A Tool-Based STPA Process

John Thomas and Dajiang Suo
Outline

• Formal approach to STPA

• Current tool-based STPA process

• New tool-based STPA process
STPA
(System-Theoretic Process Analysis)

• System engineering foundation
  – Define accidents, hazards
  – Create control structure

• Step 1: Identify unsafe control actions

• Step 2: Identify accident causal scenarios

(Leveson, 2012)
Structure of an Unsafe Control Action

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

<table>
<thead>
<tr>
<th>Context</th>
<th>Train motion</th>
<th>Stopped</th>
<th>Moving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train location</td>
<td>At platform</td>
<td>Not Aligned</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source Controller</th>
<th>Control Action</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open train door</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Control Actions</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not providing causes hazard</td>
<td>Providing causes hazard</td>
</tr>
<tr>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Formalizing Unsafe Control Actions

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

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

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

---

Much of this can be automated!

### Unsafe Control Actions

- Door open command provided while train is moving and there is no emergency
- Door open command provided too late while train is stopped and emergency exists
- Door open command provided while train is stopped, no emergency, and not at platform
- Door open command provided while train is moving and emergency exists
- Door open command **not** provided while train is stopped and emergency exists
- Door open command **not** provided while doors are closing on someone and train is stopped
Automating STPA

Automatically generated
(from control structure and PMVs)

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

Generated from simple rules (from engineers)
Detecting conflicts

- Can automatically check consistency, search for conflicts

<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
Generating safety requirements

- Example: Generated black-box model for door controller. Executable.

<table>
<thead>
<tr>
<th>Provide 'Open Doors' command</th>
<th>Behavior required for function</th>
<th>Behavior required for safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>Door State =</td>
<td>Doors not closing on person</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Doors closing on person</td>
<td>T</td>
</tr>
<tr>
<td>Train Position =</td>
<td>Aligned with platform</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>Not aligned with platform</td>
<td></td>
</tr>
<tr>
<td>Train Motion =</td>
<td>Stopped</td>
<td>T T T</td>
</tr>
<tr>
<td></td>
<td>Train is moving</td>
<td></td>
</tr>
<tr>
<td>Emergency =</td>
<td>No emergency</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Emergency exists</td>
<td>T</td>
</tr>
</tbody>
</table>

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)
Tool-assisted Process
Tool-assisted process

- System engineering foundation
  - Define hazards
  - Create control structure
- Step 1: Identify unsafe control actions
- Step 2: Identify accident causal scenarios

### Process Overview

1. Identify hazards
2. Create basic control structure
3. Basic UCA table
4. Identify process model variables
5. Define initial UCA rules
6. Identify conflicts, overlaps, and missing rules
7. Analyze conflicts
8. Analyze overlapping rules
9. Verify missing rules
10. Create safety constraints and requirements
11. Perform STPA Step 2
Tool-assisted process

1. Identify hazards, controllers, controlled processes, control actions, feedbacks.
2. Brainstorm, fill in template
3. Extract PMVs from UCAs in basic table
4. Translate basic UCAs into Rules
5. Define initial UCA rules
6. Identify conflicts, overlaps, and missing rules
7. Analyze Conflicts
8. Overlapping rules
9. Verify Missing Rules
10. Create safety constraints and requirements
11. Perform STPA Step 2

Human

Tool Assistance

Process Overview

1. Identify hazards, controllers, controlled processes, control actions, feedbacks.
2. Brainstorm, fill in template
3. Extract PMVs from UCAs in basic table
4. Translate basic UCAs into Rules
5. Define initial UCA rules
6. Identify conflicts, overlaps, and missing rules
7. Analyze Conflicts
8. Overlapping rules
9. Verify Missing Rules
10. Create safety constraints and requirements
11. Perform STPA Step 2

System Engineering Foundations

STPA Step 1

STPA Step 2: Identify Accident Causal Scenarios

Causes of UCA Causes of safe control action not followed
The Architecture of an STPA tool

The architecture of Eclipse platform is taken from eclipse.org
A Toolset for Supporting STPA and Requirement Generation

1. Add hazard labels

2. Create safety control structure

3. Add PMVs

4. Generate Context Table

5. Define Rules with rule editor

6. Identify conflicts, overlaps and missing UCAs

7. Automatically create safety Requirement (SpecTRM-RL)
Feedback from “beta” testing

• I want to **change** the control structure in the middle of the analysis
  – Add new controller responsibility
  – Change a control action
  – Change feedback / process model variable
  – Etc.
Challenge(1):
Are old rules still valid if the user changes PMV labels?

Process Overview
1. Identify hazards, controllers, controlled processes, control actions, feedbacks.
2. Create safety control structure.
3. Basic UCA Table.
4. Identify Process Model Variables.
5. Define initial UCA rules.
6. Identify conflicts, overlaps, and missing rules.
7. Analyze Conflicts.
8. Overlapping rules.
10. Create safety constraints and requirements.
11. Perform STPA Step 2.

STPA Step 1
System Engineering Foundations

STPA Step 2: Identify Accident Causal Scenarios

Human
Brainstorm, fill in template.
Extract PMVs from UCAs in basic table.
Translate basic UCAs into Rules.

Tool Assistance
- Add hazard labels with tool editor.
- Add control structure in tool editor.

Identify hazards, controllers, controlled processes, control actions, feedbacks.
- Add PMV labels with tool editor.
- Draw control structure in tool editor.

Add PMV labels inside each controller.
Add rules with Rule editor.
Identified automatically by tool.

Resolve conflicts, check overlaps, check missing UCAs.
Update or add new rules as needed.
Automatically create safety constraints and requirements.

Causes of UCA Causes of safe control action not followed.
Tool Support for Modifying PMVs
Example: Add PMVs

Before Adding PMVs
Safety Control Structure
Old PMVs and values in controller
- Train state
  - Moving
  - Stopped
- Door state
  - Person in door...
  - Person not in...
- Train position
  - At platform
  - Not at platform
- open → close
-Unnamed Feedback

Rule definition
Old Rule related to Old PMVs

After Adding PMVs
Safety Control Structure
New PMVs and values in controller
- Train state
  - Moving
  - Stopped
  - Emergency
- Door state
  - Person in door...
- Train position
  - At platform
- open → close
-Unnamed Feedback

Rule definition
New Rule related New PMVs

Add/delete PMVs

Export Rules to external files

Import Rules from external files
Are old rules still valid if the user changes PMV labels?

Example: Add PMVs

Observations:
- Contexts have been changed
- More Rules may become relevant
- New Conflicts are identified

<table>
<thead>
<tr>
<th>Control Action</th>
<th>Type</th>
<th>Train state</th>
<th>Door state</th>
<th>Train position</th>
<th>Hazards</th>
<th>Too Early/Too Late Hazard</th>
<th>Conflicts</th>
<th>Related Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>not provided...</td>
<td>Moving</td>
<td>Person l...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>open</td>
<td>not provided...</td>
<td>Moving</td>
<td>Person r...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R2</td>
</tr>
<tr>
<td>open</td>
<td>not provided...</td>
<td>Stopped</td>
<td>Person l...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R2</td>
</tr>
<tr>
<td>open</td>
<td>not provided...</td>
<td>Stopped</td>
<td>Person r...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R2</td>
</tr>
<tr>
<td>open</td>
<td>provided when</td>
<td>Moving</td>
<td>Person l...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R1</td>
</tr>
<tr>
<td>open</td>
<td>provided when</td>
<td>Moving</td>
<td>Person r...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R1</td>
</tr>
<tr>
<td>open</td>
<td>provided when</td>
<td>Stopped</td>
<td>Person l...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R1</td>
</tr>
<tr>
<td>open</td>
<td>provided when</td>
<td>Stopped</td>
<td>Person r...</td>
<td>At platform</td>
<td>not at pl...</td>
<td></td>
<td></td>
<td>R1</td>
</tr>
</tbody>
</table>