Massachusetts Institute of Technology

STPA Step 4
Building Scenarios

A Formal Scenario Approach

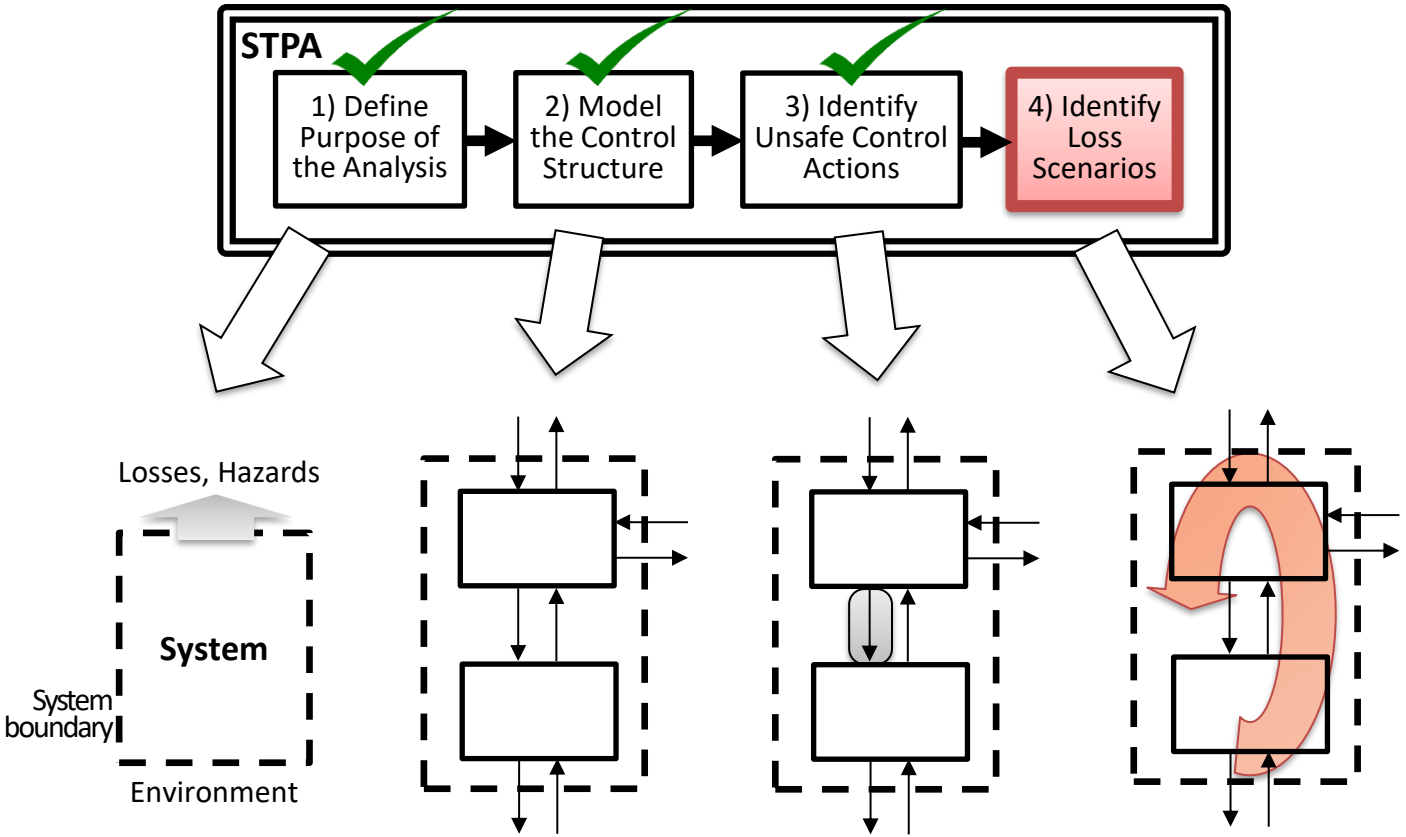
John Thomas

Any questions? Email me! JThomas4@mit.edu

© Copyright 2024 John Thomas

Agenda

- Goals for a new scenario development process
- The new process
- Results from real-world testing of the process



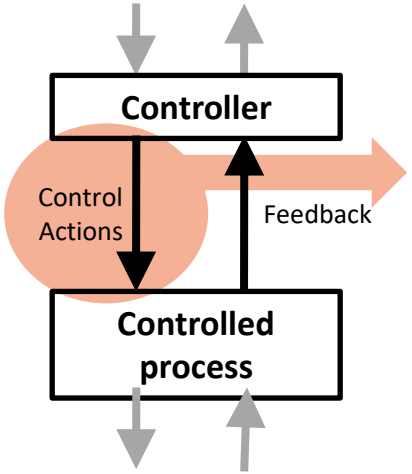
(Leveson and Thomas, 2018)

Goals for a Formal Scenario Building Approach

- Provide more scenario guidance for new practitioners who may get stuck
- Provide a formal structure for scenarios (similar to UCA syntax)
- Provide a way to review scenario completeness and find gaps
- Handle more complexity with less effort
  - Use top-down approach, not a backward search
  - Reduce repetition in scenarios

© Copyright 2024 John Thomas

Example of STPA Rigor: Unsafe Control Actions (UCA)



Example UCA:

"UCA-1: Computer provides Shift-to-Park cmd while vehicle is moving [H-1]"

Source Controller

Type

Control Action

Context

"The UCA process works well, but the scenario process feels ad-hoc. Can you make the scenarios (STPA step 4) more like this?"

© Copyright 2023 John Thomas

BLUF: Bottom Line Up Front

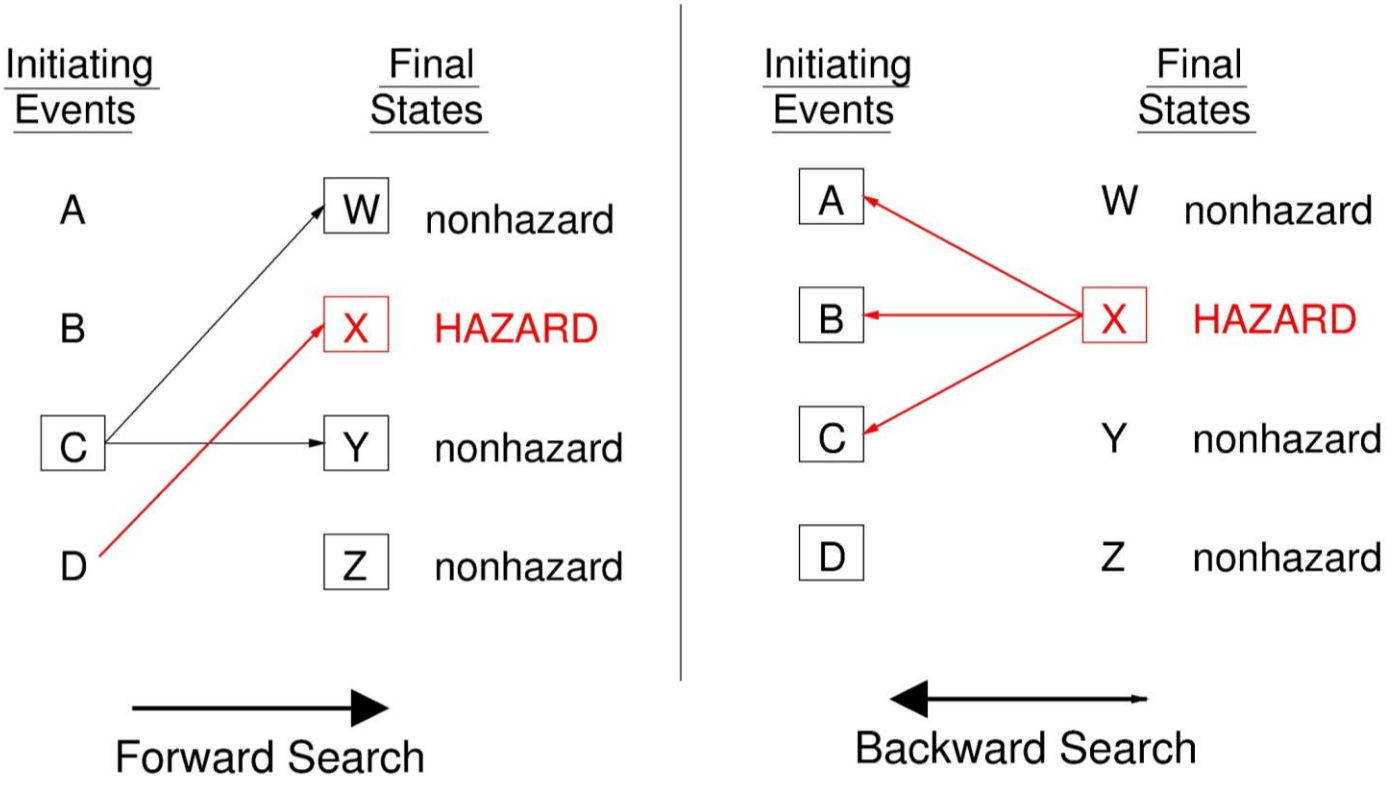
- A formal STPA scenario approach has been defined and tested
- Result: It works, with advantages and disadvantages
  - Advantage: Increased rigor and repeatability; frequently catches previously unknown flaws
  - Disadvantage: Introduces more steps and rules, which take longer to teach
- We'll discuss the validation, testing, and other observations at the end

© Copyright 2024 John Thomas

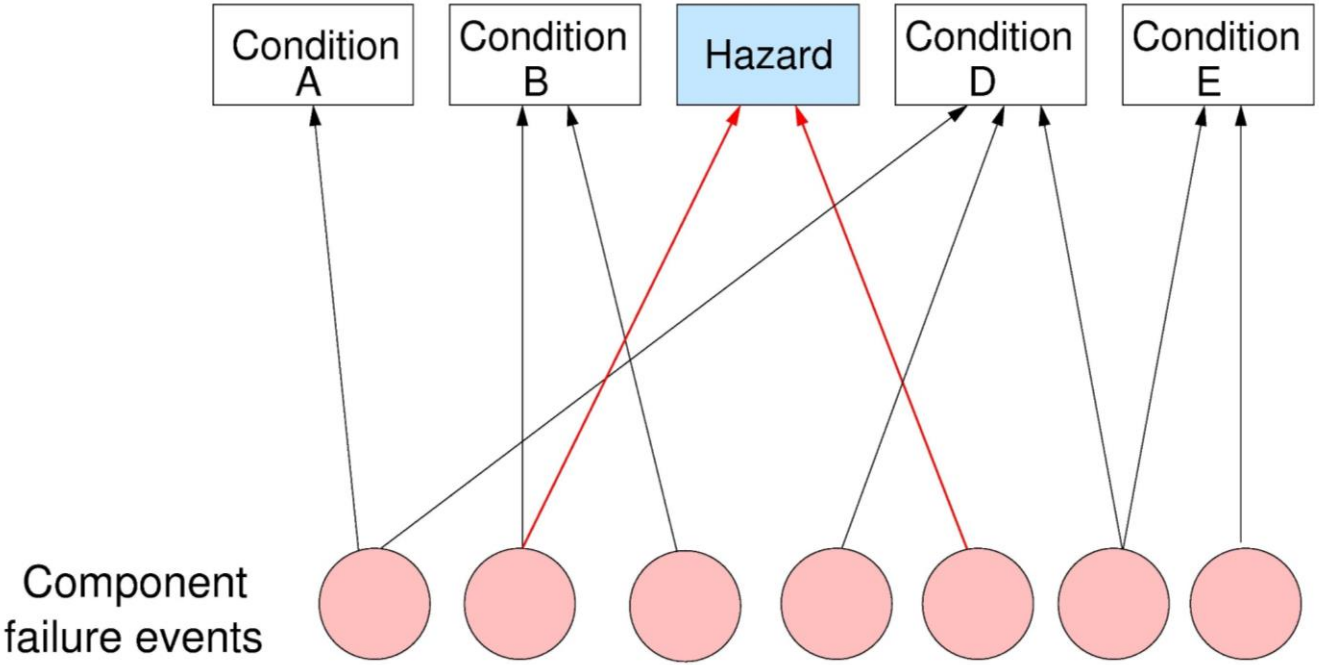
Quick review:

Hazard Analysis Search Strategies

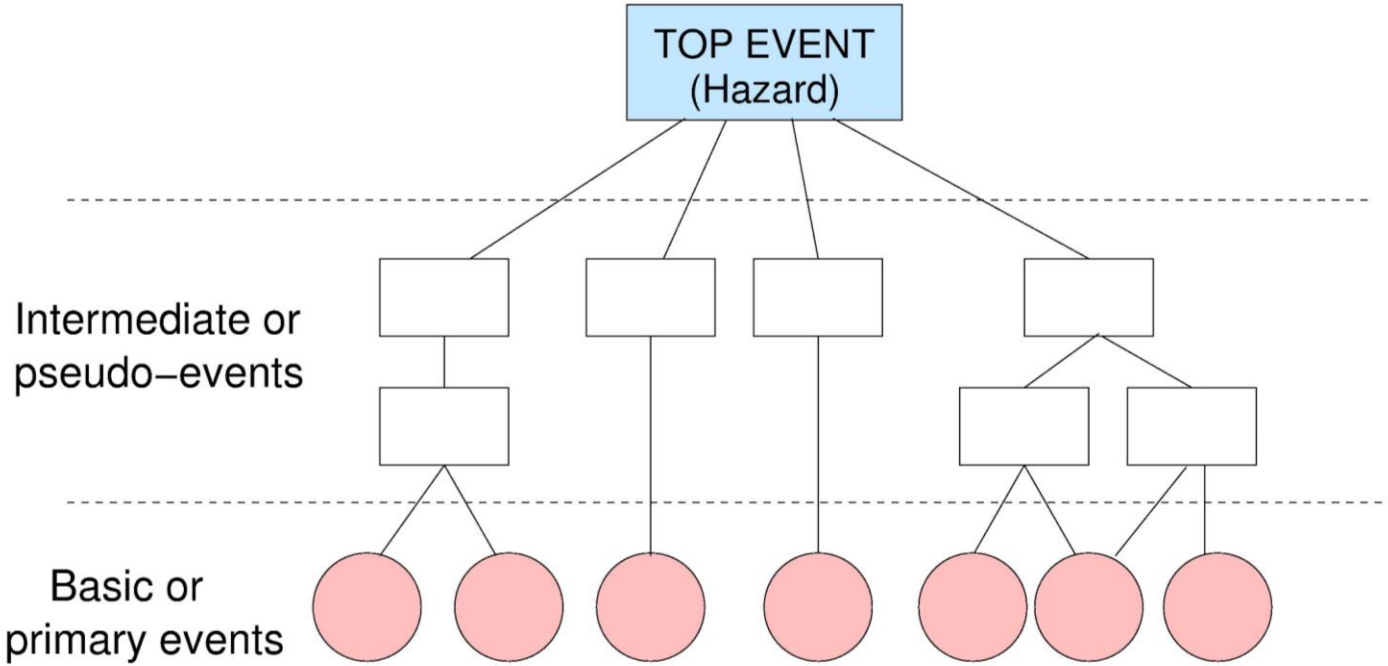
Forward vs. Backward Search



Bottom-Up Search

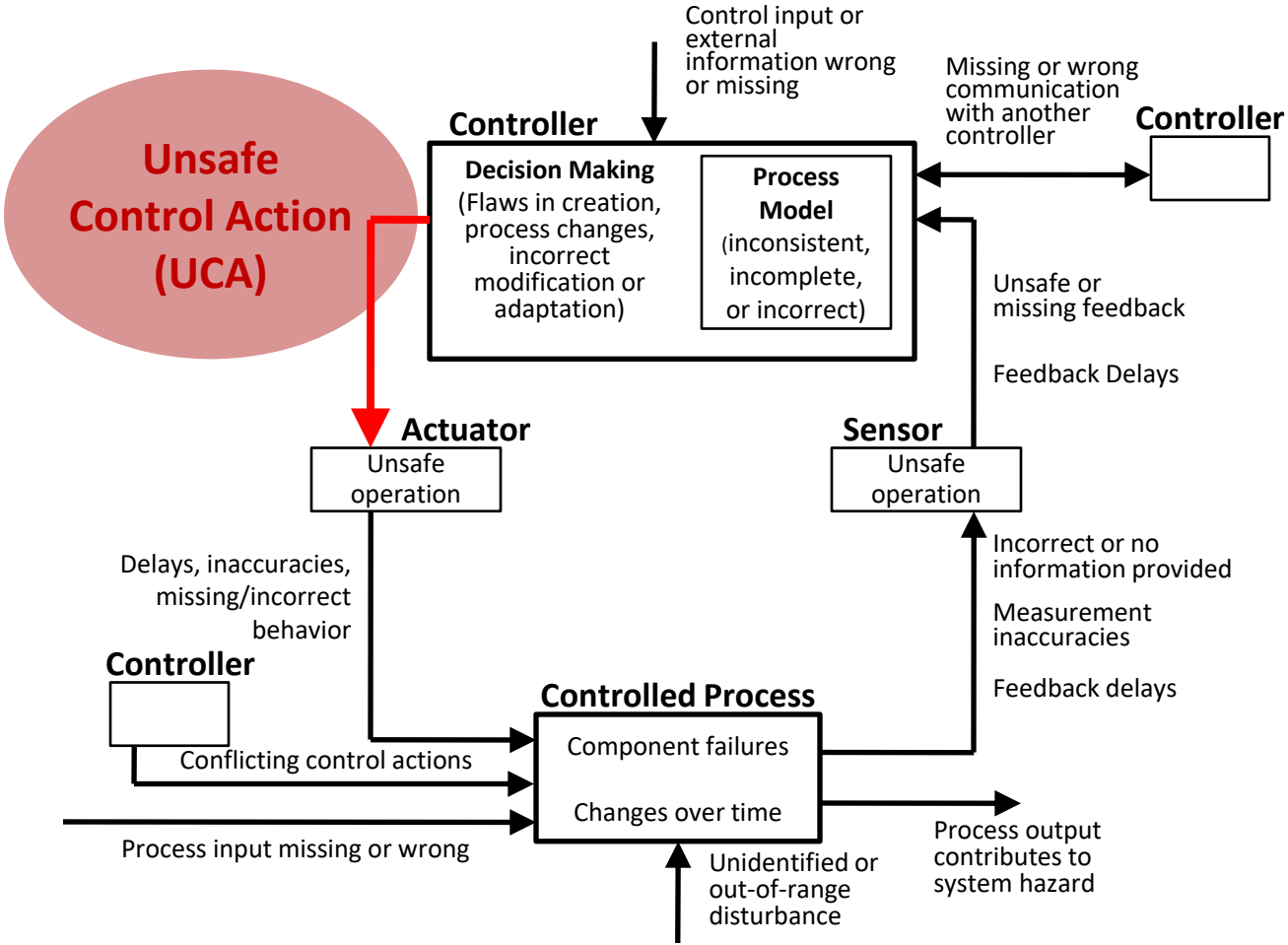


Top-Down Search



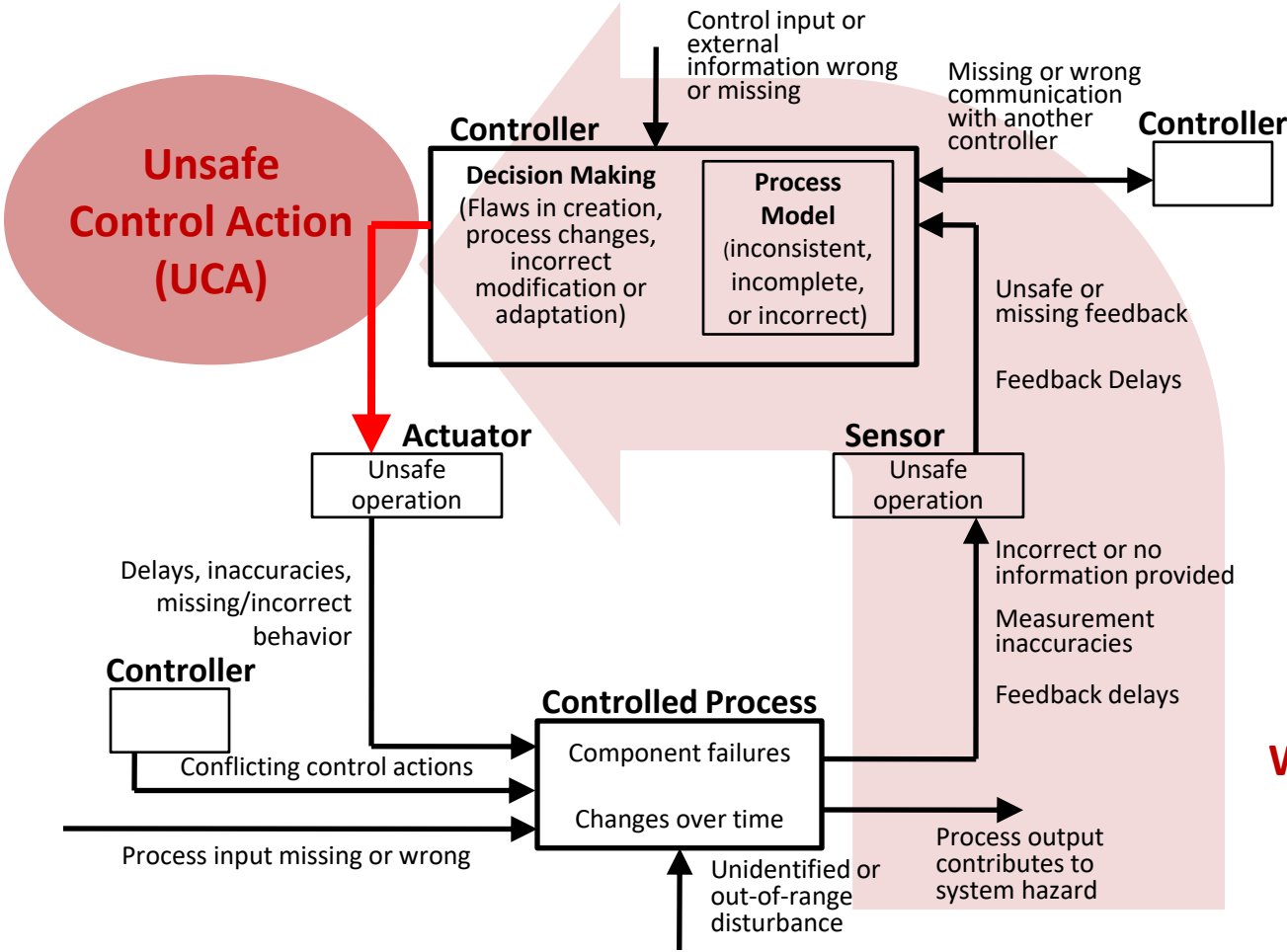
STPA Step 4: Scenario Building
The Old Process

STPA Step 4A: Identify scenarios that cause UCAs



© Copyright John Thomas 2017

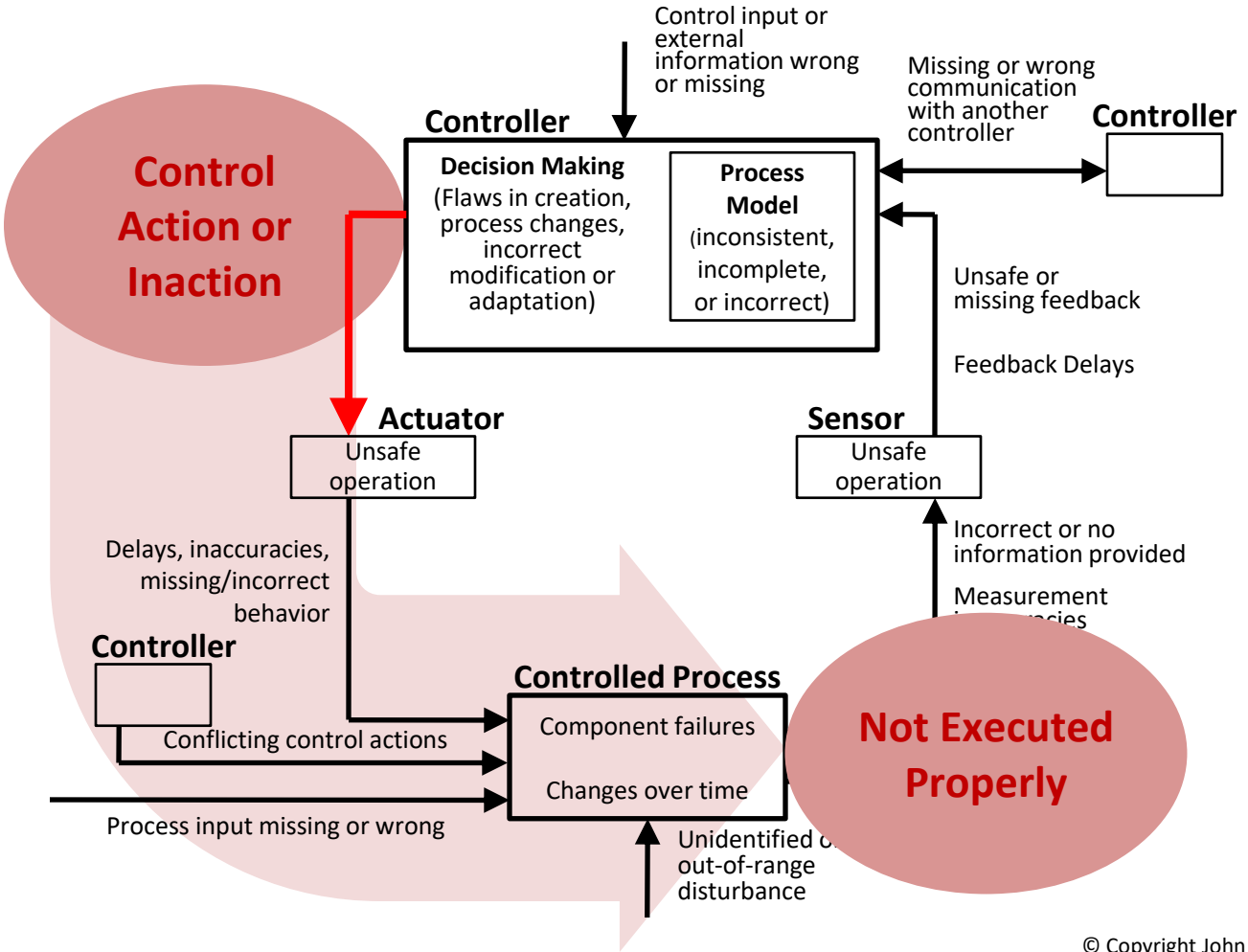
STPA Step 4A: Identify scenarios that cause UCAs



Why?

© Copyright John Thomas 2017

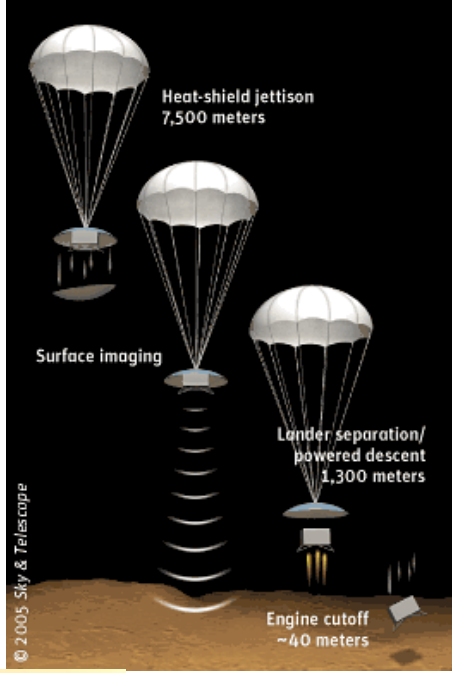
STPA Step 4B: Potential control actions not followed properly



© Copyright John Thomas 2017

Mars Polar Lander

- During descent to Mars, legs deployed (as planned)
- Footpad sensors detected vibration (within design spec)
- Momentary signal sent to computer (as required)
- Computer shut down the descent engines (as specified)
- The vehicle free-fell, fell to surface at 50 mph (80 kph), destroyed





© Copyright 2023 John Thomas



Example: Mars Polar Lander

Process Model: Incorrectly believes lander is on ground

Thruster Controller

Control Algorithm

Process Model (beliefs)

Feedback: Touchdown indication received when in air

Unsafe Control Action (UCA): Computer provides Disable-Thruster cmd when lander is in the air

Disable Thruster

Touchdown

Physical Interaction: Simultaneous leg vibration during leg deployment

Lander

© Copyright 2023 John Thomas

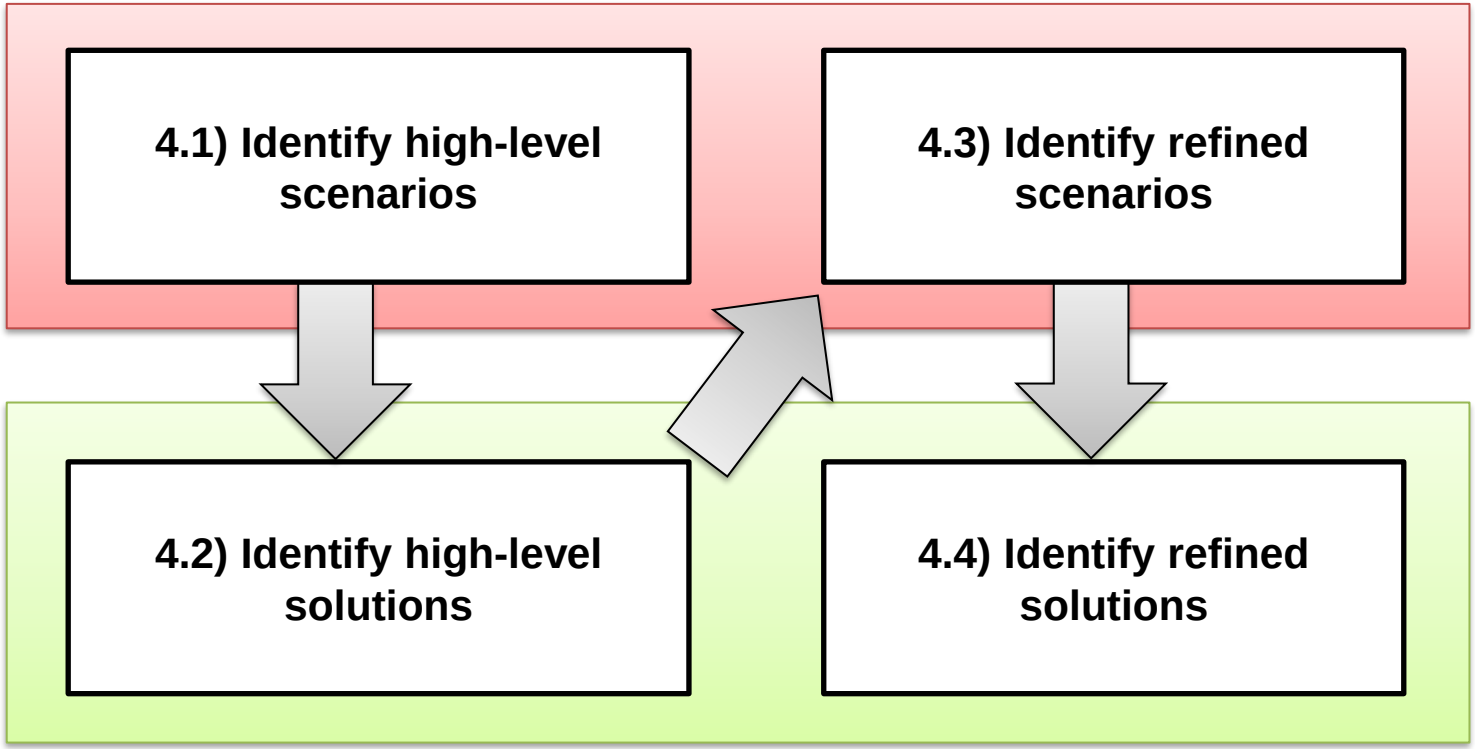
STPA Scenario Development

The New, Formal Process

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?

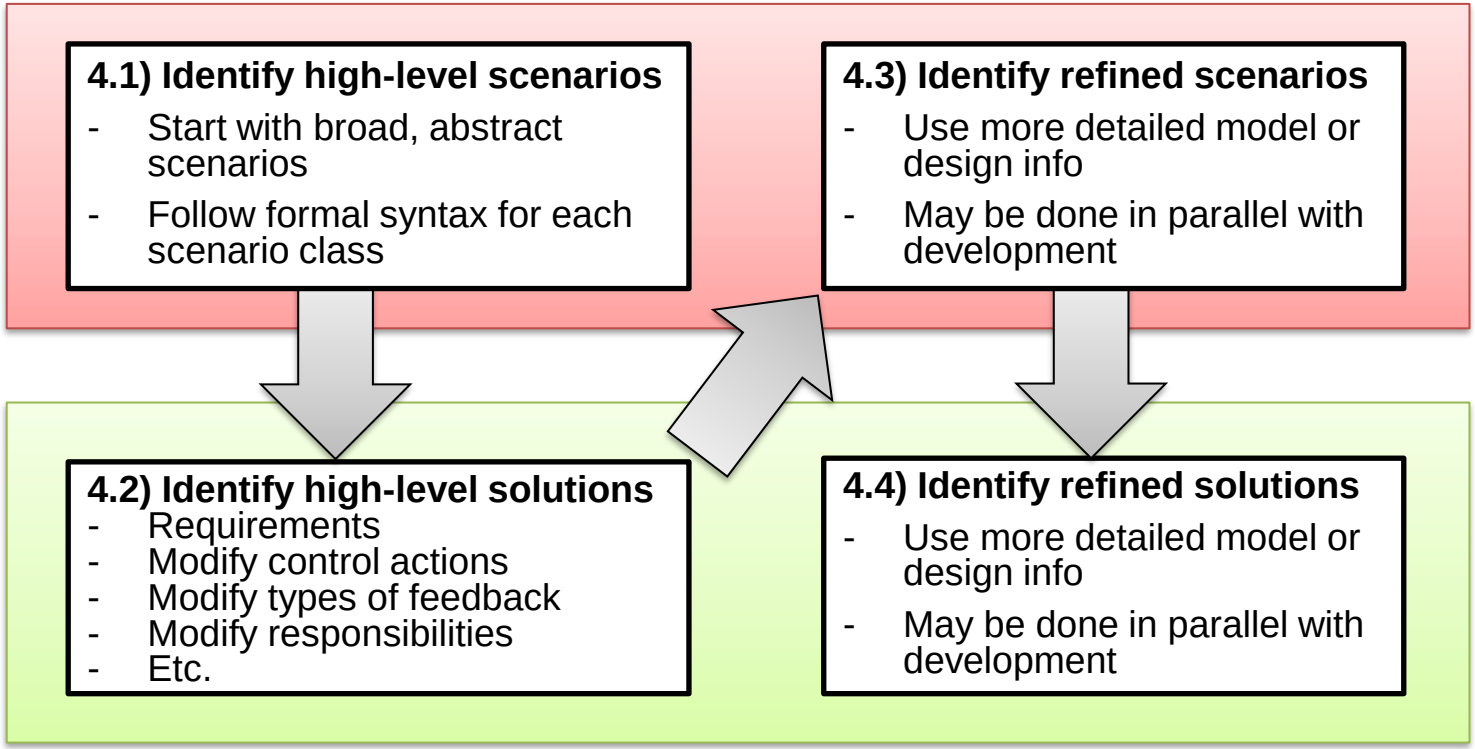


© Copyright 2024 John Thomas

STPA: Formal Scenario Building Process

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?

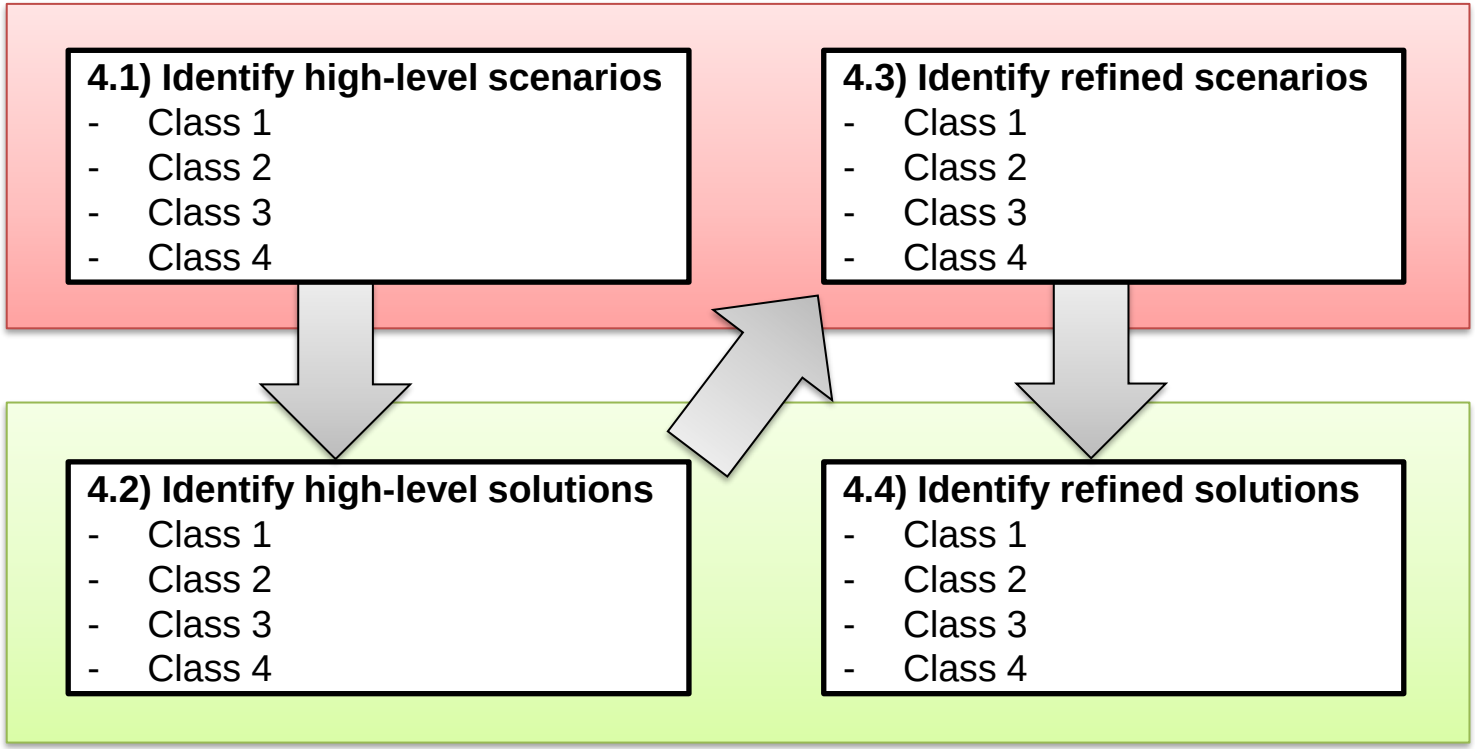


© Copyright 2024 John Thomas

STPA: Building Scenarios

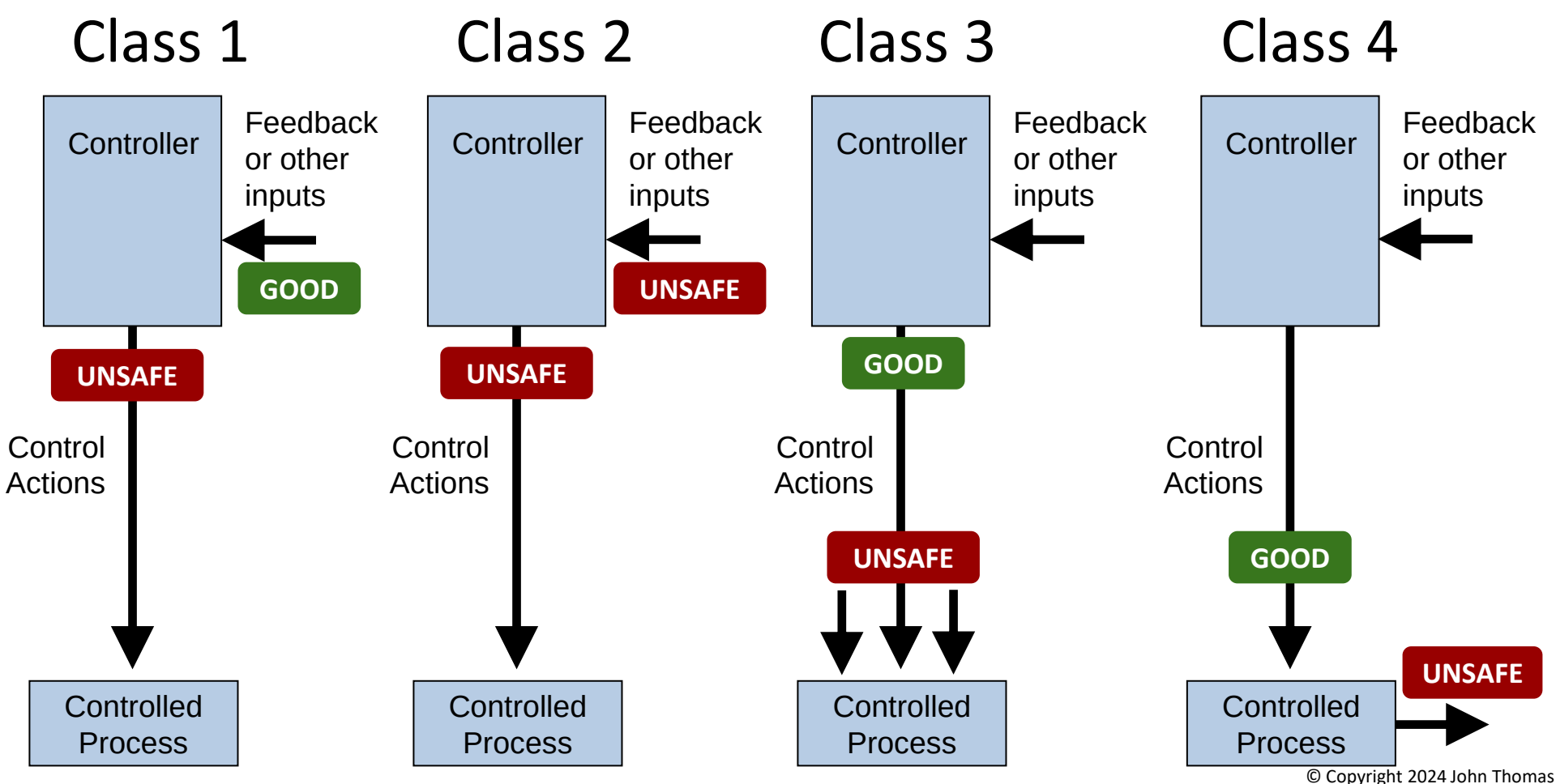
Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



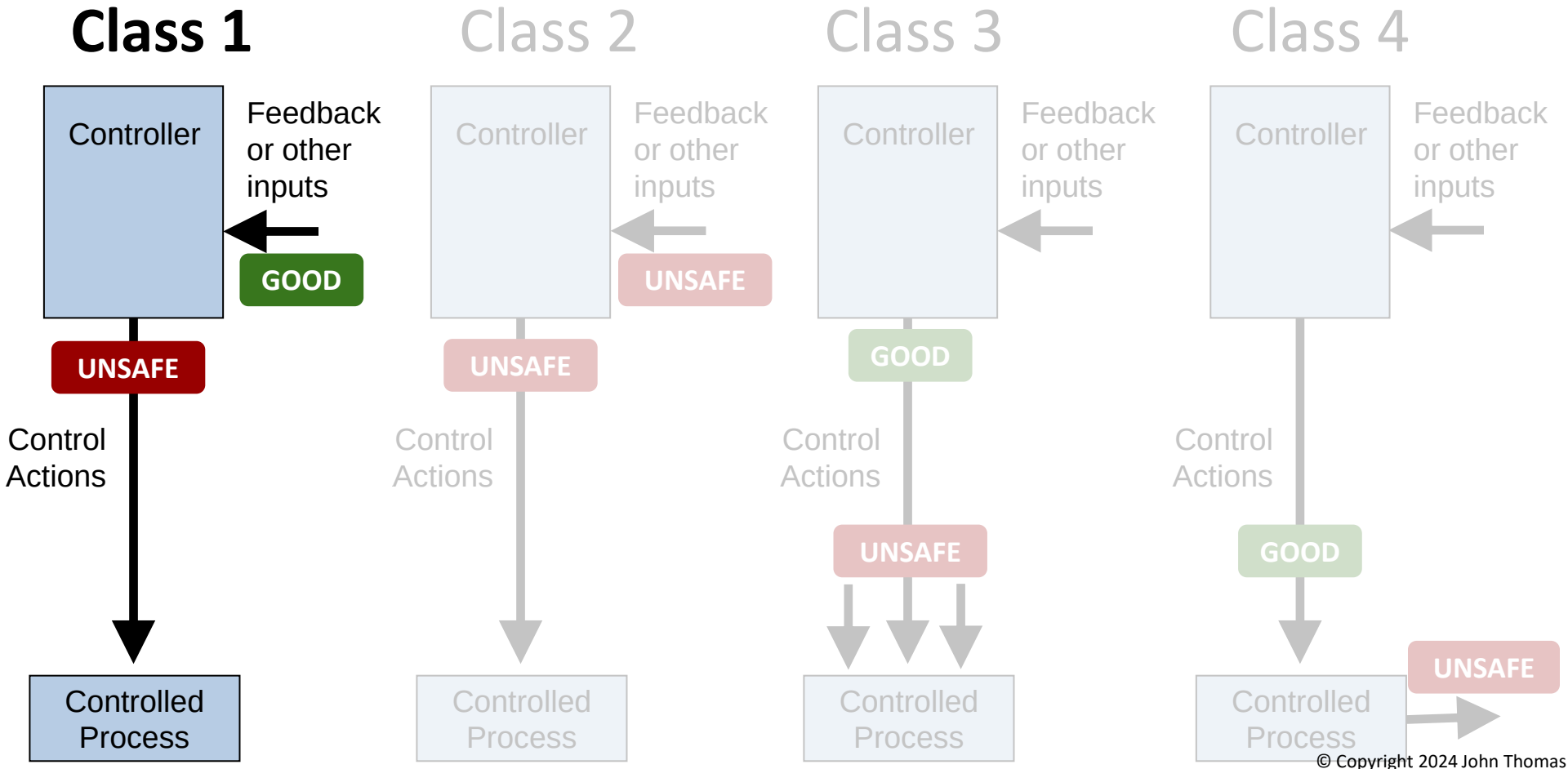
© Copyright 2024 John Thomas

STPA: Four Classes of Formal Scenarios



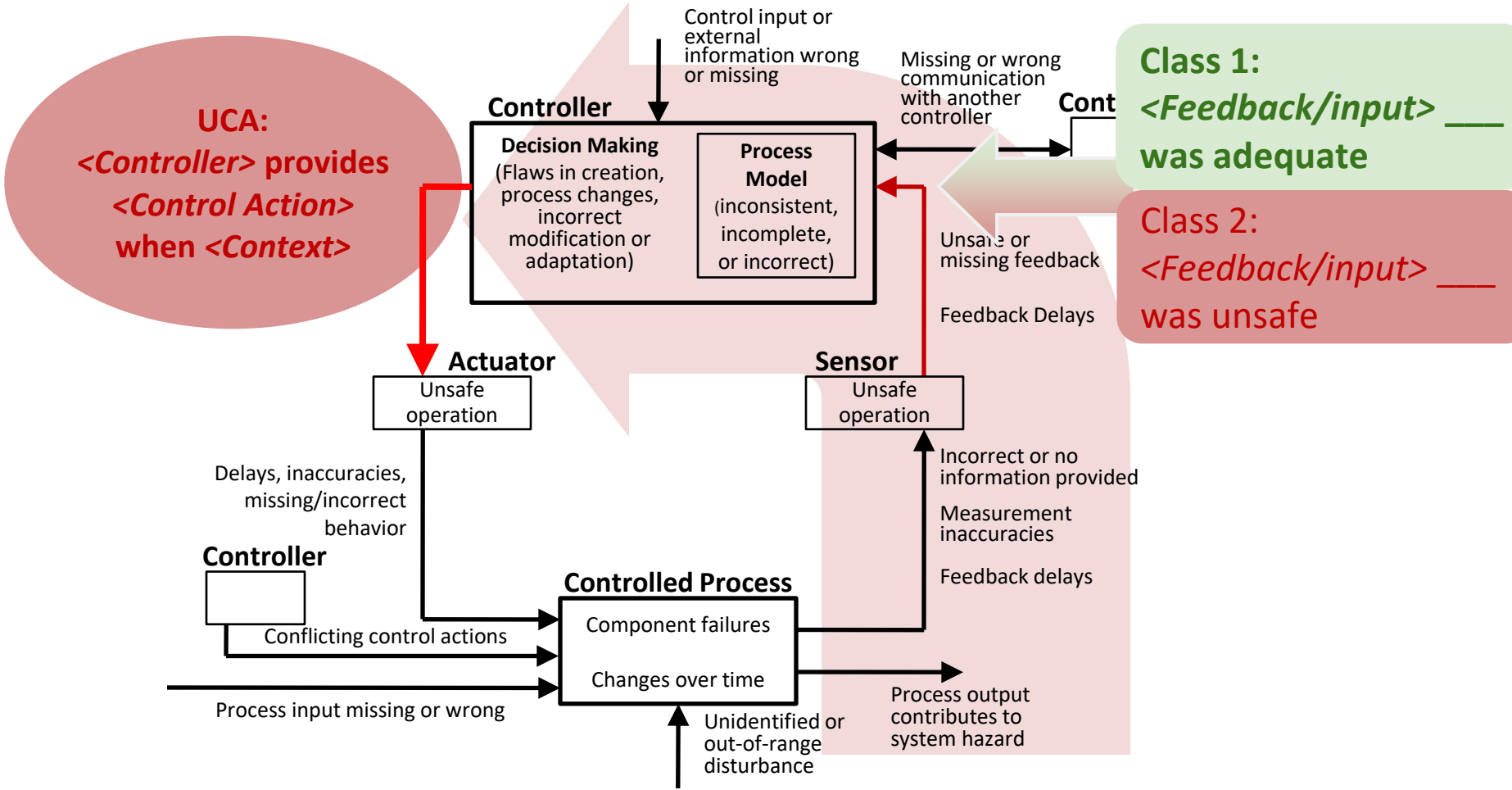
© Copyright 2024 John Thomas

Four Classes of Formal Scenarios



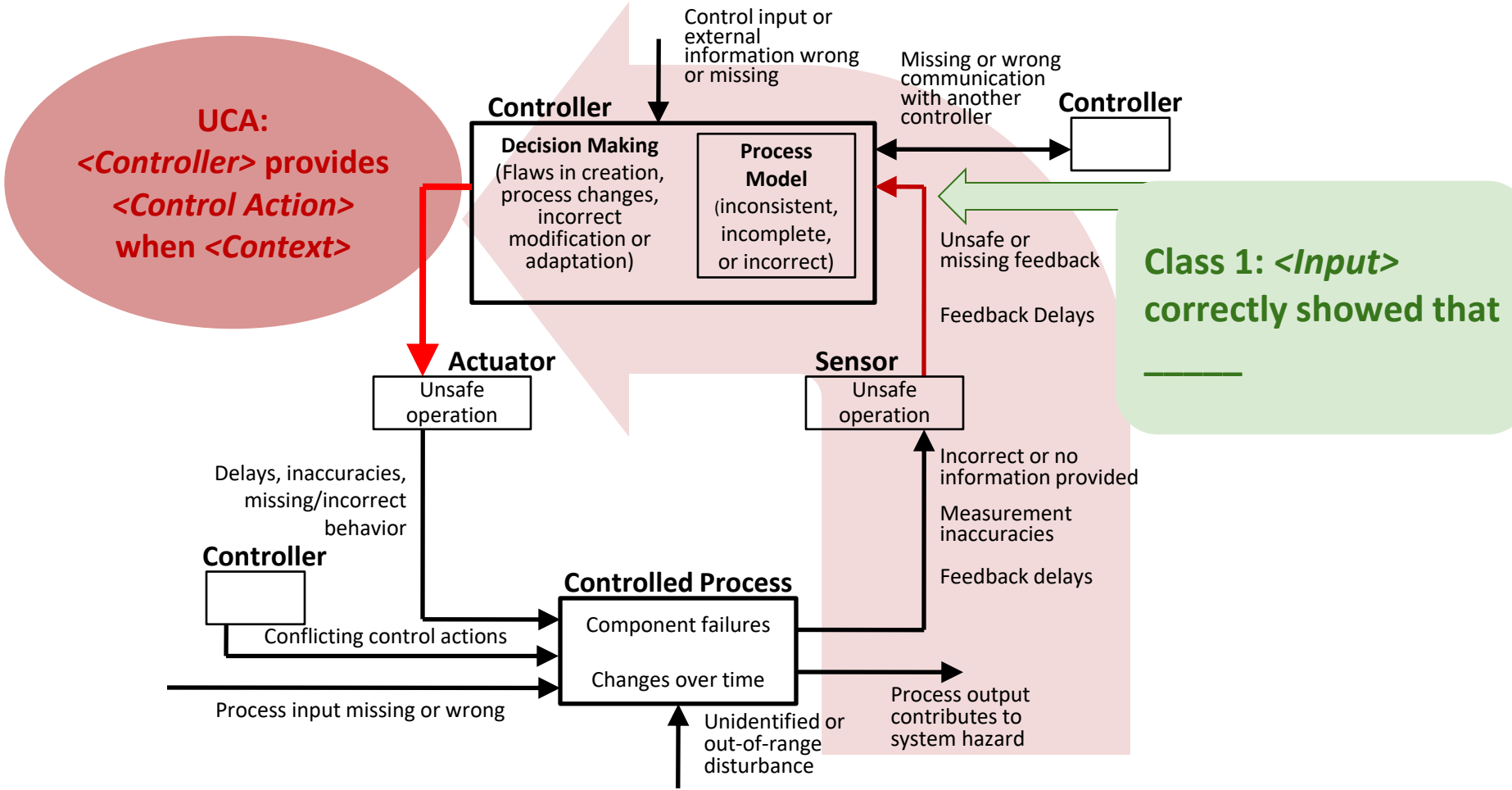
© Copyright 2024 John Thomas

STPA: Identify scenarios that cause UCAs



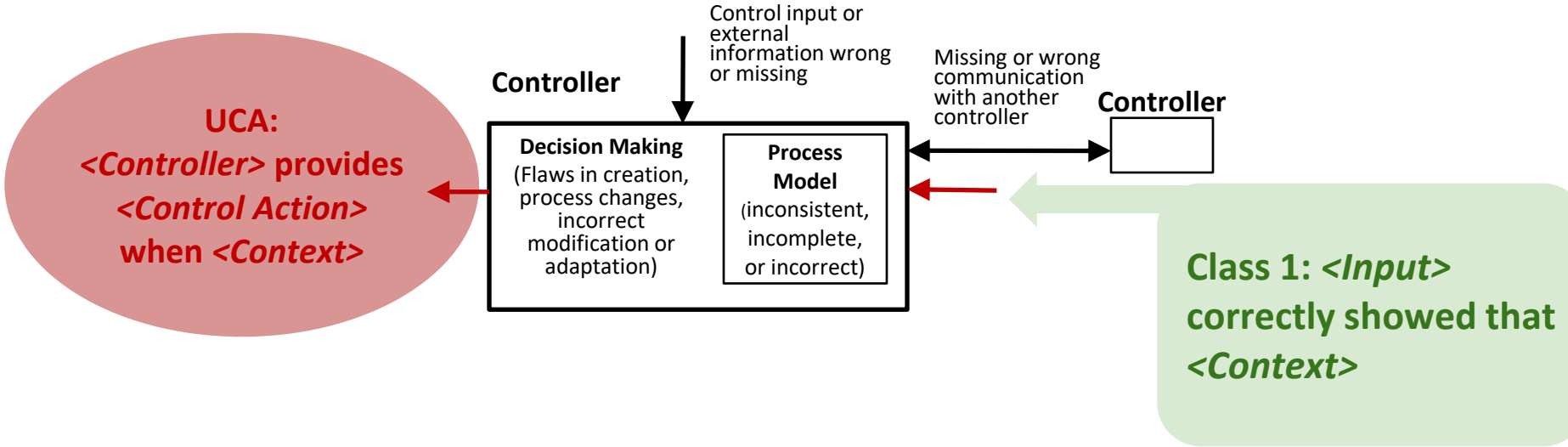
© Copyright 2024 John Thomas

STPA: Class 1 Scenario Archetype



© Copyright 2024 John Thomas

STPA: Class 1 Scenario Archetype

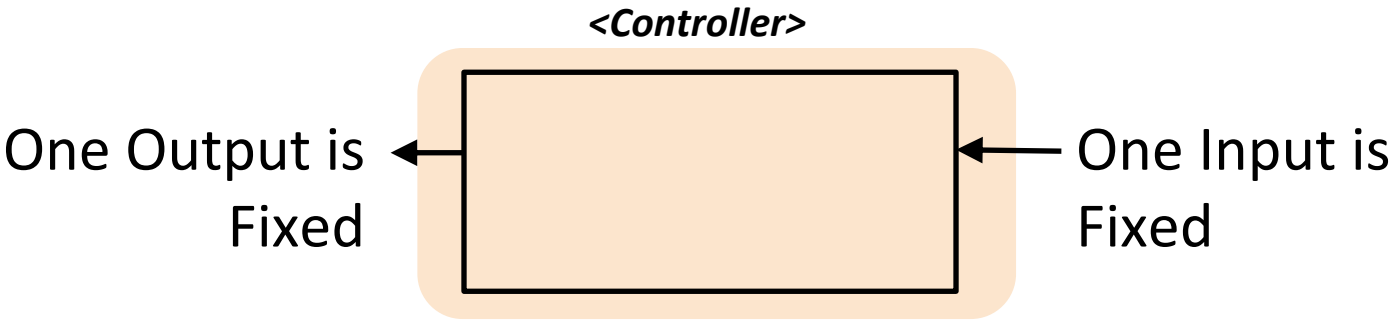


Class 1 Scenario Archetype:

- **Output**: UCA: *<Controller>* provides *<Control Action>* when *<Context>*
- **Input**: *<Input>* correctly showed that *<Context>*

© Copyright 2024 John Thomas

General Transfer Function Concept

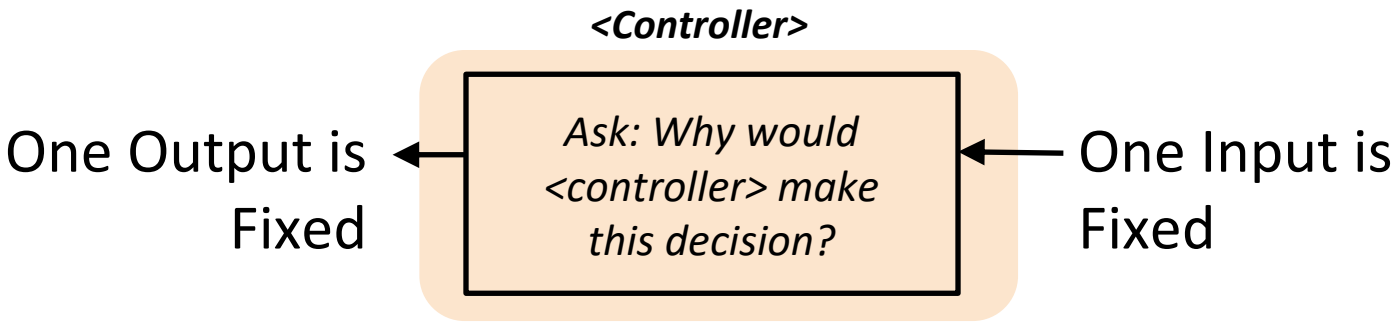


Class 1 Scenario Archetype:

- **Output**: UCA: *<Controller>* provides *<Control Action>* when *<Context>*
- **Input**: *<Input>* correctly showed that *<Context>*

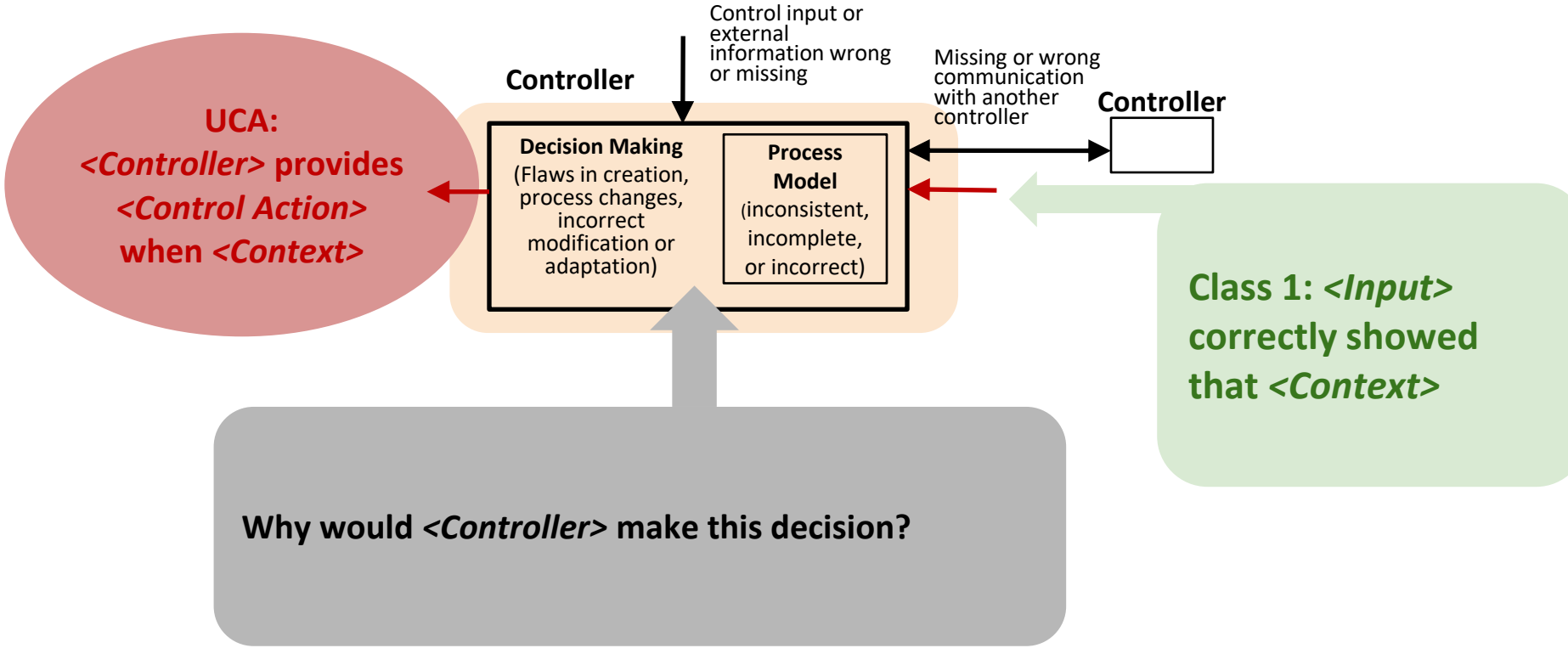
© Copyright 2024 John Thomas

General Transfer Function Concept



© Copyright 2024 John Thomas

STPA: Class 1 Scenario Archetype



We'll refine these scenarios and identify causes in another step (4.3).

© Copyright 2024 John Thomas

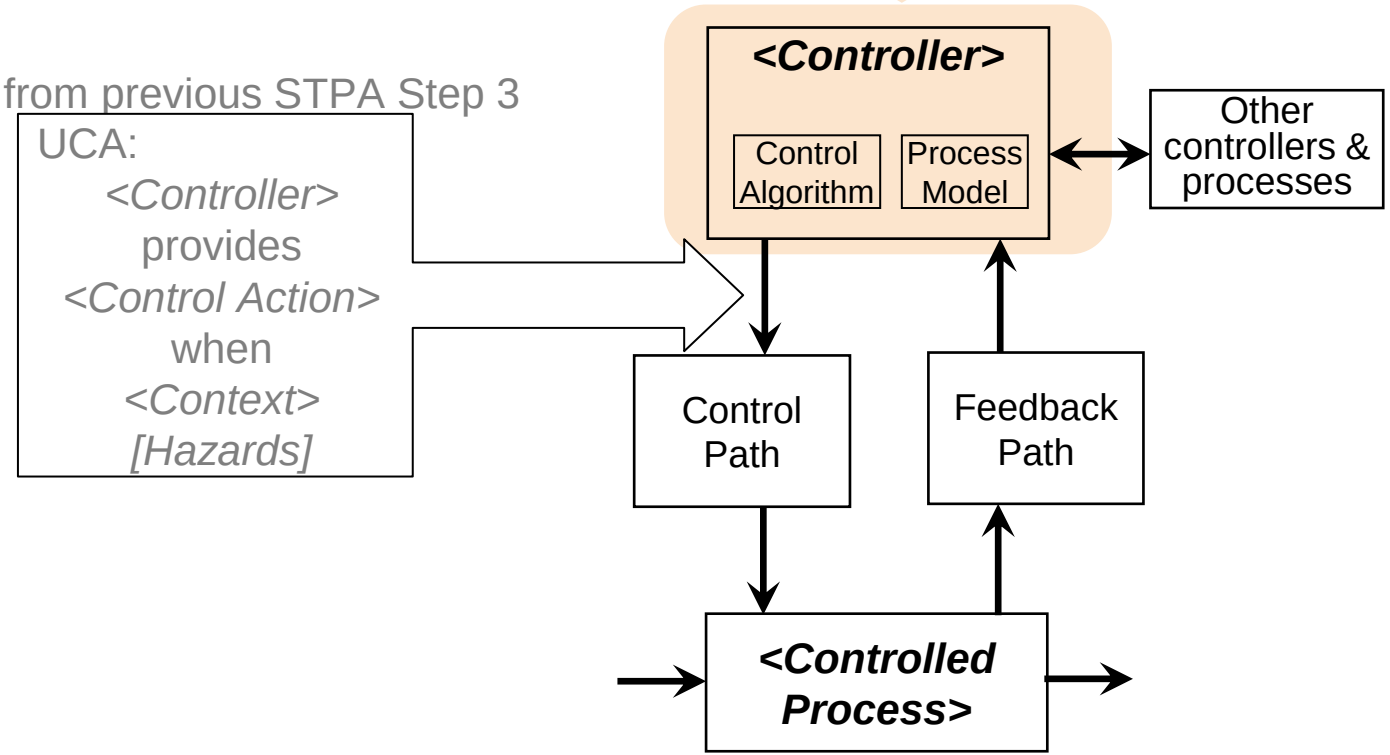
Class 1 Scenario Archetype:

Unsafe Controller Behavior

- UCA: <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Result from previous STPA Step 3

UCA:
<*Controller*>
provides
<*Control Action*>
when
<*Context*>
[Hazards]



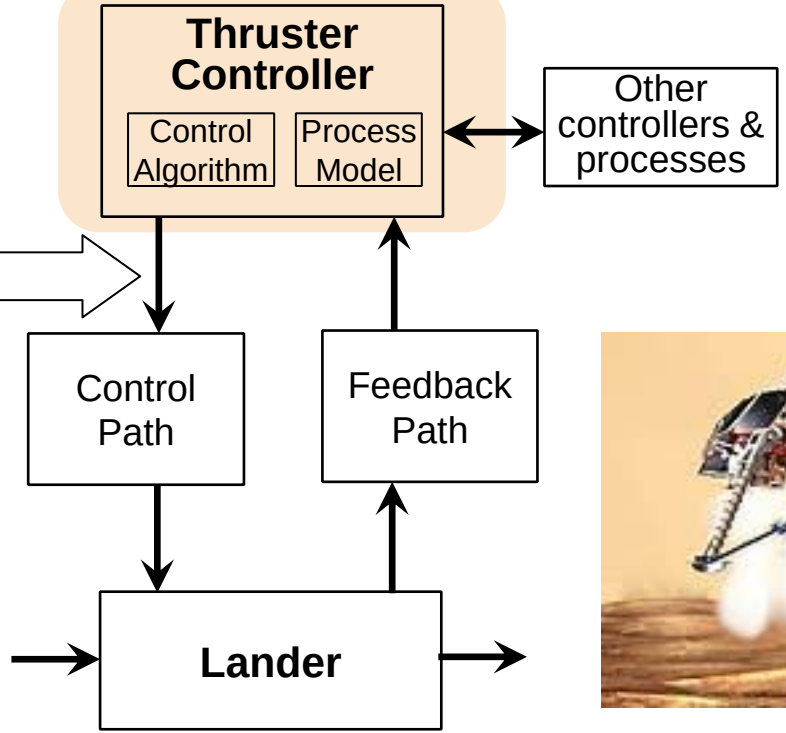
© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: <*Controller*> provides <*Control Action*> when <Context>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*





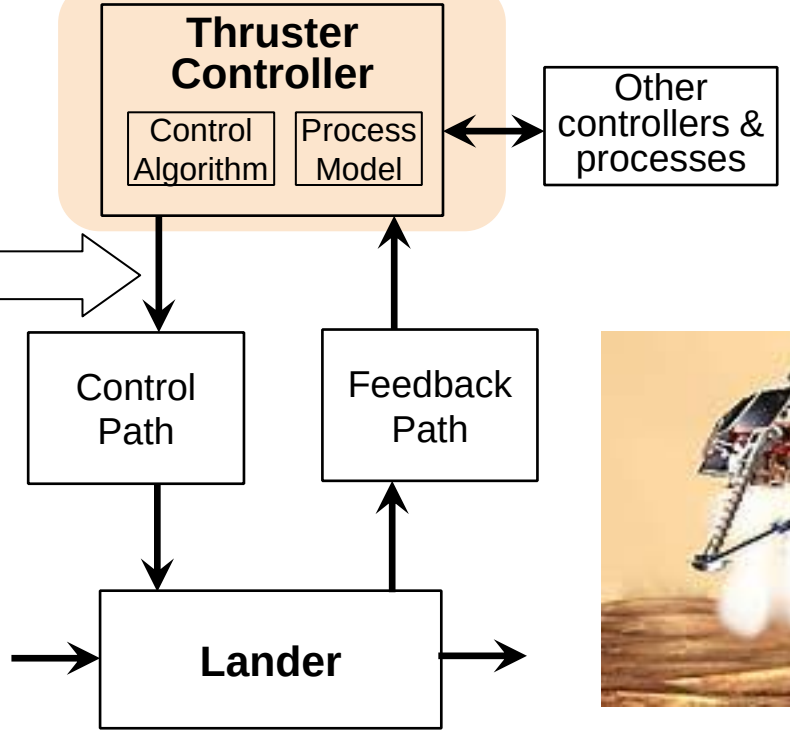
© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *<Input>* to *<Controller>* correctly indicates *<Context>*

Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*





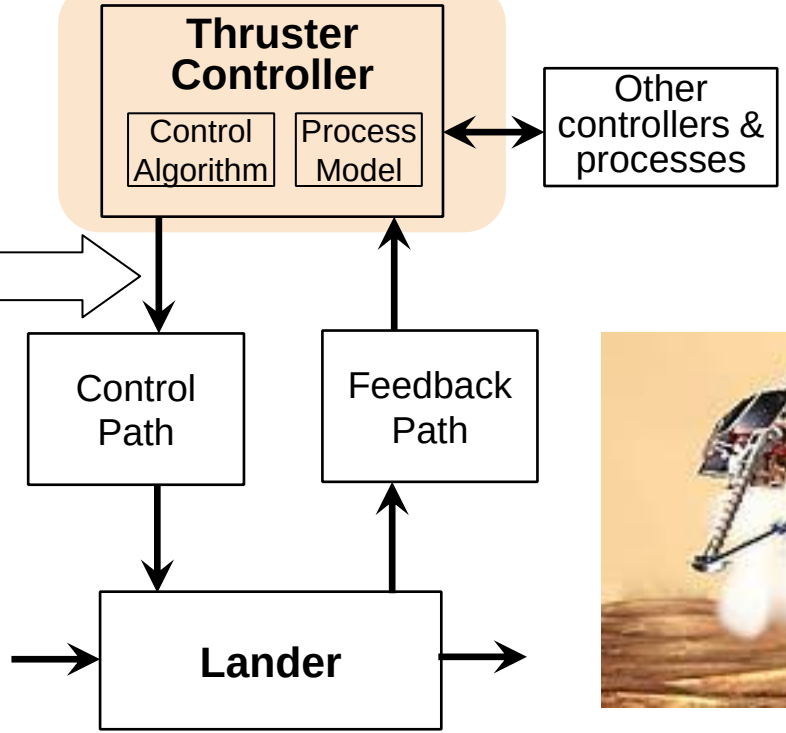
© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *<Input>* to *Thruster Controller* correctly indicates *lander is in the air*

Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*





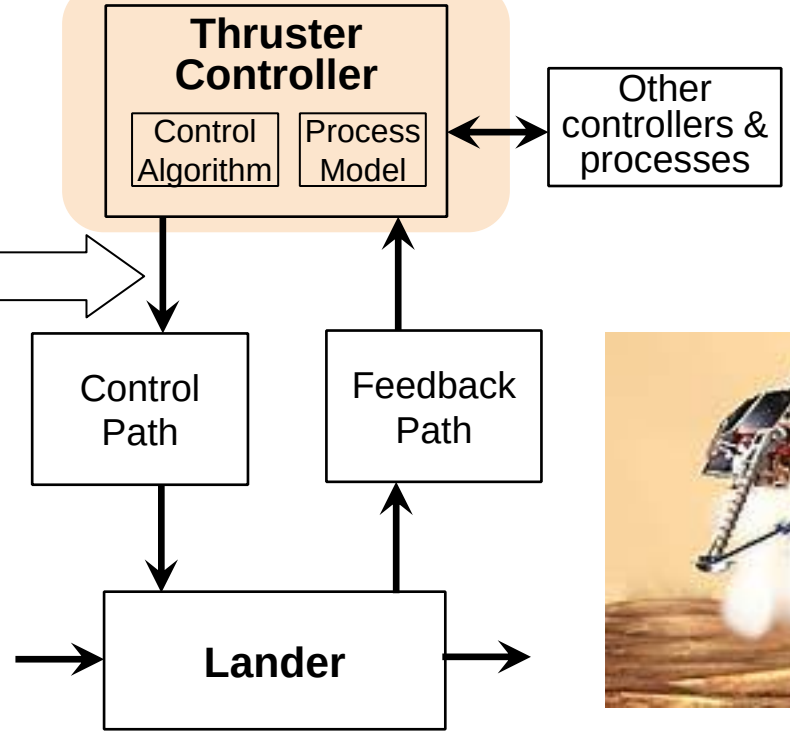
© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*

Result from previous STPA Step 3

UCA:	*UCA-2:*
<Controller> →	*Thruster Controller*
provides →	*provides*
<Control Action> →	*Disable-Thruster Cmd*
when →	*when*
<Context> →	*lander is in the air*
[Hazards] →	*[H-1]*



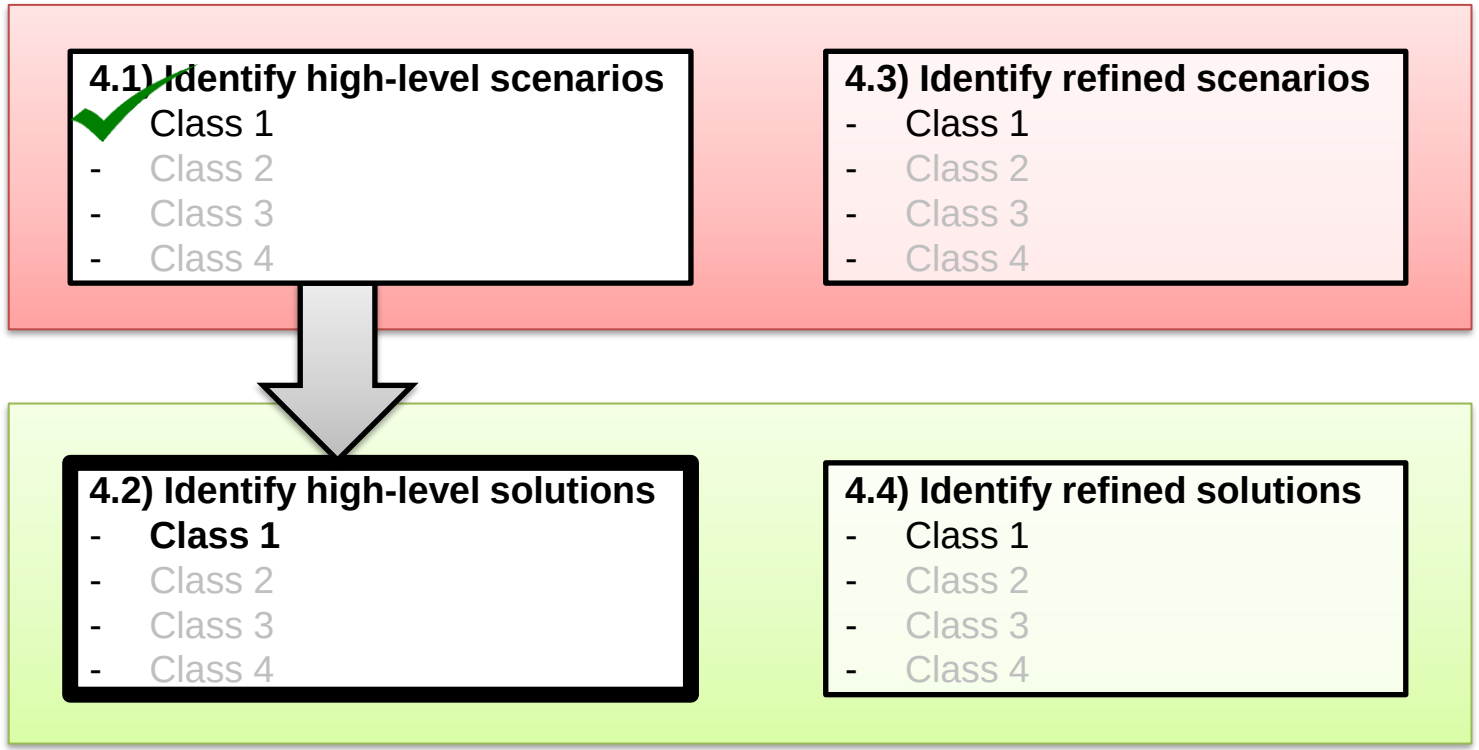


© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

4.2) Identify High-level Solutions

What is needed from each part of the control structure to prevent the scenario? For example:

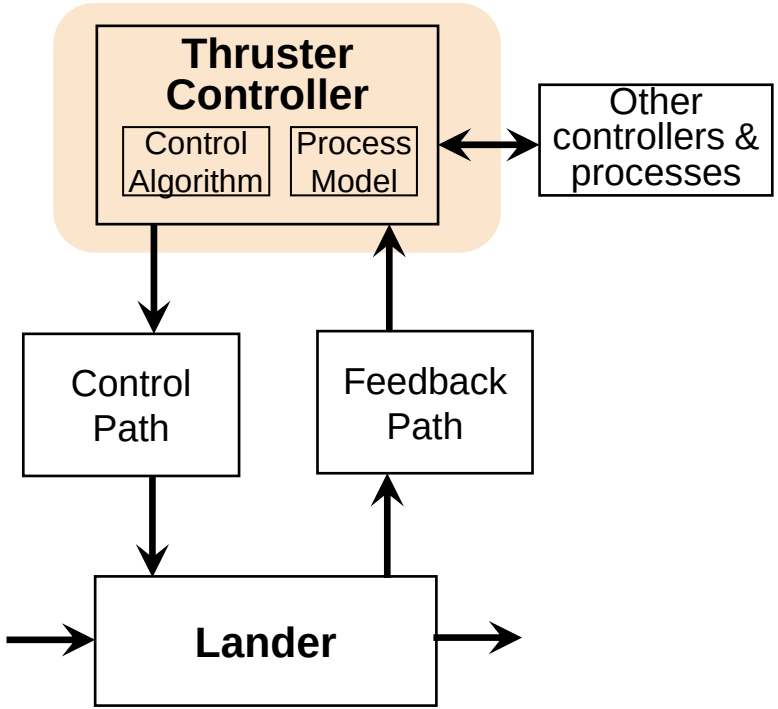
Structural mitigations

- Add/remove control actions?
- Add/remove controllers?
- Add/remove feedback?
- Add/remove processes?

Behavioral mitigations

- <*Controller*> must ___ when ____
- <*Feedback*> must ___ when ___
- <*Controller*> is/is not responsible for _____
- <*Controller*> Control Algorithm must ____
- <*Controller*> Process Model must include ____ (e.g., make controller aware of some other state)
- <*Process*> must ____

Others?



Discussion: Solutions and decisions are guided by the scenarios that must be prevented. Can help minimize unnecessary complexity.

© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*



4.2) Identify High-level Solutions

What can be done to prevent or mitigate this scenario?

- Process behavior: Add hardware / mechanical interlock
  - Mechanically prevent thruster cutoff before physical landing?
- Controller behavior: Include software interlock
  - Software must prevent thruster cutoff without simultaneous landing indication?
- Other solutions / mitigations?

© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*



4.2) Identify High-level Solutions

What can be done to prevent or mitigate this scenario?

- Process behavior: Add hardware / mechanical interlock
  - Mechanically prevent thruster cutoff before physical landing?
- Controller behavior: Include software interlock
  - Software must prevent thruster cutoff without an active landing indication
- Other solutions / mitigations?

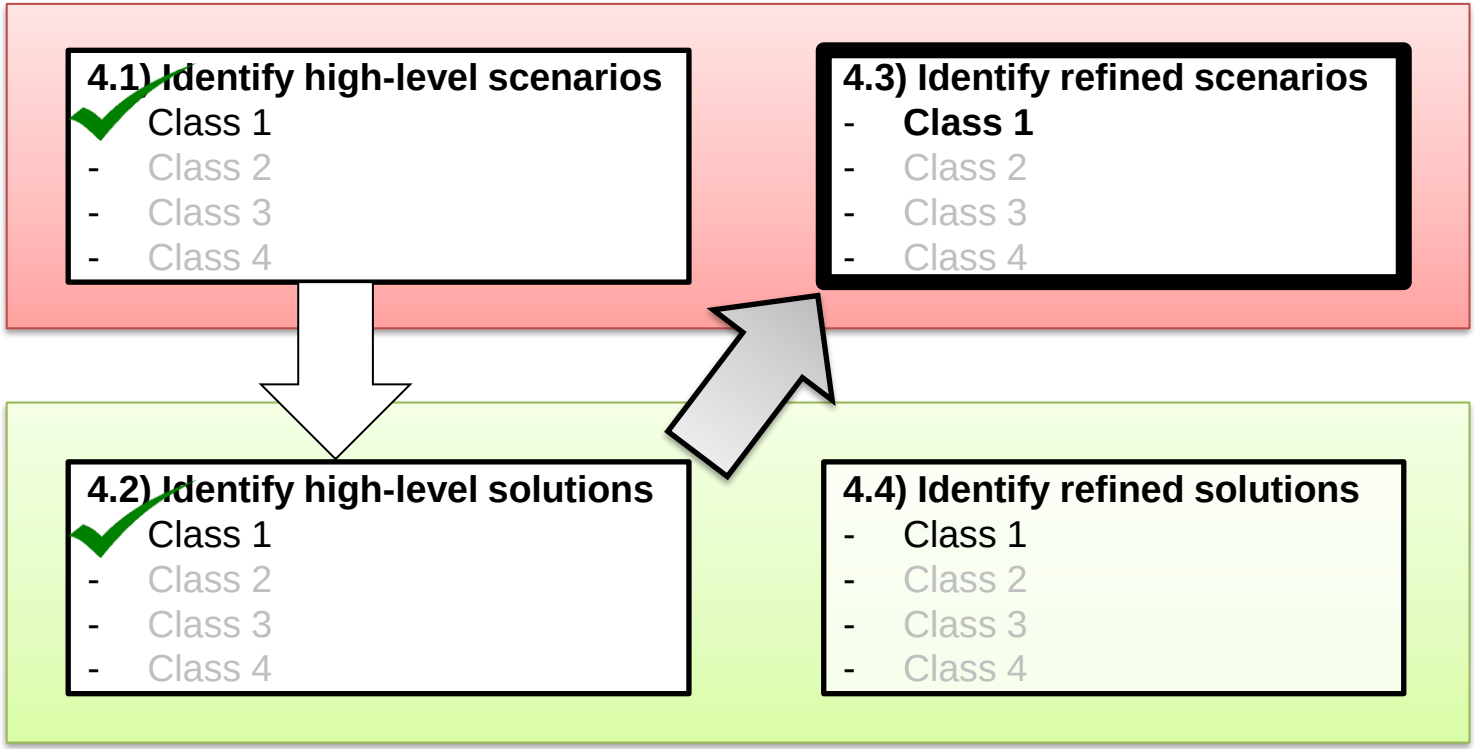
Discussion: These solutions can appear obvious when you're grounded in a scenario, which is a good thing. Imagine you don't have the scenario above, but you have a thousand pages of design info for MPL and a long list of failure-based safety assessments (e.g., software uses 3oo3 logic, so triple touchdown sensor failure is necessary to fool the software, but that likelihood is 10^{-9}). Missing functions like the one highlighted are not always obvious. Using structured scenario classes, like above, can make a missing requirement much more apparent. The actual MPL software was in fact missing the highlighted requirement, which contributed to the crash. It was overlooked by the safety assessment, which relied on failure-based methods like FTA and FMECA.

© Copyright 2024 John Thomas

STPA: Building Scenarios

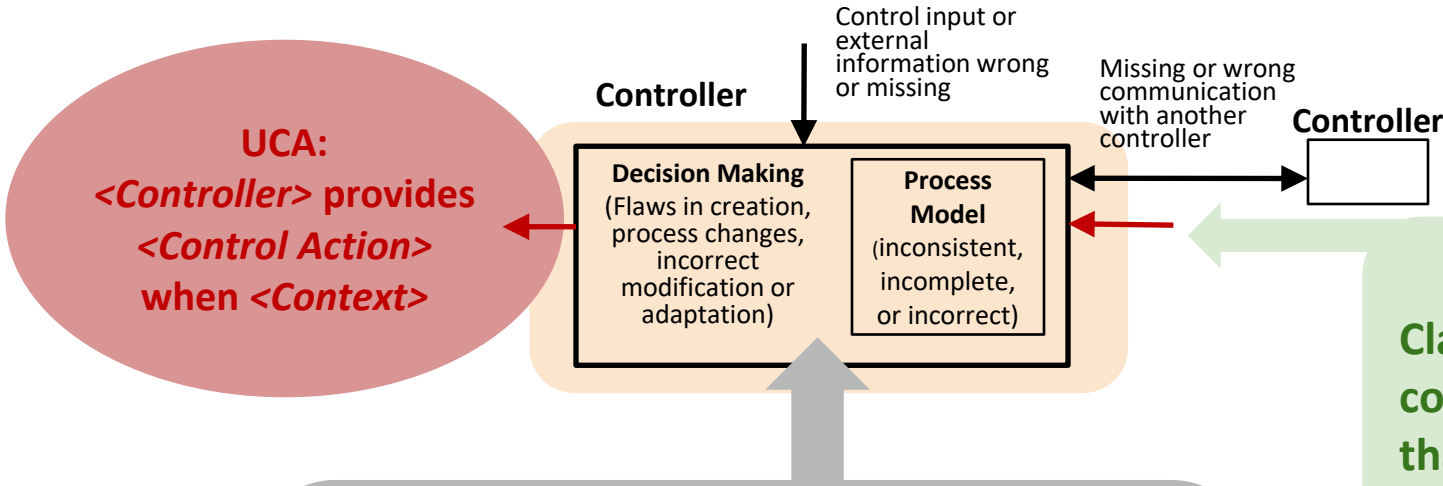
Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

STPA: Class 1 Scenario Archetype



© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Why would <Controller> make this decision?

1) Failure causes
2) Non-failure causes

A simple approach:
Go ask the SMEs!

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

Why would <Controller> make this decision?

Causes without failure:

- Because <Controller> is designed to default to ____ if _____ (will ignore ____ and ____)
- Because <Controller> also received ____, so it appeared reasonable to <Control Action>
- Because there is no other _____, so other control actions were not feasible
- Etc.

Example SMEs responses from a real project.

Time required: under 15 minutes

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

Refinement Option 1: Ask SMEs to help identify causes.

Is there a more rigorous option?

For some projects, like early concept development or a "lightweight STPA", option 1 may be enough.

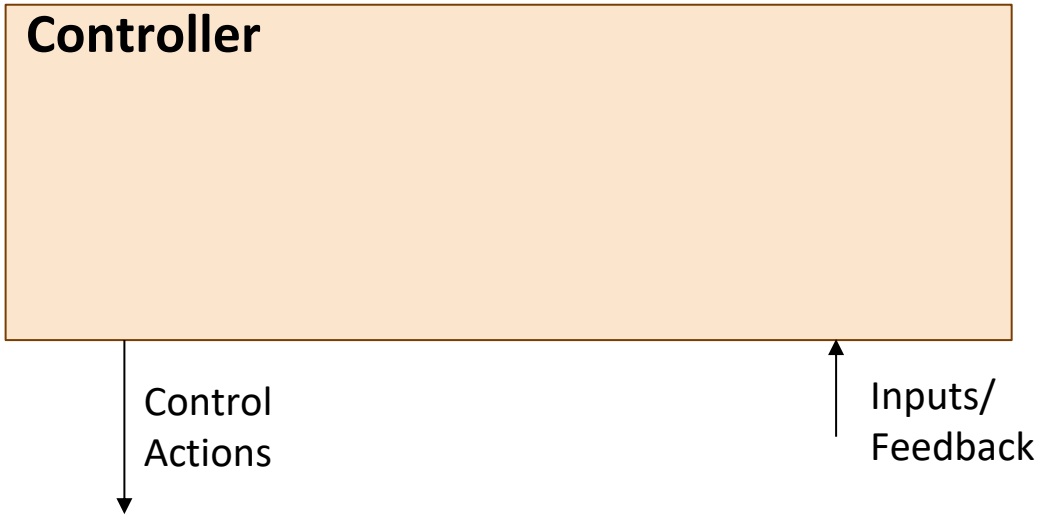
If more rigor needed, a generic controller model can be used to rigorously refine the scenarios and find actionable causes.

© Copyright 2024 John Thomas

Refinement Option 2:

Rigorous (Model-Based) Scenario Refinement

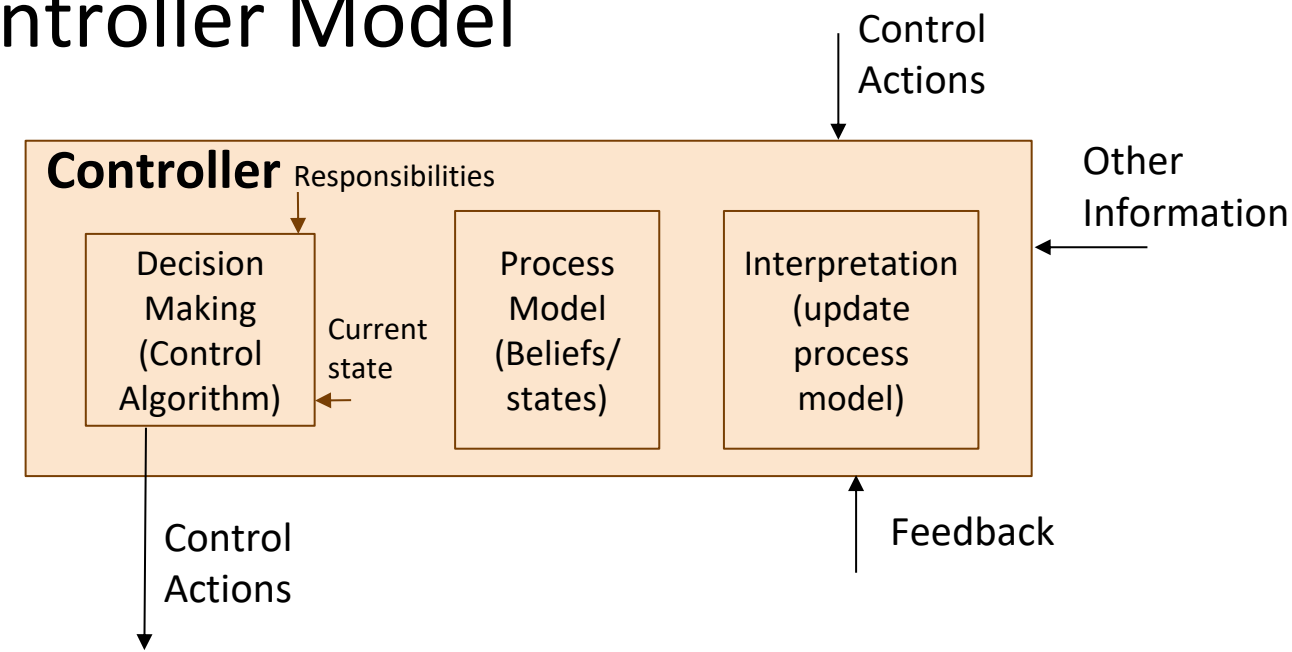
Generic Controller Model



Class 1 Scenario Archetype: **Output**: UCA **Input**: Correctly indicates UCA context (but other inputs may override or conflict)

© Copyright 2024 John Thomas

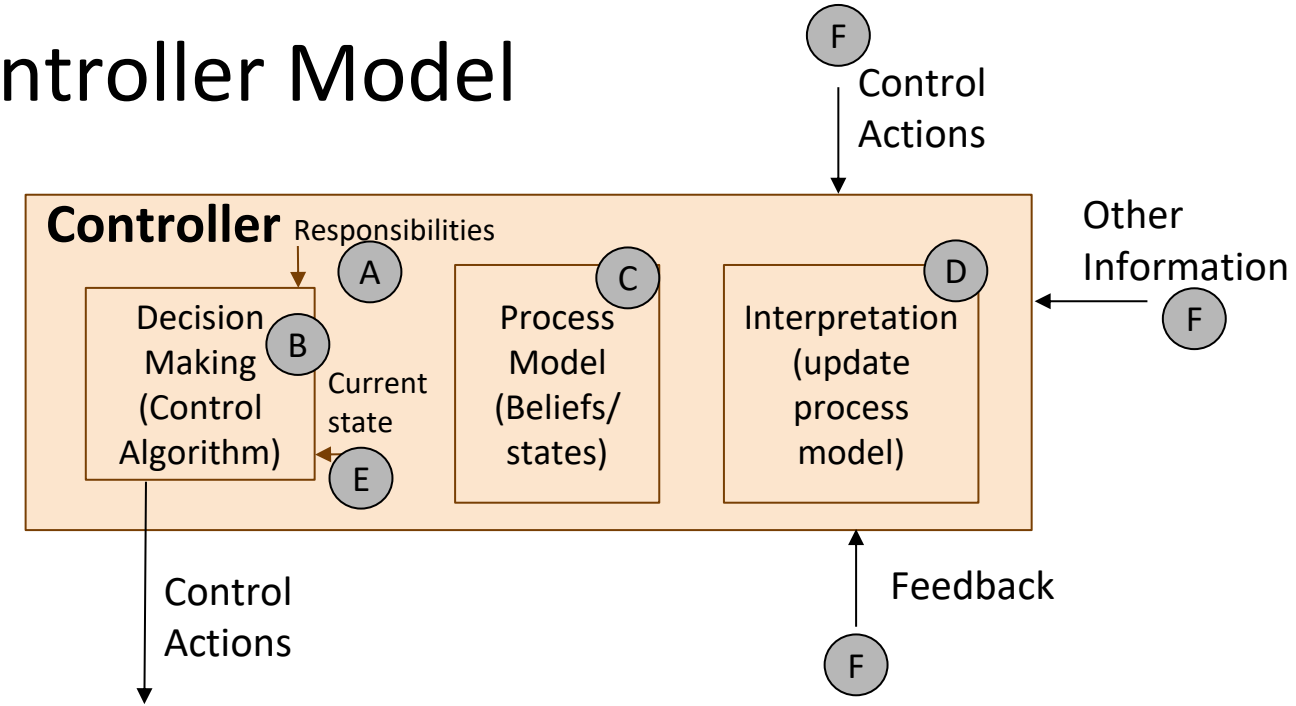
Generic Controller Model



Class 1 Scenario Archetype: **Output**: UCA **Input**: Correctly indicates UCA context (but other inputs may override or conflict)

© Copyright 2024 John Thomas

Generic Controller Model



Class 1 Scenario Archetype: **Output**: UCA **Input**: Correctly indicates UCA context (but other inputs may override or conflict)

Why would <controller> make this decision? → A D B E C F

© Copyright 2024 John Thomas

Refining Class 1 Scenario Archetype: Unsafe Controller Behavior

Common causes of Scenario Archetype 1:

A • Responsibilities (e.g., desired end states) that would produce this UCA

B • Control algorithms or decision-making rationale that would explain the UCA

C • Process models that would explain this UCA

D • Interpretation or process model updates that would explain the UCA

E • Internal controller states/modes that would explain the UCA (failure, lame duck mode, etc.)

F • Controller inputs (control actions, feedback, or other inputs) that would explain the UCA

- E.g., Conflicting/contradictory inputs, inputs from another controller, etc.
- If the input is another UCA, then make sure the new UCA is recorded in STPA Step 3. The new UCA will be analyzed using the same process.

G • Etc.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

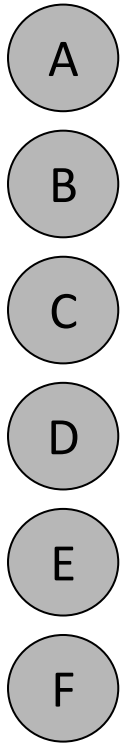
UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype



© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

A

Examples from real projects:

Responsibilities
<*Controller*> by design is responsible for always assigning ____ to every ___ (even if _____.)

Discussion: Responsibilities can be:
- Fixed (hardcoded, embedded in design of algorithm)
- Dynamic (selected in real-time, e.g., an external control action)
- Adaptive (developed and changed by the controller as needed)

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

B

Examples from real projects:

Control Algorithms
<*Controller*> control algorithm is designed to trigger <*Control Action*> during a fallback to _____ strategy, which can happen if _____

Decision Making
If ____, then <*Controller*> may select <*Control Action*> to _____

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

C

Process Models

PM-1: <Controller> incorrectly believes _____

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

C

Process Model

<*Controller*> believes _____

Discussion: Identifying a process model is usually not enough; must identify *why* it may happen so that it can be prevented or mitigated.

D

Update Process Model

PM-1 is not updated when _ PM-1 is the initial (default) belief before feedback/input ___ received
PM-1 is updated incorrectly due to feedback/input ___ that indicates ___
PM-1 is updated too late (or early) due to ___
PM-1 stops updating too soon before ____ PM-1 continues to be updated too long after ___

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

D

Update Process Model

PM-1 is not updated when _ PM-1 is the initial (default) belief before feedback/input ___ received
PM-1 is updated incorrectly due to feedback/input ___ that indicates ___
PM-1 is updated too late (or early) due to ___
PM-1 stops updating too soon before ____ PM-1 continues to be updated too long after ___

Not Provided Causes Hazard

Provided Causes Hazard

Too early / too late

Stopped too soon / Applied too long

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

C

Process Model

<*Controller*> believes _____

Why?

D

Update Process Model

<*Controller*> does not update PM-1 when ____

Why?

F

Other Feedback

<*Controller*> does not receive <*other feedback*> when <*other context*>

Why?

Unsafe Feedback (Class 2 Scenario)

- <*Controller*> does not receive <*other input/feedback*>
- <*Other context*> is true

© Copyright 2024 John Thomas

Refined Scenarios

C

Process Model

<Controller> believes _____

Why?

D

Update Process Model

<Controller> does not update PM-1 when ____

Why?

F

Other Feedback

<Controller> does not receive *<other input/feedback>* when *<other context>*

Why?

Unsafe Feedback (Class 2 Scenario)

- *<Controller>* does not receive *<other input/feedback>*
- *<Other context>* is true

Discussion: A class 1 scenario means that an input correctly indicates <context>. No assumption is made about any other inputs. There may be another input/feedback that provides conflicting info or indicates some other overriding context (like an emergency condition).

In these cases, the class 1 scenario refinement should identify the other input(s) that, together with *<UCA context>*, would trigger the UCA.

Stopping rule: The class 1 scenario refinement stops at the boundary of the controller and identifying what feedback/input would cause the UCA. The sources of feedback/inputs are analyzed in class 2 if needed.

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

D

Examples from real projects:

Interpretation of inputs

In some situations, <*Controller*> will interpret <*feedback/input*> as an indicator of ___. This interpretation can underestimate ___, causing ___.

When <*feedback/input 1*> conflicts with <*feedback/input 2*>, <*Controller*> may assume ____

When <*feedback/input*> is not available, <*Controller*> may assume ___

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

E

Examples from real projects:

Controller States / Modes

- If <*Controller*> is in ____ mode/state, it will continue to <*Control Action*> using alternate input ____.

- If ___ is disabled, then <*Controller*> can _____.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

F

Examples from real projects:

Other Inputs

<*Controller*> does not prevent <*Control Action*> when alternate input ___ is received.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype

Class 3 Scenario Archetype

Class 4 Scenario Archetype

F

Examples from real projects:

Other inputs

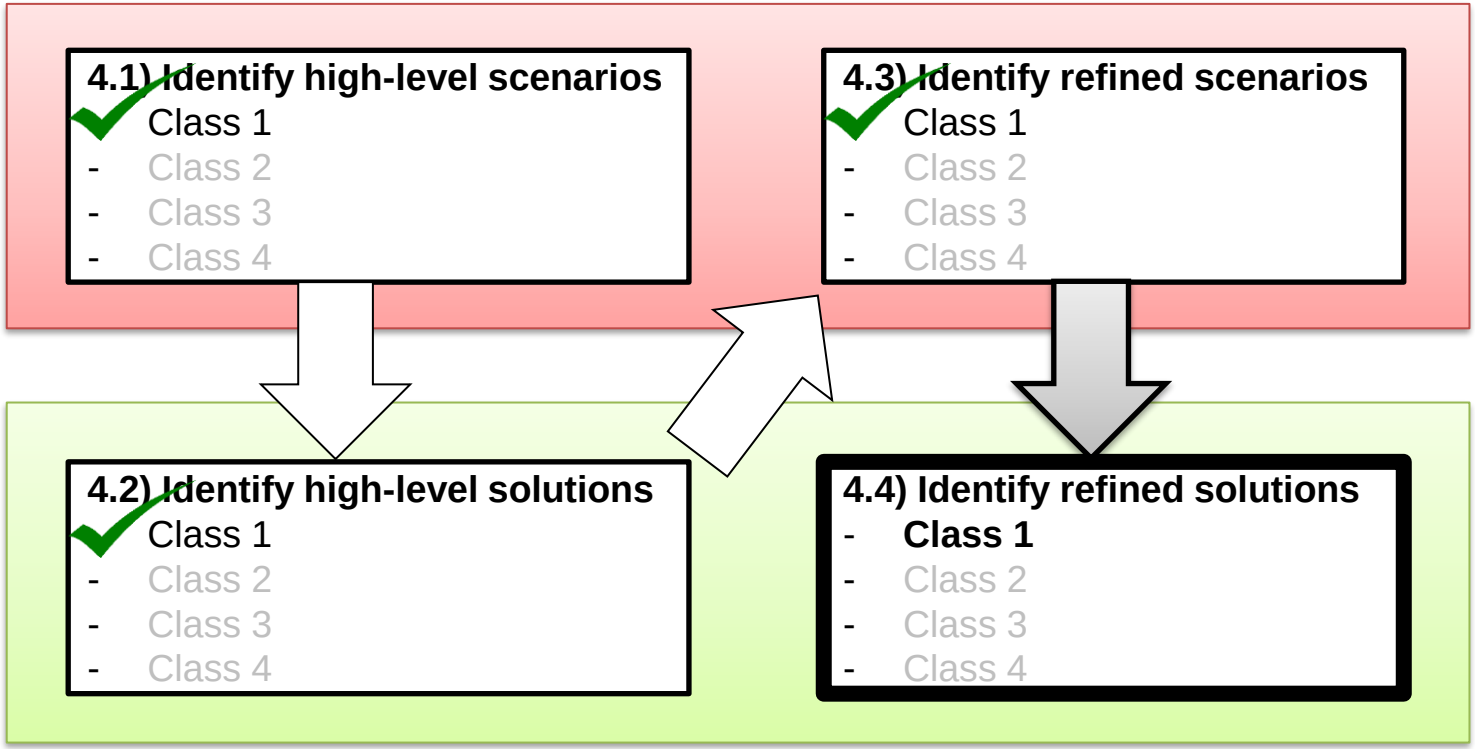
- Although <*Input*> is correct, the feedback from ___ may be incorrect and may cause <*Controller*> to ____.

© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*



Refined Scenario: Update Process Model
- By design, the *Thruster Controller* cannot listen to *Touchdown Input* if a fault is flagged at any time during descent. Will use ___ instead.

4.4) Identify Refined Solutions

What can be done to prevent or mitigate this scenario?

© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*

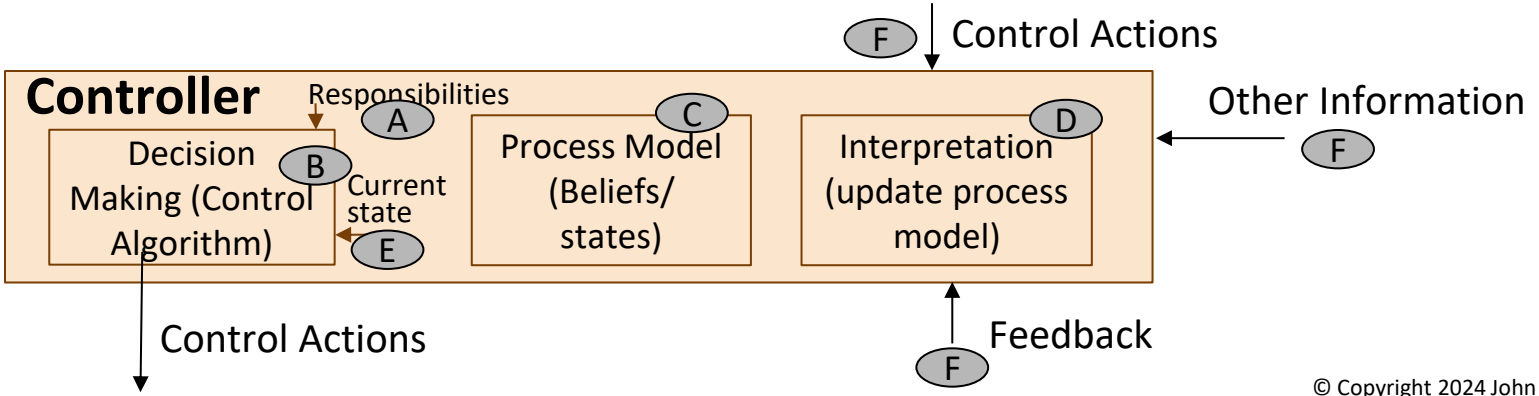


Refined Scenario: Update Process Model

- By design, the *Thruster Controller* cannot listen to *Touchdown Input* if a fault is flagged at any time during descent. Will use ___ instead.

4.4) Identify Refined Solutions

What can be done to prevent or mitigate this scenario? Consider structural and behavioral mitigations.



© Copyright 2024 John Thomas

Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*



Refined Scenario: Update Process Model

- By design, the *Thruster Controller* cannot listen to *Touchdown Input* if a fault is flagged at any time during descent. Will use ___ instead.

4.4) Identify Refined Solutions

What can be done to prevent or mitigate this scenario? Consider structural and behavioral mitigations.

- Examples of controller behavior mitigations:
  - Responsibilities: Provide a way to clear/reset/recover from the fault?
  - Update Process Model: Ignore faults when providing Disable Thruster Cmd?
  - Control Algorithm: Instead of ignoring all Touchdown Inputs, drop to 2oo3 logic?
  - Other Inputs: Use Radar Altimeter as a back up if _____?
- Other solutions / mitigations?

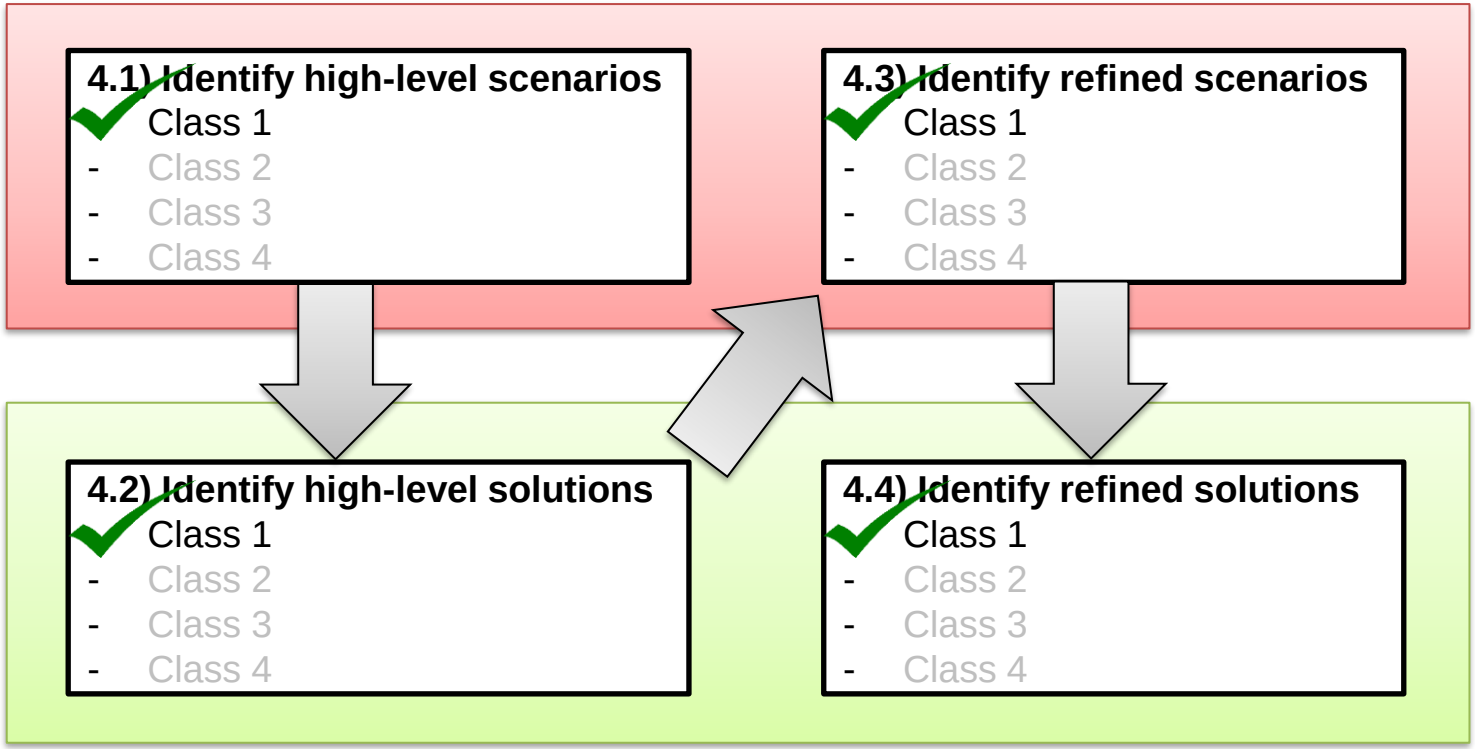
Discussion: Some solutions may be obvious. Others may require revisiting the concept, control structure, or other decisions.

© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

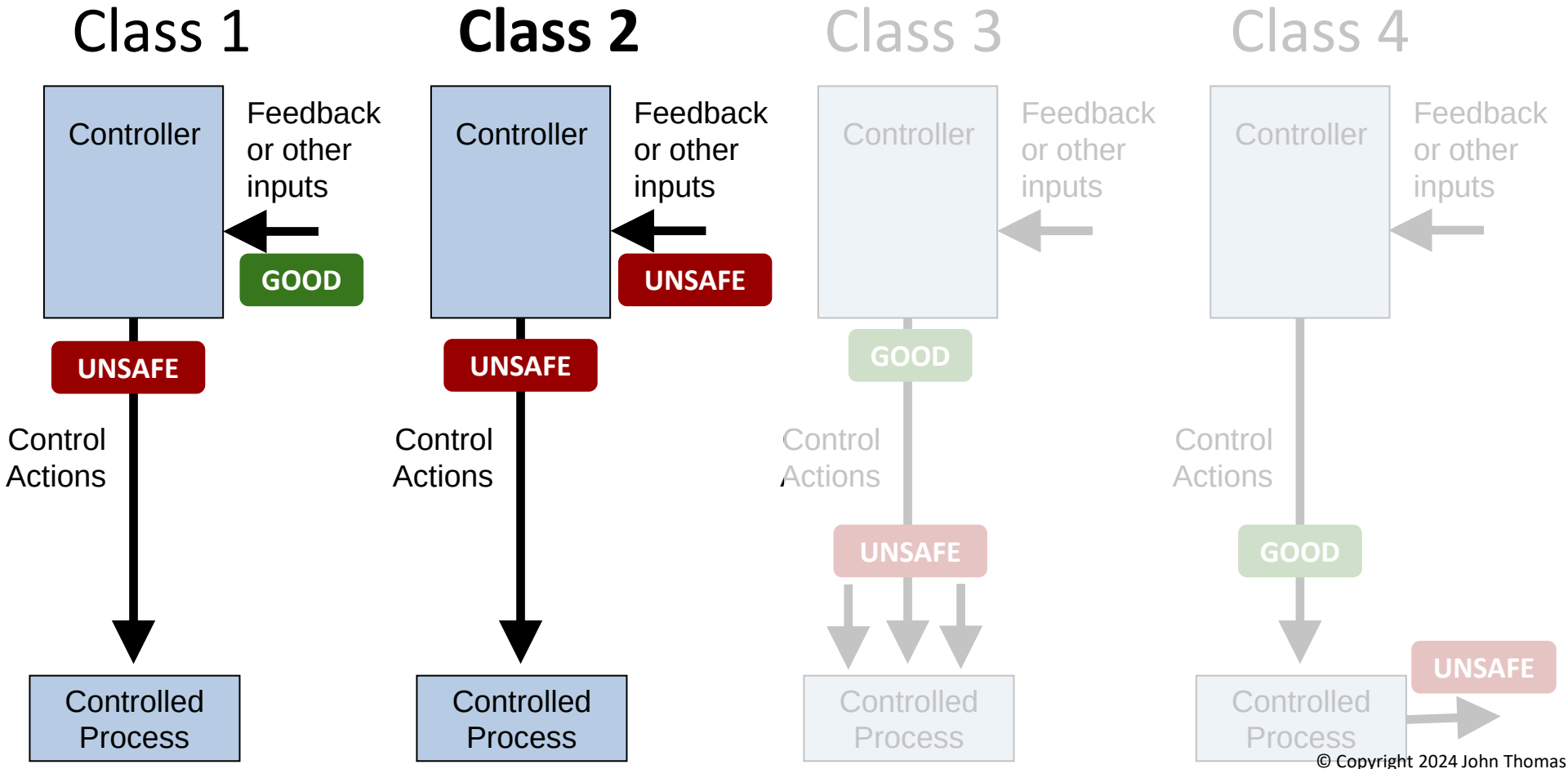
Solution Space: What must be done to prevent losses?



Done! Let's Do Scenario Class 2!

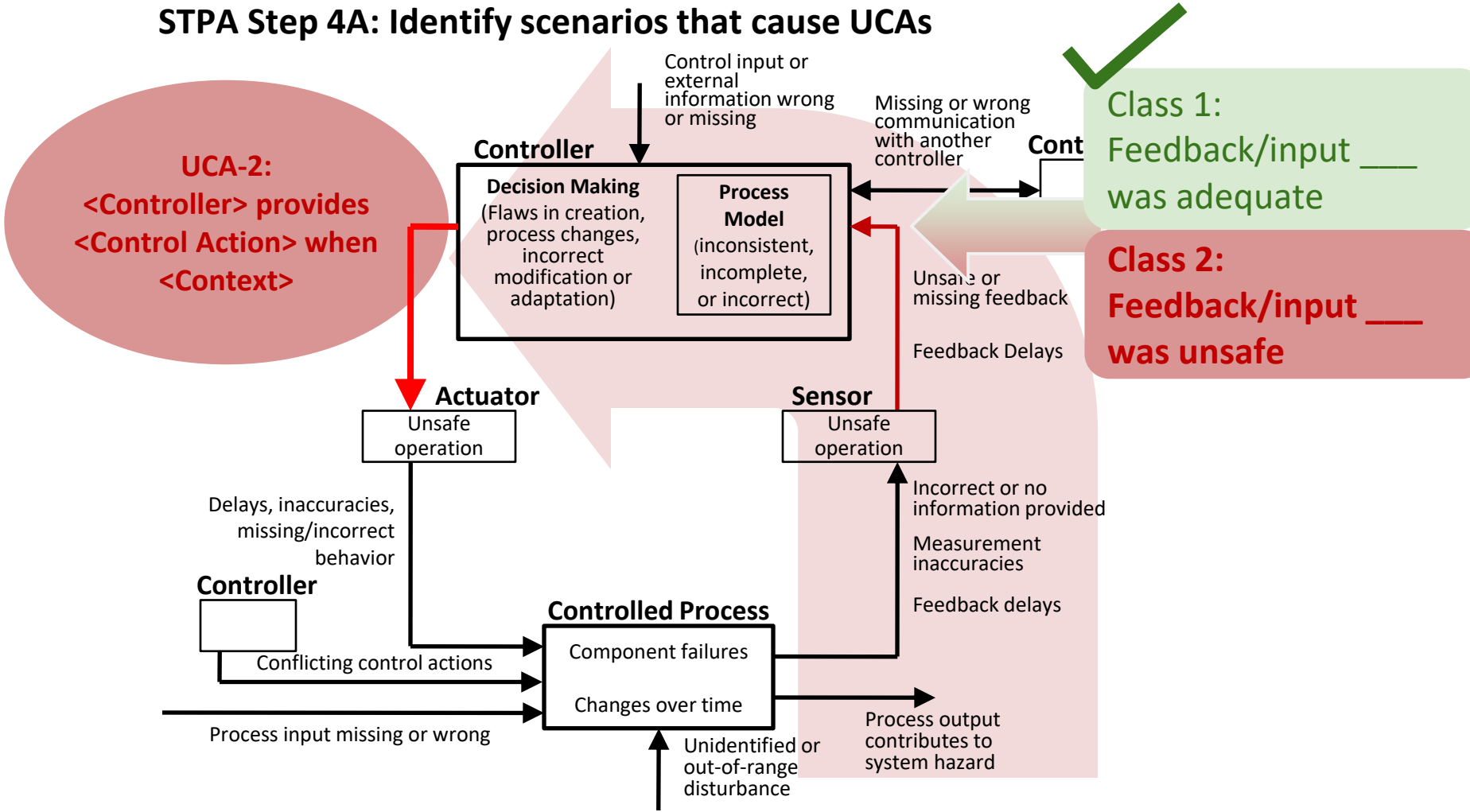
© Copyright 2024 John Thomas

Four Classes of Formal Scenarios



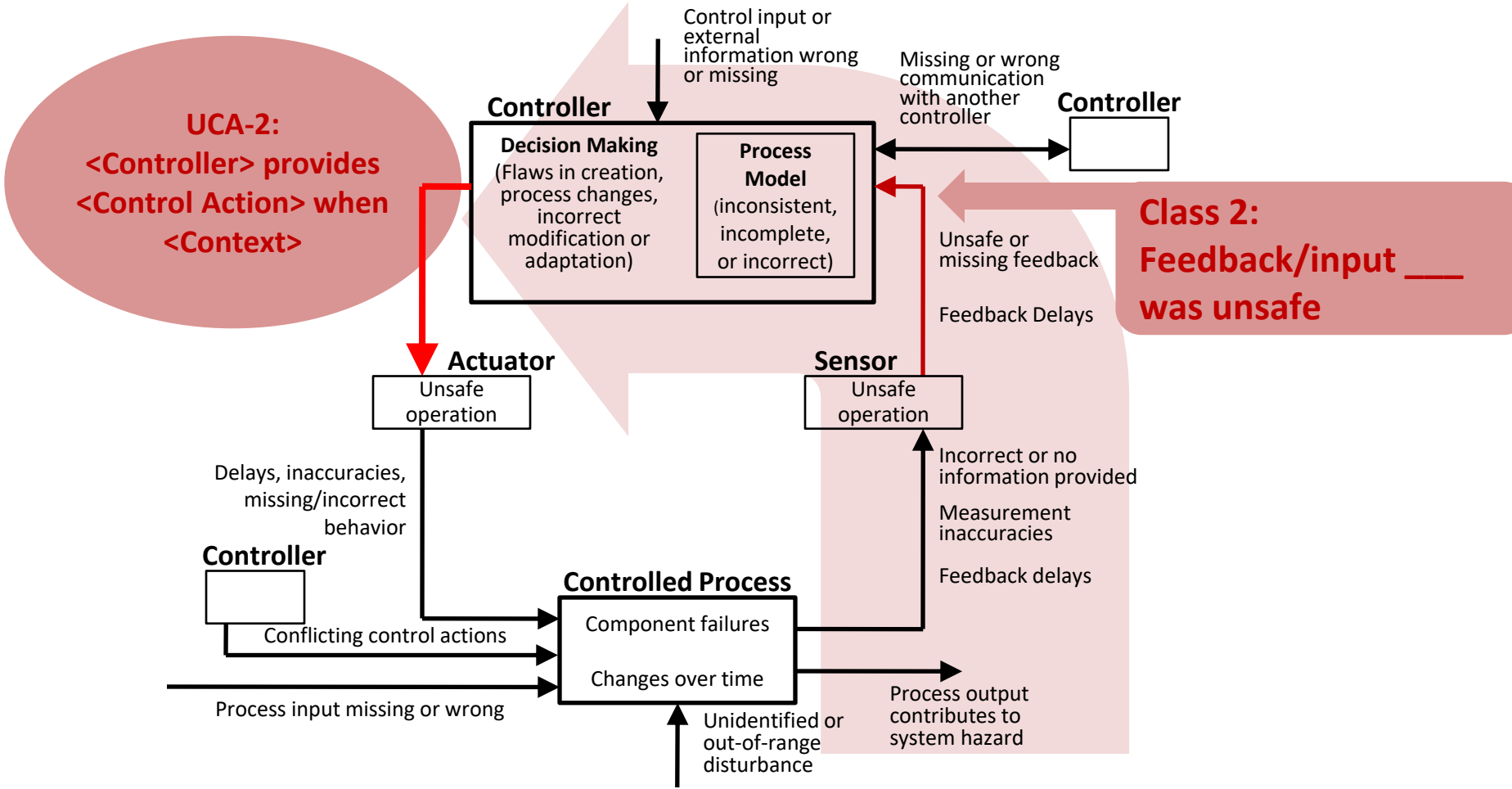
© Copyright 2024 John Thomas

STPA Step 4A: Identify scenarios that cause UCAs



© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype



© Copyright 2024 John Thomas

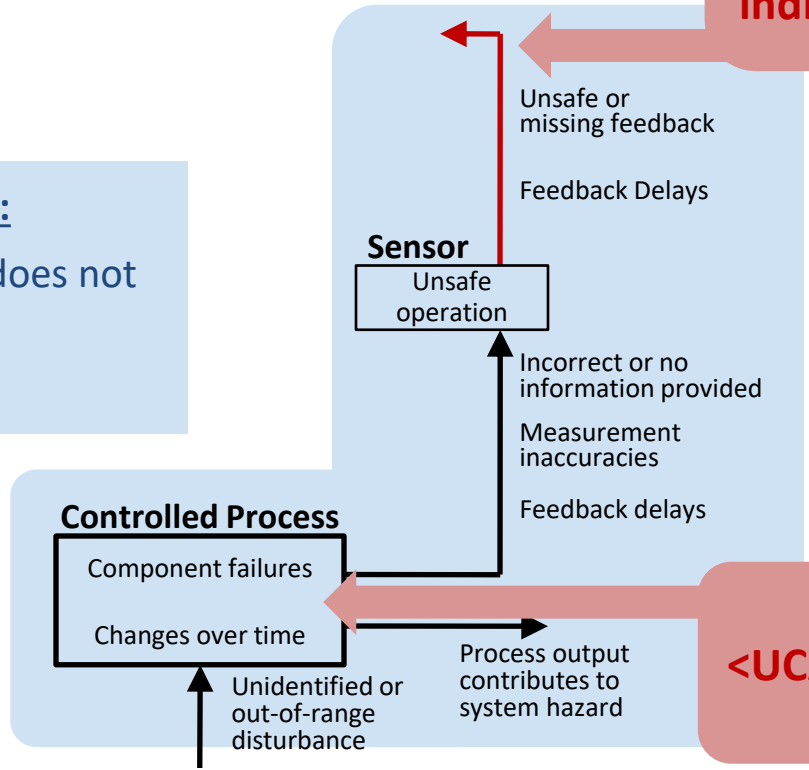
STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller> provides <Control Action> when <Context>

Class 2:
Feedback/Input does not adequately indicate ____

Class 2 Scenario Archetype:

- Feedback/Input to <Controller> does not adequately indicate <Context>
- <Context> is true



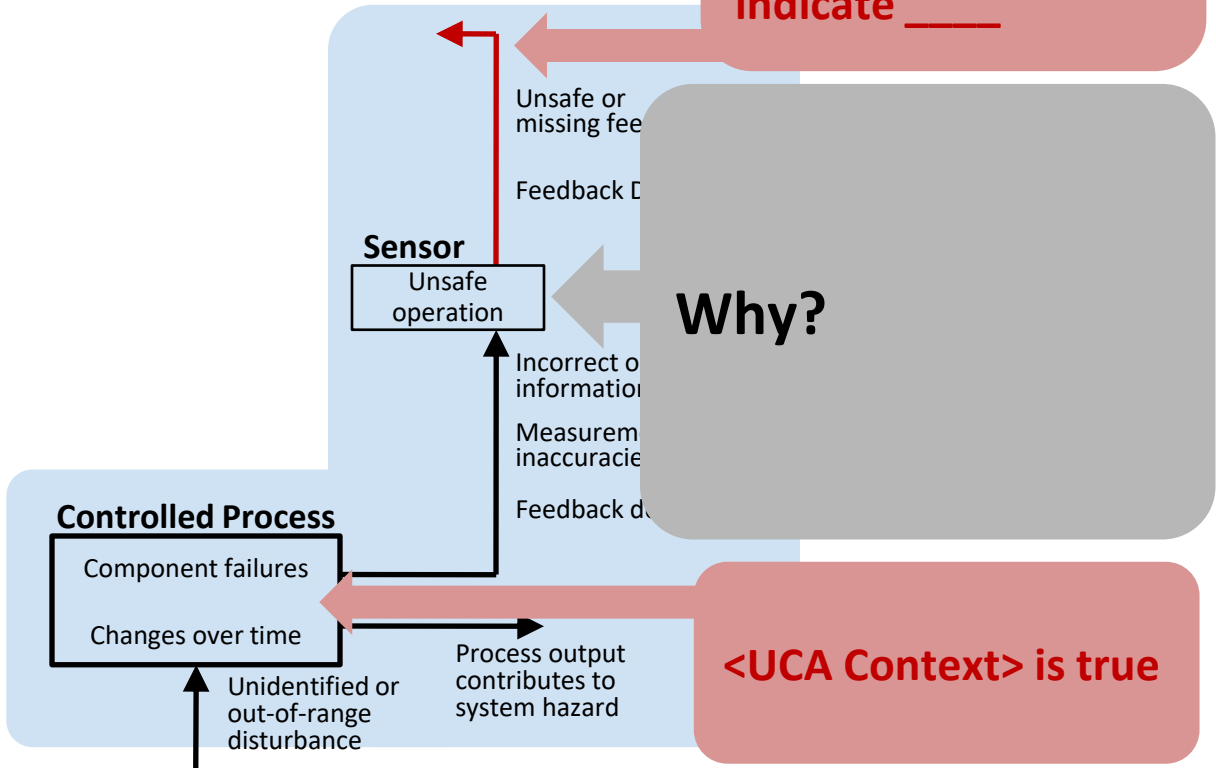
<UCA Context> is true

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller> provides <Control Action> when <Context>

Class 2:
Feedback/Input does not adequately indicate ____



Why?

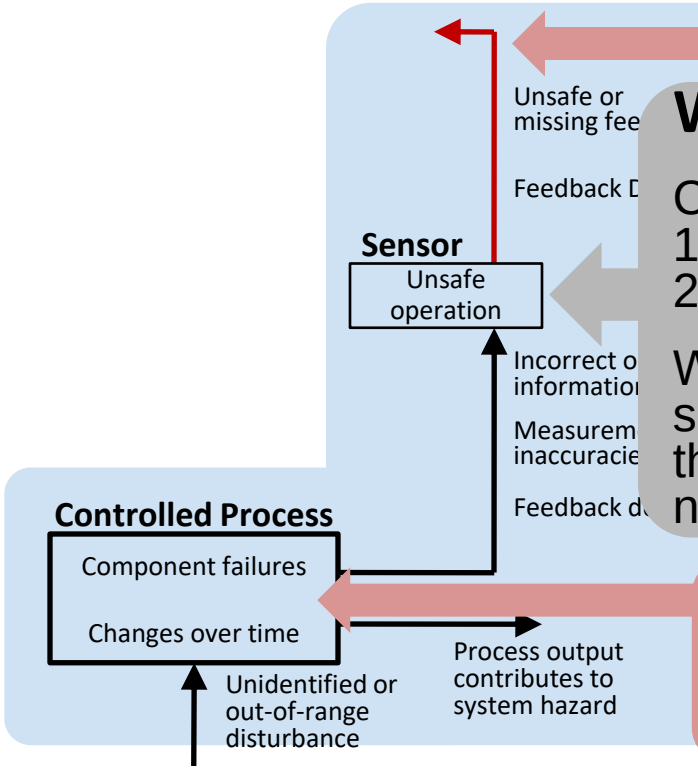
<UCA Context> is true

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller> provides <Control Action> when <Context>

Class 2:
Feedback/Input does not adequately indicate ____



Why?

Consider:
1) Failure causes
2) No failures

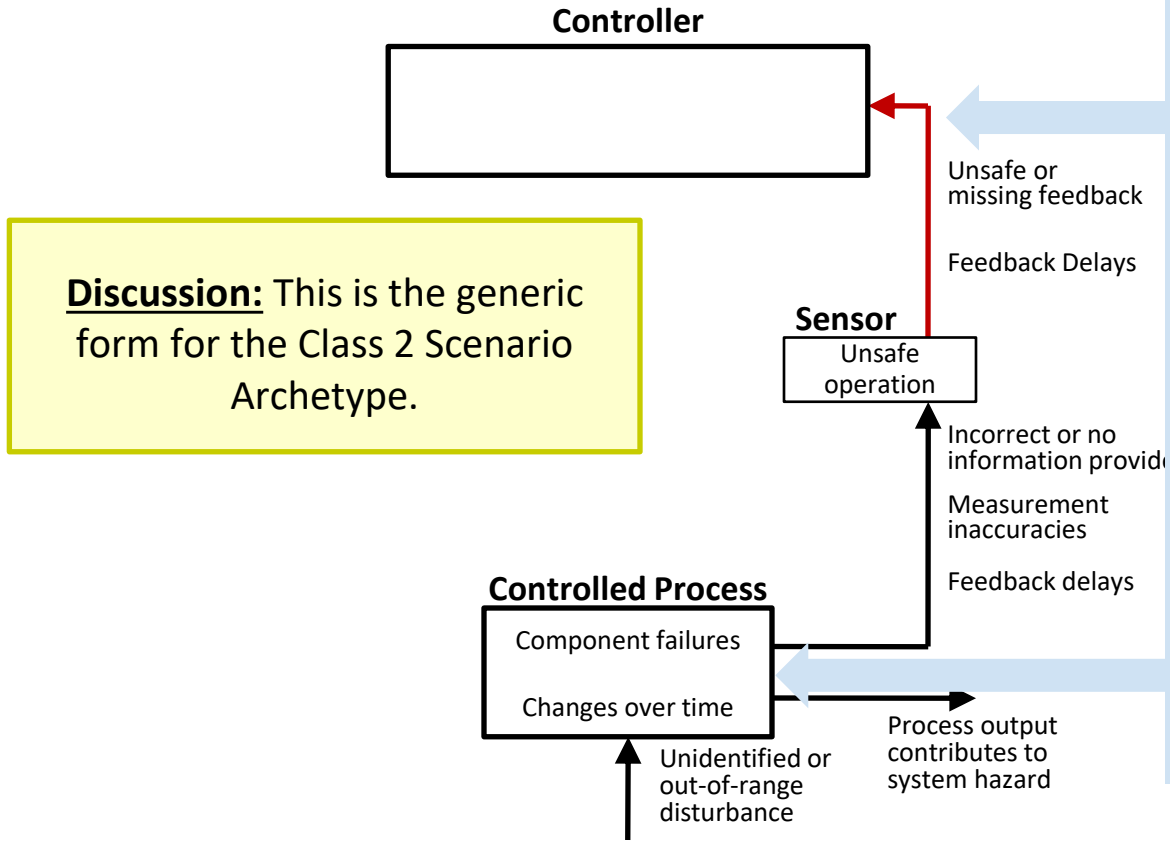
We'll refine the scenario to identify these causes in the next step.

<UCA Context> is true

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>



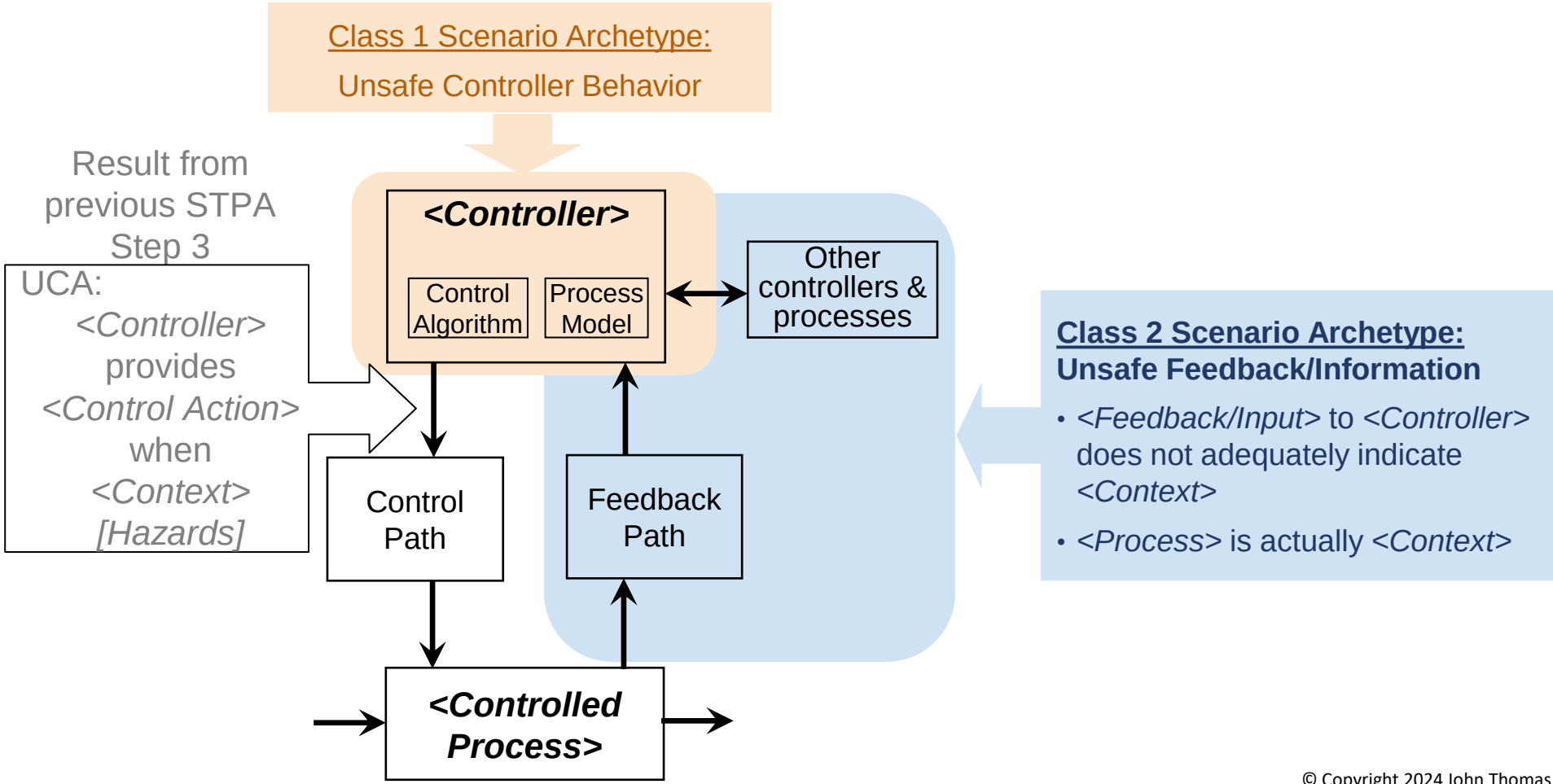
Discussion: This is the generic form for the Class 2 Scenario Archetype.

Class 2 Scenario Archetype
Unsafe Feedback

Output: <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>

Input: <*Process*> is actually <*Context*>

© Copyright 2024 John Thomas

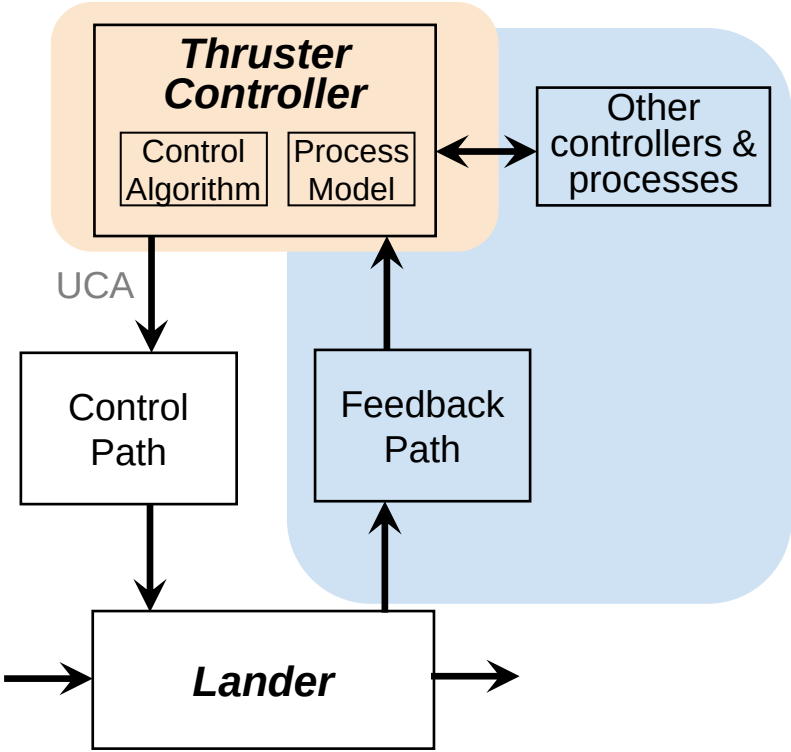


© Copyright 2024 John Thomas



Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*



Class 2 Scenario Archetype:

Unsafe Feedback/Information

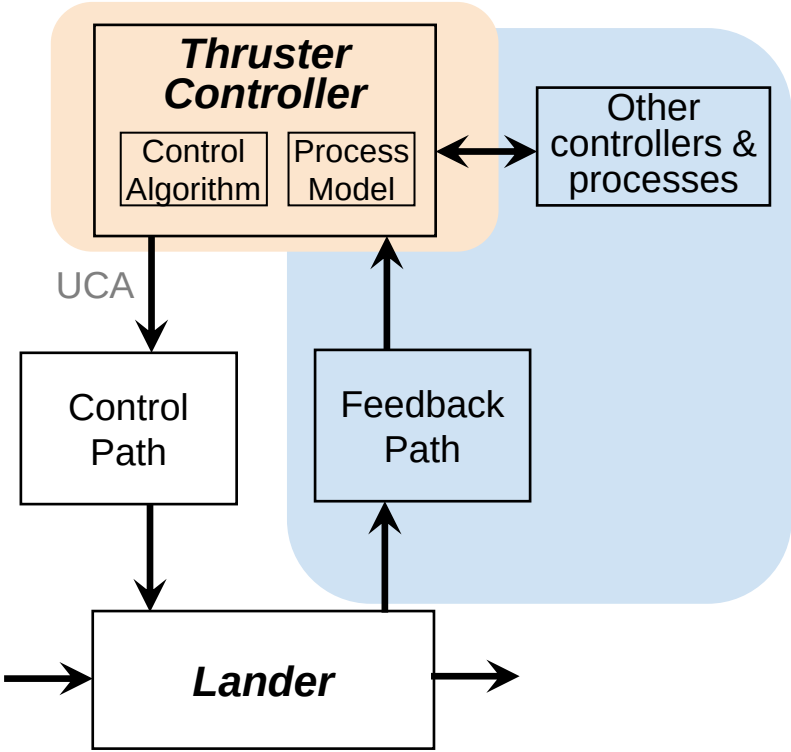
- *<Feedback/Input>* to *<Controller>* does not adequately indicate *<Context>*
- *<Process>* is actually *<Context>*

© Copyright 2024 John Thomas



Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*



Class 2 Scenario Archetype:

Unsafe Feedback/Information

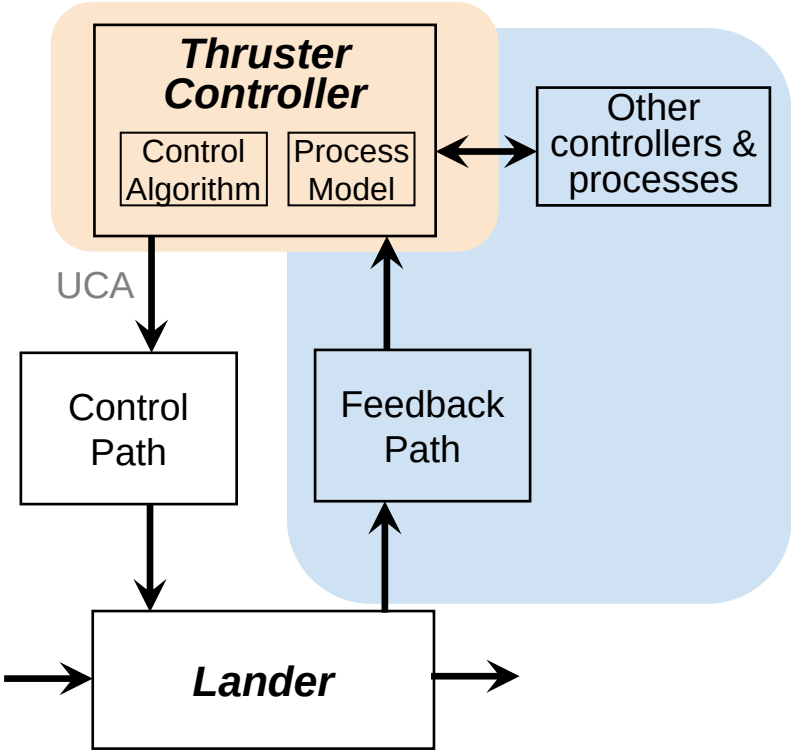
- *Touchdown feedback* does not indicate *lander in air*
- *<Process>* is actually *<Context>*

© Copyright 2024 John Thomas



Result from previous STPA Step 3

UCA:		*UCA-2:*
<Controller>	→	*Thruster Controller*
provides	→	*provides*
<Control Action>	→	*Disable-Thruster Cmd*
when	→	*when*
<Context>	→	*lander is in the air*
[Hazards]	→	*[H-1]*



Class 2 Scenario Archetype:

Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*

© Copyright 2024 John Thomas

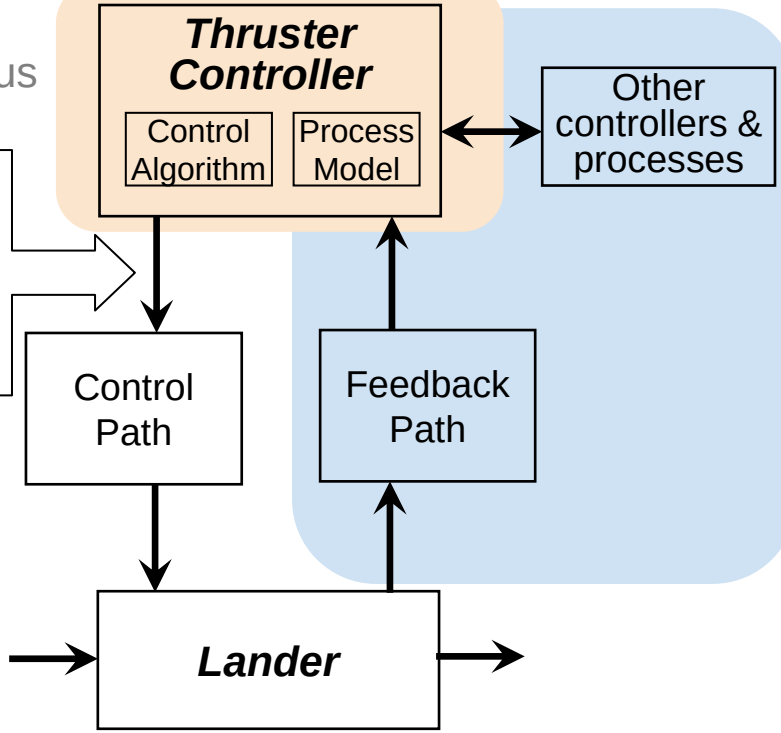


Class 1 Scenario Archetype:
Unsafe Controller Behavior

- UCA: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air*
- *Touchdown Input* to *Thruster Controller* correctly indicates *lander is in the air*

Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]



Class 2 Scenario Archetype:
Unsafe Feedback/Information

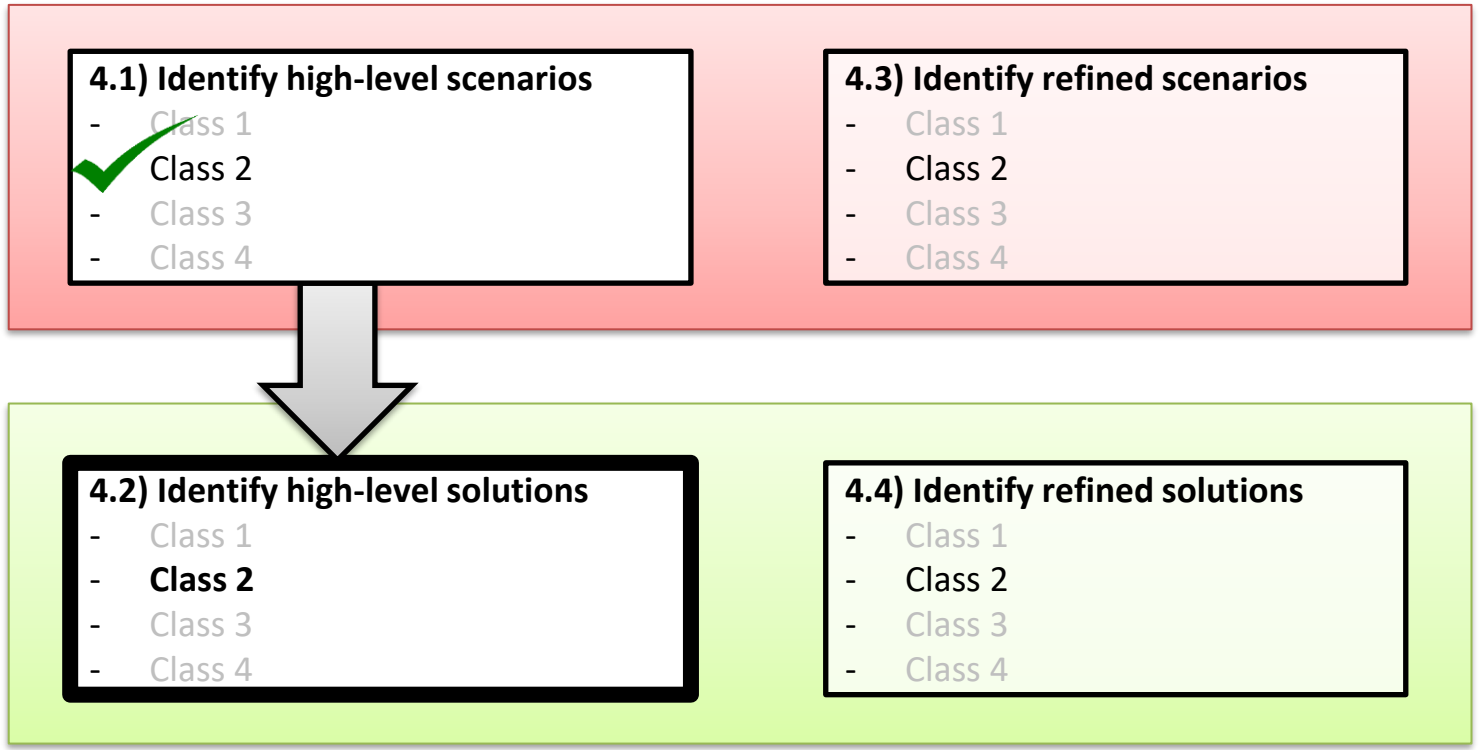
- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*

© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

4.2) Identify High-level Solutions

What is needed from each part of the control structure to prevent the scenario? For example:

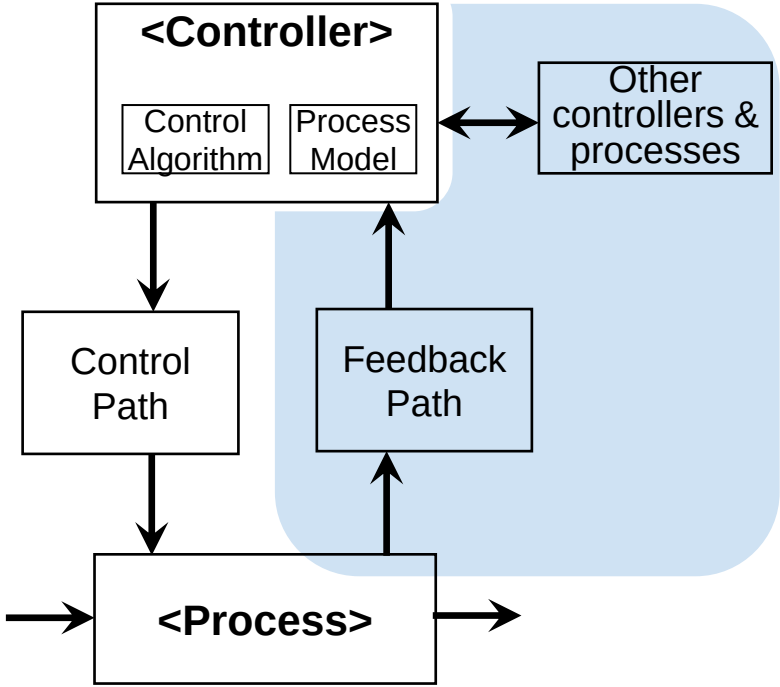
Structural mitigations

- Add/remove control actions?
- Add/remove controllers?
- Add/remove feedback?
- Add/remove processes?

Behavioral mitigations

- <*Controller*> must ___ when ____
- <*Feedback*> must ___ when ___
- <*Controller*> is/is not responsible for _____
- <*Controller*> Control Algorithm must ____
- <*Controller*> Process Model must include ____ (e.g., make controller aware of some other state)
- <*Process*> must ____

Others?



Discussion: Solutions are not necessarily limited to any one part of the control structure, like the feedback path.

© Copyright 2024 John Thomas

Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*

4.2) Identify High-level Solutions

What can be done to prevent or mitigate this scenario?

- Add Feedback:
  - Use alternate indication, such as the radar altimeter ?
- Add Controller Responsibility:
  - Software must filter out touchdown indications <TBD ms ?
- Other solutions / mitigations?

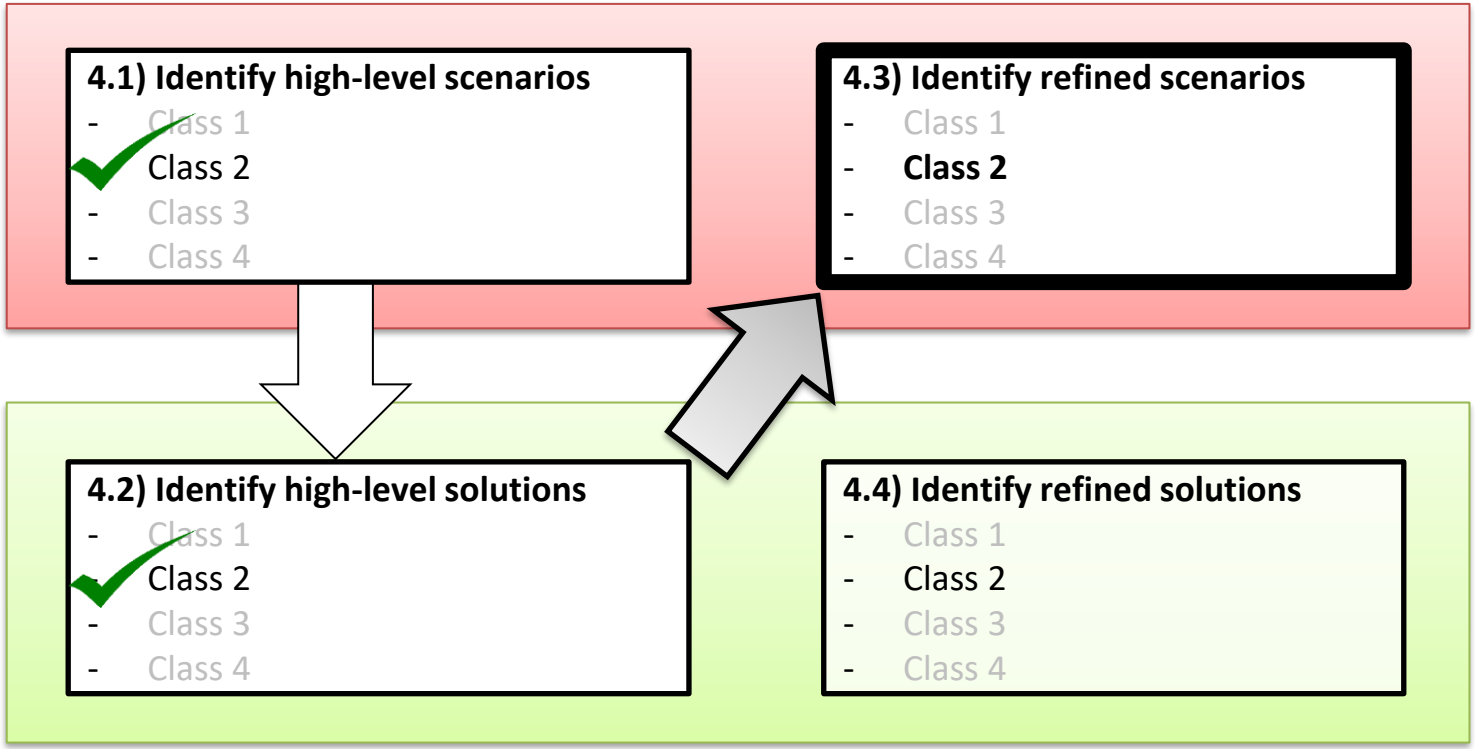


© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Ask: What can cause this?

Consider:
1) Failure causes
2) Causes without failure

Class 3 Scenario Archetype

Class 4 Scenario Archetype

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Refinement Option 1: Ask SMEs to help identify causes.

Class 3 Scenario Archetype

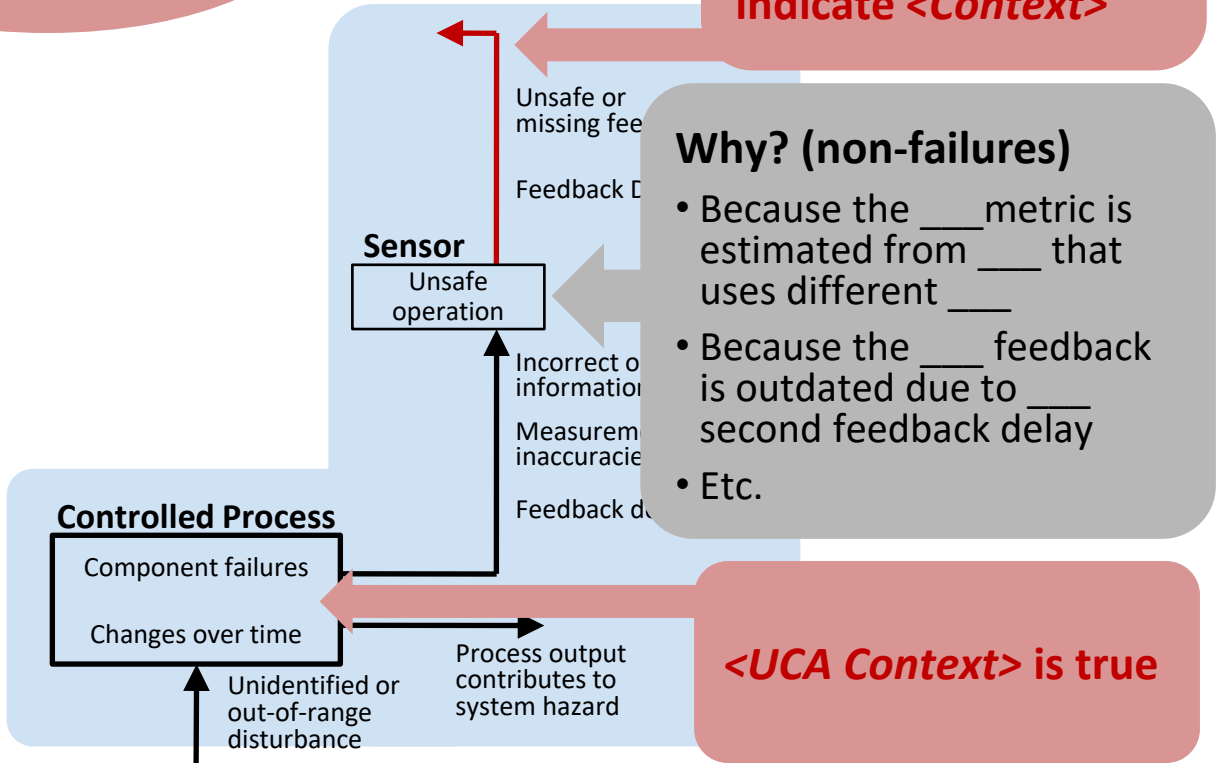
Class 4 Scenario Archetype

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller>* provides *<Control Action>* when *<Context>

Class 2:
<Feedback/Input>* does not adequately indicate *<Context>



Why? (non-failures)

- Because the ___metric is estimated from ___ that uses different ___
- Because the ___ feedback is outdated due to ___ second feedback delay
- Etc.

***<UCA Context>* is true**

Example SMEs responses from a real project.

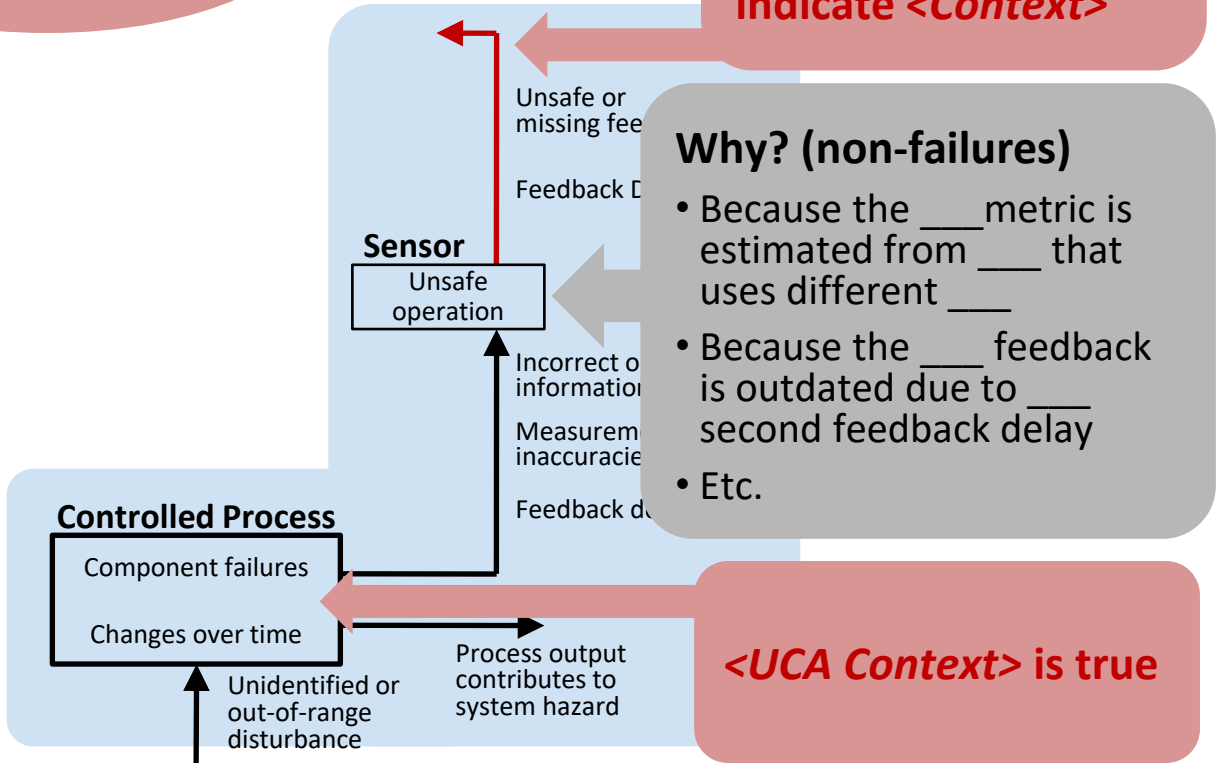
Time required: under 15 minutes

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller>* provides *<Control Action>* when *<Context>

Class 2:
<Feedback/Input>* does not adequately indicate *<Context>



Why? (non-failures)

- Because the ___metric is estimated from ___ that uses different ___
- Because the ___ feedback is outdated due to ___ second feedback delay
- Etc.

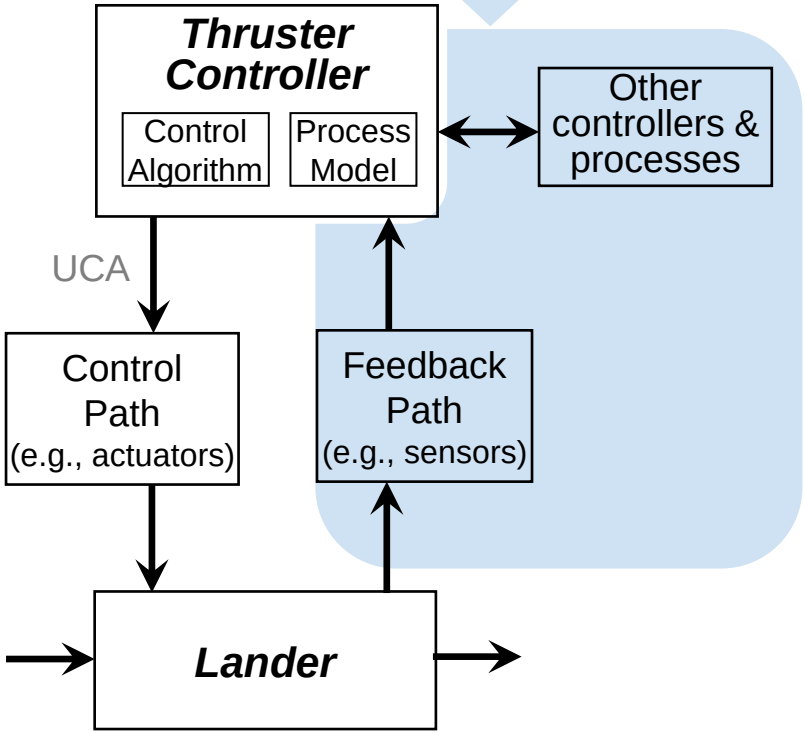
***<UCA Context>* is true**

<u>Discussion:</u> The "Why" answers may come from SMEs, not necessarily the STPA practitioner. You may not be an expert in the system. The point is for you to use this framework to ask the right questions for the team or SMEs to answer.

© Copyright 2024 John Thomas

Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*



Refined Scenarios



What can cause this?

1) Failure causes

2) Causes without failure

To refine non-failure scenarios, it can help to think through the feedback transfer function without a failure.

What does “without failure” mean?

- It means the sensors, etc., technically worked as specified

What does “touchdown sensor worked” mean?

- It means the sensor output was as specified for the sensor input

We know the feedback to controller (sensor output) was [NOT IN AIR].

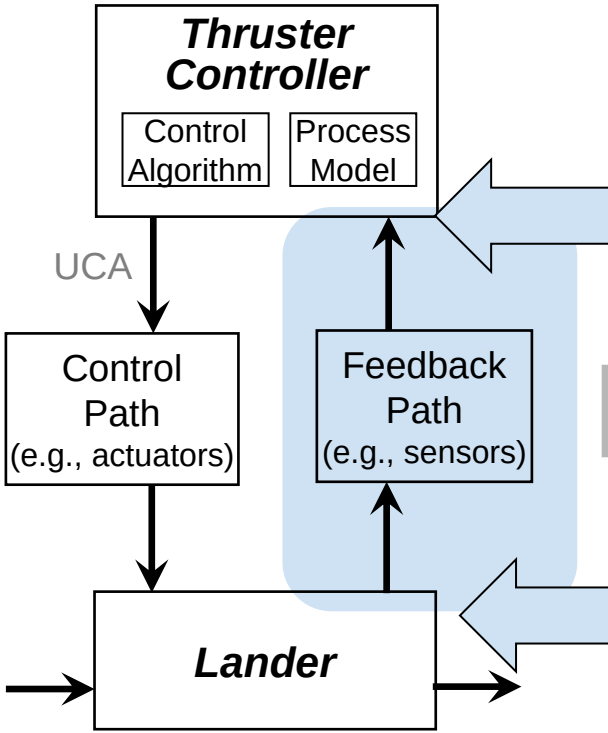
What must be the input to the feedback path if there is technically no feedback failure?

- If output = [NOT IN AIR], then input = [VIBRATION > X]

© Copyright 2024 John Thomas

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*

- *Lander* is actually *in the air*

Refined Scenarios

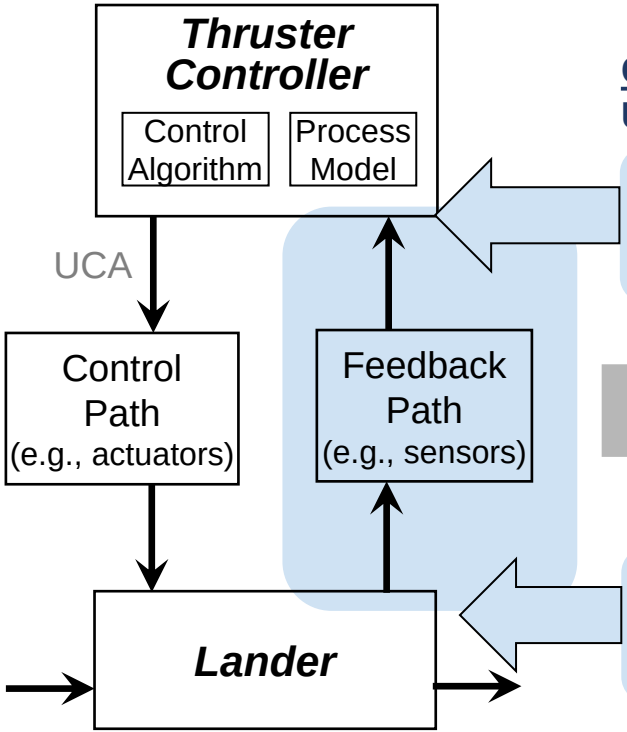
What can cause this?

- ***Without failure:***
  - *Sensor output (from scenario) = TOUCHDOWN TRUE*
  - *Sensor input (if not failed) = VIBRATION > X*

© Copyright 2024 John Thomas

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*

- *Lander* is actually *in the air*

Refined Scenarios

What can cause this?

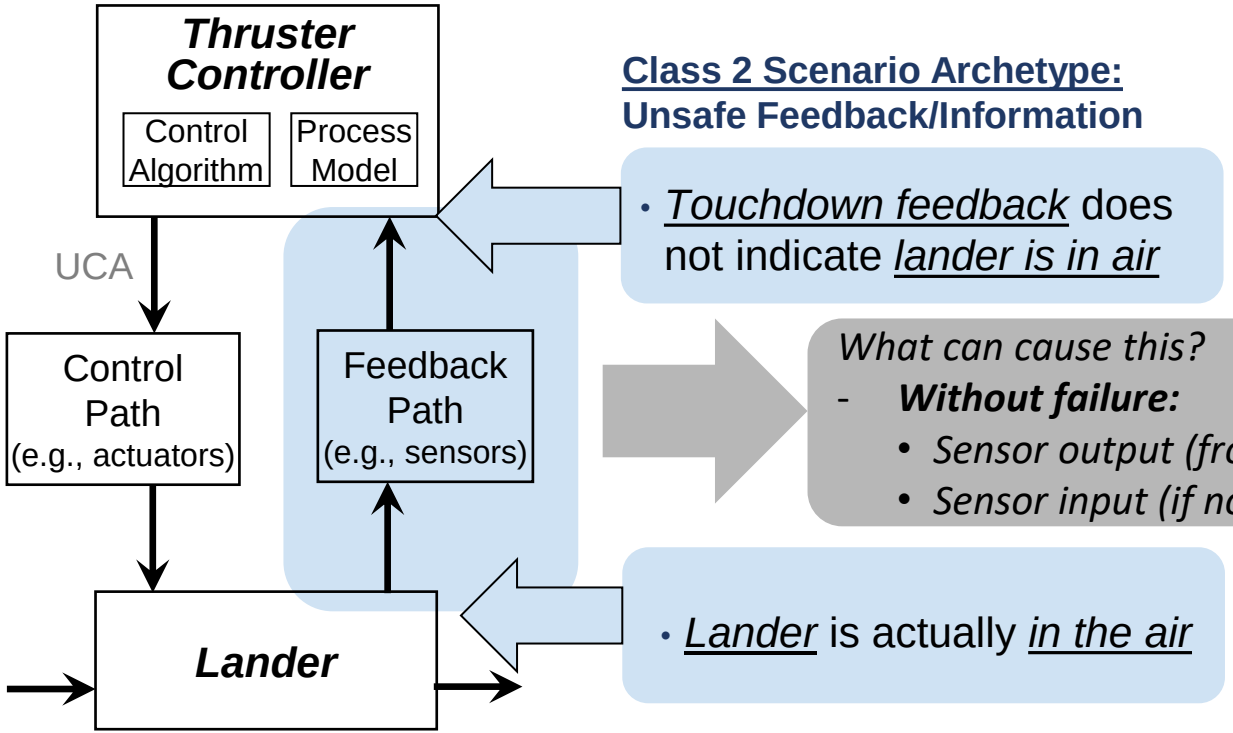
- ***Without failure:***
  - *Sensor output (from scenario) = TOUCHDOWN TRUE*
  - *Sensor input (if not failed) = VIBRATION > X*

AHA! If *Lander* is actually *in the air* AND VIBRATION > X, it will cause UCA-2: *Thruster Controller* provides *Disable-Thruster Cmd* when *lander is in the air* [H-1]

© Copyright 2024 John Thomas

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*

- *Lander* is actually *in the air*

Refined Scenarios

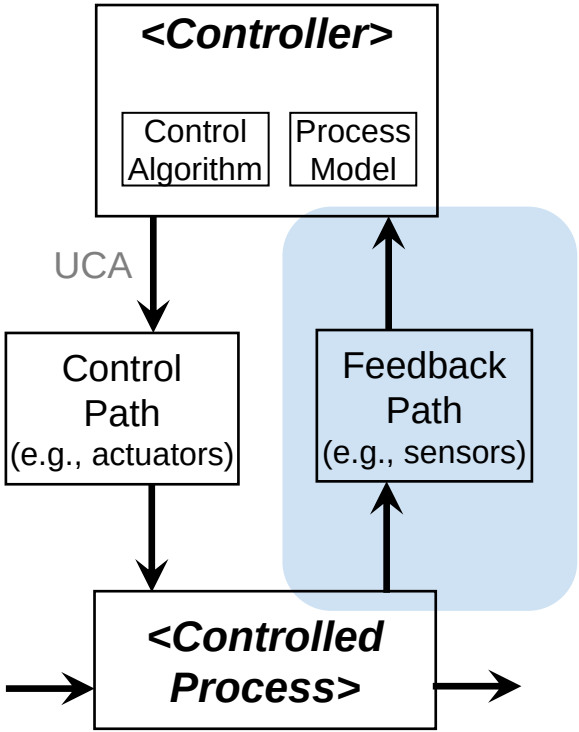
What can cause this?

- ***Without failure:***
  - *Sensor output (from scenario) = TOUCHDOWN TRUE*
  - *Sensor input (if not failed) = VIBRATION > X*

Question to SMEs:
How can we get VIBRATION > X when *Lander* is actually *in the air* ?

Consider:
1) Failure causes
2) Causes without failure

© Copyright 2024 John Thomas



AHA! If [ ] AND [ ], then it will cause UCA-X and *[H-1] !*

Question for SMES:

What could cause [ ] when [ ] ?

Consider:

1) Failure Causes

2) Causes without failure

What solutions can prevent these scenarios?

Discussion: In industry trials, this approach during STPA Step 4 has been extremely effective. It identified many real-world catastrophic non-failure scenarios missed by standard approaches, including:

- Catastrophic flaws in real aircraft control systems (caught by this STPA process after certification but just before operation)
- Safety-related deficiencies in real nuclear power plant control systems overlooked during the safety assessment and licensing
- Flaws in real automotive L2, L3, and L4 systems
- Flaws that would trigger large-scale outages in production online/IT software systems (found by STPA ~1 week prior)

© Copyright 2024 John Thomas

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Refined Scenarios

Refinement Option 1: Ask SMEs to help identify causes.

Is there a more rigorous option?

For some projects, like early concept development or a "lightweight STPA", option 1 may be enough.

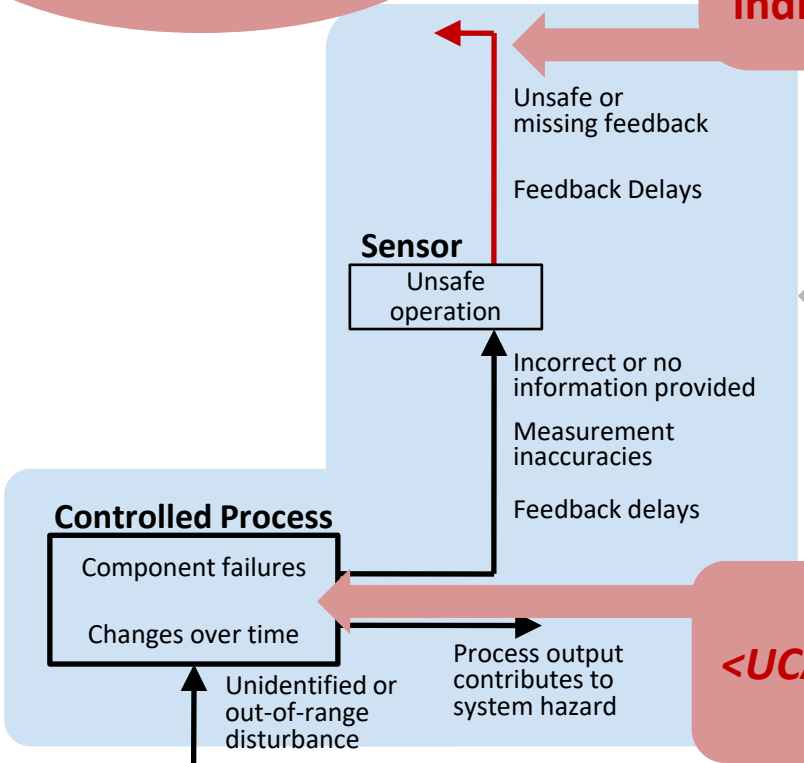
If more rigor needed, a generic feedback model can be used to rigorously refine the scenarios and find causes.

© Copyright 2024 John Thomas

STPA Step 4: Class 2 Scenario Archetype

UCA-2:
<Controller>* provides *<Control Action>* when *<Context>

Class 2:
<Feedback/Input>* does not adequately indicate *<Context>



***<UCA Context>* is true**

Refined Scenarios

Why? Common Causes:

- Feedback/info missing from design/concept
- Feedback/info not provided
- Conflicting feedback/info
- Incorrect feedback/info provided
- Too early or too late (delayed) feedback/info
- Measurement inaccuracies
- Dropouts
- Corruption
- Content incomplete
- Feedback/info provided in a way the controller can't use
- Overloaded or too much feedback/info
- Etc.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

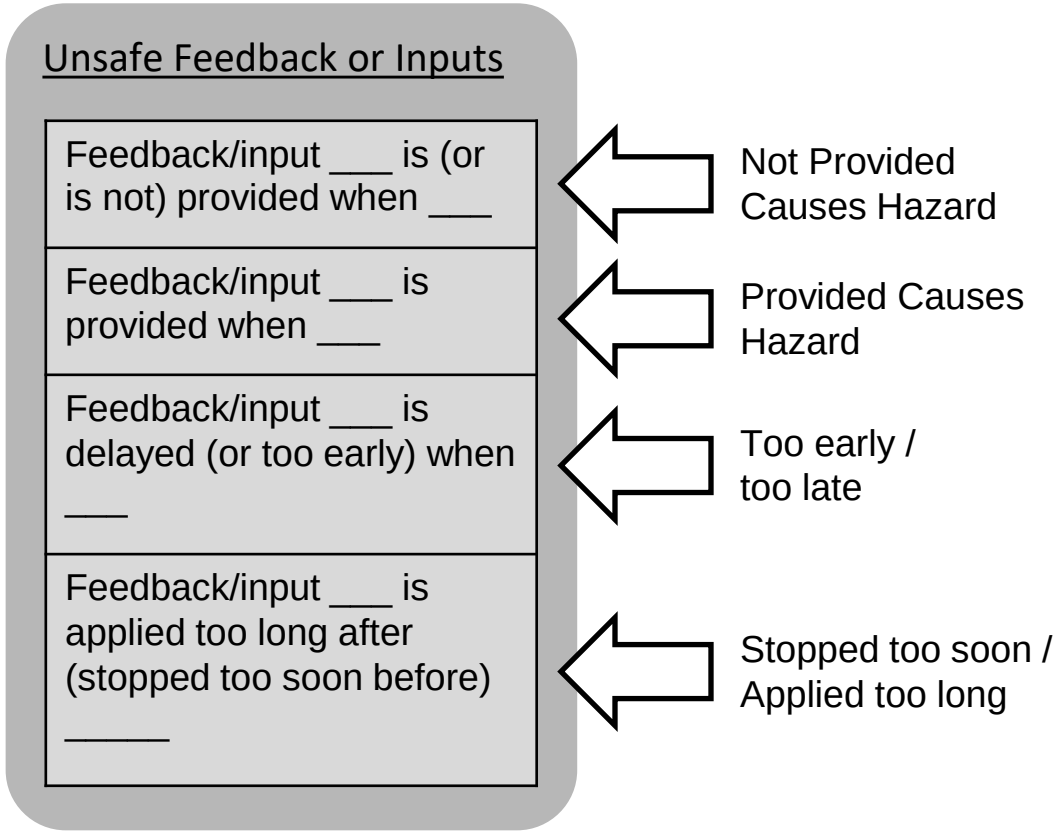
UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype

Class 4 Scenario Archetype



© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Too early or too late info

<Feedback> may be sent too early before ____ has occurred.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype

Class 4 Scenario Archetype

No info provided

<Feedback> is not provided when ____ is initialized, reset, or on power up.

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype

Class 4 Scenario Archetype

Overloaded or too much info

If <Process> is overloaded, then there might not be any indication that <Context>

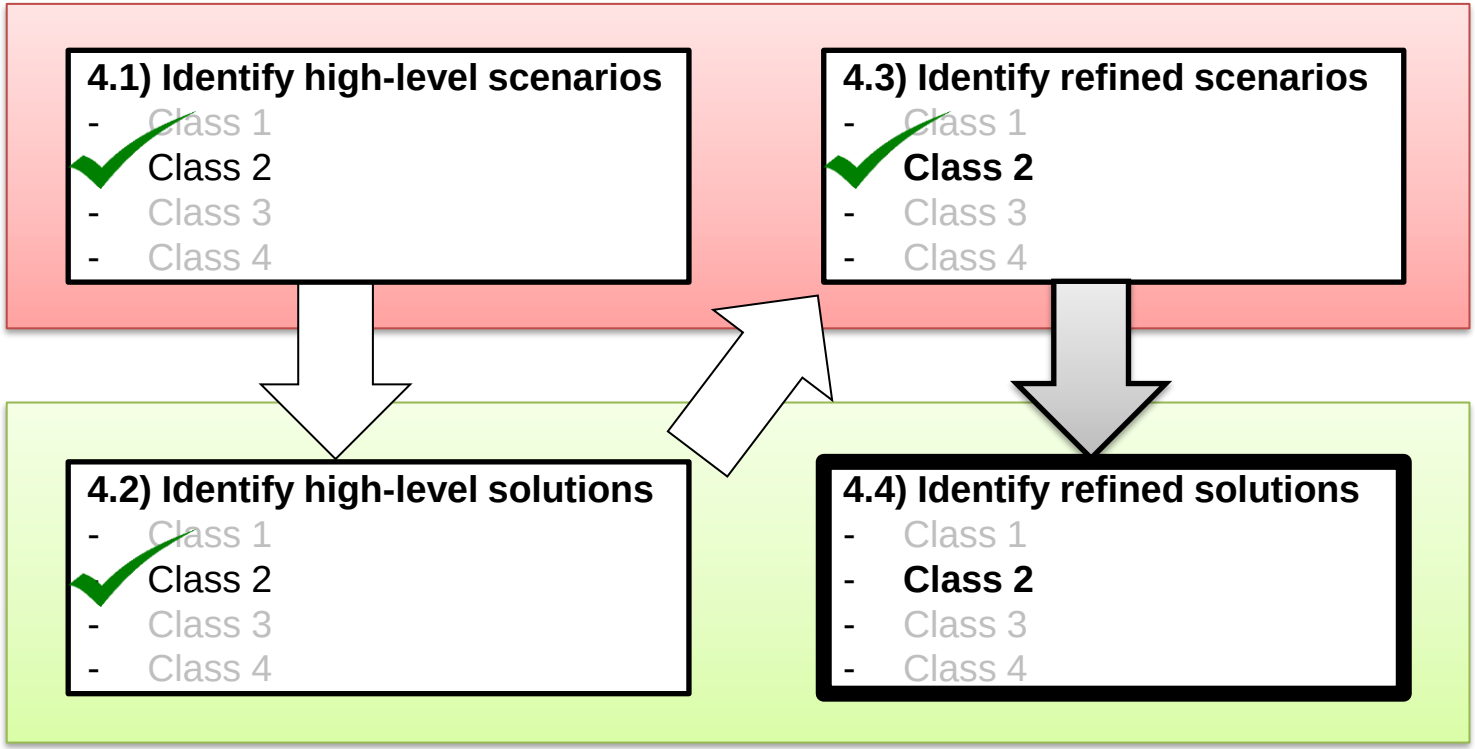
Discussion: These scenarios are used to help us think about mitigations. E.g., <Controller> currently has no feedback to indicate <process> is overloading, so the design doesn't currently have any way to correct it.

© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

Solution Space: What must be done to prevent losses?



© Copyright 2024 John Thomas

Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*



Refined Scenario:

If *Lander* is actually *in the air* and VIBRATION > X, then it will cause UCA-2: Thruster Controller provides Disable-Thruster Cmd *[H-1].*

What can cause footpad vibration in the air (non-failures):

- Leg deployment during descent
- Mechanical forces from parachute deployment
- Etc.

4.4) Identify Refined Solutions

What can be done to prevent or mitigate this scenario?

© Copyright 2024 John Thomas

Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *Touchdown feedback* does not indicate *lander is in air*
- *Lander* is actually *in the air*



Refined Scenario:

If *Lander* is actually *in the air* and VIBRATION > X, then it will cause UCA-2: Thruster Controller provides Disable-Thruster Cmd *[H-1].*

What can cause footpad vibration in the air (non-failures):

- Leg deployment during descent
- Mechanical forces from parachute deployment
- Etc.

4.4) Identify Refined Solutions

What can be done to prevent or mitigate this scenario? Consider structural and behavioral mitigations.

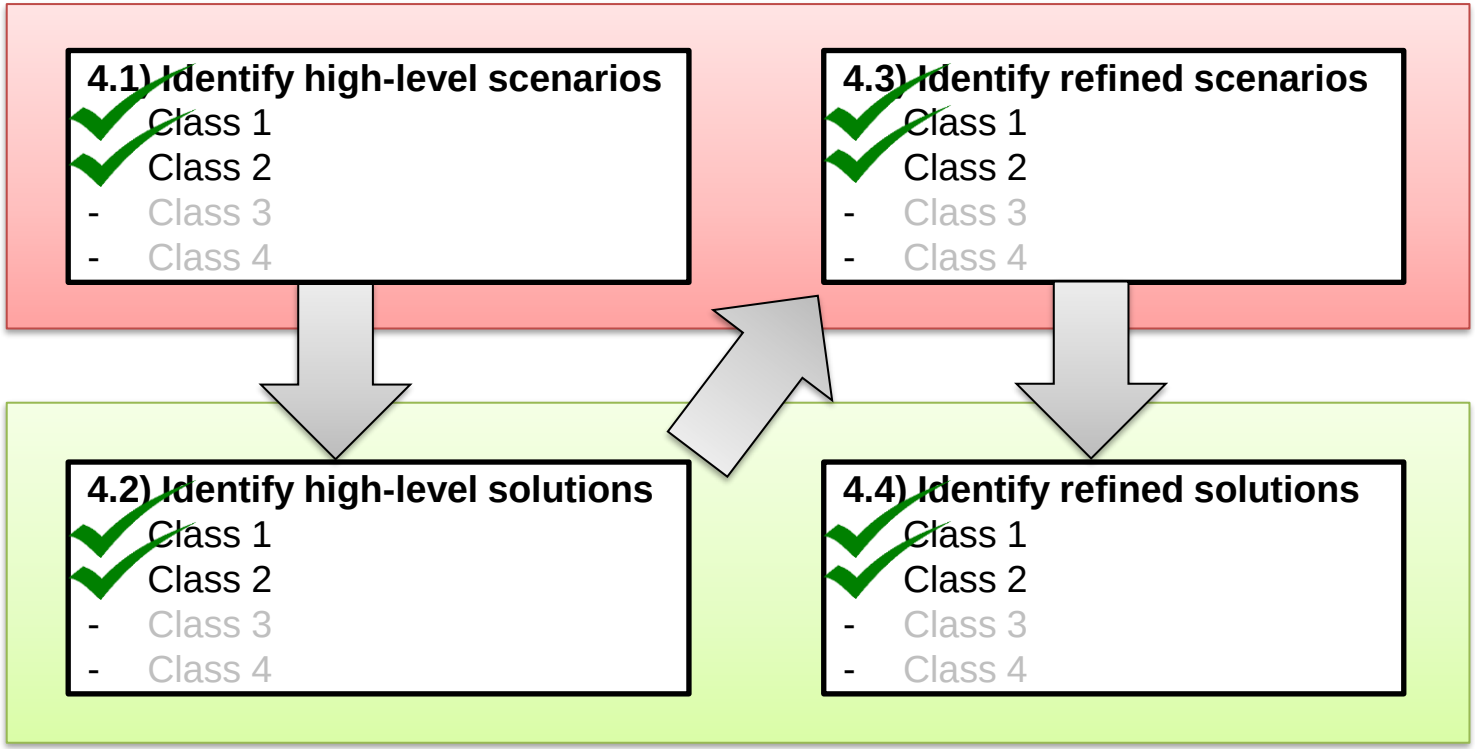
- Controller mitigations:
  - Responsibilities: Thrust Controller must not monitor touchdown indications prior to TBD ?
  - Control Algorithm: Thrust Controller must filter transient touchdown indications <TBD ms ?
- Feedback mitigations:
  - Sensors: Touchdown sensors must rely on a method other than vibration for touchdown indication.
- Others?

© Copyright 2024 John Thomas

STPA: Building Scenarios

Problem Space: What can go wrong?

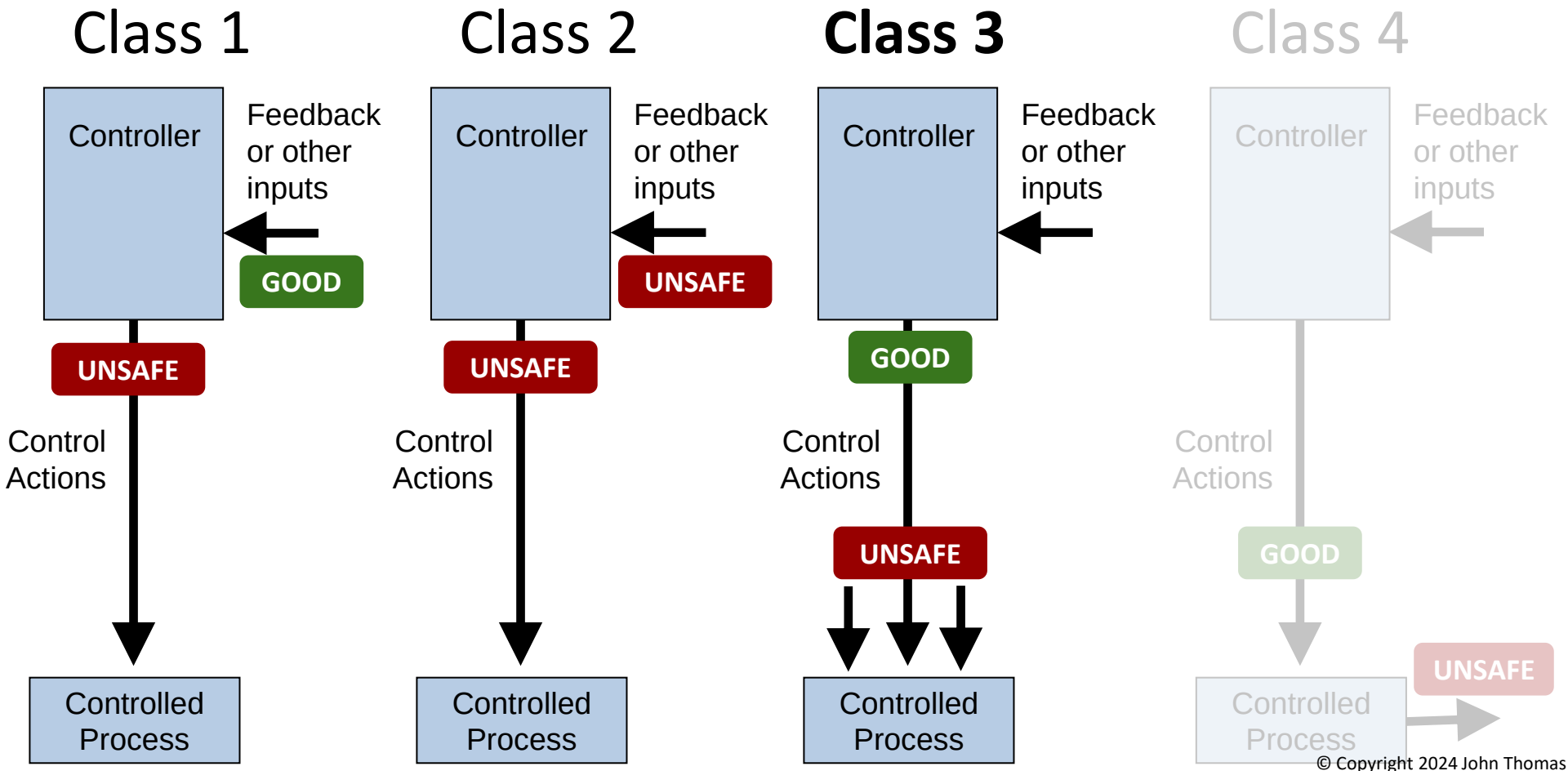
Solution Space: What must be done to prevent losses?



Done! Same pattern for classes 3-4…

© Copyright 2024 John Thomas

Four Classes of Formal Scenarios



© Copyright 2024 John Thomas

Result from previous STPA Step 3

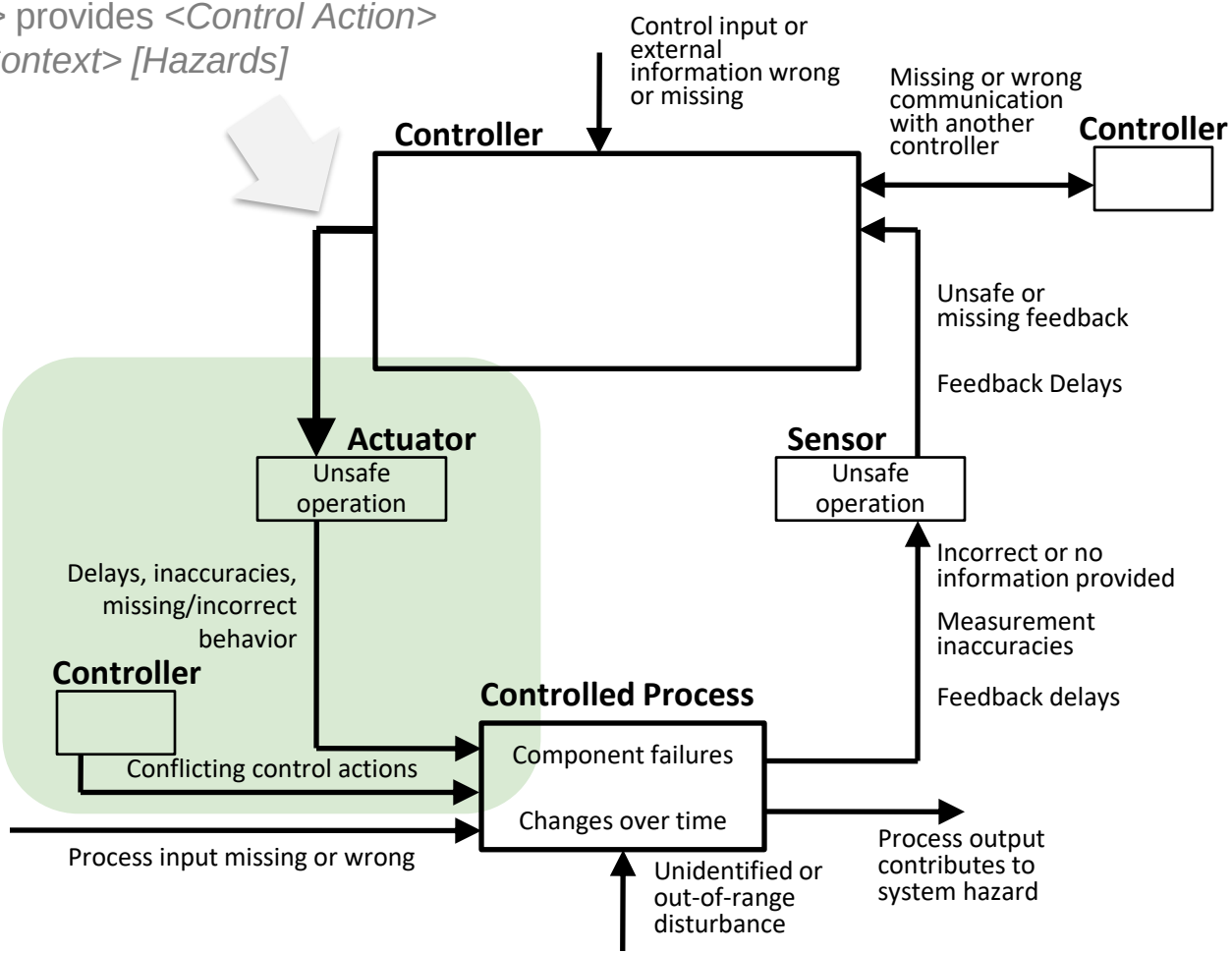
UCA: *<Controller>* provides *<Control Action>*
when *<Context> [Hazards]*

Class 3 Scenario Archetype: Unsafe Control Execution

We need to look at how the UCA can be *emulated* due to interactions involving the process's control path.

Constructing Scenario Archetype 3:

- Suppose the UCA does not happen (invert the UCA). The controller provided a "safe" control action.
- BUT… something happened on the control path making it as if the UCA had occurred



© Copyright 2024 John Thomas

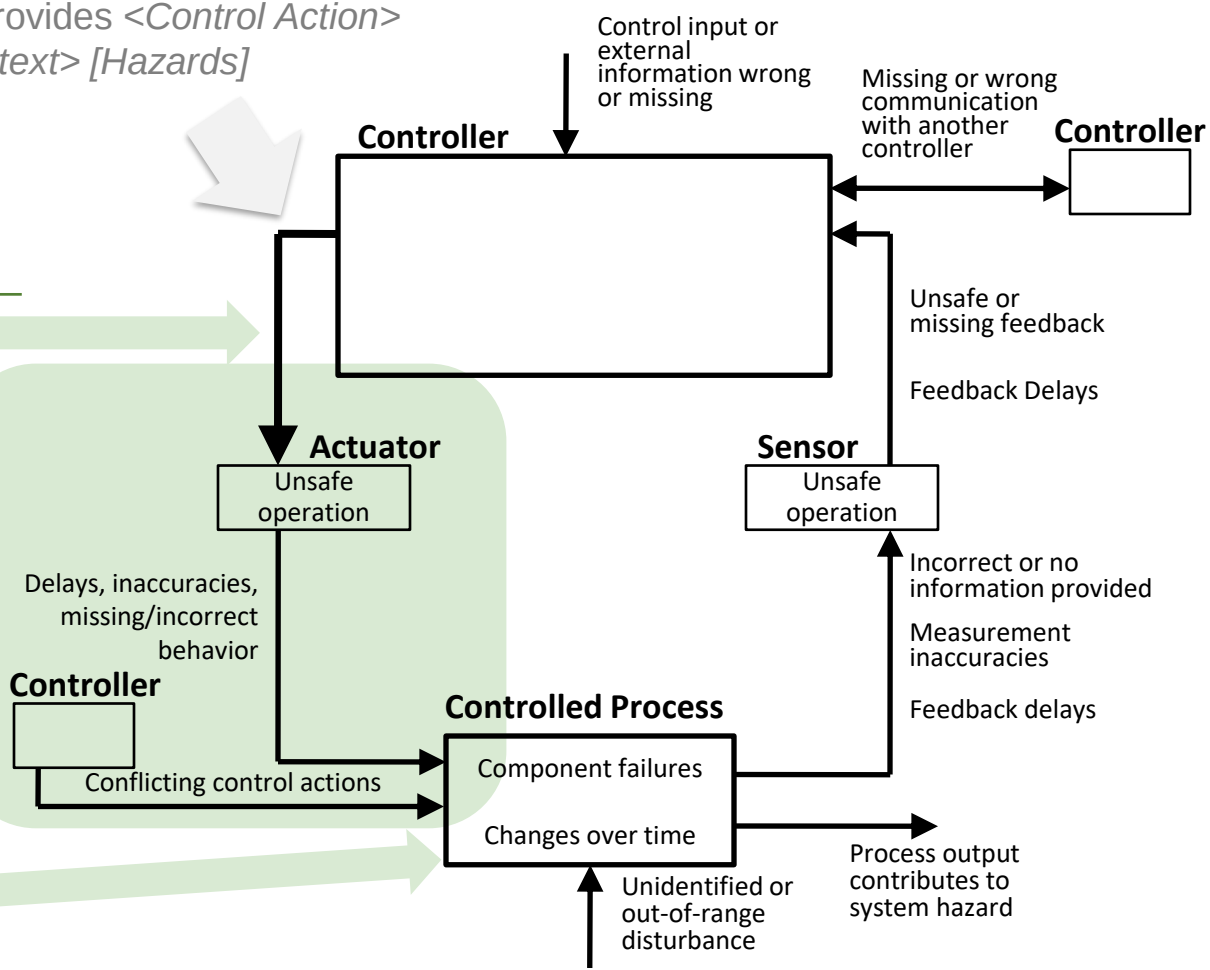
Result from previous STPA Step 3

UCA: *<Controller>* provides *<Control Action>*
when *<Context> [Hazards]*

Class 3 Scenario Archetype:
Unsafe Control Execution

- <Controller> does not provide ____

- <Process> receives ____

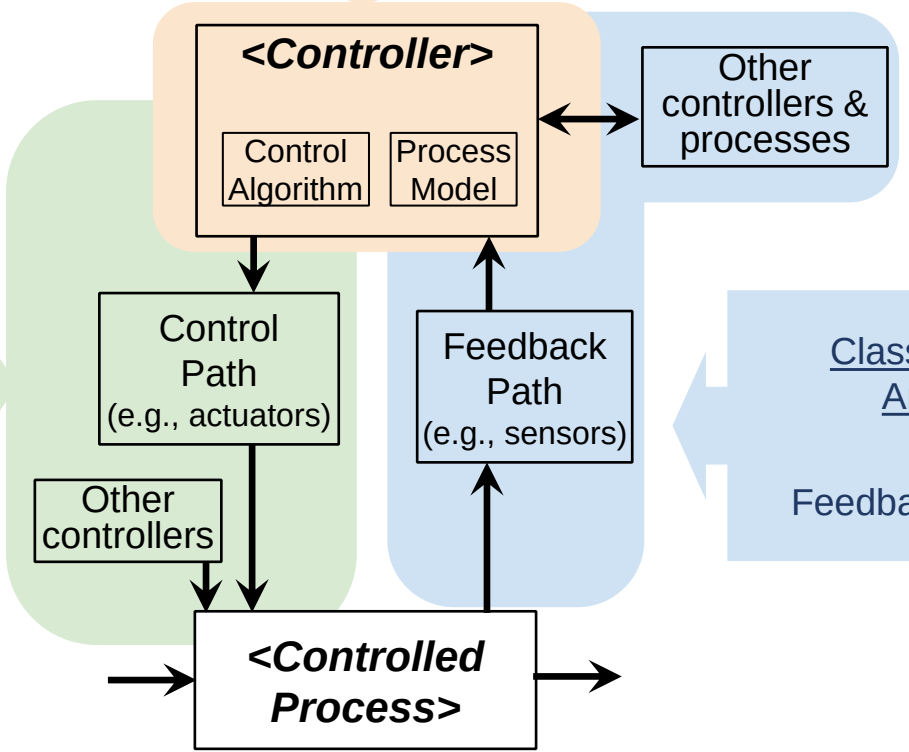


© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA: *<Controller>* provides *<Control Action>* when *<Context>* *[Hazards]*

Class 1 Scenario Archetype:
Unsafe Controller Behavior



Class 3 Scenario Archetype:
Unsafe Control Execution

- *<Controller>* does not provide *<Control Action>* when *<Context>*
- *<Process>* receives *<Control Action>* when *<Context>*

Class 2 Scenario Archetype:
Unsafe Feedback/Information

© Copyright 2024 John Thomas

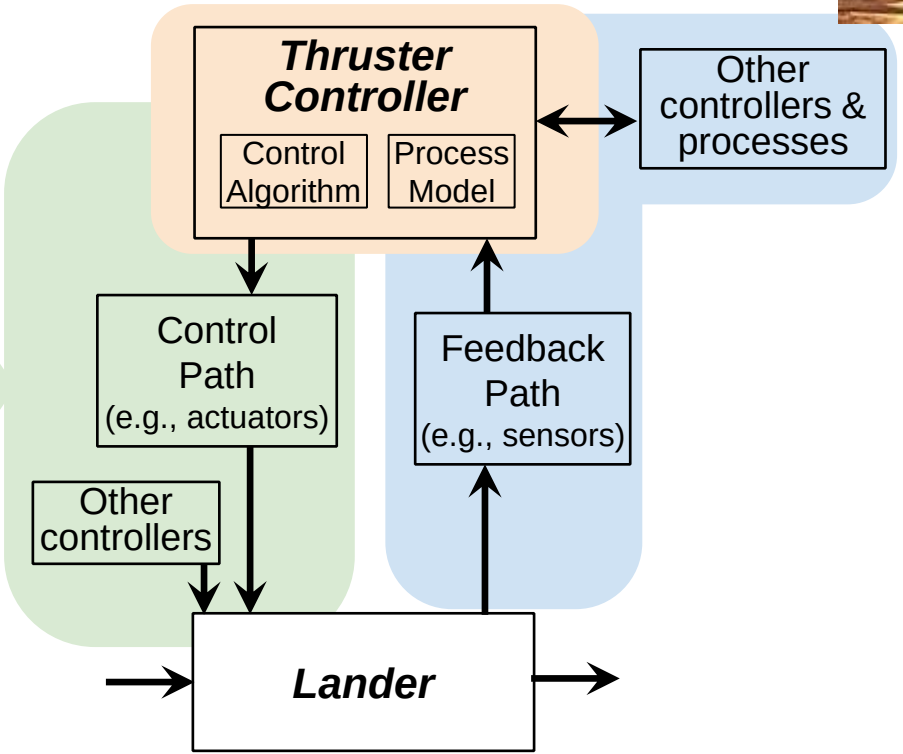
Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]



Class 3 Scenario Archetype:
Unsafe Control Execution

- *<Controller>* does not provide *<Control Action>* when *<Context>*
- *<Process>* receives *<Control Action>* when *<Context>*



© Copyright 2024 John Thomas

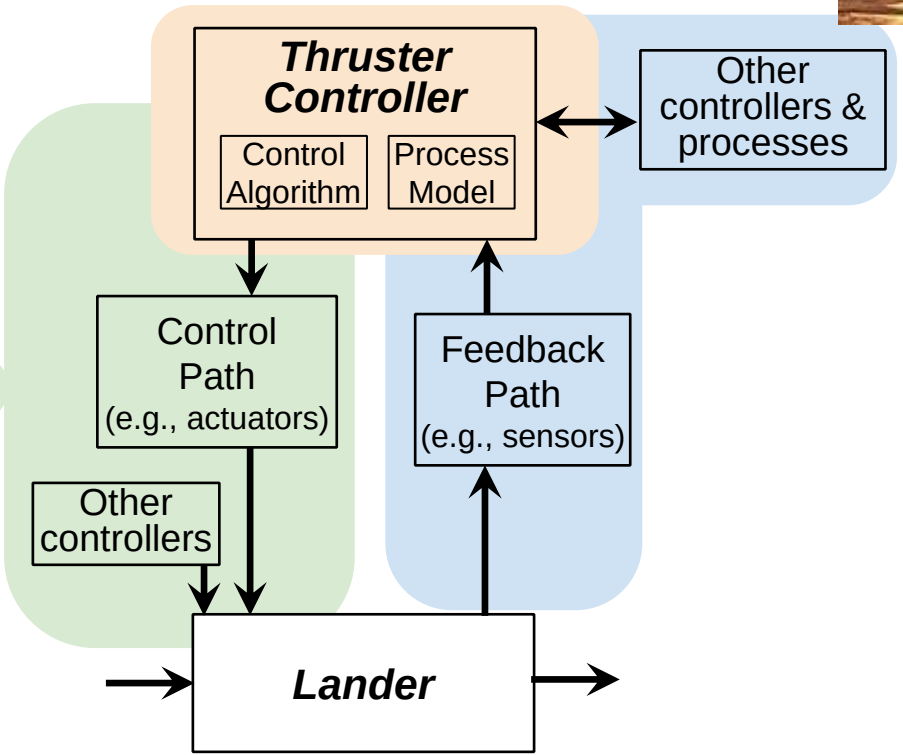
Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]



Class 3 Scenario Archetype:
Unsafe Control Execution

- *Thruster Controller* does not provide *Disable-Thruster Cmd* when *lander is in the air*
- *<Process>* receives *<Control Action>* when *<Context>*



© Copyright 2024 John Thomas

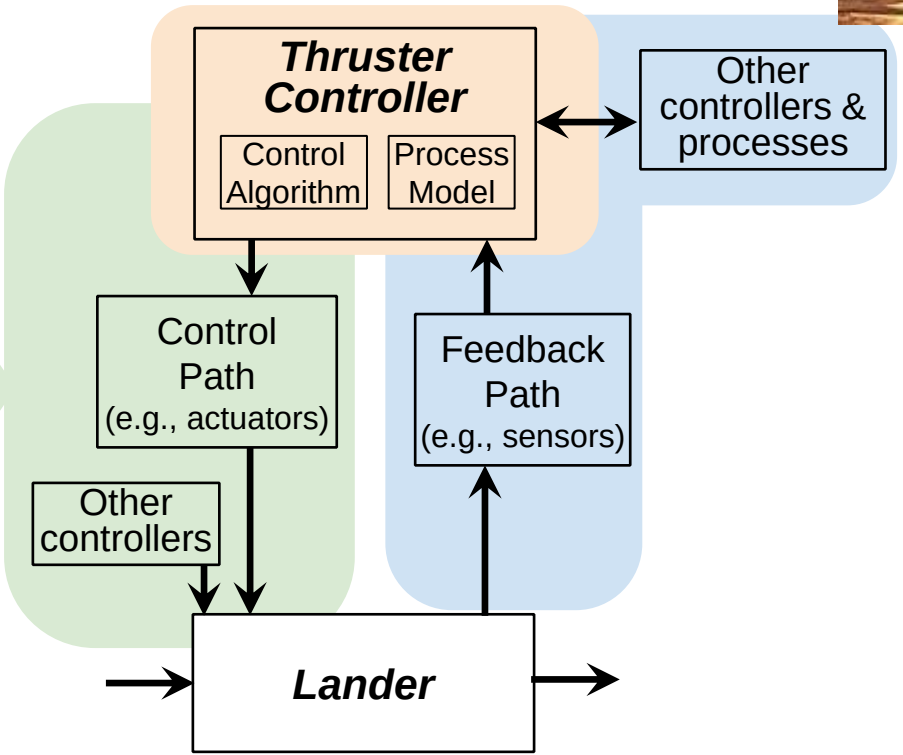
Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]



Class 3 Scenario Archetype:
Unsafe Control Execution

- *Thruster Controller* does not provide *Disable-Thruster Cmd* when *lander is in the air*
- *Lander* receives *Disable-Thruster Cmd* when *lander is in the air*



© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

Class 4 Scenario Archetype

Ask: What can cause this Scenario Archetype?

Consider:
1) Failure causes
2) Causes without failure

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

Class 4 Scenario Archetype

Refined Scenarios

Examples from real projects:

Why? Examples:
- <*Other Controller*> may generate <*Control Action*> and send it to <*Process*>
- <*Controller*> sends <*Control Action*> with Ignore bit set, but _____.
- Previous <*Control Action*> is buffered, delayed, or executed out of order
- <*Controller*> sends <*Control Action*> but it isn't received on time by <*Process*> due to ____
- <*Control Action*> may be spoofed, tampered, repudiated, denied, etc.
- Etc.

Example SME Discussion: Wait, how would <*Controlled Process*> know to ignore other ____? It might not know.

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

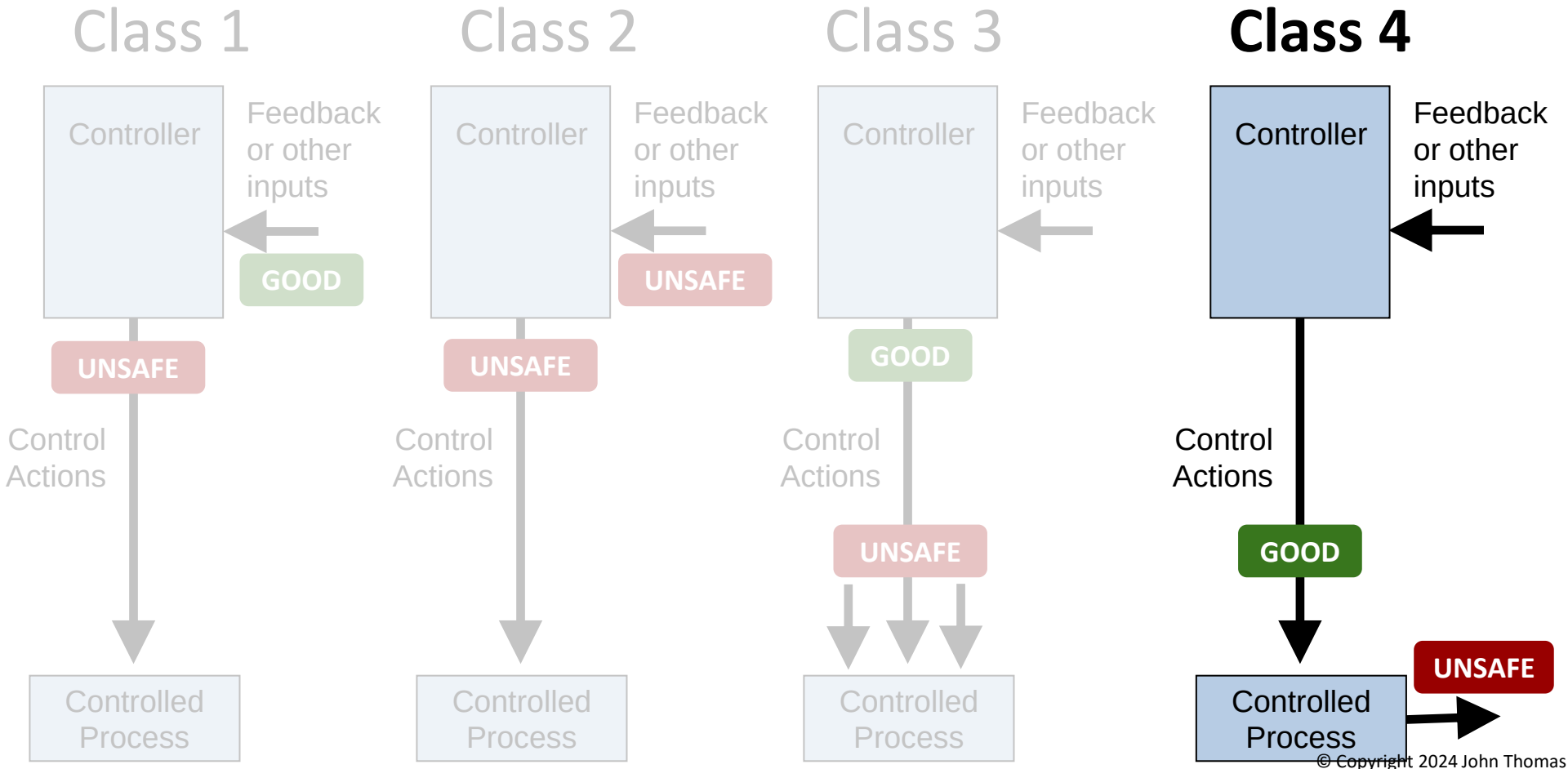
Class 4 Scenario Archetype

Refined Scenarios

Why? Common Causes:
- Conflicting control action
- Previous control actions
- Delays
- Inadequate resolution / granularity
- Dropouts
- Corruption
- Content incomplete
- Control action provided in a way the controller can't use
- Overloaded control path or too much info
- Etc.

© Copyright 2024 John Thomas

Four Classes of Formal Scenarios



© Copyright 2024 John Thomas

Result from previous STPA Step 3

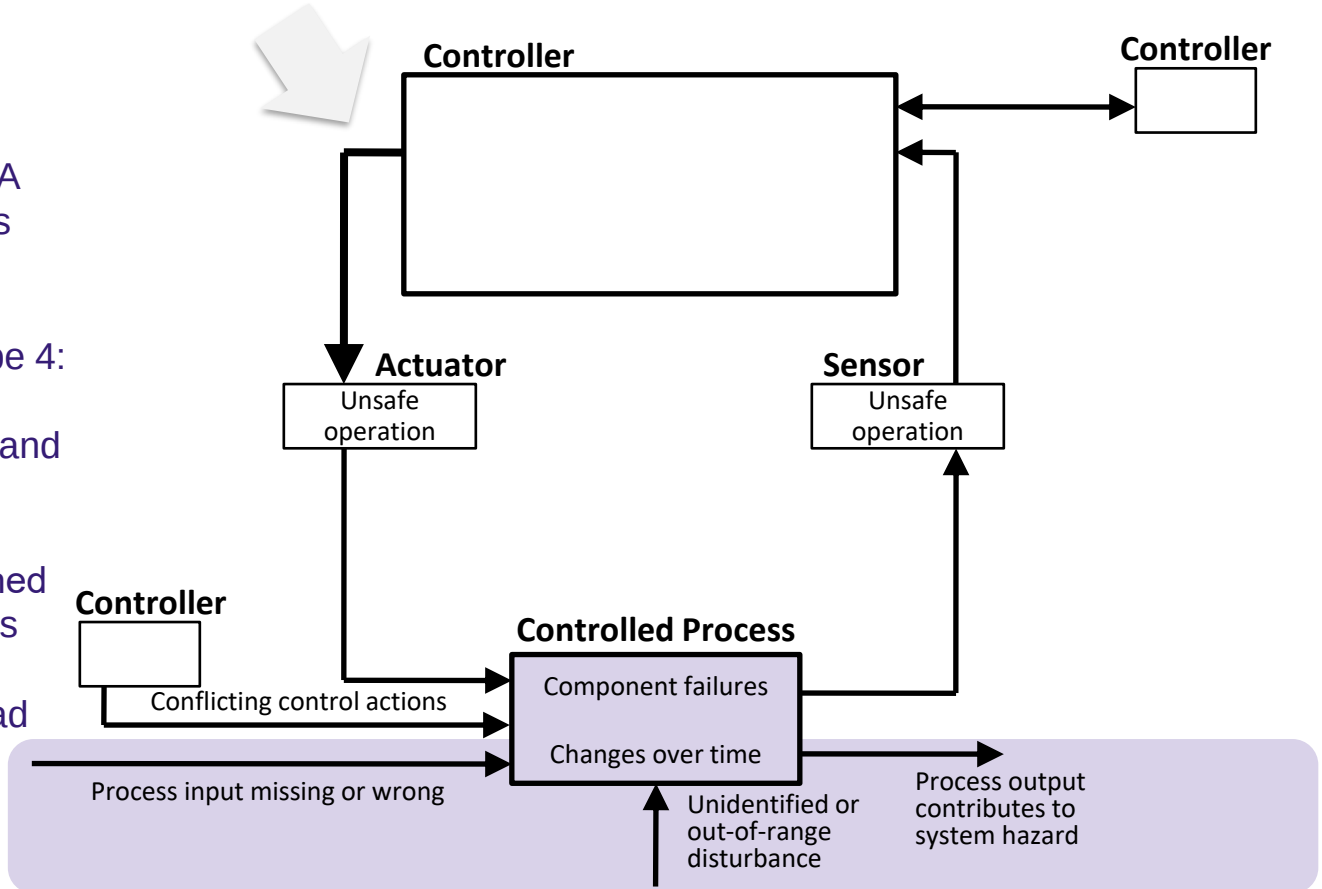
UCA: *<Controller>* provides *<Control Action>*
when *<Context> [Hazards]*

Class 4 Scenario Archetype: Unsafe Process Behavior

We need to look at how the UCA can be *emulated* due to process interactions.

Constructing Scenario Archetype 4:

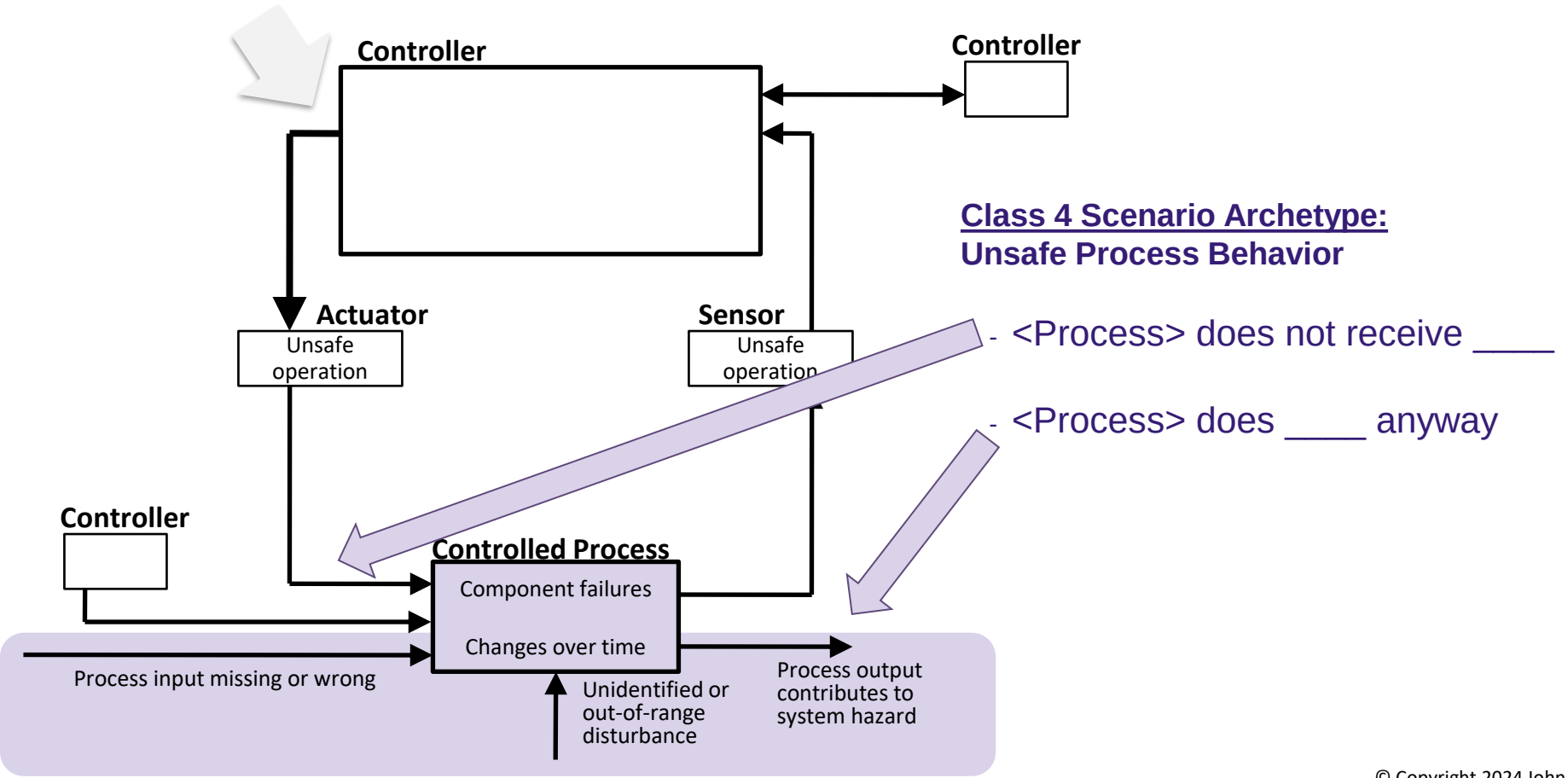
- Suppose the UCA did not happen (invert the UCA), and it was not received by the controlled process.
- BUT… something happened with the controlled process and related interactions making it as if the UCA had been provided.



© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA: <*Controller*> provides <*Control Action*>
when <*Context*> *[Hazards]*



Class 4 Scenario Archetype:
Unsafe Process Behavior

- <Process> does not receive ____
- <Process> does ____ anyway

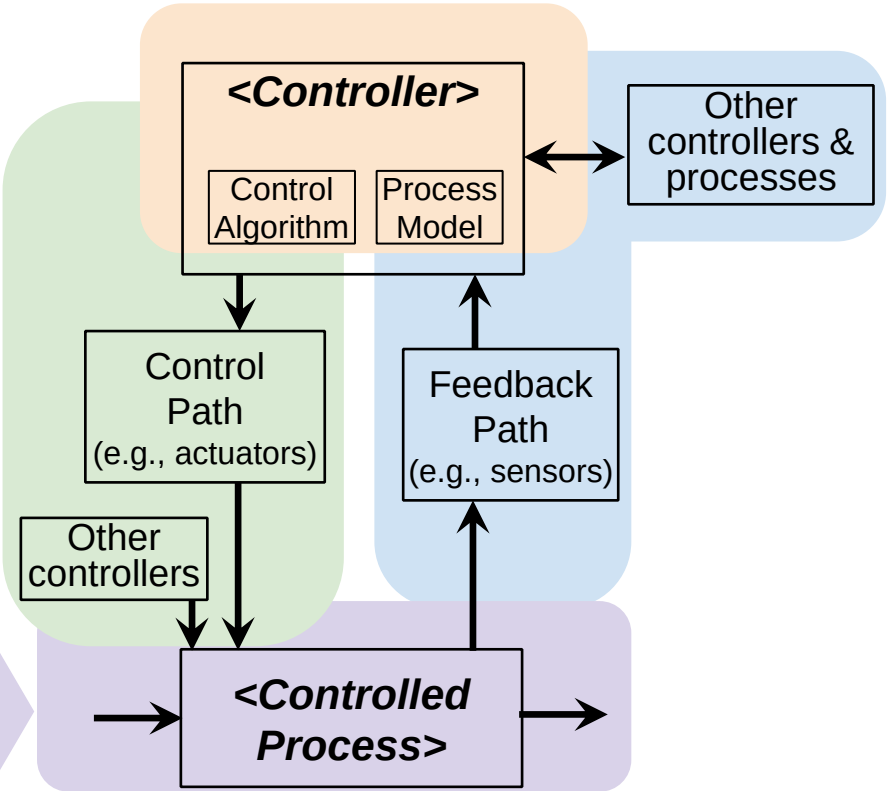
© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA: *<Controller>* provides *<Control Action>* when *<Context>* *[Hazards]*

Class 4 Scenario Archetype:
Unsafe Process Behavior

- *<Process>* does not receive *<Control Action>* when *<Context>*
- *<Process>* effectively performs *<Control Action>* when *<Context>*

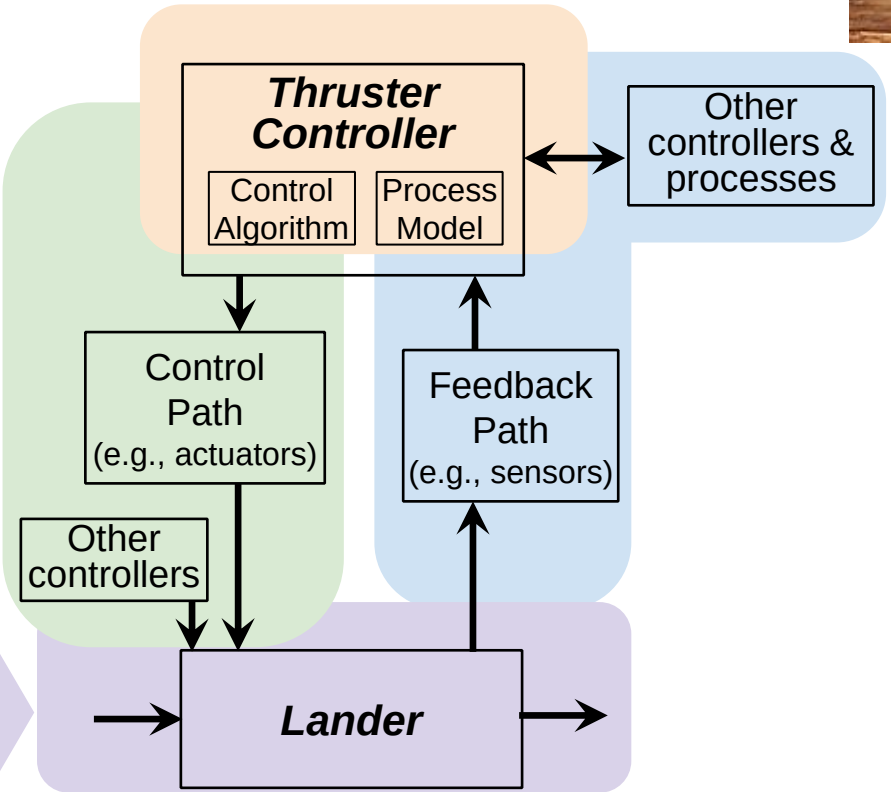


© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 4 Scenario Archetype:
Unsafe Process Behavior

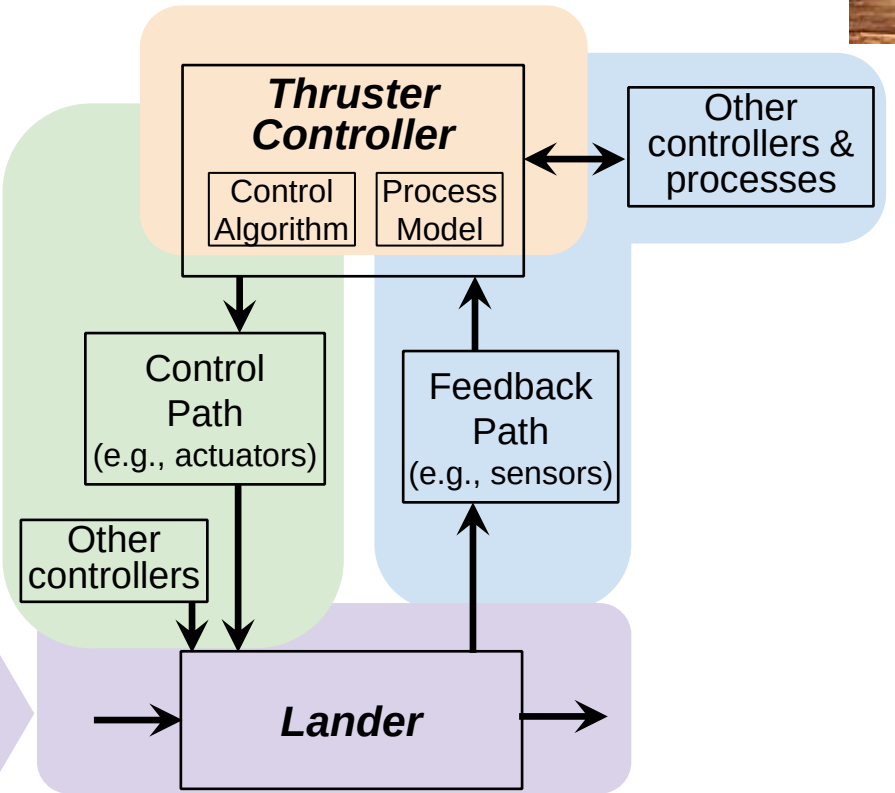
- *<Process>* does not receive *<Control Action>* when *<Context>*
- *<Process>* effectively performs *<Control Action>* when *<Context>*

© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 4 Scenario Archetype:
Unsafe Process Behavior

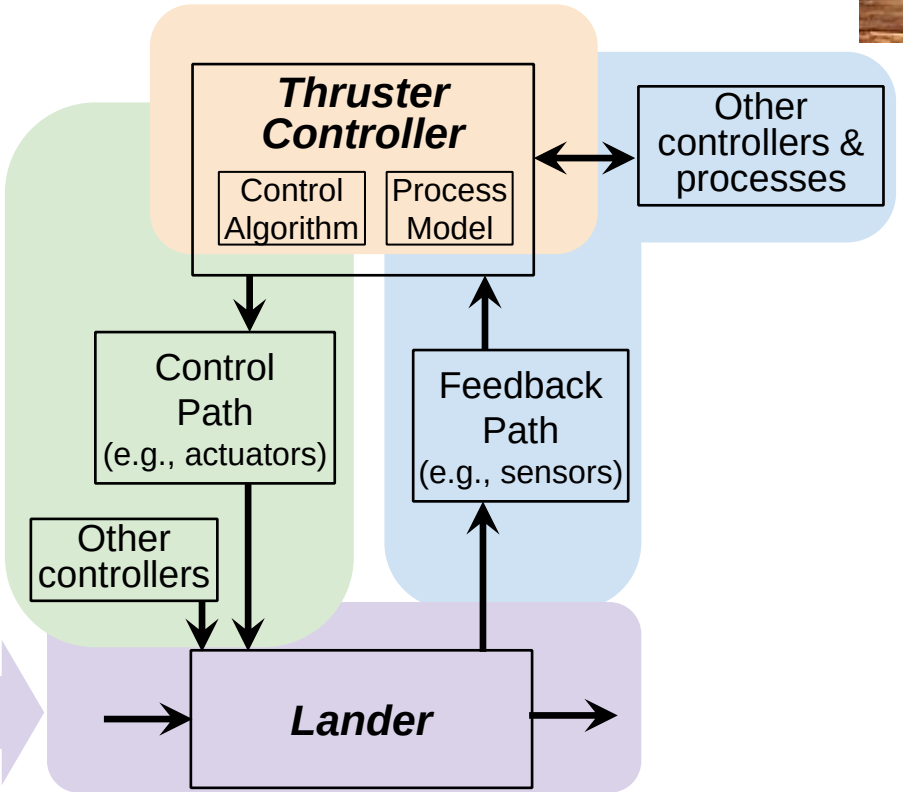
- *Lander/Thrusters* do not receive *Disable-Thruster Cmd* when *lander is in the air*
- *<Process>* effectively performs *<Control Action>* when *<Context>*

© Copyright 2024 John Thomas

Result from previous STPA Step 3

UCA-2: Thruster Controller provides Disable-Thruster Cmd when lander is in the air [H-1]





Class 4 Scenario Archetype:
Unsafe Process Behavior

- *Lander/Thrusters* do not receive *Disable-Thruster Cmd* when *lander is in the air*
- *Thrusters are* effectively *Disabled* when *lander is in the air*

© Copyright 2024 John Thomas

Scenario Archetypes

Refined Scenarios

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

Class 4 Scenario Archetype:
Unsafe Process Behavior
- <*Process*> does not receive a <*Control Action*> when <*Context*>
- <*Process*> applies <*Control Action*> when <*Context*>

Ask: What can cause this?

Consider:

1) Failure causes

2) Causes without failure

© Copyright 2024 John Thomas

Scenario Archetypes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

Class 4 Scenario Archetype:
Unsafe Process Behavior
- <*Process*> does not receive a <*Control Action*> when <*Context*>
- <*Process*> applies <*Control Action*> when <*Context*>

Refined Scenarios

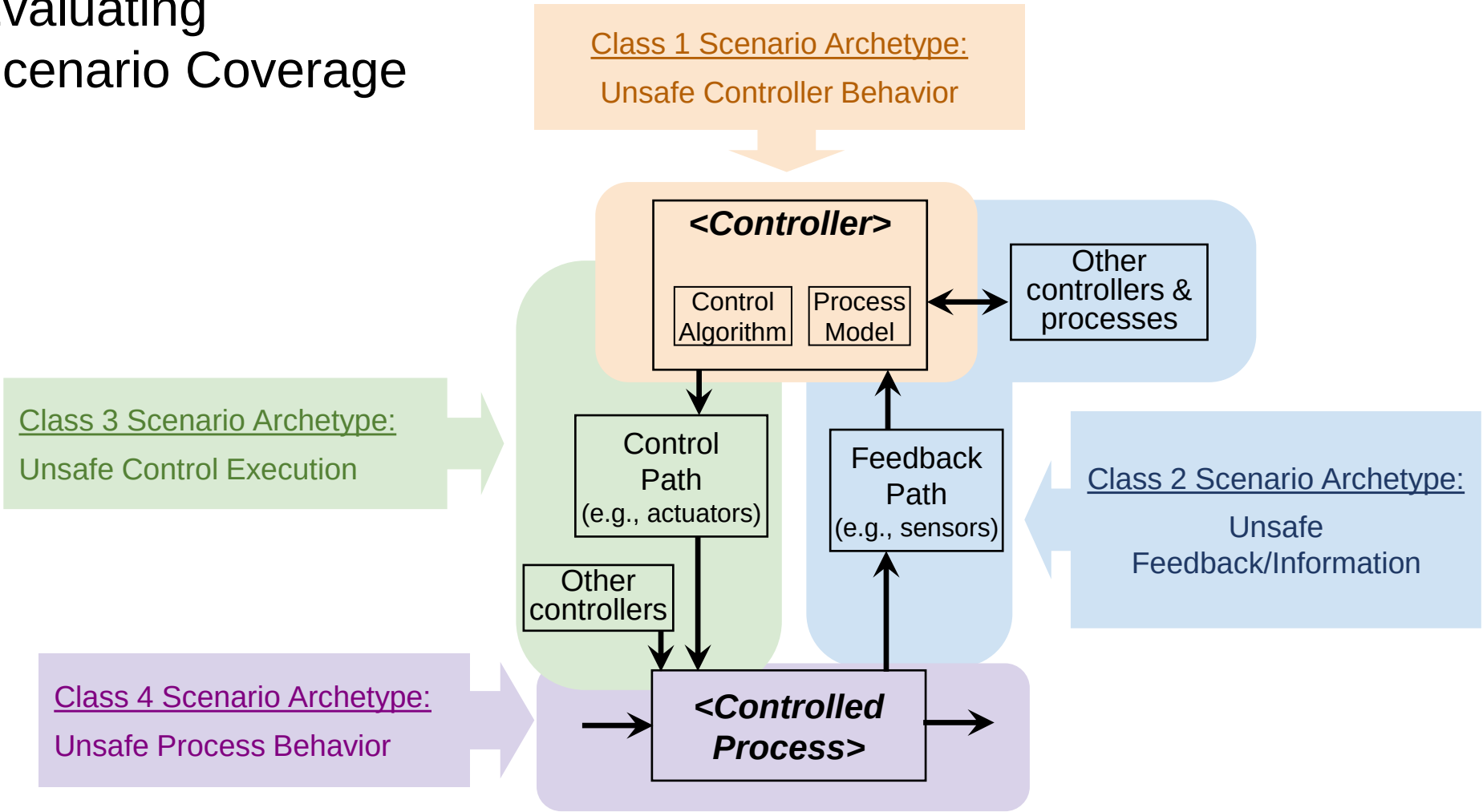
Examples from real projects:

Why?

- <Process> may mechanically ___
- <Process> may run out of ___
- <Process> responds too early before ___ (or too late after ___)
- <Process> may see that ___ is full, in which case <Process> will automatically ____, which results in ____.
- If <Process> is in ___ mode, then all ___ will be ignored.

© Copyright 2024 John Thomas

Evaluating Scenario Coverage



© Copyright 2024 John Thomas

Scenario Coverage

Class 1 Scenario Archetype:
Unsafe Controller Behavior

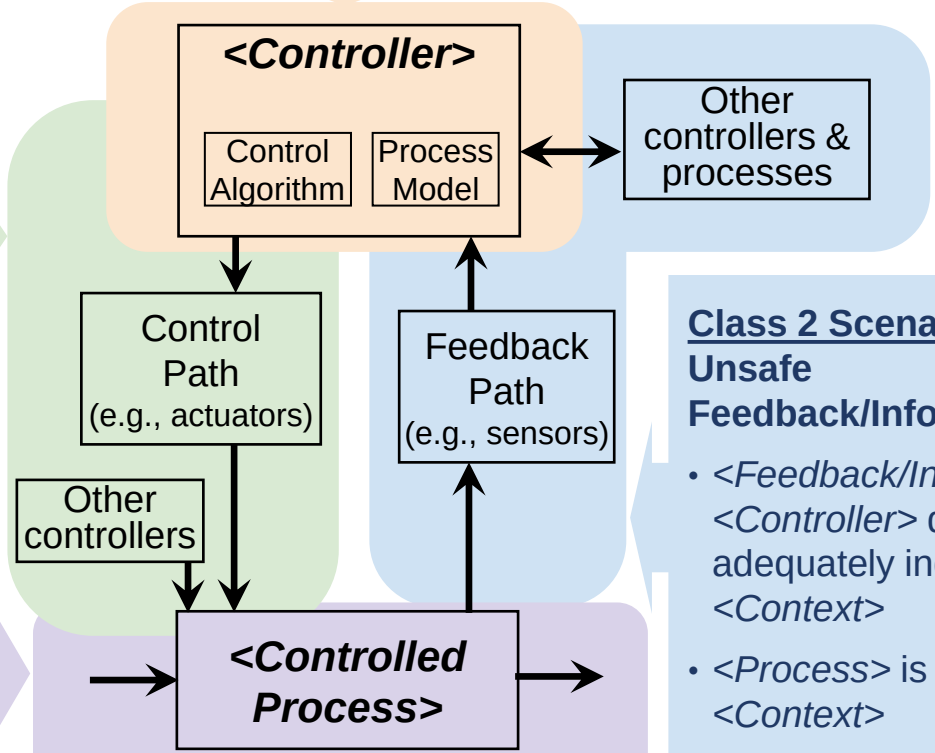
- UCA: *<Controller>* provides *<Control Action>* when *<Context>*
- *<Feedback/Input>* to *<Controller>* correctly indicates *<Context>*

Class 3 Scenario Archetype:
Unsafe Control Execution

- *<Controller>* does not provide *<Control Action>* when *<Context>*
- *<Process>* receives *<Control Action>* when *<Context>*

Class 4 Scenario Archetype:
Unsafe Process Behavior

- *<Process>* does not receive *<Control Action>* when *<Context>*
- *<Process>* effectively applies *<Control Action>* when *<Context>*



Class 2 Scenario Archetype:
Unsafe Feedback/Information

- *<Feedback/Input>* to *<Controller>* does not adequately indicate *<Context>*
- *<Process>* is actually *<Context>*

© Copyright 2024 John Thomas

STPA Formal Scenario Classes

UCA-2:
<*Controller*> provides <*Control Action*> when <*Context*>

Class 1 Scenario Archetype:
Unsafe Controller Behavior
- <*Controller*> provides <*Control Action*> when <*Context*>
- <*Input*> to <*Controller*> correctly indicates <*Context*>

Class 2 Scenario Archetype:
Unsafe Feedback/Information
- <*Feedback/Input*> to <*Controller*> does not adequately indicate <*Context*>
- <*Process*> is actually <*Context*>

Class 3 Scenario Archetype:
Unsafe Control Execution
- <*Controller*> does not provide <*Control Action*> when <*Context*>
- <*Process*> receives a <*Control Action*> when <*Context*>

Class 4 Scenario Archetype:
Unsafe Process Behavior
- <*Process*> does not receive a <*Control Action*> when <*Context*>
- <*Process*> applies <*Control Action*> when <*Context*>

Refined scenarios

1) Failure causes

2) Causes without failure

Discussion

These formal scenario classes can be used to:

- Establish contractual / regulatory compliance
- Improve consistency, inform reviews
- Generate the classes from UCAs
- Compare coverage/scope between different safety & reliability methods

© Copyright 2024 John Thomas

STPA Scenario Archetype Table

Control Action	Not providing causes hazard UCA-1	Providing causes hazard UCA-2	Too early, too late, Order UCA-3	Stopped Too Soon / Applied too long UCA-4
Scenario Class 1: Unsafe Controller Behavior	1)<controller> doesn't provide <cmd> when <context> 2)<controller> received feedback (or other inputs) that indicates <context>	1)<controller> provides <cmd> when <context> 2)<controller> received feedback (or other inputs) that indicates <context>	1)<controller> provides <cmd> too late/early after/before <context> 2)<controller> received feedback (or other inputs) that indicates <context> on time / in order	1)<controller> stops/continues providing <cmd> too soon/long 2)<controller> received feedback (or other inputs) that indicates <context> on time
Scenario Class 2: Unsafe Feedback Path	1)feedback (or other inputs) received by <controller> does not adequately indicate <context> 2)<context> is true	1)feedback (or other inputs) received by <controller> does not adequately indicate <context> 2)<context> is true	1)feedback (or other inputs) received by <controller> does not indicate <context> (too late/early/out of order) 2)<context> is true	1)feedback (or other inputs) received by <controller> does not indicate <context> (inappropriate duration) 2)<context> is true
Scenario Class 3: Unsafe Control Path	1)<controller> does provide <cmd> when <context> 2)<cmd> is not received by <controlled process> when <context>	1)<controller> does not provide <cmd> when <context> 2)<controlled process> receives <cmd> when <context>	1)<controller> does not provide <cmd> <context> (not too late/early/out of order) 2)<cmd> is received by <controlled process> <context> (too late/early/out of order)	1)<controller> provides <cmd> with appropriate duration 2)<cmd> is received by <controlled process> with <context> (inappropriate duration)
Scenario Class 4: Unsafe Controlled Process Behavior	1)<cmd> is received by <controlled process> when <context> 2)<controlled process> does not respond by <…>	1)<cmd> is not received by <controlled process> when <context> 2)<controlled process> responds by <…>	1)<cmd> is not received by <controlled process> <context> (not too late/early/out of order) 2)<controlled process> responds by <…> <context> (too late/early/out of order)	1)<cmd> is received by <controlled process> with appropriate duration 2)<controlled process> does not respond by <…> with <context> (inappropriate duration)

(Thomas, 2016), (Thomas, 2017), (Cabosky, 2020)

© Copyright 2024 John Thomas

Classes 1-4 are already in the STPA Handbook

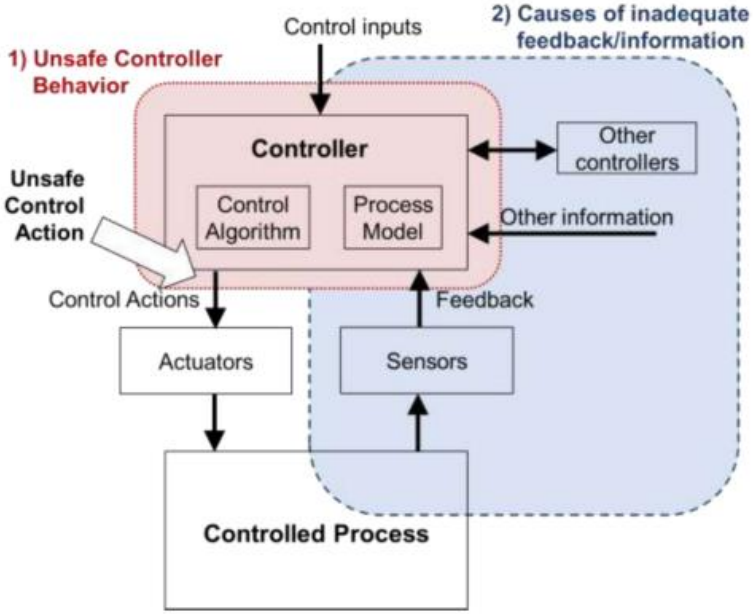


Figure 2.18: Unsafe Control Actions can be caused by (1) unsafe controller behavior and (2) inadequate feedback and other inputs

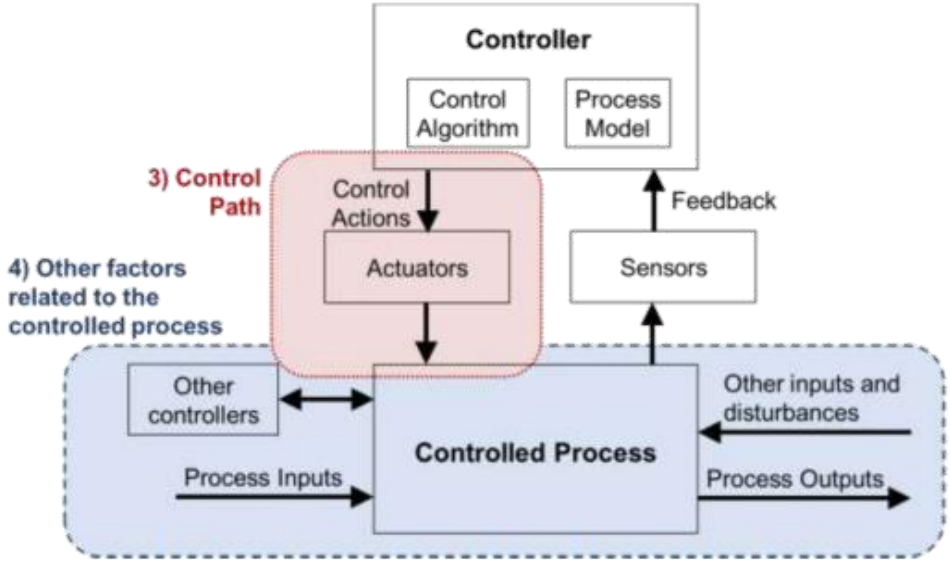


Figure 2.19: Generic control loop illustrating 1) the control path and 2) other factors that can affect the controlled process

Discussion: The new scenario process is consistent with the STPA Handbook but provides a more formal process with more guidance.

STPA Handbook, 2018

© Copyright 2024 John Thomas

Testing the New Approach: Results

© Copyright 2024 John Thomas

Many Real-World Evaluations Completed

Examples:

- (Nuclear) A New Process for Building STPA Causal Scenarios, John Thomas, 2016 MIT STAMP Workshop
- (Space) A Process for STPA: STAMP Accident Model of HITOMI and Expansion to Future Safety Culture, John Thomas and Nancy Leveson (MIT), Masa Katahira and Naoki Ishimama (**JAXA**), Nobuyuki Hoshino (JAMSS), 2017 MIT STAMP Workshop
- (Aircraft) Systems Theoretic Process Analysis Applied to **Air Force** Acquisition Technical Requirements Development, Sarah Summers, MIT Thesis, 2017
- (Aircraft) STPA Applied to Manned-Unmanned Teaming, Jeremiah Robertson, MIT Thesis, 2019
- (Auto) STPA Applied to Autonomous Vehicles, Jeff Stafford (**Renesas**), John Thomas (MIT), 2019 MIT STAMP Workshop
- (Software) STPA Applied to AV Software, Shaun Mooney (**Codethink**), John Thomas (MIT), 2019
- (Auto) Application of Hierarchy to STPA: A Human Factors Study on Vehicle Automation, Rachel Cabosky, 2020, MIT Thesis (collaboration with **GM**)
- (Military Aviation) Evaluation of STPA for Aircraft Safety Assessment, **US Army**, 2021
- (Aviation) Rotary-Wing Aircraft Development: Cybersecurity and Safety STPA Status Report, MIT & MIT-LL, 2021
- (Aviation) A Top-Down, Safety-Driven Approach to Architecture Development for Complex Systems, Justin Poh, MIT Thesis, 2022
- (Aviation) System-Theoretic Safety Analysis for Teams of Collaborative Controllers, Andrew N. Kopeikin, MIT Dissertation, 2024
- (Communications) System-Theoretic Process Analysis of a Novel Airborne Laser Communication System, Brittany Bishop, MIT Thesis, 2024
- (Software) OEM & Supplier Use of STPA for Advanced Driver-Assistance Systems, **Qualcomm**, 2024
- (Software) Collaboration with Tim Falzone, Ruben Barroso, Garrett Holthaus, **Google**, 2024

In every application, the new scenario process has identified critical loss scenarios and causes that were previously overlooked

© Copyright 2024 John Thomas

New Scenario Approach: Observations

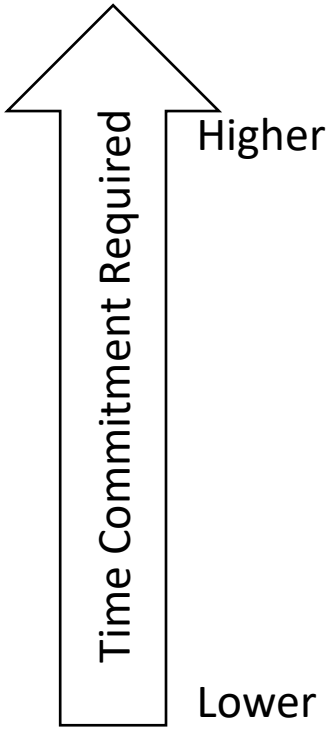
Disadvantages	**Advantages**
• More rules and structure • Takes longer to teach & learn • Unclear if it takes longer to perform (less time in some cases)	• The rules and structure provide more guidance • Enables a more directed search, less ad-hoc and informal • The rationale for the scenarios and how you found them is clearer. • So far, the new process has always captured additional cases that were previously overlooked • The top-down approach was more scalable to extremely complex systems compared to previous STPA attempts • Less repetition in the results (shorter documentation, higher information density) • Provides clear exit criteria to rigorously review and find gaps • Enables automation of some parts of Step 4 • The formal structure can enable mathematical proofs for properties like scenario coverage • Improved consistency and repeatability. Less dependence on who is doing the analysis.

© Copyright 2024 John Thomas

How to run an STPA project

Who would do this and how they would coordinate with others?

STPA Project Participants



STPA Core Team

STPA Practitioners

STPA Facilitator

STPA Practitioners
- Perform majority of STPA work
- Interdisciplinary team
- Should have basic STPA training and certification, but may not be STPA experts

STPA Facilitator
- STPA expert
- Provide STPA method guidance (and other responsibilities)

Other SMEs

- Provide specialized domain knowledge as needed by team
- May have little or no STPA familiarity
- May not be actively involved in STPA, but must be accessible by team