 스캐너가 동작하지 않는다는 것을 모른 채로 AGV를 운영할 것이다.

다른 시나리오는 AGV가 스캐너도 검출하지 못하는 장애물과 마주치는 경우이다. 차량용 구동 모터는 실제로 발생하는 힘에 대하여는 드라이브 컨트롤러에게 어떠한 피드백도 제공하지 않는다. 이것 때문에 AGV가 장애물과 부딪힌 후에도, AGV 드라이브 컨트롤러가 운전자로부터 다른 드라이브 명령을 수신할 때까지 계속 최대 동력을 적용하는 것이다. 이러한 예는 스캐너의 스캔 범위 밖의 무언가가 있는 경우일 수 있다. 예를 들면, 차량보다 높은 위치에 있지만 여전히 운송시스템의 제품 경로 내에 있는 장애물, 차량 주변 커버리지에 대한 갭을 조정한 스캐너, 스캔 범위 내에 장애물이 있지만 문제점(faults) 또는 장애물을 지워버리는 스캐너의 컨트롤 로직, 통합된 제품 운송 시스템에서 다수의 스캔 시스템을 관리하는 다수의 컨트롤러 등이다.

AGV는 크고 힘이 센 이동식 시스템으로 단순히 펜스를 치는 것으로는 AGV로부터 작업자를 안전하게 지킬 수 없다. 팀에서는 안전 스캐너를 이미 우회하였거나 의도한대로 동작하지 않는 수많은 경우에 대해 구현할 여러가지 새로운 컨트롤을 논의하였다. 그 중 하나는, 작업자가 스캐너를 끌 수 있는 기능을 제공하고, 주변 구역의 모든 작업자에게 경고등 또는 경보음 등의 피드백을 제공하도록 설계를 변경하는 것이었다. 팀은 추가적으로, AGV에서 운전자 및 주변 작업자에게 일반적인 피드백을 제공하는 방법에도 집중했다. 기존의 설계는 단순히 작업자가 명령했다고 믿는 것 또는 주변의 작업자가 로봇이 동작할 것이라고 믿는 것이 아니라, 시스템이 실제로 수행한 것을 알 수 있도록 유용하고 쉽게 표시되는 피드백 없이, 애초에 설계된 대로 그리고 운영자가

AGVs are large and powerful mobile systems that people cannot simply put a fence around to keep workers safe. Given the numerous ways that safety scanners can be and already have been bypassed or do not function as intended, the team discussed multiple new controls to implement. One was a design change to enable the ability for workers to turn the scanners off and provide feedback to everyone in the surrounding area that the scanners have been disabled, such as warning lights or an audible tone. Additionally, the team focused on ways to provide feedback from the AGV to the operator and surrounding workers in general. The original design seemed mostly to assume that the AGV would operate exactly as originally designed and as the operator intended, without helpful and readily displayed feedback to know what the system is actually doing, not just what the operator believes they commanded or what surrounding workers believe the robot should be doing.

Another discussion involved programming motor feedback logic for the drive motors of the AGV as seen in many applications of collaborative robotics. Due to the number of scenarios that resulted in the safety scanners being the only control in place and the number of ways the scanners may not maintain proper control, a secondary method of control for the mobile functions of the system was considered. While this control may still allow minimum separation to be violated as defined in the hazards, the AGV would stop moving and either avoid an accident or lessen the severity of the accident.

While these are just a few examples of the discussion that resulted from the process and results of STPA, the results for the rest of the system were very similar. In order to avoid the defined system-level hazards, there were design changes recommended for communication of process model variables between systems that required coordination and control, for system motor feedback logic for various moving parts, and for increased levels of feedback from the physical system to the human operators and surrounding workers. The team focused on determining how the system could violate the safety controls in place, potentially resulting in an accident, and then how those existing controls needed to change or what new controls were needed to be put in place.

STPA Applied to a Lockout/Tagout (LOTO) Procedure

A second brief example is included to show that STPA is also applicable to more traditional workplace safety hazards. The example is traditional lockout/tagout (LOTO), which is a safety procedure used to ensure that dangerous machines are properly shut off and not able to be started up again prior to completion of maintenance or servicing work. It requires that energy sources be isolated and rendered inoperative before work is started on the equipment involved. The isolated power sources are then locked and a tag is placed on the lock identifying the worker who has placed it. The worker then holds the key for the lock, ensuring that only he or she can start the machine. The goal is to prevent accidental startup of a machine while it is in a hazardous state or while a worker is in direct contact with it.

While this process seems relatively foolproof, in fact, a study conducted by the United Auto Workers found that 20% of the fatalities (83 of 414) that occurred among their members between 1973 and 1995 were attributed to inadequate lockout/tagout procedures.

Accidents:

A1: Person is injured

A2: Product is damaged

A3: Manufacturing equipment is damaged

A4: Delivery is delayed

Hazards:

H1: Person is exposed to hazardous energy [A1]

의도한 대로 AGV가 정확히 동작할 것이라고 가정한 것으로 보였다.

또 다른 논의는, 많은 협업 로봇 어플리케이션에서 볼 수 있듯이 AGV 드라이브 모터의 피드백 로직 프로그래밍과 관련된 것이었다. 안전 스캐너가 유일한 제어 수단이 되는 여러 시나리오와 스캐너가 적절한 제어를 유지하지 못하는 여러 경우의 수로 인해 시스템의 이동 기능을 제어하는 보조적인 방법이 검토되었다. 이 제어로 인해 위험에 정의된 최소이격거리가 여전히 위반될 수 있지만 AGV가 이동을 멈추고 사고를 피하거나 사고의 심각성을 줄이게 된다.

이러한 것들은 STPA 프로세스 및 결과에서 나온 일부 예시에 불과하지만 나머지 시스템의 결과 또한 이와 매우 유사하다. 정의된 시스템 수준의 위험을 피하기 위해 다음과 같은 설계 변경 권고사항이 있었다; 조정(coordination) 및 컨트롤이 필요한 시스템 간의 프로세스 모델 변수의 커뮤니케이션, 다양한 가동 부품(parts)을 위한 시스템 모터 피드백 로직, 물리적 시스템에서 운영자와 주변 작업자로의 피드백 수준의 증가. 팀에서는 시스템이 어떻게 안전 컨트롤을 위반하여 잠재적으로 사고를 야기할 수 있는지를 결정한 다음 기존 컨트롤이 어떻게 변경되어야 하는지 또는 어떤 컨트롤이 추가되어야 하는지를 결정하는 데 중점을 두었다.

Lockout/Tagout(LOTO) 절차에 적용된 STPA

STPA가 보다 전통적인 작업장 안전 위험에도 적용 가능하다는 것을 보여주기 위해 여기에 두 번째 간단한 예시를 소개한다. 이 예시는 전통적인 Lockout/tagout(LOTO)로, 위험한 기계를 적절하게 차단하고(shut off), 유지보수나 정비 작업이 완료되기 전에는 다시 시작하지 못하도록 보장하는 안전 절차를 말한다. 이를 위해 장비에서 작업을 시작하기 전에 전원을 분리하여 작동 불가한 상태로 만들어야 한다. 그런 다음, 격리된 전원 공급 장치를 잠그고(lock), 잠금 장치에 태그를 붙이고 그 태그를 붙인 작업자를 태그에 표시한다. 잠금 장치의 열쇠는 작업자가 가지고 있어 그 사람만이 기계를 시작할 수 있음을 보장하는 것이다. LOTO의 목표는 위험한 상태 있거나 작업자가 장비를 직접 접촉하고 있을 때 우연히 기계가 구동(startup)되는 것을 막는 것이다.

이 절차는 비교적 실수의 여지가 없어 보이지만, 실제로 미국의 자동차 노동 조합(United Auto Workers) 연구에 따르면 1973년과 1995년 사이에 발생한 사망자 수의 20%(414명 중 83명)가 lockout/tagout 절차를 부적절하게 수행한 것에서 기인한 것으로 나타났다.

사고:

A1: 사람의 부상
A2: 제품의 손상
A3: 제조 장비의 손상
A4: 배송의 지연

위험:

H1: 사람이 위험한 에너지(hazardous energy)에 노출됨 [A1]
H2: 제품이 과도하거나 부적절한 위험한 에너지에 노출됨 [A2]
H3: 제조장비가 과도하거나 부적절한 위험한 에너지에 노출됨 [A3]
H4: 중요한 시점에 작업이 지연됨 [A4]

안전 제약사항:

SC1: 사람들은 과도하거나 부적절한 형태의 위험한 에너지에 노출될 수 있는 조건에서 작업

H2: Product is exposed to excessive or inappropriate hazardous energy [A2]

H3: Manufacturing equipment is exposed to excessive or inappropriate hazardous energy [A3]

H4: Work is delayed at critical junctures [A4]

Safety Constraints:

SC1: People must not work in conditions where they can be exposed to excessive or inappropriate forms of hazardous energy. [H1]

SC2: Hazardous energy must only be activated within appropriate and established threshold conditions. [H2]

SC3: Manufacturing equipment must not be used in conditions where it can be exposed to excessive or inappropriate hazardous energy. [H3]

SC4: Work must not be delayed at critical junctures of the process (sign off, lock applied/removed, etc.) [H4]

A generic control structure for this example is shown in Figure 4.6.

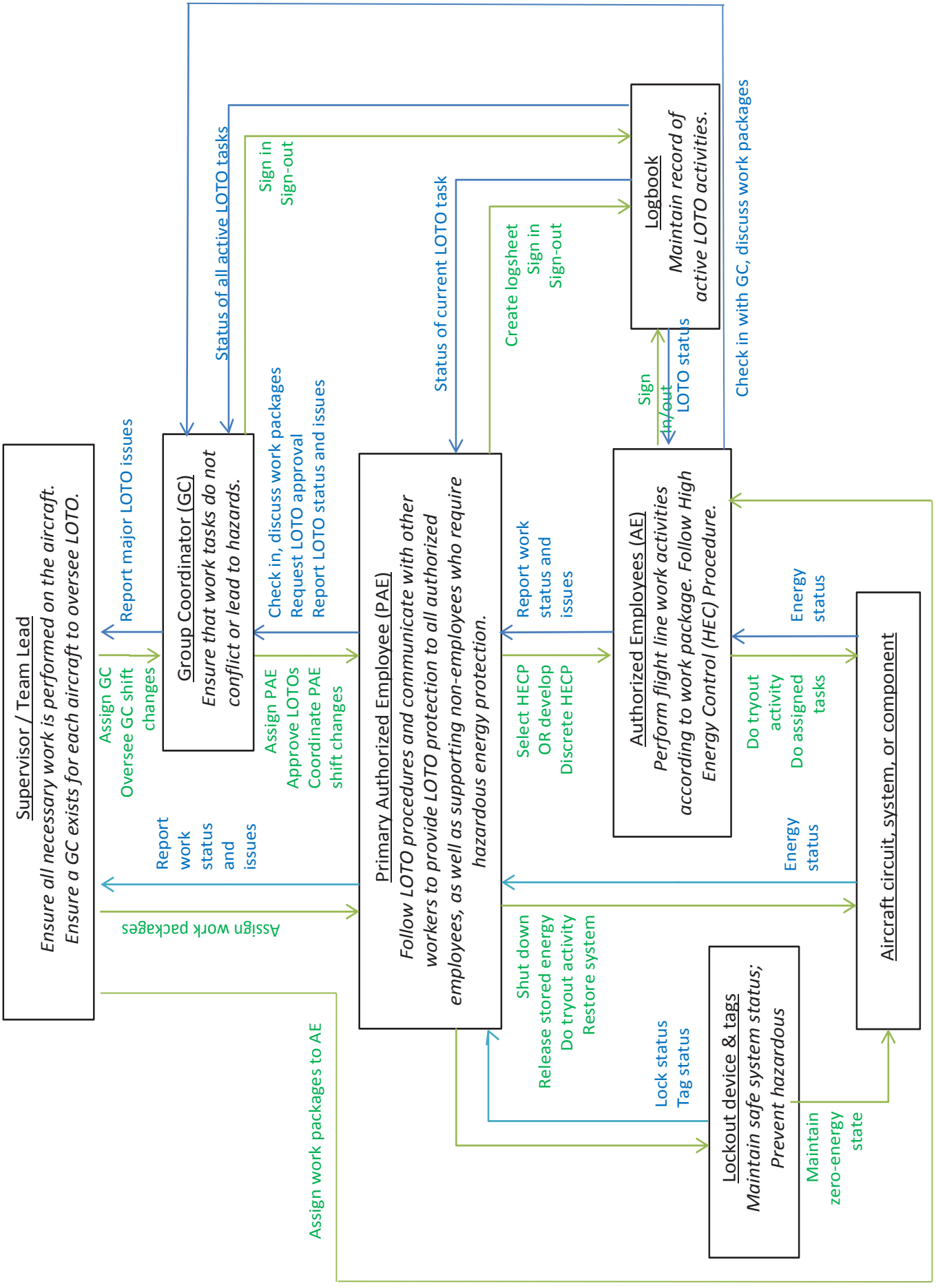
해서는 안 된다 [H1]

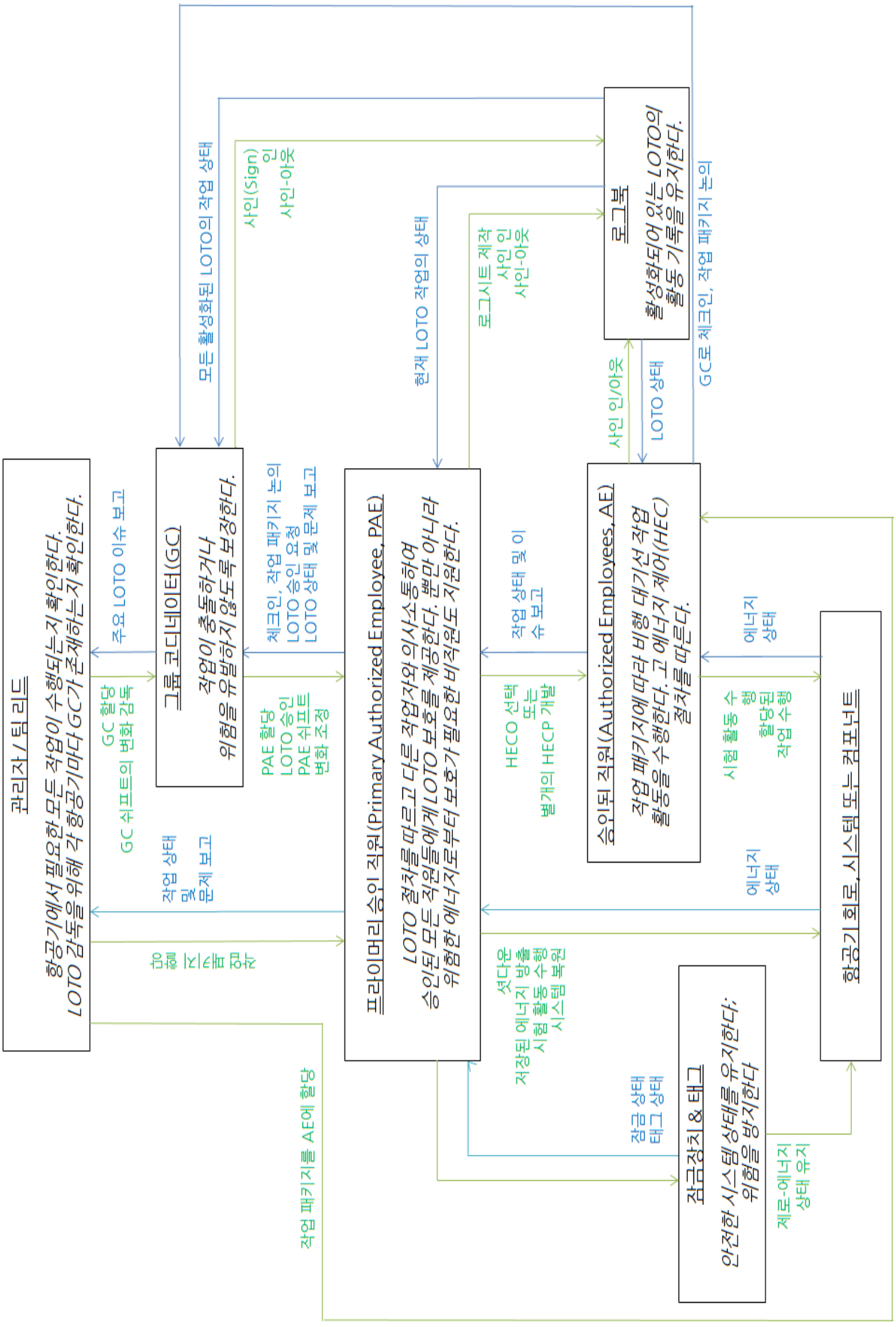
SC2: 위험한 에너지는 적절하고 확실한 임계치 내에서만 활성화되어야 한다 [H2]

SC3: 과도하거나 부적절한 위험한 에너지에 노출될 수 있는 환경에서 제조 장비를 사용해서는 안 된다 [H3]

SC4: 프로세스의 중요한 시점(종료, 잠금적용/해제 등)에 작업을 지연시켜서는 안 된다 [H4]

이 예제에 대한 일반적인 컨트롤 스트럭처는 그림 4.6에서 볼 수 있다.


Supervisor / Team Lead
Ensure all necessary work is performed on the aircraft.
Ensure a GC exists for each aircraft to oversee LOTO.
Group Coordinator (GC)
Ensure that work tasks do not conflict or lead to hazards.
Primary Authorized Employee (PAE)
Follow LOTO procedures and communicate with other workers to provide LOTO protection to all authorized employees, as well as supporting non-employees who require hazardous energy protection.
Authorized Employees (AE)
Perform flight line work activities according to work package. Follow High Energy Control (HEC) Procedure.
Logbook
Maintain record of active LOTO activities.
Lockout device & tags
Maintain safe system status; Prevent hazardous
Aircraft circuit, system, or component
Report major LOTO issues
Assign GC
Oversee GC shift changes
Report work status and issues
Assign work packages
Status of all active LOTO tasks
Sign in
Sign-out
Check in, discuss work packages
Request LOTO approval
Report LOTO status and issues
Assign PAE
Approve LOTOs
Coordinate PAE shift changes
Status of current LOTO task
Create logsheet
Sign in
Sign-out
Report work status and issues
Select HECP
OR develop
Discrete HECP
Sign in/out
LOTO status
Check in with GC, discuss work packages
Energy status
Do tryout activity
Do assigned tasks
Energy status
Shut down
Release stored energy
Do tryout activity
Restore system
Lock status
Tag status
Maintain zero-energy state
Assign work packages to AE


관리자 / 팀 리드
항공기에서 필요한 모든 작업이 수행되는지 확인한다.
LOTO 감독을 위해 각 항공기마다 GC가 존재하는지 확인한다.
그룹 코디네이터(GC)
작업이 충돌하거나
위험을 유발하지 않도록 보장한다.
프라이머리 승인 직원(Primary Authorized Employee, PAE)
LOTO 절차를 따르고 다른 작업자와 의사소통하여
승인된 모든 직원들에게 LOTO 보호를 제공한다. 뿐만 아니라
위험한 에너지로부터 보호가 필요한 비직원도 지원한다.
승인된 직원(Authorized Employees, AE)
작업 패키지에 따라 비행 대기선 작업
활동을 수행한다. 고 에너지 제어(HEC)
절차를 따른다.
항공기 회로, 시스템 또는 컴포넌트
잠금장치 & 태그
안전한 시스템 상태를 유지한다;
위험을 방지한다
로그북
활성화되어 있는 LOTO의
활동 기록을 유지한다.
모든 활성화된 LOTO의 작업 상태
사인(Sign) 인 사인-아웃
현재 LOTO 작업의 상태
로그시트 제작 사인 인 사인-아웃
사인 인/아웃
LOTO 상태
GC로 체크인, 작업 패키지 논의
주요 LOTO 이슈 보고
GC 할당
GC 쉬프트의 변화 감독
체크인, 작업 패키지 논의
LOTO 승인 요청
LOTO 상태 및 문제 보고
PAE 할당
LOTO 승인
PAE 쉬프트
변화 조정
작업 상태 및 이슈 보고
HECO 선택 또는 별개의 HECP 개발
에너지 상태
시험 활동 수행 할당된 작업 수행
에너지 상태
작업 상태 및 문제 보고
성다운 저장된 에너지 방출 시험 활동 수행 시스템 복원
잠금 상태 태그 상태
제로-에너지 상태 유지
작업 패키지를 AE에 할당

The table below shows some of the causal scenarios for two UCAs: PAE (Primary Authorized Employee) does not de-energize component being worked on and PAE de-energizes component being worked on after work has started and before it is completed and workers are safely out of the area. Some of these are specified in enough detail to identify fixes. If they are not, then more details will need to be added to the causal scenarios.

PAE does not de-energize component being worked on	…because she believed she was de-energizing the correct component because of insufficient specificity in the HECP. …because she believed she was de-energizing the correct component because of lack of training. …because she believed she was de-energizing the correct component because because of insufficient specificity in the High Energy Control procedures. …because she believed that it was not necessary to de-energize the component because of lack of training. (Training) …because she believed that it was not necessary to de-energize the component because of lack of experience. …because she believed that it was not necessary to de-energize the component because she believed that someone else had de-energized the component …because she believed that it was not necessary to de-energize the component because she believed that she had already done it
PAE de-energizes component being worked after work is started and before it is completed	…because she believed the work had not yet begun because it was not scheduled …because she believed it was not necessary to de-energize the component Because she believed that someone else had de-energized the component. …because she believed that it was not necessary to de-energize the component because she believed that she had already done it (incorrect process model).

아래 표는 두 UCA에 대한 원인 시나리오 중 일부를 보여준다: 첫 번째 UCA는 PAE(Primary Authorized Employee)가 작업 중인 컴포넌트의 전원을 차단하지 않는 것이고, 두 번째 UCA는 작업이 시작되고 작업이 완료되어 작업자들이 안전하게 구역을 벗어나기 전에 PAE가 작업 중인 컴포넌트의 전원을 차단하는 것이다. 이 중 일부는 수정사항을 식별할 수 있을 정도로 자세한 세부 사항이 명시되어 있다. 그렇지 않은 경우 원인 시나리오에 보다 상세한 사항을 추가해야 한다.

PAE가 작업 중인 컴포넌트의 전원을 차단(de-energize)하지 않음	… HECP의 불충분한 구체성(specificity)으로 인해 PAE가 정확한 컴포넌트의 전원을 차단했다고 믿었기 때문에 … 훈련 부족으로 인해 PAE가 정확한 컴포넌트의 전원을 차단했다고 믿었기 때문에 … 고 에너지 제어(High Energy Control) 절차의 불충분한 구체성(specificity)으로 인해 정확한 컴포넌트의 전원을 차단했다고 믿었기 때문에 … 훈련 부족으로 인해 PAE가 컴포넌트에 전원을 차단할 필요가 없다고 믿었기 때문에 … 경험 부족으로 인해 PAE가 컴포넌트에 전원을 차단할 필요가 없다고 믿었기 때문에 … PAE가 다른 누군가가 컴포넌트의 전원을 이미 차단하였다고 믿고, 컴포넌트의 전원을 차단할 필요가 없다고 믿었기 때문에 … PAE가 자기가 이미 전원을 차단했다고 생각해서, 컴포넌트의 전원을 차단할 필요가 없다고 믿었기 때문에
PAE가 작업이 시작되고 완료되기 전에 작업 중인 컴포넌트의 전원을 차단함(de-energize)	… 작업이 스케줄되지 않았으므로 PAE가 작업이 아직 시작되지 않았다고 믿었기 때문에 … 다른 누군가가 컴포넌트의 전원을 이미 차단하여서 PAE가 컴포넌트에 전원을 차단할 필요가 없다고 믿었기 때문에 … PAE가 자기가 이미 전원을 차단했다고 생각해서, 컴포넌트의 전원을 차단할 필요가 없다고 믿었기 때문에(잘못된 프로세스 모델)

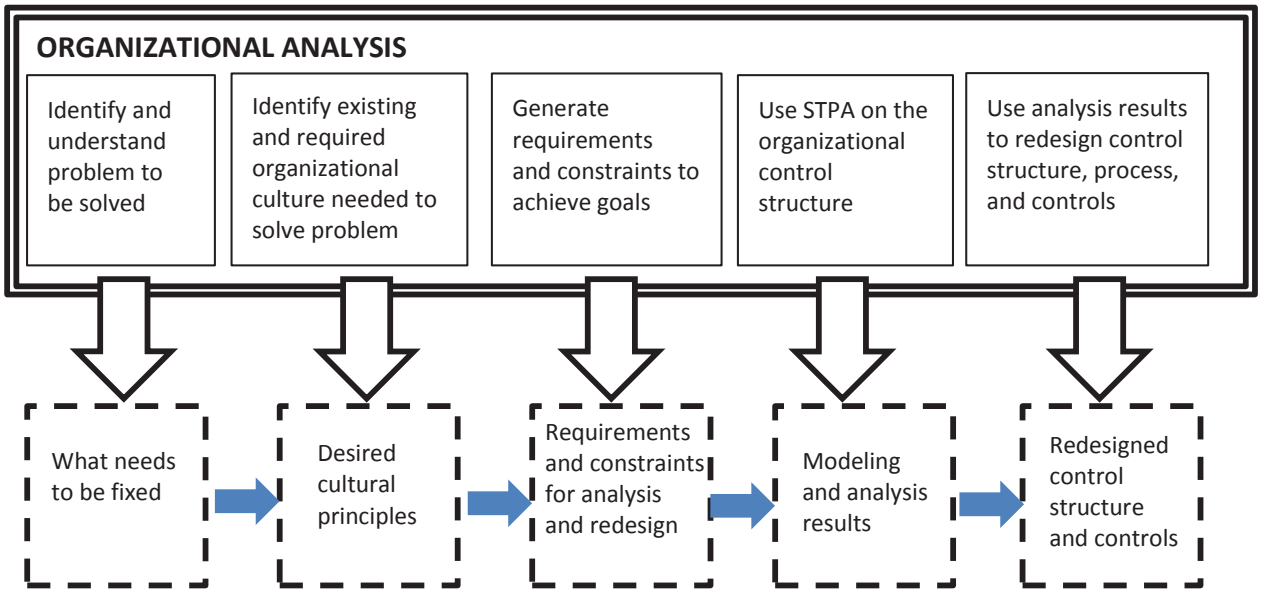
Chapter 5: Organizational and Social Analysis

Nancy Leveson

Applying STPA to an organizational or social system is essentially a system engineering or re-engineering exercise, but the thing being engineered (the product) is the organization itself. In this type of system engineering process, the goal is to determine where the system is now, where it needs to go, and how it can get there. The products and byproducts of the analysis will include:

1. The engineering and business problems that need to be solved, i.e., the goals of the analysis
2. The organizational culture required to achieve the goals
3. Requirements and constraints for the organizational structure
4. STPA analysis of the organizational structure
5. Use of the analysis results to:
    a) Design or improve the organizational structure
    b) Identify risks and leading indicators of risk and
    c) Design (or re-design) the risk management process.

Each of these is described in this chapter of the handbook. The overall process is summarized in the box below.



The application of STPA to managerial and social systems is similar to but differs in some significant ways from its use on engineered products. The most important difference involves the initial understanding of the problem to be solved. While any system engineering or re-engineering process should start with understanding the problem, this understanding process may be more complex when applied to an organization than that used for products. The goals and constraints for an aircraft, automobile, or nuclear power plant are fairly well understood and the first step in analyzing such a system is to get the stakeholders together to select the specific goals and constraints they want applied to their particular product.

In contrast, when doing an analysis of a social or managerial system (which usually already exists), the first step is to identify and understand why the goals are not being achieved currently or how they

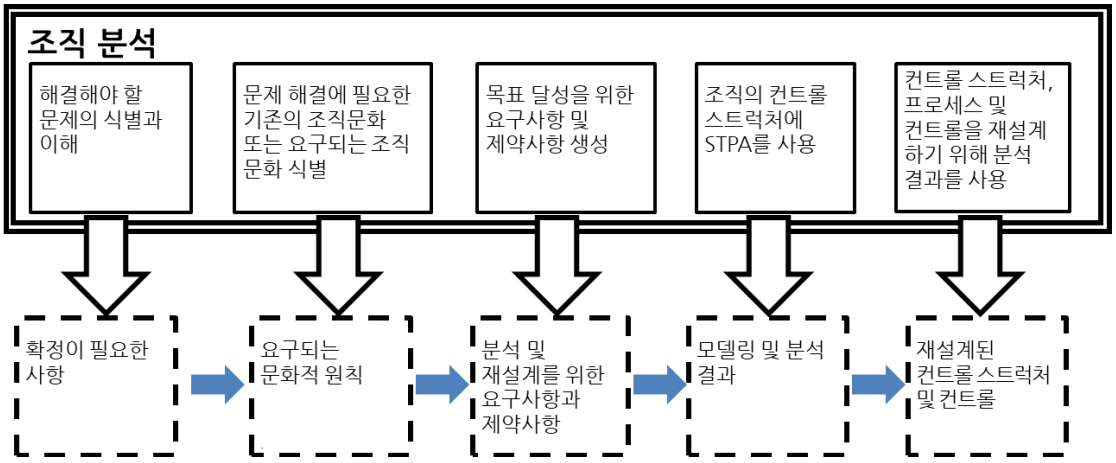
5장. 조직 및 사회 분석

Nancy Leveson

STPA를 조직(organizational) 또는 사회(social) 시스템에 적용하는 것은 본질적으로 시스템 엔지니어링 또는 리엔지니어링 작업이지만, 엔지니어링 되는 것(결과물)은 조직 자체이다. 이러한 유형의 시스템 엔지니어링 절차에서 목표는 시스템이 현재 어디에 있는지, 어디로 가야 하는지, 어떻게 도달할 수 있는지를 결정하는 것이다. 분석의 결과 및 부수적 결과에는 다음 내용들이 포함되어야 한다.

1. 해결되어야 할 엔지니어링 및 비즈니스 문제, 즉 분석의 목표
2. 목표 달성을 위해 필요한 조직 문화
3. 조직 구조에 대한 요구사항 및 제약사항
4. 조직 구조의 STPA 분석
5. 아래에 분석 결과를 사용:
    a) 조직 구조 설계 또는 개선
    b) 리스크 및 리스크의 선행지표 도출
    c) 리스크 관리 절차 설계(또는 재설계)

본 장에서는 이들 각각에 대하여 설명한다. 전체 과정은 아래 그림과 같다.



관리(managerial) 및 사회(social) 시스템에 대한 STPA의 적용은 공학 제품(engineered products)에서의 사용 방법과 유사하지만 몇 가지 중요한 차이점이 있다. 가장 중요한 차이점은 해결해야 할 문제에 대한 초기 이해에 관한 것이다. 모든 시스템 엔지니어링 또는 리엔지니어링 절차는 문제를 이해하는 것에서 시작해야 하지만, 제품에 적용하는 것보다 조직에 적용할 때 이해 절차가 더 복잡할 수 있다. 항공기, 자동차, 원자력 발전소의 목표와 제약사항은 상당히 잘 이해될 수 있으며, 그러한 시스템을 분석하는 첫 번째 단계는 이해관계자와 함께 특정 제품에 적용할 구체적인 목표와 제약사항을 선택하는 것이다.

이와 반대로 사회 또는 관리 시스템(일반적으로 이미 존재하는)을 분석할 때의 첫 번째 단계는 왜 현재 목표가 달성되지 않고 있는지, 어떻게 더 잘 달성할 수 있는지를 식별하고 이해하는 것이다. 이 핸드북에 있는 대부분의 예에서 안전에 초점을 두고 있으므로, 여기에서는 안전이나 보안이 아닌 발현속성(emergent property)에 대한 STPA 적용 방법의 예시를 살펴보겠다.

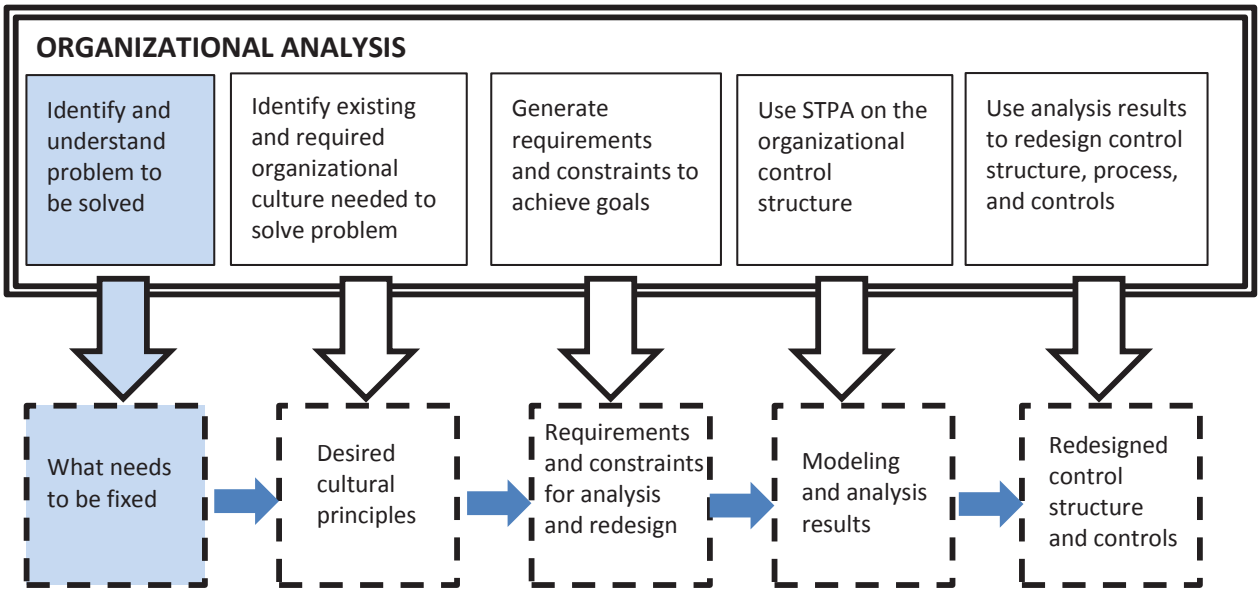
여기에 사용된 예는 시스템 엔지니어링의 효과, 즉 주어진 예산 내에서 정시에 제품을 출시하고 고객의 요구를 만족시키는 것이다. 이 예에서 고려한 회사는 정부 및 영리 기업을 포함한 외부 고객을

could be better achieved. Because most of the examples in this handbook focus on safety, the example here will examine a different property in order to illustrate how to use STPA for an emergent property different than safety or security.

The example used here is system engineering effectiveness, namely, getting products out the door on time, within budget, and satisfying the needs of customers. An assumption is made that the company being considered produces complex, engineered products for external customers, including both government and commercial entities.

Because the proprietary information obtained when we have performed this process on real companies cannot be divulged, generic results of the process and of its use on real companies is presented here instead.

Identifying the Engineering and Business Problems that Need to be Solved



Although not always followed, the first step in a successful solution of a problem is to understand the problem that is to be solved. Identifying the solution before spending time understanding the problem to be solved is not very likely to lead to success.

Interviews with stakeholders and key personnel are a useful way to identify the problems needing to be solved and the perceived causes of those problems as is simple observation. The resulting information is then used for an analysis of the current system and potential improvements. Information about how to do such interviews is contained in the social science literature, but we have been able to get the information we need using a simple and straightforward interview process that first describes our goals, promises anonymity, and then probes with simple questions, allowing lots of freedom for the interviewee to steer the answers in the direction they want.

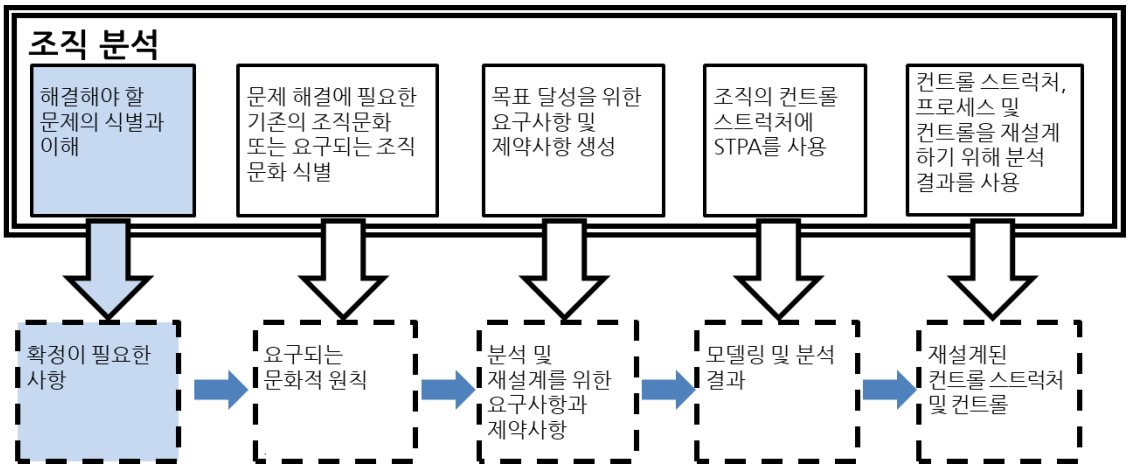
For the purpose of this example, let's suppose that the interview process asks about the reasons why system engineering may not always be successful in their company. The interviewees should at least include management, engineers, and, if possible, customers. Given a focus on system engineering, the results might turn up the following perceived explanations of the problems and why they are occurring:

- The projects do not always have the expertise and skills at the time they are needed. Skill development is not as strong as it needs to be.

대상으로 복잡한 엔지니어링 제품을 생산한다고 가정한다.

우리가 이 프로세스를 수행했을 때 실제 기업에서 얻은 독점 정보는 유출할 수 없기 때문에, 여기에서는 실제 회사에서의 프로세스 및 그 사용에 대한 일반적인 결과를 소개한다.

해결해야 할 엔지니어링 및 비즈니스 문제 식별



항상 그런 것은 아니지만, 문제의 성공적인 해결을 위한 첫 번째 단계는 해결해야 할 문제를 이해하는 것이다. 해결할 문제를 이해하는 시간을 갖기 전에 해결책을 도출하는 것은 성공으로 이어지기 어렵다.

해결해야 할 문제와 문제에 대해 인지된 원인을 간단한 관찰로써 식별하는 유용한 방법은 이해관계자 및 핵심 인력과의 인터뷰이다. 인터뷰 결과 도출된 정보는 현재 시스템의 분석과 잠재적 개선사항을 위해 사용된다. 이러한 인터뷰 방법에 대한 정보는 사회 과학 문헌(social science literature)에 있기도 하지만, 우리는 간단하고 복잡하지 않은 인터뷰 프로세스를 통해 필요한 정보를 얻을 수 있었다. 즉, 우리의 목표를 먼저 설명하고, 익명을 약속한 후 인터뷰 대상자가 그들이 원하는 대로 자유롭게 답변할 수 있게 한 상태에서 간단한 질문들을 던졌다.

이 예제의 목적을 위해 인터뷰 프로세스에서 시스템 엔지니어링이 회사에서 항상 성공하지 못하는 이유를 물어 본다고 가정해 보자. 인터뷰 대상자에는 최소한 경영진과 엔지니어를 포함해야 하고, 가능한 경우, 고객도 포함한다. 시스템 엔지니어링에 중점을 두면 아래와 같은 문제에 대한 설명과 문제의 발생 원인을 도출해낼 수 있다.

- 프로젝트에서 필요한 시점에 전문지식(expertise)과 기술(skills)이 항상 있는 것은 아니다. 기술(skill) 개발은 필요한 것만큼 그렇게 강력하지는 않다.
- 제안서 작성 및 계약 협상은 비현실적인 예산 및/또는 일정을 초래한다. 그래서 프로젝트가 뒤늦게 시작되고, 이는 제품 개발 과정에서 대충 수행하는 상황(cutting corners)으로 이어져 결과적으로 상황을 악화시키게 된다.
- 리엔지니어링 프로세스에서 해결해야할 문제를 결정하는 것이 매우 중요한 것처럼, 솔루션을 만들기 이전에 풀어야 할 문제가 무엇인지를 결정하는 것 역시 중요하며, 이는 고객의 니즈를 만족시키고 이후 시스템 엔지니어링 프로세스 자체에서도 백업하거나 재작업하는 것을 피하기 위해 필요하다. 그러나 이 예제의 회사에서는 사전 계획단계와 개발단계 초기에 거의 노력을 기울이지 않았다. 이것은 고객의 니즈와 제품이 사용될 환경을 이해하는 데 너무 적은 시간을

- The proposal writing and contract negotiations result in unrealistic budgets and/or schedules. So the projects start out behind, which leads to cutting corners in product development and, in the end, simply makes things worse.
- Just as determining the problem to be solved in the re-engineering process we are describing here is critical, so is determining what problem is being solved before creating a solution is important in satisfying customer needs and in avoiding having to backup and redo work later in the system engineering process itself. But in this example company, too little effort is usually spent in upfront planning and the early stages of development. This includes too little time understanding customer needs and the environment in which the product will be used. Engineers waste time on resolving problems that arise from bad plans and from not identifying the critical assumptions and requirements about the desired product in the early stages of development. Hasty decisions are made that cost time and money later and often result in a product that is not as useful as originally intended.
- Lack of early planning leads to being late and to continual re-planning, which wastes time and creates project "thrashing," where so much time is spent in meetings and re-planning that forward progress is seriously degraded and also to budget and schedule problems. The haste in which changes start to be made simply accelerates the problems as change impact analysis and change management starts to degrade.
- Project risk analysis and mitigation is perfunctory and becomes a "check the box" activity. Few resources are assigned to identify and mitigate risk. Often, risk analysis is given a lower priority than other activities and delegated to new engineers who do not have enough experience to do it well. In addition mitigation of risks is often unfunded and the steps to achieve it are not in the master schedule.
- System engineering gates and milestones were created to ensure readiness for the later steps in the process. While most everyone understands the need for the gates, pressure builds to remain on schedule. Longer term schedule and quality concerns are subsumed by short-term focus on predefined schedules. Project managers will often skip over milestones ("gates"), assuming they will be covered in the next phase. The projects never catch up and end up fixing problems late, which is, of course, always enormously expensive. Budgets and schedules are blown to pieces when changes are needed late in the project. Even worse, many, if not most, of the critical decisions are made in the early stages of development and cannot be undone without enormous and usually impractical redesign. At that point, the only viable solution usually is to create workarounds, which result in a deficient product. Decisions and changes made in haste often create even worse problems than those being solved.
- Related to the problems involved in rework is the often inadequate recording of design intent and rationale. Lack of such documentation can lead to solutions that were earlier discarded as inadequate being suggested and tried again.
- All of this leads to heroism and cowboy engineering becoming the standard and being highly rewarded by management. There is accountability for satisfying schedule and cost goals, but there is little accountability or high rewards for first time quality, by which is meant few changes required because of deficiencies in earlier stages of development.
- Inadequate learning may result from both success and failure along with inadequate feedback to provide continual improvement. There is too much emphasis on following rigid processes as the solution to all problems. Managers think that if they just institute the right process, all the system engineering problems will be solved while forgetting that people are involved. Management needs to be more intelligent in the way they use process as the solution to a problem, and, alternatively,

할애했음을 말한다. 엔지니어는 잘못된 계획으로 인해 발생하는 문제와 원하는 제품에 대한 중요 가정 및 요구사항을 개발 초기에 식별하지 못하여 발생하는 문제의 해결에 시간을 낭비하게 된다. 이후 시간과 돈이 들어가는 성급한 결정이 내려지고, 종종 원래 의도했던 것보다 유용하지 못한 제품이 만들어지게 된다.

- 초기 계획의 부족은 개발의 지연 또는 시간을 낭비하고 프로젝트를 "쓰레기"로 만드는 지속적인 재계획(re-planning)으로 이어진다. 회의와 재계획에 너무 많은 시간을 소비하여 앞으로의 진행이 심각하게 저하되고 동시에 예산과 일정 문제도 발생한다. 변경이 시작될 때 서두르면 변경 영향 분석(change impact analysis)과 변경 관리(change management)의 수준이 저하되어 단순히 문제를 가속화할 뿐이다.
- 프로젝트 리스크 분석 및 완화조치는 형식적이며 "체크 박스" 활동이 된다. 리스크를 식별하고 완화하는 데 필요한 자원은 거의 없다. 종종 리스크 분석은 다른 활동보다 우선순위가 낮으며 분석을 잘 수행하기에는 경험이 부족한 새로운 엔지니어에게 할당된다. 또한 리스크 완화조치는 종종 예산이 주어지지 않으며 달성을 위한 단계가 기본 일정계획(master schedule)에 포함되지 않는다.
- 시스템 엔지니어링 게이트와 마일스톤은 프로세스 이후 단계의 준비를 보장하기 위해 만들어졌다. 대부분의 사람들이 게이트의 필요성을 이해하고 있음에도 일정 압박은 여전히 존재한다. 장기적인 일정과 품질의 문제는 사전에 정의된 일정에서 단기적인 요소들의 부분합에 의해 결정된다. 프로젝트 관리자는, 다음 단계에서 다루어 질 것이라는 가정하에 마일스톤("게이트")을 종종 건너뛰는 경우가 있다. 프로젝트는 결코 건너뛴 문제를 해결하지 못하고 이후에도 문제를 해결하지 못하며 이것들에는 항상 막대한 비용이 든다. 프로젝트 후반부에 변경이 필요해지면 예산과 일정은 급격하게 소모된다. 더 안 좋은 것은, 모두 그런 것은 아니지만, 개발 초기 단계에서 중요한 많은 결정들이 이루어지며 그러한 결정은 대규모의, 일반적으로는 비현실적인 재설계 없이는 되돌릴 수 없다. 이 시점에서 실행 가능한 유일한 솔루션은 임시적인 해결책을 만드는 것이며 이로 인해 결함 있는 제품이 만들어지게 된다. 성급한 결정과 변경은 종종 해결되는 것보다 훨씬 더 심각한 문제를 만든다.
- 재작업과 관련된 문제는 종종 설계 의도와 근거를 부적절하게 기록하는 것이다. 이런 문서화의 부족은 이전에 부적절하다고 여겨 폐기하였던 솔루션을 다시 제안하고 시도하는 결과를 초래할 수 있다.
- 이 모든 것은 영웅주의와 카우보이 엔지니어링(역주: 체계적 단계를 밟지 않고 일단 결과물부터 만드는데 집중하는 접근법)이 표준이 되고 경영진으로부터 높이 보상받는 결과를 초래한다. 일정과 비용 목표를 만족시키는 것에는 책임(accountability)이 있지만, 초기 품질에 대해서는 책임이 거의 없거나 높은 보상이 없다. 이것은 초기 개발 단계의 결함을 수정하기 위한 변경이 거의 없음을 의미한다.
- 부적절한 학습은 지속적인 개선을 위한 피드백의 부적절함, 그리고 성공과 실패 모두로 인해 발생할 수 있다. 모든 문제에 대한 해결책으로 엄격한 프로세스를 따르라고 하는 것이 너무 많이 강조된다. 관리자는 단지 올바른 프로세스를 도입하기만 하면 모든 시스템 엔지니어링 문제가 해결될 것이라고 생각하지만 이는 사람들이 관련되어 있다는 사실을 잊은 것이다. 경영진은 프로세스를 문제에 대한 해결책으로 사용하는 방식을 좀 더 지능적으로 할 필요가 있고 다른 대안으로, 개발 프로세스에서 휴먼 컴포넌트를 어떻게 개발하고 지원할지의 방법에 대해서도 지능적일 필요가 있다. 회사는 "관찰된(observed)" (데이터 수집) 교훈에는 능숙하지만 "배운(learned)" 교훈에는 능숙하지 않다.

how they develop and support the human components of the development process. The company is good at lessons "observed" (collecting data) but not lessons "learned."

- Learning requires innovation, risk taking, and flexibility (in contrast to rigorously enforcing a specific process). However, there is not enough empowerment and support for those factors that promote innovation in the workforce. A common view expressed in the interviews is that employees are afraid to take risks and innovate because there is little reward for this and, in fact, often punishment ensues. So they don't step outside of what they have always done and been told to do. The fear of failure and resulting punishment can stifle innovation. The current engineering structure is described as not being able to handle disruptive innovation and as having little ability to handle change.
- Another reason for lack of innovation that often comes up in our interviews is that the bulk of engineering innovation and energy is spent resolving program problems arising from poor practices or deficient risk management and not on efforts to create better products.
- There is inadequate design focus on producibility and supportability (maintainability), resulting in cost and quality factors in the inevitable product evolution.
- Too much emphasis is placed on ensuring safety and security in a completed design or architecture and not enough on building safety and security into the product from the beginning.
- For companies that subcontract parts of their product, we often hear about ineffective supplier oversight leading to schedule slips and the need for rework.
- There is lack of independence of technical decisions from programmatic and business strategy considerations when schedule is in conflict with quality. This may be coupled with a lack of open, honest communication and independent paths to elevate technical risk and issues.
- Engineers and managers are not getting the information and feedback they need for good decision making. Sometimes they are ignoring the information they do have. Feedback loops are either nonexistent or ineffective. There are complaints that feedback is often received but ignored. There is a well-known phenomenon called *confirmation bias,* which means that people tend to believe information that supports their hypotheses and to ignore information that does not. Confirmation bias is related to *defensive avoidance*, which occurs when information one does not want to hear is disbelieved or ignored.
- Some problems often expressed, especially by those in manufacturing or by customers, are related to the relationship between the designers and the product assemblers or between the designers and the operators of the systems they design. Common problems are associated with deficient work instructions and lack of feedback from mechanics to design engineers.

While the number of problems described above may seem surprising, in fact, they are fairly common in industrial system engineering organizations.

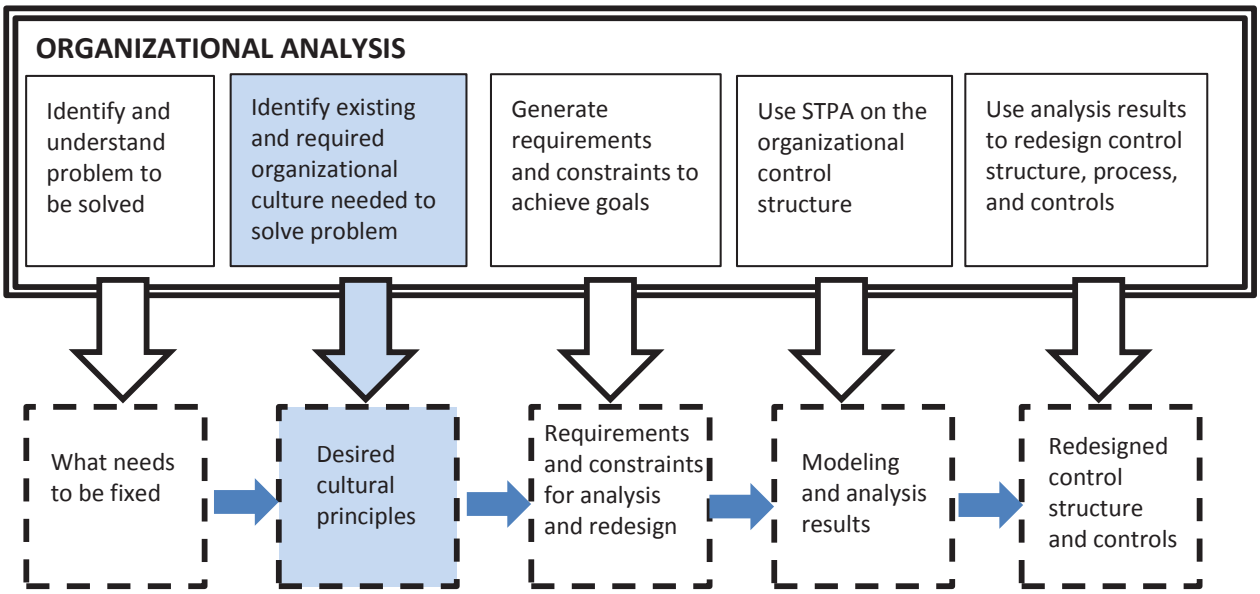
During the interviews, we usually also try to learn about the current organizational culture.

- 학습(learning)은 혁신, 위험 감수, 그리고 유연성이 필요하다(특정 프로세스를 엄격하게 강요하는 것과 대조적임). 그러나 노동력(workforce) 혁신을 촉진하는 요소를 위한 권한부여(empowerment) 및 지원은 충분하지 않다. 인터뷰에서 표현된 공통적인 견해는 직원들이 위험을 감수하고 혁신하기를 두려워한다는 것이다. 그 이유는 보상이 거의 없고 실제로, 종종 처벌이 뒤따르기 때문이다. 그래서 그들은 항상 해왔던 일에서 벗어나지 않는다. 실패와 그에 따른 처벌에 대한 두려움은 혁신을 저해할 수 있다. 현재의 엔지니어링 구조는 분열적인(disruptive) 혁신을 감당할 수 없고 변화를 처리할 수 있는 능력이 거의 없는 것으로 표현된다.
- 인터뷰에서 종종 제기되는 혁신 부족에 대한 또 다른 이유는 엔지니어링 혁신과 에너지의 많은 부분이 더 나은 제품을 만들기 위한 노력이 아니라 잘못된 관행(poor practices)이나 리스크 관리의 결함에서 야기되는 프로그램의 문제를 해결하는 데 소비된다는 것이다.
- 생산가능성 및 지원가능성(유지보수가능성)에 집중하는 적절하지 못한 설계가 있으며, 이는 불가결한 제품 진화 과정에서 비용과 품질 이슈를 야기한다.
- 완성된 설계나 아키텍처에 대한 안전(safety)과 보안(security)의 보장이 매우 강조되는 반면, 개발 초기부터 제품에 안전과 보안을 구축하는 것은 충분치 못하다.
- 제품의 부품을 하도급으로부터 공급받는 회사의 경우 효과적이지 못한 공급업체 감독으로 인해 일정이 어긋나고 재작업이 필요할 수 있다.
- 일정이 품질과 상충될 때, 프로그램 및 비즈니스 전략의 고려사항에서 기술적 결정에 대한 독립성이 부족하다. 이것은 기술적 리스크와 이슈를 보고함에 있어 개방적이고 정직한 커뮤니케이션 부족 및 독립적인 경로 부족과 연결 지을 수 있다.
- 엔지니어와 관리자가 의사 결정에 필요한 정보와 피드백을 얻지 못하고 있다. 때때로 그들은 가지고 있는 정보를 무시하는 경우도 있다. 피드백 루프가 존재하지 않거나 효과가 없다. 피드백이 종종 접수되지만 무시된다는 불만도 있다. *확증 편향(confirmation bias)*이라는 잘 알려진 현상이 있는데, 이는 사람들이 그들의 가설을 뒷받침하는 정보는 믿고 그렇지 않은 정보는 무시하는 경향이 있는 것을 말한다. 확증 편향은 듣고 싶지 않은 정보는 믿지 않거나 무시할 때 발생하는 *방어적 회피(defensive avoidance)*와 관련이 있다.
- 특히 제조업 또는 고객이 자주 제기하는 몇 가지 문제는 설계자와 제품 조립자간, 또는 설계자와 시스템 운영자 간의 관계와 관련이 있다. 흔한 문제는 부족한 작업 지침, 그리고 기계 기술자에서 설계 엔지니어로 올라오는 피드백의 부족과 관련이 있다.

위에서 언급한 문제의 수가 놀라워 보일 수 있지만 실제로는 산업의 시스템 엔지니어링 조직에서 매우 일반적이다.

우리는 인터뷰를 하는 동안 현재의 조직 문화에 대해서도 배우려고 노력하고 있다.

The Organizational Culture Required to Achieve the Goals



To discuss organizational culture, we need to start with a working definition. Edgar Shein's definition is used here:

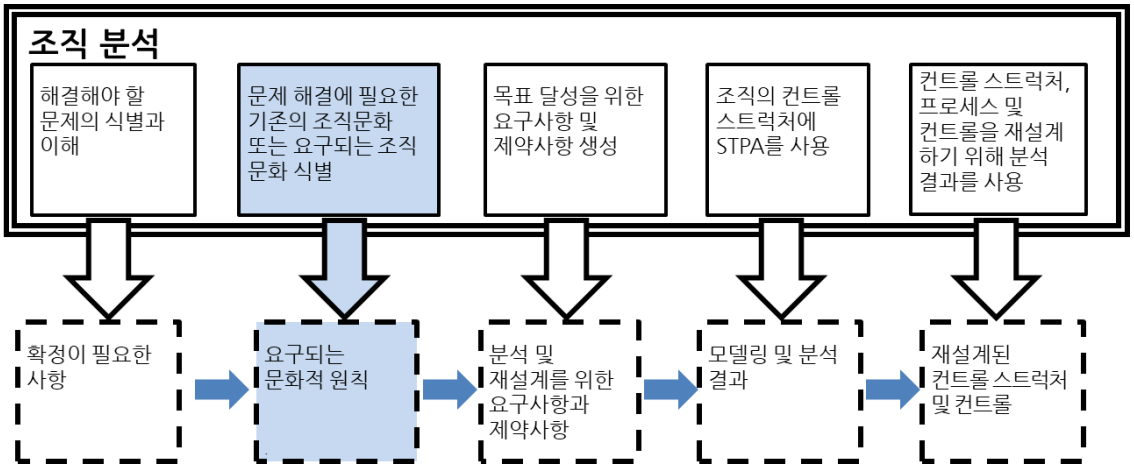
> *Organizational Culture is the shared values and deep cultural assumptions of a group or organization that provide the basis for decision making.*

To most effectively make important changes in an organization, it is necessary to change the culture to one that promotes those changes.

During the interviews, we define what we mean by organizational culture and ask the interviewees what cultural changes they think are needed in their organization. Their answers reflect the desired behavior they think the culture should promote. Some examples of the feedback we might get in interviews related to system engineering effectiveness and the organizational culture needed to improve it include:

- Less focus on schedule as the primary driver of decision making
- Ability to say "no" to customers and within the company. Marketing and sales answers are provided instead of engineering answers.
- Thinking beyond the immediate program and customer and across the programs
- More effective, clear communication
- Learning more from both failures and successes
- Spending more time up front in order to save time later
- More collaboration
- Valuing engineers and treating them with respect
- Believing that the work is not done until the customers can use the product and are happy
- Not being afraid to ask for help
- Openness and honesty about the product; don't shoot the messenger

목표를 달성하는 데 필요한 조직 문화



조직 문화를 논의하기 위해서는 먼저 업무 정의부터 시작해야 한다. 에드거 샤인(Edgar Shein)은 조직 문화를 다음과 같이 정의하였다.

> *조직 문화(organizational culture)*는 의사 결정의 기초를 제공하는 그룹 또는 조직의 공유된 가치이며 깊은 문화적 가정(assumptions)이다.

조직에서 가장 효과적으로 중요한 변화를 수행하기 위해서는, 이러한 변화를 촉진하는 문화로 바꿔야 한다.

우리는 인터뷰 중에 조직 문화가 의미하는 것을 정의하고 인터뷰 대상자에게 자신이 속한 조직이 어떤 문화적 변화를 필요로 하는지에 대한 생각을 물어보았다. 그들의 답변은 조직문화를 촉진하는 데 바람직하다고 생각하는 행동(behavior)을 반영한다. 시스템 엔지니어링 효과와 관련된 인터뷰에서 얻을 수 있는 몇 가지 피드백과 이를 개선하는 데 필요한 조직 문화의 예는 다음과 같다.

- 의사 결정의 주요 요인(driver)으로 일정에 덜 집중
- 고객에게 또는 회사 내에서 “아니요”라고 말할 수 있는 능력. 엔지니어링적인 답변 대신 마케팅 및 세일즈적인 답변 제공(역주: 기술적인 답변의 “아니요”는 문제를 회피하는 것으로 받아들여질 수 있음)
- 당면한 프로그램과 고객을 넘어, 프로그램들 전체에 걸친 사고(thinking)
- 보다 효과적이고 명확한 커뮤니케이션
- 실패와 성공 모두에서 더 많은 것을 학습
- 개발 후반부에 시간을 절약하기 위하여 많은 시간을 전반부에 투자
- 더 많은 협업
- 엔지니어를 소중히 여기며 존경심으로 대우
- 고객이 제품을 사용할 수 있고 만족할 때까지 작업이 완료되지 않았다고 믿는 것
- 도움을 요청하는 것을 두려워하지 않는 것
- 제품에 대한 개방성(openness) 및 정직성(honesty); 엉뚱한 사람에게 화풀이하지 말 것
- 국소적이 아닌, 더 큰 회사의 목표에 대해 생각하는 것
- “고통을 미루지” 않고 장기적인 것보다 단기적인 것에 우선순위를 두지 않도록 하는 것
- 리스크 완화 계획(risk mitigation plan)이 단순히 말로만 전달되고 그치는 것이 아니라, 적절한

- Thinking about the bigger company goals and not just locally
- Not "postponing the pain" and letting the short-term take priority over the long-term.
- Ensuring that the risk mitigation plans are properly resourced and not just communicated.

Clearly these are related to the problems identified in the interviews. The next step in our usual process is to take these behaviors, along with the concerns elicited in the interviews and listed in the previous section, and restate them as cultural principles (the shared organizational value system) that would provide the type of organization desired. The results are stated as a set of prevailing beliefs or values that, if implemented in the organizational design, would go a long way toward solving the identified problems. Here is an example set of cultural principles that might be created from the example interview results presented so far:

- Quality and Safety: Increasing quality and safety leads to decreasing cost and schedule and, in the long term, to increased profits. Safety is a key determinant in productivity and profitability.
- Planning: Doing good system engineering upfront saves costly rework later on. Good engineering should not require excessive effort and rework.
- Workforce and Skill Development: Effective workforce and skill development and utilization increases morale, decreases cost, and helps develop our talent base for the future.
- Technical Independence: Technical decisions will be better if they are made independent of program execution decisions. Thinking beyond the immediate program will lead to long-term benefits for the enterprise as a whole.
- Communication: Effective, clear, consistent, timely communication and the sharing of information are essential to achieving engineering goals.
- Learning and Improvement: Effective engineering requires continual learning and improvement, including learning from both success and failure. Building knowledge is as important as building products.
- Innovation: Investment in new knowledge and technology (innovation) drives growth and productivity.
- Collaboration: Successful engineering requires working across domain and structural boundaries, with all the applicable engineering disciplines included.
- Risk Management: Risk assessment and management are key to achieving performance goals.

Generating a list of desired cultural principles (values and beliefs) is only the starting point; it will require review, editing, and (when completed) buy-in from the leadership team and the engineers. The goal should be to start a discussion and begin a search for ways to change the company culture.

Once a consensus has been formed on the engineering culture that will help to achieve the goals, then the path forward in terms of requirements to implement that culture becomes more feasible to plan and implement.

자원이 투입됨을 보장하는 것

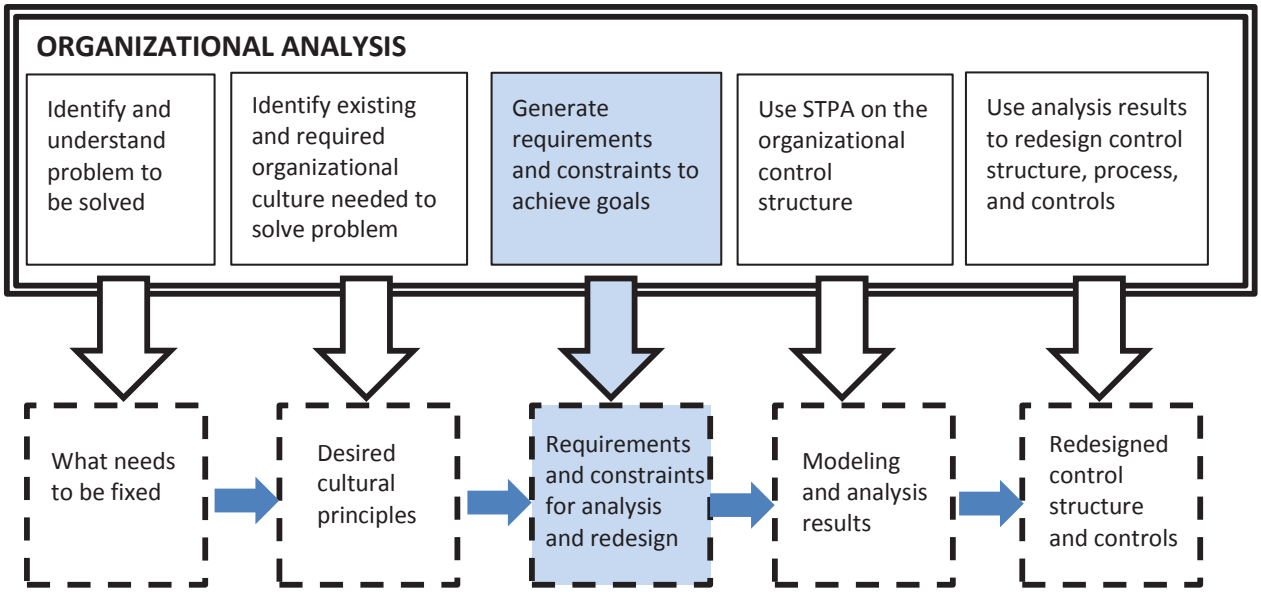
분명히 이러한 것들은 인터뷰에서 식별된 문제와 관련이 있다. 우리가 수행하는 일반적인 프로세스에서의 다음 단계는 인터뷰에서 제기되고 이전 섹션에서 열거된 우려사항을 고려하여 행동을 취하고 그것을 그들이 원하는 조직 유형을 제공하는 문화적 원칙(공유된 조직의 가치체계)으로 재작성하는 것이다. 그 결과는, 만약 조직의 설계에 구현하고자 한다면, 식별된 문제점을 해결하기 위해 장기적으로 따라야 하는 믿음 또는 가치들로 서술된다. 지금까지의 인터뷰 사례 결과에서 만들 수 있는 문화적 원칙(cultural principles)의 예시는 아래와 같다.

- 품질과 안전성(Quaility and Safety): 품질과 안전성을 높이면 비용과 일정이 줄어들고 장기적으로는 이익이 증가한다. 안전성은 생산성과 수익성을 결정하는 중요한 요소이다.
- 계획(Planning): 우수한 시스템 엔지니어링을 미리 수행하면 나중에 비용이 많이 들어가는 재작업을 줄일 수 있다. 우수한 엔지니어링은 과도한 노력과 재작업을 요구하지 않아야 한다.
- 인력 및 기술 개발(Workforce and Skill Development): 효과적인 인력과 기술의 개발 및 활용은 사기를 높이고 비용을 절감하며 미래를 위한 인재 기반을 구축하는 데 도움이 된다.
- 기술 독립성(Technical Independence): 기술 결정(technical decisions)은 프로그램 실행 결정(program execution decisions)과 독립적인 것이 더 좋다. 당장의 프로그램을 벗어나서 생각하면 기업 전체에 장기적인 혜택이 있을 수 있다.
- 커뮤니케이션(Communication): 효과적이고 명확하며 일관성 있고 시기적절한 커뮤니케이션 및 정보의 공유는 엔지니어링 목표를 달성하는 데 필수적이다.
- 학습 및 개선(Learning and Improvement): 효과적인 엔지니어링을 위해서는 성공과 실패 모두를 통해 학습하는 것을 포함하여 지속적인 학습 및 개선이 필요하다. 지식을 쌓는 것은 제품을 만드는 것만큼 중요하다.
- 혁신(Innovation): 새로운 지식과 기술에 대한 투자(혁신)는 성장과 생산성을 견인한다.
- 협업(Collaboration): 성공적인 엔지니어링을 위해서는 모든 가능한 엔지니어링 법칙을 활용하여, 도메인과 구조적인 경계를 넘어서는 협업이 필요하다.
- 리스크 관리(Risk Management): 리스크 평가 및 관리는 성과 목표 달성의 핵심이다.

원하는 문화적 원칙(가치와 신념) 목록을 작성하는 것은 시작에 불과하다. 그것을 위해서는 리더십 팀과 엔지니어로부터의 검토, 편집, 그리고 (완료 시) 승인이 필요하다. 목표는 토론하기 시작하고 회사의 문화를 변화시키는 방법을 모색하는 것이다.

목표를 달성하는 데 도움이 되는 엔지니어링 문화에 대한 합의가 형성되면, 그러한 문화를 실현하기 위한 요구사항 측면에서 앞으로 나아갈 길을 계획하고 실행하는 것이 더 쉬워진다.

Generating the Requirements and Constraints to be Implemented in an Effective Organizational Structure



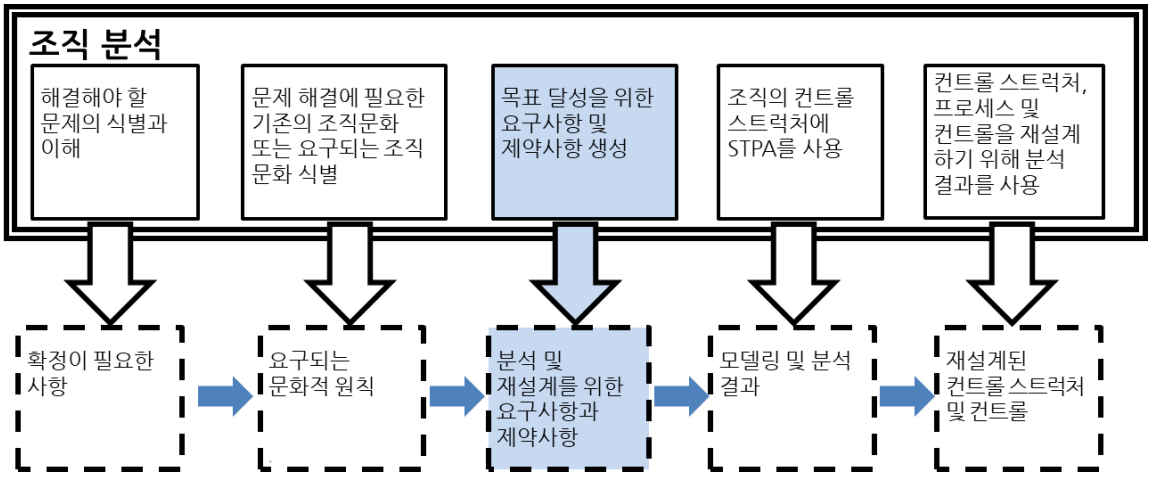
A critical step, after identifying the goals for any system engineering effort, is to create requirements. These are generated from the results of the problem identification process and the elicitation of the desired engineering culture. Here is a set of example requirements that might be generated using the sample analysis results so far in this section of the handbook. They are grouped under a restatement of each of the agreed-upon cultural goals.

Quality and Safety: Increasing quality and safety leads to decreasing cost and schedule. Safety is a key determinant in productivity and profitability.

- Each project shall have people with the appropriate skills and expertise needed to successfully achieve the project goals.
- Safety and security analysis shall start at the concept analysis stage and continue throughout the development process using safety and security analysis to inform design decisions as they are made.
- Gate (milestone) controls shall include a quality and safety analysis before they can be considered satisfied.
- Project managers shall be rewarded for first time quality[18].
- All product development shall emphasize producibility and supportability.
- Validation shall be emphasized at each stage of development. Attempts to bypass or decrease validation steps must be prevented.
- The company shall oversee the quality and safety of subcontractor products.

[18] The importance of incentivizing the "right" behavior is discussed in the chapter on Safety Management Systems. For example, giving managers bonuses for meeting timelines can lead to decreases in quality.

효과적인 조직 구조에서 구현해야 할 요구사항 및 제약사항의 생성



시스템 엔지니어링 목표를 식별한 이후, 매우 중요한 다음 단계는 요구사항(requirements)을 생성하는 것이다. 요구사항은 문제 식별 과정의 결과와 원하는 엔지니어링 문화의 도출을 통해 생성된다. 다음은 핸드북의 이 섹션에서 지금까지의 샘플 분석 결과를 사용하여 생성할 수 있는 요구사항의 예이다. 아래의 요구사항은 합의된 문화적 목표를 기준으로 그룹핑하였다.

품질과 안전성(Quality and Safety): 품질 및 안전성을 높이면 비용과 일정이 줄어든다. 안전성은 생산성과 수익성을 결정하는 중요한 요소이다.

- 각 프로젝트에는 프로젝트 목표를 성공적으로 달성하는 데 필요한 적절한 기술과 전문지식을 보유한 사람이 있어야 한다.
- 안전(safety)과 보안(security) 분석은 개념 분석 단계에서 시작하여 개발 프로세스 전반에 걸쳐 계속되어야 하며 안전 및 보안 분석을 통해 설계 결정이 내려진다.
- 게이트(마일스톤) 컨트롤의 만족 여부를 고려하기 전에 품질 및 안전 분석을 수행해야 한다.
- 최초 품질(first time quality)에 대해 프로젝트 관리자에게 보상해야 한다.[18]
- 모든 제품 개발은 생산가능성과 지원가능성을 강조해야 한다.
- 확인(validation)은 개발의 각 단계에서 강조되어야 한다. 확인 단계를 우회하거나 감소시키려는 시도를 방지해야 한다.
- 회사는 외주 업체 제품의 품질과 안전성을 감독해야 한다.

계획(Planning): 미리 시스템 엔지니어링을 잘 수행하면 나중에 비용이 많이 들어가는 재작업을 줄일 수 있다. 잘 수행한 엔지니어링은 과도한 노력과 재작업을 요구하지 않아야 한다.

- 제안서에서 미래의 성과를 예측하는 것은 고객의 목표와 같은 단순히 외부적인 기대치보다는, 적어도 부분적으로는 실행가능성(executability)과 엔지니어링 타당성(engineering feasibility)에 기

18 "올바른" 행동에 대해 인센티브하는 중요성은 안전 관리 시스템(Safety Management Systems) 장에서 논의한다. 예를 들어, 일정을 맞추는 것에 대해 관리자에게 보너스를 부여하면 품질이 저하될 수 있다.

Planning: Doing good system engineering upfront saves costly rework later on. Good engineering should not require excessive effort and rework.

- Predicting future performance in proposals shall be, at least in part, based on executability and engineering feasibility rather than simply external expectations such as customer goals.
- Detailed project planning and system analysis shall be done at the beginning of the project including detailed risk management and contingency planning; thorough understanding of customer needs, the problem to be solved by the product, and the environment in which the product will be used; and early activities to understand and mitigate risks such as simulations, prototyping, etc.
- When changes are required, rigorous change management shall be implemented.
- Subcontracting decisions shall be made with appropriate engineering input.
- Engineering plans shall include schedule and cost margins and/or avenues for flexible reallocation of resources to allow for handling inevitable 'technical unknowns."
- Plans shall be adaptable in the face of changing customer needs.

Workforce and Skill Development: Effective workforce and skill development and utilization increases morale, decreases cost, and helps develop our talent base for the future.

- Forecasting of skill needs shall be done, leading to providing appropriate training, mentoring, and growth experiences. System engineering in particular shall be trained and made a core piece of each project.
- When possible, leaders shall be selected on the basis of domain-specific experience. Such experience shall be developed for future leaders.
- Management shall take measures to ensure the workforce is appropriately rewarded and their contributions valued.

Technical Independence: Technical decisions will be better if they are made independent of program execution decisions.

- An independent authority (not the project manager) shall be responsible for determining whether a gate has been passed.
- Technical decisions shall be independent of programmatic and business strategy considerations when schedule is in conflict with quality. An independent technical authority shall be responsible for ensuring this independence.
- A higher authority than the project manager shall be responsible for resolving conflicts between the project manager and the independent technical authority.
- There shall be independent paths for elevating technical risk and other engineering issues.

Communication: Effective, clear, consistent, timely communication and the sharing of information are essential to achieving engineering goals.

- Better communication between Engineering and Business Acquisition shall be instituted, including feedback on the executability of proposals that are in preparation and the success or serious problems on contracts that have been completed (or canceled).

초해야 한다.

- 자세한 프로젝트 계획 및 시스템 분석은 프로젝트의 초기에 이루어져야 하며 상세한 리스크 관리(risk management)와 우발상황 계획(contingency planning)을 포함해야 한다. 이를 위해 고객의 니즈나 제품이 풀어야할 문제, 또는 제품이 사용되는 환경에 대한 철저한 이해가 필요하며 시뮬레이션, 프로토타이핑 등과 같은, 리스크를 이해하고 완화하기 위한 초기 활동이 필요하다.
- 변경이 필요한 경우 엄격한 변경 관리가 이행되어야 한다.
- 하도급 결정은 적절한 엔지니어링 입력(engineering input)을 통해 이루어져야 한다.
- 엔지니어링 계획에는 불가피한 '기술적으로 알 수 없는 사항(technical unknowns)'을 처리할 수 있도록 자원의 유연한 재할당을 위한 일정·비용 여유분(margin) 및/또는 방안이 포함되어야 한다.
- 변화하는 고객 요구에 대응하여 계획을 조정할 수 있어야 한다.

인력 및 기술 개발(Workforce and Skill Development): 효과적인 인력과 기술의 개발 및 활용은 사기를 높이고 비용을 절감하며 미래를 위한 인재 기반을 구축하는 데 도움이 된다.

- 기술 요구에 대한 예측이 이루어져야 하며 이는 적절한 훈련, 멘토링 및 성장 경험의 제공으로 이어진다. 특히 시스템 엔지니어링을 훈련해야 하며 시스템 엔지니어링은 각 프로젝트의 핵심 부분이 되어야 한다.
- 가능하면 도메인에 특화된 경험을 토대로 리더를 선정해야 한다. 그러한 경험은 미래의 리더를 위해 개발되어야 한다.
- 경영진은 직원에게 적절하게 보상하고 그들의 기여를 가치 있게 평가할 수 있는 방안을 마련해야 한다.

기술 독립성(Technical Independence): 기술 결정은 프로그램 실행 결정과 독립적인 것이 더 좋다.

- 독립적인 통제권자(프로젝트 관리자가 아닌)는 게이트 통과 여부(역주: 중간 결과가 만족되었는지 여부)를 결정할 책임이 있다.
- 일정이 품질과 상충될 때 기술 결정은 프로그램 및 비즈니스 전략 고려사항과는 독립적이어야 한다. 독립적인 기술 통제권자(technical authority)는 이러한 독립성을 보장할 책임이 있다.
- 프로젝트 관리자보다 높은 통제권자는 프로젝트 관리자와 독립적인 기술 통제권자간의 충돌을 해결할 책임이 있다.
- 기술적 리스크 및 다른 엔지니어링 문제를 상부에 보고하기 위한 독립적인 경로가 있어야 한다.

커뮤니케이션(Communication): 효과적이고 명확하며 일관성 있고 시기적절한 커뮤니케이션 및 정보 공유는 엔지니어링 목표를 달성하는 데 필수적이다.

- 엔지니어링 부서와 사업 조달 부서(Business Acquisition) 사이의 보다 나은 커뮤니케이션이 필요하며 여기에는 준비 중인 제안서의 실행가능성과 완료된(또는 취소된) 계약의 성공이나 심각한 문제점에 대한 피드백을 포함한다.
- 의사 결정권자가 적절한 결정을 내리는 데 필요한 정보를 얻을 수 있도록 피드백 및 정보 채널을 만들어야 한다.

학습 및 개선(Learning and Improvement): 효과적인 엔지니어링을 위해서는 성공과 실패 모두를 통

- Feedback and information channels shall be created to ensure that decision makers have the information they need to make appropriate decisions.

Learning and Improvement: Effective engineering requires continual learning and improvement, including learning from both success and failure. Building knowledge is as important as building products.

- *Post mortems* shall be produced by project managers in order to learn from both success and failure. Project managers shall be rewarded for honesty and thoroughness in these *post mortems*.[19]
- A project management information system shall be established that includes not just data from past projects but lessons learned from both successes and failures.

Innovation: Investment in new knowledge and technology (innovation) drives growth and productivity.

- Innovation, flexibility, and appropriate risk taking shall be encouraged and rewarded.
- The organization shall provide resources to produce innovative solutions to new technological challenges and opportunities and the inevitable technical issues that arise on most programs.
- Cultural barriers that prevent engineers from taking innovative risks shall be identified and removed.

Collaboration: Successful engineering requires working across domain and structural boundaries, with all the applicable engineering disciplines included.

- Working across domain and structural boundaries to promote collaboration shall be promoted and optimized.
- Work groups shall be established across domains and structural boundaries with all applicable domains included.

Risk Management: Risk assessment and management are key to achieving performance goals.

- Risk identification and assessments shall be identified by the most knowledgeable subject matter experts, clearly communicated to all of the affected personnel, and lead to a well-defined set of planned and resourced mitigations that are enacted in line with the program objectives and customer needs.
- Risk mitigation shall be planned from the start of the project based on the early risk assessment and any updates as experience is gained. Risk mitigation measures shall be funded and incorporated into the master schedule.
- Risks shall be clearly communicated to all affected personnel.

[19] Brutally honest project *post mortems* have been cited as one of the reasons for Microsoft's success. Time is provided for detailed analysis of the project results and leaders are expected to be honest about the reasons for any successes and failures. The military has a similar concept called After Action Reviews. Learning from one's own experience (and that of others) is a requirement for future success. Not rewarding people for honestly evaluating their own performance and sharing lessons learned may lead to new people being promoted that may make the same mistakes and thus inhibiting learning from experience.

해 학습하는 것을 포함하여 지속적인 학습 및 개선이 필요하다. 지식을 쌓는 것은 제품을 만드는 것만큼 중요하다.

- 프로젝트 관리자는 성공과 실패로부터 배우기 위하여 *사후 분석(Post mortems)*을 도입해야 하며, *사후 분석*에서의 정직성과 철저함에 대해 프로젝트 관리자에게 보상해야 한다.[19]
- 과거 프로젝트의 데이터뿐만 아니라 성공과 실패로부터 얻은 교훈을 포함하는 프로젝트 관리 정보 시스템(project management information system)이 구축되어야 한다.

혁신(Innovation): 새로운 지식과 기술에 대한 투자(혁신)는 성장을 촉진하고 생산성을 견인한다.

- 혁신, 유연성, 적절한 위험 감수(risk taking)를 장려하고 그에 대해 보상해야 한다.
- 조직은 새로운 기술적 도전과 기회, 그리고 대부분의 프로그램에서 발생하는 필연적인 기술적 이슈에 대한 혁신적인 솔루션이 만들어지는 데 필요한 자원을 제공해야 한다.
- 엔지니어가 혁신적인 위험을 감수하지 못하게 하는 문화적 장벽을 식별하고 제거해야 한다.

협업(Collaboration): 성공적인 엔지니어링을 위해서는 모든 가능한 엔지니어링 법칙을 활용하여 도메인과 구조적인 경계를 넘어서는 협업이 필요하다.

- 협업을 촉진하기 위해 도메인 및 구조적 경계를 넘어서서 작업하는 것이 장려되고 최적화되어야 한다.
- 작업 그룹을 만들 때는 모든 적용가능한 도메인을 포함하여 도메인과 구조적 경계를 아우르도록 구성해야 한다.

리스크 관리(Risk Management): 리스크 평가 및 관리는 성과 목표 달성의 핵심이다.

- 리스크 식별 및 평가(assessments)는 그 주제의 가장 지식이 풍부한 전문가(subject matter experts)가 확인해야 하며 관련된 모든 직원에게 명확하게 전달되어야 한다. 이는 프로그램 목표와 고객의 요구에 맞추어 시행되는, 계획되고 자원이 확보된 완화조치의 정의로 이어진다.
- 리스크 완화조치(mitigation)는 초기의 리스크 평가 및 경험에서 알게 된 모든 새로운 사실(updates)에 기반하여 프로젝트 시작부터 계획되어야 한다. 리스크 완화대책에는 자금이 지원되어야 하며 기본 일정계획(master schedule)에 포함되어야 한다.
- 리스크는 영향을 받는 모든 직원에게 명확하게 전달되어야 한다.
- 리스크 관리 계획에는 “알려지지 않은 불확실한 사항(unknown unknowns)”을 허용하기에 충분한 비용과 일정의 여유분(margin)이 포함되어야 한다.
- 리스크 관리에는 관리자가 현실적인 리스크를 무시할 수 없도록 적절한 컨트롤이 필요하다.
- 리스크 관리를 위한 적절한 자원 제공이 필요하다.

이것들은 조직의 토대(foundation)로 작용하기 위해 합의된 문화적 원칙으로부터 도출될 수 있는 요구사항의 단순한 예이다. 이외의 다른 것들도 얼마든지 있을 수 있다.

---

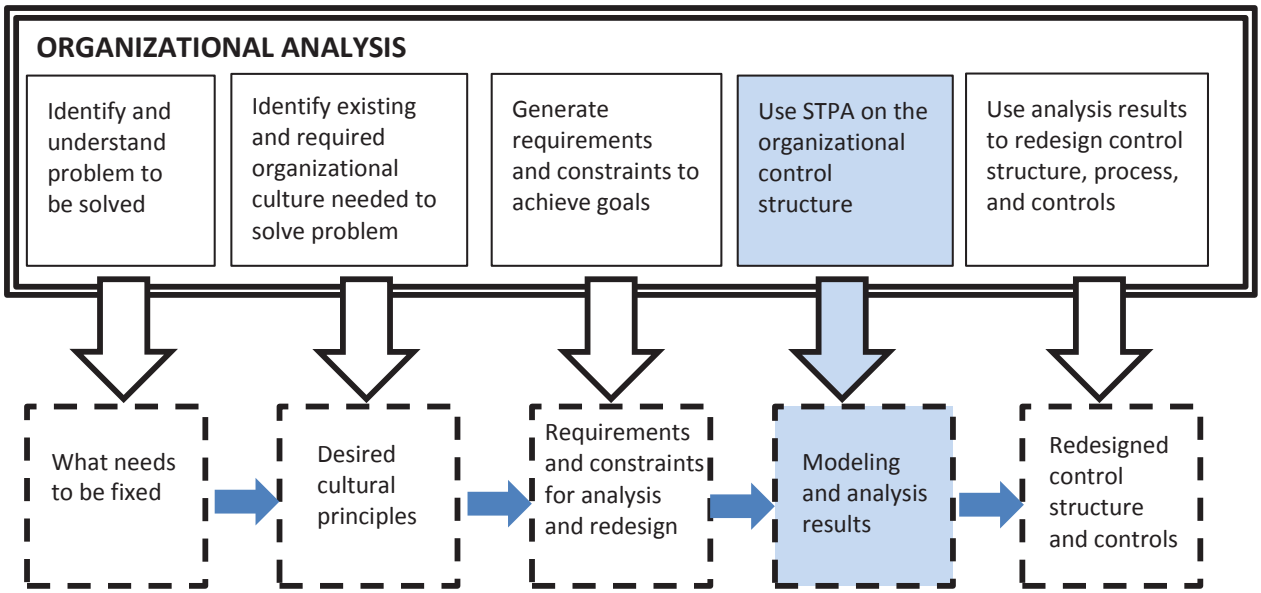
19 너무나도 솔직한 프로젝트 *사후분석(post mortems)*은 Microsoft의 성공 이유 중 하나로 언급된다. 프로젝트 결과의 상세한 분석을 위해 시간을 투자하며 리더는 성공과 실패의 이유를 정직하게 말한다. 군대에도 사후검토(After Action Reviews)라는 유사한 개념이 있다. 자신의 경험(그리고 다른 사람들의 경험)을 통해 배우는 것은 미래의 성공을 위한 필수조건이다. 정직하게 자신의 성과를 평가하고 배운 교훈을 공유하는 사람들에게 보상하지 않게 되면 새로운 사람이 승진하여 같은 실수를 반복하게 되고 경험에서 학습하는 것을 막는 결과를 초래한다.

- Risk management plans shall include cost and schedule margins sufficient to allow for "unknown unknowns."
- Risk management shall have appropriate controls such that realistic risks cannot be ignored by managers.
- Risk management shall be properly resourced.

These are simply examples of requirements that might be generated from the agreed upon cultural principles to act as the foundation of the organization. Others are possible.

Once the requirements have been agreed upon, the actual modeling and analysis can begin.

STPA analysis of the organizational structure



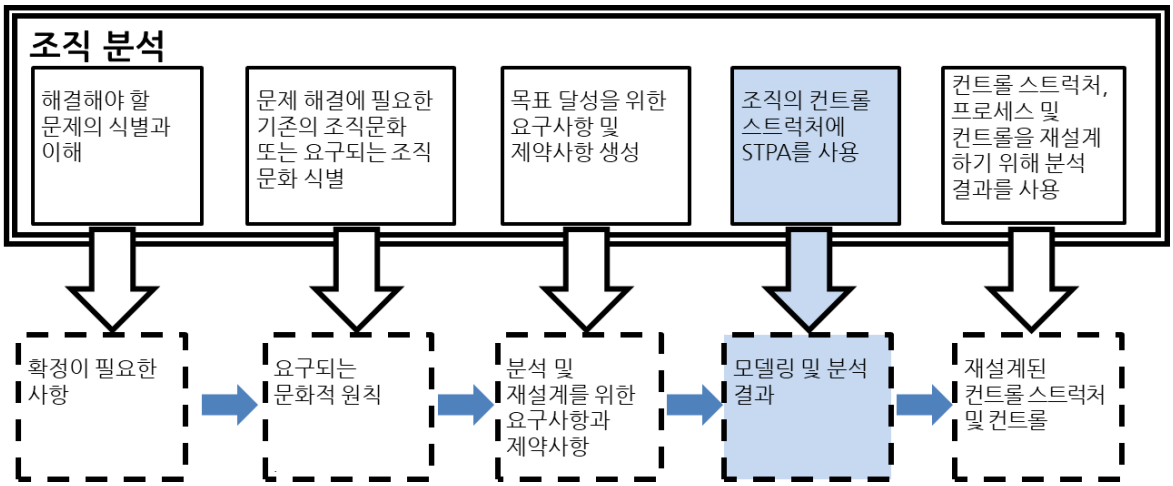
The next step in the process is to model the current system engineering control structure, to identify where the identified requirements are currently implemented (if at all) or at least where responsibility is assigned for these activities, and to determine why the concerns identified (in phase 1) are not already being adequately addressed and the requirements implemented. The results can be used to suggest changes that will better implement the requirements in the organizational design.

When the modeling is complete or at least at the point where the general control structure is understood, then the hazard analysis on the structure can begin. The hazard analysis includes the generation of leading indicators to provide early identification of assumptions about system engineering in the organization that are not true or that may no longer be true as the organization and environmental factors change over time. These leading indicators should also provide a way to identify how well the concerns and goals are being addressed.

We have found during the modeling and analysis process that people often know the solutions to their problems, but are so overwhelmed with day-to-day schedules and problems that they don't have the time to implement them. We have also found that use of the hierarchical control structure model is invaluable in helping with communication within the solution-generating groups and also with generation and evaluation of potential redesign solutions. Having a common model to guide discussion

요구사항이 일단 합의되면, 실제 모델링과 분석을 시작할 수 있다.

조직 구조의 STPA 분석



프로세스의 다음 단계는 현재 시스템 엔지니어링 컨트롤 스트럭처를 모델링하고, 식별된 요구사항이 현재 어디에 구현됐는지 식별하거나 (만약 그렇다면) 최소한 이러한 활동들의 책임이 어디에 할당됐는지 식별하며, (1 단계에서) 도출된 우려사항들이 적절하게 다루어지지 않은 채로 요구사항이 구현된 경우의 이유를 밝히는 것이다. 이러한 결과는 조직의 설계에서의 요구사항을 보다 잘 구현할 수 있는 변화를 제안하는 데 사용될 수 있다.

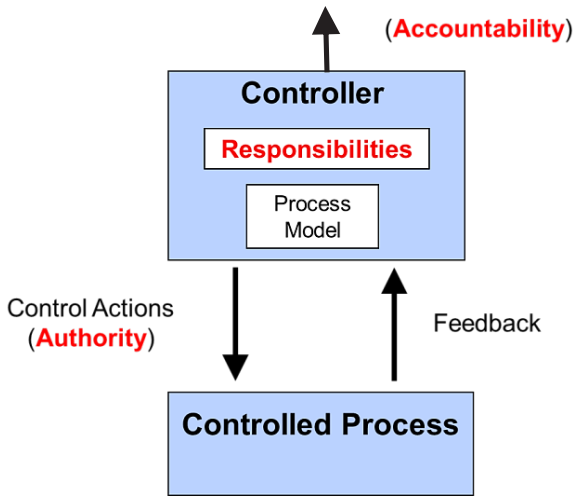
모델링이 완료되었을 때 또는 적어도 일반적인 컨트롤 스트럭처가 이해되는 시점에 조직 구조에 대한 위험 분석을 시작할 수 있다. 위험 분석은 조직 내 시스템 엔지니어링 가정에 대한 조기 식별을 가능케 하는 선행지표(leading indicator)의 생성을 포함한다. 가정은 사실이 아닐 수도 있고 시간이 지나면서 조직과 환경적 요소가 변함에 따라 더 이상 사실이 아니게 될 수도 있기 때문이다. 이러한 선행지표는 우려사항과 목표가 얼마나 잘 처리되고 있는지를 식별할 수 있는 방법도 제공해야 한다.

모델링 및 분석 과정에서 우리는 사람들이 종종 자신의 문제에 대한 해결책을 알고 있지만, 그날 그날의 스케줄과 그러한 것을 구현할 시간이 없다는 문제로 너무 압박당하고 있음을 알 수 있었다. 또한, 계층적인 컨트롤 스트럭처 모델을 사용하는 것이 솔루션 개발 그룹 내의 의사소통을 돕고 잠재적인 재설계 솔루션을 생성 및 평가하는 것에 매우 도움이 된다는 것도 알게 되었다. 토론과 문제 해결을 가이드하는 공통된 모델을 가지고 있으면 그것만으로도 문제 해결 방법을 찾아내는 데 매우 도움이 된다. 예를 들어 때때로 정교한 분석도 없이 일부 개체(entity)가 그들의 컨트롤(관리) 액션을 가이드하는 어떤 피드백도 없이 다른 개체를 컨트롤하려고 하는 것을 볼 수 있다. 효과적인 피드백을 추가하면 현재의 문제를 해결하는 데 많은 도움이 될 수 있다.

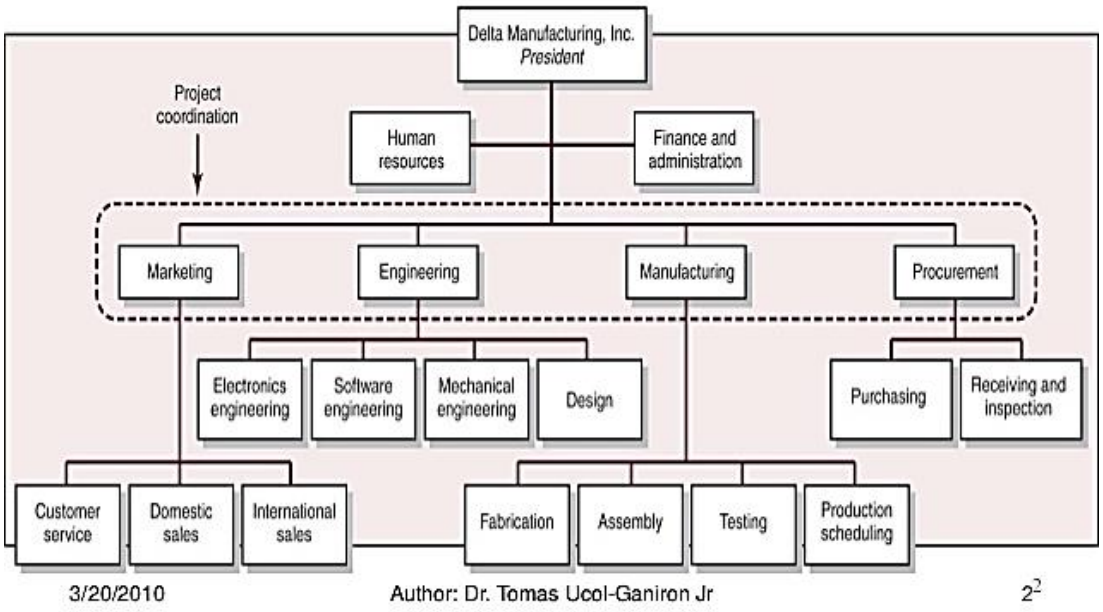
다음 그림은 이 핸드북의 앞에서 보여주었던 일반적인 루프 스트럭처(Loop Structure)를 보여주는데 여기에 책임(responsibility), 통제권한(authority) 및 의무(accountability)라는 표준 관리 요구사항을 추가한 것이다.

and problem solving is by itself very helpful in finding solutions to problems. Sometimes, for example, without any sophisticated analysis, it can be observed that some entity is trying to control another without any feedback to guide their control (management) actions. Adding effective feedback may go a long way toward solving the current problems.

The following shows the generic loop structure shown previously in this handbook but augmented with the standard management requirements of responsibility, authority, and accountability.



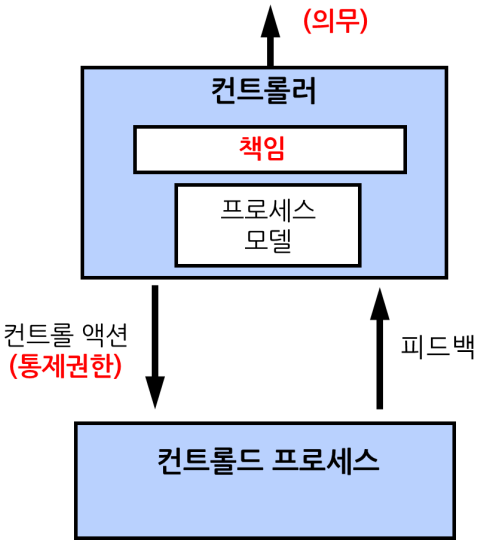
Below is shown an example system engineering structure that might result from the modeling effort.



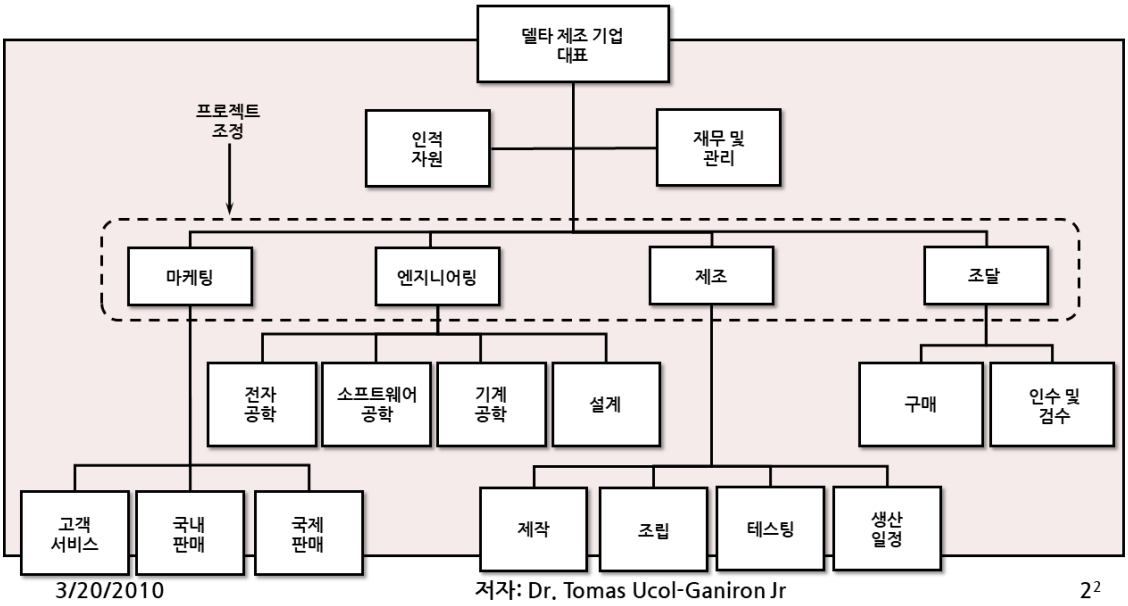
Notice that project coordination is simply assumed to occur in among four same-level controllers in this example without responsibility being assigned to any individual. It is unlikely to be an appropriate responsibility of the president of the company.

The model, of course, would include specification of the responsibilities, authority, and accountability for each function in the structure. For example, assigned responsibilities for general controllers in a product development company might include:

Human resources:



아래 그림은 모델링 활동의 결과로 나올 수 있는 시스템 엔지니어링 스트럭처의 예시이다.



이 예시에서는 프로젝트 조정(project coordination)이 어떤 개인에게 책임이 부여되지 않고 4개의 동일한 수준의 컨트롤러에서 발생한다고 단순하게 가정한다. 회사 대표(president)의 책임이 되는 것은 적절하지 않은 것 같다.

물론 모델은 스트럭처의 각 기능에 대한 책임(responsibility), 통제권한(authority) 및 의무(accountability)에 대한 명세를 포함한다. 예를 들어, 제품 개발 회사의 일반적인 컨트롤러에 부여된 책임은 다음과 같다.

- 인적 자원(Human resources):

- Training, hiring, workforce and skill development
- Project management
  - Project planning and concept development
  - Technical decision making
  - Integration of engineering efforts, communication within projects
  - Risk management and contingency management
  - Milestone and schedule management
  - Budget/resource management
  - Managing subcontracts
  - Team dynamics/communication
- System engineering
  - Implementing technical decisions
  - Specialty engineers
  - Implementing technical decisions, project requirements in component design
- Business development
  - Bringing in new business
  - Bidding on projects
- Program management
  - Program/Project risk management
  - Overseeing project progress and success, workforce management
  - Managing subcontractor relationships, supply chain
- Executive Management
  - Implementing technology/process improvements
  - Business Risk management
  - Forecasting skill and engineering needs in the future
  - Ensuring current and future skill and workforce needs are satisfied
  - Workforce “management”
- Research and Engineering Improvement
  - Innovation, new technology/process improvements
  - Identifying future business and technology trends, skill needs
- Quality assurance and system safety
  - Oversee product quality
  - Evaluate product safety

Appropriate next steps are (1) informal review sessions to ensure the correctness of the control structure as well as to identify weaknesses, (2) analysis of the control structure model using traceability from requirements to the control structure responsibilities, and (3) application of formal STPA analysis where useful. The goal is to identify where and how each of the requirements (specified previously) are being realized in the organization control structure and any weaknesses in the control structure with respect to these requirements. Examples are provided below.

  - 훈련, 고용, 인력 및 기술 개발

- 프로젝트 관리(Project management)
  - 프로젝트 계획 및 개념(concept) 개발
  - 기술적 의사 결정
  - 프로젝트에서의 엔지니어링 활동 및 커뮤니케이션 통합
  - 리스크 관리 및 우발상황(contingency) 관리
  - 마일스톤 및 일정 관리
  - 예산/자원 관리
  - 외주 관리
  - 팀 활력/의사소통
- 시스템 엔지니어링(System engineering)
  - 기술적 결정 구현
  - 전문 엔지니어
  - 컴포넌트 설계에서의 기술적 결정 및 프로젝트 요구사항의 구현
- 사업 개발(Business development)
  - 새로운 사업 유치
  - 프로젝트 입찰
- 프로그램 관리(Program management)
  - 프로그램/프로젝트 리스크 관리
  - 프로젝트 진행과 성공, 인력 관리에 대한 감독
  - 외주 업체 관계, 공급망 관리
- 경영진(Executive Management)
  - 기술/프로세스 개선 구현
  - 비즈니스 리스크 관리
  - 미래의 기술 및 엔지니어링 수요 예측
  - 현재와 미래의 기술 및 인력 요구가 충족되도록 보장
  - 인력 “관리”
- 연구 및 엔지니어링 개선(Research and Engineering Improvement)
  - 혁신, 새로운 기술/프로세스 개선
  - 미래 비즈니스 및 기술 동향, 기술 요구 파악
- 품질 보증 및 시스템 안전(Quality assurance and system safety)
  - 제품 품질 감독
  - 제품 안전 평가(evaluation)

적절한 다음 단계는 (1) 컨트롤 스트럭처의 약점을 식별하고 정확성(correctness)을 보장하기 위한 비공식 검토 세션(informal review session), (2) 요구사항에서부터 컨트롤 스트럭처의 책임(responsibilities)까지의 추적성을 사용한 컨트롤 스트럭처 모델 분석, (3) 유용한 경우 공식적인 STPA 분석(formal STPA analysis)의 적용이다. 목표는, 조직의 컨트롤 스트럭처에서 각 요구사항(이전에 명세된)이 어디에, 어떻게 실현되었는지를 식별하고, 이러한 요구사항과 관련하여 컨트롤 스트럭처의 약점을 파악하는 것이다. 아래에 사례를 소개한다.

The first step is to validate the model through review by experts on the current organizational design. Multiple reviews may be needed as reviewers may only be familiar with part of the structure.

Once the overall control structure is validated, a next step is to examine the control structure with respect to the specific problems identified in the interviews as well as to analyze the control structure design to identify general changes that may be needed to improve decision making. Often there are things that stand out immediately, such as the complexity of the structure; multiple people controlling the same process or having the same overlapping responsibility; lack of appropriate or independent oversight of decision making; or missing feedback paths (i.e., people may be making decisions without all the information needed or with inaccurate information).

To find further weaknesses, tracing from the requirements to the control structure can be done. The table below shows a few examples of potential tracing results for the requirements included above.

Requirement	Control Structure Component(s)	Weakness Identified
Risk Management	Project Management, Program Management	• Cost and schedule margins are often not sufficient to deal with contingencies • Risk identification sometimes is incomplete and important risks ignored
Technical Independence	Project Management, Program Management	• Technical decisions are not independent of programmatic and business strategy nor are they independent from the project manager
Quality and Safety	Quality assurance, project management, program management, executive management	• Safety is assurance-oriented and is not performed by or in close collaboration with the development team. • Much of quality assurance is “checklist” and perfunctory. • Gate (milestone) controls do not include a quality and safety analysis • Project managers are more highly rewarded for schedule and budget performance than for quality and safety. • Producibility and supportability are not emphasized • Contractors are minimally supervised with respect to quality and safety

The weaknesses identified will provide guidance for the redesign activities to improve the satisfaction of the requirements. Some improvements will require few structural changes. For example,

첫 번째 단계는 현재 조직 설계에 대한 전문가의 검토(review)를 통해 모델을 검증하는 것이다. 검토자가 스트럭처의 일부만 잘 아는 경우에는 여러 번의 검토가 필요할 수도 있다.

컨트롤 스트럭처가 전반적으로 검토되면, 다음 단계에서는 컨트롤 스트럭처 설계 분석을 통해 의사 결정을 개선하는 데 필요한 일반적인 변경 사항을 파악하고, 인터뷰에서 식별된 특정 문제에 대해 컨트롤 스트럭처를 검토하는 것이다. 종종 스트럭처의 복잡도와 같이 바로 눈에 띄는 것들이 있는데, 여러 사람이 동일한 프로세스를 제어하거나 오버랩핑된 동일한 책임을 가지는 경우 의사 결정에 대한 적절하거나 독립적인 감독의 부족, 누락된 피드백 경로(즉, 필요한 모든 정보가 없는 상태에서 혹은 부정확한 정보를 가진 상태에서 사람들이 의사 결정을 내릴 수 있다) 등이 그것이다.

더 많은 약점을 찾기 위해, 요구사항에서 컨트롤 스트럭처까지를 추적할 수도 있다. 아래 표는 위에 포함된 요구사항에 대한 잠재적인 추적 결과의 몇 가지 예를 보여준다.

요구사항	컨트롤 스트럭처 컴포넌트(들)	식별된 약점(weakness)
리스크 관리 (Risk Management)	프로젝트 관리, 프로그램 관리	• 긴급사태를 처리하기에 비용과 일정의 여유(margin)가 종종 충분하지 않다. • 리스크 식별이 때때로 불완전하고 중요한 리스크가 무시된다.
기술 독립성 (Technical Independence)	프로젝트 관리, 프로그램 관리	• 기술 결정이 프로그램 및 비즈니스 전략으로부터 독립적이 아니며 프로젝트 관리자로부터도 독립적이지 않다.
품질 및 안전성 (Quality and Safety)	품질 보증, 프로젝트 관리, 프로그램 관리, 경영진 관리	• 안전이 보증-지향적이며(assurance-oriented), 개발팀에 의해 또는 개발팀과 긴밀히 협력하여 수행되지 않는다. • 품질 보증의 대부분이 “체크리스트”이며 형식적이다. • 게이트(마일스톤) 제어에 품질 및 안전 분석이 포함되지 않는다. • 프로젝트 관리자가 품질과 안전보다 일정 및 예산 성과에 대해 더 높은 보상을 받는다. • 생산가능성과 지원가능성이 강조되지 않는다 • 품질 및 안전 측면에서 외주 계약업체를 최소한으로 감독한다.

식별된 약점(weakness)은 요구사항의 만족도를 높이기 위한 재설계 활동에 대한 지침을 제공할 것이다. 일부 개선사항은 구조적인 변경이 거의 필요하지 않다. 예를 들어, 품질 및 안전 측면에서 계약업체를 부적절하게 감독하는 경우였다면 계약업체의 관리자에게 책임을 단순히 할당하고 품질 보증과 안전에 대한 도움을 제공하기만 하면 된다. 보다 광범위한 구조적 변경이 필요한 수정도 있을 수 있다. 예를 들어, 그러한 계약업체 감독을 어떻게 하는지에 대한 정책이 부적절하거나 누락되어 있다면, 정책을 수립하고 변경하는 노력이 필요할 것이다.

보다 복잡한 변화의 예로 기술 독립성을 위해서는 충돌 해결을 위한 새로운 커뮤니케이션 링크와 의사 결정 절차뿐만 아니라 독립된 기술적 통제권한(independent technical authority)[20] 설정과 같은 새로운 관리 접근법이 필요하다. 독립성은 시간이 지남에 따라 퇴색되는 경향이 있으므로 이를 위해 선행지표(leading indicator)를 가지는 것이 적절하다(선행지표에 대해서는 다음 절 참조).

20 美 해군 SUBSAFE 프로그램의 독립된 기술적 통제권한은 지난 50년 이상 이 프로그램에 포함된 사고를 제거하는 데 매우 효과적이었다. 여기에 대한 더 많은 내용은 Nancy Leveson의 Engineering a Safer World (2012)의 14장과 본 핸드북의 다음 장(안전 관리 시스템, Safety Management Systems)에 있다.

clearly the inadequate supervision of contractors with respect to quality and safety requires only a simple assignment of responsibilities to the manager of the contractors and probably assistance from quality assurance and safety. Other fixes may require more extensive structural changes. If policy for how such contractor supervision should be done is inadequate or missing, for example, then some effort on policy creation and changes will be needed.

As an example of a more complex change, technical independence will require new management approaches, such as setting up an independent technical authority[20] as well as new communication links and a decision-making process for resolving conflicts. Independence tends to fade over time, so leading indicators would also be appropriate here (see the next section on leading indicators).

A final example is that separating safety from engineering decision-making tends to make it an after-the-fact assurance or compliance effort with almost no impact on the design effort. Finding safety or quality problems late is very expensive to fix and separating safety responsibilities from basic engineering is a mistake, although very common. Determining the best strategy to optimize the interaction could require extensive changes in responsibilities, communication, and structure. For example, there might be a safety engineering component in project engineering and a separate one in quality assurance, each having very different responsibilities and performing different activities.

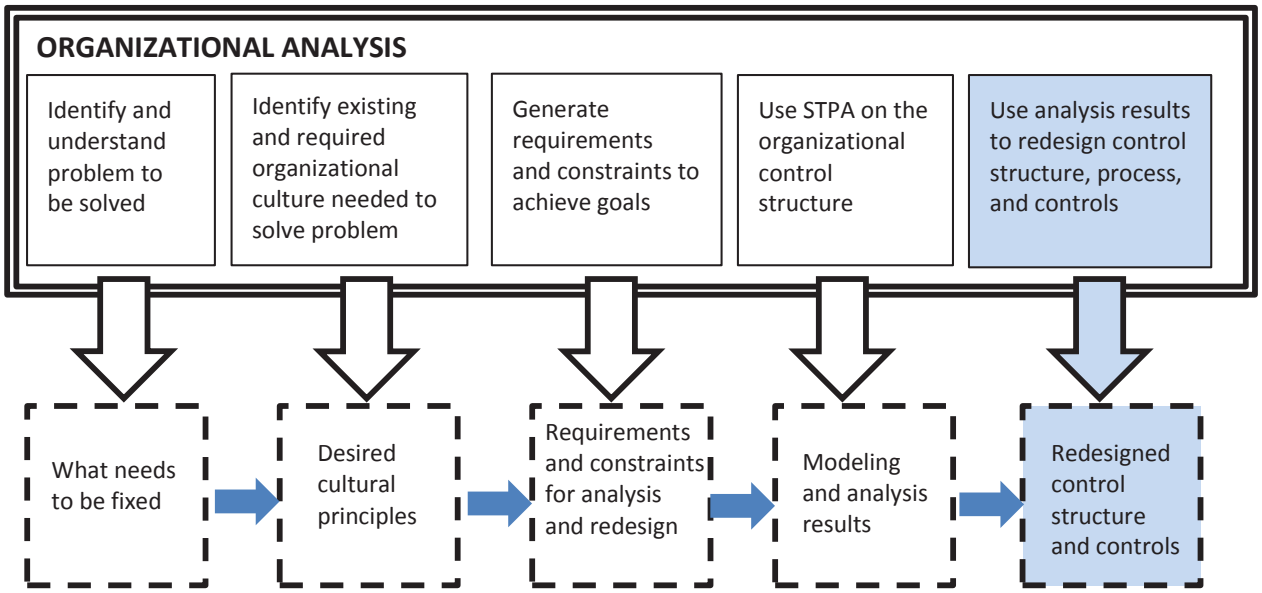
In some cases, it may be appropriate and useful to apply a full STPA causal analysis on the control structure or to parts of it. As an example, it may not be clear exactly why risk management has not been adequate as most companies put effort into performing this important task. It might be useful to identify all the potential causes, such as lack of information about what the risks actually are (flawed mental models), a culture that emphasizes positivity and not potential risks, inadequate training in risk assessment, poor procedures, pressure from management to downgrade risks on important projects, etc. After generating all the potential causes, a study might be initiated to determine the operative causes in this organization and what changes might be useful to eliminate or control them. While some hypotheses could be formulated in an *ad hoc* manner, using the step-by-step process in STPA will improve completeness and reduce the possibility that important causal factors are missed. There may be different reasons for potential deficiencies in each type of risk management control action in the organization. For example, not providing adequate emphasis on identifying some types of risks may involve different causal factors than not providing adequate contingency plans. We have found that using STPA to identify risks is much more effective than the usual brainstorming process by experienced employees.

[20] The Independent Technical Authority in the U.S. Navy SUBSAFE program has been extremely effective in eliminating accidents covered by the program for the past 50+ years. You can read more about it in Nancy Leveson, *Engineering a Safer World*, 2012, chapter 14 and in the next chapter on Safety Management Systems.

마지막 예는 엔지니어링 의사 결정에서 안전을 분리하게 되면 설계 활동에는 거의 영향을 미치지 못하고, 단순히 사후보증(after-the-fact assurance)이나 형식적인 준수에 머무르게 되는 경향이 있다는 것이다. 안전이나 품질 문제를 늦게 발견하게 되면 비용이 매우 많이 든다. 기본적인 엔지니어링에서 안전 책임을 분리하는 것은 잘못된 것이나, 그럼에도 불구하고 매우 빈번하다. 상호작용을 최적화하기 위한 최선의 전략을 결정하기 위해서는 책임(responsibility), 의사소통(communication) 그리고 구조(structure)를 크게 변경해야 할 수 있다. 예를 들어, 프로젝트 엔지니어링에는 안전 엔지니어링 컴포넌트가 있을 수 있고 품질 보증에는 별도의 컴포넌트가 있을 수 있으며, 각 컴포넌트는 매우 다른 책임을 가지며 다른 활동을 수행한다.

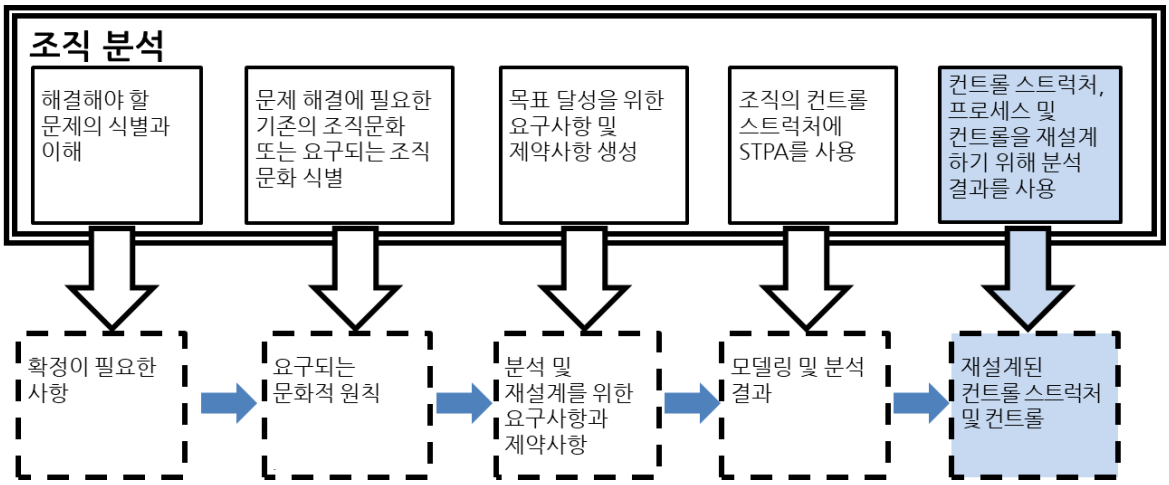
어떤 경우에는 컨트롤 스트럭처나 그 일부에 전체 STPA 원인 분석(full STPA causal analysis)을 적용하는 것이 적절하고 유용할 수 있다. 예를 들어, 대부분 회사에서 리스크 관리 활동을 중요한 작업으로 여기고 이를 위한 활동을 수행하고 있지만, 리스크 관리가 적절하지 이루어지지 못한 이유가 명확하지 않을 수 있다. 이를 위해서는 모든 잠재적인 원인을 식별하는 것이 바람직하다. 잠재적 원인의 예로는 리스크가 실제로 무엇인지에 대한 정보 부족(결함 있는 멘탈 모델), 잠재적인 리스크가 아니라 확실한 것을 강조하는 문화, 리스크 평가(assessment)에 대한 부적절한 훈련, 열악한 절차, 중요한 프로젝트의 리스크를 낮추라는 경영진의 압력 등이 있다. 모든 잠재적인 원인을 생성한 후에는 이 조직에서의 중요한 원인과 이를 제거하거나 제어하기 위해 어떤 변화가 유용한지를 결정하는 연구를 시작할 수 있다. 일부 가설은 즉흥적으로(ad hoc) 구성될 수도 있겠지만, STPA에서는 단계별 프로세스를 사용하여 완성도를 높이고 중요한 원인 요소가 누락되는 가능성을 줄이게 된다. 조직의 리스크 관리 컨트롤 액션의 각 유형에 대한 잠재적 결함(potential deficiencies)에는 서로 다른 이유가 있을 수 있다. 예를 들어 특정 유형 리스크 식별에 적절한 중점을 두지 않는다는 것은 충분한 우발상황 계획(contingency plans)을 제공하지 않는 것과는 다른, 별도의 원인 요소를 포함하고 있을 수 있다. 우리는 STPA를 사용하여 위험을 식별하는 것이 경험 많은 직원의 일반적인 브레인스토밍 보다 훨씬 효과적이라는 것을 발견했다.

Using the Results in an Organizational Redesign Process



A final step in the process is to use the analysis results in a redesign and risk management process for the control structure as a whole. Part of the risk management in the design of controls involves a leading indicator program. Applying STPA to the control structure or parts of it can be helpful in identifying leading indicators of increasing risk, which is discussed in the next chapter.

조직 재설계 프로세스에 분석결과 활용



이 프로세스의 마지막 단계는 분석 결과를 컨트롤 스트럭처 전체에 대한 재설계 및 리스크 관리 절차에 사용하는 것이다. 컨트롤 설계에서의 리스크 관리에는 선행지표(leading indicator) 프로그램이 포함된다. 컨트롤 스트럭처나 그 일부에 STPA를 적용하면 증가하는 리스크에 대한 이러한 선행지표를 식별하는 데 도움이 될 수 있으며, 이는 다음 장에서 논의한다.

Chapter 6: Identifying Leading Indicators Using STPA[21]

Nancy Leveson

Risk management is a critical part of any program and the programmatic and organizational risks need to be carefully identified and managed. Risk assessment is usually performed in an ad hoc fashion, with a group of experts brainstorming what appear to be the risks in a program and then assigning risk levels to them, often rather arbitrarily. STPA provides a structured process on which to base a risk management program.

When leading indicators are used for risk management, they are usually also created in an ad hoc manner. Using a more formal, structured system-thinking process may be more effective, just as it is more effective when considering product safety and other emergent properties as described earlier in this handbook. Using STPA to create leading indicators also has the advantage that the resulting indicators are directly traceable to hazards and accidents. This chapter describes such a structured approach to identifying leading indicators that includes using the results from STPA and other results of the overall system engineering process. The topics covered include:

- What is a leading indicator?
- What do people use now for leading indicators?
- What are assumption-based leading indicators?
- How to
  - ✓ Identify appropriate and effective leading indicators using STAMP and STPA
  - ✓ Generate assumptions on which to base leading indicators
  - ✓ Use the assumptions to create an assumption-based leading indicator program
  - ✓ Integrate leading indicators into your risk management program
- Feasibility and final thoughts

What is a Leading Indicator?

Leading indicators are used to identify the potential for an accident before it occurs so measures can be taken to prevent it. They are based on the assumption that major accidents are not due to a unique set of random, proximal events. Instead, accidents result from

> *The migration of an organization to a state of increasing risk over time*
> *as safeguards and controls are relaxed*
> *due to conflicting goals and tradeoffs and reduced perceptions of risk*
> *leading to more risky behavior.*

This assumption implies that the march toward a major accident takes place over time and, therefore, the possibility exists to detect this dangerous path and intervene. A leading indicator is a signal that intervention is necessary.

[21] This chapter, like the others in this handbook, is written as a practical "how to" guide including some introductory explanatory information. For a more complete, technical treatment with lots of references, see Nancy G. Leveson, A systems approach to risk management through leading safety indicators, *Reliability Engineering and System Safety*, Elsevier Publishers, 2014.

6장: STPA를 사용한 선행지표 식별[21]

Nancy Leveson

리스크 관리는 모든 프로그램에서 핵심적인 부분이므로 프로그램 및 조직의 리스크를 신중하게 식별하고 관리해야 한다. 리스크에 대한 평가(assessment)는 일반적으로 임시적인 방법으로 수행된다. 전문가 집단이 프로그램에서 리스크로 보이는 것들을 브레인스토밍 한 후, 그 결과에 대해 리스크 수준을 평가하는데 종종 자의적으로 평가하기도 한다. STPA는 리스크 관리 프로그램에 기반 한 체계적인 프로세스를 제공한다.

선행지표(leading indicators)를 리스크 관리에 사용할 때 이 지표들 또한 대개 임시적인 방법으로 만들어진다. 이 핸드북 앞 부분에서 설명한 바와 같이 STPA가 제품의 안전 및 기타 발현 속성을 고려할 때 효과적이었듯이, 지표를 만들 때에도 보다 정형적이고 구조화된 시스템 사고 프로세스(system-thinking process)를 사용하는 것이 효과적일 수 있다. STPA를 사용하여 선행지표를 생성하는 것은, 생성된 선행지표가 위험과 사고를 직접 추적할 수 있다는 장점이 있다. 이 장에서는 STPA의 결과와 전체 시스템 엔지니어링 프로세스 결과의 사용을 포함하여 선행지표를 식별하기 위한 구조적인 접근 방법을 설명한다. 다루는 주제는 아래와 같다.

- 선행지표란 무엇인가?
- 선행지표를 위해 현재 사람들이 사용하는 것은 무엇인가?
- 가정 기반 선행지표(assumption-based leading indicators)란 무엇인가?
- 아래를 수행하는 방법
  - ✓ STAMP와 STPA를 사용하여 적절하고 효과적인 선행지표 식별
  - ✓ 선행지표의 기반이 될 가정(assumption) 도출
  - ✓ 가정을 사용하여 가정 기반 선행지표 프로그램 생성
  - ✓ 선행지표를 리스크 관리 프로그램에 통합
- 타당성(feasibility) 및 최종 고찰

선행지표란 무엇인가?

선행지표는 사고가 발생하기 전에 사고 가능성을 식별하고 사고 방지를 위한 조치수단을 마련하는 데 사용된다. 이는 심각한 사고(accident)는 랜덤하거나 밀접한 이벤트의 특별한 집합(set)에 의해 발생하는 것이 아니라는 가정에 기반한다. 대신 사고는 다음의 경우 발생한다.

조직이 시간이 지나면서 리스크가 증가하는 상태로 바뀌고
안전장치(safeguard)와 통제가 느슨해지기 때문에
상충되는 목표와 트레이드오프, 리스크에 대한 인식 감소로 인해
더 위험한 행동으로 이어짐

이러한 가정은 시간이 지남에 따라 심각한 사고가 진행되는 것을 암시하므로 이 위험한 경로를 탐지하고 개입할 가능성이 있음을 의미한다. 선행지표는 개입이 필요하다는 신호이다.

---

21 이 장은, 다른 장과 마찬가지로, 몇 가지의 설명이 포함된 실용적인 "사용법" 가이드로 작성되었다. 많은 레퍼런스가 포함된, 보다 완전하고 기술적인 방법은 Nancy G.Leveson, A systems approach to risk management through leading safety indicators, Reliability Engineering and System Safety, Elsevier Publishers, 2014를 참고하길 바란다.

For organizations, leading indicators can provide early indications about when aspects of the product, services, or organizational behavior are starting to go off track before the long-term effects on the organizational goals are visible.

What do People Use Now for Leading Indicators?

There has been a lot of effort put into identifying general leading indicators, particularly in the petrochemical industry. The attempts have involved trying to find a few generally applicable metrics that could raise flags in any company or even industry. Some examples are maintenance backlogs, minor incidents, equipment failure rates, or the results of surveys on employee culture (beliefs). The latter assumes that all or most accidents are caused by operators. Such attempts to identify leading indicators have not been found to be particularly effective. They have also mostly been directed at occupational safety rather than system safety.

Other attempts to identify leading indicators use incident and accident reports. One drawback is that only things that have happened before can be considered. Many of these previous factors have already been eliminated or controlled. An alternative approach is to use hazard analyses to predict future scenarios. A problem here is the limited nature of the scenarios generated by traditional hazard analysis techniques, which focus almost all attention on hardware failures.

Social, organizational, and managerial factors that lead to accidents have been hypothesized to be useful as leading indicators, but mostly only a small number (5 or 6) have been identified with little real data on how effective such general indicators are. The problem is that these hypotheses are not based on any model or framework on how and why accidents occur and therefore are very incomplete. In addition, these leading indicators only include the most common factors that are shared across a large number of organizations, which may omit the most important factors for a specific organization.

An alternative, which is described in this chapter, does not try to identify general leading indicators for all accidents but instead identifies leading indicators based on the design of safety controls within a particular company or even a specific product or service. We called these Assumption-Based Leading Indicators.

What are Assumption-Based Leading Indicators?

The idea of assumptions being the basis for identifying leading indicators was originally proposed and used for risk management programs outside of engineering. RAND developed the methodology of assumption-based planning (ABP) in the late 1980s to assist U.S. Army clients with mid- and long-term defense planning and to reduce uncertainty and manage risk in Army missions.

We have taken this basic idea and created an approach to risk management in engineering. This approach is based on the premise that *useful leading indicators of increasing risk can be identified based on the assumptions underlying the safety design process for the specific organization, product, or operations*. All engineering involves assumptions about the behavior of the operational system, even when that system is an organizational or managerial structure. Such assumptions may include, for example, failure rates for a hardware component over time, how a product will be used and the environment in which the product is used or the services are provided, the basic training for people on the tools they are tasked to use, or assumptions about the information needs for decision making and how effectively the information channels operate. Other external or environmental assumptions may arise from what it is believed that customers want, which could change over time as the whole marketplace changes.

The formal definition of an assumption-based indicator is therefore:

조직에서의 선행지표는, 조직 목표에 미치는 장기적인 영향이 가시화되기 전에 제품, 서비스 또는 조직 행동의 양상이 궤도를 벗어나기 시작한 시점에 대한 초기 징후를 제공할 수 있다.

선행지표를 위해 현재 사람들이 사용하는 것은 무엇인가?

일반적인 선행지표를 식별하는 데 많은 노력이 있었으며 특히 석유화학 산업에서 그러했다. 이러한 노력에는 모든 기업 또는 산업에서 일반적으로 적용 가능한 몇 가지 지표(metric)를 찾는 것이 포함되었다. 몇 가지 예로는 유지보수 잔업, 경미한 사건(incident), 장비 고장률 또는 직원 문화(신념)에 대한 설문결과 등이 있다. 후자인 설문결과는 모든 또는 대부분의 사고가 작업자로 인해 발생한다고 가정한다. 선행지표를 식별하려는 이러한 시도는 특별히 효과적이라고 밝혀지지는 않았다. 또한 그것들은 대부분 시스템 안전보다는 직업 안전(occupational safety)에 초점이 맞추어져 있었다.

선행지표를 식별하려는 또 다른 시도에서는 사건 및 사고 보고서를 사용한다. 이러한 시도의 한 가지 단점은 이전에 일어난 일만을 고려할 수 있다는 점이며, 이 경우 기존의 요인 중 많은 부분은 이미 제거되었거나 통제된 상태이다. 또 다른 대안으로 위험 분석을 사용하여 미래의 시나리오를 예측하는 것이 있는데 이 경우 전통적인 위험 분석 기법들이 거의 모든 관심을 하드웨어 고장에만 집중함으로써 도출된 시나리오들의 특징이 제한된다는데 문제가 있다.

사고로 이어지는 사회적, 조직적, 경영적 요인들이 선행지표로 유용할 것이라고 생각되어 왔지만 그러한 일반적인 지표가 얼마나 효과적인지에 대한 실제 데이터는 대부분 극소수(5개 또는 6개)로 확인되었다. 문제는 이 가설들이 어떻게 그리고 왜 사고가 발생했는지에 대한 모델이나 프레임워크를 기반으로 하지 않기 때문에 매우 불완전하다는 것이다. 또한, 이러한 선행지표에는 많은 조직에서 공유되는 가장 일반적인 요소들만 포함되어 있기 때문에 특정 조직에서의 가장 중요한 요소가 누락될 수 있다.

이 장에서 설명하는 대안에서는 모든 사고에 대한 일반적인 선행지표를 식별하려는 것이 아니라 특정 기업, 특정 제품 또는 서비스의 안전 컨트롤(safety control) 설계에 대한 선행지표를 식별한다. 우리는 이것을 가정 기반 선행지표(Assumption-Based Leading Indicators)라 부른다.

가정 기반 선행지표(Assumption-Based Leading Indicators)란 무엇인가?

선행지표를 식별하는 기반이 되는 가정에 대한 아이디어는 원래 엔지니어링이 아닌 다른 분야의 리스크 관리 프로그램에 사용하기 위해 제안되었다. RAND는 1980년대 후반에 미 육군의 중장기 국방 계획을 지원하고 육군 미션의 불확실성을 줄이며 리스크를 관리하기 위한 가정 기반 계획(Assumption-Based Planning, ABP) 방법론을 개발하였다.

우리는 이러한 기본 아이디어를 차용하여 엔지니어링에서의 리스크 관리 접근법을 만들었다. 이 접근법은 다음의 전제를 기반으로 한다. *"증가하는 리스크에 대한 유용한 선행지표는 특정 조직, 제품 또는 운영에 대한 안전 설계 프로세스(safety design process)에 내재된 가정에 기반하여 도출할 수 있다."* 모든 엔지니어링은 운영 시스템의 동작에 대한 가정과 관련되는데 이는 시스템이 조직적 또는 관리적 구조일 때에도 동일하다. 이러한 가정에는 예를 들어, 시간 경과에 따른 하드웨어 컴포넌트의 고장률, 제품 사용 방법 및 제품이 사용되거나 서비스가 제공되는 환경, 업무를 수행하는 사람이 사용하는 도구에 대한 기본 교육, 의사 결정에 필요한 정보 및 효과적인 정보 채널 운영 방법에 대한 가정 등이 포함된다. 다른 외부적 또는 환경적인 가정들이 고객이 원하고 있다고 믿는 것으로부터 발생할 수 있으며, 이는 전체 시장이 변화함에 따라 변할 수 있다.

따라서 가정 기반 선행지표의 공식적인 정의는 다음과 같다.

An assumptions-based leading indicator is defined as a warning sign that can be used in monitoring a process to detect when an assumption is broken or dangerously weak or when the validity of an assumption is changing.

The goal of a leading indicators program should be to monitor the social, managerial, and product/service assumptions made by the designers of the engineered system, both to find assumptions that originally were incorrect—such as beliefs about how people or other system components will behave in a certain context—as well as those that were originally correct but become incorrect over time as people optimize local goals or the environment changes. Instead of trying to find leading indicators that apply to *all* organizations and systems, leading indicators are generated for a specific organization or system design based on the safety-related assumptions about that specific design.

While there may be many reasons why an organizational structure may not operate the way it was designed to, one important cause is that the organizational culture, i.e., the goals and values of the organization and those in it,[22] does not conform to that which is required to achieve the program goals. Alternatively, the culture may degrade over time, perhaps influenced by competitive, financial, or other pressures and personnel changes or because of simple regression to familiar habits. This cultural drift needs to be detected and corrected as part of an effective leading indicators program.

Where do assumption-based leading indicators come from? In the most basic sense, they represent assumptions about how the system goals will be achieved. There are, in general, three types of assumptions involved:

- That the models and assumptions used in the initial design of the engineered system (including the organizational structure) were correct.
- That the system will be constructed and operated in the manner assumed by the designers.
- That the original models and assumptions are not violated by changes in the system over time, perhaps in an attempt to improve or optimize the processes, or by changes in the environment.

These types of assumptions are used in an assumptions-based leading indicator program to deciding what should be checked, how, when, and what actions should be taken if the checking determines an assumption is no longer true (or perhaps never was). Such a program has three aspects:

1. Identifying appropriate and effective leading indicators,
2. Creating a leading indicator monitoring program to use them, and
3. Embedding this monitoring system within a carefully designed risk management program. Detection alone, in the form of the first two aspects, is not enough—there must be a management process in place to act when the leading indicators show that action is necessary.

Each of these three aspects is now described.

Identifying Appropriate and Effective Leading Indicators using STAMP and STPA

Despite much effort to avoid them, accidents still occur. Theoretically, if we design a safe system, that is, eliminate or adequately control or mitigate all the hazards and nothing changes, then we should not have accidents. The problem is that neither of these conditions is usually true in practice: no engineering process is perfect nor is human behavior. In addition, every system and its environment are

[22] The chapter on designing a Safety Management System contains a more detailed definition and discussion of organizational culture.

가정 기반 선행지표는 가정이 깨어지거나 심각하게 약화된 시점 또는 가정의 타당성(validity)이 변하는 시점을 탐지하기 위한 프로세스를 모니터링 하는데 사용할 수 있는 경고 징후(warning sign)로 정의된다.

선행지표 프로그램의 목표는 사회적, 관리적, 그리고 엔지니어링 시스템 설계자가 만든 제품/서비스에 대한 가정을 모니터링하고, 처음부터 잘못된 가정(예: 사람들 또는 다른 시스템 컴포넌트가 특정 상황에서 어떻게 동작할 지에 대한 믿음)이나 처음에는 문제가 없었지만 사람들이 로컬 목표나 환경 변화를 최적화함에 따라 시간이 지나면서 잘못된 가정을 찾는 것이다. 모든 조직과 시스템에 적용되는 선행지표를 찾으려고 노력하는 것이 아니라 특정 설계에 대한 안전-관련 가정(safety-related assumption)을 토대로 특정 조직이나 시스템 설계에 대한 선행지표를 생성하는 것이다.

조직 구조가 설계한 대로 운영되지 않는 이유는 많이 있겠지만 한 가지 중요한 이유는 조직 문화, 즉, 조직과 조직에 내재한 목표와 가치가[22] 프로그램 목표를 달성하는데 필요한 문화와 일치하지 않는다는 점이다. 또한, 문화는 시간이 지남에 따라 퇴화될 수 있고 아마도 경쟁적, 재정적 또는 다른 압력, 직원 변경 또는 익숙한 습관으로의 단순 퇴행 등에 영향을 받을 수 있다. 이러한 문화적 변화는 효과적인 선행지표 프로그램의 일환으로 탐지되고 수정되어야 한다.

가정 기반 선행지표는 어디서 비롯된 것일까? 가장 근본적으로는 시스템 목표 달성 방법에 대한 가정에서 비롯된다. 일반적으로 다음과 같은 세 가지 유형의 가정이 있다.

- 엔지니어링 시스템의 초기 설계(조직 구조 포함)에 사용된 모델 및 가정이 정확했음
- 시스템은 설계자가 가정한 방식으로 구축되고 운영될 것임
- 최초의 모델 및 가정은 시간이 지나면서 시스템이 변경되는 경우(프로세스를 개선하거나 최적화하는 시도 또는 환경의 변화)에도 위반되지 않음

이러한 유형의 가정은 가정 기반 선행지표 프로그램에서 무엇을 점검해야 할지, 어떻게, 언제 점검해야할지, 그리고 점검 결과, 가정이 더 이상 사실이 아니라고(또는 사실이었던 적이 없다고) 판단된 경우에는 무엇을 수행해야 할지를 결정하는 데 사용된다. 이러한 프로그램에는 다음과 같은 세 가지 측면이 있다.

1. 적절하고 효과적인 선행지표를 식별하고,
2. 이를 사용하기 위한 선행지표 모니터링 프로그램을 생성하고,
3. 신중하게 설계한 리스크 관리 프로그램에 이 모니터링 시스템을 내장한다. 위의 두 가지 측면으로만 탐지하는 것은 충분하지 않으며 선행지표를 통해 조치가 필요함을 확인했을 때 취해야 할 관리 프로세스가 있어야 한다.

이제 이 세 가지 측면 각각에 대해 설명하고자 한다.

STAMP와 STPA를 사용하여 적절하고 효과적인 선행지표 식별

사고를 방지하기 위한 많은 노력에도 불구하고 사고는 여전히 발생한다. 이론적으로, 안전한 시스템(safe system)을 설계하면, 즉 시스템의 모든 위험을 제거하거나 적절히 통제 또는 완화조치한 후 시스템을 변경하지 않았다면 사고는 발생하지 않아야 한다. 문제는 이러한 조건들 모두가 실제로는 사실이 아니라는 것이다: 엔지니어링 프로세스는 완벽하지 않으며 인간의 행동 또한 완벽하지 않다. 또한 모든

[22] 안전 관리 시스템(Safety Management System) 설계에 대한 장(7장)에서 조직 문화가 보다 자세하게 정의되고 논의됨

subject to change over time. The starting point in seeking more effective leading indicators is to consider the general causes of accidents.

A General Categorization of the Causes of Accidents

The causes for accidents may arise in technical system development, in operations, and in management. Usually several or all of these types of causes can be found in accident scenarios. The following list describes the reasons accident causes occur in each of these three areas:

Design and Manufacturing

- Inadequate hazard analysis: Assumptions about the system hazards or the process used to identify them do not hold.
    - HA is not performed or is not completed
    - Some hazards are not identified due to inadequacies in the hazard analysis process or in how it is performed. Important causes are omitted and therefore not handled
    - Hazards are identified but they are not handled because they are assumed to be "sufficiently unlikely."
- Inadequate design of control and mitigation measures for the identified hazards, possibly due to inadequate engineering knowledge or to inappropriate assumptions about operations.
- Inadequate construction of the control and mitigation measures.

Operations

- Controls that designers assumed would exist during operations do not actually exist, are not used, or turn out to not be effective.
- Controls exist, are used, and originally were effective, but changes over time violate the assumptions underlying the original design of the controls.
    - New hazards arise with changing conditions, were not anticipated during development, or were dismissed as unlikely to occur
    - Physical controls and mitigation measures degrade over time in ways not accounted for in the analysis and design process
    - Components (including humans) behave differently over time (violate assumptions made during design and analysis)
    - The system environment changes over time (violates assumptions made during design and analysis)

Management

- The safety management system design is flawed (see the chapter on how to design an effective safety management system in this handbook).
- The safety management system could be effective but does not operate the way it was designed (and assumed) to operate. While there may be many reasons for misbehavior, one general cause is that the safety culture, i.e., the goals and values of the organization with respect to safety, degrades over time or was ineffective from the beginning. In addition, the behavior of those making safety-related decisions may be influenced by competitive, financial or other pressures.

시스템과 환경은 시간이 지남에 따라 변경될 수 있다. 보다 효과적인 선행지표를 찾기 위한 출발점은 사고의 일반적인 원인을 고려하는 것이다.

사고 원인의 일반적인 분류

사고의 원인은 기술적인 시스템의 개발, 운영, 그리고 관리에서 발생할 수 있다. 일반적으로 이러한 유형의 원인 일부 또는 전부는 사고 시나리오(accident scenarios)에서 찾을 수 있다. 다음 목록은 이 세 가지 영역에서 사고 원인이 발생하는 이유를 설명한다.

설계 및 제조(Design and Manufacturing)

- 부적절한 위험 분석: 시스템의 위험 또는 위험을 식별하는 프로세스에 대한 가정이 유지되지 않음
  - 위험 분석(HA)이 수행되지 않았거나 완료되지 않음
  - 위험 분석 과정 또는 수행 방법상의 부적절성으로 인해 일부 위험이 식별되지 않음. 그 결과, 중요한 원인들은 누락되어 다루어지지 않음
  - 위험이 식별되었으나 "거의 발생하지 않을 것(sufficiently unlikely)"으로 간주되어 다루어지지 않음
- 부적절한 엔지니어링 지식 또는 운영에 대한 부적절한 가정으로 인해 식별된 위험에 대한 컨트롤(제어) 및 완화조치 수단이 부적절하게 설계됨
- 컨트롤(제어) 및 완화조치 수단이 부적절하게 구축(construction)됨

운영(Operations)

- 설계자가 운영 중에 존재할 것이라고 가정한 컨트롤이 실제로는 존재하지 않거나 사용되지 않거나 또는 효과적이지 않다고 판명됨
- 처음에는 효과적이었던 기존에 사용된 컨트롤이 시간이 지남에 따른 변경으로 인해 원래의 컨트롤 설계의 기반이 되는 가정을 위배함
  - 변화하는 조건에 따라 발생하는 새로운 위험이 개발 과정에서 예상되지 않았거나 발생하지 않을 것 같아서 묵살됨
  - 시간이 지남에 따라 물리적인 컨트롤 및 완화조치 수단이 분석 및 설계 프로세스에서 고려되지 않은 방식으로 저하됨
  - 컴포넌트(사람 포함)가 시간이 지남에 따라 다르게 동작함(설계 및 분석 단계에 만든 가정을 위반함)
  - 시스템 환경이 시간이 지남에 따라 변함(설계 및 분석 단계에 만든 가정을 위반함)

관리(Management)

- 안전 관리 시스템(safety management system) 설계에 결함이 있음(효과적인 안전 관리 시스템 설계 방법에 대한 본 핸드북의 장(7장) 참조)
- 안전 관리 시스템이 효과적일 수 있지만 설계(및 가정)대로 동작하지 않음. 잘못된 동작에는 많은 이유가 있을 수 있지만 한 가지 일반적인 원인은 안전 문화, 즉 안전에 대한 조직의 목표와 가치가 시간이 지남에 따라 저하되거나 처음부터 효과적이지 않았음. 또한, 안전-관련 결정을 내리는 사람들의 행동 또한 경쟁적, 재정적 또는 다른 압력에 의해 영향을 받을 수 있음

To prevent accidents, we must eliminate or reduce the occurrence of these causes. A leading indicators program can be used to attempt to detect them before an accident occurs. Let's look at each of these three categories of accident causes.

During design and manufacturing, the hazard analysis process used may be inadequate or may be skipped. Sometimes, schedule pressures lead to skipping the hazard analysis process or only superficially performing one. Accidents occur frequently today because the hazard analysis techniques are incapable of identifying the new causes in today's complex, software-intensive systems. The elimination of these factors related to accident causation arising during system development and implementation may seem relatively simple on the surface, but may be quite difficult practically and politically within some organizations. Suggestions are provided in the chapter of this handbook on integrating STPA into organizations, but extensive discussion is beyond the current scope of this handbook.

A very common cause of accidents is that the technical factors were identified during development but ruled out because they were calculated to be sufficiently unlikely that they could be ignored. The best solution for this is not to use probabilistic risk assessment or informal likelihood assessments as a justification for ignoring some hazards or their causal scenarios. This solution may not be politically possible, but a leading indicator can be created that identifies when events happen that are incorrectly judged to be impossible or remote enough to ignore. In a recent chemical plant explosion and fire that I was analyzing, a catalyst was determined in the 1970s to be inert and thus the possibility of it contributing to an accident was judged to be zero. However, over many years the composition and manufacturing process for the catalyst changed. There were, in fact, two explosions related to this catalyst at chemical plants owned by this same company. Yet, when a new plant was designed by engineers at this company (after the two explosions), the designers still assumed that the catalyst was inert and ignored it in the risk assessment. That plant later had a serious accident related to the use the same catalyst. The original assumption clearly became incorrect over time. A leading indicator should have been triggered after the first of the two previous explosions, let alone after two of them, and the occurrence of the explosions should have led to changes in design assumptions.

As another case, the causes may be identified in the hazard analysis, but control and mitigation measures may be poorly designed and implemented. An example is inadequate design of physical control mechanisms. Simple calculation or knowledge errors may be involved, but incorrect assumptions can also play an important role. The common engineering design assumption about independence when redundancy is used to protect against failures is an example. Consider the Macondo (Deep Water Horizon) blowout preventer. There was redundancy in the means to control a potential blowout, but the redundant units contained a common cause failure mode. Acceptance of the belief that blowout preventers never failed was widespread in the industry despite the fact that ineffective blowout preventers had previously contributed to several serious accidents. A clear leading indicator that the assumption was wrong existed but was ignored. Challenger is another example of a common cause failure, only this time the assumption about the independence of the O-rings was checked and invalidated scientifically many years before the fatal *Challenger* flight. However, the change was never documented in the Marshall Space Center data base where the launch decision was made. Given the very large number of accidents that have involved common mode/cause failure, it appears to be an important engineering design assumption to revisit as are others that have contributed to many accidents. In this case, a flawed safety information system was also involved.

Sometimes the hazard controls that are assumed to be used during operations are poorly designed and ineffective, perhaps because of unidentified conflicts with other controls or incentives. They also may not be adequately implemented or used. As mentioned previously, the potential for migration toward violation of the operational assumptions that underpin the safety of the system needs to be reflected in the set of leading indicators. For example, operators may start to take shortcuts or turn off

사고를 예방하기 위해서는 이러한 원인의 발생을 제거하거나 줄여야 한다. 선행지표 프로그램은 사고가 발생하기 전에 이를 발견하는 데 사용될 수 있다. 사고 원인의 세 가지 유형을 각각 살펴보자.

설계 및 제조(design and manufacturing) 과정에서 위험 분석 프로세스가 부적절하거나 생략(skip)되었을 수 있다. 때로는 일정에 대한 압박으로 인해 위험 분석 프로세스를 건너뛰거나 표면적으로만 수행하는 경우가 있다. 최근의 빈번한 사고는, 위험 분석 기법으로 오늘날의 복잡한 소프트웨어 집약적인 시스템에서의 새로운 원인(cause)을 식별할 수 없기 때문에 발생한다. 시스템 개발 및 구현 중에 발생하는 사고 원인과 관련된 요인들을 제거하는 것은 표면적으로는 비교적 단순해 보일 수 있지만 일부 조직에서는 현실적으로 그리고 정치적 이유로 꽤 어려울 수 있다. 이 핸드북에서 STPA를 조직에 통합하는 방법을 제시하고 있지만 그 외에도 이 핸드북의 범위를 넘어서는 광범위한 논의가 진행되고 있다.

가장 흔한 사고의 원인은 개발과정에서 기술적인 요인을 식별하였지만 무시할 수 있을 정도로 거의 발생하지 않을 것이라는 계산에 의해 배제되는 것이다. 이를 위한 최선의 해결책은 확률론적 리스크 평가(probabilistic risk assessment) 또는 비정형적 발생 가능성 평가(informal likelihood assessment)를 일부 위험 또는 위험의 원인 시나리오를 무시하기 위한 정당화로 사용하지 않는 것이다. 이 해결책은 정치적으로 가능하지 않을 수도 있지만 발생이 불가능하거나 충분히 무시할 정도로 먼 이야기라고 잘못 판단한 이벤트가 언제 발생하는지를 식별하는 선행지표를 만들 수 있다. 필자가 분석중인 최근 화학 공장의 폭발 및 화재의 경우, 1970년대에 촉매제가 비활성인 것으로 판단되어 그것이 사고에 기여할 가능성은 0이라고 판단했다. 하지만 수년에 걸쳐 촉매제의 구성과 제조 프로세스가 변경되었으며 사실, 이 회사가 소유한 화학 공장에서도 이 촉매제와 관련된 두 번의 폭발이 있었다. 그러나 이 회사의 엔지니어들이 새로운 공장을 설계하였을 때에도(두 번째 폭발 이후), 설계자들은 촉매제가 여전히 비활성이라 생각하고 리스크 평가에서 무시하였다. 이후 그 공장에서는 동일한 촉매제의 사용과 관련된 심각한 사고가 발생했다. 시간이 지나면서 최초의 가정은 명백히 잘못된 것이 되었다. 첫 번째 폭발 이후에 선행지표가 트리거 되었어야 하며, 폭발을 계기로 이후의 설계 가정이 변경되었어야 했다.

또 다른 경우로, 위험 분석을 통해 원인을 식별할 수 있지만 컨트롤 및 완화조치 수단이 잘못 설계되고 구현된 경우도 있다. 그 예로 물리적인 컨트롤 메커니즘의 부적절한 설계가 있다. 간단한 계산 또는 이론의 오류가 관련될 수도 있지만 정확하지 않은 가정이 중요한 원인이 될 수도 있다. 그러한 가정의 예로, 고장을 방지하기 위해 다중화(redundancy)를 적용할 때 흔히 사용되는 독립성에 대한 엔지니어링 설계 가정이 있다. Macondo(Deep Water Horizon) 폭발방지장치를 생각해보자. 이 장치에는 잠재적인 폭발을 제어하는 수단이 다중화 되어있지만 다중화된 유닛에는 공통 원인 고장(common cause failure) 모드가 포함되어 있었다. 과거에 효과적이지 않은 폭발방지장치가 여러 중대 사고의 원인이었다는 사실에도 불구하고 산업계에는 폭발 방지 장치가 절대 실패하지 않는다는 믿음이 만연해 있었고 가정이 잘못되었다는 명백한 선행지표가 존재했지만 무시되었다. 공통 원인 고장의 다른 예로 챌린저 우주왕복선 사고가 있다. 챌린저호의 비행이 있기 수 년 전에 이미 O-ring의 독립성 가정에 대한 검사가 있었고 과학적으로 틀렸다고 입증되었으나 발사 명령이 내려진 마샬 우주센터의 데이터베이스에는 변경 내용이 기록되지 않았다. 매우 많은 사고가 공통 모드/원인 고장(common mode/cause failure)과 관련되었다는 사실을 볼 때 공통 모드/원인 고장은 다시 살펴보아야 할 중요한 엔지니어링 설계 가정으로 보인다. 이 경우 결함 있는 안전 정보 시스템 또한 관련되어 있다.

때로는 운영(operations) 중 사용될 것이라고 여겨지는 위험 컨트롤이, 다른 컨트롤이나 장려책(incentives)과의 알 수 없는 충돌로 인해 잘못 설계되고 효과적이지 않은 경우가 있고 적절하게 구현되지 않거나 적절하게 사용되지 않을 수도 있다. 앞에서 언급했듯이 시스템 안전을 뒷받침하는 운영상 가

safety devices in order to operate more efficiently; or, in an air traffic control system, the airspace may become more crowded than originally considered during system design. The assumptions that need to be checked here must be documented and passed to the operators.

Finally, assumptions involved in the design and operation of the safety management system (see Chapter 6) may be or become untrue. The goals and values of those participating in an industry or organization, i.e., the safety culture, is an important assumption that when wrong can be a major factor in accidents and must be reflected in the set of leading indicators. For example, a safety policy is a basic requirement for every company or organization to communicate the desired safety culture and behavior expected of individuals. There must be a way to measure how well that policy is being followed and if adherence decreases over time. Assumptions about management behavior and decision making are also commonly found to be violated after accidents occur, and these assumptions must be monitored.

How to Generate the Assumptions on which to Base Leading Indicators

Identifying appropriate and effective leading indicators can be accomplished, at least partially, using STPA. The goal is to use the hierarchical control structure model and the controls built into the organization to identify the assumptions on which constraints about the emergent property is based and to determine how to detect any weakening effectiveness of the controls designed to achieve the system goals. Additional assumptions can be identified during the system design process, as described in this section.

A list of where assumptions may come from is shown in the box below.

Assumptions may be generated from:

- High-level system goals, especially those including quantities, generated during concept development
- System-level requirements generated from the system goals
- Assumptions about the external environment in which system will operate
- System behavioral requirements imposed by safety-related environmental requirements and constraints (including constraints on the use of the system)
- STPA-generated hazards, the hierarchical control structure, unsafe control actions, and causal scenarios
- Design features devised to manage the causal scenarios
- Operational requirements created to manage causal scenarios that cannot be entirely handled through system design features
- Limitations in the design of safety-related controls, including operational controls
    - Limitations related to achievement of functional requirements
    - Limitations related to environmental assumptions
    - Limitations related to hazards and causal factors that could not be completely eliminated or controlled in the design or operational procedures
    - Limitations arising from tradeoffs made during system design

An example aircraft collision system called TCAS (Traffic alert and Collision Avoidance System) is used to illustrate the generation of critical assumptions on which leading indicators can be based. My

정의 위반 가능성은 선행지표에 반영되어야 한다. 예를 들어 운영자는 보다 효율적인 운영을 위해 쉬운 방법을 택하거나 안전장치를 끌 수 있다. 또는 ATC(Air Traffic Control, 항공교통관제) 시스템에서 영공은 시스템 설계 시에 최초 고려했던 것보다 더 혼잡할 수 있다. 여기서 점검해야 할 가정은 문서화되어 운영자에게 전달되어야 한다.

마지막으로, 안전 관리(management) 시스템의 설계 및 운영에 관련된 가정은(7장 참조) 잘못되었거나 잘못될 수 있다. 조직이나 산업에 참여한 사람들의 목표와 가치, 즉, 안전문화는 잘못되었을 때 사고의 주요 원인이 될 수 있으므로 선행지표에 반영해야 하는 중요한 가정이다. 예를 들어 안전 정책은 모든 기업 또는 조직이 원하는 안전 문화와 개인에게 기대하는 행동을 전달하기 위한 기본 요구사항인데 이 정책이 얼마나 잘 수행되고 있는지, 준수율이 시간이 지남에 따라 감소하는지를 측정하는 방법이 필요하다. 관리활동과 의사 결정에 대한 가정도 사고 발생 후에는 흔히 위반되는 것으로 나타났으며 이러한 가정 또한 반드시 모니터링 되어야 한다.

선행지표의 기반이 되는 가정을 생성하는 방법

STPA를 사용하면 적절하고 효과적인 선행지표를 최소한 부분적으로라도 식별할 수 있다. 목표는 계층적 컨트롤 스트럭처 모델과 조직 내부의 컨트롤을 사용하여 발현 속성의 제약사항들이 기반하고 있는 가정을 식별하고 시스템 목표를 달성하기 위해 설계된 컨트롤 중 효과가 약해진 것을 감지하는 방법을 결정하는 것이다. 이 섹션에서 설명한 것처럼 시스템 설계 프로세스에서 추가적인 가정을 식별할 수 있다.

가정이 생성되는 출처는 아래와 같다.

가정은 다음으로부터 생성될 수 있음:

- 개념 개발 단계에서 생성되는 상위 수준의 시스템 목표, 특히 수량을 포함한 목표
- 시스템 목표로부터 생성된 시스템 수준의 요구사항
- 시스템이 운영되는 외부 환경에 대한 가정
- 안전-관련 환경 요구사항 및 제약사항(시스템 사용에 대한 제약사항 포함)에 따른 시스템 동작 요구사항
- STPA를 통해 생성된 위험, 계층적인 컨트롤 스트럭처, 언세이프 컨트롤 액션(UCA) 및 원인 시나리오
- 원인 시나리오를 관리하기 위해 고안된 설계 기능(feature)
- 시스템 설계 기능(feature)을 통해 완전히 처리할 수 없는 원인 시나리오를 관리하기 위한 운영 요구사항
- 안전-관련 컨트롤(운영상의 컨트롤을 포함) 설계에서의 한계
  - 기능 요구사항의 달성과 관련된 한계
  - 환경 가정에 관련된 한계
  - 설계 또는 운영 절차에서 완전히 제거되거나 통제할 수 없는 위험 및 원인 요소에 관련된 한계
  - 시스템 설계 중 발생하는 트레이드오프(tradeoffs)로 인한 한계

선행지표의 기반이 될 수 있는 중요한 가정을 생성하는 것을 설명하기 위해 TCAS(Traffic Alert and Collision Avoidance)라는 항공기 충돌 시스템의 예를 사용한다. 필자는 학생들과 함께 1990년대 초반에 TCAS의 공식적인 인증을 지원하였고 우리 중 2명은 나중에 TCAS에 대한 의도 명세서(intent

students and I assisted with the official certification of TCAS in the early 1990s, and later two of us created a TCAS intent specification.[23]

A critical part of an intent specification is the documentation of the assumptions under which the safety of the system is based. The examples here come from that specification and almost all are related to the scenarios generated by the qualitative hazard analysis[24] used for TCAS certification. A few of the assumptions did not come from the hazard analysis scenarios but instead from the design goals and the environment in which TCAS was expected to operate. Although STPA did not exist at the time that TCAS was designed and certified, we have since shown that STPA would generate all of the hazardous scenarios generated by the fault trees used plus many that were omitted from the fault tree analysis. Thus all the example assumptions below could be generated from the STPA results.

Generation of assumptions should begin during early concept development. Clues may be found when the high-level system goals and requirements contain numbers, but that is not necessary. In TCAS, two of the TCAS system goals are:

***G1**: Provide affordable and compatible collision avoidance system options for a broad spectrum of National Airspace System (NAS) users.*

***G2**: Detect potential midair collisions with other aircraft in all meteorological conditions; throughout navigable airspace, including airspace not covered by ATC primary or secondary radar systems, and in the absence of ground equipment.*

The system goals are used to create system requirements. Assumptions underlying these requirements can be generated at the same time. For example,

***1.18**: TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute [**G1**].*

Assumption: This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 feet per minute during vertical climb or controlled descent and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm.

This assumption is an example of something that will need to be checked in the future to ensure that technological changes have not contradicted it, making vulnerable all the technical design decisions that were based on it (which can be identified by the traceability pointers in an intent or other specification).

Another example system requirement is that:

***1.19.1**: TCAS shall operate in enroute and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (i.e., 24 aircraft within 5 nmi) [**G2**].*

Assumption: Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.

[23] *Intent specifications* are based on system theory, system engineering principles, and psychological research on human problem solving and how to enhance it. The goal is to assist humans in dealing with the complexity of specifying a complex system. Chapter 10 in *Engineering a Safer World* provides more information.

[24] A very extensive fault tree analysis—that included detailed human and software behavior—was used in the TCAS development and certification effort. Because the goal was completeness over quantification, no attempt was made to limit the fault tree boxes to things that could be quantified. The resulting fault tree was the most complete I have seen in my 35 years in safety engineering. It did not include all the scenarios that would be found by STPA, but it also did not exclude scenarios for which quantification was not possible which happens when the goal is to produce probabilities for the scenarios.

specification)[23]를 만들기도 했다.

의도 명세서에서 중요한 것은 시스템 안전의 기반이 되는 가정의 문서화이다. 여기의 예제는 그 명세서에서 나온 것이고 거의 모든 것은 TCAS 인증에 사용된 정성적 위험 분석(qualitative hazard analysis)[24]을 통해 생성된 시나리오와 관련이 있다. 가정 중 일부는 위험 분석 시나리오에서 나온 것이 아니라 설계 목표와 TCAS가 동작할 것으로 예상되는 환경에서 나온 것이다. TCAS가 설계되고 인증된 시점에는 STPA가 존재하지 않았지만 지금까지 FTA를 사용하여 생성한 위험 시나리오 모두와 FTA에서는 누락된 여러 시나리오를 STPA를 이용하여 만들 수 있다는 것을 보였다. 따라서 아래의 모든 가정에 대한 예는 STPA 결과에서 생성될 수 있다.

가정의 도출은 초기 개념 개발 단계에서 시작되어야 한다. 상위 수준의 시스템 목표와 요구사항에 숫자가 포함되어 있다면 단서를 찾을 수 있지만 반드시 단서가 필요한 것은 아니다. TCAS의 경우 시스템의 목표는 2가지이다.

G1: NAS(National Airspace System) 사용자들의 넓은 스펙트럼을 고려하여 알맞고 호환성 있는 충돌 방지 시스템 옵션을 제공함

G2: 모든 기상 조건에서 다른 항공기와의 공중 충돌 가능성을 탐지함. ATC의 1차 또는 2차 레이다 시스템에 의해 커버되지 않는 공역 및 지상장비가 없어서 커버되지 않는 공역을 포함한 운항 가능 공역(navigable airspace)을 대상으로 함

시스템의 목표는 시스템 요구사항을 만드는데 사용되며 이러한 요구사항에 내재된 가정도 동시에 생성될 수 있다. 예를 들면,

1.18: TCAS는 임의의 2대의 항공기가 수평으로 1,200노트, 수직으로 10,000피트/분의 속도로 가까워지는 것을 방지하기 위한 충돌 방지 보호 기능을 제공해야 함 [G1].

<u>가정:</u> 이 요구사항은 상업용 항공기가 수직 상승 또는 제어된 하강 중에 최대 600노트 및 5,000피트/분으로 동작할 수 있으며 따라서 두 비행기는 최대 수평 1,200노트 및 수직 10,000피트/분으로 접근할 수 있다는 가정에서 도출되었다.

이 가정은 미래의 기술적 변화가 가정에 기반한 모든 기술적 설계 결정을 무력화하여(의도 명세서나 다른 명세서 내의 추적성을 통해 식별할 수 있음) 가정이 모순되게 하지 않는지를 점검해야 하는 한 가지 예이다.

시스템 요구사항의 또 다른 예는 다음과 같다.

1.19.1: TCAS는 항공로(enroute) 및 터미널 구역에서 교통밀도가 평방 해리당 0.3 항공기(즉, 5해리에 항공기 24대)로 운영되어야 함 [G2]

<u>가정:</u> 교통밀도는 1990년까지 이 수준으로 증가할 수 있으며, 향후 20년간 최대 밀도가 될 것이다.

---

23 의도 명세서는 시스템 이론, 시스템 엔지니어링 원칙, 그리고 사람 문제 해결에 대한 심리연구와 이를 향상시키는 방법에 기반한다. 의도 명세서의 목표는 복잡한 시스템의 복잡성을 사람들이 처리하는 데 도움을 주는 것이다. *'Engineering a Safer World'* 10장에 더 많은 정보가 있다.

24 TCAS 개발 및 인증에는 FTA가 매우 폭넓게(사람과 소프트웨어의 상세한 행동을 포함함) 사용되었다. 정량화보단 완전성이 목표였기 때문에 Fault tree box를 정량화 할 수 있는 것으로 제한하려 하지 않았다. 그 결과 만들어진 Fault tree는 필자가 35년 동안 안전 공학분야에서 본 것 중 가장 완벽했다. STPA를 이용하여 발견할 수 있는 모든 시나리오를 포함하지 않았지만, 시나리오의 확률을 산출할 때 정량화가 불가능한 시나리오를 제외하지도 않았다.

Again, future aircraft performance limits may change or there may be significant changes in airspace management, such as reduced vertical separation or very different ways of handling air traffic. Lots of computations in TCAS are based on the assumption underlying requirement 1.19.1, and it needs to be monitored to trigger re-evaluation of the safety parameters if it changes.

Another type of assumption may be identified (and specified) to explain a decision or to record fundamental information on which the design is based. For example, the design may be based on assumptions about the environment in which the system will operate. Examples from TCAS include:

***EA1**: High-integrity communications exist among aircraft*

***EA2**: The TCAS-equipped aircraft carries a Mode-S air traffic control transponder.*[25]

***EA3**: All aircraft have operating transponders*

***EA4**: All aircraft have legal identification numbers*

***EA5**: Altitude information is available from intruding targets with a minimum precision of 100 feet.*

***EA6**: The altimetry system that provides the aircraft's pressure altitude to the TCAS equipment will satisfy the requirements in RTCA Standard …*

***EA7**: Threat aircraft will not make an abrupt maneuver that thwarts the TCAS escape maneuver.*

New technology and new types of aircraft integrated into controlled airspace could violate these assumptions. **EA4** is an example of a non-technical assumption. Identification numbers are usually provided by the aviation authorities in each country. That assumption will need to be ensured by international agreement and monitored by some international agency. The example assumption that aircraft have operating transponders (**EA3**) may be enforced by the airspace rules in a particular country and, again, must be ensured by some group. The truth of this assumption is critical as TCAS will not display any aircraft without an operating transponder nor provide a resolution advisory (RA).[26] **EA7** is an example of an assumption on the behavior of pilots and the air traffic control system and could also be violated by the introduction of unmanned or other types of new aircraft into the airspace.

Some assumptions may be imposed on the system by environmental requirements and constraints. Those assumptions may lead to restrictions on the use of the new system (which will require assumption checking) or may indicate the need for system safety and other analyses to determine the constraints that must be imposed on the system being created or on the larger encompassing system to ensure safety. Examples for TCAS include:

***E1**: The behavior or interaction of non-TCAS equipment with TCAS must not degrade the performance of the TCAS equipment or the performance of the equipment with which TCAS interacts.*

***E2**: Among the aircraft environmental alerts, the hierarchy shall be: Windshear has first priority, then the Ground Proximity Warning System (GPWS), then TCAS.*

***E3**: The TCAS alerts and advisories must be independent of those using the master caution and warning system.*

These assumptions would only need to be checked when major design changes are introduced in the aircraft.

STPA will provide important input to the assumptions and leading indicator process, starting with the specification of the system hazards and proceeding through the entire modeling and analysis process. As an example for TCAS, the system hazards were identified as:

[25] An aircraft transponder sends information that assists air traffic control in maintaining aircraft separation.

[26] A resolution advisory is basically a maneuver that will avoid the intruder.

미래 항공기의 성능 한계가 변하거나 공역 관리의 큰 변화가 있을 수도 있다. 예를 들어 수직 이격거리 감소나 전혀 다른 방법으로 항공 교통을 처리하는 것 등이다. TCAS에서의 많은 계산은 요구사항 1.19.1에 내재된 가정에 기반을 두고 있으므로, 만약 변경된 경우에는 안전 파라미터를 재평가할 수 있도록 모니터링 할 필요가 있다.

의사 결정을 설명하거나 설계의 기반이 되는 기본적인 정보를 기록하기 위해 또 다른 유형의 가정을 식별(특정)할 수 있다. 예를 들어 설계는 시스템 운영 환경에 대한 가정에 기반할 수 있다. TCAS의 예는 다음과 같다:

EA1: 높은 무결성의 항공기 간 통신
EA2: TCAS가 장착된 항공기는 Mode-S 항공교통관제 트랜스폰더(transponder)[25]를 탑재하고 있음
EA3: 모든 항공기는 동작하는 트랜스폰더를 가짐
EA4: 모든 항공기는 법적 식별 번호(identification number)를 가지고 있음
EA5: 침입 항공기(intruding targets)로부터의 고도정보는 최소 100피트 정확도로 얻을 수 있음
EA6: 항공기의 기압 고도를 TCAS 장비에 제공하는 고도계 시스템(altimetry system)은 RTCA(Radio Technical Commission for Aeronautics) 표준...의 요구사항을 충족함
EA7: 위협 항공기(threat aircraft)는 TCAS 탈출 기동(escape maneuver)을 방해하는 갑작스런 기동(abrupt maneuver)을 하지 않음

통제된 공역 내에 들어온, 새로운 기술 및 새로운 유형의 항공기는 이러한 가정을 위반할 수 있다. EA4는 비기술적인 가정의 예시이다. 식별 번호는 일반적으로 각 국가의 항공운항 감독기관(aviation authorities)에서 발급하므로 이 가정은 국제 협약에 의해 보장되고 국제기구에 의해 감시되어야 한다. 항공기는 작동하는 트랜스폰더를 가진다는 가정은(EA3) 특정 국가의 영공 규칙에 의해 강제될 수 있으며 또한 특정 그룹에 의해 보장되어야 한다. 이 가정이 사실이어야 하는 것이 매우 중요한데, 왜냐하면 TCAS는 트랜스폰더를 사용하지 않는 항공기를 표시하지 않고 대응 지침(RA, Resolution Advisory)[26] 또한 제공하지 않을 것이기 때문이다. EA7은 조종사와 ATC 시스템 동작에 대한 가정의 예이며 무인 또는 다른 유형의 새로운 항공기가 영공에 들어오는 경우 역시 위반될 수 있다.

환경 요구사항 및 제약사항에 따라 시스템에 일부 가정이 부여될 수도 있으며, 이러한 가정은 새로운 시스템의 사용을 제약할 수 있다(이 경우 가정에 대한 확인이 필요함). 이 가정은 시스템 안전을 위한 요구를 의미하는 것일 수도 있고, 시스템을 포함하는 더 큰 범위의 시스템의 안전 보장을 위해 현재 개발하는 시스템의 제약사항을 결정하는 분석 과정을 요구하는 것일 수도 있다. TCAS에 대한 이러한 예는 다음과 같다.

E1: TCAS 장비를 장착하지 않은 항공기와의 상호작용이나 동작이 TCAS 장비의 성능 또는 TCAS가 상호작용하는 장비의 성능을 저하시켜서는 안 됨
E2: 항공기 환경 경보의 계층 구조는 다음과 같아야 함 : 돌풍(windshear)이 최우선이며 그 다음이 지면근접경보장치(Ground Proximity Warning System, GPWS), 그 다음은 TCAS임
E3: TCAS 경보(alert) 및 지침(advisory)은 메인 주의-경고 시스템(master caution and warning system)을 사용하는 것과 독립적이어야 함

이러한 가정은 항공기에 주요 설계 변경 사항이 있을 때에만 점검하면 된다.

---

25 항공기 트랜스폰더(응답기)는 항공기 이격을 유지하기 위한 ATC를 지원하는 정보를 전송한다.

26 대응 지침(resolution advisory)은 기본적으로 침입 항공기를 피할 수 있는 기동임

***H1**: TCAS causes or contributes to a near midair collision (NMAC), defined as a pair of controlled aircraft violating minimum separation standards.*

***H2**: TCAS causes or contributes to an aircraft coming too close to a fixed structure or natural terrain.*

***H3**: TCAS causes or contributes to the pilot losing control over the aircraft.*

***H4**: TCAS interferes with other safety-related aircraft systems (for example, ground proximity warning)*

***H5**: TCAS interferes with the ground-based air traffic control system (e.g., transponder transmissions to the ground or radar or radio services).*

***H6**: TCAS interferes with an ATC advisory that is safety-related (e.g., avoiding a restricted area or adverse weather conditions).*

The first basic set of safety-critical assumptions is that hazards will not occur in a properly designed and operated system. Any occurrence of one of these hazards (even if an accident does not result) should trigger a complete review of the safety engineering process, in this case, the process used to eliminate or mitigate TCAS hazards. Checking an assumption after the hazard has occurred is likely too late to prevent a loss, however, but the identification of hazards serves as a starting point from which earlier checks can be derived by using STPA to identify the scenarios that can lead to a hazard.

Additional assumptions can be deduced even at this high level, for example, that there is a ground-based air traffic control system (which could change in the future) and that TCAS will not interfere with its operation. While hazards rarely change, new ones may be introduced when changes are made to the system and the process used to handle them may be undermined.

Checks for the occurrence of hazards and unsafe control actions also provide important information about the adequacy of the hazard analysis process itself. The goal of hazard analysis and safety engineering is to identify hazards and then either eliminate or prevent them. If they cannot be prevented, then they need to be mitigated. Hazards that the engineers thought were eliminated or prevented should, of course, never occur. If they do, this event is an indication of flaws in the engineering process or perhaps in the assumptions made about the operational system, such as assumptions about pilot or air traffic controller behavior. It is not just enough to fix the technical process. The holes in the development process that allowed hazardous behavior to occur also need to be fixed.

Ideally, flaws in engineering practices or operational behavior assumptions will be identified by leading indicators before the actual hazardous states occur. This goal can be achieved by identifying the assumptions underlying the hazardous scenarios identified by STPA. The hierarchical control structure design, the UCAs, and the causal scenarios will be useful.

The control structure itself contains assumptions that can be used to create leading indicators. A midair collision over Uberlingen Germany in 2002 demonstrates the role of assumptions about the operation of the control structure in safety. There were three groups involved that had potential responsibilities over the pilot's response to a potential NMAC: TCAS, the ground air traffic controller, and the airline operations center. The latter provides the airline procedures for responding to TCAS alerts and trains the pilots in them. The relevance of the other two groups is obvious. Clearly, any potential conflicts and coordination problems between these three controllers will need to be resolved in the overall design and operation of the air traffic management system. In the case of TCAS, the RA provided by TCAS was assumed to always to be followed in the case of conflicting advisories. The designers decided that because there was no practical way, at that time, to downlink information to the ground controllers about any TCAS advisories that might have been issued to the crew, the pilot was to immediately implement the TCAS advisory and the co-pilot would transmit the TCAS alert information

STPA는 시스템 위험 명세에서 시작하여 전체 모델링 및 분석 프로세스를 거쳐 가정 및 선행지표 프로세스에 중요한 정보를 제공한다. TCAS 예로 시스템 위험은 다음과 같이 식별되었다.

H1: TCAS가 NMAC(near midair collision)을 야기하거나 기여함 – NMAC은 한 쌍의 통제된 항공기가 최소이격기준을 위반하는 것을 말함
H2: TCAS가 고정된 구조물이나 자연지형에 항공기가 너무 근접하는 것을 야기하거나 기여함
H3: TCAS가 항공기에 대한 조종사의 통제력 상실을 야기하거나 기여함
H4: TCAS가 다른 안전-관련 항공기 시스템(예: 지상근접경보)를 방해함
H5: TCAS가 지상 기반 ATC 시스템(예: 지상 또는 레이다 또는 라디오 서비스로의 트랜스폰더 송신)을 방해함
H6: TCAS가 안전-관련 ATC 지침(예: 제한된 지역이나 악천후를 피하는 것)을 방해함

기본적인 첫 번째 안전-필수 가정(safety-critical assumption)은 적절하게 설계되고 운영되는 시스템에서는 위험이 발생하지 않는다는 것이다. 하지만 위험 중 하나라도 발생한다면(사고로 이어지지 않더라도) 안전 공학 프로세스의 전체적인 검토가 필요하다. 예제의 경우에는 TCAS 위험을 제거하거나 완화하는 데 사용된 프로세스의 검토가 필요하다. 위험이 발생한 후에 가정을 점검하는 것은 손실을 막기에 너무 늦은 시점일 수도 있지만 STPA를 이용하여 위험으로 이어질 수 있는 시나리오를 식별함으로써 보다 먼저 체크하는 출발점이 될 수 있다.

예를 들어 지상 기반 ATC 시스템(미래에는 변경될 수 있음)이 있고 TCAS가 ATC 시스템의 운영을 방해하지 않을 것이라는 추가적인 가정도 이러한 상위 수준에서 추론할 수 있다. 위험은 거의 변하지 않지만 시스템이 변경되면 새로운 위험이 유입될 수 있으며 기존의 프로세스로는 그러한 위험을 다루기 어려울 수 있다.

위험 및 안전하지 않은 컨트롤 액션의 발생을 점검(check)하는 것은 위험 분석 프로세스 자체의 적절성에 대한 중요한 정보를 제공한다. 위험 분석 및 안전 공학의 목표는 위험을 식별한 다음 이를 제거하거나 예방하는 것이다. 만약 예방할 수 없는 경우에는 완화해야 한다. 물론 엔지니어가 제거되거나 예방되었다고 생각한 위험이 발생해서는 안 된다. 만약 발생한다면 이것은 엔지니어링 프로세스의 결함을 나타내거나 조종사나 ATC 동작의 가정과 같은 운영 시스템에 대한 가정의 결함을 나타낸다. 이는 기술적인 프로세스 수정만으로는 충분하지 않다. 위험한 동작 발생을 허용한 개발 프로세스의 결함도 수정되어야 한다.

이상적으로는 엔지니어링 관행이나 운영 동작에 대한 가정의 결함이 실제 위험한 상태가 발생되기 전에 선행지표로 식별되는 것이 좋다. 이러한 목표는 STPA를 이용하여 식별된 위험 시나리오의 기본 가정을 확인함으로써 달성할 수 있다. 계층적인 컨트롤 스트럭처 설계, UCA, 그리고 원인 시나리오가 유용할 것이다.

컨트롤 스트럭처 자체에 선행지표를 생성하는 데 사용할 수 있는 가정이 포함되어 있다. 2002년 독일 Uberligen에 발생한 공중 충돌은 안전 관점에서 컨트롤 스트럭처 동작에 대한 가정의 역할을 증명해준다. 잠재적인 NMAC에 대한 조종사 응답과 관련하여 잠재적인 책임은 3개의 그룹으로, 즉, TCAS, 지상 관제사, 항공사 운영센터로 나눌 수 있다. 항공사 운영센터는 TCAS 경보에 응답하는 항공사 절차를 만들고 조종사를 교육한다. 다른 두 그룹의 관련성은 명백하다. 분명히 이 3개 컨트롤러 사이의 모든 잠재적 충돌 및 조정 문제는 항공 교통 관리 시스템의 전반적인 설계 및 운영 관점에서 해결되어야 한다. TCAS의 경우 상충되는 지침이 있을 시 TCAS가 제공한 대응 지침(RA, resolution advisory)을 항상 따라야 한다고 가정했다. 설계자는 그 당시에 승무원에게 발행된 TCAS 지침을 지상

by radio to the ground ATC so that the ground air traffic controller would know the state of the airspace and the advisories being given. The airline would provide the appropriate procedures and training to implement this protocol.

Several important assumptions about how conflicting advisories would be handled were violated in the Uberlinger midair collision. For example, there were supposed to be two air traffic controllers in the ground ATC tower, the pilots are supposed to follow the TCAS maneuver when there is a conflict between the advisory provided by the ground ATC system and TCAS, and the airline was assumed to be training pilots to follow the TCAS alert in such a conflict situation. The first of these assumptions had been violated for a while at night in the Swiss air traffic control center handling the two aircraft at the time of the tragedy. It is unknown whether the second one had been violated previously as that information was never checked. The third assumption, i.e., that the airline involved was training pilots to always follow TCAS when presented with conflicting advisories also had not been true for a long time, but nobody apparently was given the responsibility for ensuring that such training was occurring or they had not been exercising that responsibility. These incorrect assumptions about the operation of the control structure, if checked, could have served as leading indicators that the designed control structure was degrading.

The unsafe control actions, safety constraints, and causal scenarios also provide information about critical assumptions. The causal scenarios are used to design the controls and thus form the assumptions under which the controls are created. For example, **H5** gives rise to the following system safety constraint:

> ***SC.2****: TCAS must not interfere with the ground ATC system or other aircraft transmissions to the ground ATC system (**H5**).*

STPA can be used to identify causal scenarios for the violation of **SC.2**. This information can then be refined into a more detailed safety constraint **SC2.1**:

> ***SC2.1*** *The system design must not interfere with ground-based secondary surveillance radar, distance-measuring equipment channels, and with other radio services that operate in the 1030/1090 MHz frequency band (**2.5.1**).*

The assumption underlying a safe design of TCAS is that such interference will never occur. If it does, then this is a leading indicator that the design or operation of the system is flawed.

Humans tend to change their behavior over time and use automation in different ways than originally intended by the designers. Therefore, assumptions about operator behavior provide another important source for identifying leading indicators. For example, **H3** is that TCAS causes or contributes to the pilots losing control over the aircraft. Safety constraint **SC.6**, which is derived by STPA from **H3**, says:

> ***SC.6****: TCAS must not disrupt the pilot and ATC operations during critical phases of flight nor disrupt aircraft operation (**H3, 2.2.3, 2.19, 2.24.2**).*

Besides identifying the related hazard from which this safety constraint was derived (in this case **H3**), the specification here also points to features of the design (labeled **2.2.3**, **2.19**, and **2.24.2** in my TCAS intent specification) used to control that hazard, i.e., to enforce **SC.6**. These controls also contain important assumptions that need to be checked. The most basic assumption is that these controls will be effective in preventing the hazardous scenario and that they are implemented correctly. For example, in the STPA analysis, one of the scenarios identified that could lead to the violation of **SC.6** is that TCAS provides distracting resolution advisories while the pilot is on the ground or in the middle of taking off. A control was designed to prevent this scenario that allows the pilot to inhibit the production of potentially distracting resolution advisories during critical phases of takeoff and landing:

의 관제사에게 전달할 현실적인 방법이 없었기 때문에, 조종사는 즉시 TCAS 지침을 이행하고 부조종사는 지상 관제사가 영공의 상태와 지침이 발행된 상황을 알 수 있도록 TCAS 경고 정보를 라디오를 이용하여 지상 ATC로 전송하는 것으로 결정하였다. 항공사는 이 프로토콜을 이행하기 위한 적절한 절차를 만들고 교육하였을 것이다.

Uberlinger 공중 충돌의 경우 상충되는 지침을 어떻게 처리할 것인가에 대한 몇 가지 중요한 가정이 위반되었다. 예를 들면 그러한 가정은 지상 ATC 타워에는 2명의 항공교통관제사가 있어야 하며, 지상 ATC 시스템과 TCAS가 제공한 지침이 상충되는 경우 조종사는 TCAS를 따르며, 항공사는 지침이 상충된 상황에서 TCAS 경보를 따르도록 조종사를 교육한다는 것이다. 이 가정들 중 첫 번째는 한밤중에 두 대의 비행기를 취급하던 스위스 항공관제센터에서 위반되었다. 두 번째 가정은 정보가 확인되지 않았기 때문에 위반되었는지 여부를 알 수 없었다. 세 번째 가정, 즉, 상충되는 지침이 있을 시 TCAS를 따르도록 해당 항공사가 조종사를 훈련해야 하는 가정은 오랫동안 수행되지 않고 있었다. 하지만 그 어느 누구도 이러한 훈련이 진행되고 있는지 또는 항공사가 그 책임을 이행하지 않는지를 확인할 책임이 없었다. 만약 컨트롤 스트럭처의 동작에 대한 이러한 잘못된 가정을 점검하였다면, 설계한 컨트롤 스트럭처에 문제가 발생하고 있다는 선행지표로 활용되었을 것이다.

안전하지 않은 컨트롤 액션, 안전 제약사항 및 원인 시나리오는 중요한 가정에 대한 정보도 제공한다. 원인 시나리오는 컨트롤을 설계하는 데 사용되므로 가정을 형성하며 이 가정에 기반하여 컨트롤이 만들어진다. 예를 들어 H5는 다음과 같은 시스템 안전 제약사항을 만들어낸다.

> *SC.2: TCAS는 지상 ATC 시스템 또는 다른 항공기의 지상 ATC 시스템으로의 송신을 방해해서는 안 됨 (H5)*

SC.2를 위반하는 원인 시나리오를 식별하는 데 STPA를 사용할 수 있으며 이 정보는 보다 상세한 안전 제약사항인 SC2.1로 세분화될 수 있다.

> *SC2.1 시스템 설계는 지상 기반 2차 감시 레이다, 거리 측정 장비 채널, 그리고 1030/ 1090MHz 주파수 대역에서 작동하는 다른 라디오 서비스를 방해해서는 안 됨(2.5.1)*

안전한 TCAS 설계의 기초가 되는 가정은 그러한 간섭이 절대로 발생하지 않는다는 것이다. 만약 발생한다면 그것이 시스템 설계 또는 동작에 결함이 있다는 것을 나타내는 선행지표이다.

사람은 시간이 지남에 따라 행동이 변하고 설계자가 원래 의도한 것과는 다른 방법으로 자동화를 사용하는 경향이 있다. 따라서 운영자의 행동에 대한 가정은 선행지표를 식별하기 위한 또 다른 중요한 출처(source)가 된다. 예를 들어 H3은 TCAS가 조종사의 통제력 상실을 야기하거나 기여한다는 것인데 H3에서 STPA를 통해 도출한 안전 제약사항 SC.6은 다음과 같다.

> *SC.6: TCAS는 중요한(critical) 비행 단계에서 조종사와 ATC 동작을 혼란스럽게 하거나 항공기 운항을 방해하지 않아야 함 (H3, 2.2.3, 2.19, 2.24.2).*

이 안전 제약사항이 도출된 관련 위험(이 경우 H3)을 식별하는 것 이외에도 이 명세에는 그 위험을 컨트롤하기 위해 사용된, 즉 SC.6을 강제하기 위한 설계의 특징(TCAS 의도 명세서의 2.2.3, 2.19 및 2.24.2)을 이야기하고 있다. 이러한 컨트롤에는 점검해야 할 중요한 가정도 포함되어 있다. 가장 기본적인 가정은 이러한 컨트롤이 위험한 시나리오를 방지하는 데 효과적이며 올바르게 구현되어 있다는 것이다. 예를 들어 STPA 분석을 통해 SC.6을 위반할 수 있다고 식별된 시나리오 중 하나는 조종사가 지상에 있거나 이륙 중일 때 TCAS가 혼란을 불러일으키는 대응 지침(resolution advisories)을 제공하는 것이다. 이 시나리오를 방지하기 위해 이륙과 착륙의 중요 단계에서는 잠재적으로 혼란을 만드

SC6.1 *The pilot of a TCAS-equipped aircraft must have the option to switch to the Traffic-Advisory-Mode-Only where traffic advisories are displayed but display of resolution advisories is inhibited (**2.2.3**).*

Assumption: This feature will be used only during takeoff or in final approach to parallel runways, when two aircraft are projected to come close to each other and TCAS would call for an evasive maneuver.

Addition of the control, i.e., the ability of the pilot to inhibit TCAS resolution advisories by switching to Traffic-Advisory-Only mode, creates another hazardous scenario that must be controlled through pilot procedures, training, etc., and leads to another assumption that should be checked during operation of the system to ensure that pilots are not violating the assumption associated with **SC6.1**.

Other examples of operational requirements that were created to eliminate or control hazardous scenarios are:

OP4*: After the threat is resolved, the pilot shall return promptly and smoothly to his/her previously assigned flight path.*

OP9*: The pilot must not maneuver on the basis of a Traffic Advisory only*

Because these procedures were created to counter specific scenarios that were identified as leading to hazards, they represent a source of assumptions that should be checked to identify hazardous behavior that could lead to an accident. Note that this information about violating assumptions could potentially be collected automatically using FOQA[27] systems.

As another example, in the Uberlingen accident, there were additional causal factors not mentioned earlier. One was that maintenance was being performed on the ATC equipment at the time of the collision, which disabled the air traffic controller's aural conflict alert without him knowing that it was not working. If the air traffic controller had known the alert was disabled, he could have adjusted his behavior. This type of causal factor can be controlled in operational procedures, in this case, the procedures for performing maintenance while the ATC tower is still operating. An additional important assumption, of course, is that such procedures are being followed and this assumption will also need to be checked.

A final source for assumptions that can be used to identify leading indicators is limitations in the design of safety-related controls. These limitations should be documented as they represent important information in the decision about whether and how the system should be deployed. Some TCAS limitations are related to the basic functional requirements, for example:

L2*: TCAS does not currently indicate horizontal escape maneuvers and therefore does not (and is not intended to) increase horizontal separation.*

Other limitations are related to the environmental assumptions, for example:

L1*. TCAS provides no protection against aircraft without transponders or with nonoperational transponders (**EA3**).*

L6*: Aircraft performance limitations constrain the magnitude of the escape maneuver that the flight crew can safely execute in response to a resolution advisory. It is possible for these limitations to preclude a successful resolution of the conflict (**H3, 2.38, 2.39**)*

[27] FOQA stands for Flight Operations Quality Assurance and is sometimes called Flight Data Monitoring (FDM). It refers to programs that collect and monitor data on an aircraft to improve flight safety and efficiency. Other industries have similar monitoring programs but use different names.

는 대응 지침 생성을 조종사가 금지할 수 있는 컨트롤이 설계되었다.

SC6.1 TCAS 장착 항공기의 조종사는 트래픽 지침은 표시되나 대응 지침 표시는 금지되는 트래픽 지침 전용 모드(Traffic-Advisory-Mode-Only)로의 전환 옵션이 있어야 함 (2.2.3)

가정: 이 기능은 이륙 또는 평행활주로로 최종 접근(final approach) 단계에서 항공기 2대가 서로 접근할 것으로 예측되어 TCAS가 회피 기동(evasive maneuver)을 요구할 때만 사용됨

컨트롤의 추가, 즉 조종사가 트래픽 지침 전용 모드로 전환하여 TCAS 대응 지침을 금지할 수 있게 하는 것은 조종사 절차, 훈련 등을 통해 제어되어야 하는 다른 위험한 시나리오를 생성하며, 조종사가 SC6.1과 관련된 가정을 위반하지 않도록 시스템 운영 중에 점검해야 하는 또 하나의 가정을 이끌어 낼 수 있다.

위험한 시나리오를 제거하거나 제어하기 위한 운영 요구사항의 다른 예는 다음과 같다.

OP4: 위협이 해소된 후, 조종사는 지정된 비행경로로 신속하고 원활하게 복귀해야 함
OP9: 조종사는 트래픽 지침(traffic advisory)만을 근거로 기동(maneuver)해서는 안됨

이러한 절차는 위험을 야기하는 특정 시나리오에 대응하기 위해 만들어졌기 때문에 사고로 이어질 수 있는 위험한 행동을 식별하기 위해 점검해야 할 가정의 출처(source)를 나타낸다. 가정 위반에 대한 이러한 정보는 FOQA[27] 시스템을 사용하여 자동으로 수집될 수도 있다.

다른 예시로 Uberlingen 사고에서, 앞서 언급되지 않은 추가적인 원인 요소들이 있었다. 그 중 하나는 충돌 당시 ATC 장비의 유지보수가 진행 중이어서 항공교통관제사의 음성 충돌 경보가 작동하지 않았으나, 항공교통관제사가 이를 알지 못했다는 것이다. 만약 관제사가 경보가 작동하지 않는다는 것을 알았다면 그는 다르게 행동했을 것이다. 이러한 유형의 원인 요소는 운영 절차에서 제어될 수 있으며 이 예시의 경우 ATC 타워 운영 중에 수행되는 유지보수 절차에서 제어될 수 있다. 추가적인 중요한 가정은 물론 그러한 절차가 지켜지고 있으며 절차가 지켜진다는 가정의 점검이 필요하다는 것이다.

선행지표 식별에 사용할 수 있는 가정의 마지막 출처(source)는 안전-관련 컨트롤 설계의 한계이다. 이러한 한계는 시스템을 활용할 것인지 또는 어떻게 활용할지와 같은 결정을 내리는 데 중요한 정보가 되므로 반드시 문서화되어야 한다. TCAS의 한계 중 일부는 기본적인 기능 요구사항과 관련이 있는데 예를 들면 다음과 같다.

L2: TCAS가 현재 수평 탈출 기동(horizontal escape maneuvers)을 표시하지 않으므로 수평 이격거리(horizontal separation)를 증가시키지 않음(및 증가시키려고 하지 않음)

다른 한계는 환경적 가정과 관련 있다. 예를 들어:

L1: TCAS는 트랜스폰더가 없거나 트랜스폰더가 작동하지 않는 항공기는 보호하지 않음 (EA3)
L6: 항공기 성능 한계는 대응 지침에 대한 응답으로 승무원이 안전하게 실행할 수 있는 탈출 기동의 정도를 제한한다. 이로 인해 성공적인 해결이 불가능할 수 있다 (H3, 2.38, 2.39)
L4: TCAS는 위협 항공기의 고도 정확도에 의존한다. 이격 보증(separation assurance)은 침입 항공기 트랜스폰더의 기압 고도계 오차로 인해 저하될 수 있다 (EA5)

---

27 FOQA는 비행 운영 품질보증(Flight Operation Quality Assurance)을 의미하며, 비행 데이터 모니터링(Flight Data Monitoring)이라고도 한다. 비행 안전과 효율성 향상을 위해 항공기의 데이터를 수집하고 모니터링하는 프로그램으로 다른 산업에서도 이름은 다르지만 비슷한 모니터링 프로그램이 있다.

***L4**: TCAS is dependent on the accuracy of the threat aircraft's reported altitude. Separation assurance may be degraded by errors in intruder pressure altitude as reported by the transponder of the intruder aircraft (**EA5**)*

Assumption: This limitation holds for the airspace existing at the time of the initial TCAS deployment, where many aircraft use pressure altimeters rather than GPS. As more aircraft install GPS systems with greater accuracy than current pressure altimeters, this limitation will be reduced or eliminated.

An example assumption related to **L1** is that the operation of aircraft without transponders will be precluded in operations.

Limitations may relate to hazards or hazard causal factors that could not be completely eliminated or controlled in the design: Thus they represent accepted risks.

***L3**: TCAS will not issue an advisory if it is turned on or enabled to issue resolution advisories in the middle of a conflict.*[28]

An implied assumption here is that pilots will, except under unusual circumstances, turn TCAS on before taking off, which can be checked in performance audits.

Finally, limitations may be related to problems encountered or tradeoffs made during system design. For example, TCAS has a high-level, performance-monitoring requirement that led to the inclusion of a self-test function in the system design to determine whether TCAS is functioning correctly. The following system limitation relates to this self-test facility:

***L9**: Use by the pilot of the self-test function in flight will inhibit TCAS operation for up to 20 seconds depending upon the number of targets being tracked. The ATC transponder will not function during some portion of the self-test sequence.*

A safety-related assumption is that this behavior will be rare and therefore not result in frequent periods of non-operation of TCAS and therefore increased risk of an NMAC.

Using the Assumptions to Create an Assumption-Based Leading Indicator Program

The previous section described ways to generate assumption-based leading indicators. The assumptions must be maintained by implementing controls for the assumptions in the system design and checking leading indicators during operation of the system. How can the identified assumptions underlying safety be enforced? Some new terminology is useful in differentiating various cases.

First, in the original system design, actions to maintain assumptions, called *shaping actions*, may be taken to prevent the violation of the assumptions. More specifically, they are used to maintain assumptions, prevent hazards, and control migration to states of higher risk. Shaping actions provide feedforward control[29] over emergent properties and are built into the hierarchical control structure. Examples include physical interlocks to prevent entering a hazardous state; the use of desiccants to prevent corrosion; designing human operations to be easy and hard to omit; and creation of special management structures, such as an independent technical authority, for safety-critical decisions.

[28] The reasoning behind this limitation is beyond the scope of this handbook but arose from a scenario generated by the hazard analysis.

[29] Basic engineering terminology that may not be familiar to readers who are not trained as engineers is defined in Appendix F.

가정: 이 한계는, 많은 항공기가 GPS보다는 기압 고도계를 사용했던 초기의 TCAS 배치 시점의 공역에 대해 유효하다. 현재의 기압 고도계보다 더 높은 정확도를 가진 GPS 시스템을 항공기에 설치함에 따라 이러한 한계는 줄어들거나 사라질 것이다

L1과 관련된 가정의 예는 트랜스폰더가 없는 항공기의 운항은 운항에서 배제된다는 것이다.

한계(limitation)는 설계에서 완전히 제거되거나 제어될 수 없는 위험 또는 위험 원인 요소와 관련이 있을 수 있다. 따라서 한계는 수용된 리스크(accepted risks)를 나타낸다.

L3: 대응 지침이 발행되도록 켜져 있거나 활성화된(enabled) 경우라도 지침이 상충된다면 TCAS는 지침을 발행하지 않을 것이다.[28]

여기에 함축되어 있는 가정은 예외적인 상황을 제외하면 조종사가 이륙 전 TCAS를 켠다는 것이고 이는 성능 감사(performance audit) 시 점검할 수 있다.

마지막으로, 한계는 시스템 설계 과정에서 맞닥뜨리는 문제나 트레이드오프와 관련될 수 있다. 예를 들어 TCAS는 상위 수준의 성능 모니터링 요구사항을 가지므로, 시스템 설계 시 TCAS가 올바르게 작동하는지를 판단하기 위한 셀프-테스트 기능을 구현한다. 다음의 시스템 한계는 이러한 셀프-테스트 기능과 관련된다.

L9: 조종사가 비행 중 셀프-테스트 기능을 사용하게 되면 추적 대상의 수에 따라 최대 20초 동안 TCAS 작동이 금지되며 ATC 트랜스폰더는 셀프-테스트 시퀀스 중 일부에서 작동하지 않을 것이다.

안전-관련 가정은 이런 동작이 드물기 때문에 잦은 TCAS의 비작동(non-operation) 및 이로 인한 NMAC의 위험 증가로 이어지지는 않는다는 것이다.

가정을 사용한 가정 기반 선행지표 프로그램 작성

이전 섹션에서는 가정 기반 선행지표를 생성하는 방법에 대해 설명했다. 시스템 설계 시에 가정에 대한 컨트롤을 구현하고 시스템 운영 중에는 선행지표를 점검함으로써 가정이 유지되도록 하여야 한다. 어떻게 하면 안전의 기반이 되는, 이 식별된 가정들을 강제할 수 있을까? 몇 가지 새로운 용어(terminology)를 이용해 다양한 경우를 구별할 수 있다.

첫 번째는 기존 시스템 설계에서 *쉐이핑 액션(shaping action)*이라고 불리는, 가정을 유지하기 위한 활동을 하여 가정의 위반을 방지하는 것이다. 보다 구체적으로, 쉐이핑 액션은 가정을 유지하고 위험을 방지하며 더 높은 리스크 상태로의 진입을 컨트롤하는 데 사용된다. 쉐이핑 액션은 발현 속성에 대한 피드포워드(feedforward) 컨트롤[29]을 제공하며 계층적 컨트롤 스트럭처 내에 만들어진다. 예로는 위험한 상태로 빠지는 것을 방지하기 위한 물리적인 인터락 장치, 부식을 방지하기 위한 건조제 사용, 사람이 운영하기 쉽고 누락하기 어렵도록 설계하는 것, 독립적인 기술당국과 같은 안전-필수 결정을 위한 특별한 관리 체제를 만드는 것 등이 있다.

*헷징(우발상황) 액션(hedging(contingency) action)*은 가정이 실패하여 조치를 취하는 가능성에 대비하기 위해 사용된다. STAMP 용어에서 헷징 액션은 피드백을 사용하여 시스템 발현 속성을 제어한다. 기본적으로 시스템 동작의 모니터링 계획이나 식별된 시스템 가정(assumption) 변경에 따른 계획을 포함한다. 그러한 모니터링에는 쉐이핑 액션의 효과에 대한 모니터링도 포함된다. 예로는 페일세이프

28 이 한계에 대한 추론은 이 핸드북의 범위를 벗어나지만, 위험 분석에서 생성된 시나리오에 따른 것이다.

29 엔지니어 교육을 받지 않은 독자에게 익숙하지 않는 기본적인 엔지니어링 용어는 부록 F에 있다.

Hedging (contingency) actions are used to prepare for the possibility that an assumption will fail and take action accordingly. In STAMP terminology, hedging actions use feedback to control emergent system properties. In essence, they involve planning for monitoring the system behavior and responding to particular identified changes in the assumptions underlying the system controls. Such monitoring will include monitoring of the effectiveness of the shaping actions. An example is fail-safe design (e.g., protection and shutdown systems) that anticipates the potential for shaping actions to be unsuccessful in eliminating assumption violations in some cases.

Information for creating shaping and hedging actions can come from the STPA causal scenarios and the assumptions generated from these scenarios.

Checking assumptions need not be done continually. Some assumptions can only be violated in the presence of particular changes in the system or the environment. *Signposts* are points in the unfolding future where changes in the current controls (shaping and hedging actions) may be necessary or advisable. Signposts are points in time or specific future events or changes that trigger the checking of assumptions, including assumptions about the continued effectiveness of the controls. Management of change policies usually include various types of signposts. As an example, changes in the airspace, such as reduced separation or increased traffic density, might be identified as signposts for TCAS.

Shaping and hedging actions are designed into the safety control structure, both technical and organizational. Signposts are identified during the system design and development process and specific responses are created and specified. More general *assumption checking* during system operation involves specific checks to determine whether the assumptions underlying the design of the system are still valid. In assumption checking, risk managers and controllers monitor the system during its operation to determine whether the original assumptions are still valid. Such monitoring might include monitoring the response to signposts or perhaps changes and failures of assumptions that are known not to have been adequately or completely handled by shaping and hedging actions or have not been enforced at all by shaping and hedging actions. Checking might involve performance audits, surveys, and automatically collected FOQA or other data.

Some Ways to Enforce Assumption-Based Leading Indicators

- Shaping actions to prevent violation of the assumptions
- Hedging actions to prepare for failure of an assumption
- Assumption checking during operations
  - Sign posts to trigger specific checks
  - Checking during system operation (periodic or continual)
    - Performance audits
    - Surveys
    - Automatically collected data

In the previous chapter on organizational analysis, one of the conclusions of the example analysis was that technical independence was a desirable goal for the organization. In setting up such an independent technical authority, shaping actions need to be designed as well as feedback channels for the new controls. In addition, hedging actions might be established in case project managers simply ignore the independent authorities and make unwise decisions on their own. A third possibility, and one that is common, is that independence works well at the start, but it slowly is undermined by a series of small decisions or changes in circumstances over time. This latter case of degradation has recently been documented by the GAO for the NASA Independent Technical Authority established after the Columbia Space Shuttle loss. To find these problems early and not wait until near misses or worse occur, the

(fail-safe) 설계(예: 보호 및 셧다운 시스템)가 있는데 이는 쉐이핑 액션이 경우에 따라 가정의 위반 제거에 실패하는 가능성을 예측한다.

쉐이핑 액션 및 헷징 액션을 생성하기 위한 정보는 STPA 원인 시나리오 및 시나리오에서 생성된 가정으로부터 얻을 수 있다.

가정을 지속적으로 점검할 필요는 없다. 일부 가정은 시스템이나 환경에 특정 변화가 있는 경우에만 위반된다. *싸인포스트(Signpost)*는 컨트롤(쉐이핑 및 헷징 액션)의 변경이 필요하거나 변경이 권장되는 미래의 시점(point)을 말한다. 즉, 가정(컨트롤의 지속적인 효과에 대한 가정 포함)의 점검을 요하는 시간, 특정한 미래의 이벤트 또는 변경의 시점이다. 변경 정책 관리에는 일반적으로 다양한 유형의 싸인포스트가 포함된다. 예를 들어 이격거리의 감소나 교통밀도 증가와 같은 영공의 변화는 TCAS의 싸인포스트로 식별될 수 있다.

쉐이핑 및 헷징 액션은 기술 및 조직의 안전 컨트롤 스트럭처에 설계된다. 싸인포스트는 시스템 설계 및 개발 과정에서 식별되고, 이에 대한 대응(response)이 생성 및 명세된다. 시스템 운영 과정에서의 보다 일반적인 *가정 점검(assumption checking)*은 시스템 설계의 기초가 되는 가정이 여전히 유효한지를 확인하는 검사를 포함한다. 가정 점검에서 리스크 관리자와 컨트롤러는 원래의 가정이 여전히 유효한지를 결정하기 위해 운영 중인 시스템을 모니터링 한다. 이러한 모니터링은 싸인포스트에 대한 대응을 모니터링하는 것을 포함할 수 있으며, 쉐이핑 및 헷징 액션에서 적절하거나 완벽하게 처리되지 않았거나 전혀 시행되지 않았다고 판단되는 가정의 변경이나 실패도 포함할 수 있다. 점검에는 성능 감사, 설문 조사, 자동으로 수집된 FOQA 또는 기타 다른 데이터가 포함될 수 있다.

가정 기반 선행지표를 강제(enforce)하는 방법

- 가정 위반을 예방하기 위한 쉐이핑 액션(shaping actions)
- 가정 실패에 대비하기 위한 헷징 액션(hedging actions)
- 운영 중 가정 점검(assumption checking)
  - 특정한 점검을 유발(trigger)하는 싸인포스트(signpost)
  - 시스템 운영 중의 점검(정기적 또는 계속적)
    - 성능 감사
    - 설문 조사
    - 자동으로 수집된 데이터

조직 분석에 대한 이전 챕터에서, 예제 분석의 결론 중 하나는 기술적인 독립(technical independence)이 조직의 바람직한 목표라는 것이다. 이러한 독립적인 기술 통제권한을 설정하는 과정에서 새로운 컨트롤에 대한 피드백 채널뿐만 아니라 쉐이핑 액션 또한 설계되어야 한다. 또한 프로젝트 관리자가 독립적 통제권한을 무시하고 현명하지 못한 결정을 내리는 경우를 대비한 헷징 액션이 수립될 수 있다. 또한 흔하게 독립성은 처음에는 잘 유지되지만 일련의 작은 결정이나 시간이 흐르면서 발생되는 상황 변화에 의해 점점 약화될 수 있다. NASA의 독립 기술당국인 GAO(Government Accountability Office, 콜롬비아 우주 왕복선 사고 이후 설립)는 이러한 독립성의 저하 문제를 최근 문서화하였다. 독립성 저하 문제를 일찍 발견하고 니어 미스(near miss)나 더 나쁜 상황이 발생하기 전에 조치하기 위해서는 독립 당국 설립(이 경우) 시의 가정을 문서화하고 점검해야 한다. 이러한 점검은 가능하다면 지속적으로, 지속적인 수행이 불가능하다면 주기적으로 수행되어야 한다. 이를 통해 독립적인 의사 결정 구조의 저하(degradation)로 시스템이 더 심각한 수준의 리스크에 빠지는 시점을 식별할 수 있다. 물론 손실이 발생한 이후에 사고 및 사건을 조사하는 것은 다소 늦지만, 실패한 가정을 식별

assumptions underlying the establishment of the independent authority (in this case) need to be documented and checked, either continually if feasible or periodically if not, to identify when the system is drifting to a higher state of risk through the degradation of the independent decision-making structure. The investigation of incidents and accidents are, of course, another place where failed assumptions can be identified, although it is rather late to do so after a loss.

Each leading indicator identified by STPA and other means should be documented by:

- The associated assumption(s),
- How it will be checked
- When it will be checked
- The action(s) to take if the indicator is true (the assumption is violated).

The actual structure of this documentation will depend on the organization and their risk management system.

Integrating Leading Indicators into Your Risk Management Program

Identifying leading indicators needs to be accompanied by the design of a risk management structure that can and will act when the leading indicators show that action is necessary. The leading indicators provide a foundation for risk management. The overall risk management process involves ensuring that not only are the risks initially handled in the new system design but also that leading indicators of failure of one or more of the assumptions and constraints are identified so that action can be taken.

General management principles apply in designing a risk management system that uses leading indicators, but some specific features are especially important. Even if the information is clear and available about elevated risk in the system, many organizations are still slow to process these signals and respond. Creating and monitoring leading indicators will not help if their appearance does not result in appropriate action being taken. Too often heuristic biases intervene, particularly defensive avoidance,[30] and clear leading indicators are ignored until too late to avoid serious negative consequences. In additional to psychological biases, organizational culture and politics can cause problems in designing and reacting to leading indicators.

Successful creation and use of leading indicators will require ways to control these psychological and cultural biases. For example, to encourage effective action, leading indicators have to be carefully integrated into the overall risk management program: Not only must they be communicated to appropriate decision makers, but detailed action plans for critical scenarios should be developed and triggers specified for implementing those action plans. Responsibilities need to be assigned for checking the existence of the leading indicators and for following through if problems are found.

As indicated in the previous section, every leading indicator should indicate when and how it will be checked and must have an action associated with it. Required action plans should be created *before* the assumptions are found to be invalid in order to lessen denial and avoidance behaviors and overcome organizational and cultural blinders. Responsibility for monitoring and action may need to be assigned to an independent organization and not to the program and project managers and those under

[30] Psychologists have written extensively about the biases humans show in assessing risk. These biases impact the way we design leading indicators and how we react to their occurrence. While all these biases are important, one of the most relevant in terms of reacting to leading indicators is called “defensive avoidance.” This type of bias may be reflected in the downgrading of the perceived accuracy of a leading indicator or reflected in people’s ability to take them seriously and to accept that risk may be increasing. Defensive avoidance is based on the common psychological tendency to rationalize and avoid consideration of a topic that is stressful or conflicts with other pressing goals.

할 수 있는 또 다른 방법이기도 하다.

STPA를 비롯한 여러 방법에 의해 식별된 각 선행지표는 다음을 문서화되어야 한다.

- 관련된 가정(들)
- 점검하는 방법
- 점검되어야 하는 시점
- 지표가 참일 경우(가정이 위반됨) 취해야 할 행동

이 문서의 실제 구조는 조직 및 조직의 리스크 관리 시스템에 따라 다르다.

선행지표를 리스크 관리 프로그램에 통합

선행지표 식별에는 선행지표를 통해 조치가 필요하다는 것을 알게 되었을 때 그것을 조치할 수 있는 리스크 관리 구조의 설계가 수반되어야 한다. 선행지표는 리스크 관리 기반을 제공한다. 전반적인 리스크 관리 프로세스는 새로운 시스템 설계에서 리스크가 처음부터 다루어지는 것을 보장할 뿐만 아니라, 하나 이상의 가정 및 제약사항 실패에 대한 선행지표가 식별되고 조치되는 것을 보장한다.

선행지표를 사용하는 리스크 관리 시스템을 설계하는 데에는 일반적인 관리 원칙이 적용되지만 몇 가지 원칙들은 특히 중요하다. 증가된 리스크에 대한 정보가 명확하고 활용가능한 경우에도 많은 조직들이 이러한 신호를 처리하고 대응하는 속도가 여전히 느리다. 선행지표가 발견되었을 때 적절한 조치가 취해지지 않는다면 선행지표를 만들고 모니터링 하는 것은 도움이 되지 않는다. 또, 휴리스틱한 편견이 너무 자주 방해가 되는데 특히 방어적 회피(defensive avoidance)[30]가 그러하며, 명백한 선행지표가 너무 늦은 시점까지 무시되어 심각한 부정적 결과를 초래하기도 한다. 심리적인 편견 외에도 조직의 문화와 정치는 선행지표를 설계하고 대응하는 데 문제를 일으킬 수 있다.

선행지표를 성공적으로 만들고 사용하려면 이러한 심리적, 문화적 편견을 통제할 방법이 필요하다. 예를 들어 효과적인 행동을 장려하기 위해서 선행지표는 전체 리스크 관리 프로그램에 신중히 통합되어야 한다: 선행지표는 적절한 의사 결정권자에게 전달되어야 할 뿐 아니라 중요 시나리오에 대한 세부 실행계획을 수립하고 그 계획을 시행하기 위한 트리거를 지정해야 한다. 선행지표 존재 여부 점검과 문제가 발견되었을 때의 후속 조치에 대한 책임 할당도 필요하다.

이전 섹션에서 설명한 것처럼 모든 선행지표에는 점검이 필요한 시점 및 방법과 그와 관련된 조치가 있어야 한다. 가정이 유효하지 않다고 밝혀지기 전에는 거부(denial)와 회피(avoidance)를 줄이고 조직과 문화에 방해되는 것들을 극복할 수 있도록, 필요한 행동 계획을 수립해야 한다. 모니터링 및 조치에 대한 책임은 독립적인 조직에 할당되어야 하며 프로그램, 프로젝트 관리자, 그리고 상충되는 압력을 받는 담당자들에 할당되지 않아야 한다. 주기적으로 선행지표 목록을 재검토하고 필요할 경우 업데이트해야 한다. 시간이 지남에 따라 현재 지표를 경험에 비추어 재평가하고, 식별된 효과의 부족을 진단하는 지속적인 개선 프로세스가 만들어져야 한다. 예를 들어 선행지표에 의해 포착되지 않은 부정적 결과가 발생하는 경우 왜 선행지표가 이벤트의 발생을 방지하기 위해 제때 문제를 파악하지 못했는지, 파악했다면 왜 효과적인 조치가 취해지지 않았는지에 대한 분석이 수행되어야 한다. 그런 다음 이 정보를 이

30 심리학자들은 사람들이 리스크를 평가할 때 보이는 편견에 대해 광범위한 글을 썼다. 편견은 선행지표를 설계하는 방식과 발생했을 때의 우리의 대응 방법에 영향을 미친다. 이러한 모든 편견이 중요하기는 하지만, 선행지표 대응에 가장 관련된 용어 중 하나는 "방어적 회피(defensive avoidance)"이다. 이 유형의 편견은 선행지표 인식 정확도의 수준 저하나, 사람들이 리스크를 심각하게 받아들이고 리스크가 증가할 수 있다고 인정하는 능력에 영향을 준다. 방어적 회피는 스트레스가 되거나 다른 압박되는 목표와 충돌하는 주제를 합리화하거나 고려하지 않는 일반적인 심리적 경향에 기반한다.

conflicting pressures. Periodically, the list of leading indicators needs to be revisited and, if necessary, updated. A continuous improvement process should be created that reevaluates the current indicators over time in the light of experience and diagnoses any identified lack of effectiveness. For example, if a negative consequence occurs that was not signalled by leading indicators, an analysis should be made of why the leading indicators did not identify the problems in time to prevent the events or, if they did, why effective action was not taken. The entire leading indicators program should then be evaluated and improved using this information. Was a leading indicator identified but not checked? Or was an appropriate one never identified? Not only should the answer to these questions be used to improve the leading indicators themselves, but the reason for the omission or inadequate checking should be determined. That reason may involve a flaw in the process and lead to identifying other important programmatic assumptions that are not being checked.

Feasibility and Final Thoughts

Assuming that all of this sounds good, there is still the question of the feasibility of creating and checking leading indicators. While there are seemingly a very large number of potential leading indicators that result from the STPA process, not all of these need to be checked or to be checked frequently. Hedging and contingency actions are usually built into the safety control structure to control most of the identified causal factors. Many assumptions can be enforced by appropriate design, which should be the highest priority in dealing with the results of hazard analysis and is most feasible when hazard analysis is done in the early concept and design stages as described in the chapter on integrating STPA into system engineering.

Signposts will limit checking to the occurrence of certain triggering events. Documentation of critical assumptions should be part of any system development and specification activity. Documenting assumptions is not only important to identify leading indicators and improve safety, but also for the normal maintenance and evolution of the system. Even during initial development, people forget why they made certain decisions unless they are documented. In the case of our use of this approach to leading indicators on real problems, such as TCAS, we have found that documentation of critical assumptions are important to include in order to reduce the amount of development and testing time and to maintain the system without introducing hazards in the process. Note that I created the TCAS intent specification that included the safety-critical (and other) assumptions alone in my “spare time” over a period of a few months.

Could there still be a very large number of assumptions? Of course there may be. The usual approach to dealing with a large number of causal factors in safety is to try to reduce them by using probabilities. The problem is that there is no way to predict the probability of future events in any accurate way. Often the use of human (both engineering and management) expertise and knowledge is undervalued in comparison to a number, especially if that number is produced by a computer. But while numbers may appear more accurate, they are often only wild guesses based on little real data. Psychologists have found that, in fact, humans are very bad at estimating the probability of future events.[31] While certainly you are free to assign probability or likelihood to the leading indicators, we have found that risk levels based on engineering, management, psychological, and sociological knowledge produces better results. An additional consideration is that the independent review of non-numerical risk assessments is more likely to be “deeper” than numerical ones. People tend to believe that numbers are well founded and are less likely to challenge them unless they are wildly off the mark (and sometimes not even then).

[31] See, in particular, the work on *heuristic biases* by Tversky and Kahneman.

용하여 전체 선행지표 프로그램을 평가하고 개선해야 한다. 선행지표가 식별되었으나 점검되지 않았는가? 또는 적절한 선행지표가 식별조차 되지 않았는가? 이러한 질문에 대한 대답은 선행지표 자체를 개선하는 데 사용될 뿐만 아니라 누락되거나 부적절한 점검에 대한 이유를 밝히는 데에도 사용된다. 그 이유를 통해 프로세스의 결함을 찾을 수도 있고 점검되지 않은 다른 중요한 프로그램의 가정을 식별할 수도 있다.

타당성(feasibility) 및 최종 고찰

이 모든 것들을 수긍하더라도 선행지표를 만들고 점검하는 타당성에 대한 의문은 여전히 남아있다. STPA 프로세스를 통해 도출된 잠재적 선행지표는 매우 많지만 모든 것들을 점검하거나 자주 점검해야 할 필요는 없다. 헷징(Hedging) 및 우발상황 조치(contingency action)는 일반적으로, 식별된 원인 요소의 대부분을 제어하기 위해 안전 컨트롤 스트럭처에 설계된다. 적절한 설계를 통해 많은 가정을 강제(enforce)할 수 있는데 이는 위험 분석 결과를 처리함에 있어 최우선이 되어야 하며, STPA를 시스템 엔지니어링에 통합하는 장(3장)에서 소개된 바와 같이 초기 개념 및 설계 단계에서 위험 분석을 수행할 때 적용하는 것이 가장 타당하다.

싸인포스트는 특정 트리거 이벤트의 발생에 대한 점검을 제한한다. 필수적인 가정에 대한 문서화 작업이 시스템 개발 및 명세 활동에서 수행되어야 한다. 가정을 문서화하는 것은 선행지표를 식별하고 안전을 향상시키는 데 중요할 뿐만 아니라 시스템의 일상적인 유지보수 및 진화(evolution)에도 중요하다. 문서화하지 않으면 사람들은 초기 개발 단계에서조차 그들이 왜 그러한 결정을 내렸는지를 잊어버린다. TCAS와 같이 실제 문제에 대한 선행지표에 이러한 접근법을 사용하는 경우 필수적인 가정을 문서화에 포함하는 것이, 시스템에 위험을 유입시키지 않으면서 개발 및 테스트 시간을 대폭 줄이는 데 매우 중요하다는 것을 알게 되었다. 필자는 몇 달 간의 "여유 시간" 동안 혼자서 안전-필수(및 기타) 가정을 포함하는 TCAS 의도 명세서를 작성하였다.

여전히 많은 가정이 있을까? 물론 있을 수 있다. 안전에 대한 많은 원인 요소를 처리하는 일반적인 접근법은 확률을 사용하여 그것들을 줄이는 것이다. 문제는 미래 이벤트의 확률을 정확하게 예측하는 방법이 없다는 것이다. 때로는 사람(엔지니어링 및 관리 영역 모두)의 전문성과 지식의 사용이 특히 컴퓨터에 의해 생성된 수와 비교될 때 저평가되는 경우가 있다. 숫자가 더 정확한 것처럼 보일 수 있지만, 실제 데이터가 거의 없는 경우에는 종종 어림짐작에 지나지 않는다. 심리학자에 의하면 사실 사람은 미래 이벤트[31]를 잘 예측하지 못한다고 한다. 선행지표에 발생확률이나 발생 가능성을 할당하는 것은 분명 자유지만 우리는 엔지니어링, 관리, 심리학 및 사회학적 지식에 기반한 리스크 수준이 더 나은 결과를 가져온다는 사실을 발견했다. 추가로 고려할 것은 비 수치적(non-numerical) 리스크 평가의 독립적 검토가 수치적 리스크 평가보다 '더 깊이 있다'는 점이다. 사람들은 숫자에는 근거가 충분하며 완전히 빗나간 경우가 아니라면(그리고 때로는 그렇지 않다고 하더라도) 잘못될 여지가 적다고 믿는 경향이 있다.

[31] Tversky와 KahnemanSee의 휴리스틱 편견(heuristic biases)에 대한 연구 결과를 참조

Chapter 7: Safety Management Systems

Nancy Leveson

The previous chapter describes how to use STAMP (systems thinking principles) and STPA to analyze a sociotechnical system. This chapter concentrates on the design and analysis of a specific type of sociotechnical system: the safety management system or SMS. Every organization creating or operating safety-critical systems has an SMS, just as every such organization has a safety culture, although the SMS and safety culture may be more or less helpful (and sometimes even harmful) in achieving societal and organizational safety goals.

This chapter describes how systems thinking (STAMP) and STPA can be used in creating a new SMS or in analyzing and improving an existing one. While STPA is mostly addressed indirectly here, a safety engineering program based on STPA will only be effective if it is used within a supportive management structure. The goal of this chapter is to assist in designing such a structure.

What is an SMS?

The goal of an SMS should be to proactively control (manage) safety in every aspect of the organization. For organizations that produce products, the SMS manages safety in the product development process and in the product itself. For service industries, safety must be designed into and managed in the workplace and the services provided. Because sociotechnical systems are rarely perfect from the beginning and the world changes over time, there must be an effective learning process in place that is continually learning from experience and improving the workplace and the products and services as well as the management system itself.

There is no one design of an effective safety management system. The goal of the design will depend on such factors as the type of organization (product development or service providers), the societal culture in which the organization resides and the organizational structure and culture, the inherent safety of the product or service, environmental factors, the other goals the organization wants to achieve and the regulatory environment. Some groups, such as the FAA, have specified one approach to risk management, but that may not be an appropriate solution for everyone, and risk management is only a subset of what is required in a safety management system. In addition, particular safety management system rules imply underlying cultural values that may not be the best ones for a particular group or organization in achieving high safety levels.

There are, however, some general characteristics of a successful SMS. The systems thinking approach to designing an SMS, described in this chapter, is the result of several decades of the author's experience in system safety engineering, in reading and writing accident reports and in identifying the factors involved, as well as helping organizations to prevent accidents. Some industries and organizations have lots of accidents, others have few, and some do not have accidents at all (e.g., SUBSAFE, the U.S. Navy nuclear submarine safety program[32]). Certain factors stand out as critical with respect to which of these three categories an organization or even an industry falls. Some of the most effective ways to control safety may not be practical for everyone. Designing an SMS is a system engineering problem: The goal should be to design and operate an SMS that is as effective as is feasible to implement in your organization.

[32] SUBSAFE focuses on two safety goals: (1) watertight integrity of the submarine's hull and (2) operability and integrity of critical systems to control and recover from a flooding emergency.

7장: 안전 관리 시스템

Nancy Leveson

이전 장에서는 STAMP(시스템 사고 원리, system thinking principles)와 STPA를 사용하여 사회기술적(sociotechnical) 시스템을 분석하는 방법에 대해 설명하였다. 이번 장에서는 특정 유형의 사회기술적 시스템인 안전 관리 시스템(SMS, Safety Management System)의 설계 및 분석에 중점을 두어 설명한다. SMS와 안전 문화가 사회 및 조직의 안전 목표를 달성하는 데 어느 정도 도움이 되거나 그렇지 않을 수도 있지만(때로는 심지어 해로운 경우도 있음), 안전-필수 시스템을 만들거나 운영하는 모든 조직에는 안전 문화가 있는 것과 마찬가지로 안전 관리 시스템(SMS)이 있다.

이 장에서는 시스템 사고(thinking)(STAMP) 및 STPA를 사용하여 새로운 SMS를 만들거나 기존 SMS를 분석하고 개선하는 방법에 대해 설명한다. 여기에서 STPA는 대개 간접적으로 다루어지나, STPA 기반 안전 엔지니어링 프로그램은 지원 관리 구조(supportive management structure)에서 사용되는 경우에만 효과적일 것이다. 이 장의 목표는 그러한 지원 관리 구조를 설계하는 데 도움을 주는 것이다.

SMS란 무엇인가?

SMS의 목표는 조직의 모든 측면에서의 안전을 사전에 제어(관리)하는 것이다. 제품 생산 기업의 경우 SMS는 제품 개발 프로세스와 제품 자체에 대한 안전을 관리한다. 서비스 산업의 경우 제공되는 서비스와 작업장에 대한 안전이 설계되고 관리되어야 한다. 사회기술적 시스템은 처음부터 완벽한 경우가 거의 없으며 시간이 지남에 따라 세상이 변하기 때문에 효과적인 학습(learning) 프로세스가 있어야 한다. 이 프로세스를 통해 지속적인 경험으로부터 학습하고, 관리 시스템 자체뿐만 아니라 작업장, 제품, 그리고 서비스를 개선할 수 있다.

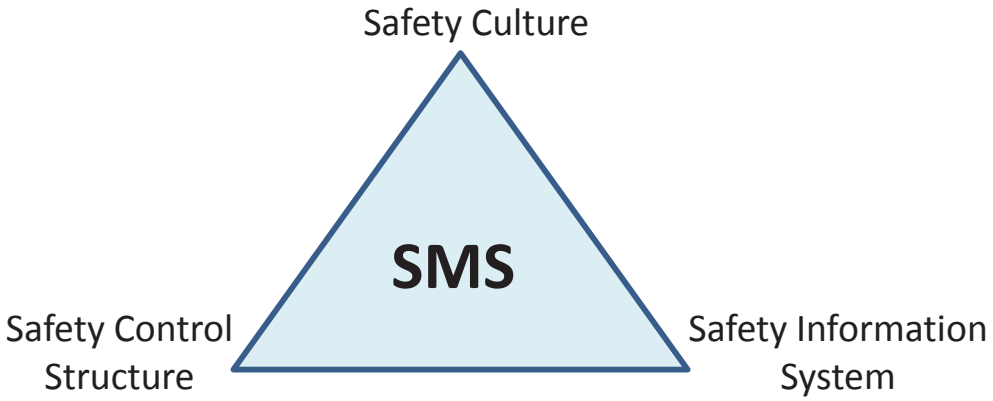
효과적인 안전 관리 시스템을 위한 하나의 설계 방법은 존재하지 않는다. 설계의 목표는 여러 요소에 의해 좌우되는데 조직의 유형(제품 개발 또는 서비스 제공 업체), 조직이 상주하는 사회 문화 및 조직 구조와 문화, 제품이나 서비스 고유의 안전성, 환경 요인, 조직이 달성하고자 하는 다른 목표 및 규제 환경에 따라 달라진다. FAA(Federal Aviation Administration)와 같은 일부 그룹에서는 리스크 관리에 대한 한 가지 접근 방식을 정의했지만 그 방법이 모두에게 적절한 솔루션은 아니며 리스크 관리는 안전 관리 시스템에서 필요한 것들 중 일부일 뿐이다. 또한 특정한 안전 관리 시스템의 규칙은 내재된 문화적 가치를 의미하는데 이 문화적 가치는 특정 그룹이나 조직이 높은 안전 수준을 달성하기 위한 최선의 방법이 아닐 수도 있다.

그럼에도 불구하고 성공적인 SMS에 대한 일반적인 특성은 있다. 이 장에서 설명하는 SMS 설계에 대한 시스템 사고(thinking) 방식은 시스템 안전 엔지니어링, 사고(accident) 보고서의 열람 및 작성, 관련 요인 식별 및 사고 방지를 위한 조직 지원 등 저자의 수십 년 경험의 결과이다. 어떤 산업 및 조직에서는 사고가 많이 발생하고 어떤 곳은 적으며 어떤 곳에서는 사고가 전혀 발생하지 않는다(예: 미국 해군 원자력 잠수함 안전 프로그램인 SUBSAFE)[32]. 조직 또는 심지어 산업에서 세 가지 요소(역주: 안전 문화, 안전컨트롤 스트럭처, 안전정보시스템) 중 일부가 특별히 중요한 것으로 부각될 수 있다. 안전을 제어하는 가장 효과적인 방법일지라도 이것이 모두에게 실용적이지는 않을 수 있다. SMS의 설계는 시스템 엔지니어링 문제이다: 그 목표는 조직에서 구현할 수 있을 만큼의 효과적인 SMS를 설계하고 운영하는 것이어야 한다.

---

[32] SUBSAFE는 2가지 안전 목표에 집중한다: (1) 잠수함 선체에 물이 새지 않는 무결성, (2) 충수(充水) 비상상태를 통제하고 복구하기 위한 필수(critical) 시스템의 운영가능성 및 무결성

There are several components of the organization that are most important in managing safety: the culture, the safety control structure (management), and the safety information system. These three components cannot be considered in isolation, though. The systems approach suggests that these must all be part of a coherent and consistent whole in order to be most effective. Culture defines what desirable and acceptable behavior is and how decisions should be made. The management or control structure determines how that culture will be implemented in the organization. Finally, the safety information system provides the information necessary to make the management structure successful in achieving the desired safety culture: even the best of intentions will not suffice without the appropriate information to carry them out.



Safety Culture

People define safety culture in different ways. I like Edgar Shein's definition of safety culture:

Safety culture is the values and assumptions under which safety-related decisions are made.

Figure 6.1 shows Shein's three levels of organizational culture. The bottom level is the essence of the culture. The values and deep cultural assumptions form the basis for creating organizational rules, policies, and practices. These documents, in turn, describe how the surface level cultural artifacts (e.g., hazard analyses, accident investigation reports, etc.) are produced.

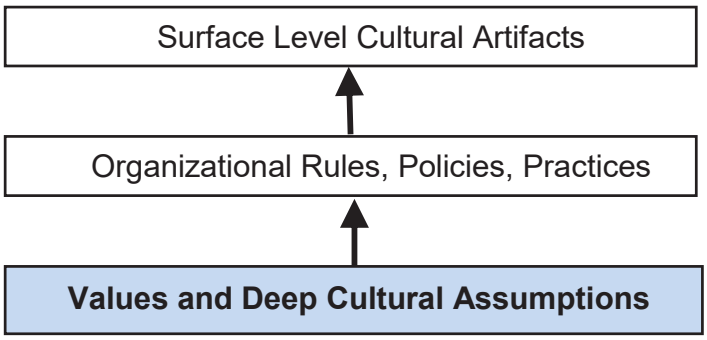
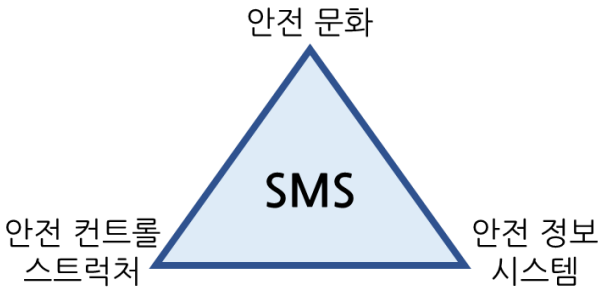


Figure 6.1: Shein's Three Levels of Organizational Culture

The middle and top levels are not the culture; they merely reflect the organizational culture. While attempting to make changes only at the top two levels may temporarily change behavior and even lower risk over the short term, superficial fixes at these levels that do not address the set of shared values and social norms on the bottom level are likely to be undone and become ineffective over time.

안전을 관리하는 데 가장 중요한 조직의 구성요소 몇 가지는 문화, 안전 컨트롤 스트럭처(관리), 그리고 안전 정보 시스템(safety information system)이다. 이 세 가지 구성요소는 서로 분리하여 고려할 수 없다. 시스템 접근법(system approach)에서는, 가장 효과적이기 위해서 이 요소들이 논리정연하고 일관된 완전체의 일부여야 한다고 제안한다. 문화는 바람직하고 수용 가능한 행동이 무엇인지, 그리고 어떻게 결정해야 하는지를 정의한다. 관리 또는 컨트롤 스트럭처는 문화가 조직 내에서 어떻게 실행될지 결정한다. 마지막으로, 안전 정보 시스템은 원하는 안전 문화를 달성하기 위한 성공적인 관리 구조를 만드는 데 필요한 정보를 제공한다: 의지가 아무리 높더라도 이를 실행하기 위한 적절한 정보 없이는 충분하지 않다.



안전 문화(safety culture)

사람들은 안전 문화를 다양한 방식으로 정의한다. 필자는 에드거 샤인(Edgar Shein)의 안전 문화 정의를 선호한다.

안전 문화는 안전-관련 결정을 내릴 때의 가치(values)와 가정(assumptions)이다.

그림 7.1은 샤인의 세 가지 수준의 조직 문화를 보여준다. 제일 아래 수준은 문화의 본질이다. 가치와 깊은 문화적 가정은 조직의 규칙, 정책 및 관행을 만드는 토대를 형성한다. 이러한 문서들은, 결국, 표면 수준의 문화적 산출물(예: 위험 분석, 사고 조사 보고서 등)이 어떻게 만들어지는지 설명한다.

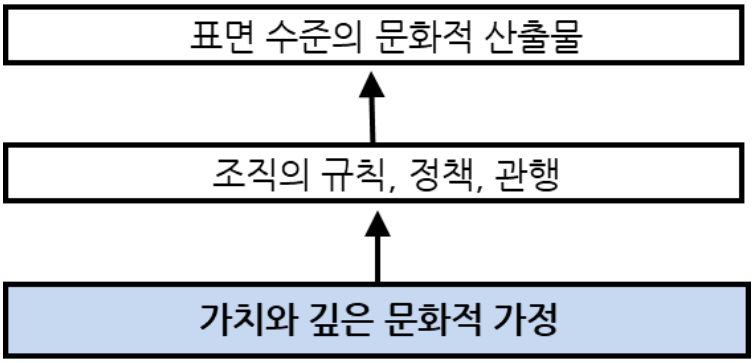


그림 7.1: 샤인(shein)의 세 가지 수준의 조직 문화

중간 및 제일 위 수준은 문화가 아니다; 그것은 단지 조직 문화를 반영할 뿐이다. 상위 2개 수준에서의 변화 시도로, 일시적으로 행동이 변화하고 단기적으로 리스크가 줄어들 수 있지만 이러한 수준에서의 표면적인 수정은 제일 아래 수준에서 공유된 가치 및 사회적 규범을 다루지 못하여 실패하거나 시간이 지남에 따라 효과가 없어질 것이다. 동시에, 문화가 내재된 환경을 바꾸지 않은 채 문화를 바꾸려는 노력 또한 실패로 끝날 것이다.

At the same time, trying to change culture without changing the environment in which it is embedded is also doomed to failure.

While sometimes people talk about "creating" a safety culture, there always exists some sort of safety culture, although the existing one may not be very useful in promoting safety. There is no such thing as a cultureless organization, industry, or society.

Often historical and environmental factors are key in creating the existing safety culture. For example, William Boeing, when building the commercial aviation industry, was faced with the fact that improved safety would be required to sell aircraft and air travel. In 1955, only 20% of the U.S. citizens were willing to fly—aircraft crashes were a relatively common occurrence compared with today. In contrast, the nuclear power industry was initiated by passage of the Price-Anderson Act in 1957 whereby they agreed to be tightly regulated by government in exchange for limited liability in case of accidents. In some industries, the safety culture was the result of strong individual leadership such as Admiral Hyman Rickover in the nuclear navy. Differences in the way safety is handled exist in each of these industries as a result.

While it is difficult to define a "good" safety culture vs. a "poor" one, there are some industries and organizations that have more accidents and major losses than others. The safety culture in organizations having relatively high accident rates tend to have one or more of the following characteristics:

- Culture of Risk Acceptance: This culture is based on the assumptions that accidents are inevitable and that nothing much can be done to prevent them beyond exhorting everyone to be careful. Accidents are considered to be the price of productivity. Often this assumption is accompanied by the belief that everyone should be responsible for safety—their own and others—and that accidents result from a lack of responsible behavior on the part of individuals. The belief is prevalent that if only everyone would act responsibly and safely, accidents would be reduced.
- Culture of Denial: In a culture of denial, risk assessment is often unrealistically low, with credible risks and warnings being dismissed without appropriate investigation: Management only wants to hear good news so that is what they are told. The focus is on showing the system is acceptably safe, not on identifying the ways it might be unsafe.
- Culture of Compliance: Focus here is on complying with government regulations. The underlying cultural belief is that complying with regulations will lead to acceptable results. Because regulatory agencies tend to focus on certifying that a product or service is safe, after the fact assurance is emphasized and often extensive "safety case" arguments are produced with little or no impact on the actual product or process safety.
- Paperwork Culture: This culture rests on the belief that producing lots of documentation and analysis paperwork leads to safe products and services. Piles of paper analyses are produced but they have little real impact on design and operations. The bulk of the safety-related paperwork may be produced by a group that is independent from and have little interaction with those who are designing and operating the products, implementing the processes, or providing the services.
- Culture of "Swagger": Safety is for sissies. Real men thrive on risk.

The features of a good safety culture are harder to specify, but they often include such things as: openness about safety and the safety goals, a willingness to hear bad news, an emphasis on doing what is necessary to increase safety rather than just complying with government regulations or producing a lot of paperwork, and believing that safety is important in achieving the organization's goals. In an effective safety culture, employees believe that managers can be trusted to hear their concerns about

종종 사람들이 안전 문화를 "창조"한다고 말하지만 어떤 종류의 안전 문화는 항상 존재한다. 물론 존재하는 안전 문화가 안전을 고취하는 데 매우 도움이 되지 않을 수도 있다. 문화가 없는 조직, 산업 또는 사회는 존재하지 않는다.

기존의 안전 문화를 창출하는 데 있어서 때론 역사적, 환경적 요소가 중요하다. 예를 들어, 윌리엄 보잉(William Boeing)은 상용 항공기 산업을 구축할 당시 항공기 및 항공 여행 상품을 판매하기 위해서는 안전 수준을 높일 필요가 있다는 사실에 직면했다. 1955년에는 미국 시민의 20%만이 항공기를 이용할 의사가 있었다 — 항공기 충돌은 오늘날에 비해 상대적으로 흔히 있는 일이었다. 이와는 대조적으로, 원자력 산업은 1957년 프라이스 앤더슨(Price- Anderson) 법안이 통과되면서 시작되었으며 사고 발생 시 유한 책임(limited liability)에 대한 대가로 정부로부터 엄격한 규제를 받기로 합의했다. 일부 산업에서의 안전 문화는 원자력을 보유한 해군(nuclear navy)의 하이만 릭오버(Hyman Rickover) 해군제독과 같은 강력한 개인 리더십의 결과였다. 결과적으로 각 산업 분야에서 안전이 처리되는 방식에는 차이가 있다.

"좋은" 안전 문화와 "좋지 않은" 안전 문화를 정의하는 것은 어렵지만 다른 곳에 비해 사고와 손실이 더 많이 발생하는 산업군과 조직이 존재한다. 사고율이 상대적으로 높은 조직의 안전 문화는 다음 중 하나 이상의 특성을 갖는 경향이 있다.

- 리스크 수용 문화(Culture of Risk Acceptance): 이 문화는, 사고는 불가피하므로 사고 방지를 위해서는 모든 사람들이 조심하도록 적극 권유하는 것 이상으로는 할 수 있는 일이 없다는 가정을 기반으로 한다. 사고는 생산성에 따르는 대가로 간주된다. 종종 이 가정은 모든 사람이 안전에—자신의 안전 및 타인의 안전—책임을 져야 하며, 사고는 개인의 책임있는 행동이 부족하여 발생한다는 믿음을 동반한다. 이 문화에서는 모든 사람이 책임감 있고 안전하게 행동하기만 한다면 사고는 줄어들 것이라는 믿음이 일반적이다.
- 부정의 문화(Culture of Denial): 부정의 문화에서는 리스크 평가(assessment)가 비현실적으로 낮은 경우가 많으며, 신뢰할만한 리스크와 경고가 적절한 조사 없이 무시된다: 관리자는 좋은 소식만을 듣기 원하고 좋은 소식만을 말한다. 시스템이 안전하지 않을 수 있는 경우를 식별하는 것이 아니라 시스템이 수용 가능한 수준으로 안전하다는 것을 보여주는 데 중점을 둔다.
- 규정 준수 문화(Culture of Compliance): 여기서는 정부 규제를 준수하는 데 중점을 둔다. 규제를 따르는 것이 수용 가능한(acceptable) 결과를 가져온다는 내재된 문화적 신념을 가진다. 규제 기관은 제품이나 서비스의 안전에 대한 자격 부여에 집중하는 경향이 있기 때문에 사실에 대한 보증이 강조된 후 실제 제품이나 프로세스 안전에 거의 또는 전혀 영향을 미치지 않는 "세이프티 케이스(safety case)" 논의만 방대하게 제기되는 경우가 많다.
- 문서작업 문화(Paperwork Culture): 이 문화는 많은 문서의 작성과 문서의 분석이 안전한 제품과 서비스로 이어질 것이라는 믿음에 기초한다. 많은 문서의 분석이 이루어지지만 대부분이 설계 및 운영에 거의 영향을 미치지 않는다. 안전-관련 문서작업의 대부분은, 제품을 설계하고 운영하거나 프로세스를 시행하거나 서비스를 제공하는 사람들과는 독립적이거나 상호작용이 적은 그룹에 의해 이루어지기도 한다.
- "허세" 문화(Culture of "Swagger"): 안전은 약한 사람들을 위한 것이다. 강한 사람들은 리스크를 통해 발전한다.

좋은 안전 문화의 특징을 구체적으로 설명하는 것은 어렵지만 좋은 안전 문화는 이런 것을 포함한다: 안전과 안전 목표에 대한 개방성, 나쁜 소식을 듣고자 하는 의지, 정부 규제 준수나 방대한 문서

safety and will take appropriate action, managers believe employees are worth listening to and are worthy of respect, and employees feel safe about reporting their concerns and feel their voice is valued. Safety is a shared responsibility where employees are part of the solution and not just considered the problem. At the same time, responsibility is not simply placed on the workforce to keep themselves and others safe.

How is the organizational safety culture established or changed? The safety culture (the values to be used in decision making) in any organization is set by the top management. A sincere commitment by management to safety is often cited as the most important factor in achieving it. Employees need to feel that they will be supported if they exhibit a reasonable concern for safety in their work and if they put safety ahead of other goals such as schedule and cost.

A manager's open concern for safety in everyday dealings with personnel can have a major impact on the reception given to safety issues. A classic story is about Paul O'Neill when he was hired from the outside as the CEO of Alcoa in 1989. He immediately announced that his primary goal was to make Alcoa the safest company in America and to go for zero injuries. Typical reactions to this announced goal were that the Alcoa board of directors had put a "crazy hippie in charge" and that he was going to destroy the company. In fact, within a year, Alcoa's profits had hit a record high and continued that way until O'Neill retired in the year 2000. At the same time, all that growth occurred while Alcoa became one of the safest companies in the world. O'Neill clearly understood that safety and productivity are not conflicting and, in fact, support each other.

In designing or engineering any system, goal setting and establishing the requirements for achieving those goals are the first step in any successful effort: If you do not know where you are going, you are unlikely to get there. Goals are necessary to guide any successful design process. Management establishes the value system under which decisions are made in an organization. Therefore, the first step in designing or improving the safety culture is for management to establish and communicate what is expected in the way of safety-related decision making and behavior.

Once the desired values have been determined by top management, the next step is to communicate the basic values of the leaders through a short written safety philosophy and more detailed safety policy to implement the general philosophy. Top management has responsibility to ensure that the written philosophy and policy has wide buy-in from the managers and employees.

Improving the safety culture, and thus safety, starts by the leaders communicating the type of safety culture they want to establish. Most companies have extensive safety policy documents that detail exactly how safety should be handled. While these are important, a shorter statement of the philosophical principles and values is a good way to communicate the expected cultural principles of the organization. This philosophical statement should define the relationship of safety to other organizational goals.

Some general cultural principles are applicable to all organizations, while the applicability of others will depend on whether the organization is developing safety-critical systems, operating them, or providing safety-related services. Some general principles that might be part of the Safety Philosophy statement:

1. All injuries and accidents are preventable.
2. Increasing quality and safety leads to decreasing cost and schedule and, in the long term, to increased profits. Preventing accidents is good business.
3. Safety and productivity go hand in hand. Improving safety management leads to improving other quality and performance factors. Maximum business performance requires safety.

작업보다는 안전성을 높이는 데 필요한 일에 중점을 두는 것, 안전이 조직의 목표를 달성하는 데 있어 중요하다고 믿는 것. 효과적인 안전 문화에서 직원들은 관리자가 안전에 대한 우려사항을 들어줄 것이고 적절한 조치가 취해질 것이라고 믿으며 관리자는 직원들의 말을 경청하고 존중할 가치가 충분하다고 믿는다. 또한 직원들은 그들의 우려사항을 보고하는 것에 대해 안전하다고 느끼고 자신의 목소리가 중요하다고 생각한다. 안전은 공동의 책임이며, 여기서 직원은 단순히 문제로 인식되는 것이 아니라 솔루션의 일부로 고려된다. 동시에, 자신과 다른 사람들의 안전을 지키기 위해 단순히 직원에게 책임을 물어서는 안 된다.

조직의 안전 문화는 어떻게 확립되고 변화될까? 어떤 조직에서든 안전 문화(의사 결정에 사용되는 가치)는 최고 경영진이 설정한다. 안전에 대한 경영진의 진지한 약속은 종종 안전 달성에 있어 가장 중요한 요인으로 인용된다. 안전에 대한 합리적인 우려를 표하고 일정이나 비용 목표보다 안전 목표를 더 우선시하는 것이 지지받을 것이라는 점을 직원들이 느낄 필요가 있다.

직원들과 일상적으로 다루는 안전에 대한 관리자의 열린 관심은 안전 이슈를 받아들이는 데 큰 영향을 줄 수 있다. 폴 오닐(Paul O'Neill)이 1989년 앨코아(Alcoa) 社의 CEO로 외부에서 고용되었을 때 이야기가 있다. 그는 취임 즉시, 그의 최고 목표는 앨코아를 미국에서 가장 안전한 회사로 만들고 부상률 제로로 만드는 것이라고 발표했다. 이 발표에 대한 전형적인 반응은 앨코아 이사회가 "정신나간 히피"를 책임자로 선임해서 회사를 망칠 것이라는 것이었다. 그러나 실제로는, 1년 만에 앨코아 수익은 사상 최고치를 기록했으며 이는 오닐이 퇴직한 2000년까지 계속되었다. 동시에 앨코아는 세계에서 가장 안전한 회사가 되었고 모든 것이 성장하였다. 오닐은 안전과 생산성은 서로 상충되지 않으며 실제로는 서로를 지원한다는 것을 분명하게 이해하고 있었다.

어떤 시스템을 설계하거나 엔지니어링 할 때 목표 설정과 그 목표를 달성하기 위한 요구사항을 수립하는 것은 성공적인 결과를 위한 첫걸음이다: 만약 당신이 어디로 가고 있는지 모른다면 당신은 그곳에 도착할 수 없다. 목표는 성공적인 설계 프로세스로 가기 위해 반드시 필요하다. 경영진은 조직에서 의사 결정을 내리는 가치 체계(value system)를 수립하므로 경영진이 안전-관련 의사 결정 및 행동 방식에 예상되는 것들을 수립하고 전달하는 것이 안전 문화를 설계하거나 개선하는 첫 번째 단계이다.

최고 경영진이 원하는 가치를 결정한 이후의 다음 단계는 짧게 작성한 안전 철학과 일반적인 철학을 실현할 상세한 안전 정책을 통해 리더의 기본 가치를 전달하는 것이다. 최고 경영진은 글로 작성된 철학과 정책을 관리자와 직원들로부터 폭 넓게 승인받을 책임이 있다.

안전 문화를 개선하는 것은, 즉, 안전은 리더가 그들이 구축하고자 하는 안전 문화의 유형을 전달하는 것에서 시작한다. 대부분의 회사는 안전을 어떻게 처리해야 하는지를 자세히 기술한 광범위한 안전 정책 문서를 가지고 있다. 이러한 것들도 중요하긴 하지만 철학적 원칙과 가치에 대한 더 짧은 성명서(statement)가 조직이 원하는 문화적 원칙을 전달하기에 더 좋은 방법이다. 이 철학적 성명서는 안전과 다른 조직 목표와의 관계를 정의해야 한다.

일부 일반적인 문화적 원칙은 모든 조직에 적용할 수 있지만 다른 문화적 원칙의 적용가능성은 그 조직이 안전-필수 시스템을 개발하는지, 그것을 운영하는지 또는 안전-관련 서비스를 제공하는지에 달라질 것이다. 안전 철학 성명서에 포함될 수 있는 몇 가지 일반적인 원칙은 다음과 같다.

1. 모든 상해 및 사고는 예방할 수 있다.
2. 품질과 안전성을 높이면 비용 및 일정이 줄어들고 장기적으로 이익이 증가한다. 사고를 방지하는 것이 일을 잘 하는 것이다.
3. 안전성과 생산성은 함께 가는 것이다. 안전 관리를 개선하면 다른 품질 및 성과 요소가 개선된

4. Safety has to be built into a product or the design of a service. Adding it later will be less effective and more expensive. After-the-fact assurance cannot guarantee a safe design where safety is not already present. It is better to build safety in than try to ensure it after the fact.
5. The goal of accident/incident causality analysis is to determine why the loss (or near loss) occurred so that appropriate changes can be made rather than to find someone or something to blame.
6. Incidents and accidents are an important window into systems that are not operating safely and should trigger comprehensive causal analysis and improvement actions.
7. Safety information must be surfaced without fear. Safety analysis will be conducted without blame.
8. Safety commitment, openness and honesty is valued and rewarded in the organization
9. Effective communication and the sharing of information is essential to preventing losses.
10. Each employee will be evaluated on his or her performance and contribution to our safety efforts.

Number 4 requires some extra explanation as it violates how many organizations and even industries treat safety today. It is common to find an emphasis on measuring or assessing safety or risk after the fact rather than building safety into products and workplaces from the beginning. Designs are either safe or they are not, and all the arguing and assessing one can do will not change that fact. In addition, assessing or verifying after the fact is not only more expensive than designing safety in originally, it is also not very effective because the possible improvements are usually very limited after the design has been completed.

After-the-fact assessment can only provide confidence that safety already exists or evidence that it does not. Emphasis on after-the-fact assessment, therefore, leads to fudging numbers, often unconsciously, such as emphasizing only positive factors or those factors that will produce positive numbers (called *confirmation bias*), considering only factors that can be measured and can provide a number while ignoring other factors (such as management and design flaws), making up numbers or manipulating them so that numerical goals are achieved, etc. That does not mean that measurement and assessment are not important, just that at best it can only assure that an effort to create a safe system has been successful or not but it cannot make the system safe if it is not safe already.

An alternative to after-the-fact assessment is to go into the assessment process with a mindset to provide evidence that the design is not safe, rather than providing an argument that it is safe. This approach is more likely to be more effective. However, by the time this assessment occurs, it is usually too expensive to fix any problems that might be found or the fixes will be more costly and less effective than building safety into the design from the beginning. Starting the safety analysis process at the beginning of and during the design process will provide the best results (see the chapter in this handbook on integrating STPA into the system engineering process). That is, instead of proving safety, the emphasis should be on identifying and eliminating or controlling hazards. Those who emphasize after-the-fact assessment of safety are unlikely to achieve high safety.

In addition, an emphasis on probabilities or likelihood in assessment can lead to overconfidence and unrealistic assessment and to the omission of important factors that are not stochastic or for which probabilities cannot be obtained. In the very successful SUBSAFE program, all evidence used in certifying a system for safety must be Objective Quality Evidence (OQE). OQE is defined as “any statement of fact, either quantitative or qualitative, pertaining to the quality of a product or service based on observations, measurements, or tests that can be verified.” Probabilistic risk assessment (or even qualitative risk assessment that includes identifying the likelihood of a hazard leading to an accident) is an attempt to predict the future and cannot be verified beyond waiting many years to see if the

다. 최대 사업 성과에는 안전이 요구된다.

4. 안전은 제품에 내장되거나 서비스 설계에 내장되어야 한다. 안전을 나중에 추가하면 효과가 덜하고 비용이 많이 든다. 안전이 이미 존재하지 않는 상황에서 사후보증(After-the-fact assurance)으로 안전한 설계를 보증할 수는 없다. 사후에 보장하기 위해 노력하는 것보다는 안전을 내부에 구축하는 것이 낫다.
5. 사고/사건 원인 분석(causality analysis)의 목표는 비난할 누군가나 무언가를 찾는 것이 아니라 손실이 발생한(또는 발생할 뻔한) 이유를 밝혀 적절한 변경이 이루어질 수 있도록 하는 것이다.
6. 사건과 사고는 안전하지 않게 운영되는 시스템을 들여다볼 수 있는 중요한 기회이며 종합적인 원인 분석 및 개선 활동을 하도록 만든다.
7. 안전 정보는 두려움 없이 표출되어야 한다. 안전 분석은 비난 없이 수행될 것이다.
8. 안전 약속, 개방성, 그리고 정직이 조직에서 가치 있게 평가되고 보상을 받는다.
9. 효과적인 의사소통과 정보 공유는 손실을 예방하는 데 필수적이다.
10. 각 직원은 자신의 성과와 안전 노력에 대한 기여도를 평가받게 된다.

오늘날 많은 조직, 그리고 산업계조차도 취급해야 하는 안전을 위반하고 있으므로 위 4번에 대해서는 추가 설명이 필요하다. 처음부터 제품과 작업장에 안전을 구축하지 않고 사후에 안전이나 리스크를 측정하거나 평가하는 것에 중점을 두는 것이 일반적이다. 설계는 안전하거나 안전하지 않을 수 있으며 모든 논쟁과 평가가 그 사실을 바꿀 수는 없다. 또한 사후 평가(assessing)나 검증(verifying)은 처음에 안전을 설계하는 것보다 비용이 많이 들뿐만 아니라 설계가 완료된 후 개선할 수 있는 것은 매우 제한적이기 때문에 그다지 효과적이지 않다.

사후 평가는 안전이 이미 존재한다는 확신이나 존재하지 않는다는 증거만 제공할 수 있다. 따라서 사후 평가에 중점을 두면 종종 무의식적으로 과장된 수치를 유발한다. 예를 들어 긍정적 요인 또는 긍정적 수치를 만드는 요인(확증 편향이라 함)만을 강조하거나, 다른 요인(관리나 설계 결함과 같은)을 무시하면서 측정할 수 있고 수치를 제공할 수 있는 요인만을 고려하는 것, 수치적 목표를 달성할 수 있도록 수치를 꾸미거나 조작하는 것 등이 있다. 이 말은 측정과 평가가 중요하지 않다는 것이 아니며 단지 이런 방식은 안전한 시스템을 만드는 노력이 성공하였는지 여부만을 확신할 수 있을 뿐 이미 안전하지 않은 시스템을 안전하게 만들 수는 없다는 것이다.

사후 평가의 대안은 설계가 안전하다는 논거를 제공하는 것이 아니라 설계가 안전하지 않다는 증거를 제시한다는 사고방식을 가지고 평가 프로세스를 시작하는 것이다. 이 접근법이 더 효과적일 수 있다. 그러나 이러한 평가가 이루어질 때까지는 발견된 모든 문제를 수정하기 위한 비용이 너무 많이 들며 이러한 수정은 처음부터 설계에 안전을 구축하는 것보다 비싸고 비효율적일 것이다. 설계 프로세스의 시작 및 수행 과정에서 안전 분석 프로세스를 시작하면 최고의 결과를 얻을 수 있을 것이다(이 핸드북의 'STPA를 시스템 엔지니어링 프로세스에 통합' 장 참조). 즉, 안전을 입증하는 대신, 위험을 식별하고 제거하거나 통제하는 것에 중점을 두어야 한다. 안전에 대한 사후 평가를 강조하는 사람들은 높은 수준의 안전을 달성하지 못할 것이다.

또한, 평가에서 발생확률이나 발생 가능성에 중점을 두면 과신(overconfidence)이나 비현실적인 평가를 할 수 있고 확률적이지 않거나 확률을 얻을 수 없는 중요한 요인을 누락할 수 있다. 매우 성공적인 프로그램인 SUBSAFE에서 시스템 안전 인증에 사용되는 모든 증거는 객관적인 품질 증거(OQE, Objective Quality Evidence)이어야 한다. OQE는 "관찰, 측정 또는 검증 가능한 테스트를 기반으로 하는 제품이나 서비스의 품질에 대한 정량적 또는 정성적인 모든 사실의 진술"로 정의된다. 확률적 리스크 평가(또는 사고로 이어지는 위험의 발생 가능성 식별을 포함하는 정성적 리스크 평가)는 미래를 예측하

predictions were valid. Few effective crystal balls exist. Assessing likelihood is only effective when the past is identical to the future so past experience can be applied or it is close enough so that extrapolation from the past can be done. But new products are created to improve on past products in some way, not to duplicate them. And rare is the system (and its environment) that does not change over time. Even if the product itself does not change or degrade, people operating or using the system start to change their behavior, their uses of the system evolve, and the environment changes.

A written statement of the safety philosophy and more detailed policy statements are a start, but they are not enough. Employees quickly identify when the written policy differs from the actual behavior of managers. To be successful, there needs to be real commitment by those at the top, not just sloganeering and going through the motions.

How is commitment demonstrated? It is shown by setting priorities and following through on them; by personal involvement (e.g., top management chairing groups where safety decisions are made); by setting up appropriate organizational structures; by appointing designated, high-ranking leaders to safety-related responsibilities and providing adequate resources for them to be effective; by assigning the best employees to safety-related activities and rewarding them for their efforts; and by responding to initiatives by others. It is also communicated by minimizing blame. Leaders need to demonstrate that their highest priority is to fix the systemic factors leading to losses and not just find someone on which to place blame (usually someone at the lowest level in the organization as possible) and then moving on. Finally, leaders need to engineer the incentive structure in the organization to encourage the behavior desired.

The major ideas in this section are summarized in the following box:

기 위한 시도이며 예측이 유효한지는 수년을 기다리지 않고는 확인할 수 없다. 효과적인 미래 예측방법(crystal balls)은 거의 없다. 발생 가능성을 평가하는 것은 과거가 미래와 동일하여 과거의 경험을 적용할 수 있거나 매우 가까운 미래라서 과거로부터 추정할 수 있을 때에만 유효하다. 그러나 새로운 제품은 과거 제품을 복제하지 않고 어떤 식으로든 개선하기 위해 만들어지며 시간이 지남에 따라 변하지 않는 시스템(그리고 환경)은 드물다. 제품 자체가 변경되거나 성능이 저하되지 않더라도 시스템을 운영하거나 사용하는 사람들의 행동이 변하기 시작하고 시스템 사용이 진화하며 환경이 바뀐다.

안전 철학을 기록한 성명서와 상세한 정책 문서가 시작이지만 이것으로는 충분치 않다. 직원들은 관리자의 실제 행동이 서면(written) 정책과 다른 경우 이를 빨리 파악한다. 성공하기 위해서는 상부에 있는 사람들이 단순히 구호만 외치거나 시늉만 하는 것이 아니라 그들의 진정한 약속이 필요하다.

약속은 어떻게 증명할 수 있을까? 그것은 우선순위를 설정하고 그것을 따르는 것; 개인적 참여(예: 안전 결정을 내리는 최고 경영 의장단); 적절한 조직 구조 설정; 안전-관련 책임을 지정할 고위 간부 리더를 임명하고 효과적이도록 적절한 자원을 제공; 우수 직원을 안전-관련 활동에 배정하고 그들의 노력에 대해 보상; 다른 사람들의 첫 발의(initiatives)에 응답하는 것 등을 통해 보여줄 수 있다. 또한 약속은 비난을 최소화함으로써 증명된다. 리더들은 그들의 가장 높은 우선순위가 비난할 사람(일반적으로 조직의 가장 하위 수준에 있는 누군가)을 찾는 것이 아니라 손실로 이어질 수 있는 시스템적 요인을 수정하며 계속 나아가야 하는 것임을 보여줘야 한다. 마지막으로, 리더들은 바람직한 행동을 장려하기 위해 조직의 인센티브 구조를 설계해야 한다.

이 섹션의 주요 아이디어는 아래 상자에 요약되어 있다.

Tips for how management can improve safety culture

- Set the goals and values to be used in decision making; establish and communicate what is expected in safety-related decision making and behavior.
- Support employees who exhibit reasonable concern for safety in their work
- Create a short written safety philosophy and more detailed safety policy
- Ensure safety philosophy and policy have wide buy-in from managers and employees
- Follow the safety philosophy in your decision making and expect the same from everyone
- Emphasize building safety in and not assurance or after-the-fact assessment
- Perform assessment with the goal of providing evidence that the design is not safe, not providing an argument that it is safe
- Require objective quality evidence for certification or assurance
- Demonstrate commitment to safety by
  - Personal involvement
  - Setting priorities and following through on them
  - Setting up appropriate organizational structures
  - Appointing high-ranking, respected leaders to safety-related roles and responsibilities
  - Providing adequate resources for safety-efforts to be effective
  - Assigning the best employees to safety-related activities, not just those who are nonessential or expendable
  - Rewarding employees for safety efforts
  - Responding to initiatives by others
- Minimize blame; focus on “why” not “who”
- Engineer the incentive structure to encourage desirable safety-related behavior

Safety Control Structure

Designing the safety control structure is just another form of system engineering, as discussed in the previous two chapters. It starts with a statement of the purpose of the effort. The SMS is:

What: A control structure that assists in creating and maintaining safety in an organization

Why: To ensure that hazards are eliminated or, if not possible, controlled (mitigated) and to promote an effective safety culture

How: By establishing management controls and responsibilities (RAAs) to manage hazards and a comprehensive and usable safety information system.

The overall goal is to design a control structure that eliminates or reduces losses. Satisfying this goal requires a clear definition of expectations, responsibilities, authority, and accountability for safety tasks at all levels of the control structure. In addition, to operate effectively the structure requires appropriate feedback and coordination between entities. It should also involve leading indicators to signal when the controls are becoming ineffective because of internal or external changes. Together the entire control structure must enforce the safety constraints on system behavior through physical design, defined processes and procedures, and social interactions and culture.

There will probably be significant differences between a safety control structure for development and one for operations. If you are a regulated industry, government regulatory agencies will need to be

경영진이 안전 문화를 향상시키기 위한 팁

- 의사 결정에 사용되는 목표와 가치를 설정한다; 안전-관련 의사 결정 및 행동에서 예상되는 것을 정하여 전달한다.
- 업무상 안전에 대한 합리적인 우려를 제기하는 직원을 지원한다.
- 짧게 문서화된 안전 철학을 만들고 보다 상세한 안전 정책을 수립한다.
- 안전 철학 및 정책에 대한 관리자와 직원의 폭넓은 승인을 확보한다.
- 의사 결정 시 안전 철학을 따르고 모든 사람에게 같은 것을 기대한다.
- 보증이나 사후 평가가 아닌 안전의 구축을 강조한다.
- 설계가 안전하다는 논거보다는 안전하지 않다는 증거를 제공하는 것을 목표로 평가를 수행한다.
- 인증 또는 보증을 위한 객관적인 품질 증거를 요구한다.
- 다음을 통해 안전에 대한 약속을 보여준다.
  - 개인적 참여
  - 우선순위를 설정하고 이를 따름
  - 적절한 조직 구조 설정
  - 안전-관련 역할 및 책임을 부여할 고위의 존경받는 리더 임명
  - 안전 노력이 효과적이도록 적절한 자원 제공
  - 불필요하거나 소모적인 사람들이 아닌, 최고의 직원을 안전-관련 활동에 배치
  - 직원의 안전 노력에 대해 보상
  - 다른 사람들의 첫 발의(initiatives)에 응답
- 비난을 최소화함; "누가"가 아닌 "왜"에 집중
- 바람직한 안전-관련 행동을 장려하기 위한 인센티브 구조 설계

안전 컨트롤 스트럭처(safety control structure)

앞의 두 장에서 설명한 것처럼 안전 컨트롤 스트럭처를 설계하는 것은 시스템 엔지니어링의 또 다른 형태이다. 그것은 업무의 목적에 대한 설명으로 시작한다. 안전 관리 시스템(SMS)은 다음과 같다.

무엇을(what): 조직의 안전을 만들고 유지하는 데 도움이 되는 컨트롤 스트럭처

왜(why): 위험을 제거하거나 제거가 불가능할 시 통제(완화)하는 것을 보장하고, 효과적인 안전 문화를 고취하기 위해

어떻게(how): 위험 관리를 위한 관리 제어와 책임(RAAs, Responsibility, Accountability, Authority) 및 포괄적이고 유용한 안전 정보 시스템을 수립함으로써

전체적인 목표는 손실을 제거하거나 줄이기 위한 컨트롤 스트럭처를 설계하는 것이다. 이 목표를 달성하기 위해서는 컨트롤 스트럭처의 모든 수준에서 안전 작업에 대한 기대, 책임, 통제권한 및 의무를 명확히 정의해야 한다. 또한 효과적인 스트럭처의 운영을 위해서는 적절한 피드백과 엔터티(entities) 간의 조화가 필요하다. 그리고 내·외부의 변화로 인해 컨트롤이 비효율적으로 변하는 것을 알리기 위한 선행지표가 있어야 한다. 동시에 전체 컨트롤 스트럭처는 물리적 설계, 정의된 프로세스 및 절차, 사회적 상호작용과 문화를 통해 시스템 동작에 대한 안전 제약사항을 강제해야 한다.

개발을 위한 안전 컨트롤 스트럭처와 운영을 위한 안전 컨트롤 스트럭처 사이에는 아마 큰 차이가

included in your safety control structure model. Other external groups may be important to include such as the courts, insurance companies, user groups, etc.

General Safety Control Structure Design Considerations

Some basic considerations when designing of evaluating a safety control structure is how to assign responsibility for safety, the appropriate place(s) in the organization for safety-related activities, communication of information and coordination of activities, managing and controlling for change, designing and encouraging feedback, and determining the design and role of the risk management procedures to be used. In addition, there must be consideration of education and training efforts that will be required and how learning and continual improvement of the control structure itself will be assured. In addition, see Appendix D "Guidelines for Designing and Evaluating an SMS."

Assigning Responsibility:

Leadership is the key to achieving high levels of safety. That means that leadership of the organizational safety functions should not just be a rotational assignment for training future leaders and managers. Organizations that have few losses appoint leaders of the safety functions who are passionate about safety and the role they play in preventing losses. There should be a career path within the organization that allows those who are committed to preventing losses to rise in the safety management organization.

As with any effective management system, there must be responsibility, authority, and accountability assigned. A belief that "everyone is responsible for their own and others safety as well as the safety of the products" leads to excessive accidents. If everyone is responsible for safety, then nobody is.

Responsibility for safety lies at every level of the organizational structure although appropriate responsibilities will differ at all of them. In some, usually high-accident organizations, a belief is prevalent that responsibility for safety should be pushed down in the control structure. The argument is used that the lower levels have more information about how their parts of the system actually work. The problem with this argument is that the lower levels also lack perspective: Although they have detailed information about their specific parts of the system, they do not have information about other parts of the system and therefore cannot anticipate or prevent unsafe interactions among components of the entire system. Lower levels also tend to have a short-term focus rather than a longer-term one while the higher organizational levels have a broader focus on system goals as opposed to the goals of their own subcomponent of the system. The higher organizational levels need to control interactions among the lower level components and ensure that safety constraints are being enforced by those beneath them.

Each level must provide oversight of the level below by ensuring they are using appropriate procedures, carrying them out correctly, and that they are effective. There also usually needs to be a management focal point (or points) with responsibility for ensuring that the safety management system is designed and working properly at each level of the management system.

있을 것이다. 만약 당신이 규제 산업에 속해 있다면 정부 규제 기관을 안전 컨트롤 스트럭처 모델에 포함시켜야 한다. 법원, 보험 회사, 사용자 그룹 등과 같은 다른 외부 그룹을 포함하는 것이 중요할 수도 있다.

일반적인 안전 컨트롤 스트럭처 설계 고려사항

안전 컨트롤 스트럭처의 평가를 설계할 때 몇 가지 기본적인 고려사항은 안전 책임을 할당하는 방법, 안전-관련 활동을 위한 조직의 적절한 배치, 정보의 커뮤니케이션 및 활동의 조화, 변경의 관리 및 제어, 피드백의 설계 및 장려, 리스크 관리 절차의 설계 및 역할의 결정 등이다. 추가적으로, 필요한 교육과 훈련이 반드시 고려되어야 하며 컨트롤 스트럭처 자체에 대한 학습과 지속적인 개선이 어떻게 보장되어야 하는지도 고려되어야 한다. 추가적인 내용은 부록 D의 "안전 컨트롤 스트럭처에 포함되어야 하는 책임"을 참조한다.

책임 할당(Assigning Responsibility):

리더십은 높은 수준의 안전을 달성하기 위한 열쇠이다. 이는 조직의 안전 기능에 대한 리더십이, 미래의 리더와 관리자를 양성하기 위해 돌아가며 할당되어서는 안 된다는 것을 의미한다. 손실이 거의 없는 조직에서는 손실을 방지하는 역할과 안전에 대해 열정적인 사람을 안전 기능의 리더로 임명한다. 회사 내에는 안전 관리 조직에서 손실 방지를 위해 헌신한 사람들이 성장할 수 있는 진로(career path)가 있어야 한다.

모든 효과적인 관리 시스템과 마찬가지로 책임(responsibility), 통제권한(authority), 그리고 의무(accountability)가 할당되어야 한다. "모든 사람은 제품의 안전뿐만 아니라 그들 자신과 다른 사람의 안전에 대한 책임이 있다"는 믿음은 과도한 사고를 초래한다. 모두에게 안전에 대한 책임이 있다면 아무도 그 책임을 지지 않는다.

적절한 책임은 조직 구조의 수준에 따라 모두 다르겠지만 모든 조직 구조 수준에서 안전에 대한 책임은 있다. 보통, 사고가 많은 조직에서는 컨트롤 스트럭처에 안전에 대한 책임을 하위 수준으로까지 밀어넣어야 한다는 생각이 널리 퍼져 있다. 이는 시스템의 각 부분이 실제로 어떻게 작동하는지에 대한 정보를 하위 수준에서 더 많이 가지고 있다고 여기기 때문이다. 이러한 생각의 문제는 하위 수준에서도 역시 전체적인 시각(perspective)이 부족하다는 것이다: 비록 하위 수준은 시스템의 특정 부분에 대한 자세한 정보가 있더라도 시스템의 다른 부분에 대한 정보가 없으므로 전체 시스템 컴포넌트 간의 안전하지 않은 상호작용을 예상하거나 예방할 수 없다. 조직의 상위 수준은 시스템의 하위 컴포넌트 목표가 아닌 시스템 목표에 더 많은 초점을 맞추는 반면, 하위 수준에서는 장기적인 관점보다는 단기적인 관점을 갖는 경향이 있다. 조직의 상위 수준에서는 하위 수준 컴포넌트 간의 상호작용을 제어해야 하고 그 바로 아래의 컴포넌트에 의해 안전 제약사항이 이행됨을 보장해야 한다.

조직 구조의 각 수준(level)은 적절한 절차를 사용하고, 올바르게 이행하며, 효과적임을 보장함으로써 각각의 아래에 있는 수준을 감독해야 한다. 또한 일반적으로 안전 관리 시스템이 각각의 수준에서 적절히 설계되고 제대로 작동되도록 임무에 책임지고 집중해야 한다.

Tips for assigning responsibility

- Appoint leaders who are passionate about safety
- Create career paths for those committed to preventing losses
- Provide a clear definition of expectations, responsibilities, authority and accountability at all levels of the safety control structure
- Do not make everyone responsible for safety
- Assign safety responsibilities throughout and at all levels of the control structure
- Assign someone to be responsible for ensuring the SMS is designed and working properly

Place in the organization:

There are some basic principles that need to be considered in deciding where to place system safety activities in the organization, and how to design and manage feedback and control change.

First, there needs to be a high-level group with enterprise-level responsibilities such as ensuring that required activities are taking place and that they are effective. This group also provides leadership and coordination. Although safety activities will permeate every part of the development and operation of large organizations, a common methodology and approach will strengthen the individual activities. The leader of this group needs to have the ability to communicate directly with top management and provide input to all types of management decision making. This implies that the person in this position must report to someone with influence and be seen as having the support of senior management.

Below this top-level safety management, there may be activities at all levels of the organization, with the higher levels having broader responsibilities and each successive lower level having more focused responsibilities appropriate to the level at which they operate, for example, at the business unit, program and project level of a development organization. A common mistake is to make safety a staff function that has little impact on line operations.

There is no one or even several correct designs for a safety control structure. An effective design will depend on the industry, organizational management style, etc. Some responsibilities that should be included in the safety control structure for development and operations are listed in an appendix of this handbook.

Second, as has been mentioned before in this handbook, separating system safety engineering from system engineering is a mistake in product development organizations; it can lead to higher costs in achieving safety goals because safety concerns must be addressed starting early in the development process. Safety engineering efforts should not be focused on after-the-fact assurance. Service organizations have different concerns, but there should be some safety engineering component within all departments or groups where decision making about safety takes place. At the same time, there are reasons for also having independent oversight and decision-making authority outside the primary decision-making group. For all types of organizations, safety needs to be designed into the workplace and decisions need to be made about the appropriate place for this responsibility in the organization.

Some general principles for allocating safety responsibilities in the organizational structure are:

- Decision makers need direct links to those who can provide safety information. If critical information has to float up a chain of command, it may be lost or modified either deliberately, usually because of schedule or budget pressures, or inadvertently. Direct communication channels provide more chance that information is received in a timely manner and without

책임 할당을 위한 팁

- 안전에 열정적인 리더를 임명한다.
- 손실 예방에 헌신한 사람들을 위한 진로(career path)를 만든다.
- 안전 컨트롤 스트럭처의 모든 수준에서 기대, 책임, 통제권한 및 의무를 명확히 정의한다.
- 모든 사람이 안전에 대한 책임을 갖게 하지 않는다.
- 컨트롤 스트럭처의 모든 수준에서 안전 책임을 할당한다.
- SMS가 적절히 설계되고 운영되는 것을 보장하기 위해 누군가에게 책임을 부여한다.

조직의 배치(Place in the organization):

조직에서 시스템 안전 활동을 어디에 배치할지, 피드백과 컨트롤 변경에 대한 설계 및 관리를 어떻게 수행할지를 결정할 때 고려해야 할 몇 가지 기본 원칙이 있다.

첫째, 필요한 활동이 추진되고 그 효과를 보장하는 것과 같은 기업 차원의 책임 있는 상위 수준의 그룹이 필요하다. 이 그룹은 리더십과 조정(coordination)도 제공한다. 안전 활동은 대규모 조직의 개발 및 운영의 모든 부분에서 수행되지만 공통된 방법론과 접근법이 있으면 각각의 활동을 강화할 수 있을 것이다. 이 그룹의 리더는 최고 경영진과 직접 의사소통하고 모든 유형의 경영 의사 결정에 의견을 줄 수 있어야 한다. 이 말은 이 위치에 있는 사람은 반드시 영향력이 있는 누군가에게 보고해야 하고, 상위 경영진의 지원을 받아야 함을 의미한다.

이러한 최상위 안전 관리 아래로, 조직 내 모든 수준에서 안전 활동이 이루어진다. 상위 수준은 더 광범위한 책임을 가지고 각 연속적 하위 수준은 운영 수준(예를 들어, 개발 조직의 사업부, 프로그램 및 프로젝트 수준)에 적합한 책임에 더 집중한다. 흔한 실수는 안전을 운영 라인에 거의 영향을 미치지 않는 지원적인 기능(staff function)으로 만든다는 것이다.

안전 컨트롤 스트럭처의 올바른 설계에 대한 정답은 존재하지 않는다. 효과적인 설계는 산업, 조직의 경영 스타일 등에 따라 달라진다. 개발 및 운영을 위한 안전 컨트롤 스트럭처에 포함되어야 하는 일부 책임은 본 핸드북 부록에 나열되어 있다.

둘째, 이 핸드북 앞에서 언급했듯이 시스템 엔지니어링에서 시스템 안전 엔지니어링(system safety engineering)을 분리하는 것은 제품 개발 조직의 실수이다; 안전에 대한 우려는 개발 프로세스 초기에 처리되어야 하므로 시스템 엔지니어링에서 시스템 안전 엔지니어링을 분리하면 안전 목표를 달성하는 데 더 많은 비용이 든다. 안전 엔지니어링 노력은 사후보증에 집중되어서는 안 된다. 서비스 조직들은 각기 다른 관심사가 있지만, 안전 엔지니어링 컴포넌트는 안전에 대한 의사 결정이 이루어지는 모든 부서 또는 그룹 내에 있어야 한다. 그와 동시에, 1차(primary) 의사 결정 그룹과는 별도로 독립적인 감독 및 의사 결정 권한을 가진 그룹을 두는 것에는 이유가 있다. 모든 유형의 조직에서 작업 공간에 안전을 설계해야 하며 이러한 책임을 조직 내 어디에서 가지는 것이 적절한지를 결정하는 것이 필요하다.

조직의 구조에서 안전 책임을 할당하기 위한 몇 가지 일반적인 원칙은 다음과 같다.

- 의사 결정권자는 안전 정보를 제공할 수 있는 사람들과 직접적으로 연결되어야 한다. 만약 필수적인 정보가 일련의 명령 체계를 거쳐야 한다면 일정이나 예산 압박으로 인해 의도적으로 또는 의도치 않게 분실되거나 수정될 수 있다. 직접적인 의사소통 채널은 이해관계가 충돌하는 그룹이 정보를

being changed by being filtered through groups with potentially conflicting interests. Some decision makers may also need fast access to information.

- Direct communication channels to most parts of the organization are required. Safety may be involved in almost all organizational activities.
- Safety must have influence on decision making, which means that decision makers have access to necessary safety information at the time safety-related decisions need to be made.

The SUBSAFE program for safety in U.S. nuclear submarines uses a unique design that satisfies many of these requirements. They describe their structure as a separation of powers or a "three-legged stool" (Figure 6.2). Managers can select only from a set of acceptable options (with respect to safety) derived by the Independent Technical Authority (ITA). Technical Authority is defined as a process that establishes and assures adherence to technical standards and policy. The ITA provides a range of technically acceptable alternatives with risk[33] and value assessments. Responsibilities (and accountability) include:

- Setting and enforcing technical standards
- Maintaining subject matter expertise
- Assuring safe and reliable operations
- Ensuring effective and efficient systems engineering
- Making unbiased independent technical decisions
- Providing stewardship of technical and engineering capabilities.

The third leg of the stool is the compliance verification organization. It is equal in authority to the program managers and the ITA.

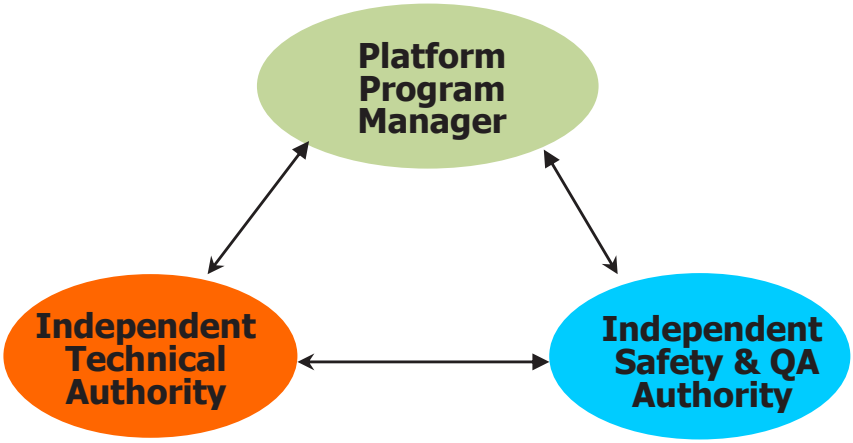


Figure 6.2: The SUBSAFE 3-Legged Stool Concept

[33] Risk may be specified quantitatively or qualitatively but, as explained elsewhere in this chapter, in SUBSAFE it must use objective quality evidence and therefore probabilistic risk assessment or likelihood estimates that cannot be tested or verified are not allowed.

필터링하여 정보가 변경되는 것을 막고 정보가 적시에 전달될 수 있는 더 많은 기회를 제공한다. 의사 결정권자가 정보를 빨리 접하고 싶어할 수도 있다.

- 조직의 대부분 부서에서 직접적인 의사소통 채널이 필요하다. 안전은 거의 모든 조직의 활동과 관련될 수 있다.
- 안전은 의사 결정에 영향을 미쳐야 한다. 이는 의사 결정권자가 안전-관련 결정을 내려야 하는 시점에, 필요한 안전 정보에 접근할 수 있어야 함을 의미한다.

미국 핵 잠수함 안전을 위한 SUBSAFE 프로그램은 이러한 많은 요구사항을 충족시키는 독특한 설계를 사용한다. 그들은 자신들의 구조를 권력(power)의 분리 또는 "세 다리 의자"로 표현한다(그림 7.2). 관리자는 독립기술당국(ITA, Independent Technical Authority)이 제공하는 일련의 수용 가능한 옵션(안전 측면에서)만 선택할 수 있다. 기술당국은 기술 표준과 정책 준수를 확립 및 보증하는 프로세스로 정의된다. ITA는 리스크[33] 및 가치 평가(assessments)를 통해 기술적으로 수용 가능한 대안의 범위를 제공한다. 책임(및 의무)에는 다음이 포함된다.

- 기술 표준의 설정 및 시행
- 주제에 대한 전문지식 유지
- 안전하고 신뢰할 수 있는 운영 보장
- 효과적이고 효율적인 시스템 엔지니어링 보장
- 편견 없는 독립적인 기술적 결정 수립
- 기술 및 엔지니어링 역량에 대한 관리(stewardship)

의자의 세 번째 다리는 준수 여부(compliance)를 검증(verification)하는 조직이다. 이 조직은 프로그램 관리자 및 ITA와 동일한 통제권한을 가진다.

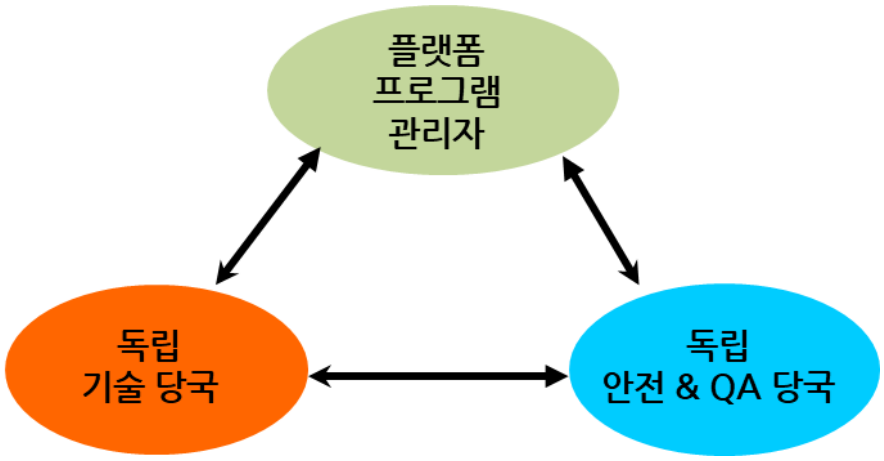


그림 7.2: SUBSAFE의 세 다리 의자 개념

[33] 리스크는 정량적 또는 정성적으로 명시될 수 있지만 이 장에서 설명한 바와 같이 SUBSAFE에서는 객관적인 품질 증명만을 사용해야 했으며 따라서 확률론적 리스크 평가(probabilistic risk assessment)나 테스트 또는 검증할 수 없는 발생 가능성 추정(likelihood estimates)은 허용되지 않는다.

Tips for where to locate activities in the control structure

- Create a high-level group with enterprise level responsibilities, such as ensuring required activities are taking place and they are effective and providing leadership and coordination.
    - This group ensures a common methodology and approach is being used by everyone
    - They must have a direct path to top management and provide input to management decision making
    - They must report to someone with influence and be seen as having the support of senior management
- Assign responsibilities so safety is not just a staff function but has direct impact and influence on line operations
- Do not separate system safety and system engineering in product development organizations. Integrate the safety function into the engineering function.
- Include safety engineering components in all departments or groups where safety-related decision making occurs.
- Provide independent oversight in addition to integrated functions
- Ensure decision makers have direct links to those who can provide safety analysis and information.
- Create direct communication channels between any parts of the organization where there are safety-related activities and safety decisions need to be made.
- Consider implementing the SUBSAFE separation of powers structure. At the least, ensure that technical decisions are independent of programmatic ones.

Communication and Coordination:

Feedback and dissemination of information to those who need it to perform their safety management roles is an important factor to consider in the design of your safety control structure. It is not just a matter of collecting lots (sometimes enormous amounts) of information. In fact, collecting too much information may degrade the information system when it overwhelms the ability to identify the information each person actually needs to make effective safety decisions. In addition, required information for improved safety-related decision making must be handy, that is, available when needed. Because it is so important in reducing losses, the safety information system is discussed separately later in this chapter.

Communication is also important in coordination of activities and responses to events. People with overlapping responsibilities need communication channels and ways to coordinate their activities to ensure that the safety constraints are enforced. For example, safety-motivated changes in one subsystem may affect another subsystem and the system as a whole. Safety activities must not end up fragmented and uncoordinated. Interactions must be defined not just between hierarchical components but also between different parts or types of systems at the same level, such as between development and manufacturing or between development and operations. Figure 6.3 shows an example of the types of safety-related information that needs to be communicated between system developers and system operators.

컨트롤 스트럭처에서 활동(activities)을 어디에 넣을 것인지에 대한 팁

- 필요한 활동이 수행되고 그러한 활동이 효과적임을 보장하며 리더십과 조정을 제공하는 등의 기업 차원의 책임 있는 상위 수준 그룹을 만든다.
  - 이 그룹은 모든 사람들이 공통된 방법론과 접근법을 사용함을 보장한다.
  - 최고 경영진과의 직접적인 경로(path)가 있어야 하며 경영 의사 결정에 의견을 제공해야 한다.
  - 영향력 있는 사람에게 보고해야 하며 상위 경영진의 지원을 받아야 한다.
- 책임을 할당하여 안전이 단순히 지원적인 기능(staff function)이 아니라 라인 운영에 직접적인 영향을 미치도록 한다.
- 제품 개발 조직에서 시스템 안전과 시스템 엔지니어링을 분리하지 않는다. 안전 기능을 엔지니어링 기능에 통합한다.
- 안전-관련 의사 결정이 이루어지는 모든 부서 또는 그룹에 안전 엔지니어링 컴포넌트를 포함시킨다.
- 통합 기능(integrated function) 이외에 독립적인 감독 기능을 제공한다.
- 의사 결정권자가 안전 분석 및 안전 정보를 제공할 수 있는 사람들과 직접 연결되도록 한다.
- 안전-관련 활동이 있고 안전 결정이 필요한, 조직의 모든 부서간에 직접적인 의사소통 채널을 만든다.
- 권력(power) 구조를 SUBSAFE(미국 해군 핵잠수함 안전 프로그램)처럼 분리하는 것을 고려한다. 적어도 기술적인 결정은 프로그램의 결정과는 독립적이어야 한다.

의사소통 및 조정(Communication and Coordination):

안전 관리 역할을 수행하기 위해 정보가 필요한 사람에게 피드백하고 정보를 보급하는 것은 안전 컨트롤 스트럭처를 설계할 때 고려해야 하는 중요한 요소이다. 단지 많은(때로는 엄청난) 정보를 수집하는 것의 문제가 아니다. 실제로, 너무 많은 정보의 수집은 각 개인이 효과적인 안전 결정을 내리는데 필요한 능력을 압도함으로써 정보 시스템을 저하시킬 수 있다. 또한 안전-관련 의사 결정 개선에 필요한 정보의 사용은 편리해야 한다. 즉, 필요할 때 사용할 수 있어야 한다. 이것은 손실을 줄이는 데 매우 중요하기 때문에 안전 정보 시스템에 대해서는 이 장의 뒷부분에서 별도로 설명한다.

의사소통은 활동의 조정과 이벤트 대응에 있어서도 중요하다. 안전 제약사항의 적용을 보장할 수 있도록 책임이 중복되는 사람들에게는 의사소통 채널과 그들의 활동을 조정하는 방법이 필요하다. 예를 들어 한 서브시스템에서 안전을 이유로 변경한 것이 다른 서브시스템 및 시스템 전체에 영향을 줄 수 있다. 안전 활동은 단편화(fragmented)되면 안 되고 조정되지 않으면 안 된다. 상호작용은 계층적인 컴포넌트간뿐만 아니라 동일한 수준에 있는 시스템의 서로 다른 파트나 유형 간에도(예: 개발과 제조간, 또는 개발과 운영 간) 정의되어야 한다. 그림 7.3은 시스템 개발자와 시스템 운영자간에 의사소통되어야 하는 안전-관련 정보 유형의 예를 보여준다.

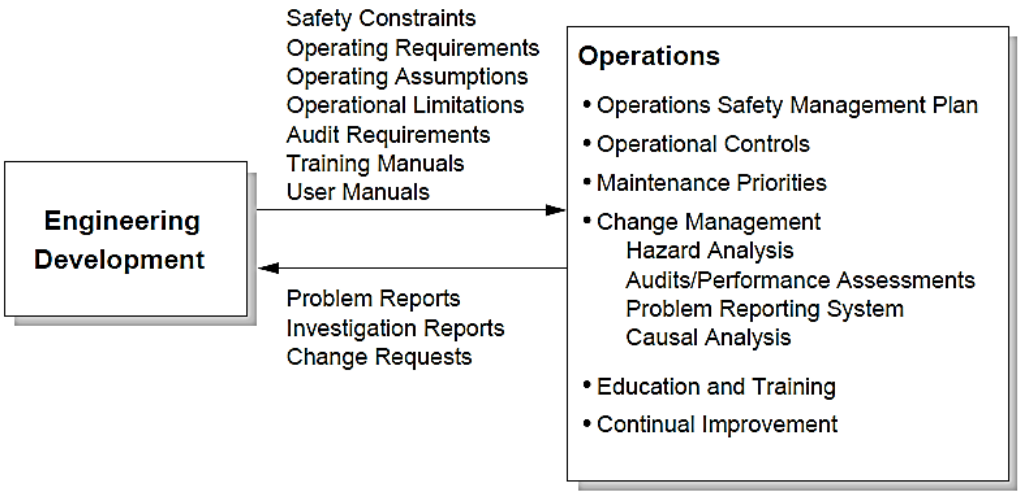


Figure 6.3. Necessary information channels between development and operations to support safety activities in both.

Similar descriptions of the information needing to be communicated among all the components of the safety control structure need to be identified and documented.

One very effective means for communication and coordination devised by the defense department is *working groups*. Defense department projects can span many years, be extremely large and complex, and often involve a large number of participants who are geographically and organizationally distributed. Coordination and communication can become a major problem in such projects. Part of the solution is provided by a hierarchical structure where the high-level management of the project may lie in the Defense Department, but there is a Prime Contractor that provides the system engineering and coordination among the subcontractors. For such a structure to work, communication becomes critical. Working groups have been successful in such coordination and communication efforts and can be adapted for less complex projects.

A working group provides an interface between two hierarchical components of the safety control structure or between two or more components at the same level. Members of these groups are responsible for coordinating efforts, reporting the status of unresolved safety issues, and sharing information about independent safety efforts. There may be working groups at each level of the control structure, with their members and responsibilities depending on the level at which they operate. A corporate working group may be composed of the safety managers for different divisions or programs while at the lower levels working groups may be composed of representatives from groups designing different parts of a product. When there are special types of safety problems that require new solutions, a safety group may be created to share experiences and approaches, such as a software safety working group. A government agency might create a safety working group for a large project composed of representatives from the agency offices and the prime contractor. A prime contractor may create a safety working group composed of the safety managers for all the subcontractors.

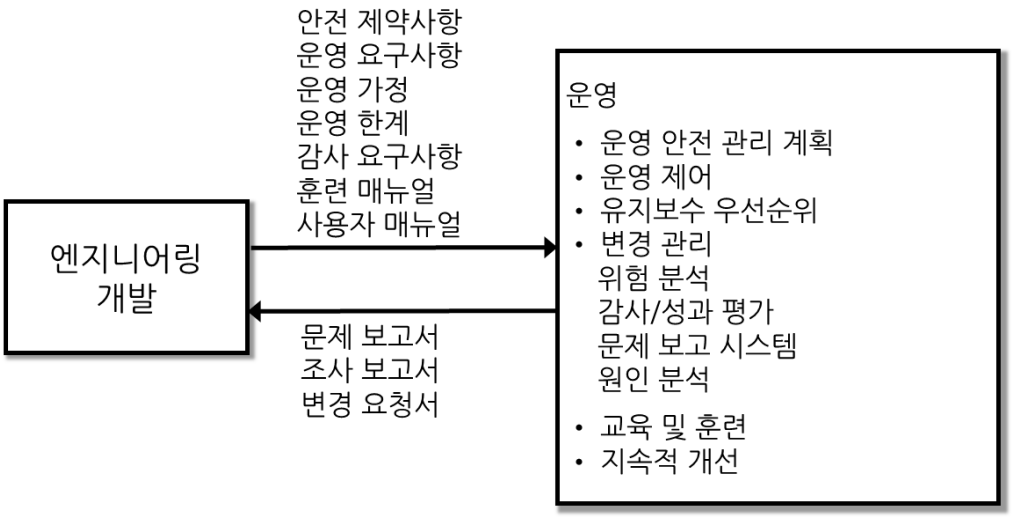


그림 7.3. 안전 활동을 지원하기 위해 개발과 운영 사이에 필요한 정보 채널

모든 안전 컨트롤 스트럭처 컴포넌트 간 전달되어야 하는 정보의 설명(similar description)을 식별하고 문서화하여야 한다.

의사소통과 조정을 위해 국방부가 고안한 매우 효과적인 수단 중 하나는 *워킹 그룹(working group)*이다. 국방부 프로젝트는 수년에 걸쳐 진행되고 매우 크고 복잡하며 종종 지리적으로나 조직적으로 분산되어 있는 다수의 참가자가 참여할 수 있다. 이러한 프로젝트에서 조정과 의사소통은 중요한 문제가 될 수 있다. 여기에 대한 해결책 중 일부는 국방부가 프로젝트의 상위 관리자가 되는 계층적 구조로 제공될 수 있지만, 주 계약업체(Prime Contractor)가 있어 이들이 시스템 엔지니어링이나 하청업체 간의 조정을 담당한다. 이러한 계층적 구조가 작동하려면 의사소통이 중요해진다. 워킹 그룹은 이러한 조정 및 의사소통에 성공적이었으며 이는 복잡도가 낮은 프로젝트에도 적용할 수 있다.

워킹 그룹은 안전 컨트롤 스트럭처의 두 계층 컴포넌트 사이 또는 동일한 수준의 둘 이상의 컴포넌트 사이의 인터페이스를 제공한다. 이러한 그룹의 구성원은 업무(efforts)를 조정하고 해결되지 않은 안전 이슈의 상태를 보고하며 독립적인 안전 업무에 대한 정보를 공유할 책임이 있다. 컨트롤 스트럭처의 각 레벨에 워킹 그룹이 있을 수 있으며 워킹 그룹의 구성원과 책임은 그들이 운영하는 레벨에 따라 달라질 수 있다. 기업 워킹 그룹은 서로 다른 부서나 프로그램의 안전 관리자로 구성될 수 있으며 하위 레벨의 워킹 그룹은 제품의 서로 다른 부분을 설계하는 그룹들의 대표들로 구성될 수 있다. 새로운 솔루션이 필요한 특별한 유형의 안전 문제가 있는 경우 경험과 접근법을 공유하기 위해 소프트웨어 안전 워킹 그룹과 같은 안전 그룹을 만들 수 있다. 정부 기관은 대규모 프로젝트를 위해 공공기관(agency office) 및 주 계약업체의 대표로 구성된 안전 워킹 그룹(safety working group)을 만들 수 있다. 주 계약업체는 모든 하청 업체의 안전 관리자로 구성된 안전 워킹 그룹을 만들 수 있다.

Tips for designing communication and coordination

- Make sure information necessary for decision making involving safety is available to decision makers.
- Provide communication channels and a way to coordinate activities among those with overlapping safety responsibilities
- Make sure safety activities are not fragmented and uncoordinated.
- Define required interactions and ensure that necessary information flow among them is defined and used
- Identify and document necessary safety-related communication channels among all components of the safety control structure.
- Ensure feedback and coordination channels exist and are working
- Consider establishing safety working groups

Managing and controlling change

Most accidents occur after some type of change.[34] This fact is not surprising as continuing to do what one has done without any changes in behavior or the environment should theoretically result in the same consequences. Adaptation and change is an inherent part of any system and is required for an organization to thrive. The problem is not change, but *unsafe* change. The SMS, therefore, must have carefully designed controls to prevent unsafe changes and to detect them if they occur despite the efforts to prevent them. The goal is to allow change as long as it does not violate the safety constraints. Two types of changes need to be considered: planned changes and unplanned changes.

Planned Changes: The SMS must continue to be effective in the face of planned changes, including changes in the organization, in human behavior and processes, in operations (including changes to maintenance phases and back to operational phases), or in the environment. Most companies have (and, if not, they should have) Management of Change (MOC) policies that require all planned changes to be evaluated with respect to their impact on safety. The cost of such evaluations will depend on the scope of the change, the quality of documentation, and how the original hazard analysis was performed. STPA incorporates complete traceability from identified hazards to their causal scenarios and to the design features that are used to prevent them. Without such traceability, the cost of reanalysis after a change may be impractical and therefore skipped.

In fact, MOC procedures are often skipped in practice, resulting in losses that need never have occurred. The safety control structure must have assigned responsibility for enforcement of MOC procedures and feedback channels to determine whether they are being followed and, if not, why. If the cause of skipping them is in the MOC change procedures themselves (too onerous, too difficult, too time consuming, etc.), then the MOC procedures may need to be changed.

The cost of implementing the MOC procedures will depend on the quality of the system documentation and how the original hazard analysis was done. A design goal for STPA was minimization of the cost in evaluating changes.

Unplanned changes: Unplanned changes present more difficult challenges. There needs to be (1) a way to identify potentially unsafe changes and (2) ways to respond to these changes. Leading indicators, as described in the previous chapter, should be created from assumptions made in the system (including the SMS) design and safety analysis process. Identifying leading indicators requires recording

[34] See Nancy G. Leveson, *Safeware*, Addison-Wesley Publishers for references.

의사소통 및 조정(coordination)을 설계하는 팁

- 안전을 포함한 의사 결정에 필요한 정보가 의사 결정권자에게 제공되는지 확인한다.
- 안전 책임이 중복되는 사람들 간의 의사소통 채널과 활동 조정 방법을 제공한다.
- 안전 활동이 단편화되거나 조정되지 않은 상태가 아님을 확실히 한다.
- 필요한 상호작용을 정의하고, 사람들 간에 필요한 정보의 흐름이 정의되고 사용됨을 보장한다.
- 안전 컨트롤 스트럭처의 모든 컴포넌트 간에 필요한 안전-관련 의사소통 채널을 식별하고 문서화한다.
- 피드백 및 조정 채널이 존재하고 작동함을 보장한다.
- 안전 워킹 그룹 개설을 고려한다.

변경의 관리 및 제어(Managing and controlling change):

대부분의 사고는 어떤 유형의 변경[34] 이후에 발생한다. 행동이나 환경에 아무런 변경 없이 해오던 일을 계속하는 것은 이론적으로는 동일한 결과를 야기하게 되므로, 이 사실은 놀라운 일이 아니다. 적응과 변경은 모든 시스템에서 본질적인 부분이며 조직의 번영을 위해서도 필요하다. 변경이 문제가 아니라 안전하지 않은 변경이 문제이다. 따라서 SMS에는 안전하지 못한 변경을 예방하고, 만약 이런 예방의 노력에도 불구하고 문제가 발생한다면 이를 탐지할 수 있는, 신중하게 설계된 컨트롤이 있어야 한다. 목표는 안전 제약사항을 위반하지 않는 한도 내에서 변경을 허용하는 것이다. 두 가지 유형의 변경(계획된 변경과 계획되지 않은 변경)을 고려한다.

계획된 변경(planned changes): SMS는 계획된 변경에 직면하여서도 계속해서 효과적이어야 한다. 계획된 변경에는 조직, 사람의 행동과 프로세스, 운영(유지보수 단계로의 변경 및 운영 단계로의 회귀 포함) 또는 환경의 변경도 포함된다. 대부분의 기업은 변경 관리(MOC, Management of Change) 정책이 있다(만약 없다면, 확보해야 한다). MOC는 모든 계획된 변경에 대해 안전에 미치는 영향의 평가를 요구한다. 그러한 평가(evaluation) 비용은 변경의 범위, 문서의 품질, 그리고 기존에 위험 분석이 수행된 방법에 따라 달라진다. STPA는 식별된 위험에서부터 그것들의 원인 시나리오와 이를 예방하기 위해 사용된 설계 기능(design feature)에 이르기까지 완전한 추적성을 가진다. 이러한 추적성이 없으면 변경 후 재분석 비용이 터무니없이 비쌀 수 있으므로, 생략하게 된다.

사실, MOC 절차가 실제로는 종종 생략되기 때문에 절대로 일어나서는 안 되는 손실이 발생한다. 안전 컨트롤 스트럭처에는 MOC 절차를 강제하기 위해 할당된 책임이 포함되어야 하며, MOC 절차가 지켜졌는지, 지켜지지 않았다면 그 이유를 알려주는 피드백 채널도 포함되어야 한다. 절차를 생략하는 원인이 MOC 변경 절차 자체(너무 힘들고, 너무 어려우며, 너무 오래 걸리는 것 등)에 있다면 MOC 절차를 변경해야 할 수도 있다.

MOC 절차 실행 비용은 시스템 문서화의 품질과 원래 위험 분석이 어떻게 수행되었는지에 따라 달라진다. STPA의 설계 목표는 변경 사항의 평가 비용을 최소화하는 것이었다.

계획되지 않은 변경(unplanned changes): 계획되지 않은 변경은 더 어려운 문제이다. 여기에는 (1) 잠재적으로 안전하지 않은 변경을 식별하는 방법과 (2) 이러한 변경에 대응하는 방법이 필요하다. 이전 장에서 설명한 바와 같이 선행지표는 시스템(SMS 포함) 설계 및 안전 분석 프로세스에서 만들어

[34] Nancy G. Leveson의 Safeware(Addison-Wesley 출판사)를 참조한다.

assumptions during design and development of the organizational structure, the SMS, and the products and workplace. STPA control structures, UCAs, and causal scenarios provide information necessary to create effective leading indicators. If the causal scenarios cannot be eliminated, information from the STPA analysis can be used to create shaping and hedging actions as well as to identify signposts and assumptions to check in audits and performance assessments.

A common reason for unsafe changes to occur is risk re-evaluation. After a period of few or no losses, people begin to re-evaluate their view of the risks downward. Nothing may have changed so risk has not really decreased, but people may believe it has. Under pressure, they start to violate their own rules and justify it by arguing that safety will not be affected. The processes in the safety control structure need to interrupt this risk re-evaluation process before safety margins are eroded. One requirement is the communication of appropriate information about the actual level of risk. Management needs to be aware of the state of risk in the processes they are controlling. That requires appropriate feedback about the state of the designed safety controls.

In addition, there needs to be an alerting function to a person with responsibility when behavior is contrary to the true level of risk. The key to preventing this phenomenon is to allow flexibility in how safety goals are achieved and to provide information that allows accurate risk assessment by decision makers.

An effective SMS is characterized by continual vigilance against degradation of the system over time. The SUBSAFE program, as an example, puts a lot of effort into combating what they describe are their three most difficult challenges:

1. Ignorance: Not knowing;
2. Arrogance: Pride, self-importance, conceit, or assumption of intellectual superiority and presumption of knowledge that is not supported by fact;
3. Complacency: Satisfaction with one's accomplishments accompanied by a lack of awareness of actual dangers or deficiencies.

Much of the design and focus of the SUBSAFE program is aimed at combating these three challenges, and the design of any SMS should include steps to provide vigilance against them.

진 가정으로부터 생성되어야 한다. 선행지표를 식별하려면 조직 구조, SMS, 제품 및 작업장을 설계하고 개발하는 과정에서의 가정(assumption)을 기록해야 한다. STPA 컨트롤 스트럭처, UCA, 원인 시나리오는 효과적인 선행지표를 만드는 데 필요한 정보를 제공한다. 원인 시나리오를 제거할 수 없는 경우에는 STPA 분석을 통해 얻은 정보를 사용하여 쉐이핑 및 헷징 액션을 수행하고 감사(audit)와 성과 평가를 체크하기 위한 싸인포스트와 가정을 식별할 수 있다.

안전하지 않은 변경이 발생하는 흔한 이유는 리스크의 재평가(re-evaluation) 때문이다. 손실이 거의 또는 전혀 없는 기간 이후에는 사람들은 리스크에 대한 자신의 관점을 낮은 방향으로 재평가하기 시작한다. 아무것도 변하지 않았기 때문에 리스크는 실제로 감소하지 않았으나 사람들은 리스크가 감소했다고 믿을 수 있다. 압박이 있는 경우에는 그들은 자신의 규칙을 위반하고 안전이 영향을 받지 않았다고 주장하며 규칙 위반을 정당화하기 시작한다. 안전을 위한 여유분(safety margin)이 없어지기 전에 안전 컨트롤 스트럭처의 프로세스는 이러한 리스크 재평가 프로세스를 중단시켜야 한다. 한 가지 요구사항은 실제 리스크 수준에 대한 적절한 정보를 전달하는 것이다. 경영진은 그들이 제어하고 있는 프로세스에서의 리스크 상태를 알고 있어야 한다. 이를 위해서는 설계된 안전 컨트롤 상태에 대한 적절한 피드백이 필요하다.

또한, 행동이 실제 리스크 수준에 위배되는 경우 책임을 가진 사람에게 경고하는 기능이 필요하다. 이러한 현상을 예방하는 열쇠는 안전 목표를 달성하는 방법에 유연성을 갖도록 허용하고 의사 결정권자가 정확한 리스크 평가를 내릴 수 있도록 정보를 제공하는 것이다.

효과적인 SMS는 시간이 지남에 따른 시스템의 저하를 지속적으로 경계한다는 특징이 있다. 예를 들어, SUBSAFE 프로그램은 다음의 가장 어려운 세 가지 문제를 해소하는 데 많은 노력을 기울였다.

1. 무지(ignorance): 알지 못함
2. 오만(arrogance): 자부심, 자존감, 자만심 또는 지적 우위의 가정(assumption) 및 사실이 뒷받침되지 않은 지식에 대한 억측(presumption)
3. 안주(安住, complacency): 실제 위험(danger) 또는 결함(deficiency)에 대한 인식 부족으로 인해 성과에 만족

SUBSAFE 프로그램의 설계와 관심의 대부분은 위의 세 가지 문제를 해결하는 데 목적이 있으며 모든 SMS의 설계에는 이 세 가지 문제를 경계하는 단계(step)가 포함되어야 한다.

Tips for managing and controlling change

- Design controls and MOC policy to prevent unsafe changes and detect if they occur
- Evaluate all planned changes, including temporary ones, for their potential impact on safety
- Assign responsibility for ensuring MOC procedures are enforced and are being followed. If they are not, find out why and fix the problems.
- Create documentation and procedures that minimize the cost of performing the MOC procedures.
- Create ways to identify unplanned changes that could be unsafe ones and to respond to these changes.
    - Devise assumption-based leading indicators and create a risk management program that effectively monitors and responds when potentially unsafe changes are identified.
    - Record assumptions during design and development of the organizational structure, the SMS, the products, and the workplace.
    - Create shaping and hedging actions and a leading indicator checking program including audits and performance checking as well as signposts
    - Implement leading indicators to signal when controls are becoming ineffective
- Ensure that decision makers have information about the current level of risk and state of the designed safety controls
- Assign responsibility to respond when feedback shows that behavior does not match the true level of risk.
- Remain vigilant against the degradation of the safety control structure and the safety culture over time and any increase in complacency.

Designing and Encouraging Feedback

Accurate risk assessment requires information flow and properly functioning feedback channels. Cultural problems are a common reason for problems in feedback. There are three general ways to implement feedback channels: audits and performance assessments, accident/incident causal analysis, and reporting systems.

Audits and Performance Assessments: There are many types of audits and performance assessments, but those whose goal is to evaluate the state of safety start from the safety constraints and the assumptions in the design of the safety controls. Audits should involve not just the products and processes but also the safety management system itself and the effectiveness of the controls designed to ensure that losses are prevented. At least some part of a safety audit and performance assessment should be focused on the operation of the safety control structure as it was designed and assumed it would operate.

The entire safety control structure must be audited and not just the lower levels. In the SUBSAFE program, even the top Admirals are subject to a SUBSAFE audit. Not only does this ensure that the program is operating as assumed, but it also provides a positive cultural component to the audit in that all employees see that even the leaders are expected to follow the SUBSAFE rules and that top management are willing to accept and resolve audit findings just like any other member of the SUBSAFE community. People at lower levels of the SUBSAFE safety control structure participate in the performance assessment of those above them, providing a visible sign of the commitment of the entire program to safety and the importance of everyone in preventing losses. Accepting being audited and

변경을 관리하고 제어하기 위한 팁

- 안전하지 않은 변경을 예방하고 발생 여부를 탐지하기 위한 컨트롤 및 MOC 정책을 설계한다.
- 임시적인 변경을 포함한 모든 계획된 변경에 대해, 이러한 변경이 안전에 미치는 잠재적인 영향을 평가한다.
- MOC 절차가 시행되고 준수되도록 책임을 할당한다. MOC 절차가 준수되지 않는 경우 이유를 찾아 문제를 해결한다.
- MOC 절차 수행 비용을 최소화하는 절차와 문서를 만든다.
- 안전하지 않을 수 있는, 계획되지 않은 변경을 식별하는 방법과 그러한 변경에 대응하는 방법을 만든다.
  - 가정 기반 선행지표를 생성하고, 잠재적으로 안전하지 않은 변경이 식별되면 이를 효과적으로 모니터링하고 대응하는 리스크 관리 프로그램을 만든다.
  - 조직 구조, SMS, 제품 및 작업장의 설계 및 개발 과정에서의 가정(assumption)을 기록한다.
  - 싸인포스트 뿐만 아니라 감사(audit)와 성과 점검을 포함하는 쉐이핑 및 헷징 액션과 선행지표 점검 프로그램을 만든다.
  - 컨트롤이 비효과적으로 변할 때, 선행지표가 신호를 보내도록 구현한다.
- 의사 결정권자가 현재의 리스크 수준 및 설계된 안전 컨트롤 상태에 대한 정보를 알도록 보장한다.
- 동작이 실제 리스크 수준과 일치하지 않는다는 피드백이 있을 시 이에 대응할 책임을 할당한다.
- 시간의 지남에 따른 안전 컨트롤 스트럭처 및 안전 문화의 저하와 안주(安住)의 증가를 경계한다.

피드백의 설계 및 장려(Designing and Encouraging Feedback):

정확한 위험 평가를 위해서는 정보의 흐름과 적절하게 작동하는 피드백 채널이 필요하다. 피드백에서 발생하는 문제의 흔한 원인은 문화적인 것이다. 피드백 채널을 시행하는 세 가지 일반적인 방법은 감사(audit) 및 성과 평가, 사고/사건 원인 분석, 그리고 보고 시스템이다.

감사 및 성과 평가(Audits and Performance Assessments): 많은 종류의 감사 및 성과 평가가 있지만 이들의 공통된 목표는 안전 컨트롤 설계에서의 안전 제약사항과 가정에서 시작하는 안전 상태(state of safety)를 평가하는 것이다. 감사는 제품과 프로세스뿐만 아니라 안전 관리 시스템 자체와 손실 예방을 위해 설계된 컨트롤의 효과도 포함해야 한다. 적어도 안전 감사 및 성과 평가의 일부에서는 안전 컨트롤 스트럭처의 동작이 설계된 대로, 그리고 그렇게 동작할 것이라 가정한대로 수행되는지에 중점을 두어야한다.

안전 컨트롤 스트럭처는 하위 수준에 대해서만 감사하는 것이 아니라 전체적으로 감사해야 한다. SUBSAFE 프로그램에서는 상위 해군제독조차도 SUBSAFE 감사의 대상이 된다. 이는 프로그램이 가정한대로 운영됨을 보장할 뿐만 아니라 모든 직원들이 리더조차도 SUBSAFE 규칙을 준수한다고 생각하며 SUBSAFE 커뮤니티의 다른 멤버와 마찬가지로 최고 경영진도 기꺼이 감사 결과를 인정하고 해결하려고 한다는 점에서 긍정적인 문화적 구성요소를 제공한다. SUBSAFE 안전 컨트롤 스트럭처의 하위

implementing improvements as a result, that is, leading by example, is a powerful way for leaders to convey their commitment to safety and to its improvement.

Participatory and non-punitive audits can be very effective. The goal of such an audit is that it be a constructive learning experience and not a judgmental process. It should be viewed as a chance to improve safety rather than a way to evaluate employees. Instead of the usual observation-only audit process by outside audit companies, experts from other parts of the organization not directly being audited should make up the audit team. Various stakeholders may play a role in the audit and even individuals from the group being audited may be on the audit team. The goal should be to create an attitude that this is a chance to improve our practices and provide a learning activity for everyone involved, including the auditors.

The audit itself should involve continuous communication with those being audited so as to obtain full understanding of any identified problems and potential solutions. Unlike the usual rules for outside audits, in a participatory audit immediate feedback should be provided and solutions discussed. Doing this will reinforce the understanding that the goal is to improve safety and not to punish or evaluate those involved. It also provides an opportunity to solve problems when a knowledgeable team is on the spot and not after the usual written report is provided months after the audit actually occurs.

An important goal of safety audits and performance assessments is to measure the level of safety knowledge and training that actually exists, not what managers think exists or what exists in the training programs and user manuals. The audits can provide important feedback about potential improvement to the training and education activities. In keeping with the non-punitive audit philosophy, knowledge assessments should not be used in a negative way that is viewed as punishment by those being assessed. Instead, the goal should be to improve training and education efforts.

Incident and Accident Investigation: Incident and accident investigation provides clear evidence that the SMS is not working as designed or expected. The investigation procedures must be embedded in an organizational structure that allows exploitation of the results. All systemic factors involved must be identified and not just the symptoms or technical factors. Assigning blame should not be the goal; rather the goal should be to find out why the safety control structure was not effective in preventing the loss or near loss. That means that the entire safety control structure must be investigated to identify each component's potential contribution to the adverse events.

Managers should not be responsible for investigating incidents that occur in their chain of command: investigators and causal analysts must be managerial and financially independent from those in the immediate management structure involved. Using trained teams with independent budgets and with high-level and independent management should be considered.

Investigation requires more than just writing a report. There must be assignment of responsibility for ensuring that appropriate measures are taken to strengthen the aspects of the safety control structure that contributed to the events. Then there should be follow-up to ensure that the fixes were effective. Too often, fixes are made but there is no attempt to determine whether the fixes were successful in improving safety management. Finally, the findings should be used as input to future audits and performance assessments. If there is a reoccurrence of the same factors that led to incidents and accidents in the past, there needs to be an investigation of why those factors were never corrected or why they reoccurred even if they were removed for a while. If fixes are not effective in removing the causes of incidents, then an investigation of the process of creating recommendations and responding to them is warranted to identify any weaknesses in these processes in the organization and to improve them. That is, not only must the factors involved in the incident be corrected but also the process that led to inadequate fixes being implemented after previous incidents or accidents.

수준에 있는 사람들이 상위 수준에 대한 성과 평가에 참여하게 하여, 안전에 대한 전체 프로그램의 약속 및 손실 예방에 있어서 모든 사람의 중요성을 명확히 보여준다. 리더가 감사(audit) 받는 것을 수용하고 결과적으로 개선을 수행하는 것은, 즉, 모범을 보이는 것은, 안전과 개선에 대한 리더의 약속을 전달하는 강력한 방법이다.

참여 및 비처벌적(non-punitive) 감사는 매우 효과적일 수 있다. 그러한 감사의 목표는 건설적인 학습 경험을 쌓는 것이지 판정을 위한 것이 아니다. 감사는 직원을 평가하는 수단이 아닌 안전을 향상시키는 기회로 여겨져야 한다. 외부 감사 기업이 수행하는, 감시만을 위한 보통의 감사 프로세스 대신, 조직의 다른 파트(직접적으로 감사를 받지 않는 파트) 전문가로 감사팀을 구성해야 한다. 다양한 이해관계자가 감사에 참여할 수 있으며 감사를 받는 그룹에 속한 개인도 감사팀에 포함될 수 있다. 감사의 목표는, 감사가 관행을 개선하는 기회이며 감사관(auditor)을 포함한 관련된 모든 사람들이 학습하는 기회라고 여기는 태도를 갖추는 것이다.

식별된 문제와 잠재적 해결책 모두를 완전히 이해할 수 있도록 감사에서는 감사 대상자와의 지속적인 의사소통이 이루어져야 한다. 외부 감사(outside audit)의 일반적인 규칙과는 달리, 참여 감사(participatory audit)에서는 즉각적으로 피드백을 주고 해결책을 논의해야 한다. 이를 통해 우리의 목표가 관련자를 처벌하거나 평가하는 것이 아닌 안전성 향상에 있다는 점을 강조할 수 있다. 또한 이는 감사가 수행되고 수 개월이 지난 후 일상적인 서면 보고를 할 때 문제를 해결하는 것이 아니라 전문지식을 갖춘(knowledgeable) 팀이 실제 현장에 있을 때 문제를 해결하는 기회를 제공한다.

안전 감사 및 성과 평가의 중요한 목표는 실제 존재하는 안전 지식 및 훈련 수준을 측정하는 것이며 관리자가 생각하는 것 또는 교육 프로그램 및 사용자 매뉴얼에 존재하는 것을 측정하는 것이 아니다. 감사는 훈련 및 교육 활동을 잠재적으로 개선하기 위한 중요한 피드백을 제공할 수 있다. 비처벌적 감사 철학에 따라 지식의 평가는 평가 대상자가 이를 처벌이라고 생각하게 되는 부정적인 방식으로 사용되어서는 안 된다. 훈련 및 교육 노력을 개선하는 것이 목표이다.

사건 및 사고 조사(Incident and Accident Investigation): 사건 및 사고 조사는 SMS가 설계대로 또는 예상대로 작동하지 않는다는 명확한 증거를 제공한다. 조사의 절차는 결과가 활용될 수 있도록 조직 구조 내에 포함되어야 한다. 조사에는 증상이나 기술적인 요인뿐만 아니라 사건·사고에 관련된 모든 시스템적인 요인들이 식별되어야 한다. 조사의 목표는 비난할 대상을 찾는 것이 되어서는 안 된다; 조사의 목표는 손실이나 손실이 발생할 뻔한 것(near loss)을 방지하는 데에 안전 컨트롤 스트럭처가 효과적이지 않은 이유를 발견하는 것이어야 한다. 이는 부정적 이벤트(adverse events)에 대한 각 컴포넌트의 잠재적 기여도를 식별하기 위해 전체 안전 컨트롤 스트럭처를 조사해야 함을 의미한다.

관리자는 자신의 명령 체계(chain of command)에서 발생하는 사건에 대해 조사할 책임이 없어야 한다: 조사관 및 원인 분석가는 관련된 직속 관리 구조에 있는 사람과 경영적, 재정적으로 독립적이어야 한다. 별도의 예산과 높은 수준의 독립적 관리를 갖춘 숙련된 팀을 고려해야 한다.

조사는 단순히 보고서를 작성하는 것 이상을 요구한다. 이벤트에 기여하는 안전 컨트롤 스트럭처를 강화하기 위한 적절한 방안(measure)이 보장될 수 있도록 책임이 할당되어야 한다. 그런 다음 이 수정사항이 효과적이었는지 확인하기 위한 후속 조치(follow-up)가 있어야 한다. 수정이 이루어졌지만 이것이 안전 관리를 개선하는 데 성공적인지를 확인하지 않는 경우가 많다. 마지막으로, 발견된 사항은 향후 감사 및 성과 평가의 입력으로 사용되어야 한다. 과거에 사건 및 사고로 이어졌던 요인이 재발하였다면 그러한 요인이 왜 수정되지 않았는지 또는 잠시 제거되었음에도 왜 다시 발생했는지에 대한 조사가 필요하다. 만약, 수정 작업이 사건의 원인을 제거하는데 효과적이지 않았다면 권고사항을 만들고 그에 대응하는 절차의 조사가 보장되어야 하며, 이를 통해 조직 절차의 약점을 파악하고 개선해야 한다. 즉, 사

Reporting Systems: Reporting systems are critical. Often, after an accident, it is found that the same events occurred multiple times in the past but were never reported or, if reported, were never corrected. These events may involve near misses. For example, several aircraft may not fly an approach correctly, but because it did not result in an accident they may not report it, even if a reporting system exists. Not until an accident occurs is action taken.

A common finding in accident reports is that a reporting system existed but was not used. These reports then usually include recommendations to train people on how to use the reporting system and to require that they use it. This recommendation assumes that the problem is with the potential reporter and not with the design of the reporting system itself.

Examination of the reporting system itself may find that it is difficult or awkward to use and that information reported appears to go into a black hole. People may believe that there is no point in going through the official reporting system because the organization will not do anything anyway about the factors reported. Often, those finding problems bypass the reporting system and simply go directly to a person or group they believe may be able to solve the problems. While this might be effective and efficient in the short term, it may lead to the same problems occurring in the future because the systemic causes are not eliminated. In some cases, reporting is inhibited by a fear that the information reported will be used against the reporter. Anonymous reporting systems can be helpful here.

Another factor to consider is that often events are not reported by front-line operators because they identified the problem before it created a perception of risk or they may have perceived it as only their own error. Most people are not trained to recognize risk when it is created as part of a normal job process. People accept the flaws in design as “normal” and perhaps already known so rather than reporting them, they just work with or around them. A related factor is that people will generally not report hazardous events when those events do not meet the criteria for required reporting. Near misses may fall in the latter category.

In general, reporting needs to be encouraged. This can be accomplished by maximizing accessibility, minimizing anxiety, and acting on the information obtained. Reporting forms or channels should be easily and ubiquitously available and not cumbersome to fill in or use. To minimize anxiety, there should be a written policy on what the reporting process is; the consequences of reporting; and the rights, privileges, protections, and obligations of those doing both the reporting and those following up on the reports. Without a written policy, ambiguity exists and people will disclose less. Alternatively, ambiguity about who is responsible for following up may lead to everyone assuming that someone else will take care of it.

Finally, encouraging reporting involves providing feedback. Immediately after a report is created, the person who provides the information should be informed that the report was received, assured that it will be investigated, and thanked for their input. A second crucial component is providing feedback later about the results of any investigation and any steps that were taken as a result to prevent a reoccurrence. Reporters should not feel like their concerns are being ignored.

고와 관련된 요인을 수정해야 할 뿐만 아니라 부적절한 수정(이전의 사건이나 사고 이후에 수정된 것)을 야기한 프로세스도 수정되어야 한다.

보고 시스템(Reporting Systems): 보고 시스템은 매우 중요하다. 동일한 이벤트가 과거 수차례 발생했지만 보고되지 않았거나 보고되었다 하더라도 수정되지 않았음을 사고가 발생한 후 알게 되는 경우가 종종 있다. 이러한 이벤트들은 니어 미스(near miss)와 관련 있을 수 있다. 예를 들어 항공기가 정확히 어프로치(approach)하지 못했지만 사고가 발생하지 않아 보고 시스템이 있더라도 보고하지 않을 수 있다. 이 경우 사고가 발생할 때까지 조치가 취해지지 않는다.

보고 시스템이 있지만 실제 사용되지 않았다는 사실을 사고 보고서에서 흔히 발견할 수 있다. 이러한 보고서에는 대개 보고 시스템의 사용 방법을 훈련하고 보고 시스템을 사용하라는 권고사항이 들어 있다. 이 권고사항은 보고자(reporter)에게 문제가 있다고 보는 것이며 보고 시스템 자체의 설계에 문제가 있다고 보는 것이 아니다.

보고 시스템을 조사해보면 시스템이 사용하기 어렵거나 불편하며 보고된 정보가 사라진다는 것을 알 수 있다. 사람들은 보고된 요인에 대해 조직이 아무런 조치를 취하지 않기 때문에 공식적인 보고 시스템을 거칠 필요가 없다고 믿을 수도 있다. 발견된 문제가 보고 시스템을 통하지 않고 문제를 해결할 수 있는 사람이나 그룹에게 단순하게 직접 전달되는 경우도 종종 있다. 이것은 단기적으로는 효과적이고 효율적일 수 있지만, 시스템적인 요인이 제거되지 않기 때문에 앞으로도 동일한 문제의 발생을 야기할 수 있다. 어떤 경우에는 보고된 정보가 보고자에게 불리하게 사용될 것이라는 두려움 때문에 보고가 되지 않을 수도 있다. 이 때에는 익명의 보고 시스템이 도움이 될 것이다.

고려해야 할 또 다른 요인은 일선 운영자(front-line operator)가 문제가 리스크로 인지되기 전에 이미 그 문제를 식별했거나, 자신만의 잘못으로 인지하여 이벤트를 보고하지 않는 경우가 종종 있다는 것이다. 정상적인 작업 프로세스의 일부로 리스크가 발생하지만 대부분의 사람들은 리스크를 인식하는 훈련을 받지 않는다. 사람들은 설계상의 결함을 “정상”으로 받아들이며, 어쩌면 이미 알려진 것이라고 생각하여 보고하지 않고 그러한 리스크가 있는 상태로 작업한다. 관련된 요인으로, 사람들은 위험한 이벤트가 보고에 필요한 기준을 충족하지 않으면 일반적으로 보고하지 않는다는 것이다. 니어 미스(near miss)는 후자에 속한다고 볼 수 있다.

일반적으로 보고는 장려되어야 한다. 이를 위해서는 접근성을 높이고 우려를 불식시키며, 취득한 정보를 기반으로 조치를 수행함으로써 달성할 수 있다. 보고 양식이나 채널은 쉽게 언제어디서나 가용해야 하며 작성하거나 사용하기가 번거롭지 않아야 한다. 우려를 불식시키기 위해서는 보고 프로세스가 무엇인지; 보고의 결과; 보고와 보고의 후속조치를 수행하는 사람의 권리, 특권, 보호 및 의무에 대한 서면 정책이 있어야 한다. 서면 정책이 없으면 모호함이 존재하여 사람들은 정보를 덜 공개할 것이다. 후속 조치를 책임질 사람이 모호한 경우에는 모든 사람이 다른 누군가가 후속 조치를 할 것이라고 생각할 수 있다.

마지막으로, 보고를 장려하기 위해서는 피드백을 제공해야 한다. 보고된 직후, 정보를 제공한 사람은 보고가 접수되었고 조사가 이루어질 것이며 보내준 정보에 감사하다는 답변을 받아야 한다. 두 번째 중요한 것은 이후 재발 방지를 위해 취해진 모든 조사 결과와 모든 단계에 대한 피드백을 제공하는 것이다. 보고자들이 그들의 우려가 무시당한다고 느껴서는 안 된다.

Tips for designing and encouraging feedback

- Design and ensure the continued efficacy of audits, performance assessments, and reporting systems.
- Audit the safety control structure itself (including all levels) and the effectiveness of the designed controls.
- Design audits so they are constructive learning experiences and not a judgmental process.
  - Include as participants members of the groups that are being audited.
  - Use audits as a way to improve safety and as a learning activity and not to evaluate employees.
- Take advantage of audits to evaluate the effectiveness of training and education activities and use them to provide feedback (knowledge assessment) to be used for improving training activities.
- Create effective system-level accident/incident causal analysis procedures that focus on *why* and not *who***.**
- Create incident and investigation procedures based on STAMP and systems thinking (such as CAST) that identify systemic factors and not just the symptoms of the deeper problems.
- Embed the investigation procedures in an organizational structure that allows exploitation of the results. Assigning blame or finding a "root cause" should not be the goal.
- Accident investigation should be managerially and financially from those in the immediate management structure involved. Consider using highly trained teams with independent budgets and high-level management. Follow up on recommendations to determine whether they were effective and, if not, then why.
- Ensure reporting systems are easy to use and available and anonymous reporting channels exist.
  - Encourage reporting and train people to know when it should be used.
  - Provide a written policy
  - Maximize accessibility, minimize anxiety, and act on information obtained.
  - Provide feedback to those using the reporting channels. Reporters need to feel like their concerns are not being ignored.

Risk Management

Sometimes the risk management system is called the "safety management system," but they are not the same. The SMS is a larger concept, that may (and usually does) involve risk management activities. Risk management is the set of activities associated with identifying hazards, analyzing them, and using this information to reduce losses through the design of products, processes, services, and workplaces. These activities will be embedded somewhere in the safety control structure. While risk management is comprised of the technical activities involved in preventing hazards, the SMS also contains organizational, managerial, and cultural aspects of safety.

STAMP implies a broader or at least different definition of risk. Risk has traditionally been defined as the severity and likelihood of hazards or accidents occurring. In contrast, in STAMP

Risk is defined in terms of the effectiveness of the controls used to enforce safe system behavior, i.e., the design and operation of the safety control structure.

피드백을 설계하고 장려하기 위한 팁

- 감사(audit), 성과 평가 및 보고 시스템의 지속적인 효능을 설계하고 보장한다.
- 안전 컨트롤 스트럭처 자체(모든 수준 포함)와 설계된 제어의 효과를 감사한다.
- 감사는 판정 프로세스가 아닌, 건설적인 학습 경험이 되도록 설계한다.
  - 감사를 받는 그룹의 구성원을 감사 참여자로 포함시킨다.
  - 감사를 직원 평가 용도로 사용하지 않고 안전을 개선하는 방법으로, 학습하는 활동으로 사용한다.
- 감사를 활용하여 훈련 및 교육 활동의 효과를 평가하고 이를 활용하여 훈련 활동 개선에 사용할 피드백(지식 평가)을 제공한다.
- '누가'가 아닌 '왜'에 초점을 맞춘 효과적인 시스템 수준의 사고/사건 원인 분석 절차를 만든다.
- STAMP와 시스템 사고(systems thinking)에 기반한 사고 및 조사 절차(예 : CAST)를 만들어, 심층 문제의 증상뿐만 아니라 시스템적 요인을 식별한다.
- 결과를 활용할 수 있는 조직 구조에 조사 절차를 포함시킨다. 비난할 상대를 찾거나 "근본 원인(root cause)"을 찾는 것이 목표가 되어서는 안 된다.
- 사고 조사는 관련된 직접적인 관리 구조의 직원들과는 경영적 및 재정적으로 독립되어 이루어져야 한다. 별도의 예산과 높은 수준의 관리를 갖춘 고도로 숙련된 팀을 활용할 것을 고려해야 한다. 권고사항에 대해 후속 조치하고 이것이 효과적인지, 효과적이지 않다면 이유는 무엇인지를 결정한다.
- 보고 시스템이 사용하기 쉽고 가용하며, 익명의 보고 채널이 있음을 보장한다.
  - 보고를 장려하고 언제 사용해야 하는지 알도록 사람들을 훈련시킨다.
  - 서면으로 작성된 정책을 제공한다.
  - 접근성을 높이고 우려를 불식시키며 취득한 정보를 바탕으로 조치한다.
  - 보고 채널을 사용하는 사람들에게 피드백을 제공한다. 보고자들은 그들의 우려가 무시되지 않는다는 느낌을 받아야 한다.

리스크 관리(Risk Management):

리스크 관리 시스템을 "안전 관리 시스템(SMS)"이라고 부르기도 하지만, 이 둘은 동일하지 않다. SMS는 보다 큰 개념으로 리스크 관리 활동을 포함할 수 있다(일반적으로 그러함). 리스크 관리는 위험을 식별 및 분석하고 이러한 정보를 사용하여 제품, 프로세스, 서비스 및 조직을 설계함으로써 손실을 줄이는 것과 관련된 일련의 활동이다. 이러한 활동은 안전 컨트롤 스트럭처의 어딘가에 포함될 것이다. 리스크 관리는 위험 예방과 관련된 기술적인 활동으로 구성되는 반면, SMS는 조직적, 경영적, 문화적 안전 측면까지도 포함한다.

STAMP는 더 광범위한 또는 적어도 리스크에 대한 다른 정의를 포함한다. 리스크는 전통적으로 위험이나 사고 발생의 심각성(severity)과 발생 가능성(likelihood)으로 정의되어 왔다. 한편, STAMP에서

> *리스크(Risk)는 안전한 시스템 동작을 강제하기 위해 사용하는 컨트롤의 효과, 즉, 안전 컨트롤 스트럭처의 설계 및 운영의 효과 측면에서 정의된다.*

Note that this definition does not require the determination of likelihood of the events occurring but rather an evaluation of the effectiveness of the controls being used to prevent them.

Creating a risk management system involves designing the procedures to be used in performing the technical risk management activities and assigning responsibility for implementing these procedures to components of the safety control. Risk management may also involve creating leading indicators to identify when risk is increasing.

In fact, an important factor in the effectiveness of the risk management system lies in being able to identify when risk is increasing before a major loss occurs. All safety efforts involve assumptions about how the components of the system will behave and about the environment in which they operate. Violations of these assumptions will undermine the original risk identification and management assumptions. Virtually all accidents have precursors that were unrecognized or for which the responses were non-existent or ineffective. Precursors to accidents are incidents or conditions in which losses do not occur but could have under other circumstances. Leading indicators are characteristics of an organization or an organization's operation that indicate the safety management system is not operating as was assumed when it was designed.

Another way of looking at this is that leading indicators are simply evidence before losses occur that the assumptions under which safety was assured were flawed or are no longer true and that risk may be increasing. As an example, before the Shell Moerdijk chemical plant explosion in the Netherlands, there had been two instances where the same conditions led to explosions in other Shell plants. These explosions were never thoroughly explored or at least the results of the analyses were not used in later design activities, such as the design of the Shell Moerdijk plant where the assumption was made that the conditions were impossible and could not lead to an explosion. Surprisingly, probabilistic risk assessments are often not re-evaluated even after concrete evidence is gathered from actual use of the system that the assumptions made in the analysis are not true. Apparently, the power of a calculated number often trumps evidence that the number is untrue. Generating evidence about the truth of risk assessments and the assumptions that underlie them should be an important part of any safety management system.

While it is unrealistic to expect people to recognize precursors that are only evident in hindsight, too often the precursors could and should have been identified and led to further investigation. *An effective SMS is characterized by continual vigilance against degradation of the safety control structure over time*.

A common paradox is that the fewer accidents an organizational has, the more likely it is that there will be more in the future. If the safety management system is effective, then losses are few. Unfortunately, people tend over time to judge the inherent amount of risk with respect to the current accident rate. If there are few accidents, namely, the safety management system is very effective, over time inherent risk is judged as low and safety management starts to degrade and become less effective. Feedback and information is required to keep the process models of decision makers up-to-date with the actual level of risk at any time.

이 정의에서는 이벤트의 발생 가능성(likelihood)을 결정할 필요가 없으며 오히려 이벤트를 예방하기 위해 사용되는 컨트롤의 효과를 평가할 것을 요구한다.

리스크 관리 시스템을 구축하는 것은 기술적인 리스크 관리 활동을 수행하는 절차를 설계하고 그 절차를 수행하는 책임을 안전 컨트롤의 컴포넌트에 할당하는 것을 포함한다. 리스크 관리에는 리스크가 증가하는 시점을 식별하기 위한 선행지표를 만드는 것도 포함된다.

사실, 리스크 관리 시스템의 효과 중 중요한 것은 중대한 손실이 발생하기 전에 리스크가 증가하는 시점을 식별할 수 있다는 것이다. 모든 안전 노력에는 시스템 컴포넌트가 작동하는 방식과 운영되는 환경에 대한 가정이 포함된다. 이러한 가정의 위반은 원래의 리스크 식별 및 관리에 대한 가정을 약화시킬 것이다. 거의 모든 사고에는 인식되지 않았거나 응답이 없거나 응답이 효과가 없는 전조(前兆, precursor)가 있다. 사고의 전조란 손실이 발생하지는 않았지만 다른 환경에서는 손실이 발생했을 수 있는 사건이나 조건이다. 선행지표란 안전 관리 시스템이 설계 당시 가정된대로 작동하지 않음을 나타내는 조직 또는 조직의 운영 특성이다.

이것을 바라보는 또 다른 방법은 선행지표가 안전을 보장하는 가정에 결함이 있거나 가정이 더 이상 사실이 아니어서 리스크가 증가할 수 있다는 것을 나타내는, 손실 발생 전의 단순한 증거라는 것이다. 예를 들어 네덜란드의 쉘 메르딕(Shell Moerdijk) 화학 공장 폭발 이전에 다른 쉘(Shell) 공장에서 동일한 조건에서 폭발이 발생한 사례가 두 번 있었다. 이 폭발들은 철저히 조사되지 않았으며 조사 결과조차도 이후 설계 활동에 활용되지 않았는데, 예로, 쉘 메르딕 공장을 설계할 때 이전에 일어난 두 폭발의 조건은 일어날 수 없으며 폭발로 이어지지 않을 것이라고 가정했다. 놀랍게도 확률론적인 리스크 평가(probabilistic risk assessment)는 종종 재평가되지(re-evaluated) 않는데, 분석을 통해 만든 가정이 실제 시스템을 사용해 본 결과 사실이 아니라는 명확한 증거가 확보된 경우에도 역시 재평가되지 않는다. 분명, 계산된 숫자의 힘이 그 숫자가 사실이 아니라는 증거보다 더 강력하다. 리스크 평가의 진실에 대한 근거와 그의 기반이 되는 가정을 도출하는 것은 모든 안전 관리 시스템에서 중요한 부분이 되어야 한다.

지나고 나서 볼 때만 확실히 알 수 있는 전조를 사람들이 인지하기를 기대하는 것이 비현실적이기는 하나, 전조들이 식별될 수 있었고 식별되었어야만 하며, 더 많은 조사로 이어졌어야 하는 경우가 너무나 많다. *효과적인 SMS는 시간이 지남에 따라 안전 컨트롤 스트럭처가 저하되는 것을 지속적으로 경계하는 특징이 있다.*

흔한 역설은 사고가 적은 조직일수록 미래에 더 많은 사고가 발생할 가능성이 높다는 것이다. 안전 관리 시스템이 효과적이면 손실은 거의 없다. 불행히도 사람들은 시간이 지남에 따라 현재 사고율과 연관시켜 내재된 리스크의 정도를 판단하는 경향이 있다. 사고가 거의 없으면, 다시 말해, 안전 관리 시스템이 매우 효과적이면 시간이 지남에 따라 내재된 리스크는 낮다고 판단되어 안전 관리는 저하되기 시작하며 효과가 줄어들게 된다. 의사 결정권자의 프로세스 모델을 항상 실제 리스크 수준과 동일한 최신 상태로 유지하려면 피드백과 정보가 필요하다.

Tips for risk management

- Create procedures for identifying hazards, analyzing them, and then using the resulting information in product development and operations. Assign responsibility to ensure this is taking place.
- Emphasize qualitative procedures that provide the information necessary for improvement and not just quantitative assessment. Don't make up quantitative assessments for likelihood that is unknowable.
- Reevaluate risk assessment results as data is obtained and when changes occur over time.
- Provide ways to identify when risk is increasing over what was originally assumed.
- When procedures are not being followed, don't just try to enforce them. Instead evaluate why they are not being followed and redesign them. Try to understand why gaps between what is specified and what is done are occurring.

Education and training

Everyone in the safety control structure, not just the lower-level controllers of the physical systems, must understand their roles and responsibilities with respect to safety and why the system—including the organizational aspects of the safety control structure—was designed the way it was. If employees understand the intent of the SMS and commit to it, they are more likely to comply with that intention rather than simply follow rules when it is convenient to do so. Training is not enough; education is required.

Education must include not only information about the hazards and safety constraints enforced by the controls, but also about priorities and how decisions are to be made. The safety philosophy statement, discussed earlier, provides information about the safety values to be used in decision making. In addition, everyone needs to know the risks they are taking in the decisions they make. Often, poor decision making arises from having an incorrect assessment of the risk being assumed. Using the non-probabilistic definition of risk provided above, this means that the decision makers must know how their decisions will impact the designed controls in the safety control structure.

Telling managers and employees to be "mindful of weak signals" (a common suggestion in the High Reliability Organization or HRO literature) simply creates a pretext for blame after a loss event occurs and hindsight provides the clarity that changes a weak signal (noise) into a strong signal. Instead, everyone must be trained on the hazards associated with the operation of a system and how to recognize them if we expect them to recognize the precursors to an accident. People need to know what to look for, not just be told to look for an undefined something.

Training should also include "why" as well as "what." Understanding the rationale behind the safety rules they are asked to follow will help reduce complacency, what appears to be reckless behavior (but to the person made perfect sense), and unintended changes leading to hazards. The rationale includes understanding why previous accidents occurred and what changes were made to try to prevent a reoccurrence. With increasing automation and people interacting with it, understanding the controlled hazards involved may require more training than was necessary in the past when the hazards were more obvious and intuitive. People who interact with complex systems need to learn more than just the procedures to follow: they must also have an in-depth understanding of the controlled physical process as well as the logic used in any automated controller (software) they may be supervising or with which they may be interacting. A list of what controllers—at all levels of the safety control structure—need to know was included in *Engineering a Safer World* and is repeated here:

리스크 관리를 위한 팁

- 위험을 식별 및 분석하고, 그 결과로 도출된 정보를 제품 개발 및 운영에 사용하기 위한 절차를 만든다. 이것이 수행됨을 보장하도록 책임을 할당한다.
- 정량적(quantitative) 평가뿐만 아니라 개선에 필요한 정보를 제공하는 정성적(qualitative) 절차를 강조한다. 알 수 없는 발생 가능성(likelihood)에 대해서는 정량적 평가를 하지 않는다.
- 데이터가 수집된 후 시간이 지나 변경이 발생할 때 리스크 평가 결과를 재평가한다.
- 리스크가 원래 가정했던 것보다 높아지는 시점을 식별할 방법을 제공한다.
- 절차가 지켜지지 않을 때 단순히 강요하려 하지 않는다. 대신, 왜 지켜지지 않는지 평가하고 재설계한다. 명세한 것과 시행된 것 사이의 갭이 왜 발생했는지 이해하기 위해 노력한다.

교육 및 훈련(Education and training):

물리적 시스템의 하위 수준 컨트롤러뿐만 아니라, 안전 컨트롤 스트럭처에 포함된 모든 사람들은 안전에 관한 자신의 역할과 책임을 이해해야 하고, 왜 시스템이—안전 컨트롤 스트럭처의 조직적 측면 포함—그런 방식으로 설계되었는지 이해해야 한다. 만일 직원이 SMS의 의도를 이해하고 수용한다면, 단순히 편해서 SMS의 규칙을 따를 때보다 의도에 맞게 더 잘 준수할 가능성이 크다. 훈련만으로는 충분하지 않고 교육이 필요하다.

교육에는 컨트롤에 의해 강제되는 안전 제약사항 및 위험의 정보뿐만 아니라 우선순위 및 의사 결정 방법에 대한 정보도 포함되어야 한다. 앞서 논의한 안전 철학 성명서는 의사 결정 시 사용할 안전 가치에 대한 정보를 제공한다. 또한 모든 사람들은 그들이 내리는 의사 결정으로 인해 부담하게 되는 리스크를 알아야 한다. 종종 리스크에 대한 부정확한 평가(assessment)로 인해 잘못된 결정이 내려지는 경우도 있다. 위에서 언급한 리스크의 비확률적(non-probabilistic) 정의를 사용하면 이는 의사 결정권자가 그들의 결정이 안전 컨트롤 스트럭처에 설계된 컨트롤에 어떻게 영향을 미치는지 알아야 한다는 것을 의미한다.

관리자 및 직원에게 "미약한 신호에도 신경을 쓸 것"(고신뢰성 조직(High Reliability Organization) 또는 HRO 문서의 일반적인 의견)이라고 말하는 것은 손실 발생 후에 비난하기 위한 구실을 만들 뿐이며 나중에는 미약한 신호(소음)를 강한 신호로 바꿔야 했었다는 생각을 갖게 만든다. 대신, 모든 사람이 시스템 운영 시의 위험에 대해 훈련받아야 하며, 만일 사고의 전조를 인식하기 위해 위험을 예측할 수 있다면 위험을 인식하는 방법을 훈련해야 한다. 사람들은 무엇을 찾아야 하는지를 알고 있어야 하며 사람들에게 정의되지 않은 무언가를 찾으라고 해서는 안 된다.

훈련에는 "무엇" 뿐만 아니라 "왜"가 포함되어야 한다. 준수해야 할 안전 규칙의 이론적 근거를 이해하는 것은 안주(安住)하는 것, 무모한 행동으로 보이는 것(그러나 그 사람에게는 완전히 이해되는 것일 수 있음), 그리고 위험에 이르는 의도치 않는 변경을 줄이는 데 도움이 된다. 이론적 근거에는 과거의 사고가 발생한 이유 및 재발을 막기 위해 만들어진 변경을 이해하는 것이 포함된다. 자동화의 증가와 자동화와 사람들 간 상호작용의 증가로, 이와 관련된 제어된 위험(controlled hazard)을 이해하는 것은 위험이 더 분명하고 직관적이었던 과거에 비해 많은 훈련을 필요로 할 수 있다. 복잡한 시스템과 상호작용하는 사람들은 따라야 할 절차 이상의 것을 배워야 한다: 그들이 감독하거나 상호작용하는 자동화 컨트롤러(소프트웨어)의 로직뿐만 아니라 제어되는 물리적인 프로세스에 대해서도 깊이 이해해야 한다. 컨트롤러—안전 컨트롤 스트럭처의 모든 수준에 있는 컨트롤러—가 알아야 하는 것의 목록이 '*Engineering a Safer World*'에 소개되어 있으며, 여기에도 다시 소개한다:

- The system hazards and the reasons behind safety-critical procedures and operational rules.
- The potential result of removing or overriding controls, changing prescribed procedures; and inattention to safety-critical features and operations: Past accidents and their causes should be reviewed and understood.
- How to interpret feedback: Training needs to include different combinations of alerts and sequences of events, not just single events.
- How to think flexibly when solving problems: Controllers need to be provided with the opportunity to practice problem solving involving safety.
- General strategies rather than specific responses: Controllers need to develop skills for dealing with unanticipated events.
- How to test hypotheses in an appropriate way: To update mental models, human controllers often use hypothesis testing to understand the current system states and to update their process models. Such hypothesis testing is common with computers and automated systems where documentation is usually so poor and hard to use that experimentation is often the only way to understand the automation design and behavior. Hypothesis testing, however, can lead to losses. Designers need to provide operators with the ability to test hypotheses safely and controllers must be educated on how to do so.
- Emergency procedures must be overlearned and continually practiced. Controllers need to be taught about operating limits and specific actions to take in case they are exceeded. Requiring operators to make good decisions under stress and without full information simply ensures they will be blamed after a loss occurs.

Training should not be a one-time event for employees but should be continual throughout their employment, if only as a reminder of their responsibilities and the system hazards. Learning about recent events and trends should be part of this training. Assessing for the effectiveness of the training, perhaps through regular audits and performance assessments, can be useful in implementing an effective improvement and learning process. Incident and accident investigation results are an important source of information about training effectiveness.

Some companies have created a practice where managers provide safety training. Training experts help manage group dynamics and curriculum development, but the training is provided by project or group leaders. By learning to teach the materials, supervisors and managers are more likely to absorb and practice the key principles. In addition, it has been found that employees pay more attention to a message delivered by their boss than by a trainer or safety expert.

- 시스템 위험과 안전-필수 절차 및 운영 규칙의 이유
- 컨트롤을 제거하거나 무시함(overriding)에 따른 잠재적 결과, 규정된 절차를 변경한 잠재적 결과; 안전-필수 기능 및 운영에 대한 부주의: 과거의 사고와 그 원인을 검토하고 이해해야 한다.
- 피드백을 해석하는 방법: 훈련에는 단일 이벤트뿐만 아니라 경보(alert) 및 이벤트 순서의 다양한 조합이 포함되어야 한다.
- 문제를 해결할 때 유연하게 생각하는 방법: 안전과 관련된 문제 해결을 연습할 수 있는 기회를 컨트롤러에게 주어야 한다.
- 특정한 대응보다는 일반적인 전략: 컨트롤러는 예기치 않은 이벤트를 다루기 위한 기술을 개발해야 한다.
- 가설을 테스트하는 적절한 방법 : 멘탈 모델을 업데이트하기 위해 사람(휴먼 컨트롤러)은 종종 가설 테스트(hypothesis testing)를 사용하여 현재 시스템 상태를 이해하고 그들의 프로세스 모델을 업데이트 한다. 이러한 가설 테스트는 문서가 매우 빈약하고 사용하기 어려워서 실험(experimentation)만이 자동화 설계와 동작을 이해하는 유일한 방법인 컴퓨터 및 자동화 시스템에서 일반적이다. 그러나 가설 테스트는 손실로 이어질 수 있다. 설계자는 운영자에게 가설을 안전하게 테스트할 수 있는 능력을 제공해야 하며 컨트롤러는 그렇게 하는 방법을 교육받아야 한다.
- 비상 절차(emergency procedure)는 숙달된 후에도 계속 실습해야 하고 지속적으로 실시되어야 한다. 컨트롤러는 운영 한계와 한계를 초과할 때 취해야 할 특정 조치에 대해 배워야 한다. 운영자가 완전한 정보 없이 스트레스 하에 올바른 결정을 내리도록 요구하는 것은 손실이 발생한 후에 단순히 책임을 전가하는 것과 같다.

훈련은 직원을 위한 일회성 이벤트가 되어서는 안 되며 그들의 책임과 시스템 위험을 상기시키기 위한 용도로만 직무수행 전체에 걸쳐 지속적으로 이루어져야 한다. 최근의 이벤트와 트렌드를 학습하는 것도 이 훈련의 한 부분이 되어야 한다. 정기적인 감사와 성과 평가(assessments)를 통해 훈련의 효과를 평가하는 것은 효과적인 개선방안 및 학습 프로세스를 시행하는 데 유용할 수 있다. 사건과 사고의 조사 결과는 훈련의 효과에 대한 중요한 정보 출처이다.

일부 기업에서는 관리자가 안전 훈련을 직접 실시한다. 훈련 전문가가 집단 역학(group dynamics) 및 커리큘럼 개발을 관리하는 데 도움을 주기는 하나 훈련을 제공하는 사람은 프로젝트나 그룹의 리더이다. 이 경우 감독자와 관리자가 자료(materials)를 가르치는 것을 목적으로 하여 배움으로서 핵심 원칙을 흡수하고 실천할 가능성이 높아진다. 또한, 직원들은 트레이너나 안전 전문가 보다 상사가 전달한 메시지에 더 많이 집중하는 것으로 나타났다.

Tips for education and training

- Educate, don't just train
- Everyone should understand their roles and responsibilities, why the system was designed the way it was, information about hazards and safety constraints enforced by the controls, and the risks they are taking in the decisions they make.
- Training should include "why" and not just "what."
- Everyone should learn:
  - The system hazards and the reasons behind safety-critical procedures and operational rules that are related to their jobs.
  - The potential result of removing or overriding controls, changing prescribed procedures, and inattention to safety-critical features and operations.
- Make sure the causes of past accidents are well disseminated and understood.
- Give people the opportunity to practice problem solving involving safety
- Teach general strategies rather than specific responses so controllers can develop skills for dealing with unanticipated events.
- Teach operators how to test hypotheses in an appropriate way and provide ways for them to learn in this way.
- Emergency procedures should be overlearned and continually practiced.
- Contingency procedures should be reviewed before any safety-critical activity.
- Training should be continuous and not just a one-time event when hired.
- Teach about recent events and trends
- Consider having managers provide safety training.

Learning and continual improvement:

Because SMS designs are rarely perfect from the beginning and the world changes over time, there must be an effective process in place to ensure that the organization is continually learning from safety incidents and improving the SMS itself as well as the workplace, the products and the services.

Experimentation is an important part of the learning process, and trying new ideas and approaches to improving safety should be encouraged but also evaluated carefully to ensure that improvement actually results.

The Safety Information System (SIS)

A comprehensive and usable safety information system is key to having a successful SMS. It provides a source of information about the effectiveness of the control structure. In STAMP terms, this is the definition of risk.

The process models of controllers and decision makers must be kept accurate and coordinated. In essence, the SIS acts as a shared process model and a source for updating individual process models. Therefore, accurate and timely feedback and data are important. The SIS can provide the information necessary to detect trends, changes and other precursors to an accident; to evaluate the effectiveness of the safety controls; to compare models and risk assessments with actual behavior; and to learn from events and improve the SMS. After major accidents, it is often found that the information to prevent the loss existed but was not used or was not available to those involved. In general, lots of information is collected only because it is required for government reports and not necessarily because it is a necessity for the operation of an effective SMS.

교육 및 훈련을 위한 팁

- 훈련만 하지 말고 교육을 동반한다.
- 모든 사람이 자신의 역할과 책임을 이해해야 하고, 시스템이 왜 현재의 방식으로 설계되었는지, 위험 및 컨트롤에 의해 강제된 안전 제약사항의 정보, 그리고 그들이 내린 결정으로 인해 감수해야 하는 리스크를 이해해야 한다.
- 훈련에는 단순히 "무엇"이 아니라 "왜"가 포함되어야 한다.
- 누구나 배워야 할 사항 :
  - 시스템 위험, 안전-필수 절차 및 직무와 관련된 운영 규칙의 이유
  - 컨트롤의 제거 또는 무시로 인한 잠재적 결과, 규정된 절차의 변경으로 인한 잠재적 결과, 그리고 안전-필수 기능 및 운영에 대한 부주의
- 과거의 사고 원인이 잘 전파되고 이해되고 있는지 확인한다.
- 사람들에게 안전과 관련된 문제 해결을 연습할 기회를 제공한다.
- 특정한 대응보다는 일반적인 전략을 가르쳐 컨트롤러가 예기치 않은 이벤트를 처리하는 기술을 개발할 수 있도록 한다.
- 가설을 테스트하는 적절한 방법에 대하여 운영자를 교육시키고, 이러한 방식으로 학습할 수 있는 방법을 제공한다.
- 비상 절차(emergency procedure)는 숙달된 이후에도 학습하고 지속적으로 연습해야 한다.
- 모든 안전-필수 활동을 하기 전에 우발상황 절차(contingency procedures)를 검토해야 한다.
- 훈련은 고용 시의 일회성 이벤트가 아니라 지속적으로 이루어져야 한다.
- 최근의 이벤트 및 트렌드에 대해 가르친다.
- 관리자가 안전 훈련을 제공하는 것을 고려한다.

학습 및 지속적인 개선(Learning and continual improvement):

SMS 설계는 처음부터 완벽할 수 없으며 시간이 지남에 따라 세상이 변하기 때문에 조직이 안전 사건으로부터 지속적으로 학습하고, 작업장·제품·서비스뿐만 아니라 SMS 자체를 개선하기 위해 효과적인 프로세스가 마련되어야 한다.

실험(Experimentation)은 학습 과정의 중요한 부분이다. 안전을 개선하기 위한 새로운 아이디어와 접근 방법의 시도를 권장해야 하지만, 동시에, 개선이 실제로 이루어지도록 신중하게 평가해야 한다.

안전 정보 시스템(SIS, the Safety Information System)

포괄적이고 유용한 안전 정보 시스템(safety information system)은 SMS를 성공적으로 만드는 열쇠이다. SIS는 컨트롤 스트럭처의 효과에 대한 정보 출처를 제공한다. STAMP 용어에서는 컨트롤 스트럭처의 효과가 리스크의 정의이다.

컨트롤러 및 의사 결정권자의 프로세스 모델은 정확하고 조정된(coordinated) 상태로 유지되어야 한다. 본질적으로 SIS는 공유된 프로세스 모델 역할을 하고 개별 프로세스 모델을 업데이트하기 위한 출처 역할을 한다. 따라서 정확하고 시기적절한 피드백과 데이터가 중요하다. SIS는 트렌드, 변경사항, 그리고 사고의 전조를 탐지하기 위해 필요한 정보; 안전 컨트롤의 효과를 평가하기 위해 필요한 정보;

To determine what should be in the SIS, the responsibilities of those in the safety control structure and the feedback they need to fulfill those responsibilities can be used, that is, the information required in their process/mental models in order to make appropriate decisions. In general, the SIS contains, at a minimum:

- The safety management plan (for both development and operations),
- The status of all safety-related activities for the system,
- The safety constraints and assumptions underlying the design, including operational limitations,
- The results of the hazard analyses (hazard logs) along with tracking and status information on all known hazards,
- The results of performance audits and assessments,
- Incident and accident investigation reports and corrective actions taken,
- Lessons learned and historical information, and
- Results of trend analysis.

Using the safety control structure design to identify what each person in it requires for safe decision making will help to design a usable SIS where each person is able to find the information they need.

Information may be collected by companies or by industries. No matter how the information is collected, understanding its limitations is important. To be most useful, the information must be accurate and timely and it must be disseminated to the appropriate people in a useful form. Three factors are involved: collection, analysis, and dissemination.

Collection: Data may be distorted by the way it is collected. Common problems are systematic filtering, suppression, and unreliability. Data collected for accident reports tend to focus on proximal events and actors but not the systemic factors involved such as management problems or organizational deficiencies. Studies[35] have shown that data from near-miss reporting by operators are filtered and primarily point to operator error as the cause of the events. Reports by management are similarly filtered and also point to operator error as the cause. Even general safety inspections tend to identify limited categories of conditions, especially when checklists are used.

Data collection tends to be more reliable for accidents that are similar to those that have occurred in the past than for new types of systems where past experience on hazards and causal factors is more limited. Software errors and computer problems are often omitted or inadequately described because of lack of knowledge, lack of accepted and consistent categorizations for such errors, or simply not considering them seriously as a causal factor.

Experience has shown that it is difficult to maintain reliable reporting of incidents over an extended period, and therefore this type of data is not very useful for estimating probabilities of events or their potential consequences. There is data to suggest that in some industries, up to three quarters of accidents may be unreported. Sometimes, information is involved that a company wants to keep secret or that those making the reports may worry will be used against them in job performance evaluations. Potential legal liability may also be involved.

Discussions about operational problems that have occurred should be documented in the SIS and analyzed for their hazard potential. While it is common, as previously discussed, for concerns to be brought directly to those who can address them, bypassing documentation in the SIS, very often these discussions or emails may contain safety issues that are not recognized as such by the participants. In addition, documenting these discussions can be beneficial as they contain valuable corporate knowledge that would otherwise be lost as people leave positions or retire or due to email retention standards. A

[35] Many scientific references for this section can be found in Nancy G. Leveson, *Safeware*, Addison Wesley Publishers, 1995, Chapter 11. We tried to keep this handbook from reading like an academic paper.

모델과 리스크 평가를 실제 행동과 비교하기 위해 필요한 정보; 이벤트로부터 배우고 SMS를 개선하기 위해 필요한 정보를 제공한다. 중대한 사고가 발생한 후, 손실을 방지하기 위한 정보가 있었지만 사용되지 않았거나 관계자가 사용할 수 없었던 경우가 종종 발견된다. 일반적으로 정부 보고서에 필요하기 때문에 많은 정보가 수집되는 것이지, 효과적인 SMS 운영을 위해 필요한 것은 아니다.

SIS에 무엇이 있어야 하는지 결정하기 위해, 안전 컨트롤 스트럭처에 있는 사람들의 책임과 그러한 책임을 수행하는 데 필요한 피드백을 활용할 수 있다. 즉, 적절한 결정을 내리기 위해 그들의 프로세스/멘탈 모델에 요구되는 정보를 활용할 수 있다. 일반적으로 SIS는 최소한 아래의 내용을 포함한다 :

- (개발 및 운영 모두에 대한) 안전 관리 계획
- 시스템에 대한 모든 안전-관련 활동의 상태
- 운영 한계를 포함하는, 설계의 기초가 되는 안전 제약사항 및 가정
- 모든 알려진 위험에 대한 추적 및 상태 정보가 포함된 위험 분석 결과(위험 로그)
- 성과 감사 및 평가의 결과
- 사건 및 사고 조사 보고서 및 취해진 시정 조치
- 교훈과 과거의 정보
- 트렌드 분석 결과

사람들이 안전 의사 결정에 필요한 것을 식별하기 위해 안전 컨트롤 스트럭처를 사용하는 것은, 필요한 정보를 찾을 수 있는 유용한 SIS를 설계 하는 데 도움이 될 것이다.

정보는 기업별 또는 산업별로 수집될 수 있다. 정보가 수집되는 방법에 관계없이 그 한계를 이해하는 것이 중요하다. 정보가 가장 유용하려면, 정보는 정확하고 시기적절해야 하며 유용한 형태로 적절한 사람들에게 보급되어야 한다. 여기에는 수집, 분석 및 보급이라는 세 가지 요소가 관련된다.

수집(collection): 데이터는 수집되는 방식에 따라 왜곡될 수 있다. 흔한 문제는 시스템적 필터링, 은폐(suppression), 신뢰성 부재(unreliability)이다. 사고 보고서를 위해 수집된 데이터는 중점적인 이벤트와 사고 관련자에 집중하는 경향이 있으며, 관리 문제나 조직적 결함과 같은 시스템적인 요인들에는 초점을 두지 않는 경향이 있다. 연구[35] 결과에 따르면 운영자에 의한 니어 미스(near-miss)의 보고 데이터는 필터링되고 대신, 이벤트 원인으로 운영자 오류를 주로 지적한다. 마찬가지로 관리자의 보고서에서도 그러한 데이터는 필터링 되며 원인으로 역시 운영자 오류를 지적한다. 일반적인 안전 검사(inspections)에서조차도 조건의 유형을 한정하여 식별하는 경향이 있는데, 특히 체크리스트가 사용될 때 그렇다.

위험과 원인 요소에 대한 과거 경험이 부족한 새로운 유형의 시스템보다는 과거에 발생한 사고와 유사한 사고에 대해 데이터를 수집할 때 그 신뢰성이 높아지는 경향이 있다. 소프트웨어 오류와 컴퓨터의 문제는 지식의 부족, 오류에 대한 인정되고 일관된 분류 체계의 부족, 또는 단순히 오류를 심각한 원인 요소로 고려하지 않아, 종종 생략되거나 부적절하게 설명된다.

경험에 따르면 오랜 기간에 걸쳐 사건을 신뢰성 있게 보고하는 것이 어렵기 때문에, 오랜 기간에 걸친 축적된 데이터는 이벤트의 발생확률(probabilities)이나 그 잠재적 결과를 추정하는 데 별로 유용하지 않다. 일부 산업에서는 최대 4분의 3까지의 사고가 보고되지 않을 수도 있다는 자료가 있다. 때로는 회사가 비밀로 하고 싶은 정보 또는 보고서를 작성하는 사람이 업무 성과 평가에 사용되는 것을 우려

[35] 이 섹션에 대한 많은 과학적 참조자료는 Nancy G. Leveson의 Safeware(Addison Wesley 출판사, 1995)의 11장을 참고한다. 우리는 이 핸드북이 학술 논문처럼 읽혀지지 않도록 노력했다.

requirement to retain and document these communications should be implemented or a corporate communication channel that automatically retains such information should be developed.

Some measures can be used to improve the comprehensiveness and reliability of data collection such as special training of data collectors, feedback of results to the data collectors, fixed routines, and anonymous reporting. Training on techniques such as CAST can encourage the inclusion of information that is often omitted. Automated monitoring systems are now commonplace, but a problem is that they can and usually do collect so much information that it is difficult to analyze the results and detect problems. STPA might be a useful way to help determine what types of information need to be collected and to identify leading indicators in the large amounts of data collected.

Analysis: Systematizing and consolidating a large mass of data into a form useful for learning is difficult. Raw quantitative data can be misleading, especially if statistical significance is not included: Data is not the same as information. Again, STPA results can be used not only to identify what data needs to be collected, but to provide guidance on the importance of the events that are occurring. In addition, data to validate the STPA causal analysis results and to identify factors that were thought to be eliminated or mitigated should be part of the SIS collection and analysis process.

The biggest problem in analysis is simply that while it is easy to design data collection channels, finding the time and manpower to analyze all the data that results may be difficult or impractical. Tools like STPA can help to identify the most important data to collect and the most useful analysis processes.

Dissemination: Disseminating information (not data) in a useful form may be the most difficult part of an SIS. Information needs to be presented in a form that people can learn from, apply to their daily jobs, and use throughout the life cycle of projects, not just in the conceptual design stage. And it must be updated in a timely manner: accidents have resulted from changes during operations or due to insufficient updates to the hazard analysis when engineering modifications are made. STPA stresses usable documentation that includes rationale and intent as well as traceability to assist in the change process. Also, the assigned safety-related responsibilities to individuals and groups in the safety control structure will provide the information necessary to tailor information to the needs of those that receive it. Providing too much information to individuals can be as dangerous as providing too little.

The method for presenting information should be adaptable to the cognitive styles of the users and should be integrated into the environment in which safety-related decisions are made. Again, the safety control structure design process will provide a great deal of useful input about what information is needed, when it is needed, and how it will be used by individual controllers in the control structure.

하는 정보가 있을 수 있다. 잠재적인 법적 책임에 관련된 정보도 있을 수 있다.

운영상 발생한 문제에 대한 논의는 SIS에 문서화되어야 하며 위험의 가능성(potential)이 분석되어야 한다. 앞서 논의한 바와 같이 우려가 SIS에 문서화되지 않고 그것을 해결할 수 있는 사람에게 직접 전달되는 경우가 흔한데, 이 과정의 논의나 이메일에 참여자들이 인식하지 못하는 안전 이슈가 포함되는 경우가 빈번하다. 또한 이러한 논의에는 사람들이 부서를 이동하거나 은퇴하거나 또는 기본 이메일 저장기간으로 인해 없어질 수 있는, 소중한 기업의 지식을 포함하고 있으므로 논의를 문서화하는 것이 유익하다. 커뮤니케이션을 확보하고 문서화하기 위한 요구사항을 만들어야 하며 그러한 정보를 자동으로 확보하는 기업의 커뮤니케이션 채널이 개발되어야 한다.

데이터 수집의 완전성(comprehensiveness)과 신뢰성을 향상시키기 위해 데이터 수집가(collector)에 대한 특별한 훈련, 데이터 수집가에 대한 결과 피드백, 고정적인 루틴 및 익명 보고 등과 같은 일부 방법들을 사용할 수 있다. 종종 누락되는 정보를 CAST와 같은 기법의 교육을 통해 누락되지 않게 할 수 있다. 자동화된 모니터링 시스템이 흔히 사용되고 있지만 문제는 모니터링 시스템이 매우 많은 정보를 수집할 수 있고 또 일반적으로 수집하기 때문에 결과를 분석하고 문제를 발견하는 것이 오히려 어렵다는 점이다. STPA는 수집해야 할 정보의 유형을 결정하고 수집된 대량의 데이터에서 선행지표를 식별하는 데 유용할 것이다.

분석(Analysis): 대량의 데이터를 학습에 유용한 형태로 체계화하고 통합하는(consolidating) 것은 어렵다. 가공되지 않은 정량적 데이터(raw quantitative data)는, 특히 통계적 유의도가 포함되지 않은 경우에는 잘못 해석될 수 있다: 데이터는 정보와 다르다. 다시 말해, STPA의 결과는 수집해야 할 데이터를 식별하는 데 사용할 수 있을 뿐만 아니라 발생하는 이벤트의 중요성에 대한 지침을 제공하는 데에도 사용할 수 있다. 추가적으로, STPA 원인 분석 결과를 확인(validate)하기 위한 데이터와, 제거되거나 완화될 것으로 여겨진 요인을 식별하기 위한 데이터도 SIS에서 수집 및 분석되어야 한다.

분석의 가장 큰 문제는 단순히는, 데이터 수집 채널을 설계하는 것은 쉽지만 모든 결과 데이터를 분석하는 데 필요한 시간과 인력을 찾는 것이 어렵고 비현실적이라는 것이다. STPA와 같은 도구는 수집해야 하는 가장 중요한 데이터와 가장 유용한 분석 프로세스를 식별하는 데 도움을 줄 수 있다.

보급(Dissemination): 정보(데이터가 아닌)를 유용한 형태로 보급하는 것이 SIS의 가장 어려운 부분일 수 있다. 정보는 사람들이 배우고 일상 업무에 적용할 수 있으며 개념 설계 단계에서뿐만 아니라 프로젝트의 수명주기 전반에 걸쳐 사용할 수 있는 형태로 존재해야 한다. 또한 적시에 업데이트되어야 한다: 운영 중 변경으로 인해 사고가 발생하거나 엔지니어링 수정이 발생했을 때 위험 분석에 대한 업데이트가 불충분하여 사고가 발생해 왔다. STPA에서는 사용가능한 문서를 강조하며, 이 문서에는 변경 프로세스를 지원할 수 있도록 추적성뿐만 아니라 근거와 의도가 포함되어야 한다. 또한 안전 컨트롤 스트럭처에서 개인과 그룹에 할당된 안전-관련 책임은 정보를 받는 사람들의 요구에 맞게 테일러링하는 데 필요한 정보를 제공한다. 개인에게 너무 많은 정보를 제공하는 것은 너무 적은 정보를 제공하는 것만큼 위험할 수 있다.

정보를 나타내는 방법은 사용자의 인지(cognitive) 스타일에 따라 조정 가능해야 하며 안전-관련 결정이 이루어지는 환경에 통합되어야 한다. 또한, 안전 컨트롤 스트럭처의 설계 프로세스는 어떤 정보가 필요한지, 언제 필요한지, 그리고 개별 컨트롤러가 이를 어떻게 사용할 것인지에 대한 많은 유용한 정보를 제공할 것이다.

Tips for designing a safety information system

- Accurate and timely feedback and data are important
- The SIS should provide the information necessary to detect trends, changes and other precursors to an accident; to evaluate the effectiveness of the safety controls; to compare models and risk assessments with actual behavior; and to learn from events and improve the SMS.
- Use the defined responsibilities of those in the safety control structure to identify the information they need to keep their process models accurate enough for good decision making.
- Understand the limitations of your collected data.
- To be most useful, information must be accurate and timely and it must be disseminated to the appropriate people in a useful form.
- Find ways to collect data that minimize distortion of the data (filtering, suppression, and unreliability).
- Keep detailed information on actual safety-related incidents and ensure that information gets to the people who need it.
- Create ways to improve the comprehension and reliability of data collection.
- Try to include statistical significance on numeric data if possible.
- Use STPA to identify what data needs to be collected and provide guidance on the importance of the events that are occurring.
- Collect data to validate the STPA causal analysis results and to identify factors that were thought to be eliminated or mitigated.
- Ensure that data is analyzed and not just collected.
- Present data in a way that people can learn from it and apply it to their daily jobs.
- Keep data up to date.
- Document design rationale and intent and provide traceability to assist in the change process.
- Tailor the information provided and the presentation format to the needs of those receiving it.
- Providing too much data, particularly raw data, can be as dangerous as providing too little.
- Adapt the presentation of information to the cognitive styles of the users and integrate it into the environment in which safety-related decisions are made.
- Use the safety control design process to determine what information is needed, when it is needed, and how it will be used.

Summary

In general, effective safety management requires:

- Commitment and leadership at all levels
- A strong corporate safety culture
- A clearly articulated safety vision, values and procedures, shared among stakeholders
- A safety control structure with appropriate assignment of responsibility, authority, and accountability.

안전 정보 시스템(SIS) 설계를 위한 팁

- 정확하고 시기적절한 피드백과 데이터가 중요하다.
- SIS는 트렌드, 변경 및 사고의 전조 탐지를 위해 필요한 정보; 안전 컨트롤의 효과를 평가하기 위해 필요한 정보; 모델과 리스크 평가를 실제 행동과 비교하기 위해 필요한 정보; 이벤트를 통해 배우고 SMS를 개선시키기 위해 필요한 정보를 제공해야 한다.
- 안전 컨트롤 스트럭처에 정의된 책임을 사용하여, 올바른 의사 결정을 위해 프로세스 모델을 충분히 정확하게 유지하는 데 필요한 정보를 식별한다.
- 올바른 의사 결정을 내리기 위한 프로세스 모델을 정확히 유지하는 데 필요한 정보를 식별하기 위해 안전 컨트롤 스트럭에 정의된 책임을 사용한다.
- 수집된 데이터의 한계를 이해한다.
- 정보가 가장 유용하기 위해서는 정보는 정확하고 시기적절해야 하며 유용한 형태로 적절한 사람들에게 보급되어야 한다.
- 데이터의 왜곡(필터링, 은폐, 신뢰성 부재)을 최소화하여 수집하는 방법을 찾는다.
- 실제 안전-관련 사건에 대한 상세한 정보를 확보하고 정보가 필요한 사람들에게 확실하게 전달한다.
- 데이터 수집의 완전성과 신뢰성을 향상시키는 방법을 만든다.
- 숫자 데이터는 가능하면 통계적 유의도를 포함하도록 한다.
- 수집해야 할 데이터를 식별하고 발생하는 이벤트의 중요성에 대한 지침을 제공하기 위해 STPA를 사용한다.
- STPA 원인 분석 결과를 확인(validate)하기 위한 데이터 및 제거되거나 완화되어야 할 요인을 식별하기 위한 데이터를 수집한다.
- 데이터는 수집만 하는 것이 아니라 분석해야 한다.
- 사람들이 배울 수 있고 일상 업무에 적용할 수 있는 방식으로 데이터를 제시한다.
- 데이터를 최신 상태로 유지한다.
- 설계의 근거와 의도를 문서화하고 변경 프로세스를 지원하기 위한 추적성을 제공한다.
- 제공된 정보를 테일러링하고 정보를 받는 사람들의 필요에 맞게 표현 형식도 테일러링 한다.
- 너무 많은 데이터, 특히 가공되지 않은(raw) 데이터를 너무 많이 제공하는 것은 너무 적은 데이터를 제공하는 것만큼 위험할 수 있다.
- 사용자의 인지 스타일에 따라 정보 표현을 조정하고 안전-관련 결정이 이루어지는 환경에 정보를 통합한다.
- 안전 컨트롤 설계 프로세스를 사용하여 어떤 정보가 필요한지, 언제 필요하고 어떻게 사용할 것인지를 결정한다.

- Feedback channels that provide an accurate view of the state of safety at all levels of the safety control structure
- Integration of safety into development and line operations (not just a separate and independent group or a separate subculture)
- Individuals with appropriate knowledge, skills, and ability
- Stakeholders with partnership roles and responsibilities
- A designated process for resolving tensions between safety priorities and other priorities
- Risk awareness and communication channels for disseminating safety information
- Controls on system migration toward higher risk
- An effective and usable safety information system
- Continual improvement and learning
- Education, training, and capability development

For more information, the reader is referred to Chapter 13 of *Engineering a Safer World* and to Chapters 11 and 12 in *Safeware*. Appendix D of this handbook also contains a list of the responsibilities that need to be included in an effective Safety Control Structure.

요약

일반적으로, 효과적인 안전 관리에는 다음과 같은 것들이 필요하다:

- 모든 수준(level)에서의 약속과 리더십
- 강력한 기업 안전 문화
- 이해관계자 간에 공유되는 명확하게 표현된 안전 비전, 가치 및 절차
- 책임, 통제권한 및 의무를 적절하게 할당한 안전 컨트롤 스트럭처
- 안전 컨트롤 스트럭처의 모든 수준에서 안전 상태를 정확하게 볼 수 있는 피드백 채널
- 안전을 개발 및 라인 운영에 통합(별도의 독립적인 그룹이나 별도의 하위 문화가 아닌)
- 적절한 지식, 기술 및 능력을 갖춘 개개인
- 파트너십 역할과 책임을 갖춘 이해관계자
- 안전 우선순위와 다른 우선순위 간의 갈등(tension) 해소를 위해 지정된 프로세스
- 리스크 인식 및 안전 정보 보급을 위한 의사소통 채널
- 시스템의 리스크가 높은 상태로 진입하는 것을 제어
- 효과적이고 사용가능한 안전 정보 시스템
- 지속적인 개선 및 학습
- 교육, 훈련 및 역량 개발

자세한 내용은 '*Engineering a Safer World*'의 13장과 '*Safeware*'의 11장 및 12장을 참조한다. 이 핸드북의 부록 D에는 효과적인 안전 컨트롤 스트럭처에 포함되어야 할 책임에 대한 목록이 소개되어 있다.

Chapter 8: Introducing STPA into Your Organization

John Thomas

Nancy Leveson

For relatively small organizations, introducing STPA may simply involve training and perhaps demonstration projects to establish that the practicality and advantages of using STPA are worthwhile. For larger organizations, other challenges may be involved such as changing the culture, changing the standard processes used, and training large numbers of people to do something new and different than they have in the past. This chapter examines some issues that arise when an organization decides to evaluate or adopt STPA for widespread use throughout the organization.

Training

Learning STPA requires some practice and a "learning by doing" approach. Formal, in-person training is very effective, and we've found that professionals can be trained and ready to begin trying STPA in just a few days. The most effective STPA training is interactive with hands-on exercises used to reinforce the process. Surprisingly, people who have been using traditional hazard analysis techniques may have the most difficulty in learning STPA—there seems to be a fair amount of unlearning required and extra training may be necessary. System engineers tend to learn the fastest in our experience.

There are tradeoffs that depend on the size of the training class. Small classes (under 10 people) can provide an in-depth and detailed understanding of STPA and can be a good option for small organizations that want to begin a small pilot study to evaluate STPA. Large classes (over 50 people) take longer and limit the amount of interaction that is possible, but can be a good option for organizations that need to provide a basic awareness of STPA to a large group. Medium classes (around 20-30 people) tend to offer a reasonable compromise.

For very large organizations in which hundreds or even thousands of people need to be trained, a "train the trainers" approach may be desirable. The most effective way to produce STPA experts who can serve as future trainers and facilitators (see below) is to immerse candidates in real projects where STPA is actively used. Attending a short training class is not enough to produce STPA experts, but those who have been immersed in a few large STPA projects are great candidates for future trainers and facilitators. One approach that has worked well is to allow one or more facilitators-in-training to shadow other STPA facilitators working on real projects. The facilitators in training learn a great deal by seeing first-hand the challenges encountered in different projects and the questions that are raised. Once they demonstrate the ability to provide effective guidance, solve challenges, and answer detailed STPA questions from other participants, they may be ready to facilitate on their own.

This handbook should provide more training options that were not possible before it was available. In addition, we are producing videos to go with the handbook.

Facilitators

STPA is most useful when integrated into the design effort and not performed as an independent effort by a separate group. However, not everyone will want or need to be trained as an STPA expert. The idea of a facilitator is used in HAZOP for the same reasons we encourage it for STPA, that is, experts on the system design make the best STPA team participants but may need guidance in actually applying the technique if they are not familiar with it.

8장: 조직에 STPA 도입

상대적으로 규모가 작은 조직에서 STPA를 도입하기 위해서는 훈련이나 STPA 실용성 및 장점을 보여주는 시범 프로젝트를 간단히 활용해 볼 수 있다. 대규모 조직의 경우에는 문화의 변화, 사용하는 표준 프로세스의 변경, 과거와는 다른 새로운 것을 하도록 많은 사람을 훈련하는 것과 같은 또 다른 도전과제가 있을 수 있다. 이 장에서는 STPA를 광범위하게 사용하기 위해 조직이 STPA를 평가하거나 채택하기로 결정할 때 발생하는 몇 가지 이슈를 검토해본다.

훈련(Training)

STPA를 배우려면 약간의 연습과 "실습하며 배우는(learning by doing)" 접근 방법이 필요하다. 정형화된 강의실 훈련(in-person training)이 매우 효과적인데 우리가 확인한 바로는 전문가의 경우에는 훈련을 받고 불과 며칠 만에 STPA 적용을 시작할 수 있었다. 가장 효과적인 STPA 훈련 방식은 상호소통하며 직접 손으로 연습하여 절차를 확실히 익히는 것이다. 놀랍게도 전통적인 위험 분석 기법을 사용해 온 사람들은 STPA를 배우는 데 가장 어려움을 겪을 수 있다 — 많은 양의 지식을 고의적으로 잊어야 할 수도 있으며 추가 훈련을 해야 할 수도 있다. 경험상 시스템 엔지니어가 가장 빠르게 학습하는 경향을 보였다.

훈련하는 수업(class)의 규모에 따라 장·단점이 존재한다. 소규모 수업(10명 미만)은 STPA에 대한 심층적이고 상세한 이해를 제공할 수 있어, STPA를 평가하기 위해 작은 규모의 파일럿 연구를 하고자하는 작은 조직에 좋은 선택지가 될 수 있다. 대규모 수업(50명 이상)은 오래 걸리고 상호 소통하는 양에 제한이 있지만 기본적인 STPA 지식을 많은 사람에게 제공하고자 하는 조직에게는 좋은 선택지가 될 수 있다. 중간 규모 수업(약 20-30명 정도)은 합리적인 절충안이 될 수 있다.

수백 또는 수천 명의 사람들을 훈련해야 하는 매우 큰 조직의 경우 "강사 훈련(train the trainers)" 접근 방식이 바람직할 수 있다. 미래의 강사나 퍼실리테이터(아래 참조)로 활동할 STPA 전문가를 양성하는 가장 효과적인 방법은 후보자를 STPA가 활발히 사용되는 실제 프로젝트에 깊게 참여하도록 만드는 것이다. 짧은 훈련 수업에 참여하는 것은 STPA 전문가를 배출하는 데는 충분하지 않으며 몇 개의 대규모 STPA 프로젝트에 깊게 참여한 사람들이 향후 강사나 퍼실리테이터가 되기 위한 훌륭한 후보이다. 한 가지 좋은 방법은 실제 프로젝트를 수행하는 STPA 퍼실리테이터에게 한 명 또는 그 이상의 훈련 중인 퍼실리테이터가 그림자처럼 따라다니도록 하는 것이다. 훈련 중인 퍼실리테이터는 서로 다른 프로젝트에서 겪게 되는 어려움과 제기되는 의문을 직접 경험함으로써 많은 것을 배우게 된다. 효과적으로 가이드하고 문제를 해결할 수 있으며 STPA에 대한 상세한 질문에 대답할 수 있는 능력을 갖추면 퍼실리테이터가 될 준비가 된 것이다.

이 핸드북은 훈련 옵션을 더 제공해야 한다. 추가적으로, 우리는 핸드북과 함께 사용할 수 있는 영상을 제작 중이다.

퍼실리테이터(Facilitators)

STPA는 별도 그룹에서 독립적으로 수행되지 않고 설계 작업에 통합되어 수행될 때 가장 유용하다. 그러나 모든 사람이 STPA 전문가가 되기 위한 훈련을 원하지 않을 수 있고 그럴 필요도 없다. HAZOP에서 퍼실리테이터의 아이디어를 활용하는 것처럼 STPA에도 퍼실리테이터의 아이디어를 활용할 것을 권장한다. 즉, 시스템 설계 전문가가 최고의 STPA 팀을 위한 참여자이기는 하지만 만일 그가 STPA에 익숙하지 않다면 실제로 STPA를 적용하기 위한 가이드가 필요할 수 있다.

The STPA facilitator's job is to provide STPA expertise and guidance throughout the process and to help avoid common traps that may be encountered. The facilitator can provide guidance during project selection and scoping; can help identify initial team members and the expertise that will be needed; may develop an initial set of losses, hazards, and potentially an initial control structure based on previous similar projects; and provide overall coordination throughout the process. For large projects, the facilitator will break up the process into smaller components that can be done successfully by individuals or by small groups of experts. The facilitator can identify and schedule tasks that may be done in parallel by different groups, and eventually the facilitator can bring these components together into the overall analysis. Facilitators may also lead the meetings of the participants and manage the interactions among the team members.

The facilitator may provide some initial STPA training at the start of the project, and they should be able to answer any STPA-related questions that arise during the project. The facilitator generally reviews the results during the process and at the end of the process to ensure the method is being followed correctly and that no gaps are overlooked. At the conclusion of the project, the facilitator may compile the results into a final report and ensure the findings are provided to the appropriate people.

Team Composition

STPA is best performed by a small team that includes different types of experts. An initial control structure can help identify the areas of expertise that will be needed. The interdisciplinary team typically includes expertise from different areas such as a software expert, an operations expert, a maintenance expert, etc. In some cases it may not be possible to include all the relevant experts on the STPA team (for example when the experts are in high demand). In that case you may want to ensure that the team will at least have access to people with the right technical knowledge and that you can get answers to questions that arise during the analysis.

Personalities matter. The engineers and designers of a system may have the most expertise, but they can sometimes be defensive and see the STPA effort as a challenge to the quality of their work. They may not want to contribute potential UCAs because they believe they have already been mitigated in the design, or they may argue that identified UCAs were assumed to be impossible and should be removed or not analyzed. On the other hand, outsiders or other designers not involved in this particular design may have less knowledge about the system but they may not be as defensive and may be happy to propose potential flaws. It may help to explain to engineers and designers that STPA is a worst-case analysis method and that it can help make their job easier by avoiding rework, not harder. It is also helpful to begin applying STPA as early as possible, before the design is completed and before decisions have already been made. It is most advantageous when STPA is applied by the project engineers themselves to help guide decisions and make informed choices. In any case, personalities matter and ideally you should try to find knowledgeable experts who are open to new approaches and who aren't hesitant to identify problems that may have been overlooked previously.

We have often found that initial skepticism by those deeply involved in a project design is quickly overcome when they find UCAs or scenarios that had not been previously identified. They then quickly become STPA promoters.

Those with background or experience in testing can be excellent choices for an STPA team. Testers can be quite knowledgeable about what can go wrong and usually they are not only willing but eager to find problems that may have been overlooked. Testers rarely assume that requirements, specifications, or design decisions are correct and their job involves constantly questioning the assumptions and claims that have been made. These are all excellent characteristics for STPA team members.

STPA 퍼실리테이터의 임무는 프로세스 전체에 걸쳐 STPA 전문 지식과 가이드를 제공하고 흔히 발생할 수 있는 함정에 빠지지 않도록 하는 것이다. 퍼실리테이터는 프로젝트를 선택하거나 범위를 지정하는 과정에서 가이드를 제공할 수 있고 초기 팀 구성원 및 필요한 전문 지식을 식별하는 데 도움이 될 수 있다. 또한 과거 유사 프로젝트를 기반으로 손실(losses), 위험, 컨트롤 스트럭처에 대한 초기 세트(set)를 개발할 수 있고 프로세스 전반에 걸쳐 전체적인 조정 역할을 제공한다. 대규모 프로젝트에서 퍼실리테이터는 프로세스를 작은 구성요소로 분해하여 개인 또는 소규모 전문가 그룹이 성공적으로 수행할 수 있게 한다. 퍼실리테이터는 여러 그룹에서 작업을 병렬로 처리할 수 있도록 작업을 식별하고 스케줄 할 수 있으며 궁극적으로 이러한 구성요소를 묶어 종합적으로 분석할 수 있다. 또한 퍼실리테이터는 참여자들의 회의를 주도하고 팀원들 간의 상호작용을 관리할 수 있다.

퍼실리테이터는 프로젝트 시작 시점에 몇 가지 초기 STPA 훈련을 제공할 수 있으며 프로젝트 중에 발생하는 STPA 관련 질문에 답변할 수 있어야 한다. 퍼실리테이터는 일반적으로, 기법이 정확히 준수되었고 간과된 갭이 없는지를 확인하기 위해 프로세스 진행 중과 프로세스 종료 시에 결과를 검토한다. 프로젝트가 종료되면 퍼실리테이터는 결과들을 모아 최종 보고서를 작성하고 결과(findings)가 적절한 사람들에게 제공됨을 보장한다.

팀 구성(Team Composition)

STPA는 다양한 유형의 전문가가 있는 소규모 팀에서 수행하는 것이 가장 좋다. 초기 컨트롤 스트럭처는 필요한 전문 지식을 파악하는 데 도움이 될 수 있다. 여러 영역에서 모인 팀(interdisciplinary team)은 일반적으로 소프트웨어 전문가, 운영 전문가, 유지보수 전문가 등과 같이 다양한 분야의 전문 지식을 가진다. 경우에 따라 STPA 팀에 모든 관련된 전문가를 포함시키는 것이 불가능할 수도 있다(예를 들어 수요가 높은 전문가일 때). 이 경우 팀이 적어도 올바른 기술 지식을 가진 사람들에게 접근할 수 있도록 하여, 분석 중 발생하는 질문에 대한 답변을 얻게 할 수 있다.

성향(personalities)이 중요하다. 시스템 엔지니어와 설계자는 가장 전문 기술이 뛰어날 수 있지만 때로는 방어적이며 STPA 작업을 자신들의 업무 품질에 대한 도전으로 생각할 수도 있다. 그들은 설계에서 이미 UCA가 완화되었다고 생각하기 때문에 잠재적인 UCA에 기여하기를 원하지 않을 수 있다. 또한, 그들은 식별된 UCA가 불가능한 것이기 때문에 삭제되거나 분석되지 않아도 된다고 주장할 수도 있다. 반대로, 특정 설계에 관여하지 않은 외부인이나 다른 설계자는 시스템에 대한 지식이 부족할 순 있지만 방어적이지 않을 수 있고 잠재적 결함(flaw)을 기꺼이 말할 수 있다. STPA는 최악의 상황을 분석하는 방법(worst-case analysis method)이며 재작업을 피함으로써 일을 어렵지 않게 더 쉽게 할 수 있다고 엔지니어와 설계자에게 설명하는 것이 도움이 될 수 있다. 설계가 완료되기 전에 그리고 결정이 내려지기 전에, 가능하면 빨리 STPA를 적용하는 것이 도움이 된다. 프로젝트 엔지니어의 결정을 가이드하고 정보에 기반한 선택을 돕기 위해, 프로젝트 엔지니어가 STPA를 직접 적용하는 것이 가장 좋다. 어쨌든 성향은 중요하며 이상적으로는 새로운 접근법에 개방적이고, 이전에 간과했을 수 있는 문제의 식별을 주저하지 않는 지식이 풍부한 전문가를 찾기 위해 노력해야 한다.

우리는 종종 프로젝트 설계에 깊이 관여한 사람들의 초기의 회의적인 태도가 이전에 식별되지 않은 UCA나 시나리오가 발견되면 신속하게 극복된다는 것을 발견했다. 그 이후 그들은 곧 STPA 프로모터(promotor)가 된다.

테스팅 백그라운드나 경험이 있는 사람들도 STPA 팀을 위한 탁월한 선택일 수 있다. 테스터는 무엇이 잘못될 것인지를 잘 알고 있고, 일반적으로, 간과할 수 있는 문제를 찾을 의지가 있으며 또한 간절히 바란다. 테스터는 요구사항, 명세, 설계 결정이 정확하다고 가정하는 일이 거의 없으며 자신의 직

Cost and Return on Investment

STPA is very efficient. Figure 8.1 shows the relative amount of time spent by all STPA team members on a recent industry project. For most industry STPA projects, most of the time is actually not spent learning STPA or performing STPA but in learning how the system to be analyzed works (which would need to be done for any method that identifies hazard causes). About a quarter of the time is spent finding answers to "what if" technical questions about the system that had not have been considered previously. A relatively small amount of time is typically spent actually learning STPA and following the process.

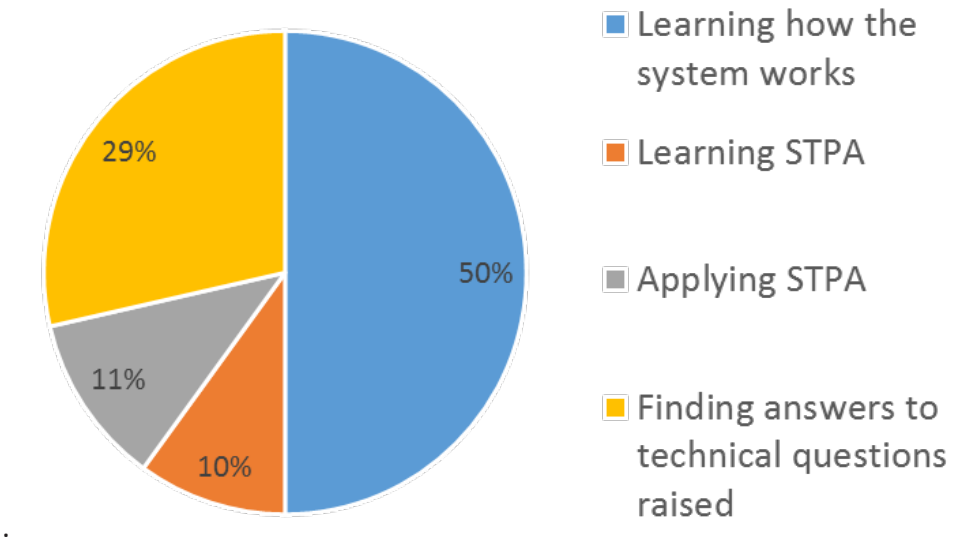


Figure 8.1: Relative amount of time spent on different tasks during a recent industry STPA project

STPA is not a paper-pushing exercise. Most of the time is spent exploring real concerns in the system that can immediately be used to drive decisions and provide insight about how to make improvements.

Although we do not have a lot of carefully collected data comparing the time required, people are reporting that STPA requires about 2 or 3 orders of magnitude less time and resources than the more traditional hazard analysis techniques and produces more complete results. Typically, the first use is most costly because of learning costs, but the cost quickly drops off on repeat projects.

An Example Introductory Use of STPA in a large organization

Chapter 4 describes the use of STPA in one large organization for a complex, highly automated system. We repeat it here. In this case, STPA was applied by an eight-member cross-functional team composed of members from integration, engineering, operations, and maintenance. All the people involved were considered to be leads or subject matter experts within their scope of responsibility for the system. The facilitator was an MIT master's student who was trained in STPA but had little knowledge about the new automated manufacturing process being introduced to the plant. The other members of the team had no knowledge of STPA at the beginning of the project.

The analysis was completed over the course of a three-week period. A group setting for the assessment was only used twice during the process for less than three hours each, while the rest of the assessment was completed in one-on-one meetings between the facilitator and the respective expert for the portion of the system being discussed.

무는 만들어진 가정과 주장에 끊임없이 의문을 제기하는 것이라고 여긴다. 이러한 모든 특징은 STPA 팀 구성원으로서 훌륭한 특징이다.

비용 및 투자 수익률(Cost and Return on Investment)

STPA는 매우 효율적이다. 그림 8.1은 최근 산업 프로젝트에서 모든 STPA 구성원이 소비한 상대적인 시간을 보여주고 있다. 대부분의 산업에서 STPA 프로젝트는 실제로 STPA를 학습하거나 STPA를 수행하기 위하여 시간을 사용하기 보다는 분석할 대상 시스템이 어떻게 동작하는지 파악하기 위해 대부분의 시간을 사용한다(이것은 위험 원인(hazard causes)을 식별하는 모든 기법에서 동일하게 수행되는 것임). 이전까지 고려되지 않았던, 시스템의 "...이라면 어떻게 되는가?(what if)"라는 기술적 질문에 대한 해답을 찾는 데에 약 25%의 시간이 소요된다. STPA를 실제로 배우고 프로세스를 따라가는 데에는 상대적으로 적은 시간이 소요되는 것이 일반적이다.

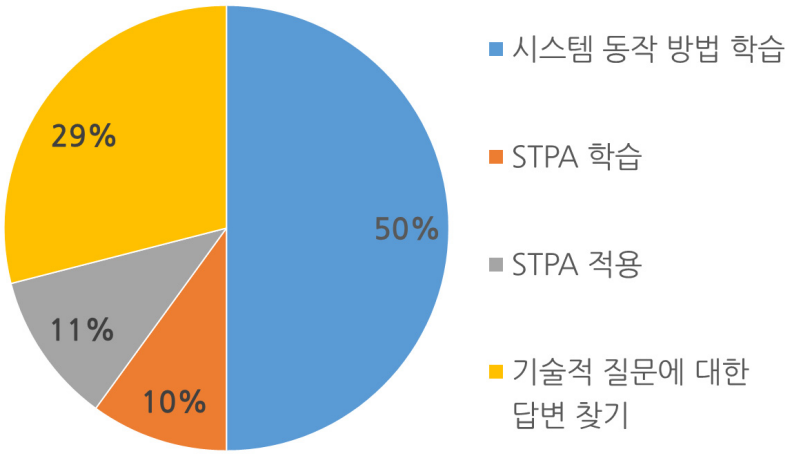


그림 8.1: 최근 산업계의 STPA 프로젝트에서 서로 다른 업무에 소요된 상대적인 시간

STPA는 형식적으로 문서만 작성하는 활동(paper-pushing exercise)이 아니다. 대부분의 시간은 시스템의 실제 관심사(의사 결정을 유도하고 어떻게 개선할지에 대한 통찰력을 제공하는 데 즉시 사용될 수 있는 것)를 찾는 데 사용된다.

비록 수집에 소요된 시간에 비해 면밀히 수집된 데이터가 충분하지는 않지만, STPA는 기존의 위험 분석 기법보다 시간과 자원이 약 2~3배 덜 필요하며 보다 완벽한 결과를 도출해낸다고 사람들이 보고한 바 있다. 통상적으로 STPA를 처음 사용할 때는 학습 비용으로 인하여 가장 많은 비용이 들지만, 프로젝트를 반복 수행할 경우에는 비용이 급감한다.

대규모 조직에서의 STPA 도입 사례

4장은 복잡하고 고도로 자동화된 시스템을 위해 하나의 대규모 조직에서 STPA를 사용하는 방법에 대해 설명하고 있는데 여기서 다시 한번 언급한다. 이 경우에는 통합, 엔지니어링, 운영 및 유지보수에서 선발된 8명의 복합기능팀(cross-functional team)에서 STPA를 적용하였다. 관련된 모든 사람들은 시스템 책임 범위의 리더 또는 주제에 대한 전문가로 볼 수 있다. 퍼실리테이터는 MIT 석사과정 학생으로, STPA 훈련은 받았지만 공장에 도입되는 새로운 제조 자동화 공정에 대한 지식이 거의 없었다. 팀의

Because STAMP and STPA were relatively new for the company and entirely new for the team, the first group meeting was used as an instruction period to provide an overview of the methodology being used and then to define the accidents (or losses) the team wanted to avoid, the hazards and system boundaries, and an initial draft of the safety control structure for this application.

The initial control structure was developed by the group as a whole to ensure that all of the major components were captured. After developing the first draft of the control structure, follow-on meetings were scheduled with individual experts or small groups with the appropriate technical expertise on the different parts of the system to add detail.

Once the appropriate level of control structure detail was developed, the one-on-one or small group discussions continued with the identification of Unsafe Control Actions and scenarios. The results of this portion of the analysis were then discussed in another group setting to confirm the individual findings of the group and discuss potential mitigations for the system. This process occurred over the course of three weeks, and the time spent in group, one-on-one meetings, and individual facilitator work totaled approximately 300 person hours.

In general, the scenarios identified included such causal factors as hardware failures, incorrect process models (stemming from missing or incorrect feedback), system component interactions, and the role of management and procedures on safety combined with schedule pressures in the factory. After the team identified STPA scenarios, they used the results to develop new controls for the system in order to eliminate or prevent entering hazardous states. Examples are included in Chapter 4 of this handbook.

Because STPA uses a model of the control structure, the assessment can focus on specific control loops within the model after the complete model is constructed. This ability to divide up the modeling and analysis process provided for easy scheduling of meeting times and numerous meetings within the available time frames of everyone within the group. During these meetings, details such as control actions, process models, feedback, etc. were added and any conflicts or issues were resolved with the affected parties.

Other Advice

We intend to include more information about integrating STPA into large organizations as we obtain it, including return on investment data. The fact that so many organizations are adopting STPA after trying it on several projects and comparing it with what they do now is encouraging. If you have any input to provide, please tell us.

다른 구성원은 프로젝트 초기에 STPA를 알지 못한 상태였다.

분석은 3주간 진행하여 완료되었다. 평가(assessment)를 위한 그룹 설정은 프로세스에서 각 3시간 미만으로 단 2회만 있었으며, 나머지 평가는 논의되는 시스템 각 부분에 대해 퍼실리테이터와 각 전문가 간의 1:1 미팅을 통해 완료되었다.

STAMP와 STPA는 회사 입장에서 비교적 새롭고 팀 입장에서도 완전히 새로운 것이었기 때문에 첫 그룹 미팅은 이론을 가르쳐주는 시간(instruction period)으로 사용되었는데 사용될 기법에 대한 개요를 소개한 다음,팀에서 피하고자 하는 사고(또는 손실), 위험과 시스템 경계, 그리고 이 어플리케이션에 대한 안전 컨트롤 스트럭처의 초안을 정의하였다.

첫 컨트롤 스트럭처를 개발할 때는 모든 주요 컴포넌트가 포함되었는지를 보장하기 위해 모두가 하나의 그룹이 되어 개발하였다. 컨트롤 스트럭처 초안 개발 후, 세부 사항 추가를 위해 시스템 각 부분의 적절한 기술 전문성을 갖춘 개별 전문가나 소그룹과의 후속 회의가 스케줄되었다.

적절한 수준의 컨트롤 스트럭처 세부 사항이 개발되면 1:1 또는 소그룹 토론이 이어져 UCA와 시나리오를 식별하였다. 그런 다음, 분석의 부분적인 결과는 또 다른 그룹에서 논의되는데 여기서는 각각의 결과를 컨펌하고 잠재적인 시스템 완화조치(mitigation)를 논의하였다. 이 과정은 3주에 걸쳐 이루어졌으며 그룹, 1:1 미팅 및 각 퍼실리테이터 작업에 소요된 총 시간은 약 300시간이었다.

일반적으로, 식별된 시나리오의 원인 요인으로는 하드웨어 고장, 잘못된 프로세스 모델(누락되거나 부정확한 피드백에서 기인), 시스템 컴포넌트의 상호작용이 있으며 공장의 일정 압박과 결합된 안전절차 및 관리의 역할도 포함된다. 팀이 STPA 시나리오를 식별한 후에는 그 결과를 사용하여 시스템이 위험한 상태에 빠지는 것을 방지 또는 제거하기 위해 시스템에 새로운 컨트롤을 개발한다. 이 핸드북의 4장에 그에 해당하는 예제가 있다.

STPA는 컨트롤 스트럭처 모델을 사용하기 때문에 평가(assessment)는 전체 모델이 구축된 후 모델 내의 특정 컨트롤 루프에 집중할 수 있다. 모델링과 분석 프로세스를 분리할 수 있는 이러한 STPA의 특징으로 인해 그룹 내 모든 사람들의 가능한 시간 범위 내에서 수많은 회의와 회의시간을 쉽게 스케줄링 할 수 있었다. 회의를 수행하는 과정에서 컨트롤 액션, 프로세스 모델, 피드백 등과 같은 세부 사항이 추가되었으며, 갈등(conflict) 또는 이슈는 관련된 당사자 간 해결하였다.

기타 조언

우리는 정보가 확보되는 대로 투자 수익률(ROI) 데이터를 포함하여 STPA를 대규모 조직에 통합하는 것에 대한 더 많은 정보를 제공하려고 한다. 상당히 많은 조직이 STPA를 여러 프로젝트에 적용하고, 이전에 수행한 것과 비교한 후에 STPA를 채택한다는 사실은 고무적이다. 만일 어떤 의견이라도 있다면 알려주길 바란다.

Appendix A : Examples of Hazards

Hazards are often very similar within an industry. Once you have identified the hazards appropriate for your industry, product, or services, you are likely to be able to reuse the list with perhaps small changes. This appendix contains some example lists of hazards that have been used on real projects. Of course, the stakeholders are the ultimate decision makers about what types of losses and hazards will be considered. The losses that are considered important to the stakeholders for a particular project will obviously affect the hazards identified.

Nuclear Power Plant

Losses:

- L1: People injured or killed
- L2: Environment contaminated
- L3: Equipment damage (economic loss)
- L4: Loss of electrical power generation

Hazards

- H1: Release of radioactive materials [L1, L2, L3, L4]
- H2: Reactor temperature too high [L1, L2, L3, L4]
- H3: Equipment operated beyond limits [L3, L4]
- H4: Reactor shut down [L4]

Aircraft

Losses:

- L1. Loss of life or serious injury to people
- L2. Damage to the aircraft or objects outside the aircraft

Hazards

- H-1: Aircraft violate minimum separation standards in flight [L1, L2]
- H-2: Controlled flight of aircraft into terrain [L1, L2]
- H-3: Loss of aircraft control [L1, L2]
- H-4: Aircraft airframe integrity is lost [L1, L2]
- H-5: Aircraft environment is harmful to human health [L1, L2]
- H-6: Aircraft departs designated taxiway, runway, or apron on ground [L1, L2]
- H-7: Aircraft comes too close to other objects on the ground [L1, L2]

Radiation Therapy

Losses:

- L1: The patient is injured or killed from overexposure or undertreatment.
- L2: A nonpatient is injured or killed by radiation.
- L3: Damage or loss of equipment.
- L4: Physical injury to a patient or nonpatient during treatment.

부록 A: 위험(Hazard)의 예

위험은 하나의 산업분야 내에서는 종종 매우 유사하다. 산업, 제품, 서비스에 적합한 위험을 일단 식별하면 약간만 변경하여 위험 목록을 재사용할 수 있다. 본 부록에서는 실제 프로젝트에서 활용되었던 위험 목록의 사례를 소개한다. 물론 어떤 유형의 손실(loss)과 위험을 고려해야 하는지 판단하는 궁극적인 의사결정권자는 이해관계자(stakeholder)이다. 특정 프로젝트의 이해관계자들이 중요하다고 생각하는 손실들이 위험을 식별하는 데 큰 영향을 미친다.

원자력발전소(nuclear power plant)

손실(Losses):

L1: 사람의 부상 또는 사망
L2: 환경오염
L3: 장비손상(경제적 손실)
L4: 전력생산 손실

위험(Hazards):

H1: 방사성 물질의 유출 [L1, L2, L3, L4]
H2: 원자로 온도가 너무 높음 [L1, L2, L3, L4]
H3: 장비가 한계치를 초과하여 운영됨 [L3, L4]
H4: 원자로 정지 [L4]

항공기(aircraft)

손실(Losses):

L1: 인명 손실 또는 심각한 부상
L2: 항공기 또는 항공기 외부의 물체 손상

위험(Hazards):

H-1: 항공기가 비행 중 최소이격기준을 위반함 [L1, L2]
H-2: CFIT(Controlled flight into terrain, 역주: 정상적으로 제어 중인 항공기가 의도하지 않게 지면, 산, 물, 장애물을 향해 비행하여 기체가 파손되는 사고) [L1, L2]
H-3: 항공기 제어의 불가 [L1, L2]
H-4: 항공기 기체 무결성이 손실됨 [L1, L2]
H-5: 항공기 환경이 사람의 건강에 해로움 [L1, L2]
H-6: 항공기가 지상에 있는 지정된 유도로, 활주로, 계류장을 이탈함 [L1, L2]
H-7: 항공기가 지상의 물체에 너무 가까이 접근함 [L1, L2]

방사선 치료(radiation therapy)

손실(Losses):

L1: 환자가 방사선에 과다 노출 또는 치료 부족으로 인하여 부상을 입거나 사망함
L2: 환자가 아닌 사람이 방사선에 의해 부상을 입거나 사망함
L3: 장비가 손상되거나(damage) 손실(loss)됨
L4: 치료 중 환자나 환자가 아닌 사람이 신체적 상해를 입음

위험(Hazards):

Hazards:

H1: Wrong dose: Dose delivered to patient is wrong in either amount, location, or timing [L1].

H1a: Right patient, right dose, wrong location.

H1b: Right patient, wrong dose, right location.

H1c: Right patient, wrong dose, wrong location.

H1d: Wrong patient.

H2: A nonpatient is unnecessarily exposed to radiation [L2]

H3: Equipment is subject to unnecessary stress [L3].

H4: Persons are subjected to nonradiological injury [L4].

Military Aviation

Mishaps

M-1: Loss of or damage to the aircraft or equipment on the aircraft

M-2: Serious injury or fatality to personnel

M-3: Inability to complete the mission

Hazards

H-1: Violation of minimum separation standards from fixed or moving objects [M-1, M-2, M-3]

H-2: Inability to control the aircraft [M-1, M-2, M-3]

H-3: Loss of airframe integrity [M-1, M-2, M-3]

Additional hazards may be relevant depending on the missions of the aircraft. As an example, for the weapon system on the aircraft:

H-4: Uncommanded detonation [M-1, M-2, M-3]

H-5: Uncommanded launch [M-1, M-2, M-3]

H-6: Collateral damage or friendly fire [M-1, M-2, M-3]

H-7: Non-deployment (detonation and/or launch) of ordinance when commanded [M-3]

Automotive

Losses

L1: Loss of life or serious injury to people

L2: Damage to the vehicle or objects outside the vehicle

Hazards

H1: Vehicle does not maintain safe distance from nearby objects [L1, L2]

H2: Vehicle enters dangerous area/region [L1, L2]

H3: Vehicle exceeds safe operating envelope for environment (speed, lateral/longitudinal forces) [L1, L2]

H4: Vehicle occupants exposed to harmful effects and/or health hazards [L1, L2]
(e.g. fire, excessive temperature, inability to escape, door closes on passengers, etc.)

H1: 잘못된 투여량: 환자에게 처방된 투여의 양(amount), 위치(location) 또는 타이밍(timing)이 잘못됨 [L1]
H1a: 대상 환자 맞음, 투여량 맞음, 위치 잘못됨
H1b: 대상 환자 맞음, 투여량 잘못됨, 위치 맞음
H1c: 대상 환자 맞음, 투여량 잘못됨, 위치 잘못됨
H1d: 대상 환자가 아님
H2: 환자가 아닌 사람이 불필요하게 방사선에 노출됨 [L2]
H3: 장비가 불필요한 스트레스를 받음 [L3]
H4: 사람이 비(非)방사선적인(nonradiological) 상해를 입음 [L4]

군용 항공(Military Aviation)

미스햅(Mishaps):
M-1: 항공기 또는 항공기 장비의 손실(loss) 또는 손상(damage)
M-2: 사람의 심각한 부상 또는 사망
M-3: 미션 완수 불가능

위험(Hazards):
H-1: 고정된 또는 이동하는 물체로부터의 최소이격기준을 위반 [M-1, M-2, M-3]
H-2: 항공기 제어 불능 [M-1, M-2, M-3]
H-3: 기체 무결성 손실 [M-1, M-2, M-3]

항공기의 미션에 따라 관련된 추가 위험이 있을 수 있다. 예를 들어, 항공기의 무기 시스템의 경우,

H-4: 명령하지 않은 폭발(detonation) [M-1, M-2, M-3]
H-5: 명령하지 않은 발사(launch) [M-1, M-2, M-3]
H-6: 부수적 피해(역주: 군사 행동으로 인한 민간인의 인적·물적 피해) 또는 아군을 포격 [M-1, M-2, M-3]
H-7: 명령이 내려졌으나, 전개(폭발 및/또는 발사)되지 않음 [M-3]

자동차(automotive)

손실(Losses):
L1: 인명 손실(loss) 또는 심각한 부상
L2: 차량 또는 차량 외부 물체의 손상(damage)

위험(Hazards):
H1: 차량이 주변 물체와 안전거리를 유지하지 않음 [L1, L2]
H2: 차량이 위험한 구역/지역에 진입함 [L1, L2]
H3: 차량이 환경(속도, 횡력/종력)에 대한 안전한 동작 범위를 초과함 [L1, L2]
H4: 차량 탑승자가 유해한 영향 및/또는 건강상 위험에 노출됨 [L1, L2]
(예: 화재, 과도한 온도, 탈출 불가, 문 닫힘으로 인한 위험(door closes on passengers, 역주: 사람이 차량에 갇힌 상황 등)

Appendix B: Example Hierarchical Control Structures

This appendix shows some control structures we have used on our own projects.

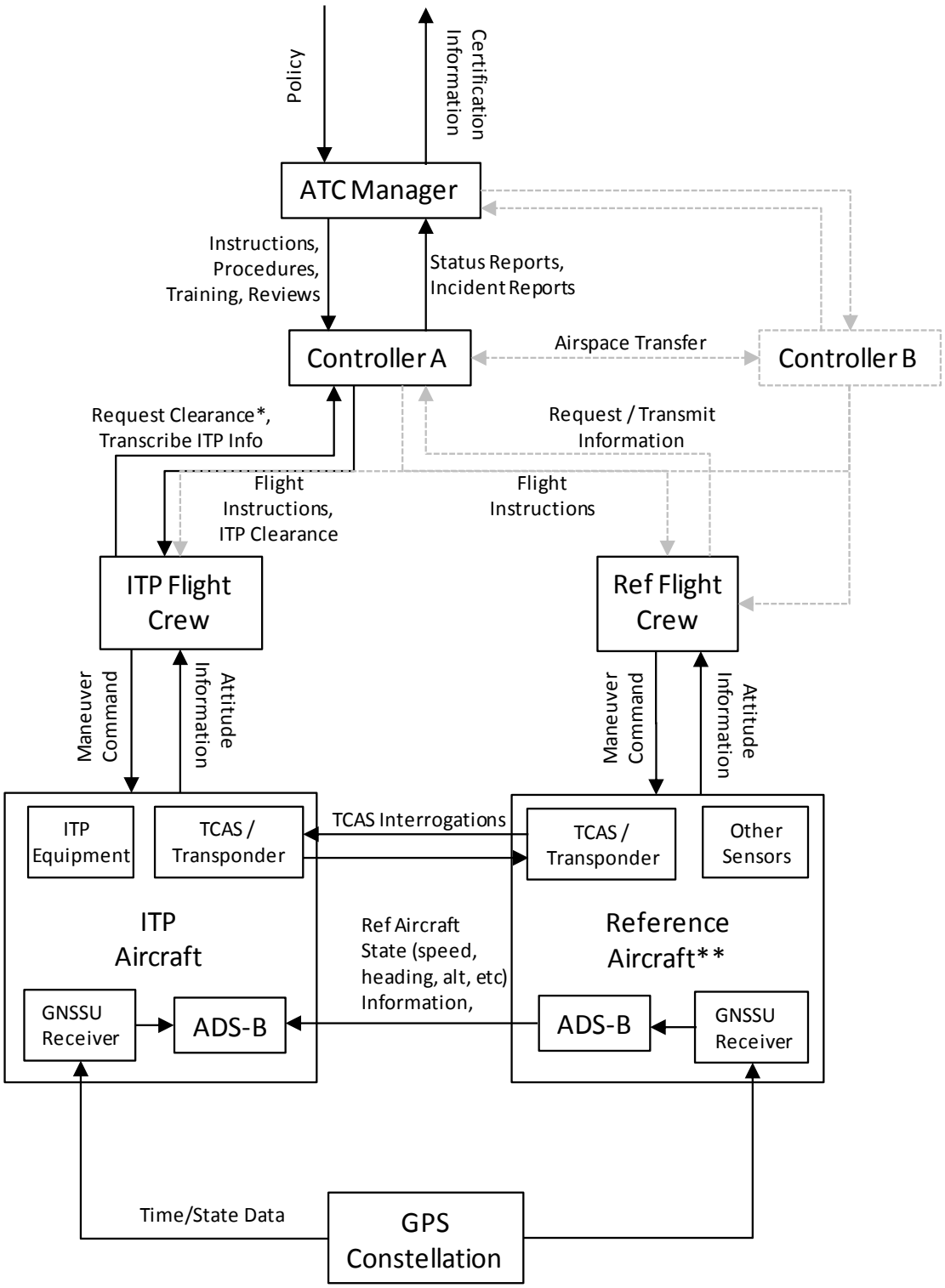


Figure B.1: Control structure for NextGen In-Trail Procedure with new equipment (aviation)

부록 B: 컨트롤 스트럭처의 예

이 부록에서는 우리가 자체 프로젝트에서 사용한 몇 가지 컨트롤 스트럭처를 소개한다.

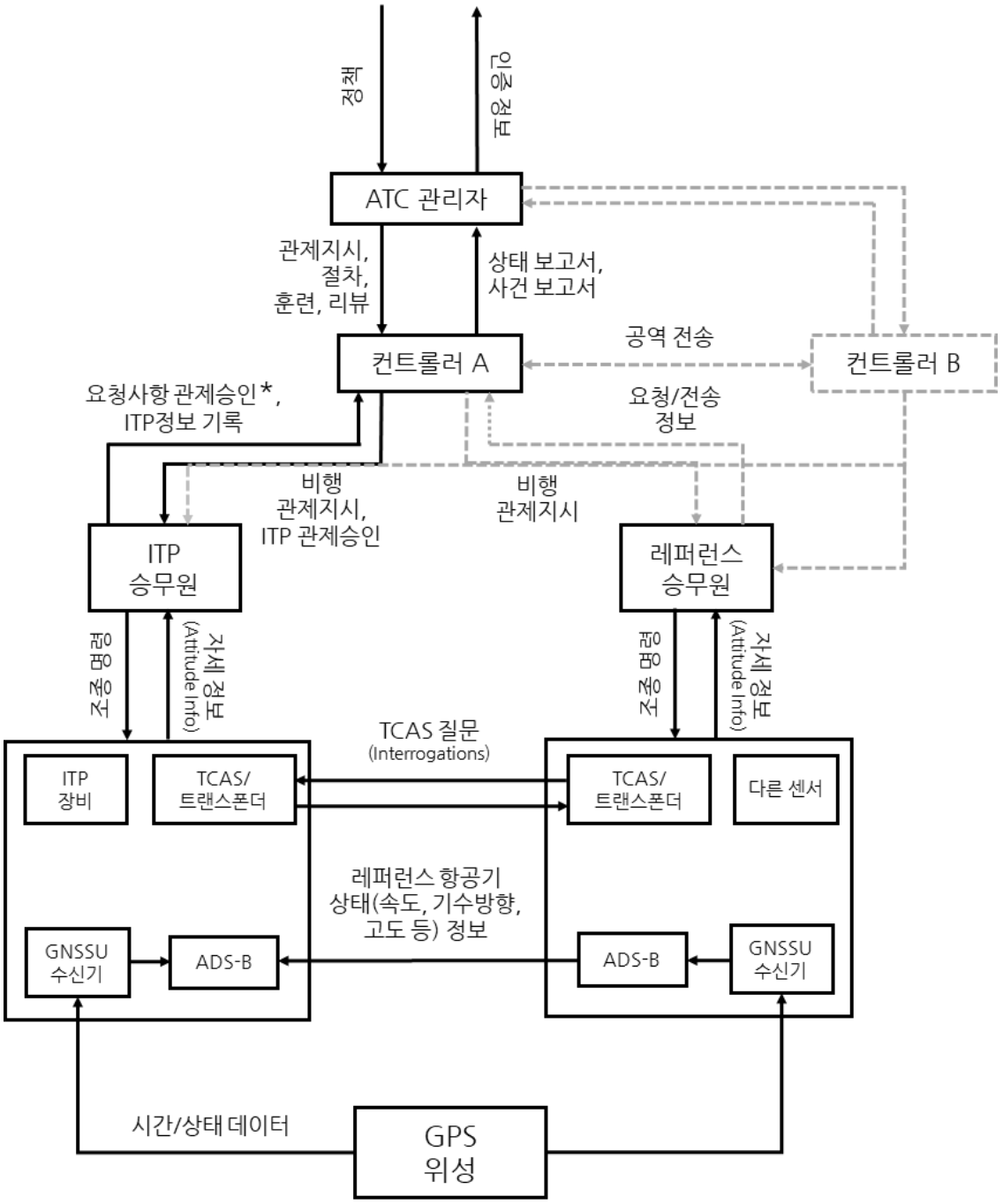


그림 B.1: 새로운 장비를 사용하는 차세대 시스템(NextGen)의 ITP(In-Trail Procedure)에 대한 컨트롤 스트럭처(항공)

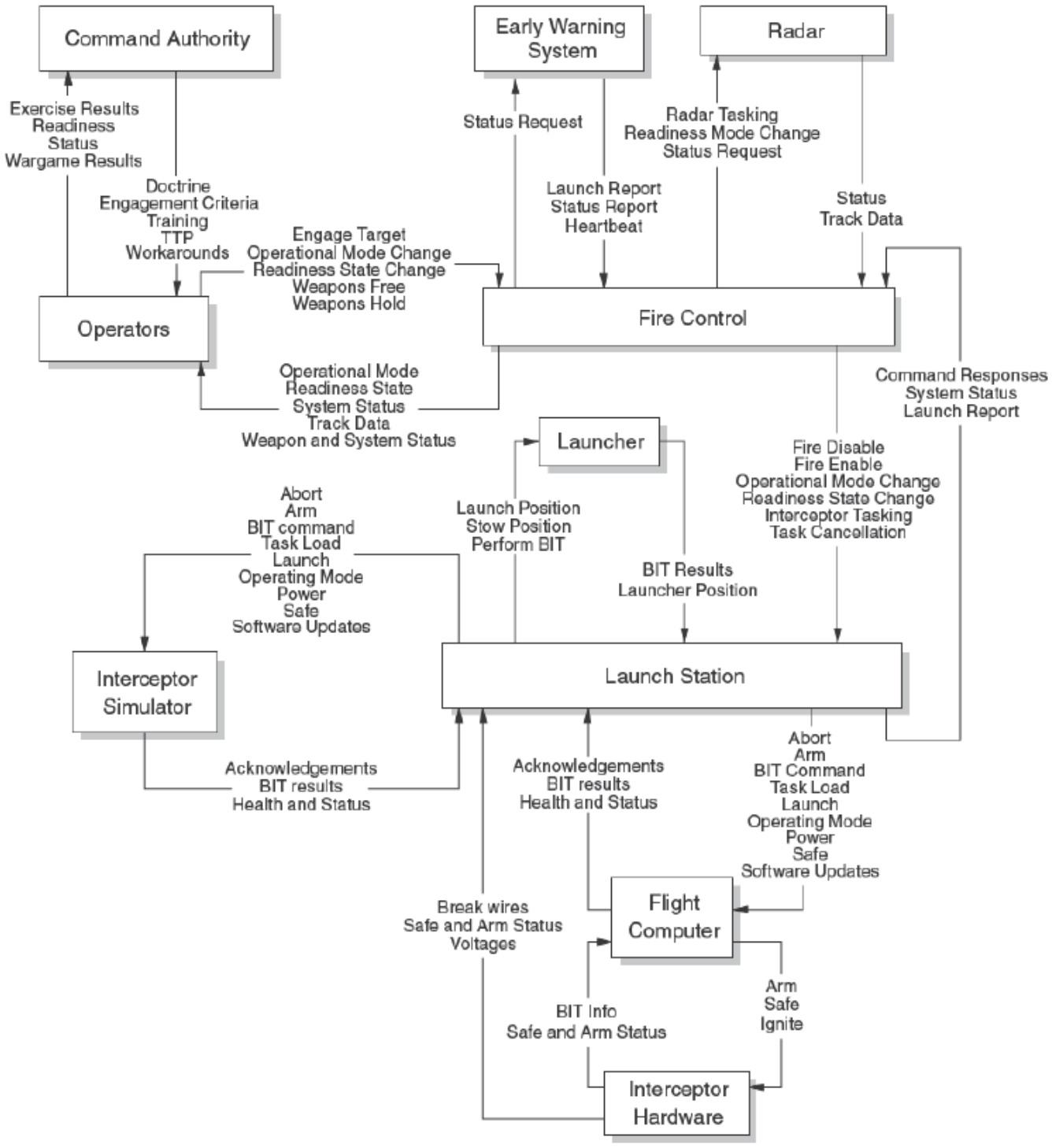


Figure B.2: Control structure for a Fictional Missile Intercept System

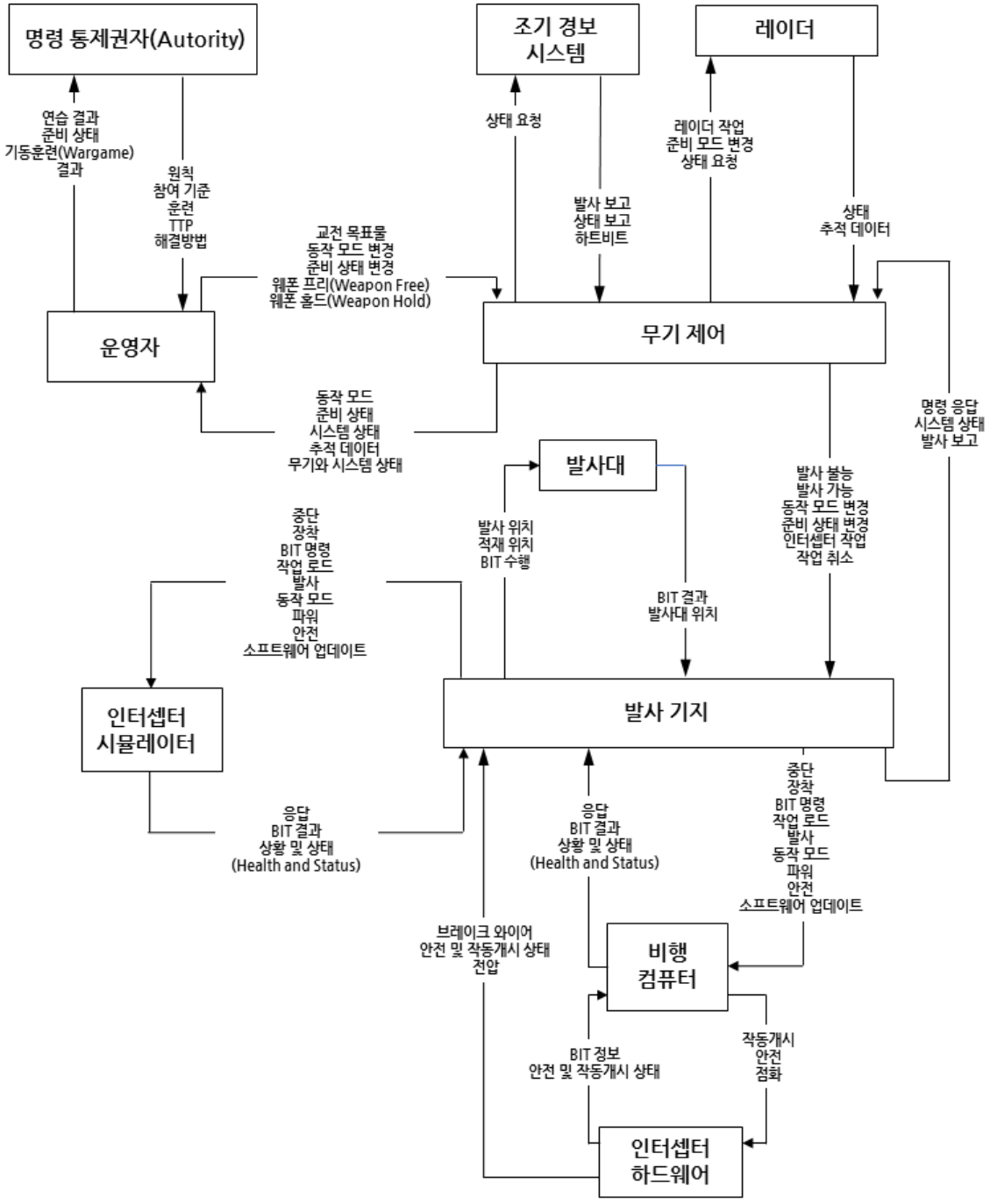


그림 B.2: 가상 미사일 요격(Intercept) 시스템의 컨트롤 스트럭처

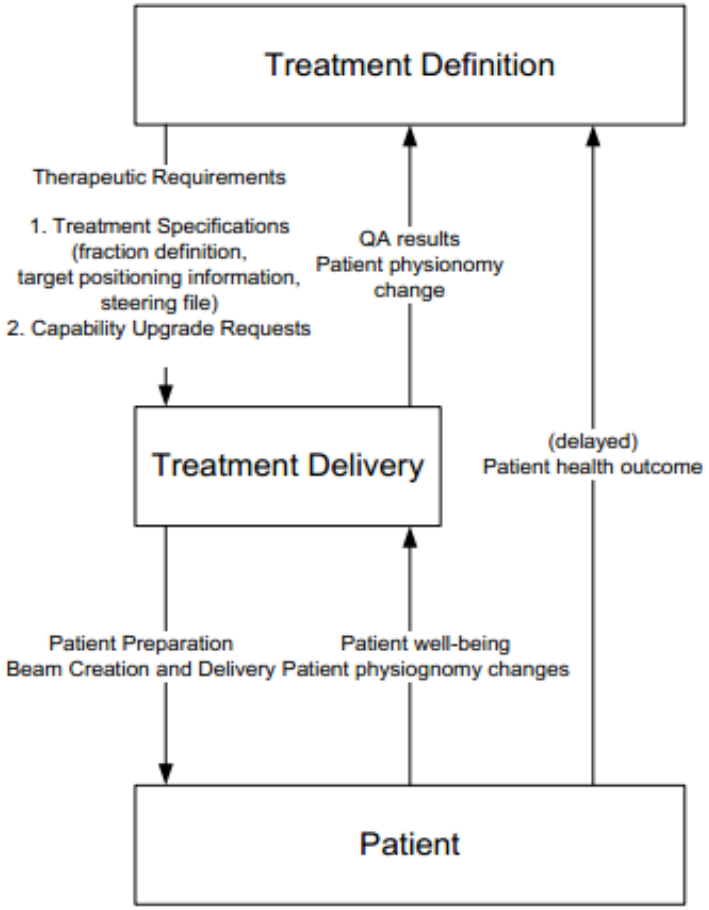


Figure B.3: High-level (abstract) control structure for a proton therapy machine

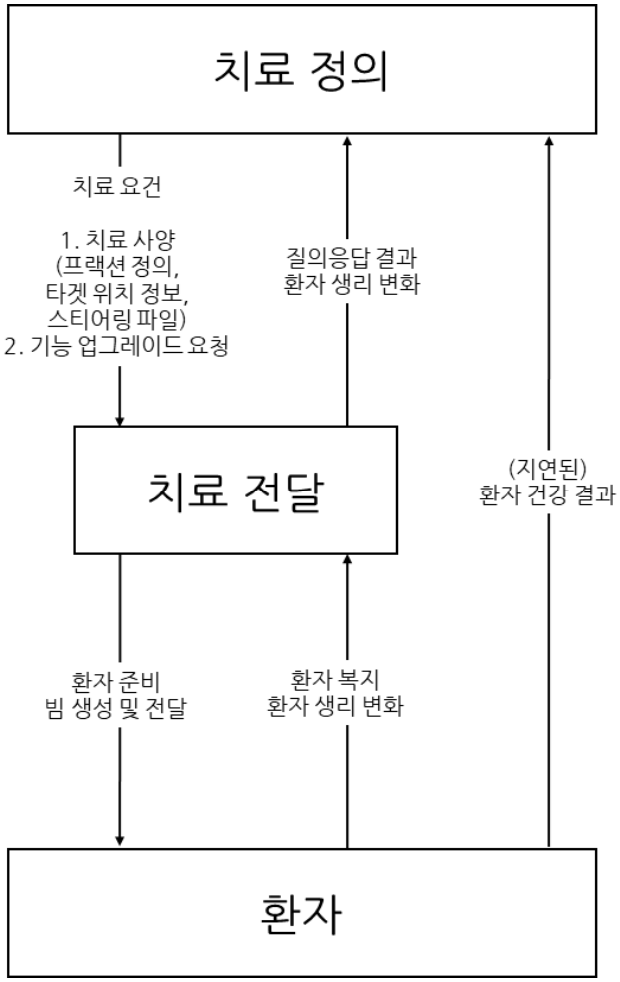


그림 B.3: 양성자 치료기(proton therapy machine)에 대한 상위(추상화) 수준의 컨트롤 스트럭처

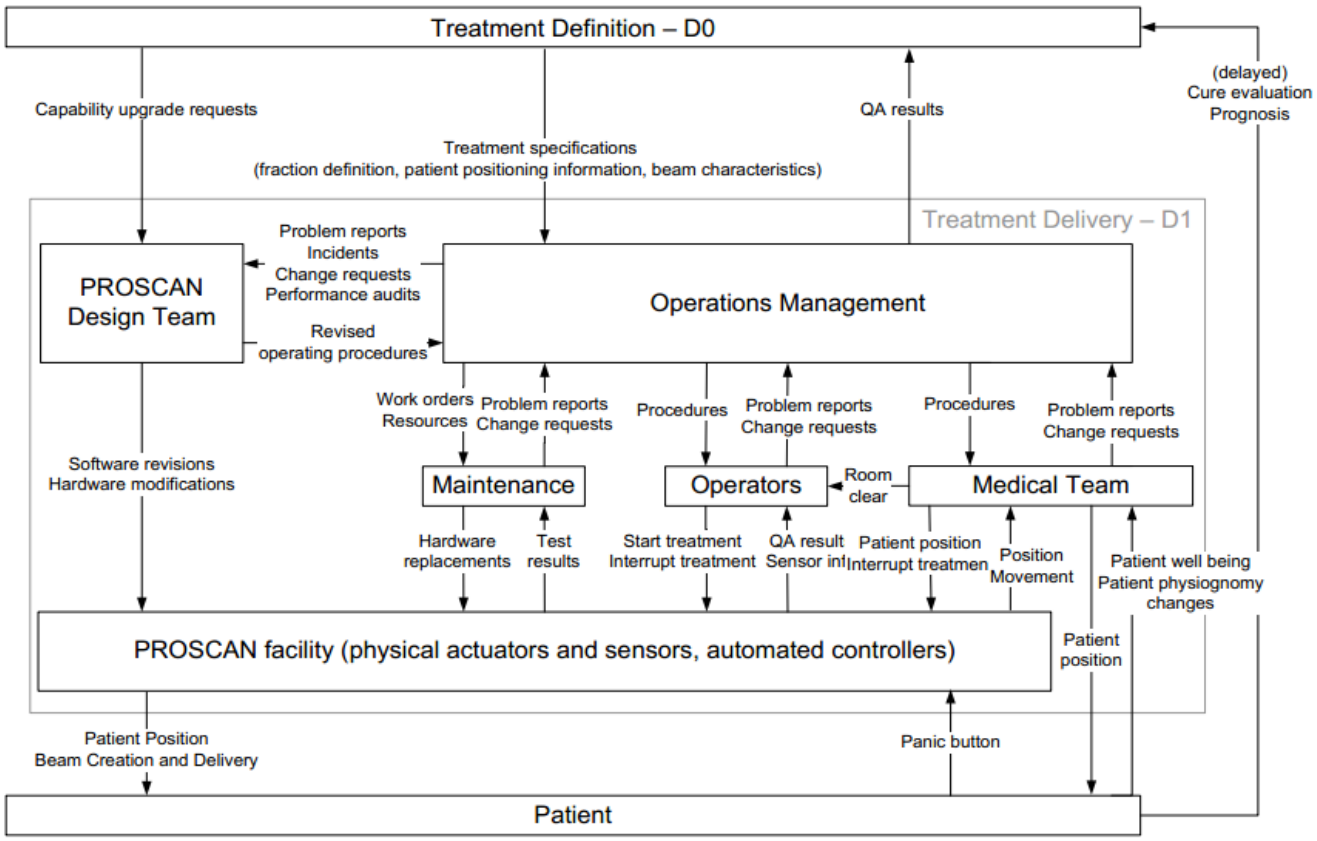


Figure B.4: Zooming into the Treatment Delivery part of Figure B.3 with the Treatment Definition and Patient components shown to provide context

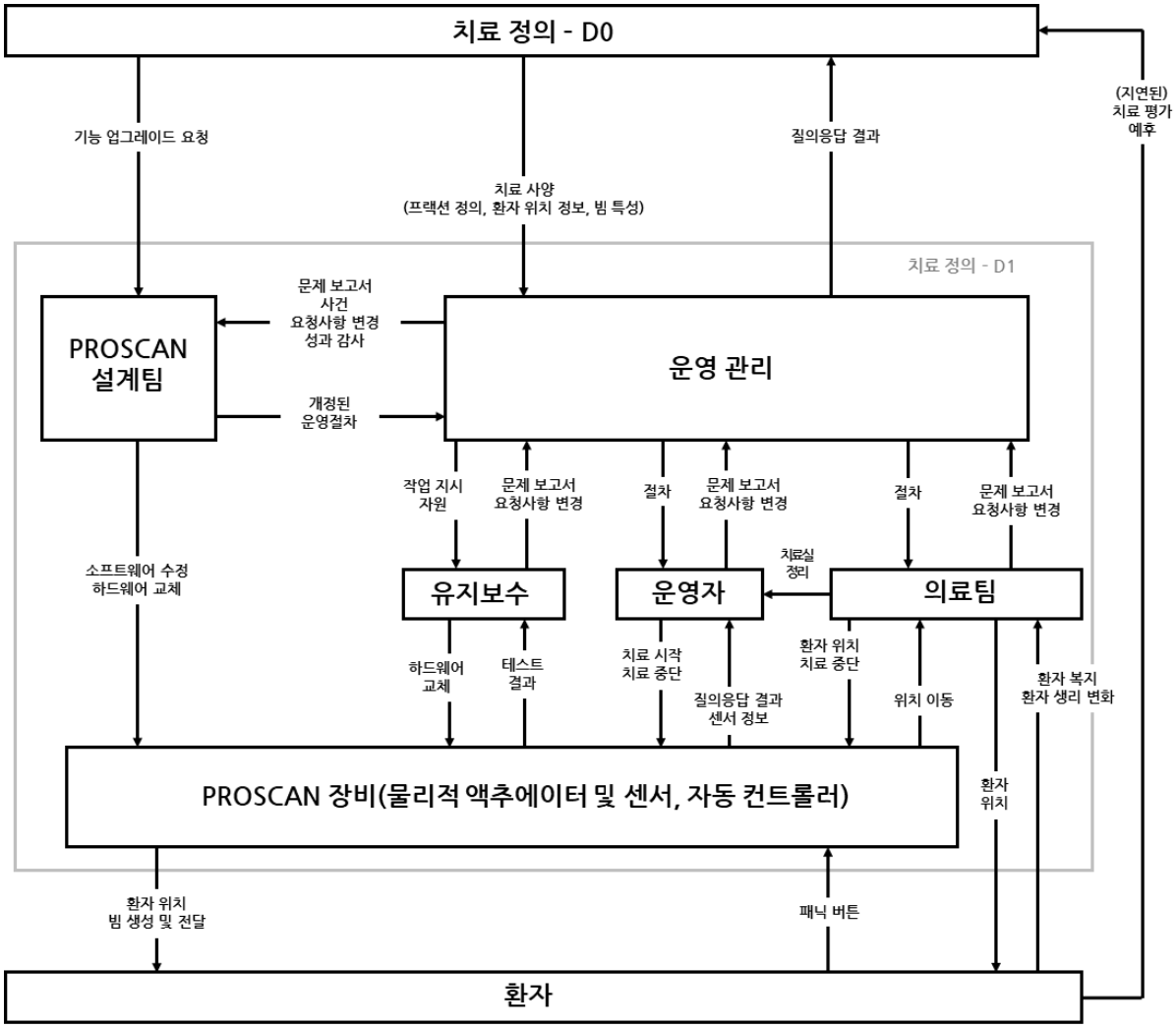


그림 B.4: 컨텍스트를 제공하기 위해 그림 B.3의 '치료 제공' 부분을 확대(상세화)

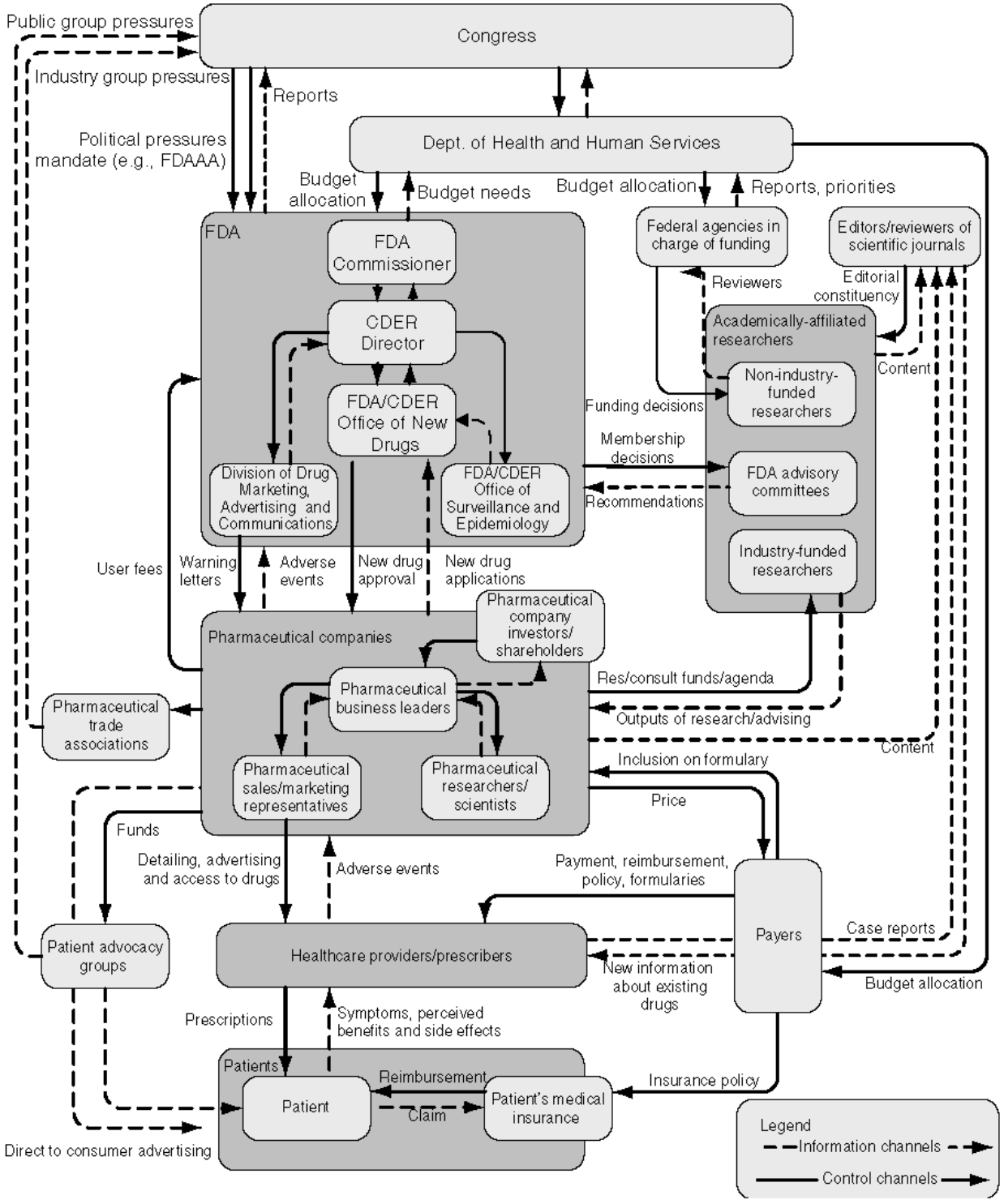


Figure B.5: U.S. Pharmaceutical Approval Safety Control Structure

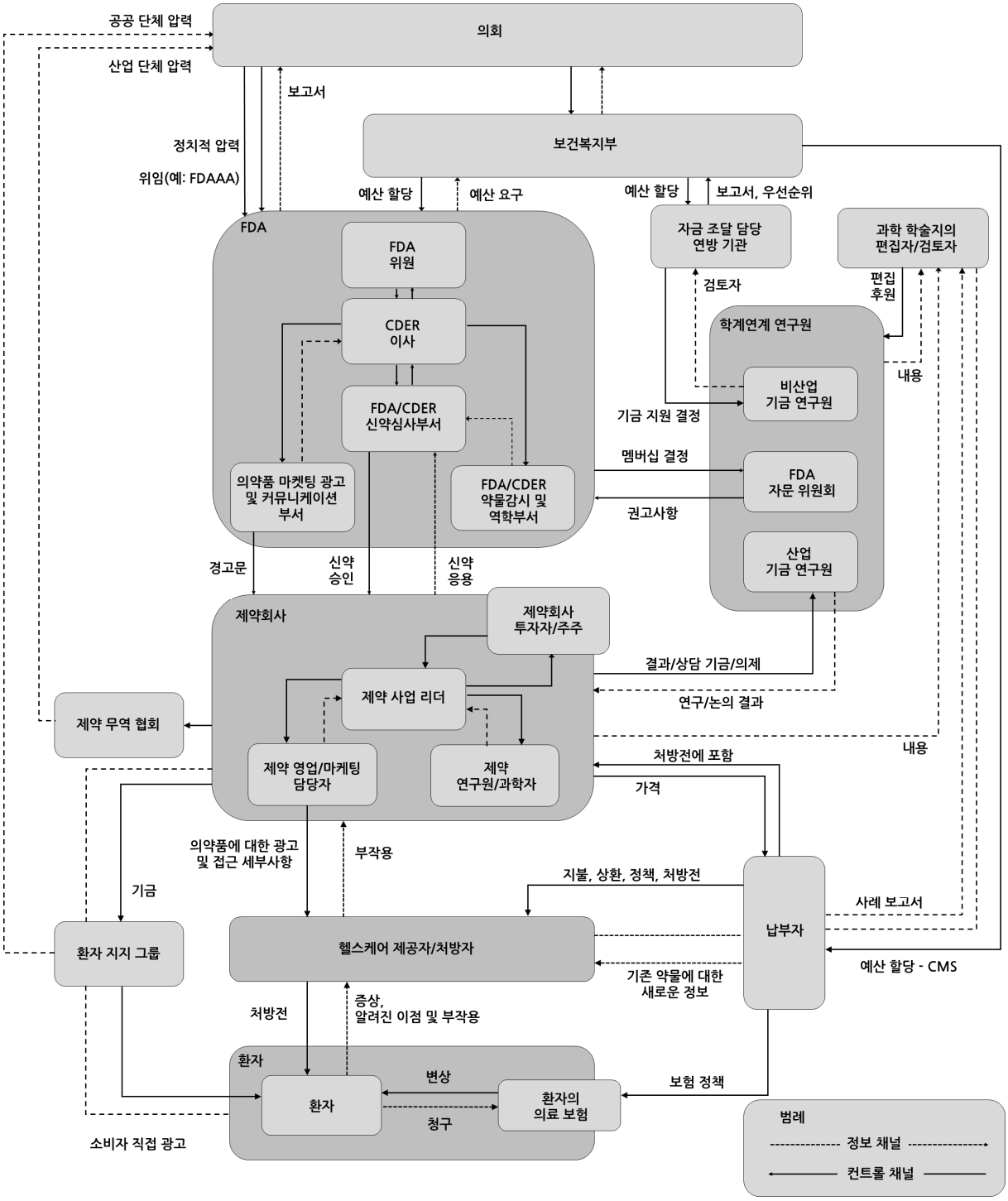


그림 B.5: 미국 제약 승인(U.S. Pharmaceutical Approval)의 안전 컨트롤 스트럭처

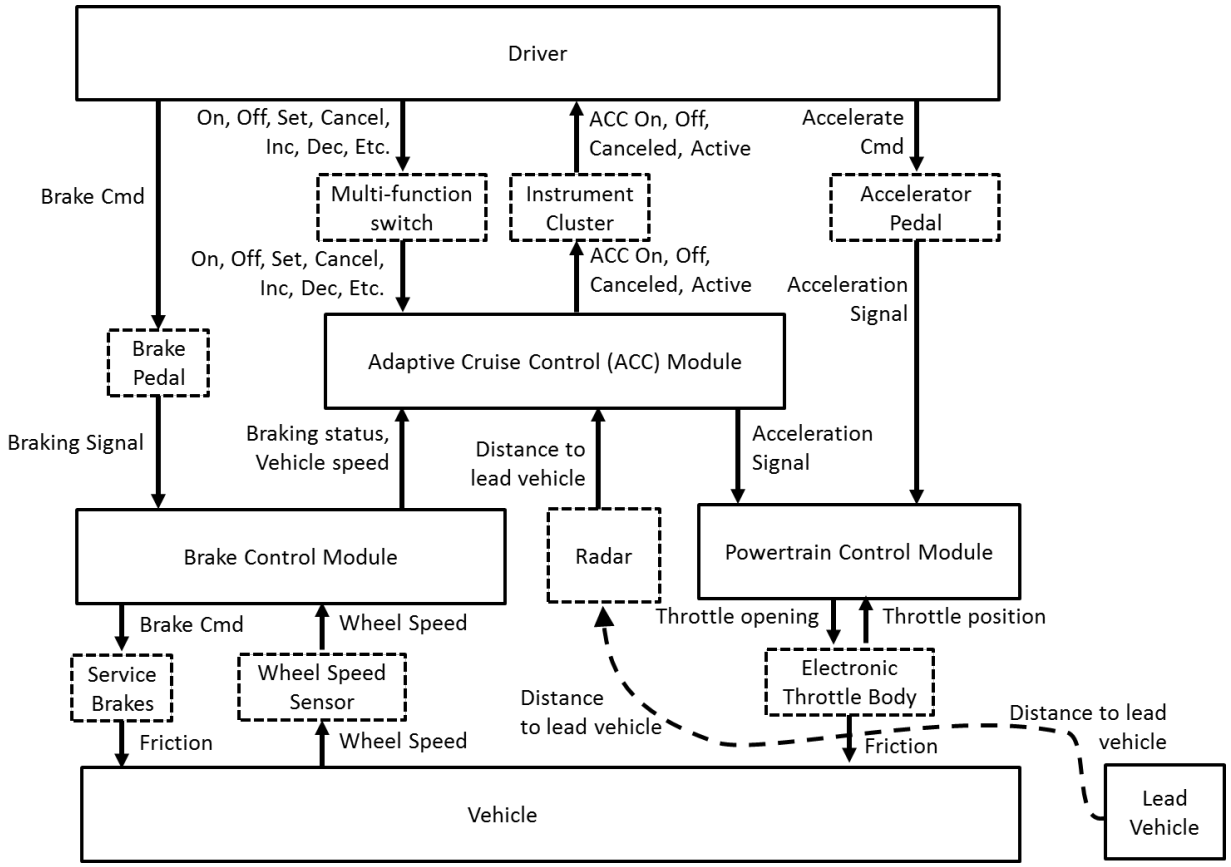


Figure B.6: Automotive Adaptive Cruise Control System

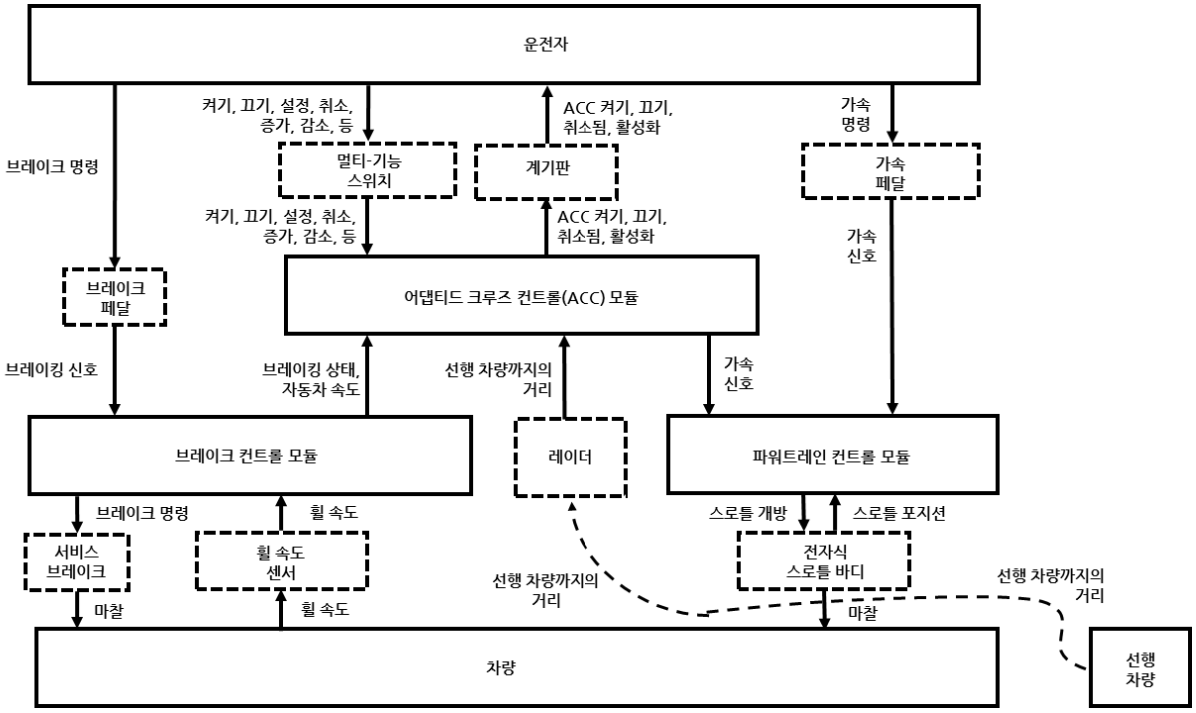


그림 B.6: 차량용 어댑티브 크루즈 컨트롤(Adaptive Cruise Control) 시스템

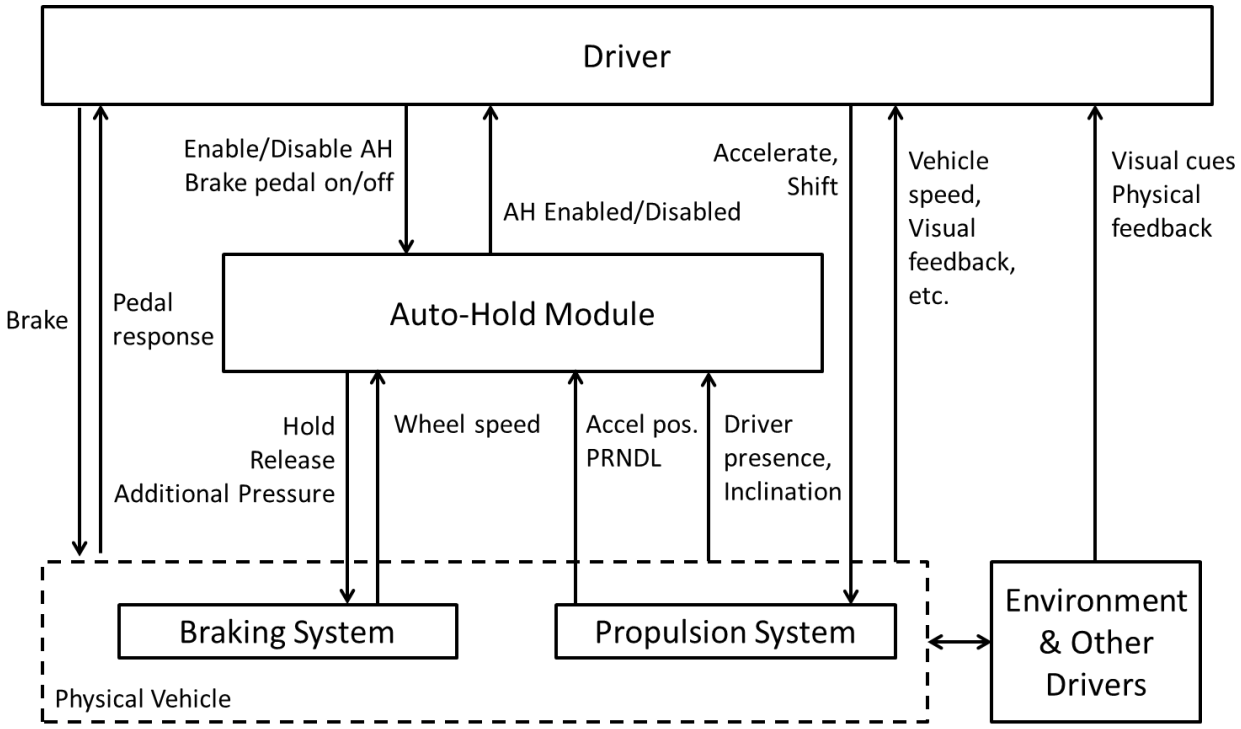


Figure B.7: Control Structure for Automotive Auto-Hold System

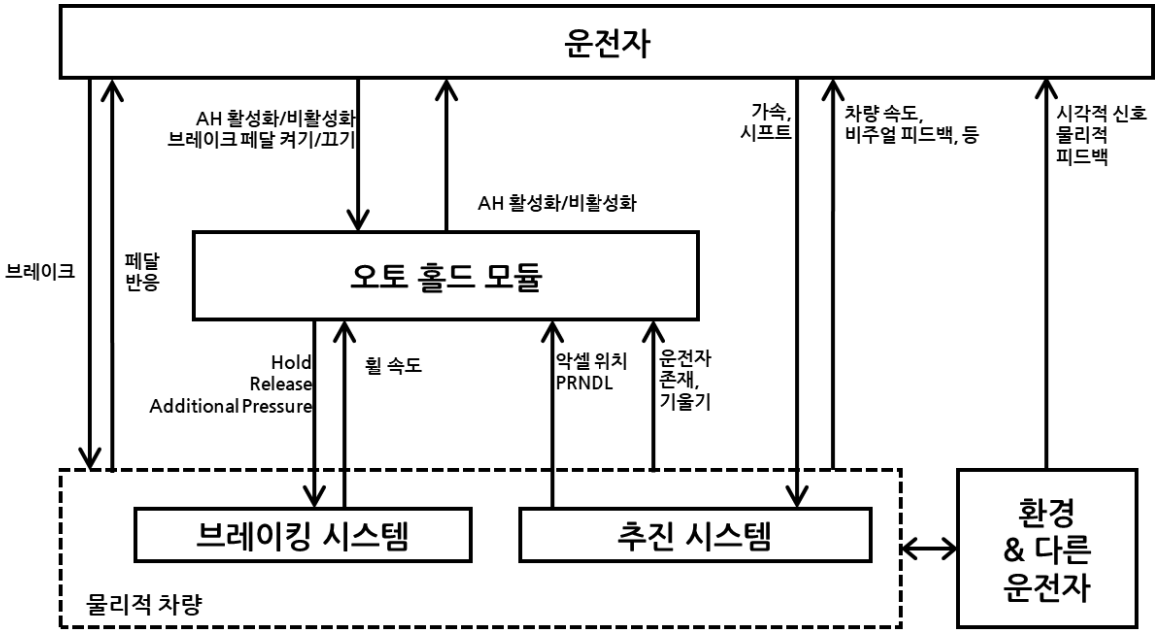


그림 B.7: 자동차 오토 홀드(Auto-Hold) 시스템의 컨트롤 스트럭처

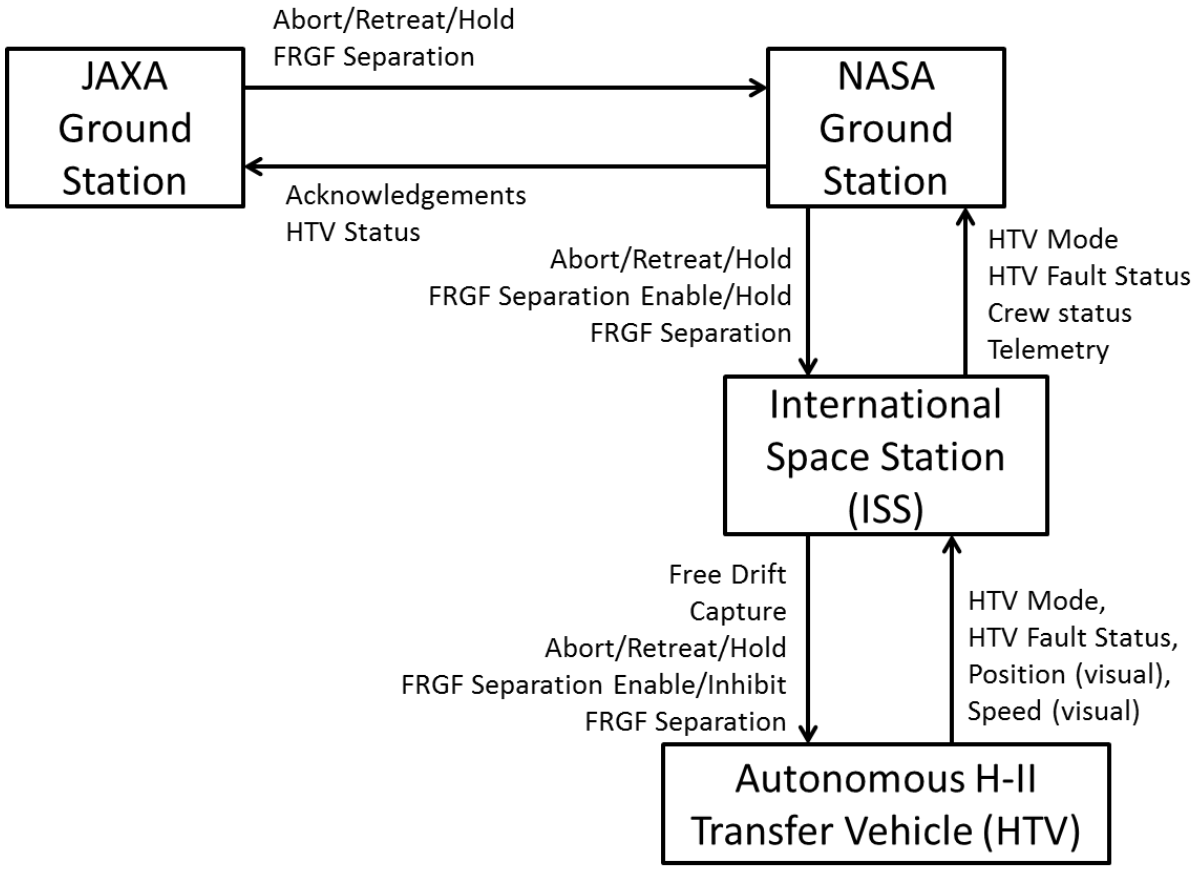


Figure B.8: Control structure for example autonomous space vehicle operations

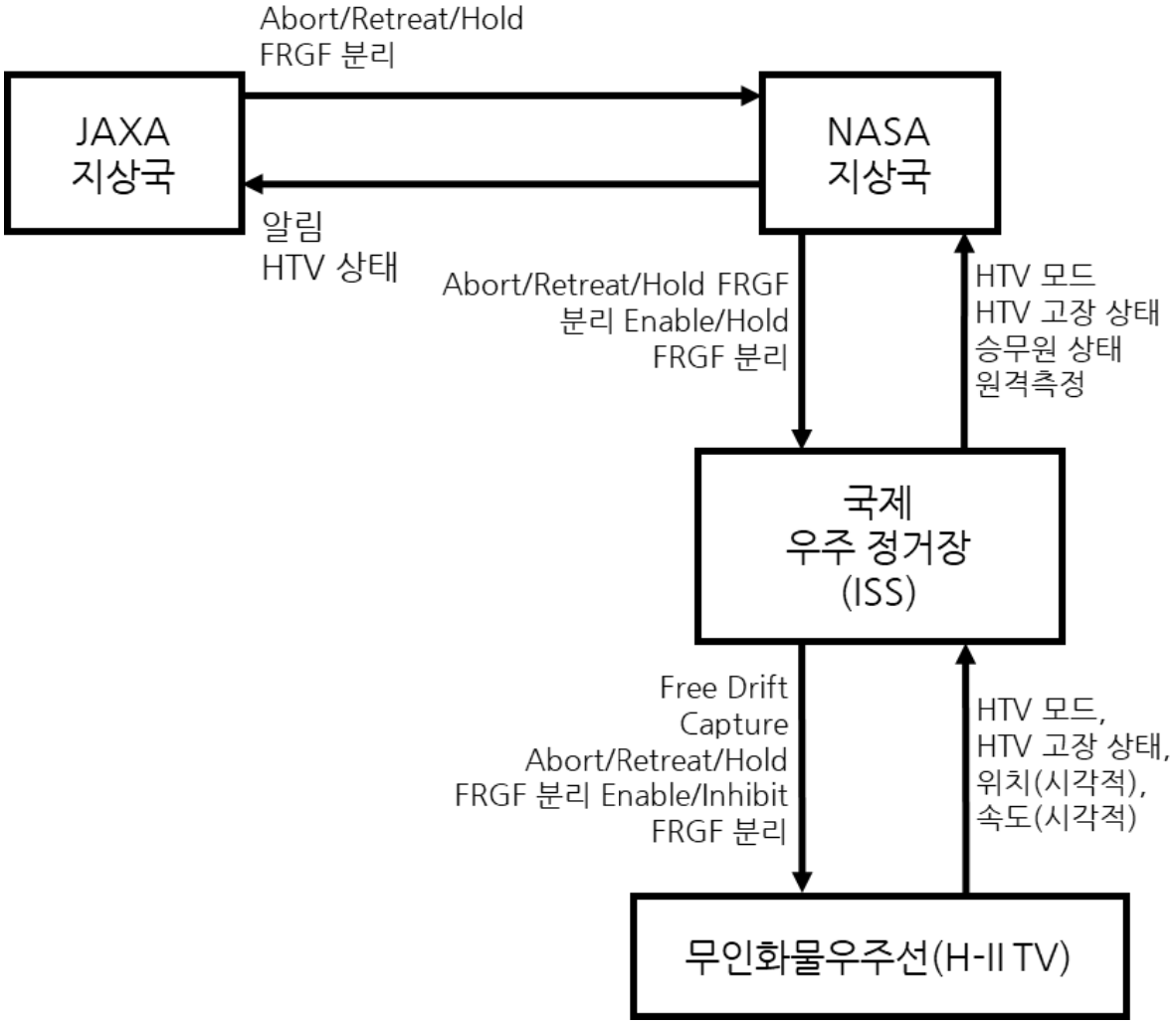


그림 B.8: 무인우주선(Autonomous space vehicle) 동작에 대한 컨트롤 스트럭처 예시

Appendix C: More Examples of UCA Tables

Figure C.1: UCAs for Automotive Auto-Hold System

Control action (by Auto-Hold)	Not providing causes hazard	Providing causes hazard	Incorrect Timing/Order	Stopped Too Soon / Applied too long
Hold Command	UCA-AH-1: AH does not provide HOLD when vehicle stops and brake pedal released [H-1,2]	UCA-AH-2: AH provides HOLD when driver is applying the accelerator [H-1] UCA-AH-3: AH provides HOLD when AH is DISABLED [H-1] UCA-AH-4: AH provides HOLD when vehicle is moving [H-1] UCA-AH-5: AH provides HOLD when driver is not applying brake [H-1,2]	UCA-AH-6: AH provides HOLD too early before the required time at rest has not been met [H-1] UCA-AH-7: AH provides HOLD too late after vehicle stops, begins to roll [H-1]	N/A
Release Command	UCA-AH-10: AH does not provide RELEASE when driver applies accelerator pedal [H-1,2]	UCA-AH-12: AH provides RELEASE when driver is not applying accelerator [H-1,2]	UCA-AH-13: AH provides RELEASE too early before there is sufficient wheel torque [H-1,2] UCA-AH-13: AH provides RELEASE too late after accelerator applied, engine torque exceeds wheel torque [H-1,2]	N/A
Additional Pressure Command	UCA-AH-14: AH does not provide ADDITIONALPRESSURE when vehicle is slipping and AH is active [H-1,2]	UCA-AH-15: AH provides ADDITIONALPRESSURE when AH is not active [H-1,2] UCA-AH-16: AH provides ADDITIONALPRESSURE when brake system specs are exceeded [H-1,2]	UCA-AH-16: AH provides ADDITIONAL-PRESSURE too late after vehicle is slipping [H-1,2]	N/A

부록 C: UCA 테이블의 추가 예

표 C.1: 자동차 오토 홀드(Auto-hold) 시스템에 대한 UCA

컨트롤액션 (오토홀드(Auto-Hold)에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
Hold 명령	UCA-AH-1: 차량 정차 시 브레이크 페달을 놓았을 때 AH가 HOLD를 제공하지 않음 [H-1,2]	UCA-AH-2: 운전자가 가속 페달(accelerator)을 밟을 때 AH가 HOLD를 제공함 [H-1] UCA-AH-3: AH가 DISABLED 상태일 때 HOLD를 제공함 [H-1] UCA-AH-4: 차량이 움직일 때 AH가 HOLD를 제공함 [H-1] UCA-AH-5: 운전자가 브레이크를 밟지 않았을 때 AH가 HOLD를 제공함 [H-1,2]	UCA-AH-6: 정지에 필요한 시간이 충족되기 전에 AH가 HOLD를 너무 일찍 제공함 [H-1] UCA-AH-7: 차량이 정지했다가 움직이기 시작할 때, AH가 HOLD를 너무 늦게 제공함 [H-1]	N/A
Release 명령	UCA-AH-10: 운전자가 가속 페달을 밟았을 때 AH가 RELEASE를 제공하지 않음 [H-1,2]	UCA-AH-12: 운전자가 가속 페달을 밟지 않았을 때 AH가 RELEASE를 적용함 [H-1,2]	UCA-AH-13: 충분한 휠토크가 발생하기 전에 AH가 RELEASE를 너무 일찍 제공함 [H-1,2] UCA-AH-13: 가속 장치가 적용되고 엔진토크가 휠토크를 초과한 이후, AH가 RELEASE를 너무 늦게 제공함 [H-1,2]	N/A
Additional Pressure 명령	UCA-AH-14: AH가 활성화 상태이고 차량이 미끄러질 때 AH가 ADDITIONAL PRESSURE를 제공하지 않음 [H-1,2]	UCA-AH-15: AH가 활성화 상태가 아닐 때 AH가 ADDITIONAL PRESSURE를 제공함 [H-1,2] UCA-AH-16: 브레이크 시스템 사양이 초과되었을 때 AH가 ADDITIONAL PRESSURE를 제공함 [H-1,2]	UCA-AH-16: 차량이 미끄러진 후에 AH가 ADDITIONAL PRESSURE를 너무 늦게 제공함 [H-1,2]	N/A

Table C.2: UCAs for autonomous H-II Transfer Vehicle (HTV) operations

Control Action (from ISS Crew)	Not providing causes hazard	Providing causes hazard	Too Early, Too Late, Order	Stopped Too Soon / Applied too long
Abort	ISS crew does not provide Abort Cmd when emergency condition exists [H-1]	ISS crew provides Abort Cmd when HTV is captured [H-1] ISS crew provides Abort Cmd when ISS is in Abort path [H-1]	ISS crew provides Abort Cmd too late to avoid collision [H-1] ISS crew provides Abort Cmd too early before capture is released [H-1]	N/A
Free Drift	ISS crew does not provide Free Drift Cmd when HTV is stopped in capture box [H-1]	ISS crew provides Free Drift Cmd when HTV is approaching ISS [H-1]	ISS crew provides Free Drift Cmd too late, more than X minutes after HTV stops [H-1] ISS crew provides Free Drift Cmd too early before HTV stops [H-1]	N/A
Capture	ISS crew does not perform Capture when HTV is in capture box in free drift [H-1]	ISS crew performs Capture when HTV is not in free drift [H-1] ISS crew performs Capture when HTV is aborting [H-1] ISS crew performs Capture with excessive/insufficient movement (can impact HTV, cause collision course) [H-1]	ISS crew performs Capture too late, more than X minutes after HTV deactivated [H-1] ISS crew performs Capture too early before HTV deactivated [H-1]	ISS crew continues performing Capture too long after emergency condition exists [H-1]

표 C.2: 무인 우주 화물선(Autonomous HTV(H-II Transfer Vehicle))에 대한 UCA

컨트롤 액션 (우주정거장 직원으로부터)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
Abort	비상 상황 발생 시 우주정거장 승무원이 Abort 명령을 제공하지 않음 [H-1]	무인 우주 화물선이 캡처되었을 때 (captured) 우주정거장 승무원이 Abort 명령을 제공함 [H-1] 우주정거장이 Abort path에 있을 때 우주정거장 승무원이 Abort 명령을 제공함 [H-1]	우주정거장 승무원이 충돌을 피하기 위한 Abort 명령을 너무 늦게 제공함 [H-1] 캡처가 해제(release) 되기 전에 우주정거장 승무원이 Abort 명령을 너무 일찍 제공함 [H-1]	N/A
Free Drift	무인 우주 화물선이 캡처박스(Capture box)에서 멈췄을 때 우주정거장 승무원이 Free Drift 명령을 제공하지 않음 [H-1]	무인 우주 화물선이 우주정거장에 접근하고 있을 때 우주정거장 승무원이 Free Drift 명령을 제공함 [H-1]	무인 우주 화물선이 정지하고 X분 이상이 지난 후 우주정거장 승무원이 Free Drift 명령을 너무 늦게 제공함 [H-1] 무인 우주 화물선이 정지하기 전에 우주정거장 승무원이 Free Drift 명령을 너무 일찍 제공함 [H-1]	N/A
Capture	무인 우주 화물선이 캡처박스에서 프리 드리프트 중, 우주정거장 승무원이 Capture 명령을 수행하지 않음 [H-1]	무인 우주 화물선이 프리 드리프트 중이 아닐 때 우주정거장 승무원이 Capture 명령을 수행함 [H-1] 무인 우주 화물선이 중단(abort)될 때 우주정거장 승무원이 Capture 명령을 수행함 [H-1] 우주정거장 승무원이 과도하거나 불충분하게 Capture 명령을 수행함 (무인 우주 화물선에 영향을 주어 충돌(collision course)을 야기할 수 있음) [H-1]	무인 우주 화물선이 비활성화(deactivated) 되고 X분 이상 경과한 후 우주정거장 승무원이 Capture 명령을 너무 늦게 수행함 [H-1] 무인 우주 화물선이 비활성화(deactivated) 되기 전에 우주정거장 승무원이 너무 일찍 Capture 명령을 수행함 [H-1]	비상 상태가 발생한 후 우주정거장 승무원이 Capture 수행을 너무 오래 지속함 [H-1]

Table C.3: UCAs for Aircraft Flight Crew related to the Wheel Braking System

Control Action (by Flight Crew)	Not providing causes hazard	Providing causes hazard	Too soon, too late, out of order	Stopped too soon, applied too long
CREW.1	CREW.1a1:	CREW.1b1:	CREW.1c1:	CREW.1d1:
Manual braking via brake pedals	Crew does not provide manual braking during landing, RTO, or taxiing when Autobrake is not providing braking or is providing insufficient braking [H4.1]	Crew provides manual braking with insufficient pedal pressure [H4.1]	Crew provides manual braking before touchdown (causes wheel lockup, loss of control, tire burst) [H4.1]	Crew stops providing manual braking command before safe taxi speed (TBD) is reached [H4.1, H4.4]
		CREW.1b2:	CREW.1.c2:	CREW.1d2:
		Crew provides manual braking with excessive pedal pressure (resulting in loss of control, passenger/crew injury, brake overheating, brake fade or tire burst during landing) [H4-1, H4-5	Crew provides manual braking too late (TBD) to avoid collision or conflict with another object (can overload braking capability given aircraft weight, speed, distance to object (conflict), and tarmac conditions)[H4-1, H4-5]	Crew provides manual braking too long (resulting in stopped aircraft on runway or active taxiway) [H4-1]
		CREW.1b3:		
		Crew provides manual braking provided during normal takeoff [H4-2, H4-5]		
CREW.2	CREW.2a1:	CREW.2b1:	CREW.2c1:	
Arm autobrake	Crew does not arm Autobrake before landing (causing loss of automatic brake operation when spoilers deploy. Crew reaction time may lead to overshoot.) [H4-1, H4-5]	Crew does not arm Autobrake to maximum level during takeoff. (assumes that maximum braking force is necessary for rejected takeoff) [H4-2]	Crew provides arm command too late (TBD) (resulting in insufficient time for BSCU to apply brakes) [H4-1, H4-5]	

표 C.3: 휠 브레이킹 시스템(Wheel Braking System)과 관련된 항공기 승무원에 대한 UCA

컨트롤 액션 (항공기 승무원에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
CREW.1 브레이크 페달을 통한 매뉴얼 브레이킹	CREW.1a1: 착륙, 이륙중지(RTO), 지상활주 중 오토브레이크가 브레이킹을 제공하지 않거나 불충분한 브레이킹을 제공할 때, 승무원이 매뉴얼 브레이킹을 제공하지 않음 [H4.1]	CREW.1b1: 승무원이 매뉴얼 브레이킹을 불충분한 페달 압력으로 제공함 [H4.1]	CREW.1c1: 승무원이 터치다운 전에 매뉴얼 브레이킹을 제공함(휠 락업, 제어 불능, 타이어 파열을 유발) [H4.1]	CREW.1d1: 안전한 지상활주 속도(TBD)에 도달하기 전에 승무원이 매뉴얼 브레이킹 명령의 제공을 멈춤 [H4.1, H4.4]
		CREW.1b2: 승무원이 매뉴얼 브레이킹을 과도한 페달 압력으로 제공함(이로 인해, 착륙 중, 제어 불능 승객/승무원 부상, 브레이크 과열, 브레이크 페이드 또는 타이어 파열을 야기) [H4-1, H4-5]	CREW.1c2: 승무원이 다른 물체와의 충돌(collision and conflict)을 피하기 위한 매뉴얼 브레이킹을 너무 늦게(TBD) 제공함 (주어진 항공기 무게, 속도, 물체와의 거리(충돌) 및 타맥(tarmac) 상태에 따라 브레이킹 능력에 과부하를 야기) [H4-1, H4-5]	CREW.1d2: 승무원이 매뉴얼 브레이킹을 너무 오래 제공함 (활주로 또는 사용중인 유도로에서 항공기 정지를 야기함) [H4-1]
		CREW.1b3: 승무원이 정상적인 이륙 중 매뉴얼 브레이킹을 제공함 [H4-2, H4-5]		
CREW.2 오토브레이크 작동개시 (Arm autobrake)	CREW.2a1: 착륙하기 전에 승무원이 오토브레이크를 작동개시(arm)하지 않음 (스포일러를 전개할 때 오토매틱 브레이크 동작 불능을 유발할 수 있음. 승무원의 반응 시간으로 인해 오버슛(overshoot)이 발생할 수 있음) [H4-1, H4-5]	CREW.2b1: 이륙하는 동안 승무원이 오토브레이크를 최대 수준으로 작동개시(arm)하지 않음(이륙중지(RTO) 시 최대치의 브레이킹 파워가 필요하다고 가정) [H4-2]	CREW.2c1: 승무원이 오토브레이크 작동개시(arm) 명령을 너무 늦게(TBD) 제공함(BSCU가 브레이크를 적용할 시간이 불충분함) [H4-1, H4-5]	

	Crew.2a2: Crew does not arm Autobrake prior to takeoff (resulting in insufficient braking during rejected takeoff, assuming that Autobrake is responsible for braking during RTO after crew throttle down) [H4-2]	CREW.2b2: Crew armed autobrake with too high of a deceleration rate for runway conditions (resulting in loss of control and passenger or crew injury). [H4-1, H4-5]		
CREW.3 Disarm Autobrake	CREW.3a1: Crew does not disarm Autobrake during TOGA (resulting in loss of acceleration during (re)takeoff) [H4-1, H4-2, H4-5]	CREW.3b1: Crew disarms Autobrake during landing or RTO (resulting in loss of automatic brake operation when spoilers deploy. Crew reaction time may lead to overshoot) [H4-1, H4-5]	CREW.3c1: Crew disarms Autobrake more than TBD seconds after (a) aircraft descent exceeds TBD fps, (b) visibility is less than TBD ft, (c) etc…, (resulting in either loss of control of aircraft or loss of acceleration during (re)takeoff) [H4-1, H4-2, H4-5]	
CREW.4 Power off BSCU	CREW.4a1: Crew does not power off BSCU in the event of abnormal WBS behavior (needed to enable alternate braking mode) [H4-1, H4-2, H4-5]	CREW.4b1: Crew powers off BSCU when Autobraking is needed and WBS functioning normally [H4-1, H4-5]	CREW.4c1: Crew powers off BSCU too late (TBD) to enable alternate braking mode in the event of abnormal WBS behavior [H4-1, H4-5]	N/A
		CREW.4b2: Crew powers off BSCU when Autobrake is needed (or about to be used) and WBS if funtioning normally [H4-1, H4-5]	CREW.4c2: Crew powers off BSCU too early before Autobrake or Anti-Skid behavior is completed when it is needed [H4-1, H4-5]	

컨트롤 액션 (항공기 승무원에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
	Crew.2a2: 이륙 전에 승무원이 오토브레이크를 작동개시(arm)하지 않음 (승무원이 스로틀을 낮춘 후 이륙중지(RTO) 과정에서 오토브레이크가 브레이킹의 책임을 져야 한다고 가정하면, 이륙중지 시 브레이크가 불충분한 결과를 초래할 수 있음) [H4-2]	CREW.2b2: 승무원이 활주로 컨디션 대비 높은 감속률로 오토브레이크를 작동개시(arm)함 (제어 불능, 승객 혹은 승무원 부상을 야기함) [H4-1, H4-5]		
CREW.3 오토브레이크 작동해제 (Disarm Autobrake)	CREW.3a1: 승무원이 TOGA 중 오토브레이크를 작동해제(disarm) 하지 않음(이륙·재이륙 중 가속 손실을 야기함) [H4-1, H4-2, H4-5] * [역주] TOGA: Take Off and Go Around, (이륙복행)	CREW.3b1: 승무원이 착륙 또는 이륙중지 중 오토브레이크를 작동해제(disarm)함 (스포일러를 전개할 때 오토매틱 브레이크 동작 불능을 유발할 수 있음.승무원의 반응 시간으로 인해 오버슛(overshoot)이 발생할 수 있음) [H4-1, H4-5]	CREW.3c1: 승무원이 아래 (a)~(c) 후 TBD초 이상 지나서 오토브레이크를 작동해제(disarm)함 (a) 항공기 하강이 TBD fps를 초과 (b) 사이가 TBD ft 미만 (c) 기타 등등 (항공기 제어 불능 또는 이륙·재이륙 시 가속 불능을 야기함) [H4-1, H4-2, H4-5]	
CREW.4 BSCU 전원차단 (Power off BSCU)	CREW.4a1: 승무원이 WBS가 비정상적으로 작동하는 경우에 BSCU 전원을 끄지 않음 (대체 브레이킹 모드 활성화가 필요함) [H4-1, H4-2, H4-5]	CREW.4b1: WBS 가능이 정상이고 오토브레이킹이 필요한 상황에서 승무원이 BSCU 전원을 끔 [H4-1, H4-5]	CREW.4c1: WBS가 비정상적으로 작동하는 상황에서 대체 브레이킹 모드를 활성화하기에는 너무 늦은 시점(TBD)에 승무원이 BSCU 전원을 끔 [H4-1, H4-5]	N/A
		CREW.4b2: WBS 가능이 정상이고 오토브레이크가 필요한(또는 곧 사용 예정인) 상황에서 승무원이 BSCU의 전원을 끔 [H4-1, H4-5]	CREW.4c2: 승무원이 오토브레이크 또는 미끄럼 방지가 필요할 때 해당 동작이 완료되기 전에 너무 빨리 BSCU 전원을 끔 [H4-1, H4-5]	

		CREW.4b3: Crew powers off BSCU when Anti-Skid functionality is needed (or will be needed) and WBS is functioning normally [H4-1, H4-5]		
CREW.5 Power on BSCU	CREW.5a1: Crew does not power on BSCU when Normal braking mode, Autobrake, or Anti-Skid is to be used and WBS functioning normally [H4-1, H4-5]		CREW.5c1: Crew powers on BSCU too late after Normal braking mode, Autobrake, or Anti-Skid is needed [H4-1, H4-5]	N/A

컨트롤 액션 (항공기 승무원에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
		CREW.4b3: WBS 가능이 정상이고 미끄럼 방지 기능이 필요할(필요하게 될 예정일) 때, 승무원이 BSCU 전원을 끔 [H4-1, H4-5]		
CREW.5 BSCU 전원공급 (Power on BSCU)	CREW.5a1: WBS 가능이 정상이고 노멀 브레이킹 모드, 오토브레이크 또는 미끄럼 방지가 사용예정일 때 승무원이 BSCU 전원을 켜지 않음 [H4-1, H4-5]		CREW.5c1: 노멀 브레이킹 모드, 오토브레이크 또는 미끄럼 방지가 필요할 때 승무원이 너무 늦게 BSCU 전원을 켬 [H4-1, H4-5]	N/A

Table C.4: UCAs for Aircraft Brake System Control Unit (BSCU) Autobrake

Control Action (by BSCU)	Not providing causes hazard	Providing causes hazard	Too soon, too late, out of order	Stopped too soon, applied too long
BSCU.1 Autobrake Brake command	BSCU.1a1: Autobrake does not provide Brake command during RTO to V1 (resulting in inability to stop within available runway length) [H4-1, H4-5]	BSCU.1b1: Autobrake provides excessive Braking commands during landing roll [H4-1, H4-5]	BSCU.1c1 Autobrake provides Braking command before touchdown (resulting in tire burst, loss of control, injury, other damage) [H4-1, H4-5]	BSCU.1d1 Autobrake stops providing Brake during landing roll before TBD taxi speed attained (causing reduced deceleration)[H4-1, H4-5]
	BSCU.1a2 Autobrake does not provides Brake command during landing roll when BSCU is armed (resulting in insufficient deceleration and potential overshoot) [H4-1, H4-5]	BSCU.1b2 Autobrake provides Braking command inappropriately during takeoff (resulting in inadequate acceleration) [H4-1, H4-2, H4-5]	BSCU.1c2 Autobrake provides Brake command more than TBD seconds after touchdown (resulting in insufficient deceleration and potential loss of control, overshoot) [H4-1, H4-5]	BSCU.1d2 Autobrake provides Brake command too long (more than TBD seconds) during landing roll (causing stop on runway) [H4-1]
		BSCU1b3 Autobrake provides Brake command with insufficient level (resulting in insufficient deceleration during landing roll) [H4-1, H4-5]	BSCU1c3 Autobrake provides Brake command at any time before wheels have left ground and RTO has not been requested (brake might be applied to stop wheels before gear retraction) [H4-1, H4-2, H4-5]	BSCU.1d3 Autobrake provides Brake command for tire lock until less than TBD seconds before touchdown during approach (resulting in loss of control, equipment damage) [H4-1, H4-5]

Table C.4: 항공기 브레이킹 시스템 유닛(BSCU)의 오토브레이크에 대한 UCA

컨트롤 액션 (BSCU에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
BSCU.1 오토브레이크 Brake 명령	BSCU.1a1: 오토브레이크가 이륙중지(RTO)에서 V1까지 Brake 명령을 제공하지 않음 (가능한 활주로 길이 내에서 멈추지 못하는 상황을 초래함)	BSCU.1b1: 오토브레이크가 착륙활주 중 과도한 Brake 명령을 제공함 [H4-1, H4-5]	BSCU.1c1 오토브레이크가 터치다운 전에 Brake 명령을 제공함(타이어 파손, 제어 불능, 부상, 기타 다른 손상을 초래함) [H4-1, H4-5]	BSCU.1d1 오토브레이크가 착륙활주 중 TBD 지상활주 속도에 이르기 전에 Brake 제공을 멈춤 (감속 감소를 야기함) [H4-1, H4-5]
	BSCU.1a2 BSCU가 작동개시(arm)된 상태에서 착륙활주하는 동안 오토브레이크가 Brake 명령을 제공하지 않음 (감속이 충분히 이루어지지 않아 잠재적인 오버슛을 초래함) [H4-1, H4-5]	BSCU.1b2 오토브레이크가 이륙 중 Brake 명령을 부적절하게 제공함 (부적절한 가속을 초래함) [H4-1, H4-2, H4-5]	BSCU.1c2 오토브레이크가 터치다운 후 TBD초 이상 후에 Brake 명령을 제공함 (불충분한 감속, 잠재적인 제어 불능, 오버슛을 초래함) [H4-1, H4-5]	BSCU.1d2 오토브레이크가 착륙활주 중에 너무 오랫동안 (TBD 초 이상) Brake 명령을 제공함(활주로에서 정지를 야기함) [H4-1]
		BSCU.1b3 오토브레이크가 불충분한 수준으로 Brake 명령을 제공함 (착륙활주하는 동안 불충분한 감속을 초래함) [H4-1, H4-5]	BSCU.1c3 바퀴가 지면에서 올라가고 이륙중지가 요청되기 전 모든 시점에, 오토브레이크가 Brake 명령을 제공함(랜딩기어를 접기 전에 바퀴를 멈추기 위해 브레이크가 적용되었을 수 있음) [H4-1, H4-2, H4-5]	BSCU.1d3 항공기 어프로치 과정에서 오토브레이크가 터치다운 전 TBD 초 미만까지 타이어 잠금을 위한 Brake 명령을 제공함 (제어 불능, 장치 손상을 야기함) [H4-1, H4-5]

BCSU.1a4 Autobrake does not provide Brake command after takeoff (needed to lock wheels, results in potential equipment damage during landing gear retraction or wheel rotation in flight) [H4-6]	BSCU.1c4 Autobrake provides Brake more than TBD seconds after V1 during rejected takeoff (assumes that Autobrake is responsible for braking during RTO after crew throttle down) [H4-2]

컨트롤 액션 (BSCU에 의해)	제공하지 않음이 위험을 유발함	제공함이 위험을 유발함	잘못된 시점이나 순서	너무 빨리 중지되거나 너무 오래 적용됨
	BSCU.1a4 오토브레이크가 이륙 후 Brake 명령을 제공하지 않음 (바퀴를 잠그기 위해 필요함. 랜딩 기어 접음 또는 비행 중 바퀴 회전으로 잠재적인 장비 손상을 초래함) [H4-6]		BSCU.1c4 이륙중지 중에 V1을 지난 후 오토브레이크가 TBD초 미만으로 Brake를 제공함 (이륙중지(RTO) 동안 승무원이 스로틀을 낮춘 후에는 오토브레이크가 브레이킹의 책임을 져야 한다고 가정함) [H4-2]	

Appendix D: Responsibilities to be Included in the Safety Control Structure

The list below (from *Engineering a Safer World*, Chapter 13) can be used in creating a new safety control structure, as a checklist for those creating a model of their existing safety control structure or initiating an activity to improve it, and in identifying inadequate controls and control structures when performing incident and accident analysis. It is not meant to be exhaustive and will need to be supplemented for specific industries and safety programs.

This list only contains general responsibilities and does not indicate how they should be assigned. Appropriate assignment of the responsibilities to specific people and places in the organization will depend on the management structure of each organization. Each general responsibility may be separated into multiple individual responsibilities and assigned throughout the safety control structure, with one group actually implementing the responsibilities and others above them supervising, leading or directing, or overseeing the activity. Of course, each responsibility assumes the need for associated authority and accountability, as well as the controls, feedback, and communication channels necessary to implement the responsibility.

General Management

- Provide leadership, oversight, and management of safety at all levels of the organization.
- Create a corporate or organizational safety policy. Establish criteria for evaluating safety-critical decisions and implementing safety controls. Establish distribution channels for the policy. Establish feedback channels to determine whether employees understand it, are following it, and whether it is effective. Update the policy as needed.
- Establish corporate or organizational safety standards and then implement, update, and enforce them. Set minimum requirements for safety engineering in development and operations and oversee the implementation of those requirements, including any contractor activities. Set minimum physical and operational standards for hazardous operations.
- Establish incident and accident investigation standards and ensure recommendations are implemented and effective. Use feedback to improve the standards.
- Establish management of change requirements for evaluating all changes for their impact on safety, including changes in the safety control structure. Audit the safety control structure for unplanned changes and migration toward states of higher risk.
- Create and monitor the organizational safety control structure. Assign responsibility, authority, and accountability for safety.
- Establish working groups.
- Establish robust and reliable communication channels to ensure accurate management risk awareness of the development system design and the state of the operating process. These channels should include contractor activities.
- Provide physical and personnel resources for safety-related activities. Ensure that those performing safety-critical activities have the appropriate skills, knowledge, and physical resources.
- Create an easy-to-use problem reporting system and then monitor it for needed changes and improvements.
- Establish safety education and training for all employees and establish feedback channels to determine whether it is effective along with processes for continual improvement. The education should include reminders of past accidents and causes and input from lessons learned and trouble

부록 D: 안전 컨트롤 스트럭처에 포함될 책임

아래의 내용(*Engineering a Safer World*, 13장에 있음)은 새로운 안전 컨트롤 스트럭처를 만드는 데 사용할 수 있고 기존의 안전 컨트롤 스트럭처 모델을 만들거나 개선 활동을 시작하기 위한 체크리스트로 사용할 수 있으며 사건 및 사고 분석 수행 시 부적절한 컨트롤과 컨트롤 스트럭처를 식별하는 데에도 사용할 수 있다.

이 목록에는 일반적인 책임만 포함되어 있으며 책임을 할당하는 방법을 명시하고 있지는 않다. 조직에서 특정 사람과 지위에 대해 책임을 적절히 할당한다는 것은 각 조직의 관리 구조에 따라 달라진다. 각각의 일반적인 책임은 다수의 개별 책임으로 나누어져 안전 컨트롤 스트럭처 전반에 걸쳐 할당되는데 한 그룹에서 실제로 책임을 이행하고 그 상위의 다른 그룹에서 이를 감독, 지도 또는 지시하거나 활동을 전체적으로 감독한다. 물론, 각 책임에는 책임 이행에 필요한 컨트롤, 피드백, 의사소통 채널뿐만 아니라 관련된 통제권한과 의무가 필요하다고 가정한다.

일반적인 관리(General Management)

- 조직의 모든 수준에서 리더십, 감독, 안전 관리를 제공한다.
- 기업 또는 조직의 안전 정책(safety policy)을 수립한다. 안전-필수 결정을 평가하고 안전 컨트롤을 구현하기 위한 기준을 수립한다. 정책의 배포 채널을 구축한다. 직원이 정책을 이해하고 있는지, 따르고 있는지, 효과적인지를 확인하기 위한 피드백 채널을 구축한다. 필요 시 정책을 업데이트 한다.
- 기업 또는 조직의 안전 표준(safety standard)을 수립하고 이행, 업데이트 및 강제한다. 개발과 운영에서의 안전 엔지니어링에 대한 최소 요구사항을 설정하고, 그러한 요구사항(계약자의 활동 포함)의 이행을 감독한다. 위험한 작업을 위한 최소한의 물리적, 운영 표준을 설정한다.
- 사건 및 사고 조사 표준을 수립하고, 권고사항(recommendation)이 시행되고 효과적인지 확인한다. 피드백을 활용하여 표준을 개선한다.
- 안전에 영향을 미치는 모든 변경사항을 평가하기 위한 변경 요구사항 관리 체계를 수립한다. 변경 요구사항에는 안전 컨트롤 스트럭처의 변경도 포함한다. 안전 컨트롤 스트럭처에 대한 계획되지 않은 변경이나 리스크가 높은 상태로 진입하는 것을 감사(audit)한다.
- 조직의 안전 컨트롤 스트럭처를 만들고 모니터링 한다. 안전을 위한 책임, 통제권한 및 의무를 할당한다.
- 워킹 그룹을 만든다.
- 견고하고 신뢰할 수 있는 의사소통 채널을 구축하여 개발 시스템 설계 및 운영 프로세스 상태의 관리 리스크(management risk)를 정확히 인식할 수 있게 한다. 이러한 채널에는 계약자의 활동이 포함되어야 한다.
- 안전-관련 활동을 위한 물리적, 인적 자원을 제공한다. 안전-필수 활동을 수행하는 사람이 적절한 기술, 지식 및 물리적 자원을 갖추고 있는지 확인한다.
- 사용하기 쉬운 문제 보고 시스템(problem reporting system)을 만들고 필요한 변경사항이나 개선사항이 있는지 모니터링 한다.
- 모든 직원을 대상으로 안전 교육 및 훈련을 실시하고, 교육 및 훈련의 지속적인 개선을 위해 프로세스와 함께 효과적인지를 확인하기 위한 피드백 채널을 구축한다. 교육 내용에는 과거 사고를 상기시키는 것(reminder), 교훈(lessons learned)과 트러블 보고서(trouble reports)로부터 얻은 원

reports. Assessment of effectiveness may include information obtained from knowledge assessments during audits.
- Establish organizational and management structures to ensure that safety-related technical decision making is independent from programmatic considerations, including cost and schedule.
- Establish defined, transparent, and explicit resolution procedures for conflicts between safety-related technical decisions and programmatic considerations. Ensure that the conflict resolution procedures are being used and are effective.
- Ensure that managers who are making safety-related decisions are fully informed and skilled. Establish mechanisms to allow and encourage all employees (including front-line operators) and contractors to contribute to safety-related decision making.
- Establish an assessment and improvement process for safety-related decision making.
- Create and update the organizational safety information system.
- Create and update safety management plans.
- Establish communication channels, resolution processes, and adjudication procedures for employees and contractors to surface complaints and concerns about the safety of the system or parts of the safety control structure that are not functioning appropriately. Evaluate the need for anonymity in reporting concerns.

Development

- Implement special training for developers and development managers in safety-guided design and other necessary skills. Update this training as events occur and more is learned from experience. Create feedback, assessment, and improvement processes for the training.
- Create and maintain the hazard log. Establish and maintain documentation and tracking of hazards and their status.
- Establish working groups.
- Design safety into the system using system hazards and safety constraints. Iterate and refine the design and the safety constraints as the design process proceeds. Ensure the system design includes consideration of how to eliminate or reduce contextual factors that cause or contribute to unsafe operator behavior that, in turn, contributes to system hazards. Distraction, fatigue, etc. are risk factors resulting from design that is dependent on humans performing in a way the designer imagined they would rather than behaving as normal humans would in such situations.
- Document operational assumptions, safety constraints, safety-related design features, operating assumptions, safety-related operational limitations, training and operating instructions, audits and performance assessment requirements, operational procedures, and safety verification and analysis results. Document both what and why, including tracing between safety constraints and the design features to enforce them.
- Perform high-quality and comprehensive hazard analyses to be available and usable when safety-related decisions need to be made, starting with early decision making and continuing through the system's life. Ensure that the hazard analysis results are communicated in a timely manner to those who need them. Establish a communication structure that allows communication downward, upward, and sideways (i.e., among those building subsystems). Ensure that hazard analyses are updated as the design evolves and test experience is acquired.
- Train engineers and managers to use the results of hazard analyses in their decision making.
- Maintain and use hazard logs and hazard analyses as experience is acquired. Ensure communication of safety-related requirements and constraints to everyone involved in development.

인이나 정보를 포함해야 한다. 효과의 평가(assessment)에는 감사 과정에서의 지식 평가 정보가 포함될 수 있다.

- 안전-관련 기술적 의사 결정이 비용과 일정을 포함하는 프로그램의 고려사항과는 독립적임을 보장하기 위한 조직 및 관리 구조를 구축한다.
- 안전-관련 기술적 결정과 프로그램 고려사항 간의 충돌에 대해, 규정되고 명백하며 명시적인 해결 절차를 수립한다. 충돌 해결 절차가 사용되고 있고 효과적인지를 확인한다.
- 안전-관련 결정을 내리는 관리자가 충분한 정보를 가지고 있고 숙련된 사람인지 확인한다. 안전-관련 의사 결정에 모든 직원(일선 운영자 포함) 및 계약자가 참여하도록 독려하기 위한 메커니즘을 수립한다.
- 안전-관련 의사 결정을 위한 평가 및 개선 프로세스를 수립한다.
- 조직의 안전 정보 시스템(SIS)을 만들고 업데이트 한다.
- 안전 관리 계획을 만들고 업데이트 한다.
- 의사소통 채널, 해결 프로세스, 판정(adjudication) 절차를 수립하여 직원 및 계약자가 시스템 안전 또는 적절하게 기능하지 못하는 안전 컨트롤 스트럭처 부분에 대한 불만과 우려를 제기할 수 있도록 한다. 우려 보고에 대해 익명성이 필요한지 평가한다.

개발(Development)

- 개발자와 개발 관리자를 대상으로 안전이 고려된 설계(safety-guided design) 및 다른 필요한 기술에 대한 특별 훈련을 실시한다. 이벤트가 발생하거나 경험을 통해 더 많은 것을 배울 때마다 이 훈련 프로그램을 업데이트 한다.
- 위험 로그를 작성하고 유지관리 한다. 위험과 위험의 상태에 대해 문서화하고 추적성을 수립하여 유지관리 한다.
- 워킹 그룹을 만든다.
- 시스템 위험과 안전 제약사항을 활용하여 시스템에 안전을 설계한다. 설계가 진행됨에 따라 설계와 안전 제약사항을 반복하여 상세화한다. 운영자의 안전하지 않은 행동을 유발하거나, 그러한 행동에 기여하여 결국 시스템 위험을 야기하는 컨텍스트적(contextual) 요인을 제거 또는 감소시키는 방법에 대한 고려사항이 시스템 설계에 포함되어 있음을 보장한다. 특정 상황에서 일반적인 사람들이 행동하는 방식으로 설계하지 않고 설계자가 상상한 사람들의 행동 방식에 의존하여 설계할 때 주의산만(distraction), 피로(fatigue) 등의 리스크 요인이 발생할 수 있다.
- 운영상의 가정, 안전 제약사항, 안전-관련 설계 기능, 동작 가정, 안전-관련 운영상의 제약, 훈련 및 운영 지침, 감사 및 성과 평가 요구사항, 운영 절차, 안전 검증(verification) 및 분석 결과를 문서화한다. 안전 제약사항과 그것을 강제하기 위한 설계 특징 간의 추적성을 포함하여, 무엇(what)과 왜(why)에 대한 사항을 문서화한다.
- 안전-관련 결정이 내려질 때 가용하고 사용할 수 있도록, 초기 의사 결정부터 그리고 시스템 수명주기 전체에서 지속적으로 고품질의 포괄적인 위험 분석을 수행한다. 위험분석 결과가 그것을 필요로 하는 사람들에게 적시에 전달되도록 한다. 의사소통이 아래쪽, 위쪽, 옆쪽(즉, 서브시스템 간에)으로 가능하도록 의사소통 구조를 구축한다. 설계가 진화(evolve)되고 테스트 경험이 축적됨에 따라 위험 분석을 업데이트 한다.
- 엔지니어와 관리자가 위험 분석 결과를 의사 결정에 사용할 수 있도록 훈련한다.
- 경험이 축적됨에 따라 위험로그와 위험분석 결과를 유지관리하고 활용한다. 안전-관련 요구사항과

- Gather lessons learned in operations (including accident and incident reports) and use them to improve the development processes. Use operating experience to identify flaws in the development safety controls and implement improvements.

Operations

- Create an operations safety management plan
- Develop special training for operators and operations management to create needed skills and update this training as events occur and more is learned from experience. Create feedback, assessment, and improvement processes for this training. Train employees to perform their jobs safely, understand proper use of safety equipment, and respond appropriately in an emergency.
- Establish working groups.
- Maintain and use hazard logs and hazard analyses during operations as experience is acquired.
- Ensure all emergency equipment and safety devices are operable at all times during hazardous operations. Before safety-critical, non-routine, potentially hazardous operations are started, inspect all safety equipment to ensure it is operational, including the testing of alarms.
- Perform an in-depth investigation of any operational anomalies, including hazardous conditions (such as water in a tank that will contain chemicals that react to water) or events. Determine why they occurred before any potentially dangerous operations are started or restarted. Provide the training necessary to do this type of investigation and proper feedback channels to management.
- Create management of change procedures and ensure they are being followed. These procedures should include hazard analyses on all proposed changes and approval of all changes related to safety-critical operations. Create and enforce policies about disabling safety-critical equipment.
- Perform safety audits, performance assessments, and inspections using the hazard analysis results as the preconditions for operations and maintenance. Collect data to ensure safety policies and procedures are being followed and that education and training about safety is effective. Establish feedback channels for leading indicators of increasing risk.
- Use the hazard analysis and documentation created during development and passed to operations to identify leading indicators of migration toward states of higher risk. Establish feedback channels to detect the leading indicators and respond appropriately.
- Establish communication channels from operations to development to pass back information about operational experience.
- Perform in-depth incident and accident investigations, including all systemic factors. Assign responsibility for implementing all recommendations. Follow up to determine whether recommendations were fully implemented and effective.
- Perform independent checks of safety-critical activities to ensure they have been done properly.
- Prioritize maintenance for identified safety-critical items. Enforce maintenance schedules.
- Create and enforce policies about disabling safety-critical equipment and making changes to the physical system.
- Create and execute special procedures for the startup of operations in a previously shutdown unit or after maintenance activities.
- Investigate and reduce the frequency of spurious alarms.
- Clearly mark malfunctioning alarms and gauges. In general, establish procedures for communicating information about all current malfunctioning equipment to operators and ensure they are being followed. Eliminate all barriers to reporting malfunctioning equipment.
- Define and communicate safe operating limits for all safety-critical equipment and alarm procedures. Ensure that operators are aware of these limits. Assure that operators are rewarded for

제약사항을 개발과 관련된 모든 사람에게 전달한다.

- 운영 과정에서의 교훈(사고 및 사건 보고서 포함)을 모아서 개발 프로세스를 개선하는 데 사용한다. 운영 경험을 활용하여 안전 컨트롤 개발 시 결함을 식별하고 개선한다.

운영(Operations)

- 운영 안전 관리 계획(operations safety management plan)을 수립한다.
- 운영자와 운영 관리자를 위한 특별 훈련을 개발하여 필요한 기술을 확보하고, 이벤트가 발생하거나 경험을 통해 더 많은 것을 배우게 될 때마다 이를 업데이트한다. 훈련에 대한 피드백, 평가 및 개선 프로세스를 만든다. 직원들이 직무를 안전하게 수행할 수 있고 안전 장비의 올바른 사용법을 이해할 수 있으며 비상 시 적절하게 대응할 수 있도록 훈련시킨다.
- 워킹 그룹을 만든다.
- 경험이 축적됨에 따라 운영과정에서의 위험로그와 위험분석 결과를 유지관리하고 활용한다.
- 위험한 작업 시 모든 비상 장비(emergency equipment)와 안전 장치가 항상 작동할 수 있도록 한다. 안전-필수적, 비일상적, 잠재적으로 위험한 작업을 시작하기 전에 알람 테스트를 포함하여 모든 안전 장비가 작동하는지를 검사한다.
- 위험한 조건(예: 물에 반응하는 화학 물질이 있는 탱크 내의 물) 또는 이벤트와 같은 모든 운영 이상상황(anomalies)에 대해 심층적인 조사를 수행한다. 잠재적으로 위험한 운영이 시작되거나 재시작되기 전에 운영 이상상황이 발생한 이유를 조사한다. 이러한 유형의 조사 및 적절한 피드백 채널 관리에 필요한 훈련을 제공한다.
- 변경 절차에 대한 관리 체계를 구축하고 지켜지도록 한다. 변경 절차에는 제안된 모든 변경에 대한 위험분석과 안전-필수 동작에 관련된 모든 변경사항의 승인이 포함되어야 한다. 사용하지 못하게 하는 안전-필수 장비(역주: 변경 이후에는 장비의 사용 제한이 필요할 수 있음)에 대한 정책을 만들고 강제한다.
- 운영 및 유지보수의 전제 조건으로서의 위험분석 결과를 활용하여 안전 감사, 성능 평가 및 검사(inspection)를 수행한다. 안전 정책과 절차가 준수되고 있는지, 안전에 대한 교육과 훈련이 효과적인지를 확인하기 위한 데이터를 수집한다. 증가하는 위험에 대한 선행지표 피드백 채널을 구축한다.
- 개발 과정에서 생성되어 운영에 전달된 위험분석 결과와 문서를 사용하여 높은 리스크 상태로의 전환에 대한 선행지표를 식별한다. 선행지표를 탐지하고 적절하게 대응하기 위한 피드백 채널을 구축한다.
- 운영에서 개발로의 피드백 채널을 구축하여 운영 경험의 정보를 역으로 전달한다.
- 사건 및 사고 조사 시 모든 시스템적 요인을 포함하여 심층적으로 조사한다. 모든 권고사항의 시행을 위해 책임을 할당한다. 권고사항이 모두 시행되었고 효과적인지를 확인하기 위한 후속조치를 수행한다.
- 안전-필수 활동에 대한 독립적인 검사(check)를 수행하여 안전-필수 활동이 적절하게 수행되었는지 확인한다.
- 식별된 안전-필수 항목에 대한 유지보수 우선순위를 지정한다. 일정에 따라 유지보수를 강제한다.
- 사용하지 못하게 하는 안전-필수 장비 및 물리적 시스템의 변경에 대한 정책을 만들고 강제한다.
- 종료된(shutdown) 유닛의 작업시작 또는 유지보수 이후의 작업시작을 위한 특별 절차를 만들고 실행한다.

following the limits and emergency procedures, even when it turns out no emergency existed. Provide for tuning the operating limits and alarm procedures over time as required.
- Ensure that spare safety-critical items are in stock or can be acquired quickly.
- Establish communication channels to plant management about all events and activities that are safety-related. Ensure management has the information and risk awareness they need to make safe decisions about operations.
- Ensure emergency equipment and response is available and operable to treat injured workers.
- Establish communication channels to the community to provide information about hazards and necessary contingency actions and emergency response requirements.

- 오작동(spurious) 알람의 빈도를 조사하여 감소시킨다.
- 오작동 알람과 게이지를 명확히 표시한다. 일반적으로, 현재 오작동 중인 모든 장비에 대한 정보를 운영자에게 전달하는 절차를 수립하고 그 절차가 준수되는지 확인한다. 오작동 중인 장비에 대한 보고를 방해하는 모든 장애물을 제거한다.
- 모든 안전-필수 장비 및 알람 절차에 대한 안전 운영의 한계를 정의하고 전달한다. 운영자가 이러한 한계를 알고 있도록 한다. 비상 상황이 아니라고 판명되는 경우라도 운영자가 한계 및 비상 절차(emergency procedure)를 따르면 보상받을 수 있도록 보장한다. 시간이 지난 후 필요한 경우 운영 한계 및 알람 절차를 조정하도록 한다.
- 여분의 안전-필수 물품을 구비하거나 신속하게 구비될 수 있도록 한다.
- 안전-관련 모든 이벤트 및 활동에 대해 공장 관리자와의 의사소통 채널을 구축한다. 관리자가 운영에 대한 안전한 결정을 하는 데 필요한 정보 및 리스크를 인식하도록 한다.
- 부상당한 작업자를 치료하기 위한 비상 장비 및 대응이 가용하고 운영 가능하도록 한다.
- 커뮤니티에 의사소통 채널을 구축하여 위험, 필수적인 우발상황 조치(contingency actions), 비상 대응(emergency response) 요구사항에 대한 정보를 제공한다.

Appendix E: Limitations in Analytic Decomposition for Software-Intensive Systems (an Avionics Example)

Classic approaches to ensuring a property in a system apply analytic decomposition, that is, the individual components are analyzed and the results are combined to ensure the property exists in the system as a whole. This approach works for systems where the coupling and interactions among components are relatively simple, an assumption that primarily holds for electro-mechanical systems.[36] For software-intensive systems, the assumption underlying this approach about independence and reducibility (e.g., the components do not have indirect or unintended impact on each other) do not hold. The aircraft industry is starting to discover this fact. A paper by Bartley and Lingberg[37] describes the problem and provides examples of occurrences on aircraft in flight. They discuss the problem in the context of Integrated Modular Avionics (IMA) systems, which is a set of aircraft avionics functions that may be created by different manufacturers and updated individually over time.

To allow shared information in an IMA system, interfaces are typically specified in an Interface Control Document (ICD) for electrical interactions, mechanical or physical interactions, and software module interactions. A typical software interface specification includes things such as digital signal characteristics; data transmission format, coding, timing, and updating requirements; data and data element definition, message structure and flow, operational sequence of events; and error detection and recovery procedures. The ICD does not explicitly prescribe the underlying logic or behavior that actually generates the variables that should go back onto the interface.

An assumption underlying the use of the ICD is that data and control coupling between components are minimized and that if a change to one function does not require a change to the ICD, then the functions using those parameters cannot be impacted by the change. This assumption is false, as shown in an example involving flaps control provided by Bartley and Lingberg.

Figure E.1 shows the interface between the flaps system controller in which a discrete variable of "flaps extended" is passed between supposedly independent avionics functions. Without further definition of the exact technical meaning of flaps extended, a true state of the flaps-extended discrete variable could possibly mean any of the following:

- Both left and right trailing-edge (TE) flap surfaces are detected in the 1 or greater flap detent.[38]
- Both left and right trailing-edge flap surfaces are not detected in the "up" flap detent.
- The flap lever handle is detected in the 1 or greater flap handle detent.
- The flap lever handle is not detected in the up-flap handle detent.

All four of these possibilities have differing characteristics from each other, and different receiving functions might use the flaps-extended discrete in a different way. For example, once the trailing-edge flaps begin to move, the logic that determines "flaps not in the up position" will be satisfied almost immediately. In contrast, the flaps may take 5 to 10 seconds to fully reach the "flaps detected in the '1' detent" position. Furthermore, if the flap surfaces will not respond to a valid command due to a hydraulic system failure, the flap lever position will no longer reflect the true position of the flap surfaces once the lever is moved out of the up detent. Obviously, how a signal is computed needs to be

[36] Even for these systems, results from combining component analyses can be incorrect when subtle indirect interactions among components distort the combined component interactions.

[37] Gregg Bartley and Barbara Lingberg, Certification Concerns of Integrated Modular Avionics (IMA) Systems, Federal Aviation Administration, Washington D.C.

[38] A detent is a device used to mechanically resist or arrest the rotation of a wheel, axle, or spindle.

부록 E: 소프트웨어 집약적인 시스템에 대한 분석적 분해의 한계(항공전자 사례)

시스템의 속성을 보장하기 위한 고전적인 방법은 분석적 분해(analytic decomposition)를 적용하는 것이다. 즉, 개별 컴포넌트를 분석한 후 그 결과를 통합하여 속성이 시스템에 전체로서 존재함을 보장하는 것이다. 이 방법은 컴포넌트 간의 결합도(coupling) 및 상호작용이 비교적 단순한 시스템에 적용되는데 전자-기계(electro-mechanical) 시스템[36]에 주로 적용되는 가정이다. 하지만 소프트웨어 집약적인 시스템의 경우에는 독립성 및 환원성(reducibility)에 대한 가정(예: 컴포넌트가 서로 간에 간접적 또는 의도치 않은 영향이 없음)이 적용되지 않는다. 항공기 산업에서 이러한 사실이 발견되기 시작했다. 바틀리(Bartley)와 링버그(Lingberg)의 논문[37]에서는 이러한 문제를 언급하고 비행중인 항공기에서의 사례를 제시하고 있다. 그들은 통합 모듈형 항공전자(Integrated Modular Avionics, IMA) 시스템에서의 문제점을 논의한다. IMA 시스템이란 서로 다른 제조사에서 만들 수 있고 시간이 지나면서 개별적으로 업데이트 될 수 있는 항공전자 기능의 집합(set)을 말한다(역주: 항공기에 탑재되는 시스템의 효율적인 업그레이드가 가능하도록 규격에 기반하여 시스템을 모듈화하여 개발한 것).

IMA 시스템에서 정보를 공유하기 위한 인터페이스는 전형적으로 전기적 상호작용, 기계적 또는 물리적 상호작용, 소프트웨어 모듈 상호작용이 기술된 인터페이스 컨트롤 문서(Interface Control Document, ICD)에 명시된다. 전형적인 소프트웨어 인터페이스 명세에는 디지털 신호 특성; 데이터 전송 형식, 코딩, 타이밍, 업데이트 요구사항; 데이터 및 데이터 요소의 정의, 메시지 구조 및 흐름, 이벤트의 작동 순서; 오류 탐지 및 복구 절차 등이 포함된다. 인터페이스로 반환되어야 하는 변수를 실제로 생성하는 기본적인 로직이나 동작이 ICD에 명확하게 기술되지는 않는다.

ICD를 사용하는 데 내재된 가정은 첫째로, 컴포넌트 간의 데이터 및 컨트롤 커플링이 최소화되었고 둘째로, 한 개의 기능 변경이 ICD의 변경을 요구하지 않는다면 해당 파라미터를 사용하는 기능은 변경에 영향을 받지 않는다는 것이다. 이 가정은 잘못된 것이며 이는 바틀리와 링버그의 플랩(flap) 컨트롤 예시에서 볼 수 있다.

그림 E.1은 플랩 시스템 컨트롤러간의 인터페이스를 보여주는데 여기서 독립적인 것처럼 보이는 항공전자 기능 간에 "플랩 전개(flaps-extended)"이라는 개별 변수가 전송된다. 보다 정확한 플랩 전개의 기술적 정의는 차치하고 플랩 전개 개별 변수의 실제 상태는 다음 중 하나를 의미할 수 있다.

- 왼쪽·오른쪽 모두에서, 뒷전(trailing edge) 플랩 표면이 1개 이상의 플랩 디텐트[38]에 검출됨
- 왼쪽·오른쪽 모두에서, 뒷전 플랩 표면이 "up" 플랩 디텐트에 검출되지 않음
- 1개 이상의 플랩 핸들 디텐트에서 플랩 레버 핸들이 검출됨
- up-플랩 핸들 디텐트에서 플랩 레버 핸들이 검출되지 않음

이 4가지 상태들은 서로 다른 특성을 가지며 다른 수신 기능이 서로 다른 방식으로 플랩 전개라는 개별 변수를 사용할 수 있다. 예를 들어 뒷전 플랩이 움직이기 시작하면 "up 포지션이 아닌 플랩"을 결정하는 로직은 거의 즉시 만족될 것이다. 반면, 플랩이 1 디텐트 위치에 대응되는 플랩 각까지("flaps detected in the '1' detent" position) 전개되는 데 5~10초가 걸릴 수 있다. 또한 유압 시스템 고장으

36 이러한 시스템에서도 컴포넌트의 결과를 합쳐서 얻은 결과는, 컴포넌트 간의 미묘한 간접적 상호작용이 컴포넌트의 상호작용을 왜곡하게 되는 경우에는 부정확할 수 있다.

37 Gregg Bartley and Barbara Lingberg, Certification Concerns of Integrated Modular Avionics(IMA) Systems, Federal Aviation Administration, Washington D.C.

38 디텐트(detent)는 바퀴, 차축 또는 스핀들의 회전을 기계적으로 저지하거나 정지시키는 데 사용되는 장치임

understood by the designers of any function that uses that particular signal. Differences such as those described may be of critical importance to using the system.

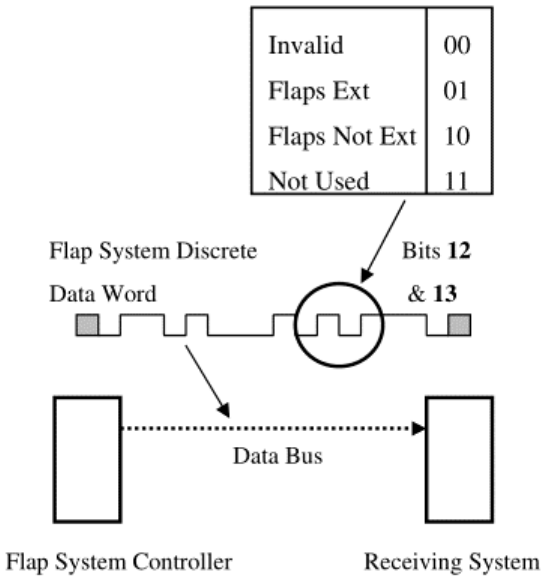


Figure E.1: Flaps-extended (Ext) discrete data transmission [3].

If only the information in Figure E.1 is communicated via the data bus, then a change in the logic of the flap system controller may negatively impact the behavior of the function using the flaps-extended variable. The receiving system(s) developers will thus not be privy to the change, and the system-level impact will not be understood. For example, a problem discovered later in the development cycle, after the design of the individual applications have been ”finalized,” may lead to a change in the logic of the flap system controller.

One potential method of addressing this problem with the flap alert logic is to compute the flaps-extended variable using the flap lever position instead of the actual flap surface position. The physical meaning of the flaps-extended variable has changed.

According to Bartley and Lingberg:

> The point illustrated is that it cannot not [sic] be left to the designers and system experts of the function being updated (in this example, the flap system) to determine the effect of that change on downstream users. The experts that are required to assess the impact of the change are the designers and analysts of the using system, not the source system. A Change Impact Analysis that crosses functional boundaries is clearly required for this example, even though the ICD for the source function did not change. Additionally, this example illustrates why robust partitioning does not totally insulate functions residing in one partition from changes to a function in a different partition.

Traditional hazard analysis techniques such as FMEA and fault tree analysis focus on individual component failure or faulted modes. This focus on component reliability assumes that robust partitioning and interface control are valid and that components do not interact either directly or indirectly. However, as the example illustrates, these assumptions usually do not hold for complex systems.

The problem can be theoretically explained and solved using STPA.

로 플랩 표면이 유효한 명령에 응답하지 않고 레버가 up 디텐트를 벗어나게 되면 플랩 레버 포지션은 더 이상 플랩 표면의 실제 위치를 반영하지 않게 된다. 특정 시그널을 사용하는 모든 기능의 설계자는 그 시그널이 계산되는 방법을 명백히 이해해야 한다. 앞서 설명된 차이점은 시스템 사용에 매우 중요할 것이다.

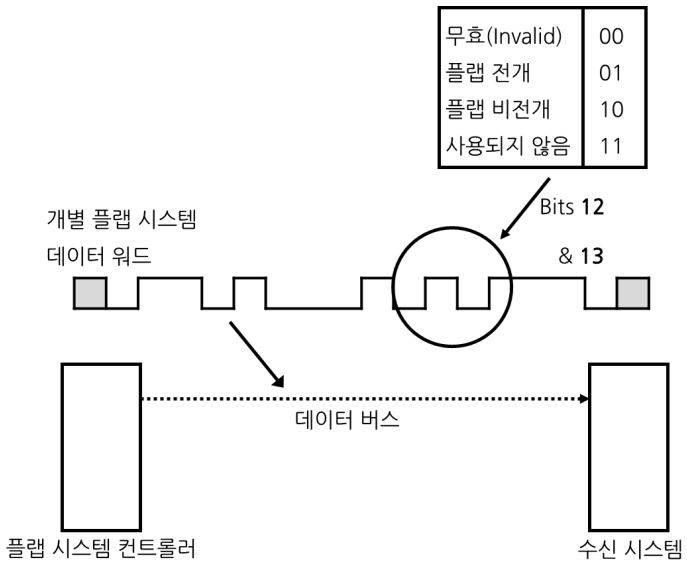


그림 E.1: 플랩-전개(Ext) 개별 변수 데이터 전송 [3].

그림 E.1에서 만약 정보(역주-예:00,01,10,11)만이 데이터 버스를 통해 전달된다고 할 때 플랩 시스템 컨트롤러의 로직을 변경하는 것이 플랩 전개 변수를 사용하는 함수의 동작에 부정적인 영향을 줄 수 있다. 즉, 플랩 전개 변수를 사용하는 수신 시스템을 개발하는 사람은 변경을 신경쓰지 않을 것이며 시스템 수준의 영향을 이해하지 못할 것이다. 예를 들어, 각 어플리케이션의 설계가 "완료된", 개발 사이클 후반에 발견된 문제는 플랩 시스템 컨트롤러의 로직을 변경시킬 수 있다.

플랩 경고(alert) 로직을 통해 이러한 문제를 해결하는 한 가지 방법은 실제 플랩 표면의 포지션 대신 플랩 레버의 위치를 사용하여 플랩 전개 변수를 계산하는 것이다. 플랩 전개 변수의 물리적 의미는 아래와 같이 변경되었다.

바틀리와 링버그에 따르면:

> 그림에서의 요점은, 업데이트되는 기능(이 예에서는 플랩시스템)의 설계자와 시스템 전문가는 그 변경이 하위 사용자에게 미치는 영향을 판단할 수 없다는 것이다. 변경의 영향을 평가해야 하는 전문가는 변경 전의 원래 시스템이 아닌 사용하는 시스템의 설계자와 분석가이다. 이 예에서, 원래 함수에 대한 ICD가 변경되지 않았지만, 함수의 경계를 넘는 변경 영향 분석(Change Impact Analysis)이 분명히 필요하다. 또한 이 예시는 견고한 파티셔닝을 통해서도 한쪽 파티션에 있는 함수가 다른 쪽 파티션에 있는 함수의 변경으로부터 완전히 분리될 수 없는 이유를 보여준다.

FMEA, FTA와 같은 전통적인 위험 분석 기법은 개별 컴포넌트의 고장 또는 고장 모드에 초점을 맞추고 있다. 이러한 컴포넌트 신뢰성에 대한 초점은, 견고한 파티셔닝과 인터페이스 컨트롤이 유효하고 컴포넌트가 직간접적으로 상호작용하지 않는다고 가정한 것이다. 그러나 예시에서 보는 바와 같이 이러한 가정은 복잡한 시스템에는 일반적으로 성립되지 않는다.

이 문제는 STPA를 사용하여 이론적으로 설명하고 해결할 수 있다.

Appendix F: Basic Engineering and System Engineering Concepts for Non-Engineers

In reviews of an early draft of this handbook, we discovered that we had assumed some engineering background that not all readers had: Users of STPA are not necessarily trained in engineering. This appendix provides an introduction to basic engineering concepts that may be unfamiliar to some users of this handbook.

I have attempted to write the sections in this appendix to be independent so that readers can pick and choose sections that interest them. No assumption is made about any particular educational background in covering these topics, which include:

- What is a "system"?
- Basic systems engineering concepts and terminology
- The concept of "control" in engineering
- Systems theory vs. complexity theory

What is a "System"?

The most basic concept underlying everything else in this handbook is that of a *system*.

> *System*: A set of things (referred to as system components) that act together as a whole to achieve some common goal, objective, or end.

Some definitions of a system leave out the "goal" or "objective" and state essentially that a system is a connected set of things or parts forming a unified whole. This definition does not capture the way the term "system" is usually used. A shoe, a star, and a locomotive are a set of things or potential parts of a system but most people would not normally consider this to be a system. It could be considered as a system if a purpose could be conceived for considering these individual things together; the purpose is basic to the concept.

Other definitions state that the components of the system must be interdependent or connected or interacting, but none of these conditions are really necessary to have a system and constrain the definition in a way that excludes things that usually are considered to be systems. The system components may be either directly or indirectly connected to each other, with the latter including connections involving the system purpose only and various types of non-linear interdependencies.

The consequence of this definition is that a goal or objective for the system is fundamental. But there may be different objectives for those defining and viewing systems. An airport is used throughout this section to illustrate various points. To a traveler, the purpose of an airport may be to provide air transportation to other locales. To local or state government, an airport may be a means to increase government revenue and economic activity in the area of the airport. To the airlines, the purpose of an airport may be to take on and discharge passengers and cargo. To the businesses at the airport, the purpose is to attract customers to whom they can sell products and services. Therefore, when talking about a system, it is always necessary to specify the purpose of the system that is being considered.

A system is an abstraction, that is, a model conceived by the viewer.

The observer may see a different system purpose than the designer or focus on different relevant properties. Specifications, which include the purpose of the system, are critical in system engineering. They ensure consistency of mental models among those designing, using, or viewing a system, and they enhance communication. Notice that different components of the airport may be included in a

부록 F: 비-엔지니어를 위한 기본적인 공학 및 시스템 공학 개념

이 핸드북의 초안을 검토하면서 일부 공학적 배경지식은 모든 독자가 가진 것은 아니라고 전제했음을 알게 되었다: STPA 사용자가 반드시 공학 교육을 받아야 하는 것은 아니다. 이 부록에서는 일부 사용자에게는 익숙하지 않을 수도 있는 기본적인 공학 개념에 대해 소개한다.

독자가 관심 있어 하는 섹션을 선택할 수 있도록 이 부록의 섹션을 독립적으로 작성했다. 다음과 같은 주제를 커버하기 위한 특정 교육 배경에 대해서는 어떠한 가정도 하지 않는다:

- "시스템"이란 무엇인가?
- 기본 시스템 공학 개념 및 용어
- 공학에서의 "컨트롤"의 개념
- 시스템 이론 vs 복잡도 이론

"시스템"은 무엇인가?

이 핸드북에서 다루는 모든 것의 기반이 되는 기본 개념은 *시스템*의 개념이다.

> *시스템*: 공통된 목표, 목적 또는 결과를 달성하기 위해 전체로써 함께 동작하는 일련의 요소(시스템 컴포넌트라고 함)

일부의 시스템 정의에서는 "목표" 또는 "목적"을 생략한 채, 통합된 전체를 형성하는 연결된 일련의 사물 또는 부품이라고 시스템을 정의하기도 한다. 이러한 정의는 "시스템"이라는 용어가 일반적으로 사용되는 의미를 뜻하지 않는다. 하나의 신발, 별, 기관차는 시스템의 일부 또는 잠재적 부분이지만 대부분의 사람들은 이를 시스템으로 생각하지 않는다. 이러한 개별적 사물들을 모두 함께 고려하기 위한 목적이 있다면 이는 시스템으로 간주할 수 있다; 목적은 시스템 개념의 기본이다.

다른 정의에서는 시스템의 컴포넌트가 상호 의존적이거나, 연결되거나, 상호작용해야 한다고 명시하지만 이 중 어느 것도 시스템이 필수적으로 갖추어야 하는 것은 아니며, 일반적으로 시스템이라고 여겨지는 것을 제외하는 방식으로 시스템 정의를 제한할 필요는 없다. 시스템 컴포넌트는 직접 또는 간접적으로 서로 연결될 수 있는데 후자는 시스템 목적만으로 연결된 경우나 다양한 유형의 비선형 상호 의존성과 관련하여 연결된 경우를 포함한다.

시스템의 정의에서 중요한 것은 시스템의 목표나 목적이 필수적(fundamental)이라는 것이다. 그러나 시스템을 정의하거나 바라보는 사람에 따라 목적이 다를 수 있다. 이 섹션에서는 공항을 예제로 하여 다양한 포인트를 설명한다. 여행자에게 있어 공항의 목적은 다른 지역으로 항공 교통편을 제공하는 것이다. 지방 정부 또는 주 정부에 있어 공항은 공항 지역의 정부 수입과 경제 활동을 증대시키는 수단일 것이다. 항공사에게 공항의 목적은 승객과 화물을 인계하는 것이다. 비즈니스 측면에서의 공항의 목적은 제품과 서비스를 판매할 수 있는 고객을 확보하는 것이다. 그러므로 시스템에 대해 말할 때에는 항상 고려하는 시스템의 목적을 명시해야 한다.

> *시스템은 추상적인 것, 즉, 관찰자(viewer)가 생각하는 모델이다.*

관찰자는 시스템의 목적을 설계자와는 다르게 보거나 관련된 다른 속성에 초점을 둘 수 있다. 시스템의 목적을 포함하는 명세(specification)는 시스템 엔지니어링에서 매우 중요하다. 그것은 시스템을 설계,

particular “airport” system, such as passenger check-in counters, ramps to the planes, and taxiways in the airline view of an airport versus shops and customers from the commercial view. Note again that these are models or abstractions laid upon the actual physical world by human minds. The components that are considered in any “airport system” or subsystem and the role they play in the system as a whole may be different for each concept (model) of an airport system or of airport subsystems. More on this topic can be found in the next section on the importance of specifications in systems engineering. The basic idea here is that the purpose or goals of the system being considered must be specified and agreed upon by those modeling and analyzing a particular system and that these aspects of systems are abstractions or models imposed by the viewer on the real world objects.

For engineered (man-made) systems, the purpose is usually specified before creating the system although observers of the system may interpret the purpose differently than originally intended by the designers. For natural systems, behavior may be interpreted as purposeful by the observers of the system.

There are some basic assumptions underlying the concept of a system:

- The system goals can be defined
- Systems are atomistic, that is, they can be separated into components with interactive behavior or relationships between the components whether the interactions are direct or indirect.

Systems themselves may be part of a larger system or be divisible into subsystems (viewed as components of the overall system). Note that the definition here is what is called “recursive” in mathematics and computer science. That is, the definition is made in terms of itself. A system is usually part of a larger system and it can be divisible into subsystems. Figure F.1 shows a typical abstraction hierarchy for a system labeled A (for example the system “airport”) and for three subsystems, which in turn can be conceived as systems themselves.

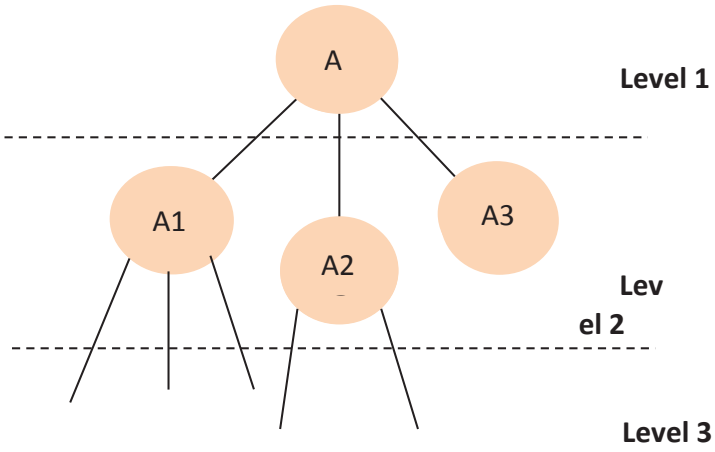


Figure F.1: The abstraction System A made be viewed as composed of three subsystems. Each subsystem is itself a system.

Figure F.1 is not a connection diagram. Rather it is a hierarchical abstraction that shows the different views of a system at different levels of abstraction or modeling. Figure F.2 shows an abstraction where A itself is conceived as part of a larger system AB.

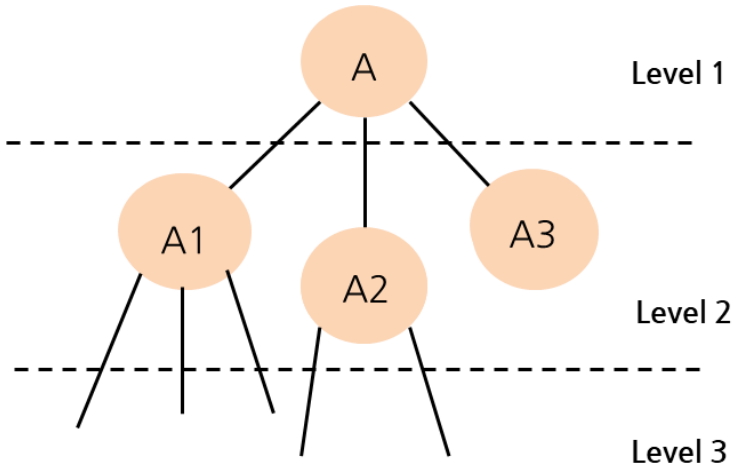
사용 또는 관찰하는 사람들 사이에서 멘탈 모델의 일관성을 보장하고 커뮤니케이션을 향상시킨다. 특정 "공항" 시스템에는 공항의 다른 컴포넌트가 포함될 수 있다. 예를 들면, 공항 관점에서 볼 때는 승객 체크인 카운터, 비행기로의 램프(ramp), 항공사 유도로가 포함될 수 있고 상업적 관점에서 볼 때는 상점과 고객이 포함될 수 있다. 이러한 것들은 실제 물리적 세계를 대상으로 사람의 마음이 만들어낸 모델 또는 추상이다. 모든 "공항 시스템" 또는 공항 서브시스템에서 고려되는 컴포넌트들과, 시스템 전체에서 그 컴포넌트들이 수행하는 역할은 공항 시스템 또는 공항 서브시스템의 각 개념(모델)에 따라 다를 수 있다. 이 주제에 대한 더 자세한 내용은 시스템 엔지니어링에서의 명세 중요성을 다루는 다음 섹션에서 확인할 수 있다. 여기서의 기본적인 아이디어는 고려하고자 하는 시스템의 목적이나 목표가 시스템을 모델링하고 분석하는 사람들 간 명세되고 동의되어야 한다는 것이며 시스템의 이러한 측면은 실제 세계의 오브젝트에 대해 사람들이 보는 추상화나 모델인 것이다.

엔지니어링 된(사람이 만든) 시스템의 경우, 일반적으로 시스템을 만들기 전에 목적을 명세하는데 물론 시스템을 바라보는 사람이 원래 설계자의 의도와는 다르게 목적을 해석할 수도 있다. 자연의 시스템의 경우에는 시스템을 바라보는 사람이 시스템의 동작을 의도대로 해석할 수 있다.

시스템 개념의 기본 가정은 다음과 같다:

- 시스템 목표를 정의할 수 있다.
- 시스템은 원자성을 가진다(atomistic). 즉, 시스템은 컴포넌트로 나누어질 수 있으며 컴포넌트는 컴포넌트 간 상호작용하는 동작이나 관계를 가진다. 여기서의 상호작용은 직접적 또는 간접적일 수 있다.

시스템은 더 큰 시스템의 부분이거나 서브시스템(전체 시스템의 컴포넌트로 간주되는)으로 나눌 수 있다. 이 정의는 수학이나 컴퓨터 과학에서 "재귀적(recursive)"이라고 불리는 것이다. 즉, 정의는 그 자체로 이루어진다. 시스템은 일반적으로 더 큰 시스템의 부분이며 서브시스템으로 나눌 수 있다. 그림 F.1은 A라는 시스템(예: "공항" 시스템)의 세 가지 서브시스템에 대한 전형적인 추상화 계층 구조를 나타내며, 서브시스템들은 그 자체로 시스템으로 볼 수 있다.



그림F.1: 세 개의 서브시스템으로 구성된 추상화 시스템 A.
각 서브시스템이 하나의 시스템임

그림 F.1은 연결을 나타낸 다이어그램(connection diagram)이 아니며 계층적 추상화를 나타낸 것으로 추상화 또는 모델링의 각 수준에서 시스템의 여러 관점을 보여준다. 그림 F.2는 A 자체가 더 큰

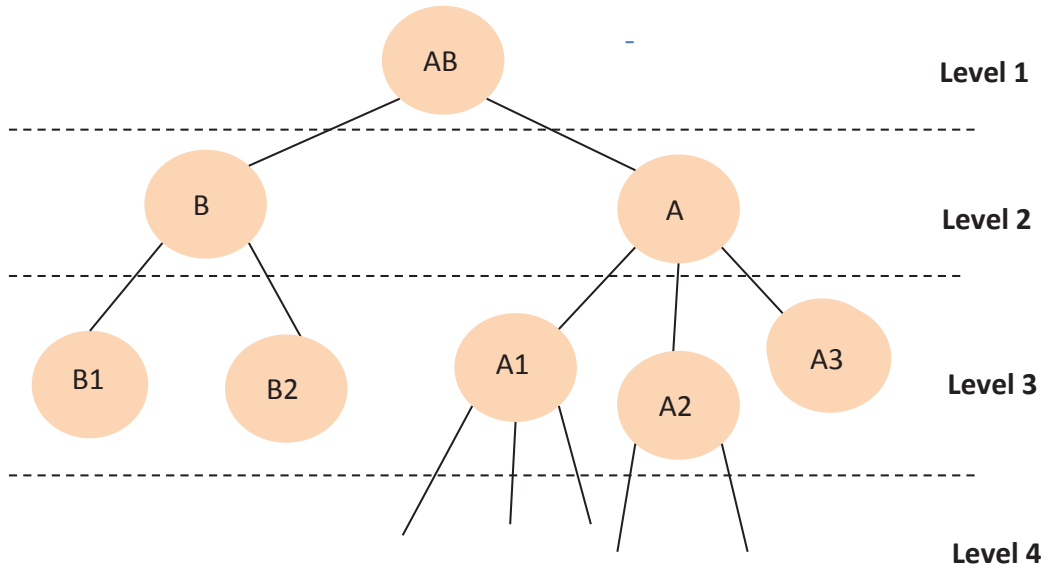
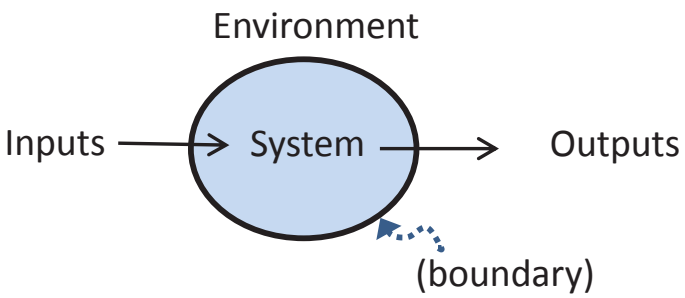


Figure F.2: System A can be viewed as a component (subsystem) of a larger system AB

The recursive nature of the definition of a system is important because many people have suggested that "systems of systems" must be treated differently than systems. In fact, the same general system engineering and analysis methods and techniques are applicable to all systems. A "system of systems" is just a "system" using the definition of a system as defined in system theory and as the concept is used in engineering.

Systems have *states*. A state is a set of relevant properties describing the system at any time. Some properties of the state of an airport viewed as an air transportation system may be the number of passengers at a particular gate, where the aircraft are located and what they are doing (loading passengers, taxiing, taking off, or landing). The components of the state that are relevant depend on how the boundaries are drawn between the system and its environment.

The *environment* is usually defined as the set of components (and their properties) that are not part of the system but whose behavior can affect the system state. Therefore the system has a state at a particular time and the environment has a state. The concept of an environment implies that there is a boundary between the system and its environment. Again, this concept is an abstraction created by the viewer of the system and need not be a physical boundary. What is part of the system or part of the environment will depend on the particular system and its use at the time.



시스템 AB의 일부로 인식되는 추상화를 보여준다.

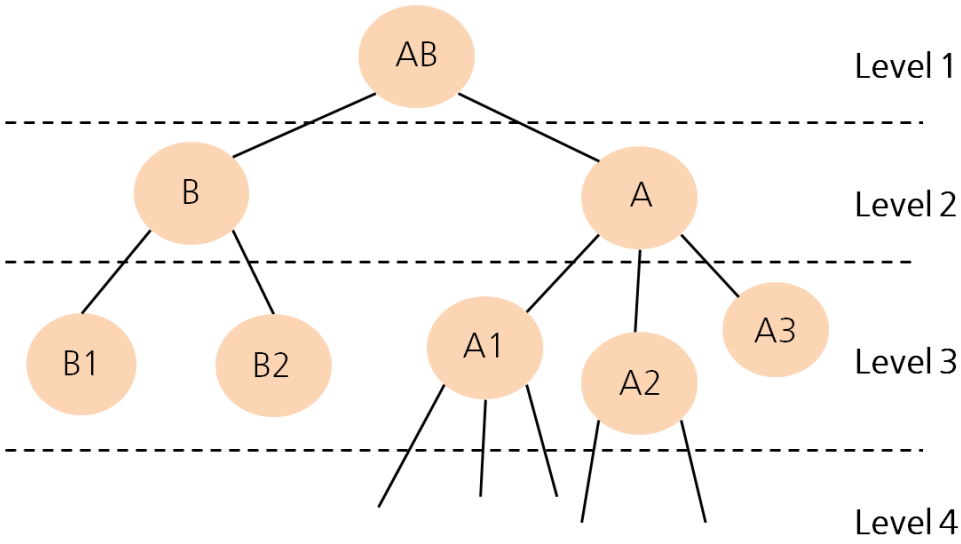
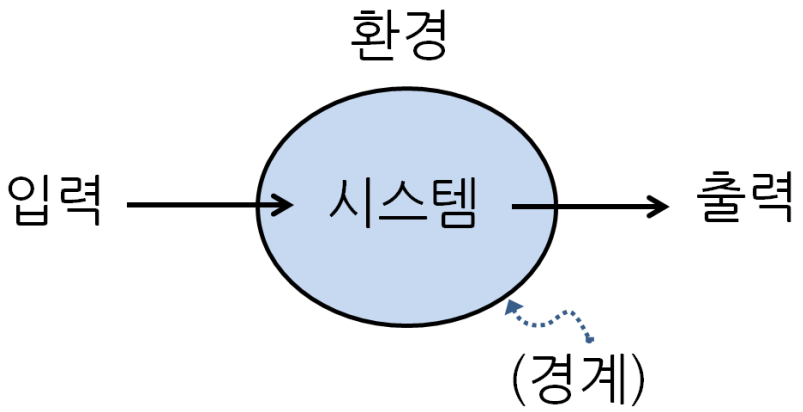


그림 F.2: 시스템 A는 더 큰 시스템 AB의 컴포넌트(서브시스템)로 볼 수 있음

많은 사람들이 시스템과 "시스템의 시스템(systems of systems)"을 다르게 취급해야 한다고 주장하므로 시스템 정의에서 재귀적인 성격을 언급하는 것이 중요하다. 실제로, 일반적인 시스템 엔지니어링, 분석 방법 및 기술이 모든 시스템에 동일하게 적용된다. "시스템의 시스템(system of systems)"은 시스템 이론에서의 시스템 정의를 사용한 것으로, 엔지니어링에서 사용되는 개념인 "시스템"일 뿐이다.

시스템에는 *상태(state)*가 있다. 상태는 모든 시점에서의 시스템을 설명하는 관련 속성(properties)의 집합이다. 항공 운송시스템 관점에서 공항의 일부 상태 속성은 특정 게이트에서의 승객 수, 항공기가 위치한 곳, 그리고 무엇을 하는지(승객 탑승, 유도, 이륙, 또는 착륙)를 의미할 수 있다. 관련된 상태의 컴포넌트는 시스템과 환경간의 경계가 그려지는 방식에 따라 다르다.

*환경(environment)*은 시스템의 일부가 아니지만, 일반적으로 행동이 시스템의 상태에 영향을 줄 수 있는 컴포넌트(및 그것들의 속성)의 집합으로 정의된다. 따라서 시스템은 특정 시간에서의 상태가 있고 환경도 상태가 있다. 환경의 개념은 시스템과 환경 사이에 경계가 있음을 의미한다. 다시 말해, 이 개념은 시스템 관찰자에 의해 만들어진 추상적인 것이며 물리적 경계일 필요는 없다. 시스템의 일부 또는 환경의 일부가 무엇인지는 특정 시스템과 그 당시의 사용에 따라 달라진다.



System *inputs* and *outputs* cross the system boundary. This model is usually called an *open* system. There is also a concept of a *closed system* (which has no inputs or outputs), but this concept does not have much relevance for the engineered systems with which we are most concerned in system safety.

What is Systems Engineering?

System engineering is the attempt to put structure into the design and construction of a system in order to improve the results of the engineering effort. For relatively simple systems, system engineering may not be necessary. Systems engineering came into being after World War II when we started building significantly more complex systems, especially missile and other defense systems, and found that an informal design and construction process often resulted in systems that did not satisfy the goals envisioned for the system and the constraints on its potential behavior, i.e., constraints how it achieved the goals. Lack of a structured process also led to projects being late and over budget.

System engineering is a large subject, but only two concepts are necessary to understand this handbook. The first is the concept of the system engineering phases and process. The most basic model of these phases is shown in Figure F.3.

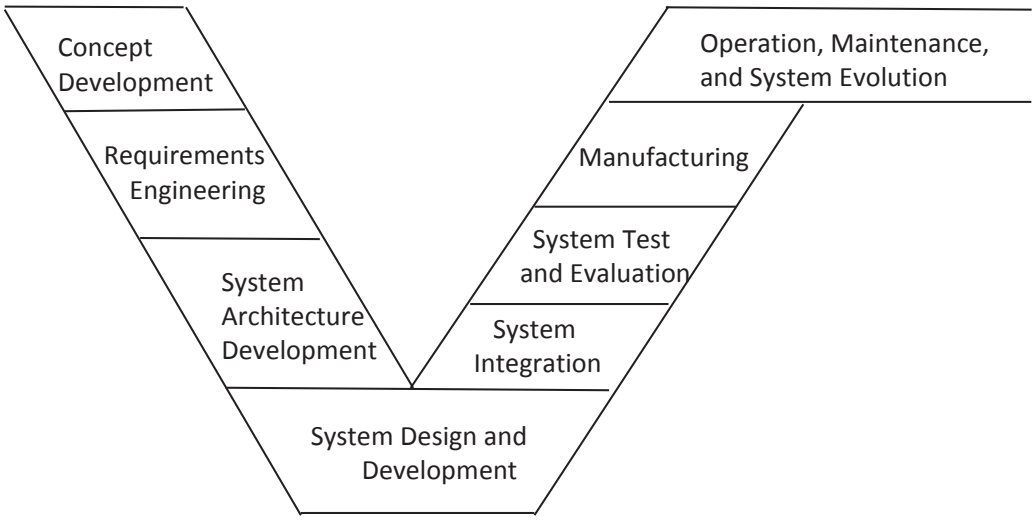


Figure F.3: The Basic System Engineering "V-model"

This model is called the V-model, although different labels may be placed on the various components. Sometimes the phases are drawn in a straight line (if the line tilts downward it is called the "waterfall" model), but there is no difference except in the way it is drawn. The basic idea is that the best results ensue if a structured process is used, where the system concepts are first developed and the basic goals and constraints on the system are defined. There may be various feasibility and other analyses done at this early time in development to stop the process from going down paths that require retracing later. Requirements engineering is usually the next step in the process where detailed requirements are developed for the system given the basic goals and constraints earlier agreed upon. System architecture development means that a basic architecture or high-level design of the system is created before detailed design and development. The left side of the "V" represents the basic design process.

The right side of the V shows the later stages in development when assurance and manufacturing take place. Sometimes operation of the system is included (which is then called a life-cycle model)—as

시스템 *입력(input)*과 *출력(output)*은 시스템 경계를 가로지른다. 이 모델을 일반적으로 *개방형 시스템(open system)*이라고 부른다. 입력 또는 출력이 없는 *폐쇄형 시스템(closed system)*이라는 개념도 있지만 이 개념은 시스템 안전에서 우리가 가장 관심을 갖는 엔지니어드 시스템과는 별로 관련이 없다.

시스템 엔지니어링이란 무엇인가?

시스템 엔지니어링은 엔지니어링 활동의 결과를 향상시키기 위해 시스템의 설계 및 구축 프로세스를 구조화하는 것이다. 상대적으로 간단한 시스템의 경우 시스템 엔지니어링이 필요하지 않을 수도 있다. 시스템 엔지니어링은 제2차 세계 대전 이후 미사일 및 기타 방어 시스템과 같은 매우 복잡한 시스템을 구축하기 시작했을 때 생성되었다. 비정형(informal) 설계 및 구축 프로세스로 인해 종종 시스템의 계획된 목표를 충족시키지 못하였고 시스템의 잠재적 동작에 대한 제약사항, 즉, 어떻게 목표를 달성할 것인지에 대한 제약사항을 충족시키지 못하였다. 또한 구조화된 프로세스의 부족으로 인해 프로젝트가 지연되고 예산이 초과되었다.

시스템 엔지니어링은 커다란 주제이지만 이 핸드북을 이해하기 위해서는 두 가지 개념만 필요하다. 첫 번째는 시스템 엔지니어링 단계와 프로세스의 개념이다. 이러한 단계의 가장 기본적인 모델은 그림 F.3에 나와 있다.

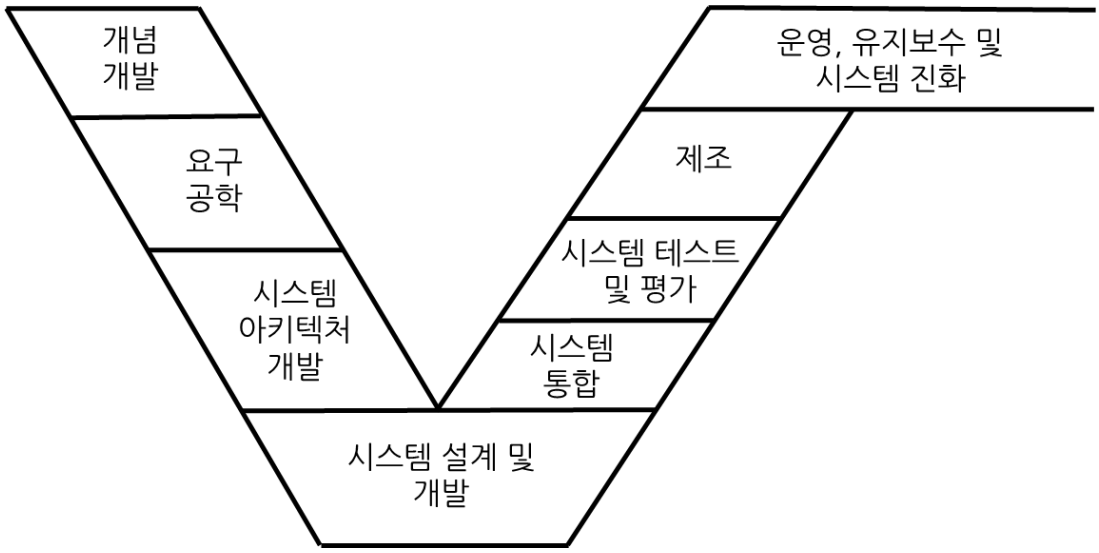


그림 F.3: 기본적인 시스템 엔지니어링 "V-모델"

이 모델을 V-모델이라 부르는데 각 단계를 다른 명칭으로 칭할 수도 있다. 때로는 각 단계가 직선으로 그려지는 경우도 있다(선이 아래쪽으로 기울면 "폭포수" 모델이라고 부름). 그러나 그려지는 방법을 제외하고는 차이가 없다. 기본 아이디어는, 시스템 개념(system concepts)을 처음에 개발하고 시스템의 기본 목표와 제약사항을 정의하는 구조화된 프로세스를 사용한다면 최상의 결과를 도출할 수 있다는 것이다. 나중에 프로세스를 재추적하는 상황을 방지하기 위해 개발 초기 단계에 여러 타당성(feasibility) 및 다른 분석을 수행할 수 있다. 요구공학(requirements engineering)은 일반적으로 프로세스의 다음 단계로, 합의된 시스템의 기본 목표와 제약사항에 기반하여 상세한 요구사항을 개발한다. 시스템 아키텍처 개발(system architecture development)은 상세 설계와 개발을 수행하기 전에 시스템의 기본 아키텍처 또는 시스템의 상위 수준 설계를 하는 것을 의미한다. V-모델의 왼쪽은 기본 설계 프로세스를 나타낸다.

was done in Figure F.3 and Chapter 3 of this handbook because safety analysis needs to continue through the entire life of the system. In right side of the V-model process, the various components that have been designed and constructed are integrated and go through a testing and evaluation phase. They are then manufactured and used.

The figure above is purposely simplified to reduce clutter and only reflects the overall process components. There is always lots of bouncing around the phases and not one straight line process. Requirements are never totally specified at the beginning, for example, as the design process itself will generate more understanding of the problem being solved and therefore result in adding to or changing the original requirements and constraints. Serious problems during development may occasionally require the project to revert totally to an earlier phase. But this step-wise process is generally followed.

In addition, this model should not be taken too literally. The labeling of phases does not mean that some parallelism is not possible and, in fact, common. Waiting for all testing to take place, for example, until the end creates grave risks to a project. Testing of various kinds may start in the earliest concept phase (perhaps in the form of simulations or prototyping) and usually continues through every phase. Also, various labels may be applied to the phases, depending on the type of system being designed. The goal is simply to define a structured process that will be tailored for the various needs of specific projects. Chapter 3 shows how STPA can generally be used in the various phases and activities involved in engineering a complex system.

Another aspect of the V or waterfall model is that it is documentation driven. In my experience, the key to successfully designing a complex system lies in specification. There are usually large numbers of people involved in such systems (sometimes in the thousands) and development and operation occurs over what can be a very long time frame. Specification is necessary to ensure that the result satisfies the needs of the users and that decision making does not descend into chaos because of communication problems.

Specification starts in the earliest stages of development. The previous section of this appendix defined a system as an abstraction whose purpose is imposed by the viewer of the system. In order to ensure that everyone agrees on the basic objectives of the system being created, specifications must at the very least specify the system purpose (goals), the relevant system properties of interest or desired, and any constraints on how the goals can be achieved. In addition, the system boundary must be identified (i.e., what is within the boundaries and under the control of the designers and what is part of the environment that is assumed by the designers and not under their control). Other aspects of the initial specification include the inputs and outputs, the physical or logical components, the structure, the relevant interactions among components and how their behavior affects the overall system state. As development proceeds, the initial specification will be refined and details developed.

Traceability is another important concept in system engineering. Very large and complex systems may have thousands of pages of detailed specifications by the time they are completed. Showing the relationship between parts of the system and what assumptions or previous results of the development process particular decisions were based on is a requirement for creating and operating the system. For example, in STPA, unsafe control actions are traced to system-level hazards and scenarios traced to the unsafe control actions they cause. When changes need to be made, it is possible to determine what needs to be changed without starting over from scratch. Traceability is also a way to record some of the types of rationale that underlie decision making.

The Concept of “Control” in Engineering

The concept of control is important in the conception, design, and operation of a system. A system can conceivably operate without any type of control over its behavior, but it would be difficult to ensure

V-모델의 오른쪽은 보증과 제조(assurance and manufacturing)가 이루어지는 개발의 후반 단계를 보여준다. 시스템의 전체 수명주기 동안 안전 분석을 계속 수행해야 하므로 이 핸드북의 그림 F.3 및 3장에서와 같이 때때로 시스템의 운영이 포함된다(이를 수명주기 모델이라고 함). V-모델 프로세스의 오른쪽 부분에서, 설계되고 구축된 여러 컴포넌트가 통합(integrated)되어 테스트 및 평가(testing and evaluation) 단계를 거친다. 그런 다음 제작되고 사용(manufactured and used)된다.

위의 그림은 혼란을 줄이기 위해 의도적으로 단순화 되었으며, 전체 프로세스 컴포넌트만을 반영하고 있다. 하나의 직선 프로세스가 아니라 항상 여러 단계를 거치면서 수차례 앞뒤를 오간다. 예를 들면, 요구사항은 초기에 완전히 명세되지는 않는데 이는 설계 프로세스에서 문제 해결의 이해를 높이게 됨에 따라 기존의 요구사항과 제약사항을 추가하거나 변경하기 때문이다. 개발 중 심각한 문제로 인해 프로젝트가 완전히 초기 단계로 되돌아가야 할 수도 있다. 그러나 일반적으로 이 단계별 절차를 따른다.

또한, 이 모델을 있는 그대로 사용해서는 안 된다. 단계를 구분하여 표시한 것이 각 단계를 병행하는 것이 불가능함을 의미하는 것은 아니며 실제로도 병행한다. 예를 들어 모든 테스트가 끝날 때까지 기다리는 것은 프로젝트에 심각한 위험을 초래한다. 초기 개념 단계에서 다양한 종류의 테스트를 시작할 수 있으며(시뮬레이션 또는 프로토타입 형태로) 일반적으로 모든 단계에 걸쳐 계속된다. 또한 설계 중인 시스템의 유형에 따라 다양한 명칭이 각 단계에 적용될 수 있다. 목표는 프로젝트의 다양한 요구에 맞춰 구조화된 프로세스를 정의하는 것이다. 3장에서는 복잡한 시스템 엔지니어링과 관련된 다양한 단계와 활동에서 일반적으로 STPA가 어떻게 사용되는지 보여준다.

V-모델 또는 폭포수 모델의 또 다른 측면은 문서 중심(documentation driven)이라는 것이다. 경험상, 복잡한 시스템을 성공적으로 설계하는 핵심은 명세에 있다. 일반적으로 그러한 시스템(때로는 수천개의 시스템)에는 많은 사람들이 관련되어 있으며 매우 긴 시간을 들여 개발과 운영이 이루어진다. 명세(specificaton)는, 결과가 사용자의 요구를 충족시키고 의사 결정이 커뮤니케이션 문제로 인해 혼란에 빠지지 않도록 하기 위해 필요하다.

명세는 개발 초기 단계부터 시작된다. 이 부록의 이전 섹션에서는 시스템을 추상적인 것으로 정의하였고 그 목적이 시스템 관찰자에 의해 설정된다고 하였다. 모든 사람이 시스템의 기본 목표에 동의하는지 확인하기 위해 적어도 명세는 시스템 목적(목표), 관심 있거나 원하는 시스템의 관련 속성, 목표 달성 방법에 대한 제약사항을 명세해야 한다. 또한 시스템 경계가 식별되어야 한다(즉, 무엇이 경계 내에 있고 설계자의 컨트롤 하에 있는지, 무엇이 설계자의 컨트롤에서 벗어나 설계자에 의해 가정된 환경의 일부인지). 초기 명세의 다른 측면에는 입력 및 출력, 물리적 또는 논리적 컴포넌트, 구조, 컴포넌트간의 관련 상호작용, 그것들의 동작이 전체 시스템 상태에 미치는 영향 등이 포함된다. 개발이 진행됨에 따라 초기 명세가 상세화되고 세부 사항이 개발된다.

추적성(traceability)은 시스템 엔지니어링에서 또 다른 중요한 개념이다. 매우 크고 복잡한 시스템에서는 완성하기까지 수천 장의 세부 사양을 작성하는 경우도 있다. 시스템 부분간의 관계를 보여주고 특정 결정의 기반이 된 개발 프로세스의 이전 결과 또는 가정을 보여주는 것은 시스템을 만들고 운영하기 위한 요구사항이 된다. 예를 들어, STPA에서 언세이프 컨트롤 액션(UCA)은 시스템 수준의 위험으로 추적되고 시나리오는 그 시나리오를 초래한 UCA로 추적된다. 변경이 필요한 경우에는 처음부터 다시 시작하지 않고 변경 사항을 결정할 수 있다. 또한 추적성은 의사 결정의 기초가 되는 몇 가지 유형의 이론적 근거를 기록하는 방법이기도 하다.

엔지니어링에서 “컨트롤”의 개념

컨트롤의 개념은 시스템 개념, 설계, 운영에 있어 중요하다. 시스템의 동작에 대한 어떤 유형의 컨트롤

that the system satisfied its goals while at the same time constraining the way it achieves those goals so that adverse, unwanted consequences do not also result.

Control is basic to engineering, especially complex systems. Engineers are dealing not with living things, but with man-made things —automobiles, dams, aircraft, chemical plants, spacecraft, washing machines, robots—that are built to accomplish some goal. If they are not inert (like a bridge), there is a need to operate or control their operation, i.e., the states that they get into.

Non-engineers sometimes interpret the word "control" differently than the way it is used in engineering. Control is not about forcing individual compliance to specified roles and procedures through some type of militaristic-style authority. Rather, control is being used here in a very broad sense and may be implemented through design controls like interlocks and fail-safe design, through process controls such as maintenance and manufacturing procedures, or even social constructs such as culture, incentive structures, and individual self-interest. Without the imposition of some form of control, there would be no way to ensure that the system goals and constraints are achieved.

Simple design features acting as controls often can improve safety without also increasing complexity and cost, but this usually requires considering safety early in the design process. The most effective way to increase safety through design is to use a design that eliminates hazardous states. Next best is to design such that hazards are rare and easy to handle if they do occur. A design that reacts to the occurrence of hazards by minimizing the resulting damage is clearly the least desirable form of hazard control but often such controls must be included.

A control loop is thus only one form of control. It is emphasized in *Engineering a Safer World* because the other aspects of basic design for safety were covered so extensively in my previous book titled *Safeware*. In addition, system theory emphasizes the concept of the control loop or active control. It is not possible to teach the techniques used to design for safety in this appendix, but some basic concepts such as the distinction between p*assive* and *active* control may be helpful in understanding general safety design techniques.

There are three types of design techniques to "control" safety. The first is to use an intrinsically or inherently safe design. An intrinsically safe design is one that is not capable of producing the hazard of concern. For example, the system design may not have sufficient energy to cause an explosion. If the concern is with the use of potential toxic substances, there may be non-toxic substitutes. If the potential exists for lost information over a communication network, then more direct communication may be used if it is feasible.

If an intrinsically safe design is not possible, the use of passive control mechanisms may be possible. Passive controls do not require a positive action in order to be effective, but instead rely on basic physical principles, such as gravity. A physical interlock that prevents the system from entering a hazardous state without any overt control action is a passive control mechanism. Examples include:

- A pressure sensitive mat or light curtain that shuts off power to a robot or dangerous machinery if someone comes near.
- A physical device that ensures the correct sequencing of valve turn-off and turn-on or ensures that two valves cannot both be on or off at the same time.
- A freeze plug in a car's engine cooling system where expansion will force the plug out rather than crack the cylinder if water in the block freezes.
- A fusible plug in a boiler that becomes exposed if there is excessive heat and the water drops below a predetermined level. In that event, the plug melts and the opening permits the steam to escape, which reduces the pressure in the boiler and prevents an explosion.

없이도 시스템이 운영될 수 있을 것 같지만, 원하지 않은 부정적 결과가 발생하지 않도록 목표를 달성하는 방법을 제한하면서 시스템 목표를 달성하는 것은 어렵다.

컨트롤은 엔지니어링의 기본이며 복잡한 시스템에서는 특히 그렇다. 엔지니어는 살아있는 것이 아니라 사람이 만든 것을 다룬다. 예를 들면, 어떤 목표를 달성하기 위해 만들어진 자동차, 댐, 항공기, 화학 공장, 우주선, 세탁기, 로봇 등이다. 자력으로 움직이는 것이 아닌 경우(다리(bridge)와 같이), 그것들의 동작 즉, 그 상태를 조작하거나 컨트롤해야 한다.

엔지니어가 아닌 사람들은 "컨트롤"이라는 단어를 엔지니어링에서 사용하는 방식과 다르게 해석하는 경우가 있다. 컨트롤은 군사적 방식(militaristic-style)의 통제권한을 통해 특정 역할과 절차를 개별적으로 준수할 것을 강요하는 것이 아니다. 오히려 컨트롤은 매우 광범위한 의미로 사용되고 있으며 인터락(interlock)과 페일세이프(fail-safe) 설계와 같은 설계 컨트롤, 유지보수 및 제조 절차와 같은 프로세스 컨트롤, 또는 문화, 인센티브 구조 및 개인 이익과 같은 사회적 구조를 통해서 구현될 수 있다. 어떤 형태든 컨트롤을 부과하지 않으면 시스템 목표와 제약사항을 달성할 수 있는 방법은 없다.

컨트롤로써 동작하는 간단한 설계의 특징은 종종 복잡도와 비용을 증가시키지 않으면서도 안전을 향상시킬 수 있다는 것이다. 그러나 일반적으로 설계 초기에 안전을 고려해야 한다. 설계를 통해 안전을 향상시키는 가장 효과적인 방법은 위험한 상태를 제거하는 설계를 하는 것이다. 차선책으로는, 위험이 발생하는 경우 그 위험이 매우 드물게 발생하고 다루기 쉽도록 설계하는 것이다. 발생하는 피해를 최소화하는 방식으로 위험 발생에 대응하는 설계는 그다지 바람직하지 않은 위험 컨트롤 방식이지만 종종 그러한 컨트롤이 포함되어야 한다.

이와 같이 컨트롤 루프는 컨트롤의 한 가지 형식일 뿐이다. 안전을 위한 기본 설계의 다른 측면들은 *Safeware*라는 책에서 광범위하게 다루어졌기 때문에 *Engineering a Safer World*에서는 컨트롤 루프를 강조하였다. 또한 시스템 이론은 컨트롤 루프 또는 능동 제어(active control)의 개념을 강조한다. 이 부록에서 안전을 설계할 때 사용하는 기법을 가르칠 수는 없지만 *수동(passive)* 및 *능동(active)* 제어의 구분과 같은 일부 기본 개념은 일반적인 안전 설계 기술을 이해하는 데 도움이 될 수 있다.

안전을 "컨트롤"하는 세 가지 유형의 설계 기법이 있다. 첫 번째는 본질적으로 또는 선천적으로 안전한 설계를 하는 것이다. 본질적으로 안전한 설계는 우려되는 위험이 발생할 수 없는 설계를 말한다. 예를 들어, 폭발을 일으킬 만큼 충분한 에너지를 갖지 않도록 시스템을 설계할 수 있다. 잠재적 독성 물질 사용의 우려가 있는 경우, 독성이 없는 대체 물질을 사용할 수 있다. 통신 네트워크를 통해 정보 손실 가능성이 있는 경우, 가능하면 직접 통신을 사용할 수 있다.

본질적으로 안전한 설계가 불가능한 경우 수동 제어 메커니즘을 사용할 수 있다. 수동 제어는 효과적이기 위해 적극적(positive) 조치가 필요하지 않고 대신 중력과 같은 기본적인 물리적 원리를 이용한다. 어떤 명시적인 컨트롤 액션 없이 시스템이 위험한 상태로 들어가는 것을 방지하는 물리적 인터락은 수동 제어 메커니즘이다. 다음과 같은 예시가 있다.

- 누군가가 가까이 오면 로봇이나 위험한 기계의 전원을 차단하는 압력 감지 매트 또는 라이트 커튼
- 밸브를 끄고 켜는 올바른 순서를 보장하거나, 두 밸브를 동시에 켜거나 끌 수 없게 하는 물리적 장치
- 블록의 물이 얼어붙은 경우 실린더를 깨뜨리지 않고 팽창시켜 플러그를 강제로 끄는 자동차 엔진 냉각 시스템의 동결 플러그
- 과열이 발생하고 물이 사전에 설정한 수준 이하로 떨어지면 노출되는 보일러의 용해 플러그. 이 경우, 플러그가 녹고 입구에서 증기가 빠져 나와 보일러의 압력을 감소시키고 폭발을 방지함
- 특정 제한 속도 이상을 허용하지 않는 속도 조절기 또는 위험한 수준 이하의 압력을 유지하는

- A speed governor that does not allow speeds over a specific limit or relief valves that maintain pressure below dangerous levels.

Passive controls are often designed so that the system essentially fails into a safe state. In fact, the basic approach to aircraft safety is to build the system to be fail safe. Some examples of fail-safe design are:

- Old railway semaphores that used weights to ensure that if the cable (controlling the semaphore) broke, the arm would automatically drop into the STOP position.
- Air brakes that are held in an off position by air pressure. If the brake line breaks, air pressure is lost and the brakes are automatically applied.
- A deadman switch
- A bathyscope where ballast is held in place by magnets. If electrical power is lost, the ballast is released and the bathyscope ascends to the surface.
- Designs where the effects of a failure (such as fan blades being ejected when an aircraft engine fails) are contained so that adjacent components are not affected.

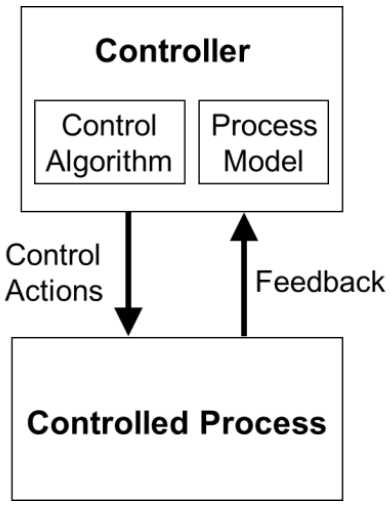
An *active control*, in contrast, requires a hazardous condition to be detected and corrected. Active controls today often involve the use of computers and the basic control loops described and used throughout this handbook. Notice that there is more that can go wrong with active controls, e.g., the failure or hazardous state may not be detected or it may be detected but not corrected or it may not be corrected in time to prevent a loss. Passive controls are more reliable as they usually depend on physical principles and simpler designs while active control depend on less reliable detection and recovery mechanisms and usually much more complex designs. Passive controls, however, tend to be more restrictive in terms of design freedom and are not always feasible to implement in complex systems.

Active controls usually involve a control loop.

What is a Control Loop?

A control system commands, directs, or regulates the behavior of other devices or processes using control loops. Such control systems may range from a single home heating controller using a thermostat to control a boiler to large industrial control systems used for controlling complex processes and machinery.

A very basic control loop is shown below (and elsewhere in this handbook):



In order to control a process, the controller must have a *goal* or goals, which can include maintaining constraints on the behavior of the controlled process. In addition, the controller must have some way to

완화 밸브

수동 제어는 시스템이 기본적으로 안전한 상태로 실패하도록 설계되는 경우가 많다. 실제로 항공기 안전의 기본 접근법은 시스템을 페일세이프(fail-safe)하게 만드는 것이다. 페일세이프 설계의 예시는 다음과 같다.

- 세마포어를 제어하는 케이블이 끊어지면 차단기가 자동으로 STOP 위치로 떨어지게 하기 위해 무게 추를 사용하는 옛날의 철도 세마포어
- 공기 압력에 의해 오프(off) 위치에 고정되는 에어브레이크. 브레이크 라인이 끊어지면 공기 압력이 손실되고 브레이크가 자동으로 적용됨
- 데드맨 스위치(deadman switch)
- 밸러스트(ballast)가 자석에 의해 유지되는 심해 관측경. 전력이 손실되면 밸러스트가 해제되고 심해 관측경은 수면으로 올라감
- 설계 시 실패의 영향을 고려하여(예를 들어, 항공기 엔진 장애 시 팬 날개가 튀어 나옴), 실패로 인해 인접한 컴포넌트가 영향 받지 않음

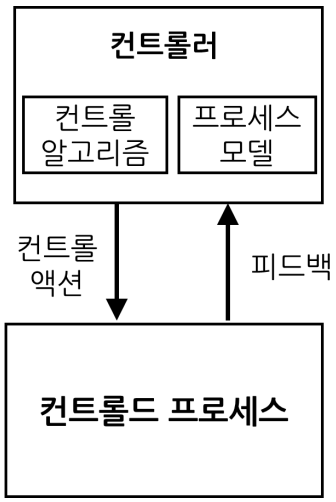
반대로 *능동 제어(active control)*는 위험 조건의 탐지 및 수정이 필요하다. 오늘날 능동 제어는 이 핸드북의 전반에 걸쳐 기술되고 사용되는 기본 컨트롤 루프를 종종 사용한다. 실패 및 위험한 상태가 탐지되지 않거나 탐지되어도 수정되지 않은 경우 또는 손실 방지를 위한 수정이 제때 이루어지지 않는 경우 등 능동 제어와 관련하여 문제가 있을 수 있다는 점을 유의해야 한다. 능동 제어는 신뢰성이 떨어지는 탐지와 복구 메커니즘, 그리고 보통 훨씬 복잡한 설계에 의존하는 반면, 수동 제어는 일반적으로 물리적 원리와 단순 설계에 의존하기 때문에 보다 신뢰할 수 있다. 그러나 수동 제어는 설계 자유도라는 측면에서는 보다 제한적이 경향이 있으며 복잡한 시스템에서 구현하는 것이 항상 가능하지는 않다.

능동 제어에는 일반적으로 컨트롤 루프가 수반된다.

컨트롤 루프란 무엇인가?

컨트롤 시스템은 컨트롤 루프를 사용하여 다른 장치나 프로세스의 동작을 명령, 지시 또는 규제한다. 이러한 컨트롤 시스템은 보일러를 제어하기 위해 온도조절장치를 사용하는 단일 가정용 난방 컨트롤러부터, 복잡한 프로세스와 기계를 제어하는 대형 산업 컨트롤 시스템까지 다양하다.

아래 그림은(그리고 이 핸드북의 다른 부분에) 매우 기본적인 컨트롤 루프를 보여준다:



affect the behavior of the controlled process, i.e., the state of the controlled process or system. These are labeled control actions in the figure. In engineering, control actions are implemented by *actuators*. In order to know what control actions are necessary, the controller must be able to determine the current state of the system. In engineering terminology, information about the state of the system is provided by *sensors* and is called *feedback*. Finally, the controller must contain some model of the state of the controlled process as explained elsewhere in the handbook.

As an example, a simple thermostat may have a goal of maintaining a temperature within a specific range. Feedback provides information to the controller about the current state of the controlled process, in this case the temperature of the air in the room. The feedback is used to update the controller's model of the process. Other information may also be part of the process model, such as environmental information (outside air temperature) or basic information about natural temperature fluctuations. The controller uses the process model to decide what control actions to provide, e.g., apply hot air or cool air, in order to change the state of the room temperature (controlled process) to stay within the required range.

There are two general operational modes for a control loop: feedback control and feedforward control. *Feedback control* is what was described in the paragraph above. When driving, a driver may read the speedometer (feedback) and decide to brake or step on the accelerator to keep the automobile's speed at a desired level.

An example of *feedforward control* occurs when the driver approaches a hill. The driver could wait until the car starts to slow down and then step on the accelerator, but more likely an experienced driver will anticipate that acceleration will be required to maintain a constant speed when going up the hill and the driver will increase acceleration before the car actually slows down. Thus, in feedforward control, the controller uses a model of the current state of the process (in this case the car speed) and the future (operating on an incline) and then provides a control action without specific feedback to identify the need.

Often feedback and feedforward control are used in tandem. In our example, the driver might anticipate the need to step on the accelerator before feedback is received using feedforward control, but feedback can be used to adjust the feedforward control actions so that a constant speed is maintained on the incline if the feedforward calculations are not perfect.

In the chapter of this handbook on leading indicators, *shaping actions* are described as attempts to maintain a safe state by designing controls to maintain assumptions, prevent hazards (unsafe states in the controlled process), and control migration of the process to states of higher risk. Shaping actions provide a type of feedforward control. The type of control used may be passive or active. *Hedging or contingency actions* provide a type of feedback control as they involve monitoring the system for signs of increasing risk and responding accordingly. Hedging actions that include monitoring the effectiveness of the shaping actions provide a combination of feedforward and feedback control.

The engineering concept of control (which is based on system theory) can also be applied to social and organizational systems and has been used extensively in management theory and social sciences.

Systems Theory vs. Complexity Theory

STAMP is based on systems theory, not complexity theory. Contributions to systems theory have been made by many people, but Ludwig von Bertalanffy, an Austrian biologist, is credited as one of the founders of what he called general systems theory. Norbert Weiner explored the implications for mathematics and engineering. Weiner called his theory *cybernetics* but that term has faded over time and Bertalanffy's terminology is generally used today.

컨트롤러는 프로세스를 컨트롤하기 위해 하나 이상의 *목표(goal)*를 가져야 하며 컨트롤드 프로세스의 동작에 제약사항을 유지해야(maintaining constraints) 한다. 또한 컨트롤러에는 컨트롤드 프로세스의 동작에 영향을 주기 위한 방법이 있어야 한다. 즉, 컨트롤드 프로세스의 동작은 컨트롤드 프로세스의 상태 또는 시스템의 상태를 말한다. 이 방법을 그림에서 컨트롤 액션으로 표기하였다. 엔지니어링에서 컨트롤 액션은 *액추에이터(actuators)*에 의해 구현된다. 컨트롤러는 어떤 컨트롤 액션이 필요한지 알기 위해 시스템의 현재 상태를 결정할 수 있어야 한다. 공학 용어에서 시스템 상태 정보는 *센서(sensors)*가 제공하며 이것을 *피드백(feedback)*이라고 부른다. 마지막으로, 핸드북의 다른 부분에서 설명된 바와 같이 컨트롤러에는 컨트롤드 프로세스 상태에 대한 모델(model of the state of the controlled process)이 포함되어야 한다.

예를 들어 간단한 온도조절장치는 온도를 특정 범위로 유지하는 목표를 가질 수 있다. 피드백은 컨트롤러에게 컨트롤드 프로세스의 현재 상태 정보를 제공하는데 이 경우에는 방의 온도를 제공한다. 피드백은 컨트롤러의 프로세스 모델을 업데이트 하는 데 사용된다. 환경 정보(외부 기온)나 자연적인 온도 변동에 대한 기본적인 정보와 같은 다른 정보들도 프로세스 모델의 일부가 될 수 있다. 컨트롤러는 어떤 컨트롤 액션을 제공할지를 결정하는 데 프로세스 모델을 사용한다. 예를 들면 방의 온도가 요구 범위 내에서 유지될 수 있도록 방 온도를 변경하기 위한(컨트롤드 프로세스) 열풍 또는 냉풍을 작동한다.

컨트롤 루프에는 피드백(feedback) 컨트롤과 피드포워드(feedforward) 컨트롤이라는 두 가지 일반적인 운영 모드가 있다. *피드백 컨트롤(feedback control)*은 위 단락에서 설명한 것이다. 운전할 때 운전자는 속도계(피드백)를 읽고 자동차의 속도를 원하는 수준으로 유지하기 위해 브레이크를 밟을지 액셀러레이터를 밟을지를 결정한다.

*피드포워드 컨트롤(feedforward control)*의 예로 운전자가 언덕에 접근하는 경우를 들 수 있다. 운전자가 차의 속도가 느려질 때까지 액셀러레이터를 밟는 것을 기다릴 수도 있지만 숙련된 운전자는 언덕을 올라갈 때 일정한 속도가 유지되기 위해서는 가속이 필요함을 예상하고 차가 실제로 느려지기 전에 가속도를 높일 가능성이 높다. 이와 같이 피드포워드 컨트롤에서 컨트롤러는 프로세스의 현재 상태(이 경우, 차의 속도)와 미래(경사면에서 동작)의 모델을 사용하여 컨트롤의 필요 여부를 식별하는, 특정한 피드백 없이도 컨트롤 액션을 제공한다.

종종 피드백과 피드포워드 컨트롤은 동시에 사용된다. 이 예시에서, 운전자는 피드포워드 컨트롤을 사용하여 피드백을 받기 전에 액셀러레이터를 밟아야 한다고 예측할 수도 있지만 피드포워드 계산이 완벽하지 않은 경우에는, 경사에서 일정한 속도가 유지되도록 피드백을 사용하여 피드포워드 컨트롤 액션을 조정할 수 있다.

이 핸드북의 선행지표 장(6장)에서는 *쉐이핑 액션(shaping actions)*에 대해 가정을 유지하고 위험(컨트롤드 프로세스의 안전하지 않은 상태)을 방지하며 프로세스가 더 높은 상태의 리스크로 진입하는 것을 통제하기 위해 컨트롤을 설계하여 안전한 상태를 유지하는 것으로 설명하고 있다. 쉐이핑 액션은 일종의 피드포워드 컨트롤을 제공하며, 사용되는 컨트롤 유형은 수동 또는 능동일 것이다. *헷징 또는 우발상황 조치(Hedging or contingency actions)*는 일종의 피드백 컨트롤을 제공하는데, 증가하는 리스크 징후 및 대응을 모니터링하는 시스템과 관련되어 있기 때문이다. 쉐이핑 액션의 효과 모니터링을 포함하는 헷징 액션은 피드포워드와 피드백 컨트롤을 모두 제공한다.

컨트롤의 공학적 개념(시스템 이론에 기반한)은 사회, 조직 시스템에도 적용될 수 있으며 경영 이론 및 사회 과학 분야에서 광범위하게 사용되어 왔다.

Systems theory, as you have seen in this handbook and perhaps elsewhere, is a set of principles that can be used to understand the behavior of complex systems, whether they be natural or man-made systems. *Systems thinking* is the term often used to describe what people are doing when they apply systems theory principles. These principles are briefly described in Chapter 1 of this handbook.

Complexity theory grew out of systems theory and other concepts in the 1960s and is usually associated with the Santa Fe Institute and researchers working there. Some commonalities exist between Systems Theory and Complexity Theory in that both include terms like emergence and focus on complex system behavior as a whole rather than on reduction or decomposition into components. The basic components of system theory are emergence, hierarchy, communication and control and these also are included in complexity theory. Both systems theory and complexity theory also include concepts of feedback and feedforward control, adaptability, nonlinear interactions, and constraints. Both reject reductionism or decomposition as a principle for understanding system behavior.

There are, however, significant differences. Complexity theory was created to describe natural systems where seemingly independent agents spontaneously order and reorder themselves into a coherent system using laws of nature that we do not yet fully understand. Systems theory, in contrast, is more appropriate for man-made and designed systems where the system is purposely created by humans using some engineering or design process and where the basic design is known and changes are controlled.

Another major difference is that systems theory considers all systems as displaying emergent behavior while complexity theory divides systems into four types: simple, complicated, complex, and chaotic, each with different degrees or types of emergence.

As such, systems theory appears to be most appropriate for engineered or designed systems while complexity theory is most appropriate for natural systems where the design is unknown, such as the weather, and for sociological systems, such as communities, that are not designed and where there is a lack of order and it is very hard or impossible to predict the emergent behavior. Complexity theory deals with behavior that cannot be designed but instead must be experienced and studied as it arises.

Systems theory, at least for me, is easier to understand and use. Complexity theory frameworks can be difficult and confusing to apply and understand and may be overkill for the engineering goals related to improving safety and other emergent system properties. Thus, systems theory was selected as the basis for STAMP.

시스템 이론 vs 복잡도 이론(Systems Theory vs. Complexity Theory)

STAMP는 복잡도 이론이 아닌 시스템 이론을 기반으로 한다. 많은 사람들이 시스템 이론에 기여해 왔지만 오스트리아의 생물학자인 베르탈란피(Ludwig von Bertalanffy)는 일반적인 시스템 이론의 창시자 중 한명으로 꼽힌다. 노버트 위너(Norbert Weiner)는 수학 및 공학에서의 의미를 연구하였다. 그의 이론은 *사이버네틱스(cybernetics)*라는 것인데 이 용어는 시간이 지남에 따라 사라졌고 베르탈란피의 용어가 오늘날 일반적으로 사용되고 있다.

이 핸드북 또는 다른 곳에서 볼 수 있는 시스템 이론은 자연적인 것이든 인공적인 시스템이든 간에 복잡한 시스템의 동작을 이해하는 데 사용될 수 있는 일련의 원칙이다. *시스템 사고(systems thinking)*란 시스템 이론의 원칙을 적용할 때 사람들이 무엇을 하는지 설명하기 위해 자주 사용되는 용어이다. 이 원칙들은 이 핸드북의 1장에 간략하게 기술되어 있다.

*복잡도 이론(complexity theory)*은 1960년대 시스템 이론 및 다른 개념에서 발전하였으며 일반적으로 산타페 연구소(Santa Fe Institute)와 그 곳에서 일한 연구원들과 관련되어 있다. 시스템 이론과 복잡도 이론 사이에는 몇 가지 공통점이 있다. 모두 발현(emergency)과 같은 용어를 사용하고 시스템을 컴포넌트로 환원하거나 분해하기 보다는 전체로 간주하여 복잡한 시스템의 동작에 초점을 맞춘다는 점이 그것이다. 시스템 이론의 기본 컴포넌트는 발현, 계층구조, 커뮤니케이션, 컨트롤이며 이것은 복잡도 이론에도 포함된다. 시스템 이론과 복잡도 이론 모두 피드백과 피드포워드 컨트롤, 적응가능성, 비선형적 상호작용, 제약사항의 개념을 포함한다. 두 이론 모두 시스템의 동작을 이해하는 원칙으로 환원주의(reductionism) 또는 분해(decomposition)를 받아들이지 않는다.

그러나 중요한 차이점이 있다. 복잡도 이론은, 독립적으로 보이는 에이전트들이 우리가 아직 완전히 이해하지 못하는 자연의 법칙에 따라 스스로 일관된 시스템으로 정렬하거나 재정렬하는 자연 시스템을 묘사하기 위해 만들어졌다. 반면, 시스템 이론은 사람이 만든, 설계된 시스템에 보다 적합한 것이다. 여기서 시스템이란 엔지니어링 또는 설계 프로세스를 통해 사람이 의도를 가지고 만든 것으로 기본적인 설계를 알고 있으며 변경을 컨트롤 할 수 있다.

또 다른 중요한 차이점으로 시스템 이론은 모든 시스템이 발현적 동작(emergent behavior)을 보이는 것으로 간주하는 반면, 복잡도 이론은 시스템을 단순한(simple), 복합적인(complicated), 복잡한(complex), 혼란스러운(chaotic)의 4가지 유형으로 나눈다는 것이다. 각 유형은 각각 다른 정도 또는 각기 다른 유형의 발현 속성을 갖는다.

이와 같이 복잡도 이론은 날씨와 같이 설계를 알 수 없는 자연 시스템, 그리고 커뮤니티와 같이 설계되지 않았고 질서가 결여되어 발현적 동작을 예측하기 매우 어렵거나 불가능한 사회학적 시스템에 가장 적합하며, 시스템 이론은 엔지니어드 시스템 또는 설계된 시스템에 가장 적합한 것으로 보인다. 복잡도 이론은 설계할 수는 없지만 대신 발생하였을 때 경험하고 학습해야 하는 동작을 다룬다.

적어도 저자에게 있어 시스템 이론은 이해하기 쉽고 사용하기 쉽다. 복잡도 이론의 프레임워크는 적용 및 이해하기가 어렵고 애매모호하며 안전 개선 및 다른 시스템 발현 속성과 관련된 엔지니어링 목표를 달성하기에는 다소 무리가 있을 수 있다. 따라서 시스템 이론이 STAMP의 기반으로 선택되었다.

Appendix G: A Control Model to Assist In Causal Scenario Generation

The goal of STPA is to identify the causal scenarios that can lead to a hazardous state. A model was provided (see Figure G.1 below) in *Engineering a Safer World* that showed some of the general reasons why hazardous states might be generated. The goal of providing this model was to assist in generating scenarios. One problem, however, is that the model is very abstract and leaves out details that can be useful in scenario generation. The model also combines automated and human controllers in one general box labelled "Controller," another abstraction that leaves out important differences. Later, in Chapter 9 of the book, another model was created to describe design techniques. This model, too, is overly general. Since the book was published, other more detailed models for human controllers have also been generated. In this appendix, we describe a model that is more complete and hopefully more useful to those identifying causal scenarios.

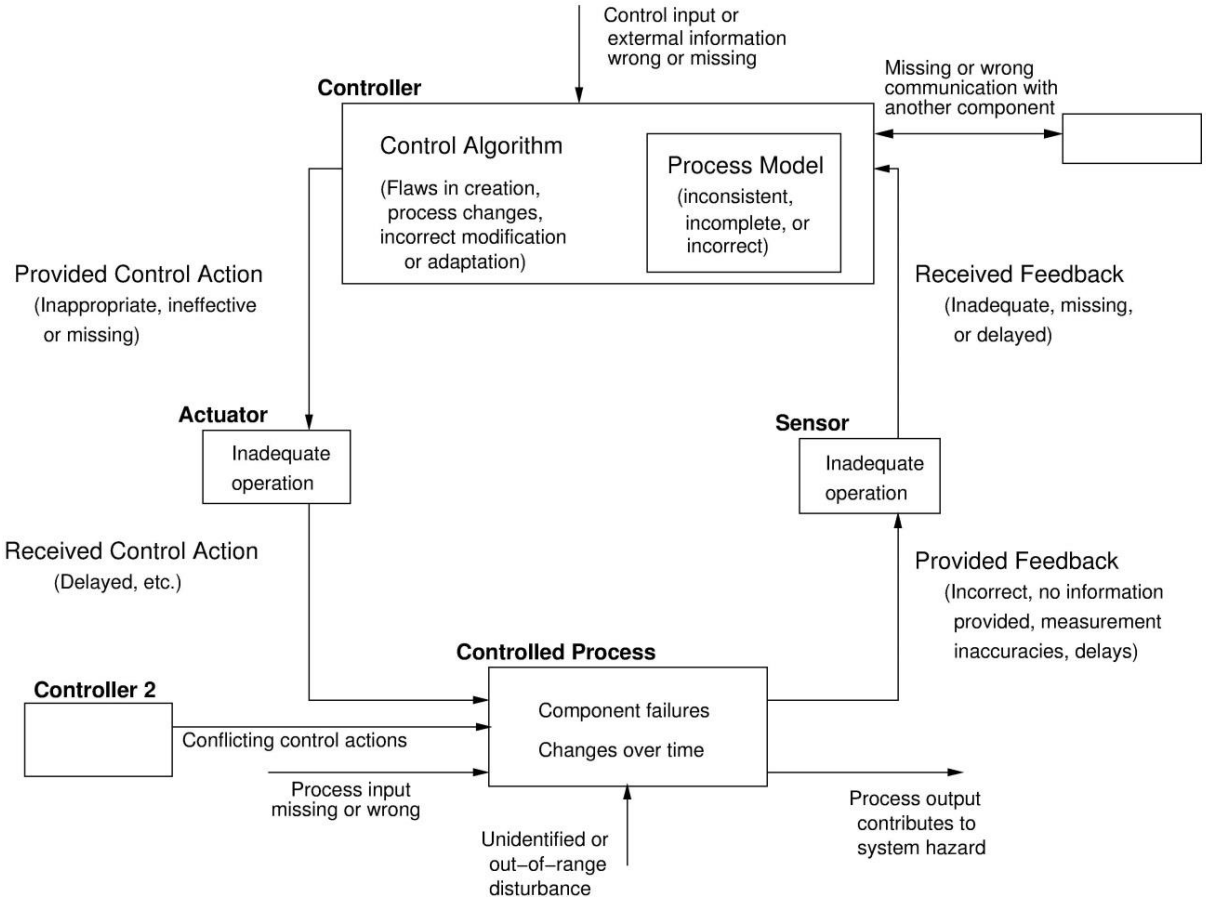


Figure G.1: Previous General Model to Assist in Generating Causal Scenarios

부록 G: 원인 시나리오 생성을 도와주는 컨트롤 모델

STPA의 목표는 위험한 상태로 이어질 수 있는 원인 시나리오(causal scenarios)를 식별하는 것이다. *Engineering a Safer World*에서는 위험한 상태가 생길 수 있는 몇 가지 일반적인 이유 중 일부를 보여주는 모델을 제시하고 있다(아래 그림 G.1 참조). 이 모델을 제공하는 목적은 시나리오 생성을 돕기 위함이다. 그러나 한 가지 문제점은 이 모델이 매우 추상적이며 시나리오 생성에 도움이 되는 세부 사항이 없다는 것이다. 이 모델은 또한 자동화 컨트롤러와 휴먼 컨트롤러를 "컨트롤러"라는 이름을 붙인 하나의 일반 상자로 묶었는데, 이것은 중요한 차이점을 안 보이게 하는 또 다른 추상화이다. 나중에 (Engineering a Safer World의) 9장에서 설계 기술을 설명하기 위한 또 다른 모델을 고안하였으나 이 모델 역시 지나치게 일반적이다. 책이 출간된 후 휴먼 컨트롤러에 대한 보다 자세한 모델도 만들었다. 본 부록에서는 원인 시나리오를 식별함에 있어 보다 완벽하고 유용한 모델에 대해 설명한다.

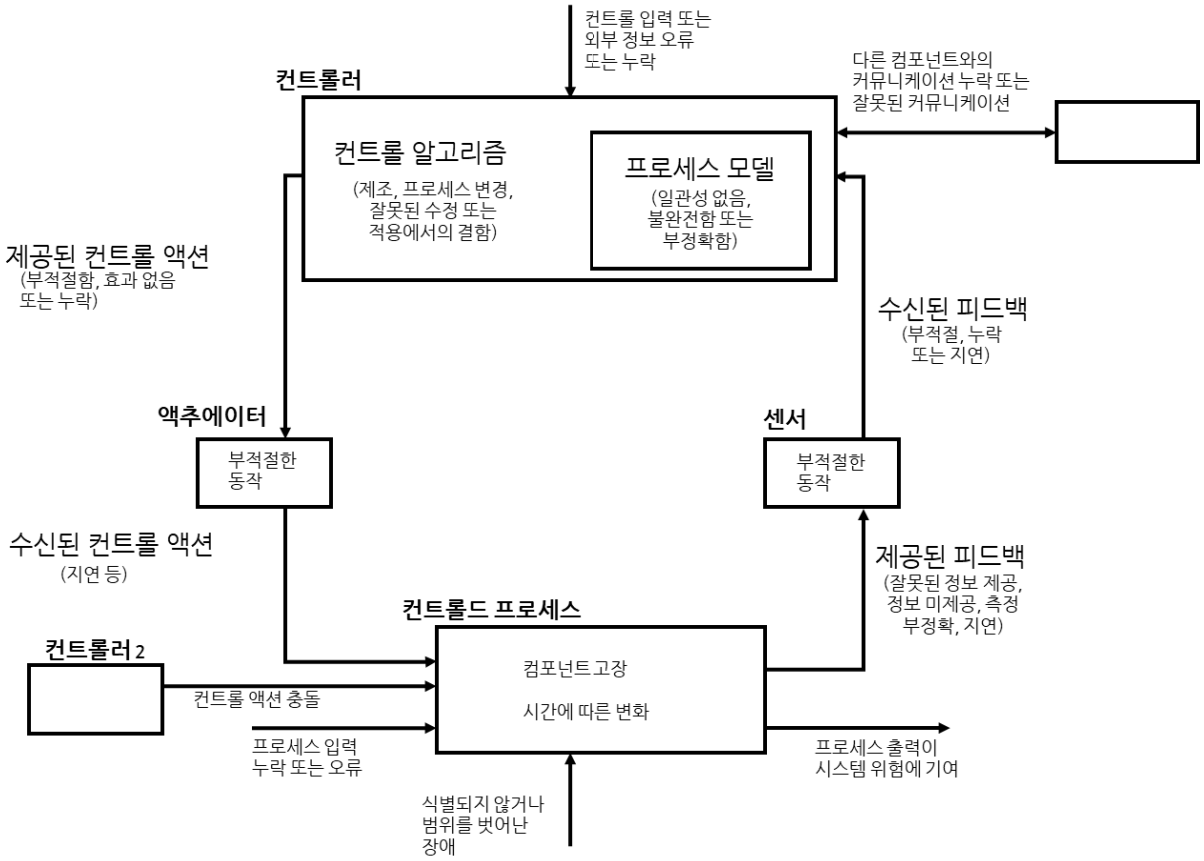


그림 G.1 : 원인 시나리오 생성을 지원하는 이전의 일반적인 모델

An Abstract Model for Causal Scenario Generation

The new model is shown below in Figure G.2. The parts of the model are then described along with how they may be used to create scenarios. There are too many potential causal factors to include them all on the general model diagram itself so they are described separately. When generating causal scenarios using this model as a guide, you will start from a specific control model for your application (system). The generic components below will correspond to the parts of your model. The generic guidance can be used to assist you in generating the specific causal scenarios for your system.

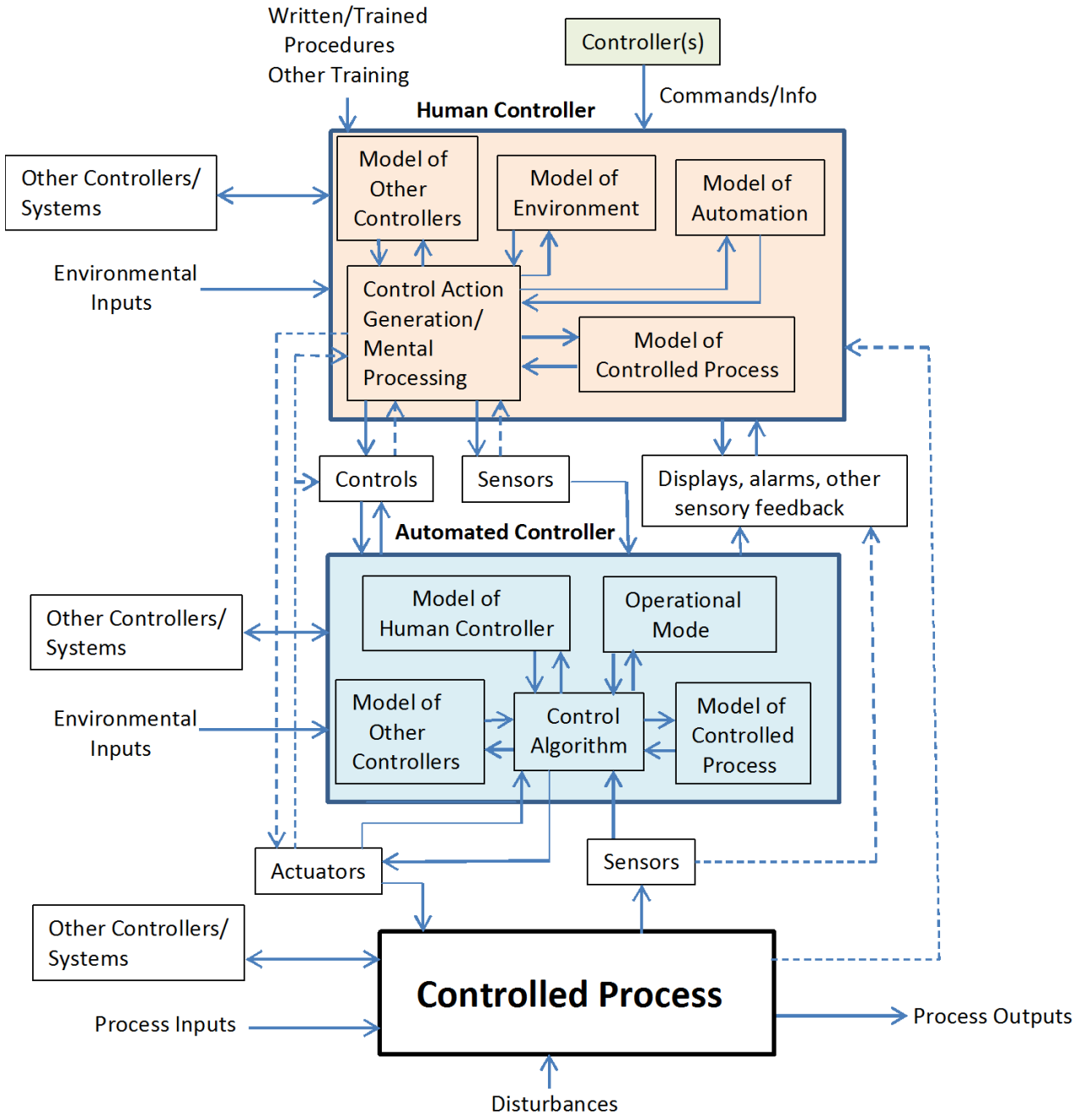


Figure G-2. New General Causal Control Model for STPA Causal Analysis Scenario Generation

원인 시나리오 생성을 위한 추상 모델

새로운 모델을 아래 그림 G.2에 보였으며, 모델의 일부를 시나리오 생성에 사용되는 방법과 함께 설명한다. 잠재적인 원인 요소들을 일반적인 모델 다이어그램 자체에 포함하기에는 그 수가 너무 많아 별도로 분리하여 설명한다. 만약 당신이 이 모델을 가이드로 사용하여 원인 시나리오를 생성한다면 어플리케이션(시스템)의 특정 컨트롤 모델에서부터 시작하게 될 것인데 아래 그림의 일반적인 컴포넌트가 당신의 모델의 일부와 일치할 것이다. 이러한 일반적인 가이드를 사용하면 시스템에 대한 특정 원인 시나리오를 생성하는 데 도움이 될 것이다.

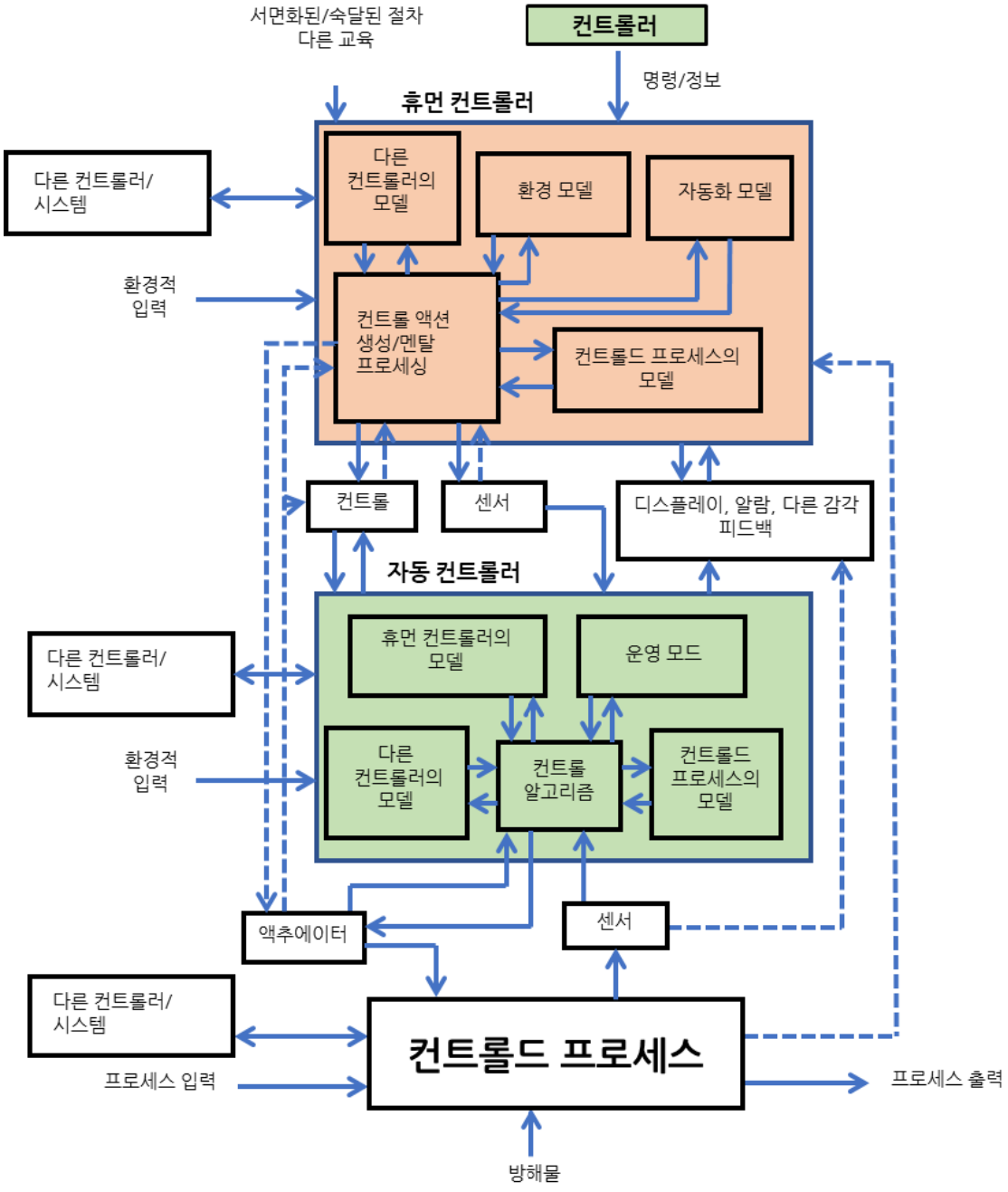


그림 G.2. STPA 원인 분석 시나리오 생성을 위한 새로운 일반적인 원인 컨트롤 모델

Controlled Process

A hazard is defined in terms of the state of the controlled process at the bottom of the figure, e.g., the attitude of the aircraft, the speed of the automobile, the position of the robot. States are made up of components or variables. As the goal of STPA is to identify how the controlled process gets into a hazardous state, then we need to look at the ways the controlled process state can change state. Anything that can change that state is potentially part of a causal scenario for a hazard.

Failures or Degradation over Time of the Controlled Process Components

Failures of the controlled process hardware can cause it to get into a hazardous state. Changes over time of that hardware, such as corrosion, might make it operate differently than designed. Controls such as hardware interlocks may also fail or degrade in such a way that a hazardous state is created. Remember, the goal is not to do a FMEA, that is, to look at all possible failures and their results. The goal is to start from the hazard and determine what conditions in the controlled process could contribute to that hazardous state. As an example, let's assume that the hazard of concern to the causal analysis is the loss of pitch control of an aircraft. Failure of the aircraft pitch control devices, such as the slats, flaps, ailerons, and elevator, could lead to the hazard. If the hazard is a lack of effective braking (deceleration), that may be due to insufficient hydraulic pressure (pump failure, hydraulic leak, etc.).

External Disturbances

Disturbances to the process can lead to a hazardous state, such as lightning affecting an aircraft's electrical system. Or a bird is sucked into an engine and affects the power (thrust) of the aircraft. The national airspace may be disrupted by weather that increases the hazardous state for individual aircraft currently in that airspace. The environment may interfere with the proper execution of the controlled process, such as inadequate braking due to a wet runway (e.g., the wheels hydroplane).

Direct Inputs to the Controlled Process

Direct inputs to the process may affect its state. For an aircraft, loading and unloading passengers or cargo change the weight, which can affect the controllability of the aircraft. For the national airspace, inputs are aircraft that enter the airspace and outputs are those that leave it. If the components of the controlled process hazardous state are affected by any inputs (or outputs), then those process inputs and outputs must be considered in causal scenario generation.

Controllers of the Controlled Process

Clearly, the controllers of a hazardous process can potentially contribute to a hazardous state. The model shows the primary controllers in the blue and tan boxes. In an aircraft, these may be the human pilots and automated flight control system(s). Note that the model does not imply that the pilot(s) are physically onboard the aircraft. The model in Figure G.2 shows another box (to the left of the controlled process) representing other controllers or systems that can impact the state of the controlled process or its components. For example, maintenance activities or the lack of them can contribute to hazards in physical systems. The impact of other automated and human controllers is described separately below.

Automated Controllers

Only one automated controller is shown in the diagram (the blue box) although there may be several of these and they may be designed in a hierarchical structure. Because the model for automated and human controllers always have the same components, only one is shown here.

Except in very simple systems, automated controllers provide control actions to actuators, which change the controlled process state. Sensors provide feedback about the current process state.

컨트롤드 프로세스

위험은 그림의 아래 부분에 해당하는 컨트롤드 프로세스의 상태로 정의되는데 예를 들면, 항공기의 고도, 자동차의 속도, 로봇의 위치 등이다. 상태는 컴포넌트 또는 변수로 구성된다. STPA의 목표는 컨트롤드 프로세스가 어떻게 위험한 상태에 빠지는지를 식별하는 것이므로 컨트롤드 프로세스의 상태가 바뀌는 방식을 잘 살펴보아야 한다. 컨트롤드 프로세스의 상태를 바꿀 수 있는 것은 모두 잠재적으로 위험에 대한 원인 시나리오의 일부가 된다.

시간 경과에 따른 컨트롤드 프로세스 컴포넌트의 실패 또는 저하

컨트롤드 프로세스 하드웨어의 고장은 위험한 상태를 유발할 수 있다. 예를 들면 부식과 같은 시간 경과에 따른 하드웨어 변화로 인해 설계된 것과 다르게 동작할 수 있다. 하드웨어 인터락과 같은 컨트롤 역시 이러한 이유로 고장이나 저하를 일으킬 수 있다. 여기서 목표는 FMEA를 수행하는 것이 아님을, 즉, 모든 가능한 실패와 그 결과를 살펴보는 것이 아님을 기억해야 한다. 우리의 목표는 위험에서부터 시작하여 컨트롤드 프로세스의 어떤 상태가 위험한 상태에 기여할 수 있는지를 판단하는 것이다. 예를 들어, 원인을 분석하고자 하는 위험이 항공기의 피치(pitch) 제어 손실이라고 가정해 보자. 슬랫(slat), 플랩(flap), 에일러론(aileron), 승강타(elevator)와 같은 항공기 피치(pitch) 제어 장치의 실패가 위험을 초래할 수 있다. 만약 위험이 효과적인 브레이킹(감속) 부족이라면 유압 부족(펌프 고장, 유압 누출 등)이 원인일 수 있다.

외부의 방해(disturbance)

프로세스를 방해하는 것은 위험한 상태를 유발할 수 있는데 예를 들면, 항공기의 전기 시스템에 영향을 주는 번개 같은 것이다. 또는 새가 엔진에 빨려 들어가 항공기의 동력(추진력)에 영향을 주는 것도 위험한 상태를 유발할 수 있다. 국가 공역(national airspace)이 날씨로 인해 방해를 받게 되면 해당 영공 내 비행기의 위험한 상태가 증가할 수 있다. 젖은 활주로로 인해(예: 바퀴 수막현상) 브레이킹이 불충분한 것과 같이 환경이 컨트롤드 프로세스의 정상적인 실행을 방해할 수 있다.

컨트롤드 프로세스로의 직접적인 입력

컨트롤드 프로세스로의 직접적인 입력은 컨트롤드 프로세스 상태에 영향을 줄 수 있다. 항공기의 경우 승객/화물의 적재 및 하역은 무게를 변경시키므로 항공기의 조종가능성(controllability)에 영향을 줄 수 있다. 국가 공역에서의 입력은 공역에 진입하는 항공기이며 출력은 공역을 떠나는 항공기이다. 만약 컨트롤드 프로세스 컴포넌트의 위험 상태가 어떤 입력(또는 출력)에 영향을 받는다면 이러한 프로세스 입·출력은 반드시 원인 시나리오 생성 시 고려해야 한다.

컨트롤드 프로세스의 컨트롤러

분명히, 위험한 프로세스의 컨트롤러는 잠재적으로 위험한 상태를 일으킬 가능성이 있다. 이 모델에서 주요한 컨트롤러를 파란색 및 황색 상자로 표시했다. 항공기의 경우, 조종사 및 자동 비행 제어 시스템이 이 상자에 해당될 수 있다. 이 모델은 조종사가 물리적으로 항공기에 탑승하고 있음을 의미하지는 않는다. 그림 G.2의 모델에서 컨트롤드 프로세스의 왼쪽에는 또 다른 상자가 있는데 이 상자는 컨트롤드 프로세스나 그 컴포넌트의 상태에 영향을 줄 수 있는 다른 컨트롤러 또는 시스템을 나타낸다. 예를 들어 유지보수 활동이나 유지보수 활동의 부족은 물리적 시스템의 위험에 기여할 수 있다. 다른 자동화 컨트롤러 및 휴먼 컨트롤러의 영향은 아래에서 별도로 설명한다.

자동화 컨트롤러(Automated Controller)

다이어그램(파란색 상자)에는 하나의 자동화 컨트롤러만 표시되어 있지만 자동화 컨트롤러는 여러 개

Control Path from the Automated Controller to the Controlled Process through the Actuator

As stated in Chapter 2 of this handbook, this control path transfers control actions from the automated controller to the controlled process. The control path may consist of a simple actuator, may involve a series of actuators, or it may transfer control actions through a complex network with switches, routers, satellites, or other equipment. No matter how it is implemented, problems along this control path can lead to hazardous controlled system states when control actions needed to prevent a hazard are not executed, are improperly executed, or are delayed to the point where they become hazardous under some conditions. Examples include actuator failure, transmission problems along the control path, or delays in sending or executing the control action. For hazards related to security flaws, an adversary may impede a control action required to prevent a hazard or inject a control action that leads to a hazard in the controlled process.

Feedback Path from the Controlled Process to the Automated Controller through Sensors

As with the actuator control path, flaws in the feedback path may stem from transmission problems including delays, measurement deficiencies and inaccuracies, sensor failure, or other flaws related to sensor design or operation. Again, an adversary might negatively impact the feedback path. If a flawed process model of the automated controller can contribute to an unsafe control action, then the feedback path should be considered in the causal scenario generation. For example, data may be delayed to the point where it no longer reflects the current state and the controller acts on incorrect information. Do not stop with "flawed process model" in your causal analysis generation. It does not provide you with enough information to provide safeguards against the unsafe control action. Instead, you need to determine why the process model could be flawed.

Automated Control Algorithm

The automated control algorithm has two primary functions: (1) generate control actions and (2) maintain accurate information (models) about the state of the controlled process and external system components and environment that can impact the generation of control actions. Flaws in either of these functions can lead to UCAs.

1. Generating Control Actions

The automated control algorithm may generate unsafe control actions due to flaws in the control algorithm itself, unsafe commands provided to the automated controller from an external source, or flaws in the models that the automated controller has of the controlled process and its environment. The causes of the latter are described in the next section.

STPA generates the safety requirements and constraints for the automated control algorithm. These must, of course, be verified to be correct. A causal factor here, then, is inadequate communication and verification of the operational safety requirements and constraints for the control algorithm.

2. Maintaining Internal Models of the Contextual Factors Associated with UCAs

The safety of decision making about what control actions to perform can also be impacted by the information that the automated controller uses to make those decisions, even if the actual automated algorithms satisfy the safety requirements and constraints. The automated controller uses four types of information in that decision making: (1) the state of the controlled process, (2) the state of other controllers of the controlled process, (3) the operational mode of the automation, and (4) the state of the human controller (in some sophisticated new systems). Various types of flaws in these models can lead to unsafe control actions by the automated controller. The context for the UCAs will contain the information about what flaws in the model can lead to the UCA. For example, the BSCU Autobrake provides a brake control action during a normal takeoff. This UCA then can occur when the BSCU is confused about the state of the controlled process, that is, it does not know that the aircraft is in a

가 있을 수 있으며 계층적 구조로 설계될 수도 있다. 자동화 컨트롤러 및 휴먼 컨트롤러의 모델은 항상 동일한 컴포넌트를 포함하므로 여기에는 하나만 표시한다.

매우 간단한 시스템을 제외하고 자동화 컨트롤러는 액추에이터에 컨트롤 액션을 제공하여 컨트롤드 프로세스의 상태를 변경시킨다. 센서는 현재 프로세스 상태에 대한 피드백을 제공한다.

자동화 컨트롤러에서 액추에이터를 거쳐 컨트롤드 프로세스로 가는 컨트롤 경로

본 핸드북 2장에서 설명한 것처럼 이 컨트롤 경로는 자동화 컨트롤러에서 컨트롤드 프로세스로 컨트롤 액션을 전송한다. 컨트롤 경로는 간단한 액추에이터로 구성되거나 일련의 액추에이터를 포함할 수도 있고 스위치, 라우터, 위성 또는 다른 장비를 포함한 복잡한 네트워크를 통해 컨트롤 액션을 전송할 수도 있다. 컨트롤 경로가 어떻게 구현되었든 간에 위험을 방지하는 데 필요한 컨트롤 액션이 수행되지 않거나, 부적절하게 수행되거나, 특정 조건의 위험해지는 상황까지 지연되는 경우 이 컨트롤 경로의 문제가 컨트롤드 시스템의 위험한 상태를 유발할 수 있다. 이러한 예로는 액추에이터 고장, 컨트롤 경로상의 전송 문제, 컨트롤 액션의 전송 또는 실행에서의 지연이 있다. 보안 결함과 관련된 위험의 경우 공격자가 위험 방지를 위한 컨트롤 액션을 방해하거나 컨트롤드 프로세스에 위험을 초래하는 컨트롤 액션을 주입할 수 있다.

컨트롤드 프로세스에서 센서를 거쳐 자동화 컨트롤러로 가는 피드백 경로

피드백 경로의 결함은 액추에이터 컨트롤 경로의 경우와 마찬가지로 전송 문제가 원인일 수 있으며 지연, 측정 결함 및 부정확성, 센서 오류, 센서의 설계나 작동과 관련된 다른 결함이 포함될 수 있다. 또한, 공격자가 피드백 경로에 부정적인 영향을 줄 수도 있다. 자동화 컨트롤러의 프로세스 모델 결함이 안전하지 않은 컨트롤 액션에 기여한다면 원인 시나리오 생성 시 피드백 경로를 고려해야 한다. 예를 들면, 데이터가 더 이상 현재 상태를 반영하지 않는 시점까지 지연되어 컨트롤러가 잘못된 정보에 의해 동작하는 경우이다. 원인 분석 시 “결함 있는 프로세스 모델”이라는 원인에서 멈추면 안 된다. 이것은 안전하지 않은 컨트롤 액션에 대한 안전장치를 도입하는 데 충분한 정보를 제공하지 못한다. 대신, 프로세스 모델에 결함이 발생할 수 있는 이유를 찾아야 한다.

자동화 컨트롤 알고리즘(Automated Control Algorithm)

자동화 컨트롤 알고리즘은 두 가지 주요 기능을 가진다: (1) 컨트롤 액션을 생성하는 것 (2) 컨트롤 액션 생성에 영향을 줄 수 있는 컨트롤드 프로세스의 상태, 외부 시스템 컴포넌트 및 환경에 대한 정확한 정보(모델)를 유지하는 것. 이러한 기능의 결함으로 인해 UCA가 발생할 수 있다.

1. 컨트롤 액션을 생성

자동화 컨트롤 알고리즘은 컨트롤 알고리즘 자체의 결함, 외부 소스에서 자동화 컨트롤러에 제공하는 안전하지 않은 명령, 또는 컨트롤드 프로세스 및 그 환경과 관련한 자동화 컨트롤러 모델의 결함으로 인해 안전하지 않은 컨트롤 액션을 생성할 수 있다. 후자의 원인은 다음 섹션에서 설명한다.

STPA는 자동화 컨트롤 알고리즘에 대한 안전 요구사항 및 제약사항을 생성한다. 물론, 안전 요구사항과 제약사항이 정확한지 반드시 확인해야 한다. 그러므로 여기서 원인 요소는 컨트롤 알고리즘의 운영 안전 요구사항 및 제약 조건에 대한 부적절한 커뮤니케이션과 검증이다.

2. UCA와 관련된 컨텍스트적인 요소(contextual factors)의 내부 모델을 유지

어떤 컨트롤 액션을 수행할지에 대한 의사 결정 안전성은 자동화 컨트롤러가 그러한 결정을 내리는 데 사용하는 정보에 영향을 받을 수 있으며, 심지어 실제 자동화 알고리즘이 안전 요구사항과 제약사항을 만족하는 경우라 할지라도 영향을 받을 수 있다. 자동화 컨트롤러는 의사결정에서 네 가지 유형의

normal takeoff state when a brake action command is given. The causal scenarios will include the reasons for the BSCU Autobrake being confused about the aircraft state and the reasons that a brake command might be given under these conditions.

Model of the Controlled Process

The model of the controlled process is the state that the automated controlled thinks that the controlled process is in. This model may include information about the environment in which the controlled process is operating. The model is updated by the controller, which we will define as part of the control algorithm (if it is separate, that won't matter[39]). The automated control algorithm gets direct information about that state from sensors that are designed to measure controlled process variables. There may also be designs where the process model is automatically updated without going through the control algorithm. An example is the loading of specific flight data right before launch of the Space Shuttle. Missile systems also do this. Appendix B of *Engineering a Safer World* details a very costly accident arising from errors (typos) in a software flight data load.

If the control algorithm is separated into several pieces (such as handling of regular braking and automated braking), there are various ways that updating of the controlled process model may be flawed through unknown or unconsidered interactions among them. These cases should be considered when identifying scenarios that lead to UCAs.

In updating the controlled process model, the control algorithm may receive indirect information about the state of a controlled process or make assumptions about it. One common assumption is that when the control algorithm issues a control action, it may assume the action is executed and update its process model. For example, the Electronic Flight Control System on an aircraft issues a control command for an actuator to move the elevator a certain number of degrees. It could then assume both that the elevator has moved and that it has moved by the specified amount. Reasons for the commanded movement not occurring could involve flaws in the communication links to the actuators, inadequate actuator operation, and failures or faults in the controlled process (the elevator itself). Feedback about the non-execution of the command may not be included in the system design or may be included in the design but not received or not processed by the automated controller. Various types of technical design issues may also be involved such as latency (for example, the time between when an input is received and when it is processed, which will be impacted by software architecture design decisions such as the use of interrupts or polling). Data age problems have also led to hazards, that is, data is used that is no longer valid or output commands are delayed in their execution to the point where they are hazardous. The latter can occur because of delays in the actuation path or because of conditions in the controlled process. For example, on one military aircraft, a maintenance worker was working on the bomb bay door and activated a mechanical interlock to prevent the door from closing while he was in its path. At the same time, other people working in the cockpit issued a command to close the bomb bay door. The command did not execute (because of the mechanical interlock) until several hours later when the maintenance worker released the mechanical interlock. The worker was killed.

There are many other reasons for inconsistencies between the process model and the actual process state (and between models in various controllers). One possible reason is delays in updating process models, which can lead to inconsistencies and unsafe control actions. In initial startup, designers may have assumed default values that are not correct in some cases. For example, some aircraft automation

[39] It may be useful to separate the various functions of the automated controller software, but they are combined here. One reason to separate them is that the design of the software will be easier to implement or change if it is separated logically. It might also be easier to verify that the software is correct.

정보를 사용한다: (1) 컨트롤드 프로세스의 상태, (2) 컨트롤드 프로세스의 다른 컨트롤러 상태, (3) 자동화 컨트롤러의 운영모드, (4) 휴먼 컨트롤러 상태(매우 복잡한 몇몇 새로운 시스템에서). 이러한 모델에서의 다양한 유형의 결함이 자동화 컨트롤러에 의한 안전하지 않은 컨트롤 액션을 야기할 수 있다. UCA의 컨텍스트는 모델의 어떠한 결함이 UCA를 야기할 수 있는지에 대한 정보를 포함한다. 예를 들어 BSCU 오토브레이크가 정상적인 이륙 중 브레이크 컨트롤 액션을 제공한다는 UCA가 있을 수 있다. BSCU가 컨트롤드 프로세스의 상태를 혼동하게 되면, 즉, 브레이크 액션 명령이 내려졌을 때 항공기가 정상적인 이륙 상태에 있다는 것을 모른다면 UCA가 발생할 수 있다. 원인 시나리오에는 BSCU 오토브레이크가 항공기 상태를 혼동한 이유와 이러한 조건에서 브레이크 명령이 내려진 이유가 포함된다.

컨트롤드 프로세스의 모델

컨트롤드 프로세스의 모델은 자동화 컨트롤러가 생각하는 컨트롤드 프로세스의 상태이다. 이 모델에는 컨트롤드 프로세스가 작동하는 환경에 대한 정보가 포함될 수 있다. 이 모델은 컨트롤러에 의해 업데이트 되는데 우리는 이를 컨트롤 알고리즘의 일부로 정의한다(분리되어 있는 경우에는 상관없음[39]). 자동화 컨트롤 알고리즘은 컨트롤드 프로세스 변수를 측정하도록 설계된 센서로부터 상태 정보를 직접 얻는다. 또한 컨트롤 알고리즘을 거치지 않고 프로세스 모델이 자동으로 업데이트 되도록 설계될 수도 있다. 예를 들면 우주 왕복선 발사 전에 특정 비행 데이터를 불러오는 경우이다. 미사일 시스템도 마찬가지이다. *Engineering a Safer World*의 부록 B에서는 소프트웨어 비행 데이터 로드 중 오류(오타)로 인해 발생한, 비용 손실이 매우 컸던 사고에 대해 자세히 설명하고 있다.

컨트롤 알고리즘이 몇 가지로 구분되는 경우(보통의 브레이킹과 오토브레이킹을 구분하는 것과 같이), 알려지지 않았거나 고려되지 않은 업데이트 간의 상호작용에 의해 컨트롤드 프로세스 모델의 업데이트에 결함이 발생하는 많은 상황들이 있다. UCA로 이어지는 시나리오를 식별할 때 이러한 경우를 반드시 고려해야 한다.

컨트롤드 프로세스 모델을 업데이트 할 때 컨트롤 알고리즘은 컨트롤드 프로세스 상태에 대한 간접적인 정보를 받거나 컨트롤드 프로세스 상태를 가정할 수 있다. 흔한 가정 중 하나는, 컨트롤 알고리즘이 컨트롤 액션을 내보내는 시점에 액션이 실행되고 해당 프로세스 모델이 업데이트 된다는 것이다. 예를 들어 항공기의 전자 비행 제어 시스템(Electronic Flight Control System)에서 액추에이터가 승강타를 일정 각도만큼 움직이게 하는 컨트롤 명령을 내보낸다 하자. 그렇다면 승강타가 움직였고 지정된 양만큼 움직였다고 가정할 수 있다. 명령된 움직임이 발생하지 않은 이유는 액추에이터로의 커뮤니케이션 링크 결함(flaw), 액추에이터의 부적절한 작동, 컨트롤드 프로세스(승강타 자체)의 고장(failure)이나 결함(fault) 등이 있을 수 있다. 시스템 설계 시 명령이 실행되지 않은 것에 대한 피드백을 설계하지 않았을 수 있으며 설계에 포함됐지만 자동화 컨트롤러가 피드백을 수신하지 못했거나 처리하지 못했을 수 있다. 지연과 같은 다양한 유형의 기술적 설계 문제가 있을 수도 있다(지연의 예로, 입력 수신 시점과 처리 시점 사이의 시간 차이가 있으며 이는 인터럽트나 폴링(polling)과 같은 소프트웨어 아키텍처 설계 결정에 의해 영향을 받음). 데이터 수명 문제 또한 위험으로 이어질 수 있다. 즉, 더 이상 유효하지 않은 데이터를 사용하거나 출력 명령이 위험한 상태에 이르는 시점까지 지연되는 경우이다. 후자는 실행 경로(actuation path)에서의 지연이나 컨트롤드 프로세스의 상태로 인해 발생할 수 있다. 예를 들어 유지보수 작업자가 군용 항공기 폭탄 투하실 문에서 작업을 하고 있었고 그가 문이 닫히는 경로에 있

[39] 자동화 컨트롤러 소프트웨어의 다양한 기능을 분리하는 것이 유용할 수 있으나, 여기서는 분리하 지 않았다. 기능을 분리하는 한 가지 이유는, 논리적으로 분리되어 있으면 소프트웨어의 설계를 구현하거나 수정하기 쉽다는 것이다. 또한 소프트웨어가 올바른지를 확인(verify)하기도 쉽다.

is supposed to be powered up and initialized before takeoff. Default values are often provided for that case. However, if the device is started up later than takeoff, the default values may not hold. TCAS, for example, assumes that it has been started while the aircraft is on the ground and initializes the system with values corresponding to being on the ground. The same is true of process models for devices that are shut down for maintenance and restarted with perhaps an assumption that the controlled process state is the same as when the system was shut down for the maintenance procedures. Another example is that human controllers may decide to restart a device when it does not appear to be working correctly. TCAS, for example, allows pilots to reboot it in the air. The device, however, restarts with initialization values that may not be correct at that point in the flight. Input to update the process model may also arrive before the device is powered up, after shut down, or while the device is disconnected (off-line) and therefore may be ignored.

Model of the Operational Modes

The model of the operational modes may also be related to the context defined in the UCAs. There are four types of modes that must be considered: (1) the controlled process mode, (2) the automation mode, (3) the supervisory mode, and the display mode.

1. Controlled Process Mode

The controlled process mode identifies the operating mode of the controlled process. For example, an aircraft (controlled process) may be in takeoff mode, landing mode, or cruise mode. The requisite modes to model will depend on the contexts identified for the UCAs. For example, if the aircraft is in cruise mode (rather than in landing mode), operation of the reverse thrusters can be unsafe. That context should be identified in the UCA context table (i.e., operation of the reverse thrusters when the aircraft is not in landing mode). The reasons (causal scenarios) for the UCA will be at least partially related to the automation being confused about the current mode of the controlled process.

2. Basic Automation Operating Mode

In addition to the controlled process, the automation itself (automated controlled in Figure G.2) has modes of operation. Those modes will often be used in the logic of its control algorithm and affect its outputs. Examples include nominal (normal) behavior mode, shutdown mode, and fault-handling mode. As an example, the automation may be in partial shutdown mode after detecting a fault in itself or in the controlled process. In the Air France 447 loss, the autopilot shut itself off (went into shutdown mode) when a pressure probe (pitot tube) on the outside of the aircraft iced over and the autopilot could no longer tell how fast the aircraft was moving. The fly-by-wire system, at the same time, continued operating but switched into a mode where it no longer provided protection against aerodynamic stall. On modern Airbus aircraft, operational modes of the electronic flight control system include *normal*, *alternate*, *direct*, and *mechanical laws*, each of which changes the behavior of the automated flight control system.

Partial shutdown modes in automated controllers can also be very confusing to human controllers, which will be covered later under the things that can go wrong with human controllers. Mode confusion arises when the mode of the controlled system is different than the controller thinks it is. Mode confusion has contributed to many accidents and must be considered when generating causal scenarios.

In fault-handling mode, the automated controller may change its behavior because it believes there is a fault in the controlled process or in itself and different behavior is necessary to provide safe control.

3. Supervisory Mode

This mode identifies who or what is controlling the automation at any time, when multiple supervisors may be in control of (provide control commands) to the automated controller. Basically, the

을 때에는 문이 닫히지 않도록 기계적 인터락을 활성화하였다. 그와 동시에 조종실에 있는 다른 작업자가 폭탄 투하실의 문을 닫는 명령을 내렸다. 이 명령은 유지보수 작업자가 기계적 인터락을 해제할 때까지 몇 시간 동안은 실행되지 않았지만(기계적 인터락 때문에) 인터락 해제 후 그 작업자는 사망하였다.

프로세스 모델과 실제 프로세스 상태 간(그리고 다양한 컨트롤러 모델 간)의 불일치에는 많은 이유가 있다. 가능한 이유 중 하나는 불일치와 안전하지 않은 컨트롤 액션을 야기할 수 있는 프로세스 모델의 업데이트 지연이다. 초기 시작 시 설계자가 특정 상황에서 기본값(default values)이 잘못되었다고 가정했을 수 있다. 예를 들어 일부 항공기 자동화 컨트롤러는 이륙 전에 전원을 켜고 초기화하도록 되어 있는데 이 때 종종 기본값이 사용된다. 그러나 그 장치가 이륙 이후에 시작되면 기본값이 유효하지 않을 수 있다. 예를 들면 TCAS가 항공기가 지상에 있을 때 TCAS 기동을 시작했을 것이라고 가정하고 지상에 있을 때에 해당하는 값으로 시스템을 초기화시키는 경우이다. 이는 유지보수를 위해 시스템을 종료한 시점과 장치를 재시작한 시점의 컨트롤드 프로세스 상태가 동일할 것이라는 가정 하에, 시스템을 유지보수 한 후에 재시작하는 프로세스 모델의 경우도 마찬가지이다. 또 다른 예로, 휴먼 컨트롤러는 장치가 제대로 동작하지 않는 것처럼 보이면 장치를 다시 시작할 수도 있다는 것이다. 예로, 조종사는 비행 중일 때 TCAS를 재부팅 할 수 있다. 그렇게 되면 장치는 비행 중인 시점에서는 정확하지 않을 수도 있는 초기값으로 재시작하게 된다. 프로세스 모델을 업데이트하기 위한 입력이, 장치 전원이 켜지기 전이나 장치가 종료된 후 또는 장치의 연결이 끊어져 있을 때(오프라인) 들어와 무시될 수도 있다.

운영 모드의 모델

운영 모드의 모델 또한 UCA에서 정의된 컨텍스트와 관련이 있을 수 있다. 고려해야 할 모드는 4가지이다: (1) 컨트롤드 프로세스 모드, (2) 자동화 모드, (3) 감독 모드, (4) 디스플레이 모드

1. 컨트롤드 프로세스 모드(Controlled Process Mode)

컨트롤드 프로세스 모드는 컨트롤드 프로세스가 운영되는 모드를 식별한다. 예를 들어, 항공기(컨트롤드 프로세스)는 이륙 모드, 착륙 모드 또는 운항 모드가 있다. 모델링에 필요한 모드는 UCA에서 식별된 컨텍스트에 따라 달라진다. 예를 들어 항공기가 착륙 모드가 아닌 운항 모드에 있으면 역추진력 장치의 작동이 안전하지 않을 수 있다. 이 컨텍스트는 UCA 컨텍스트 테이블에서 식별되어야 한다(즉, 항공기가 착륙 모드가 아닐 때 역추진력 장치의 작동). UCA에 대한 이유(원인 시나리오)는 적어도 컨트롤드 프로세스의 현재 모드를 혼동하는 자동화와 부분적으로 관련이 있다.

2. 기본 자동화 운영 모드(Basic Automation Operating Mode)

컨트롤드 프로세스뿐만 아니라 자동화 자체(그림 G.2의 자동화 컨트롤러)에도 운영 모드가 있다. 이러한 모드는 종종 컨트롤 알고리즘 로직에 사용되어 출력에 영향을 미친다. 예로, 정상(nominal, normal) 동작 모드, 종료 모드 및 오류 처리 모드가 있다. 일례로, 자동화가 자체적으로 또는 컨트롤드 프로세스를 통해 결함을 탐지하게 되면 부분 종료 모드가 될 수 있다. 에어프랑스 447 사고의 경우, 항공기 외부의 압력 프로브(피토관, pitot tube)가 얼고 항공기가 얼마나 빨리 움직이는지를 자동조종장치가 더 이상 알 수 없게 되면 자동조종장치는 스스로 종료한다(종료 모드로 들어감). 이 때 전기신호식 비행 제어 시스템(fly-by-wire system)이 계속 작동했지만 더 이상 공기 역학적 스톨(aerodynamic stall)에 대한 보호 기능을 제공하지 않는 모드로 전환되었다. 최신 에어버스 항공기의 전자 비행 제어 시스템 운영 모드에는 *정상(normal), 대체(alternative), 직접(direct)* 및 *기계 법칙(mechanical laws)*이 있으며, 각 모드에 따라 자동 비행 제어 시스템의 동작이 달라진다.

자동화 컨트롤러의 부분 종료 모드는 휴먼 컨트롤러에게는 매우 혼란스러울 수 있는데 휴먼 컨트롤

supervisory mode allows coordination of control action implementation among multiple supervisors. For example, a flight guidance system in an aircraft may be issued direct commands by a human pilot or may receive commands from other system components, which could be either human or automation. While Figure G.2 abstracts all automated controllers into one box for generality, there may be and usually is more than one automated controller. One controller may control multiple others in a hierarchical arrangement, or multiple controllers may operate in parallel, or both. This, of course, can get very complicated, e.g., the multiple controllers may each be supervised by different controllers, etc.

4. Display Mode

This model specifies what type of information should be displayed to a human controller of the automated controller. The current display mode will affect both the information provided as well as how the user will interpret it. One of the components of the model of a human controller in Figure G.2 is the current display mode he/she is seeing. If these display modes (in the human and the automation) become inconsistent, then serious problems can result, i.e., the behavior on the part of one or both can lead to a system hazard. This type of mode confusion is called "Interface Interpretation Mode Confusion" and is described further in Chapter 9 of *Engineering a Safer World*.

Model of the Human Controller

In some systems today, a model of the state of the human controller, such as drowsiness or inattention, is used by the automated control algorithm to determine its behavior. Increasingly, sensors are being used to gauge the state of human controllers. For example, cars try to detect whether the driver is intoxicated or inattentive through various types of sensors that provide input to the automated controller. A dotted line from the sensor to the human controller is shown in Figure G.2 to denote a design where the human gets feedback about whether the sensor is working or what the state of the sensed variables are (e.g., the driver's intoxication level).

Model of Other Controllers

If the automation has multiple potential controllers, then the automation may need to have a model of important state variables in those controllers.

Environmental Inputs

There may be inputs from the environment that go directly to the automated controller and not through the human controller. For example, some automation today gets direct information about location from GPS. GPS location information is calculated from signals in the environment (e.g., through antennas) and goes into the automated controller, not through the human controller. In the newest aircraft, there is something called ADS-B (Automatic Dependence Surveillance) which can be used to communicate information between different aircraft, again without going through the human controller. Note that this does not include information about the state of the aircraft itself (e.g., input from pitot tubes) which is sent via sensors in or on the controlled process and is part of the model of the controlled process.

Other controllers/systems

In some complex systems today and certainly increasingly in future systems, there may be multiple controllers of the automation. For example, there may be the usual human control of the aircraft automation with perhaps some control of the aircraft automation outside the aircraft itself (usually on the ground but it could also reside on another aircraft to which the automated aircraft is tethered), which may be human or automated. The control responsibilities may be shared by different controllers.

러에 의해 잘못될 수 있는 경우는 나중에 다루도록 한다. 모드의 혼동은 제어되는 시스템의 모드가 컨트롤러가 생각하는 모드와 다를 때 발생한다. 모드 혼동으로 인해 많은 사고(accident)가 발생했으므로 원인 시나리오 생성 시 반드시 고려해야 한다.

오류 처리 모드에서 자동화 컨트롤러의 동작을 변경할 수 있다. 왜냐하면 자동화 컨트롤러는 컨트롤드 프로세스나 컨트롤러 자체에 오류가 있다고 생각하여 안전한 컨트롤을 제공하기 위해 다른 동작이 필요하다고 판단하기 때문이다.

3. 감독 모드(Supervisory Mode)

이 모드는 자동화 컨트롤러를 여러 감독자들이 제어할(제어 명령을 제공할) 때 누가 또는 무엇이 자동화를 제어하고 있는지를 식별한다. 기본적으로 감독 모드는 여러 감독자 간의 컨트롤 액션 시행을 조정한다. 예를 들어 항공기의 비행 유도 시스템(flight guidance system)은 조종사로부터 직접 명령을 받거나, 사람 또는 자동일 수 있는 다른 시스템 컴포넌트로부터 명령을 받을 수 있다. 그림 G.2에서는 일반화를 위해 모든 자동화 컨트롤러를 하나의 상자로 추상화했지만 일반적으로 하나 이상의 자동화 컨트롤러가 있을 수 있다. 하나의 컨트롤러가 계층적 배열에서 여러 다른 컨트롤러를 제어할 수 있고 여러 컨트롤러가 병렬로 동작할 수 있으며 둘 다 일수도 있다. 물론 이것은 매우 복잡할 수 있는데 예를 들어 다중 컨트롤러(multiple controller)가 각각 다른 컨트롤러 등에 의해 감독될 수도 있다.

4. 디스플레이 모드(Display Mode)

이 모드는 자동화 컨트롤러 중 휴먼 컨트롤러에게 어떤 유형의 정보를 표시해야 하는지를 지정한다. 현재 디스플레이 모드는 제공된 정보 및 사용자가 이를 해석하는 방법에 모두 영향을 미친다. 그림 G.2의 휴먼 컨트롤러 모델 컴포넌트 중 하나는 휴먼 컨트롤러가 보고 있는 현재 디스플레이 모드이다. 이러한 디스플레이 모드(사람과 자동화에서)가 일치하지 않으면 심각한 문제가 발생할 수 있다. 즉, 사람과 자동화 중 하나 또는 둘 모두의 동작이 시스템 위험을 초래할 수 있다. 이러한 유형의 모드 혼동을 “인터페이스 해석 모드 혼동(Interface Interpretation Mode Confusion)”이라고 부르며 *Engineering a Safer World*의 9장에 자세히 설명되어 있다.

휴먼 컨트롤러의 모델

오늘날 몇몇 시스템에서는 자동화 컨트롤러의 동작을 결정하기 위해 자동화 컨트롤 알고리즘에 졸음이나 부주의와 같은 휴먼 컨트롤러의 상태 모델을 사용한다. 휴먼 컨트롤러의 상태를 측정하기 위해 점점 더 많은 센서가 사용된다. 예를 들어 자동차는 운전자의 음주나 부주의를 탐지하기 위해 자동화 컨트롤러에 입력을 제공하는 다양한 유형의 센서를 사용한다. 그림 G.2에는 센서에서 휴먼 컨트롤러로 점선이 그려져 있는데 이는 센서의 작동 여부나 감지된 변수의 상태가 어떤지를(예: 운전자의 음주 수준) 사람에게 피드백 해주는 설계를 나타낸다.

다른 컨트롤러의 모델

만일 자동화에 여러 개의 잠재적 컨트롤러가 있다면 그 자동화에는 해당 컨트롤러의 중요 상태 변수 모델이 필요할 수 있다.

환경으로부터의 입력

휴먼 컨트롤러를 통하지 않고 환경에서 자동화 컨트롤러로 직접 들어오는 입력이 있을 수 있다. 예를 들어, 오늘날 일부 자동화는 GPS로부터 위치 정보를 직접 받는다. GPS 위치 정보는 환경 내의 신호(예: 안테나를 통해)로부터 계산된 후 휴먼 컨트롤러를 거치지 않고 자동화 컨트롤러로 바로 입력된다. 최신 항공기에는 다른 항공기간 정보를 전달할 때 사용되는 ADS-B (Automatic Dependence Surveillance)라는 것이 있는데, 이것 역시 휴먼 컨트롤러를 거치지 않는다. 컨트롤드 프로세스 내에서

Figure G.2 shows only one controller (in tan) and the possibility of others in a box connected to the automation as each of the controllers will have the same internal details.

As mentioned when describing the operational mode, the automation may need to know what controller is controlling it at any time. With shared control comes the possibility of conflicting commands. Some confusion may result from inconsistencies between the models in the automation and the models in its controllers and lead to UCAs. The possible types of inconsistencies and confusion can provide important contributions to causal scenario generation.

Transmission of information between the automation and its supervisor(s)

Human controllers transmit control commands via various types of controls and get feedback from the automated controller through displays. Once again, flawed transmission of information between the automated controller and its controllers will be an important (although straightforward) part of the causal scenario generation process.

Direct changes to the automated controller that do not go through the control algorithm

In some systems today, and certainly increasingly in future systems, the control algorithm and other software, as well as all the internal models used by the automation's control algorithm may be directly changeable from the outside without the human controller knowing about it. For example, on some aircraft, new software may be updated on an aircraft over the internet without the pilot or even the airline's maintenance personnel being involved (for example, the changes may originate with an OEM). The changes may affect both the control algorithm itself and the models used by it, such as maps and other information in the automated controller and indirectly (perhaps through displays) by human controllers. There are clearly potential safety and especially security issues here. Such changes may provide the potential for generating the contextual factors in a UCA.

Human Controllers

Human controllers are the most complex component of the model in Figure 1 (although system designers are quickly increasing the complexity of automated controllers) and thus provide some of the most important parts of the causal scenario generation process. There are many relevant components of the model that will contribute, including control action generation and mental processing and the mental models related to these functions.

Control Action Generation/Mental Processing

Although these two functions could be modeled as different components, they have important interactions that can be lost in such a separation. Therefore they are combined here. This component has two basic functions: generating control actions and updating mental models.

<u>Generate control actions</u>:

Control actions are generated using information from all the models in the human controller noted in Figure 1 (environment, automation, and controlled process) as well as training, written procedures, and experience. Control action generation will be affected by commands from other controllers with which the human operator interacts. For example, a pilot is partially controlled by control actions or information received from the airline operations center or from other controllers, such as Air Traffic Control or onboard instruments and systems such as TCAS (Traffic Alert and Collision Avoidance

센서를 통해 전송되거나 컨트롤드 프로세스로 전송되는 (컨트롤드 프로세스 모델의 일부인) 항공기 자체의 상태 정보(예: 피토관으로부터의 입력)는 서로 전달되는 정보에는 포함되지 않는다는 점에 유의한다.

다른 컨트롤러/시스템

오늘날 일부 복잡한 시스템과 미래의 시스템에서는 분명히, 점점 더 많은 컨트롤러가 자동화될 수 있다. 예를 들어, 일반적으로 사람이 제어할 수 있는 항공기 자동화가 있을 수 있고 항공기 외부(사람 또는 자동화)에서도 항공기를 제어할 수 있다(보통 지상에서 이루어지지만 테더링된 다른 항공기에서 제어할 수도 있음). 제어의 책임은 여러 컨트롤러가 공유하여 가질 수 있다. 그림 G.2에서는 자동화 컨트롤러에 연결된 하나의 컨트롤러(황색 상자)만을 보여주지만, 동일한 내부 요소을 가지고 있는 다른 컨트롤러도 있을 수 있다.

운영 모드 설명 시 언급한 바와 같이 자동화는 항상 어떤 컨트롤러가 컨트롤하는지를 알아야 한다. 공유 컨트롤(shared control)을 사용하면 명령 간 충돌이 발생할 수 있다. 자동화 모델과 컨트롤러 모델 간의 불일치로 인해 혼란이 생길 수 있으며 이로 인해 UCA가 발생할 수 있다. 가능한 불일치 유형과 혼동은 원인 시나리오 생성에 중요한 역할을 할 수 있다.

자동화와 감독자 간의 정보 전송

휴먼 컨트롤러는 다양한 유형의 컨트롤을 통해 컨트롤 명령을 전송하고 디스플레이를 통해 자동화 컨트롤러로부터 피드백을 받는다. 다시 말하지만, 자동화 컨트롤러와 컨트롤러 사이의 정보 전송 결함은 원인 시나리오 생성 프로세스의 중요한(간단함에도 불구하고) 부분이다.

컨트롤 알고리즘을 거치지 않는 자동화 컨트롤러의 직접적인 변경

오늘날 몇몇 시스템을 포함하여 미래의 시스템에서 분명 확실한 것은, 컨트롤 알고리즘 및 다른 소프트웨어뿐만 아니라 자동화 컨트롤 알고리즘에 사용되는 내부 모델까지도 휴먼 컨트롤러가 인지하지 못한 채 외부에서 직접 변경할 수 있다는 것이다. 예를 들어 일부 항공기에서는 조종사나 항공사 유지보수 인력 없이도 인터넷을 통해 항공기에 새로운 소프트웨어가 업데이트 될 수 있다(예: 변경사항이 OEM에서 시작될 수 있음). 변경사항은 컨트롤 알고리즘 자체에 영향을 줄 수 있고 자동화 컨트롤러의 지도 및 다른 정보를 사용하는 모델에도 영향을 줄 수 있으며 휴먼 컨트롤러에 의해 간접적으로 (어쩌면 디스플레이를 통해) 사용되는 모델에도 영향을 줄 수 있다. 여기에는 분명 잠재적인 안전 이슈가 있으며 특히 보안의 이슈가 있다. 이러한 변경사항들이 UCA에서 컨텍스트적 요소를 생성할 수 있을 것이다.

휴먼 컨트롤러(Human Controllers)

휴먼 컨트롤러는 그림 G.2의 모델 중 가장 복잡한 컴포넌트이므로(시스템 설계자가 자동화 컨트롤러의 복잡도를 빠르게 증가시키고 있긴 하지만) 원인 시나리오 생성 프로세스에서 가장 중요한 것을 제공한다. 여기에는 컨트롤 액션 생성, 멘탈 프로세싱, 그리고 이러한 기능과 관련된 멘탈 모델을 포함하여 모델의 연관된 많은 컴포넌트가 있다.

컨트롤 액션 생성/멘탈 프로세싱

비록 이 두 가지 기능들은 다른 컴포넌트로 나누어 모델링 할 수 있지만 이렇게 분리하여 모델링할 경우 중요한 상호작용이 손실될 수 있다. 그러므로 여기에서는 이 둘을 한꺼번에 다룬다. 이 컴포넌트에는 컨트롤 액션 생성과 멘탈 모델 업데이트라는 두 가지의 기본적인 기능이 있다.

System).[40] When there is unclear delineation of responsibilities for control, coordination challenges increase as do the causal scenarios for unsafe control actions. For example, the FAA defines the relationship between pilots and airline operations centers as 50/50 with respect to making some specific decisions related to aircraft control. Without careful delineation of responsibilities (which may be ultimately the responsibility of the Director of Operations for the airline), confusion about who is making the decisions can lead to hazards and accidents (and this has happened).

There are, of course, a very large number of incorrect behaviors possible during control action generation. The UCA generation process (the results of which are documented in the UCA tables) to identify the context in which control actions are unsafe, along with the process used by the human controller to identify the current context, will serve to limit the number of causal scenarios generated. During the scenario generation process, the analyst will need to determine how the identified context associated with the UCA could occur and why the controller might be confused about the current context.

The human controller can in some systems interact directly with the controlled process (shown by dotted lines between the human controller and the actuators and sensors and the controlled process) without going through the automation.

Update mental models:

Besides generating control actions, a major responsibility of human controllers is to update their mental models. Mental models are always updated through processing by the human mind, although some of this processing may be subconscious. Thus there are two-way arrows in the model (humans both update and use information in their mental models and things can go wrong in both functions).

The information the human uses in this updating is partially obtained through the displays, partially through inputs from its controllers or other controllers with which it interacts (e.g., information transmitted from air traffic control or the airline operations center), and partially from direct sensory input to the controller (e.g., looking out the window and seeing a tornado approaching or feeling turbulence on an aircraft). For example, a pilot receives information (such as NOTAMs) sent from the airline operations center or from other controllers, such as Air Traffic Control. In some systems, the human operators have the ability to directly observe the actuators operating, which is depicted in Figure G.2 by a dotted line from the actuator to the human controller. For example, in some plants operators may be able to walk up to the actuators and valves and watch them open and close to check for proper operation. As another example, pilots usually do a walk-around before taking off or during a pre-flight inspection they may move the controls and directly watch the aircraft ailerons to ensure they move in the correct direction. Human controllers may also receive input from the state of the controls, for example, the driver may see the steering wheel move on its own and assume that the automation has issued a control command related to steering.

Model updating may also be affected by the training that the human operator (controller) receives. Another subtle type of input may come from the reaction of the controls when the pilots operate them (shown as a dotted line in Figure G.2 from the controls to the human controller). Operators may make assumptions about the state of the automated controller and controlled process based on the

[40] While TCAS could be considered to be an automated controller (in the middle of the hierarchical control structure in Figure G.2), pilots are required to follow TCAS resolution advisories (control commands) and thus in some respects TCAS controls the pilots. However, pilots can input settings for TCAS to control how it generates resolution advisories. As more and more functions are automated on aircraft and other systems, various types of "partnerships" between humans and automation will be necessary and coordination challenges will increase.

컨트롤 액션 생성:

컨트롤 액션은 훈련, 서면 절차, 경험뿐만 아니라 그림 G.2에서 언급한 휴먼 컨트롤러의 모든 모델에서 얻어진 정보(환경, 자동화 및 컨트롤드 프로세스)를 사용하여 생성된다. 컨트롤 액션 생성은 운영자가 상호작용하는 다른 컨트롤러의 명령에 영향을 받는다. 예를 들어 조종사는 항공사 운영센터 또는 다른 컨트롤러(ATC 또는 TCAS와 같은 온보드 계측기/시스템)로부터 받은 정보나 컨트롤 액션에 의해 부분적으로 컨트롤 된다[40]. 컨트롤에 대한 책임이 명확하게 기술되어 있지 않은 경우 UCA에 대한 원인 시나리오와 마찬가지로 조정(coordination) 문제가 증가한다. 예를 들어 FAA는 항공기 컨트롤과 관련된 특정 결정을 내리는 것에 대해 조종사와 항공사 운영센터 간의 관계를 50/50으로 정의한다. 책임에 대한 상세한 설명이 없으면(항공사 운영 책임자의 궁극적인 책임일 수 있음) 누가 결정을 내리는지에 대한 혼란으로 인해 위험과 사고가 발생할 수 있다.

물론, 컨트롤 액션을 만드는 과정에서 잘못된 동작이 매우 많이 발견될 수 있다. 휴먼 컨트롤러가 현재의 컨텍스트를 식별하기 위해 사용한 프로세스와 마찬가지로, 컨트롤 액션을 안전하지 않게 하는 컨텍스트를 식별하기 위한 UCA 생성 프로세스(이 결과는 UCA표에 문서화됨)에서도 생성되는 원인 시나리오의 수를 제한한다. 시나리오 생성 프로세스 과정에서 분석가는 UCA와 관련 있는 식별된 컨텍스트가 어떻게 발생할 수 있는지, 왜 컨트롤러가 현재 컨텍스트에 대해 혼란을 일으킬 수 있는지에 대해 알아낼 필요가 있다.

휴먼 컨트롤러는 일부 시스템에서 자동화를 거치지 않고 컨트롤드 프로세스(휴먼 컨트롤러, 액추에이터, 센서, 컨트롤드 프로세스 사이의 점선으로 표시됨)와 직접 상호작용할 수 있다.

멘탈 모델 업데이트:

컨트롤 액션을 생성하는 것 외에, 휴먼 컨트롤러의 중요한 책임은 자신의 멘탈 모델을 업데이트하는 것이다. 멘탈 모델 업데이트 중 일부는 무의식적일 수 있지만 멘탈 모델은 사람의 마음에 따라 항상 업데이트 된다. 따라서 이 모델에는 화살표가 양방향으로 있다(사람들은 그들의 멘탈 모델 정보를 업데이트 및 사용하며 이 두 기능(업데이트, 사용) 모두 잘못될 수 있음).

이 업데이트에서 사람이 사용하는 정보 중 일부는 디스플레이를 통해 얻고 일부는 그 컨트롤러나 상호작용하는 다른 컨트롤러의 입력(예: ATC 또는 항공사 운영센터에서 전송된 정보)을 통해서 얻으며 또 일부는 직접적인 감각 입력(예: 창밖을 통해 토네이도를 보거나 항공기에서 난기류를 느끼는 것)을 통해서 얻는다. 예를 들어 조종사는 항공사 운영센터 또는 ATC와 같은 다른 컨트롤러에서 보낸 정보(예: NOTAM)를 받는다. 일부 시스템에서는 운영자가 액추에이터의 운영 상황을 직접 볼 수 있는데 이것은 그림 G.2에서 액추에이터부터 휴먼 컨트롤러까지의 점선으로 나타나 있다. 예시로, 일부 공장의 운영자는 액추에이터 및 밸브까지 걸어가서 그것이 열리고 닫히는 것을 관찰하며 적절히 동작되는지 확인할 수 있다. 다른 예로, 조종사는 일반적으로 이륙 전 주변을 돌아보거나 비행 전 점검(pre-flight inspection) 과정에서 조종 장치를 움직여서 항공기의 에일러론이 정확한 방향으로 움직이는지를 직접 눈으로 확인한다. 휴먼 컨트롤러는 조종 장치의 상태로부터 입력을 받을 수도 있는데, 예를 들어 운전자는 스티어링 휠이 움직이는 것을 보고 자동화가 스티어링과 관련된 컨트롤 명령을 보낸 것을 유추할 수 있다.

---

[40] TCAS는 자동화 컨트롤러로 간주될 수 있지만(그림 G.2의 계층적 컨트롤 스트럭처 중간에 있음), 조종사는 TCAS 대응 지침(resolution advisories)(컨트롤 명령)을 따라야하므로 어떤 면에서는 TCAS가 파일럿을 컨트롤하는 것이다. 그러나 조종사는 TCAS가 대응 지침을 생성하는 방법을 컨트롤하기 위해 설정값을 입력할 수 있다. 항공기 및 다른 시스템에서 점점 더 많은 기능이 자동화됨에 따라 사람과 자동화 간에 다양한 유형의 "파트너십"이 필요하며 둘 간의 조정 문제가 점점 증가할 것이다.

movement of controls, such as the movement of the control column on an aircraft or the steering wheel on an automated car.

Interactions between updating models and generating control actions:

As with automated controllers and their updating of their process models, there can be important interactions between the control generation process and the mental models of human controllers similar to those that were described for automated controllers. For example, if a human controller issues a command to "Do X," human controllers are likely to assume that the "Do X" command has been executed by the Automated Controller and the actuators for the Controlled Process and unconsciously update their mental models about the state of the automation and/or the controlled process. If, in reality, the "Do X" command is never executed, then the process models of the human controller will be inconsistent with the actual state of the automation and/or controlled process. There can be many reasons why the command may not be executed, for example, it may be blocked and ignored if the automation considers the command to be unsafe or not possible at that point in time or there can be faults or failures in the controlled process.

This same type of flawed updating of process models was described in the section on the automated control algorithm. But in the case of the automation, the algorithm can be checked to ensure that this problem does not occur. It is more difficult to preclude this from human mental processing.

Feedback about the non-execution of the command may be provided to the human controller, but the human may not notice it because of distraction or other reasons.

Mental Models used by Human Controllers

Four types of mental models are identified in Figure G.2 for the human controller: a model of the environment, a model of the state of the automation, a model of the controlled process, and a model of other controllers. Any of these models may be inconsistent with the actual state and thus lead to unsafe control actions. These models have similar content and use as the same models in the automated controller, although the reasons for their being flawed may be quite different. The causal scenarios must include potential reasons for (causes of) dangerous inconsistencies that can give rise to the contextual conditions in the UCAs.

Because the human controller is usually controlling both the automation (directly) and the controlled process (indirectly but in some cases directly), humans need models of the states of both.

In order to supervise automation, humans need to have basic understanding of how the automation operates. Confusion about this can lead to unsafe control and losses.

There are a large number of reasons why human mental models may be incorrect. Some of the same reasons given earlier for models being incorrect in automated controllers may apply. But human processing presents many other opportunities. We highly recommend that human factors experts be part of the causal scenario generation involving human controllers. Here is a list of some (but only some) factors that should be considered:

- Mode confusion is a common cause of accidents in mode-rich systems where the human is confused about the mode of the automation or the automation may be confused about the current mode of the controlled process. Mode confusion may be caused by incorrect updating in process models, inputs may be incorrect or delayed, updating may be delayed or the human controller may be informed of the mode change but does not notice or process that information. When automation can change the mode of the controlled process without being directed to do so by the human controller, mode confusion and potential unsafe control actions can result.
- "Situation awareness" is commonly cited as the cause of accidents without a careful definition of what that term means. A large number of errors can fall into this category. Note that the

모델 업데이트는 운영자(컨트롤러)가 받는 훈련에 영향을 받을 수도 있다. 또 다른 미세한(subtle) 유형의 입력은 조종사가 조종 장치를 조작할 때 조종 장치의 반응에서 생길 수 있다(그림 G.2에서 조종 장치에서 휴먼 컨트롤러까지 점선으로 표시됨). 운영자는 항공기 조종대(control column)의 이동 또는 자동화 차량의 스티어링 휠과 같은 조종 장치의 움직임을 기반으로 자동화 컨트롤러 및 컨트롤드 프로세스의 상태에 대한 가정을 세울 수 있다.

모델 업데이트와 컨트롤 액션 생성 간의 상호작용:

앞서 자동화 컨트롤러에 대해 설명했듯이, 자동화 컨트롤러와 프로세스 모델 업데이트 간에 상호작용이 있는 것과 마찬가지로, 컨트롤 생성 프로세스와 휴먼 컨트롤러의 멘탈 모델 간에도 중요한 상호작용이 있을 수 있다. 예를 들어, "X를 수행"이라는 명령을 내린 휴먼 컨트롤러는 자동화 컨트롤러와 컨트롤드 프로세스의 액추에이터가 "X를 수행" 명령을 실행한다고 가정하고 자동화 컨트롤러와(또는) 컨트롤드 프로세스의 상태에 대한 그들의 멘탈 모델을 무의식적으로 업데이트한다. 만일, "X를 수행" 명령이 실제로 실행되지 않았다면 휴먼 컨트롤러의 프로세스 모델은 자동화와(또는) 컨트롤드 프로세스의 실제 상태와 일치하지 않게 된다. 명령이 실행되지 않을 수 있는 이유에는 여러 가지가 있다. 예를 들어 명령이 안전하지 않거나 해당 시점에 실행이 불가능하다고 자동화가 판단하는 경우 또는 컨트롤드 프로세스에 오류나 실패가 있는 경우 "X를 수행" 명령은 차단되고 무시될 수 있다.

프로세스 모델에 대한 이러한 유형의 잘못된 업데이트는 자동화 컨트롤 알고리즘 섹션에서 설명했다. 그러나 자동화의 경우 알고리즘을 검사하여 이런 문제가 발생하지 않도록 할 수 있다. 사람의 멘탈 프로세싱에서 이런 문제를 막는 것은 훨씬 어렵다.

명령 미수행에 대한 피드백이 휴먼 컨트롤러에게 제공될 수 있지만 주의산만이나 기타 다른 이유로 인해 사람이 이를 알아차리지 못할 수 있다.

휴먼 컨트롤러가 사용하는 멘탈 모델

그림 G.2에는 환경 모델, 자동화의 상태 모델, 컨트롤드 프로세스 모델, 다른 컨트롤러의 모델 등 4가지 유형의 멘탈 모델이 식별되어 있다. 이 모델들은 실제 상태와 일치하지 않을 수 있으며 이로 인해 안전하지 않은 컨트롤 액션이 발생할 수 있다. 결함의 이유는 많이 다르더라도 이 모델들의 내용은 비슷하며 자동화 컨트롤러에서와 동일한 모델을 사용한다. 원인 시나리오에는 UCA의 컨텍스트적인 조건을 야기하는 위험한 불일치(의 원인)의 잠재적인 이유를 반드시 포함해야 한다.

휴먼 컨트롤러는 대개 자동화(직접적인 것)와 컨트롤드 프로세스(간접적인 것이나, 경우에 따라 직접적인 것)를 모두 제어하기 때문에 사람은 둘 모두에 대한 상태 모델이 필요하다.

사람이 자동화를 감독하기 위해서는 자동화가 어떻게 작동하는지에 대한 기본적인 이해가 필요하다. 이에 대한 혼란은 안전하지 않은 컨트롤과 손실을 초래할 수 있다.

사람의 멘탈 모델이 정확하지 않은 이유는 많다. 앞서, 자동화 컨트롤러의 모델이 정확하지 않은 이유 중 일부가 동일하게 적용될 수도 있지만 사람의 처리 과정에서는 보다 다양한 이유가 있을 수 있다. 따라서 휴먼 컨트롤러에 관련된 원인 시나리오 생성 시에는 인적 요인(human factor) 전문가를 포함할 것을 강력히 권고한다. 다음은 고려해야 할 몇 가지(일부임) 요인이다.

- 모드 혼동(mode confusion)은 모드가 많은(mode-rich) 시스템에서 발생하는 사고의 공통 원인(common cause)이다. 모드가 많은 시스템에서는 사람이 자동화의 모드를 혼동하거나 자동화가 컨트롤드 프로세스의 현재 모드를 혼동할 수 있다. 모드 혼동은 프로세스 모델의 업데이트가 잘못되어 발생할 수 있다. 또한 입력이 잘못되거나 지연될 수 있고 업데이트가 지연될 수 있으며, 휴

modeling used in STPA can identify situation awareness errors as they simply mean that the human controller mental models do not match the real state.

- Humans can easily be confused by automation that is nondeterministic or acts one way most of the time but in a few special cases behaves differently. Such automation also greatly increases training difficulties.
- Some automation is programmed such that the logic can result in side effects that are not intended or known by human controllers.
- While the design of the system may include feedback to humans, there are many reasons why that feedback may not be noticed such as distraction or lowered alertness when monitoring something that rarely changes or is rarely incorrect.
- Complacency and overreliance on automation by humans is increasingly becoming a problem in automated systems today.
- Automation may fail so gracefully that the human controller does not notice the problem until late in the process. Humans may also think that automation is shut down or has failed when it has not. This type of problem has arisen when robots and humans must work in the same areas. The logout/tagout problem, where humans think energy is off but is actually on, leads to a large number of accidents in the workplace.

This list is only meant to indicate that there are many causes that must be considered when humans are part of a system. It is far from exhaustive. Working with a human factors expert is recommended if causal scenarios need to get to this level of detail in order to be eliminated or controlled.

Some Final Thoughts

The model and guidance provided in this appendix should not be used as a checklist in generating causal scenarios. It is almost surely incomplete, particularly for some unique types of systems. You will need to think carefully yourself about how your system works. We have found it very helpful to have experts on the design and operation of the system participate in causal scenario generation. The model in this appendix may be helpful, however, in precluding some basic omissions and in helping to identify a more complete set of scenarios.

We also caution that trying to look at all potential flaws in the system and see if they can lead to a UCA, basically the process used in FMEA, is hopeless for complex systems. Always start with the hazards and with the UCAs and their contextual factors and then work backward from them to identify potential causes. There may still be a large number, but orders of magnitude fewer than trying to identify everything that can go wrong and then see if it will lead to a UCA.

먼 컨트롤러가 모드의 변경에 대한 정보를 수령하였으나 이를 알아차리지 못하거나 처리하지 못하여 모드 혼동이 발생할 수 있다. 휴먼 컨트롤러의 직접적인 명령 없이 자동화가 컨트롤드 프로세스의 모드를 변경할 수 있는 경우 모드 혼동이나 잠재적인 안전하지 않은 컨트롤 액션이 발생할 수 있다.

- "상황 인식(situation awareness)"은 사고의 원인으로 흔히 인용되는데, 용어의 의미를 명확히 정의하지 않는 경우 많은 오류가 이 범주에 속할 수 있다. STPA가 사용하는 모델링에서의 상황 인식은 휴먼 컨트롤러 멘탈 모델과 실제 상태가 일치하지 않는 것을 의미하므로, STPA는 상황 인식 오류를 식별할 수 있다.
- 비(非)결정론적(nondeterministic)이거나, 대부분 한 가지 방식으로 동작하지만 특별한 경우에만 다르게 동작하는 자동화에 대해 사람들은 쉽게 혼동할 수 있다. 그러한 자동화를 훈련받는 것은 매우 어렵다.
- 일부 자동화의 경우 휴먼 컨트롤러가 의도하지 않거나 알지 못하는 부작용이 발생하도록 로직이 프로그램되어 있다.
- 시스템 설계 시 사람으로의 피드백이 포함되겠지만 다양한 이유로 피드백을 알아차릴 수 없게 되는 경우가 있다. 예를 들어 거의 변화가 없거나 틀릴 일이 없는 것을 모니터링 하는 경우에는 주의가 산만해지거나 조심성이 낮아질 수 있다.
- 사람이 자동화에 안주(complacency)하거나 지나치게 의존하는 것은 점점 오늘날 자동화 시스템의 문제가 되고 있다.
- 자동화가 너무 점진적으로 실패하면 휴먼 컨트롤러가 프로세스의 후반부까지도 문제를 인식하지 못할 수 있다. 자동화가 셧다운 또는 실패하지 않았지만 사람이 자동화가 셧다운 되었거나 실패했다고 생각할 수도 있다. 이러한 유형의 문제는 로봇과 사람이 같은 공간에서 일해야 할 때 발생한다. LOTO(logout/tagout) 문제에서, 사람이 실제로는 전원이 켜져 있지만 꺼져 있다고 잘못 생각하는 것은 작업장에서 수많은 사고를 초래한다.

위 목록은 사람이 시스템의 일부인 경우 고려해야 할 원인이 많음을 보여주기 위한 것이며 모든 것을 빠짐없이 나열한 것은 아니다. 위험 제거나 통제를 위해 원인 시나리오를 이 정도 수준으로 상세화하고자 한다면 인적 요인(human factor) 전문가와 함께 작업할 것을 권장한다.

최종 고찰

원인 시나리오를 생성할 때 이 부록에 제시된 모델 및 지침을 체크리스트로 사용해서는 안 된다. 특히, 특별한 유형의 시스템에는 분명히 완전하지 않다. 시스템이 어떻게 작동하는지를 신중히 생각할 필요가 있다. 우리는 시스템 설계 및 운영 전문가가 원인 시나리오 생성에 참여하는 것이 매우 도움이 된다는 것을 알게 되었다. 이 부록에 있는 모델은 몇 가지 기본적인 누락사항들을 제외하고는 원인 시나리오의 완전한 세트를 식별하는 데 도움이 될 것이다.

또한, 시스템의 잠재적 결함을 모두 살펴보고 그것들이 UCA를 유발할 수 있는지 확인하는 프로세스는 FMEA에서 기본적으로 수행하는 프로세스인데 이를 복잡한 시스템에 적용하는 것은 바람직하지 않다고 경고한다. 항상 위험과, UCA 및 UCA의 컨텍스트적인 요소로부터 출발하여 역방향으로 잠재적인 원인을 식별해야 한다. 그렇게 하여도 여전히 매우 많은 분석을 해야 하지만, 잘못될 수 있는 모든 경우를 식별한 다음 그것들이 UCA를 유발하는지 확인하는 것보다는 분석량이 수십 배 적을 것이다.

STPA HANDBOOK

영문/한글판 v1.1

2019.12

발행처 : 한국정보통신기술협회

http://sw.tta.or.kr/notify/STPAHandbook(ENG_KOR).jsp