Extending and Automating STPA for Requirements Generation and Analysis

John Thomas
Traditional Safety Engineering

Create Design

Hazard Analysis (FTA, FMEA, HAZOP, etc)
- Find chains of failure events that lead to accidents

Resolution
- Add redundancy, protective functions, “patches”

(Vincoli, 1993); (Bahr, 1997); (FAA SSH, 2000); (AF SSH, 2000)
Traditional Hazard Analysis Methods

- Failure Modes Effects and Criticality Analysis (1949)
  - Reliability technique; start with component failures, find effects
- Fault Tree Analysis (1961)
  - Top-down approach; start with hazard, find failure combinations
- Hazards and Operability Analysis (1960s)
  - Apply guidewords to components, find consequences
- Event Tree Analysis (1975)
  - Start with initiating event, trace forward in time

- Require fairly detailed design
- Not much help creating safety requirements from the start
- Especially complex software requirements

(Hammer, 1972); (Lawley, 1974); (Vesely et al, 1981); (Rasmussen, 1975); (Rasmussen, 1990); (Garrett, 2002); (Lisagor, 2006)
STPA (System-Theoretic Process Analysis)

- Built on STAMP model
- Start from hazards
- Identify hazardous control actions and safety constraints
- Identify scenarios that lead to violation of safety constraints

Traditionally applied ad-hoc without systematic procedures

(Leveson, 2011)
STPA Control Flaws

Inadequate Control Algorithm (Flaws in creation, process changes, incorrect modification or adaptation)

Process Model (inconsistent, incomplete, or incorrect)

Controller

Inadequate operation

Controlled Process

Component failures

Changes over time

Uncertainty or out-of-range disturbance

Unidentified or out-of-range disturbance

Controller

Inadequate or missing feedback

Feedback Delays

Incorrect or no information provided

Measurement inaccuracies

Feedback delays

Process output contributes to system hazard

Controller

Inadequate or missing communication with another controller

Inappropriate, ineffective, or missing control action

Controller

Inadequate operation

Actuator

Inadequate operation

Sensor

Inadequate operation

Controller

Missing or wrong communication with another controller

Inadequate control action

Controller

Process input missing or wrong

Controller

Unidentified or out-of-range disturbance

Need to create requirements specification without control flaws
Formal (model-based) requirements specification language

Example: SpecTRM-RL Model of TCAS II Collision Avoidance Logic

<table>
<thead>
<tr>
<th>INTRUDER.STATUS</th>
<th>=</th>
<th>Other-Traffic</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other-Traffic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proximate-Traffic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potential-Threat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Threat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unknown</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Formal mathematical representation:

Other-Traffic =
(Alt-Reporting == Lost) \land \neg \text{Bearing-Valid} \lor (Alt-Reporting == Lost) \land \neg \text{Range-Valid} \lor
(Alt-Reporting == Lost) \land \text{Bearing-Valid} \land \text{Range-Valid} \land \neg \text{Proximate-Traffic-Condition} \land
\neg \text{Potential-Threat-Condition} \lor (\text{Other-Aircraft \ in-state On-Ground})

(Leveson, 2000), (Zimmerman, 2002)
Structure of a Hazardous Control Action

Example: “Operator provides open train door command when train is moving”
Structure of a Hazardous Control Action

Example: “Operator provides open train door command when train is moving”

- Source Controller (SC)
- Type (T)
- Control Action (CA)
- Context (Co)

Four parts of a hazardous control action
- Source Controller: the controller that can provide the control action
- Type: whether the control action was provided or not provided
- Control Action: the controller’s command that was provided / missing
- Context: the system or environmental state in which command is provided

Process Model

Train motion
- Stopped
- Moving

Train location
- At platform
- Not Aligned
Identifying Hazardous Control Actions

• Type 1: Providing control action causes hazard
  – 1a) Define potential contexts (combinations of process model values)
  – 1b) Determine whether the control action is hazardous in each context
  – 1c) Determine whether control action can still be hazardous if too early/too late
• Type 2: Not providing control action causes hazard
  – Same as above, but for an absence of the selected control action
Example: Train door controller

System Hazards

H-1: Doors close on a person in the doorway

H-2: Doors open when the train is moving or not at platform

H-3: Passengers/staff are unable to exit during an emergency
Example: Control loop

Commands:
- Open door
- Stop opening door
- Close door
- Stop closing door

Other Inputs
- Train motion
- Train position
- Emergency Indicator

Feedback
- Door position
- Door obstruction

Door Controller

Door Actuator

Physical Door

Door Sensors
Example: Control loop

Door Controller

Process model
- Door position
  - Fully open
  - Fully closed
  - Partially open
  - Unknown
- Train position
  - Aligned with platform
  - Not aligned with platform
  - Unknown
- Train motion
  - Stopped
  - Train is moving
  - Unknown
- Emergency
  - No emergency
  - Evacuation required
  - Unknown

Commands:
- Open door
- Stop opening door
- Close door
- Stop closing door

Other Inputs
- Train motion
- Train position
- Emergency Indicator

Feedback
- Door position
- Door obstruction

Door Actuator

Physical Door

Door Sensors
STPA Process

- Identify hazards
- Create control structure
- Create process model
- Identify Unsafe Control Actions
  - For each control action, consider:
    - 1) Providing causes hazard
    - 2) Not providing causes hazard
- Identify Causes of Unsafe Control Actions
1) Control action is provided

- Control action: *Door Open* command

- 1a) Define potential contexts (combinations of process model variables)

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Train Position</th>
<th>Door Obstruction</th>
<th>Door Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>Aligned with platform</td>
<td>Not obstructed</td>
<td>Closed</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>Aligned with platform</td>
<td>Not obstructed</td>
<td>Open</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>Yes</td>
<td>Aligned with platform</td>
<td>Obstructed</td>
<td>Closed</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
1) Control action is provided

Control action: *Door Open* command

- 1a) Define potential contexts (combinations of process model variables)
- 1b) Determine whether the control action is hazardous in each context

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Train Position</th>
<th>Door Obst. / Position</th>
<th>Hazardous?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command</td>
<td>Moving</td>
<td>No</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>Yes</td>
</tr>
<tr>
<td>Door open command</td>
<td>Moving</td>
<td>Yes</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>Yes*</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>Yes</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>No</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>Not at platform</td>
<td>(doesn’t matter)</td>
<td>Yes</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>At platform</td>
<td>(doesn’t matter)</td>
<td>No</td>
</tr>
</tbody>
</table>

*Design decision: In this situation, evacuate passengers to other cars. Meanwhile, stop the train and then open doors.*
1) Control action is provided

Control action: *Door Open* command
- 1a) Define potential contexts (combinations of process model variables)
- 1b) Determine whether the control action is hazardous in each context
- 1c) Determine whether control action can still be hazardous if too early/too late

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Train Position</th>
<th>Door Obst. / Position</th>
<th>Hazardous?</th>
<th>Hazardous if provided too early?</th>
<th>Hazardous if provided too late?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command</td>
<td>Moving</td>
<td>No</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Door open command</td>
<td>Moving</td>
<td>Yes</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>Yes*</td>
<td>Yes*</td>
<td>Yes*</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>Yes</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>Not at platform</td>
<td>(doesn’t matter)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Door open command</td>
<td>Stopped</td>
<td>No</td>
<td>At platform</td>
<td>(doesn’t matter)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
2) Control action is **not** provided

Control action: *Door Open* command

- 2a) Identify process model variables
- 2b) Determine whether the absence of control action is hazardous in each context

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Train Position</th>
<th>Door Obst. / Pos.</th>
<th>Hazardous?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command not provided</td>
<td>Stopped</td>
<td>Yes</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Door open command not provided</td>
<td>Stopped</td>
<td>(doesn’t matter)</td>
<td>(doesn’t matter)</td>
<td>Closing on obstruction</td>
<td><strong>Yes</strong></td>
</tr>
<tr>
<td>Door open command not provided</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>No</strong></td>
</tr>
</tbody>
</table>
## Resulting List of Hazardous Control Actions

<table>
<thead>
<tr>
<th>Hazardous Control Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command provided while train is moving and there is no emergency</td>
</tr>
<tr>
<td>Door open command provided too late while train is stopped and emergency exists</td>
</tr>
<tr>
<td>Door open command provided while train is stopped, no emergency, and not at platform</td>
</tr>
<tr>
<td>Door open command provided while train is moving and emergency exists</td>
</tr>
<tr>
<td>Door open command not provided while train is stopped and emergency exists</td>
</tr>
<tr>
<td>Door open command not provided while doors are closing on someone and train is stopped</td>
</tr>
</tbody>
</table>

Much of this can be automated to assist the safety engineer!
Generating safety requirements

Hazardous Control Actions

Formal (model-based) requirements specification

<table>
<thead>
<tr>
<th>Hazard</th>
<th>State</th>
<th>Hazardous Control</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt-Reporting</td>
<td>in-state</td>
<td></td>
<td>Lost</td>
</tr>
<tr>
<td>Bearing-Valid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Range-Valid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proximate-Traffic-Condition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potential-Threat-Condition</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Other-Aircraft in-state On-Ground</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Generating safety requirements

- Formal requirements can be derived using:
  - Discrete mathematical structure for hazardous control actions
  - Predicate calculus to obtain necessary requirements
- Automatically generate formal requirements given these relationships!

Diagram:
- Hazardous Control Actions
  - Discrete Mathematical Representation
  - Predicate calculus / state machine structure
  - Formal (model-based) requirements specification
Hazardous control actions: mathematical representation

Example: “Operator provides open train door command when train is moving”

Hazardous control action as 4-tuple (SC, T, CA, Co) where:

– SC ∈ Controllers [from control structure]
– T ∈ {Provided, Not Provided}
– CA ∈ ControlActions(SC)
– Co = {V, SC} | (V ∈ PMV) ∧ (SC ∈ PMS) ∧ SC child V

Process Model

Train motion
- Stopped
- Moving

Train location
- Aligned
- Not Aligned

Process Model Variables (PMV)
Process Model States (PMS)
Hazardous control actions -> formal requirements specifications

1. Describing hazardous, functional, and required behavior
   - \( HP(h \in H, ca \in CA, c \in Co) \)
     • True iff providing command \( ca \) in context \( c \) will cause hazard \( h \)
   - \( HNP(h \in H, ca \in CA, c \in Co) \)
     • True iff not providing command \( ca \) in context \( c \) will cause hazard \( h \)
   - \( FP(f \in F, ca \in CA, c \in Co) \)
     • True iff providing command \( ca \) in context \( c \) is necessary to achieve function \( f \)
   - \( R(ca \in CA, c \in Co) \)
     • True iff command \( CA \) is required to be provided in context \( c \)

2. Consistency checks
   - \( \forall h_1 \in H, h_2 \in H \rightarrow \exists ca \in CA, c \in C : HP(h_1, ca, c) \land HNP(h_2, ca, c) \)
     • For every potential context, it must be possible to avoid hazardous control actions/inactions. In other words, if it is hazardous to provide \( CA \) then it should be non-hazardous to not provide \( CA \)
   - \( \forall h \in H, f \in F \rightarrow \exists ca \in CA, c \in C : HP(h, ca, c) \land FP(f, ca, c) \)
     • For every potential context, if it is necessary to provide a command to fulfill a function then it must not be hazardous to provide the command in that context

3. Requirements generation (SpecTRM-RL tables)
   - Compute \( R(ca \in CA, c \in C) \) to satisfy the following:
     - \( \forall h, ca, c : h \in H \land ca \in CA \land c \in C \rightarrow [HP(h, ca, c) \rightarrow \neg R(ca, c)] \)
     - \( \forall h, ca, c : h \in H \land ca \in CA \land c \in C \rightarrow [R(ca, c) \rightarrow HNP(h, ca, c)] \)
     - \( \forall f, ca, c : f \in F \land ca \in CA \land c \in C \rightarrow [FP(f, ca, c) \rightarrow R(ca, c)] \)
Generating safety requirements

• Example: Generated black-box model for door controller

<table>
<thead>
<tr>
<th>Behavior required for function</th>
<th>Behavior required for safety</th>
</tr>
</thead>
</table>

**Provide 'Open Doors' command**

| Door State = | Doors not closing on person | | | | T |
|--------------|-----------------------------|---|---|---|
| Doors closing on person | | | | |
| Train Position = | Aligned with platform | T |
| Not aligned with platform | | | | |
| Train Motion = | Stopped | T | T | T |
| Train is moving | | | | |
| Emergency = | No emergency | | | |
| Emergency exists | | | | T |

Open Doors =
(Train Position in-state Aligned) \(\land\) (Train Motion in-state Stopped) \(\lor\) (Train Motion in-state Stopped) \(\land\) (Emergency in-state exists) \(\lor\) (Door State in-state closing on person) \(\land\) (Train Motion in-state Stopped)
Detecting conflicts

• Can automatically check consistency using info in context tables

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Hazardous?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command</td>
<td>Moving</td>
<td>Yes</td>
<td>Yes*</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Train Motion</th>
<th>Emergency</th>
<th>Hazardous?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door open command not provided</td>
<td>Moving</td>
<td>Yes</td>
<td>Yes*</td>
</tr>
</tbody>
</table>

• Example: Conflict between opening the door vs. not opening the door
Summary

• Systematic process for performing STPA
• Method to help automate STPA
• Drives the creation of requirements and definition of control algorithms from the STPA analysis
• Automatically generating formal safety requirements
• Analyze not only safety aspects, but also functional goals
• Consistency checks to detect safety vs. functional conflicts
Thank you!

Questions?