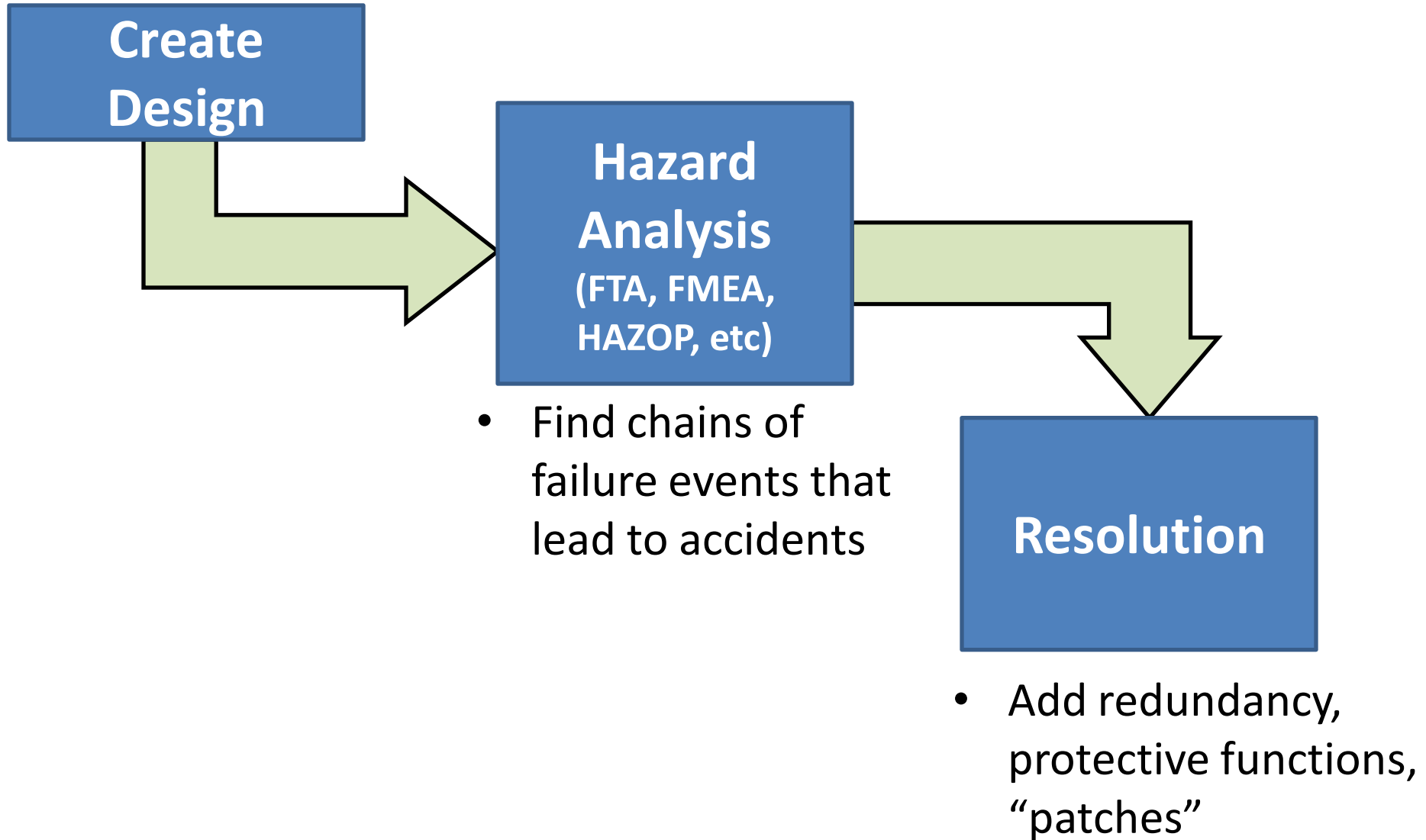


# Extending and Automating STPA for Requirements Generation and Analysis

**John Thomas**

# Traditional Safety Engineering



# Traditional Hazard Analysis Methods

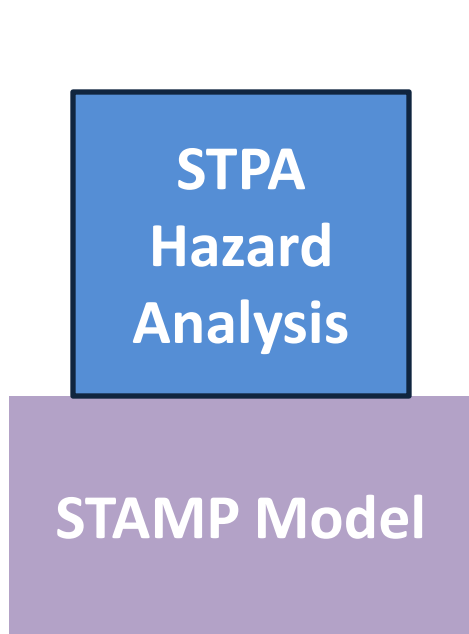
**Hazard  
Analysis  
Method**

**Model of Accident  
Causation**

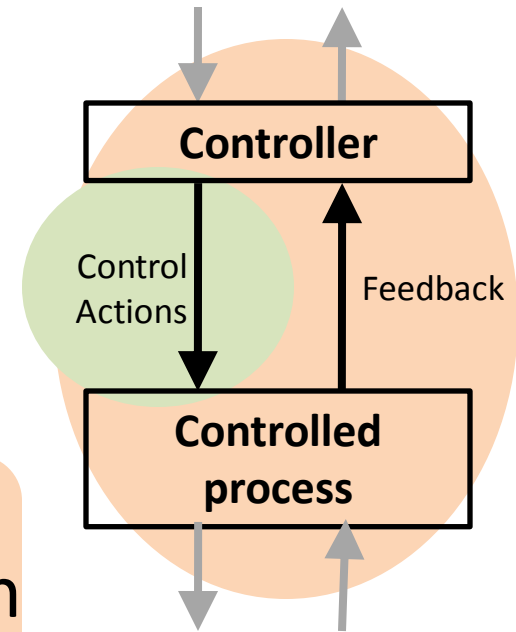
- Failure Modes Effects and Criticality Analysis (1949)
  - Reliability technique; start with component failures, find effects
- Fault Tree Analysis (1961)
  - Top-down approach; start with hazard, find failure combinations
- Hazards and Operability Analysis (1960s)
  - Apply guidewords to components, 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**

# STPA (System-Theoretic Process Analysis)



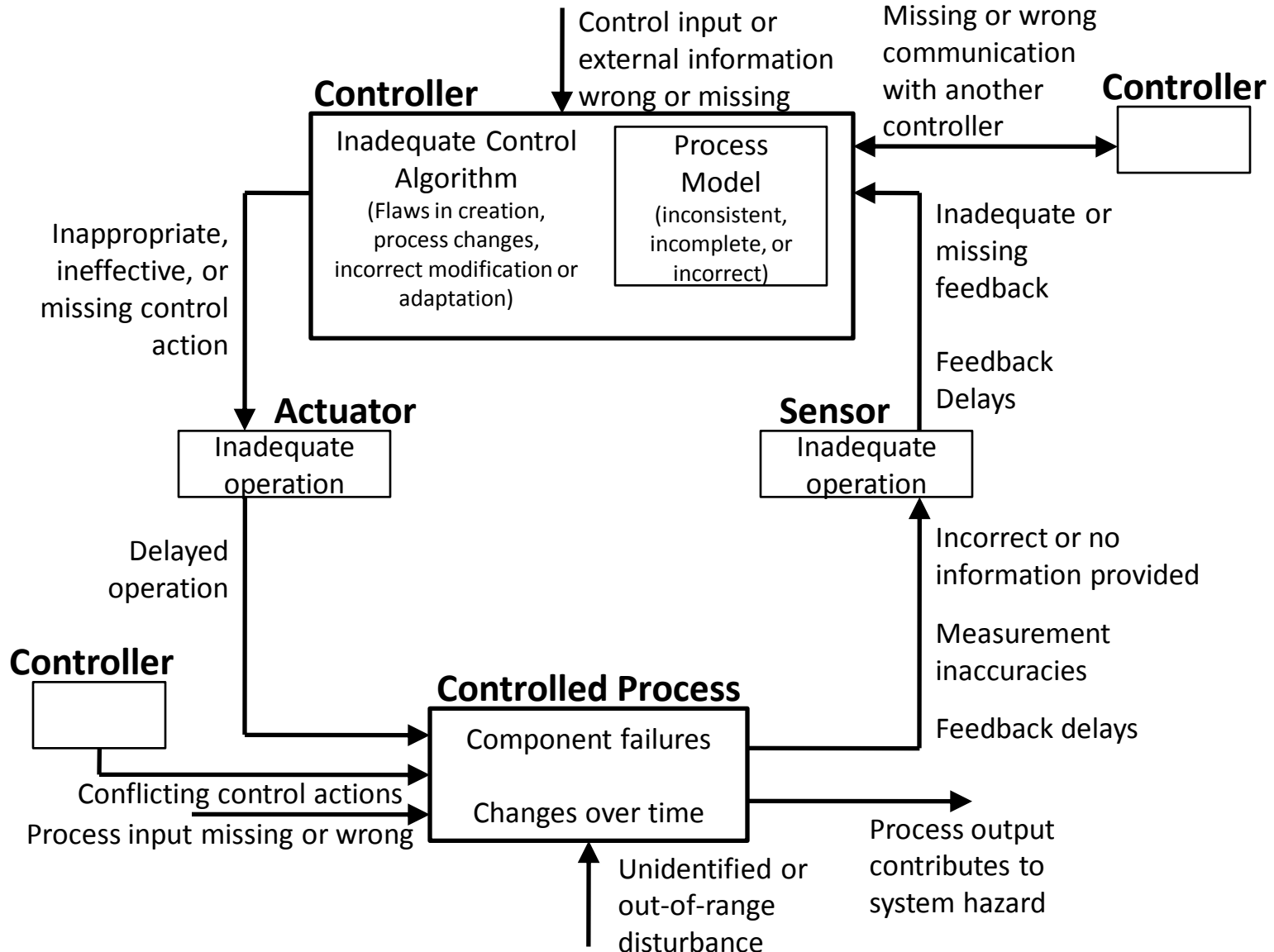
- Built on STAMP model
- Start from hazards
- Identify hazardous control actions and safety constraints
- Identify scenarios that lead to violation of safety constraints



Traditionally applied ad-hoc without systematic procedures

(Leveson, 2011)

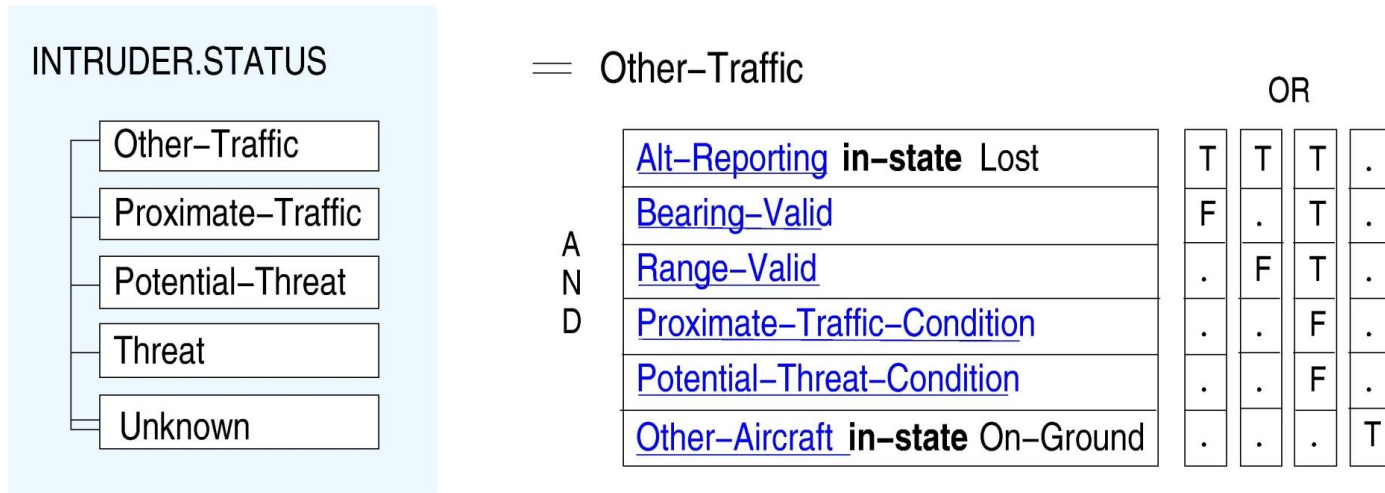
# STPA Control Flaws



Need to create requirements specification without control flaws

# Formal (model-based) requirements specification language

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



## Formal mathematical representation:

Other-Traffic =

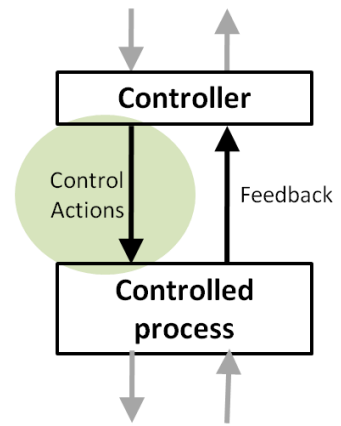
$$\begin{aligned}
 & (\text{Alt-Reporting} == \text{Lost}) \wedge \neg \text{Bearing-Valid} \vee (\text{Alt-Reporting} == \text{Lost}) \wedge \neg \text{Range-Valid} \vee \\
 & (\text{Alt-Reporting} == \text{Lost}) \wedge \text{Bearing-Valid} \wedge \text{Range-Valid} \wedge \neg \text{Proximate-Traffic-Condition} \wedge \\
 & \neg \text{Potential-Threat-Condition} \vee (\text{Other-Aircraft} == \text{On-Ground})
 \end{aligned}$$

(Leveson, 2000), (Zimmerman, 2002)

# Structure of a Hazardous Control Action

Example:

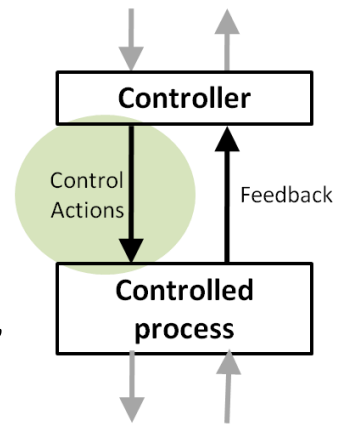
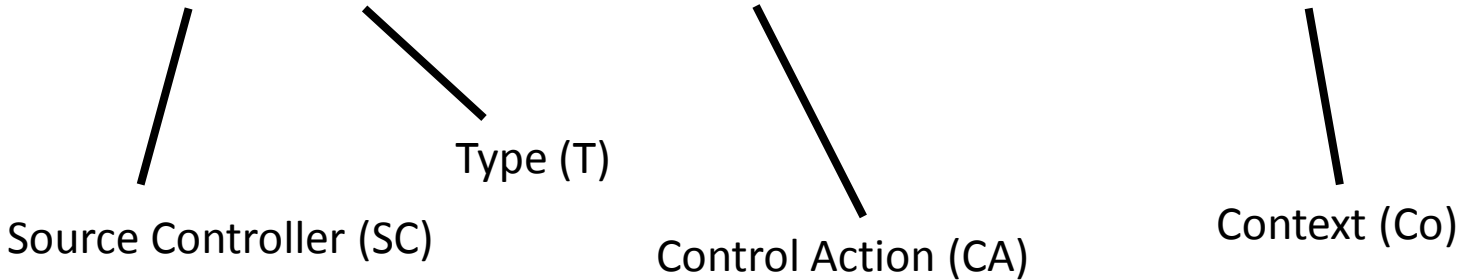
“Operator provides open train door command when train is moving”



# Structure of a Hazardous Control Action

Example:

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



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

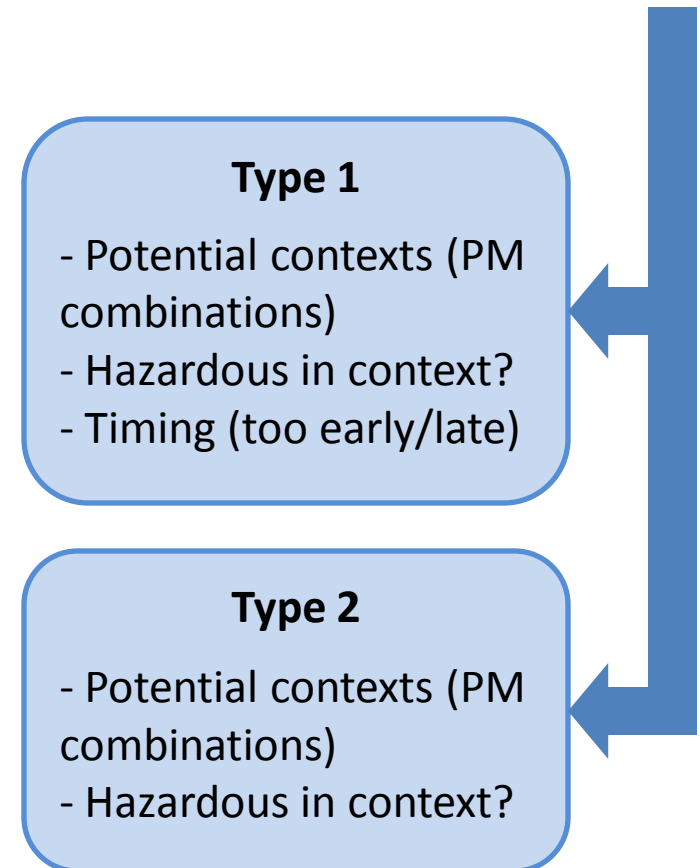
|                |   |                            |
|----------------|---|----------------------------|
| Train motion   | [ | Stopped<br>Moving          |
| Train location | [ | At platform<br>Not Aligned |



# 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

Hazards, controller,  
control actions,  
process model



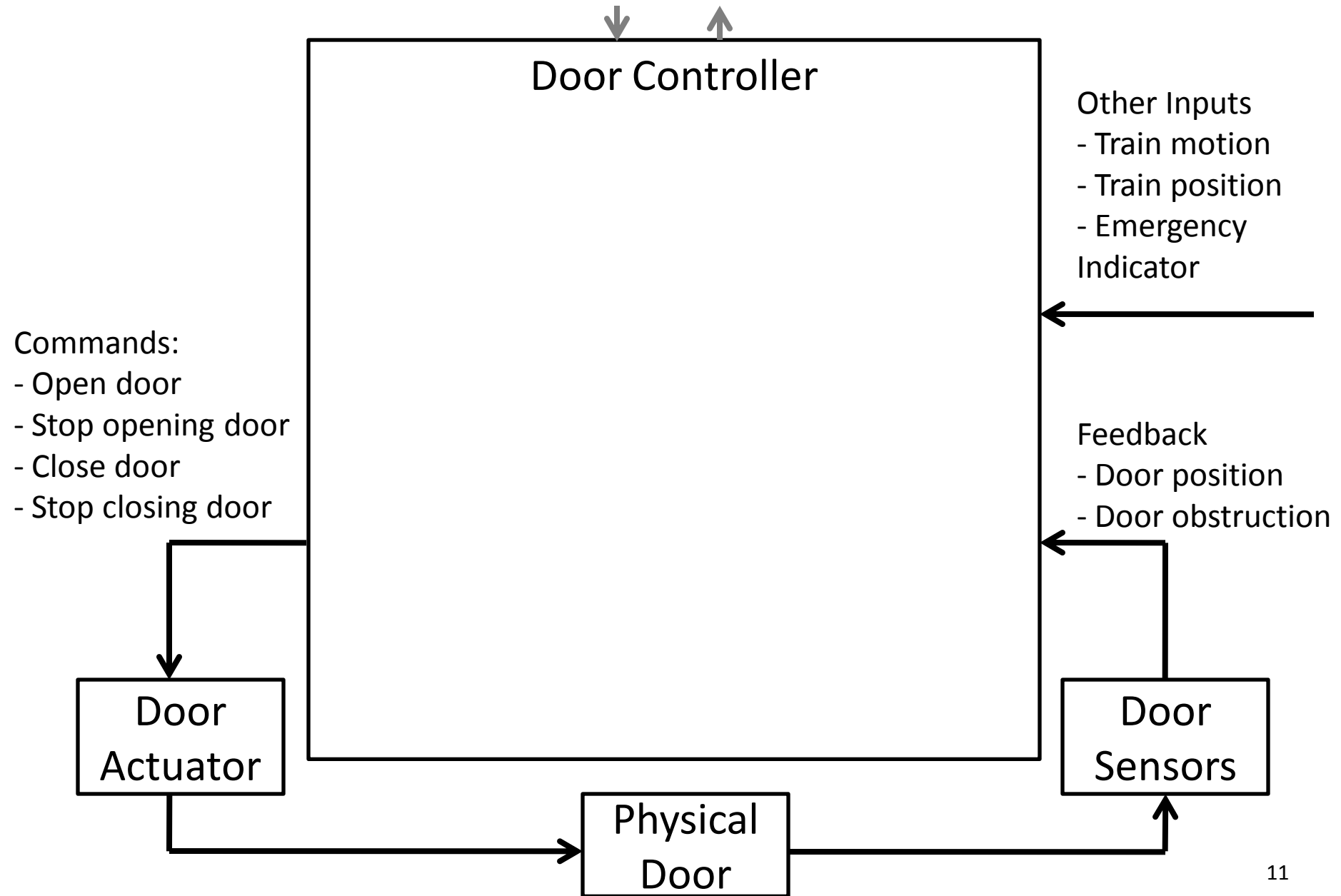
# Example: Train door controller



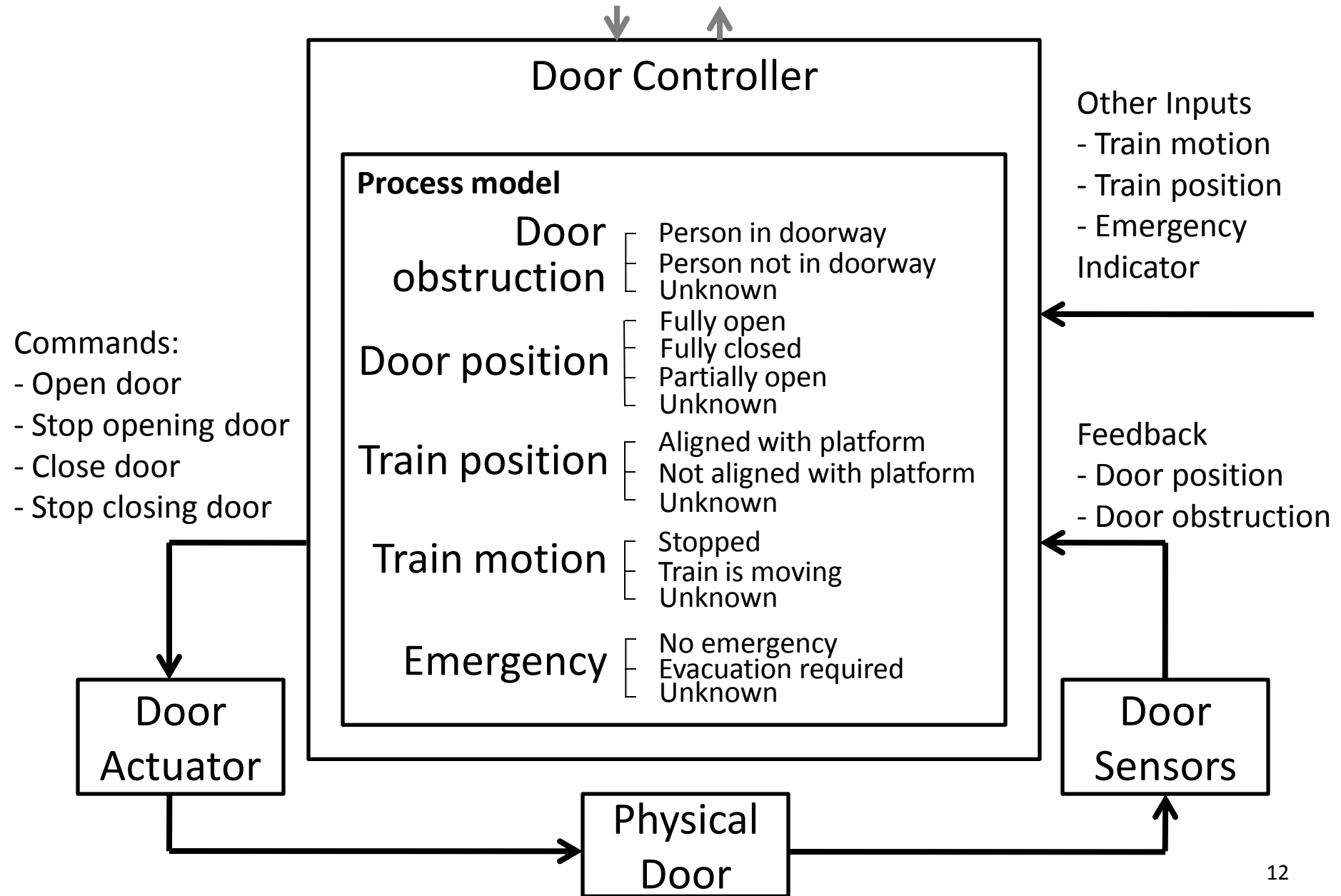
## System Hazards

- 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



# Example: Control loop



# STPA Process

✓ Identify hazards

✓ Create control structure

✓ Create process model

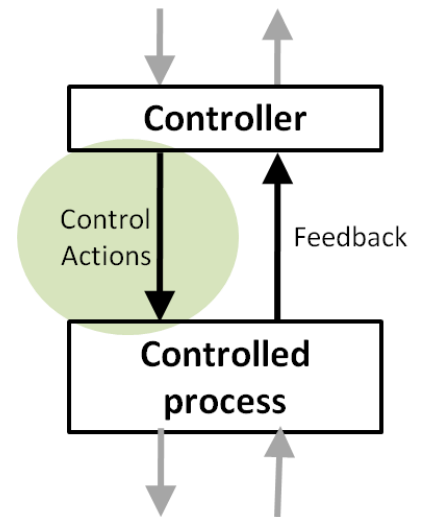
➔ Identify Unsafe Control Actions

– 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)

| Control Action    | Train Motion | Emergency | Train Position        | Door Obstruction | Door Position |
|-------------------|--------------|-----------|-----------------------|------------------|---------------|
| Door open command | Stopped      | No        | Aligned with platform | Not obstructed   | Closed        |
| Door open command | Stopped      | No        | Aligned with platform | Not obstructed   | Open          |
| Door open command | Stopped      | Yes       | Aligned with platform | Obstructed       | Closed        |
| ...               | ...          | ...       | ...                   | ...              | ...           |

# 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

| Control Action    | Train Motion | Emergency | Train Position   | Door Obst. / Position | Hazardous? |
|-------------------|--------------|-----------|------------------|-----------------------|------------|
| Door open command | Moving       | No        | (doesn't matter) | (doesn't matter)      | Yes        |
| Door open command | Moving       | Yes       | (doesn't matter) | (doesn't matter)      | Yes*       |
| Door open command | Stopped      | Yes       | (doesn't matter) | (doesn't matter)      | No         |
| Door open command | Stopped      | No        | Not at platform  | (doesn't matter)      | Yes        |
| Door open command | Stopped      | No        | At platform      | (doesn't matter)      | No         |

\*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

| Control Action    | Train Motion | Emergency | Train Position   | Door Obst. / Position | Hazardous ? | Hazardous if provided too early? | Hazardous if provided too late? |
|-------------------|--------------|-----------|------------------|-----------------------|-------------|----------------------------------|---------------------------------|
| Door open command | Moving       | No        | (doesn't matter) | (doesn't matter)      | Yes         | Yes                              | Yes                             |
| Door open command | Moving       | Yes       | (doesn't matter) | (doesn't matter)      | Yes*        | Yes*                             | Yes*                            |
| Door open command | Stopped      | Yes       | (doesn't matter) | (doesn't matter)      | No          | No                               | Yes                             |
| Door open command | Stopped      | No        | Not at platform  | (doesn't matter)      | Yes         | Yes                              | Yes                             |
| Door open command | Stopped      | No        | At platform      | (doesn't matter)      | No          | No                               | No                              |



## 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

| Control Action                 | Train Motion | Emergency        | Train Position   | Door Obst. / Pos.      | Hazardous? |
|--------------------------------|--------------|------------------|------------------|------------------------|------------|
| Door open command not provided | Stopped      | Yes              | (doesn't matter) | (doesn't matter)       | Yes        |
| Door open command not provided | Stopped      | (doesn't matter) | (doesn't matter) | Closing on obstruction | Yes        |
| Door open command not provided | (all others) |                  |                  |                        | No         |

# Resulting List of Hazardous Control Actions

## Hazardous 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

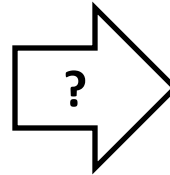
Door open command not provided while train is stopped and emergency exists

Door open command not provided while doors are closing on someone and train is stopped

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

# Generating safety requirements

**Hazardous Control  
Actions**

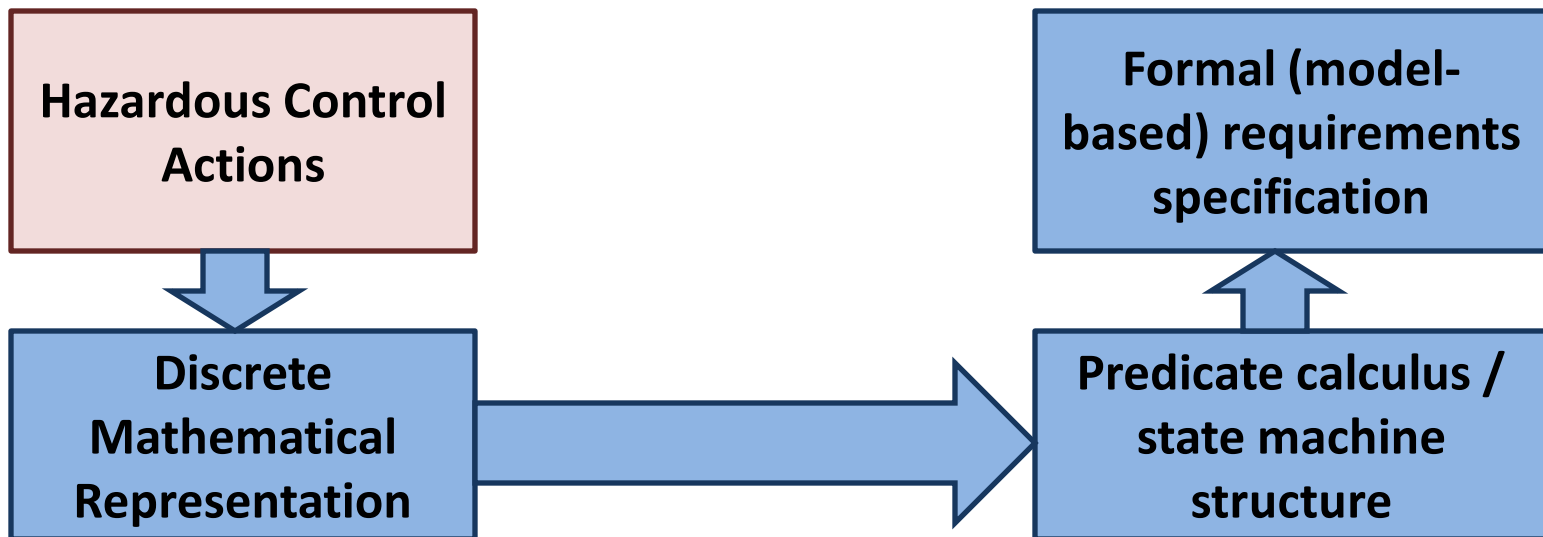


**Formal (model-  
based) requirements  
specification**

|   |   |   |   |   |
|---|---|---|---|---|
| <a href="#">Alt-Reporting in-state</a> Lost       | T | T | T | . |
| <a href="#">Bearing-Valid</a>                     | F | . | T | . |
| <a href="#">Range-Valid</a>                       | . | F | T | . |
| <a href="#">Proximate-Traffic-Condition</a>       | . | . | F | . |
| <a href="#">Potential-Threat-Condition</a>        | . | . | F | . |
| <a href="#">Other-Aircraft in-state</a> On-Ground | . | . | . | T |

# 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!



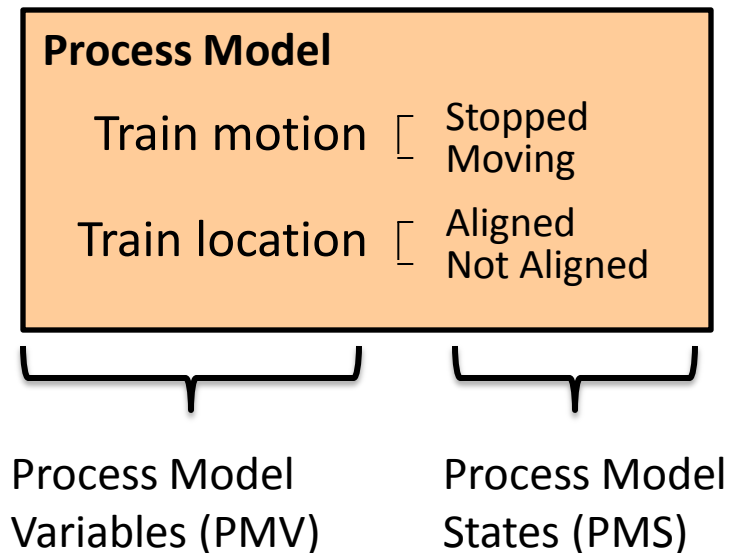
# Hazardous control actions: mathematical representation

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



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

- $SC \in \text{Controllers}$  [from control structure]
- $T \in \{\text{Provided, Not Provided}\}$
- $CA \in \text{ControlActions}(SC)$
- $Co = \{V, SC\} \mid (V \in \text{PMV}) \wedge (SC \in \text{PMS}) \wedge SC \text{ child } V$



# 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

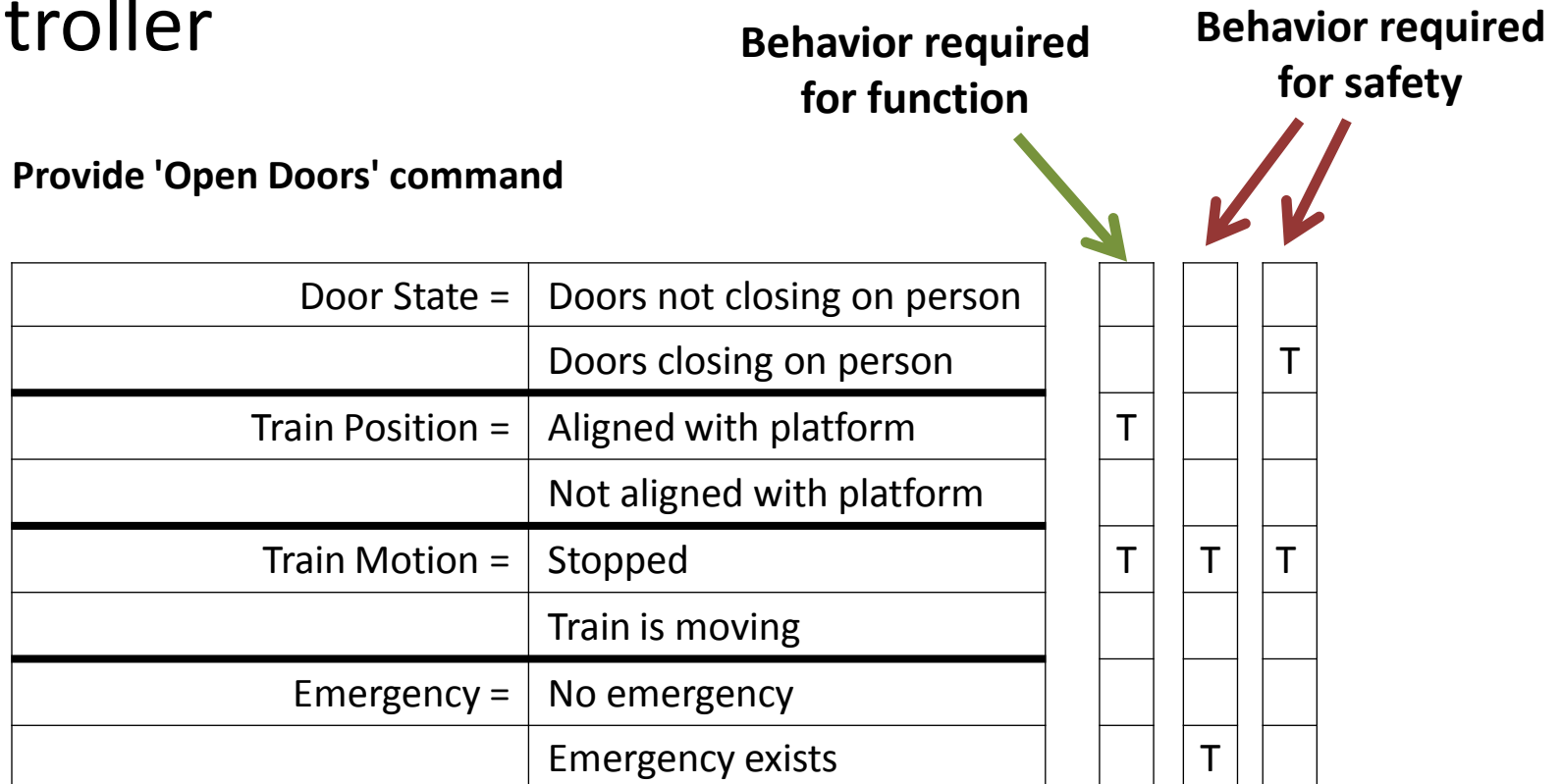
- $\forall h1 \in H, h2 \in H \rightarrow \exists ca \in CA, c \in C : HP(h1, ca, c) \wedge HNP(h2, ca, c)$ 
  - For every potential context, it must be possible to avoid hazardous control actions/inactions. In other words, if it is hazardous to provide  $CA$  then it should be non-hazardous to not provide  $CA$
- $\forall h \in H, f \in F \rightarrow \exists ca \in CA, c \in C : HP(h, ca, c) \wedge F(f, ca, c)$ 
  - 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:
- $\forall h, ca, c: h \in H \wedge ca \in CA \wedge c \in C \rightarrow [HP(h, ca, c) \rightarrow \neg R(ca, c)]$
- $\forall h, ca, c: h \in H \wedge ca \in CA \wedge c \in C \rightarrow [R(ca, c) \rightarrow HNP(h, ca, c)]$
- $\forall f, ca, c: f \in F \wedge ca \in CA \wedge c \in C \rightarrow [FP(f, ca, c) \rightarrow R(ca, c)]$

# Generating safety requirements

- Example: Generated black-box model for door controller



Open Doors =

$(\text{Train Position in-state Aligned}) \wedge (\text{Train Motion in-state Stopped}) \vee (\text{Train Motion in-state Stopped}) \wedge (\text{Emergency in-state exists}) \vee (\text{Door State in-state closing on person}) \wedge (\text{Train Motion in-state Stopped})$

# Detecting conflicts

- Can automatically check consistency using info in context tables

| Control Action    | Train Motion | Emergency | Hazardous? |
|-------------------|--------------|-----------|------------|
| Door open command | Moving       | Yes       | Yes*       |

| Control Action                 | Train Motion | Emergency | Hazardous? |
|--------------------------------|--------------|-----------|------------|
| Door open command not provided | Moving       | Yes       | Yes*       |

- 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?