



# STPA Hazard Analysis Practice

March 23, 2026

John Thomas, MIT

Theo Klein, Staff Site Reliability Engineer, Google

Garrett Holthaus, Technical Writer, Google



Site Reliability Engineering

# Software Outages and Incident Response

- Software **incident**: increased latency, errors, service down completely, etc.
- **Site Reliability Engineers (SREs)** handle incident response for established, critical services
- SREs **triage** the issue and act quickly to **mitigate**
- Later, we **resolve** the incident by fixing the cause(s)

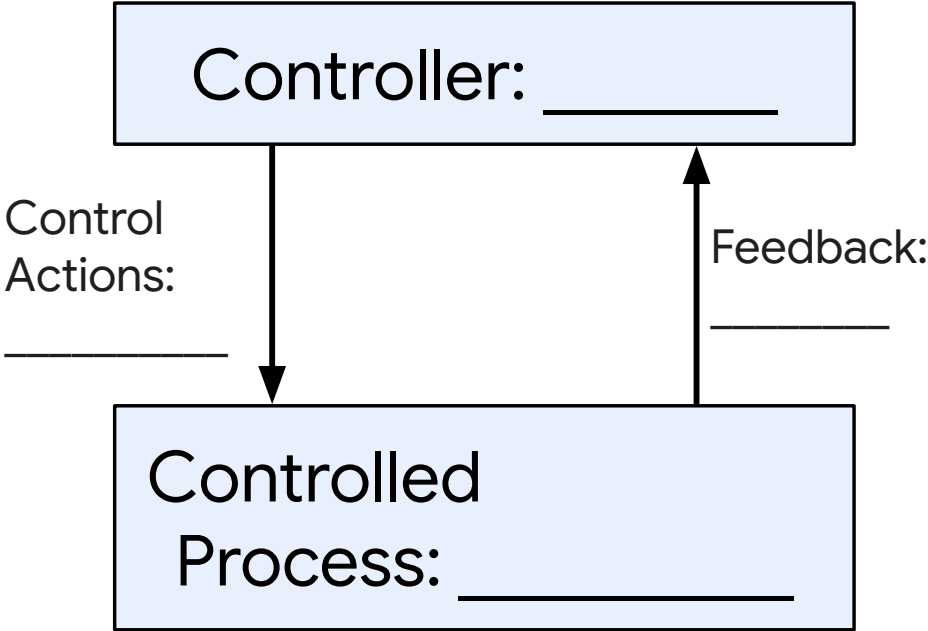


AI generated image created by Gemini in March 2026

# Activity: Build the Control Structure

# Breakout 1: Draw the control structure for incident response

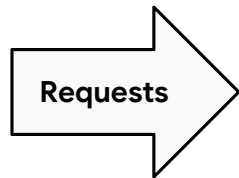
- 1. Read the SRE On-Call Primer
- 2. Look for key elements:
  - a. Controlled Process
  - b. Controller
  - c. Control Actions
  - d. Feedback
- 3. Draw and label the control structure—Fill in the blanks with the actual controller name, controlled process name, etc.



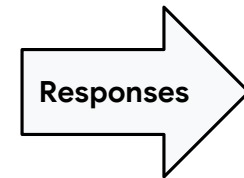
# Incident Response Control Structure



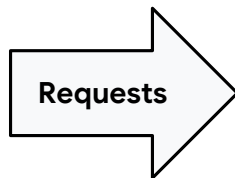
**Site Reliability Engineering**



**Production Service**



- **Responsibilities:** Ensure the Prod Service responds within latency target
- **Control Action:** Rollback to previous version
- **Decision Making:** IF errors AND errors correlate to new version THEN roll back
- **Mental Model 1:** Current state of production is returning errors
- **Mental Model 2:** Correlation between errors and release of new version



## Site Reliability Engineering

## Production Service

Let's run STPA!

# Step 1: Define Losses

- **Losses:** Unacceptable system outcomes
  - L-1 Loss of revenue
  - L-2 Loss of brand trust
  - L-3 Loss of legal compliance
  - L-4 Injury or loss of life
- Losses should not reference low-level details or causes

# Step 1: Define Hazards

**Hazard:** A system state that will lead to a loss under worst case environmental conditions

- H-1 Production Service is unreachable [L-1, L-2]
- H-2 Production Service delivers outdated or inaccurate data [L-1, L-2, L-3]
- H-3 Production Service unable to respond within the desired time [L-1, L-2]
- H-4 Production Service delivers data to unauthorized entities [L-2, L-3]

# Step 1: Define Safety Constraints

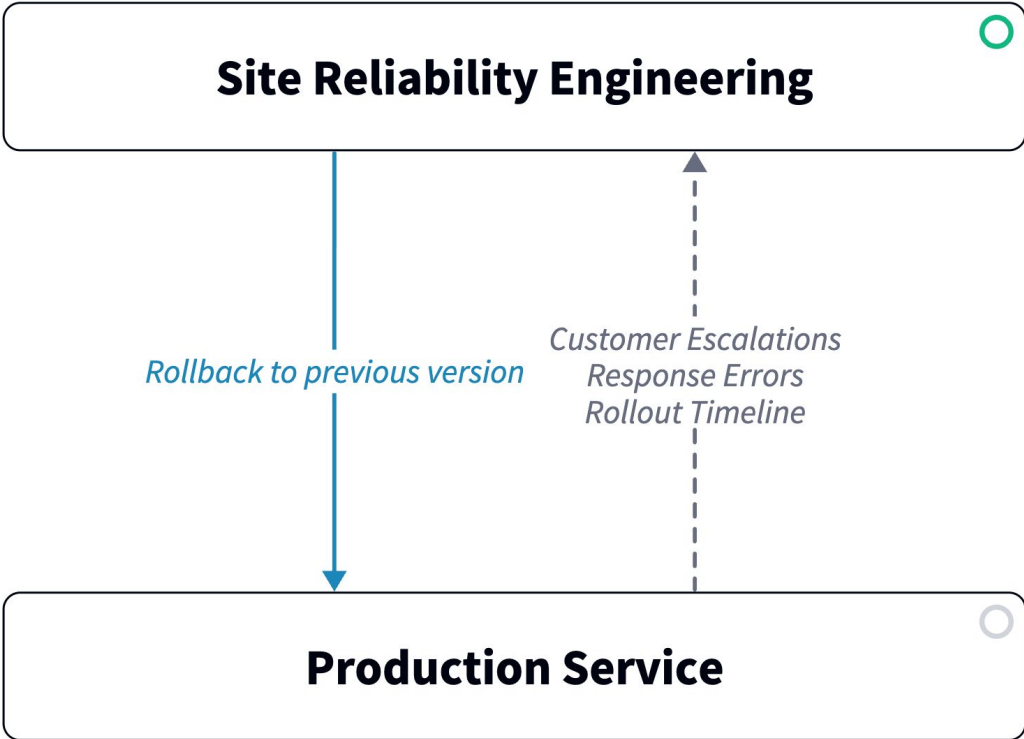
**Safety Constraint:** Condition or behavior that must be satisfied to prevent hazard

- H-1 Production Service is unreachable
  - SC-1.1 Production service must be reachable
  - SC-1.2 If the production service is unreachable, incident response must be notified
- H-2 Production Service delivers data that is outdated, inaccurate, or misleading
  - SC-2.1 Production service must deliver accurate and up-to-date data
  - SC-2.2 If the data is inaccurate or out-of-date, the customer and incident response must be notified

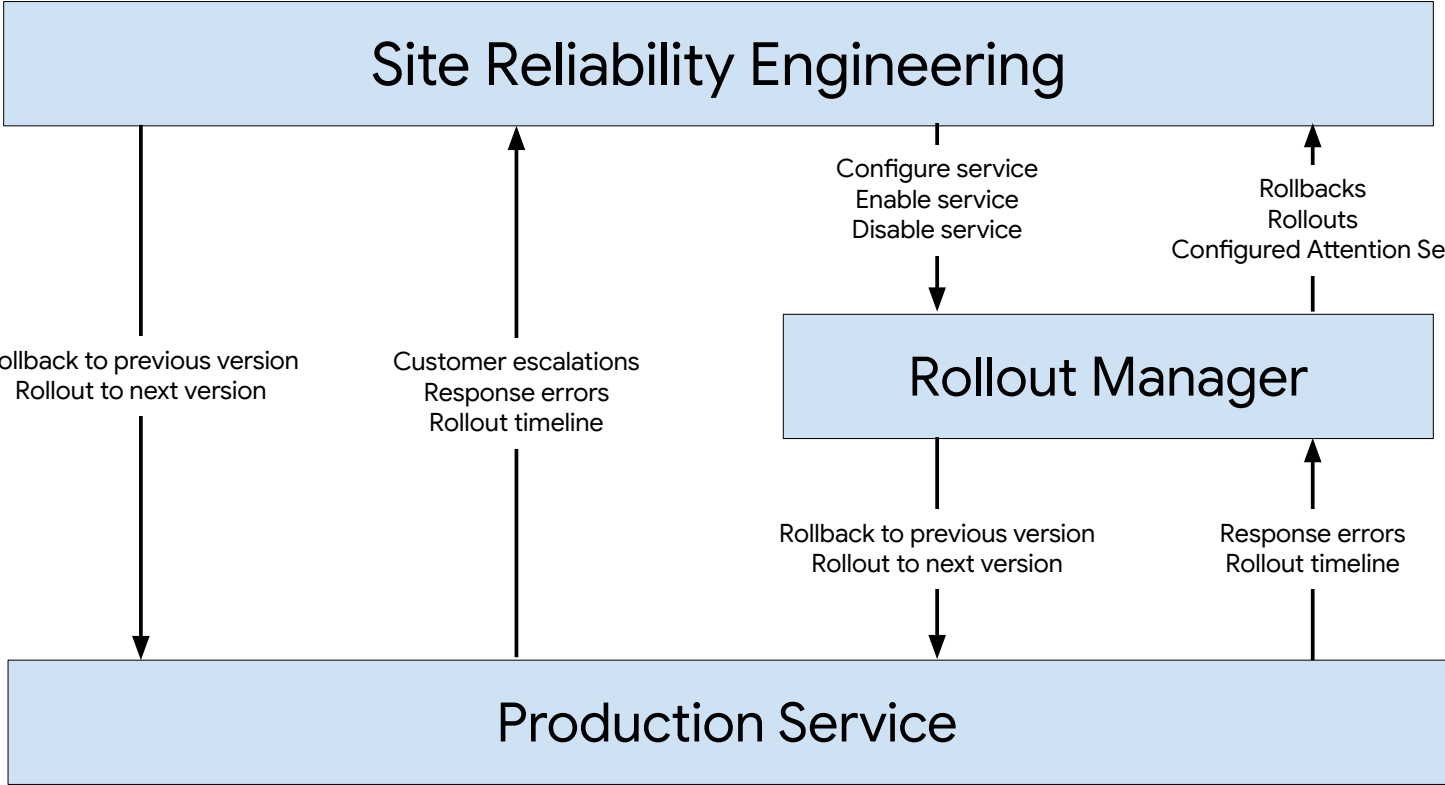
# Step 1: Define Safety Constraints (cont.)

- H-3 Production Service unable to respond within desired time
  - SC-3.1 Production service must respond within maximum specified latency
  - SC-3.2 If the production takes too long to respond, incident response must be notified
- H-4 Production Service delivers data to unauthorized entities
  - SC-4.1 Production service must deliver data only to authorized entities
  - SC-4.2 If production service delivers data to unauthorized entities, security incident response and legal must be notified

# Step 2: Model the System



# More detailed control structure



# Breakout 2: Unsafe Control Actions

Control  
Action:  
Rollback

	Not providing causes hazard	Providing causes hazard	Too Early, Too Late, Out of Order	Stopped Too Soon / Applied too long
	<p>&lt;controller&gt; does not provide &lt;control action&gt; when &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p>	<p>&lt;controller&gt; provides &lt;control action&gt; when &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p>	<p>&lt;controller&gt; provides &lt;control action&gt; too late after (&gt;TBD seconds after) &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p> <p>&lt;controller&gt; provides &lt;control action&gt; too early before (&gt;TBD seconds before) &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p>	<p>&lt;controller&gt; stops providing &lt;control action&gt; too soon before (&gt;TBD seconds before) &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p> <p>&lt;controller&gt; continues providing &lt;control action&gt; too long after (&gt;TBD seconds after) &lt;context&gt; [<b>&lt;hazard IDs&gt;</b>]</p>



Write ~4 UCAs for columns 1 & 2. If time permits, try columns 3 & 4.

## Step 3: Unsafe Control Actions

- **Type 1: Provided**
- UCA-1: SRE **provides** Rollback to previous version when \_\_\_\_\_
- UCA-1: SRE provides Rollback to previous version when the current version is not causing client impact [H-5?]
  - H-5: Production service unable to maintain release cadence [L-3]
- UCA-2: SRE provides Rollback to previous version when the previous version has a security issue [H-4]
- UCA-3: SRE provides Rollback to previous version when the previous version is incompatible with the current database organization (schema) [H-1, H-2]

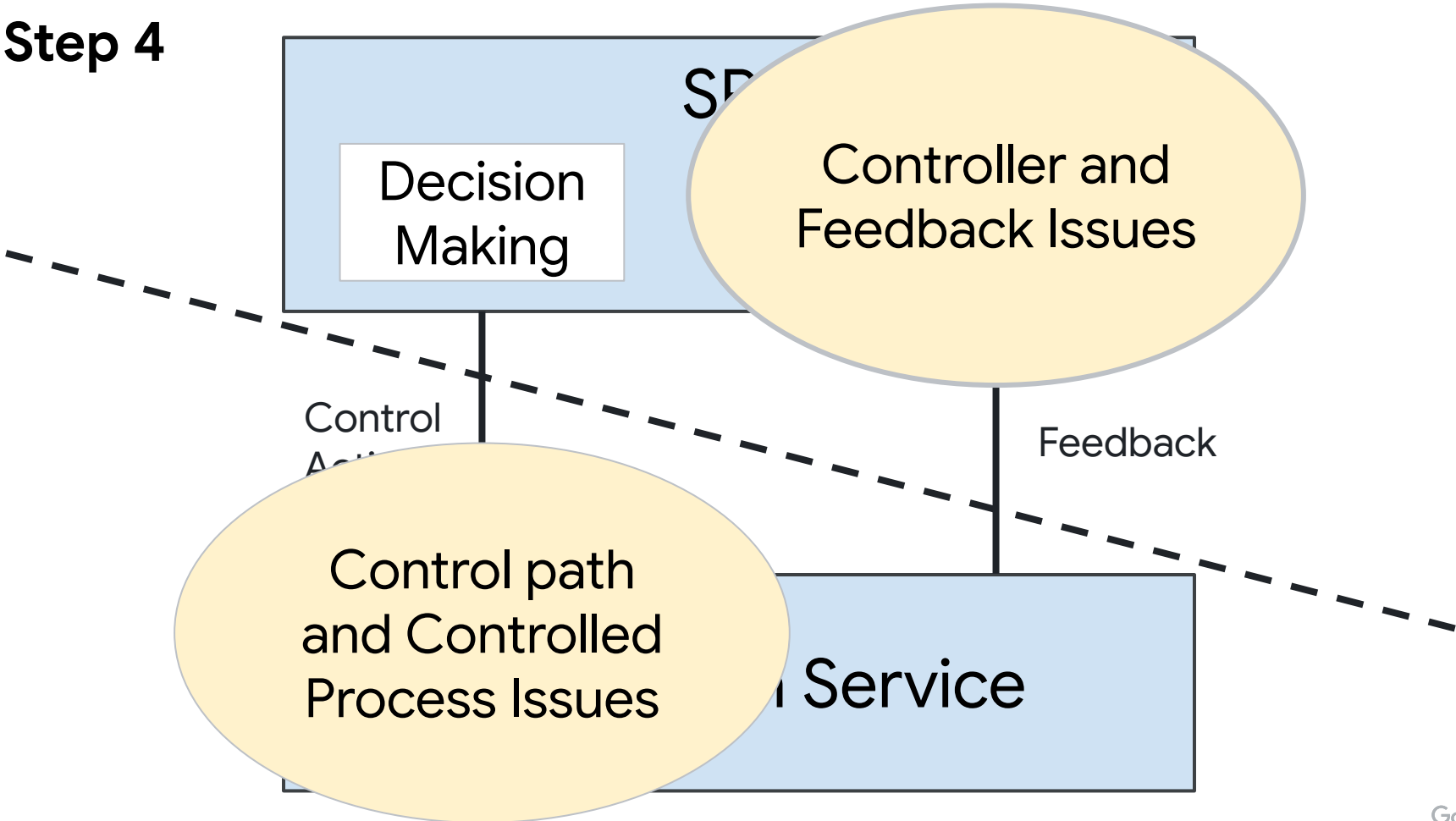
## Step 3: Unsafe Control Actions cont.

- **Type 2: Not provided**
- UCA-4: SRE **does not provide** Rollback to previous version when \_\_\_\_\_
- UCA-4: SRE does not provide Rollback to previous version when the current version is causing client impact [H-1, H-2, H-3]
- UCA-5: SRE does not provide Rollback to previous version when the current version is corrupting data [H-2]
- UCA-6: SRE does not provide Rollback to previous version when the current version isn't legally compliant [H-2, H-4]

## Step 3: Unsafe Control Actions cont.

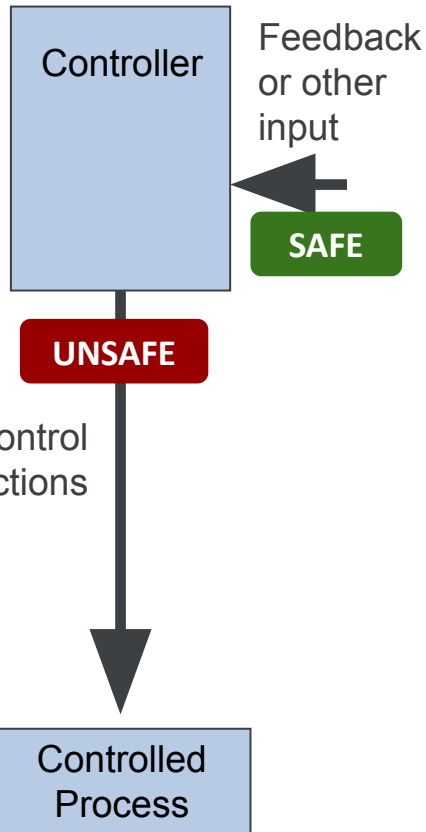
- **Type 4: Provided too long, stopped too soon**
- Rollouts and rollbacks are *progressive* - i.e. roll out to 1% first, then 10%, then 100%
- SRE continues providing Rollback cmd too long after rollback begins interfering with production service
- SRE stops providing Rollback cmd too soon before the production service issue is mitigated
  - I.e. SRE stops the rollback before it has fully completed, and some percentage of production is still using the version with errors

# Step 4

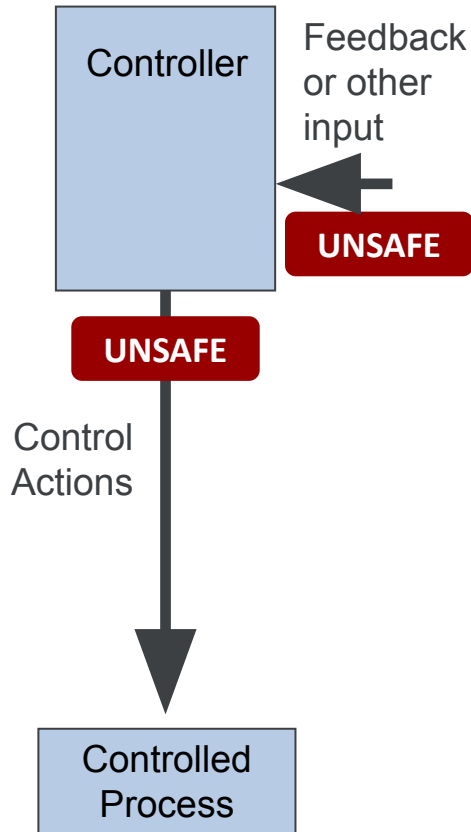


# STPA: Four Classes of Formal Scenarios

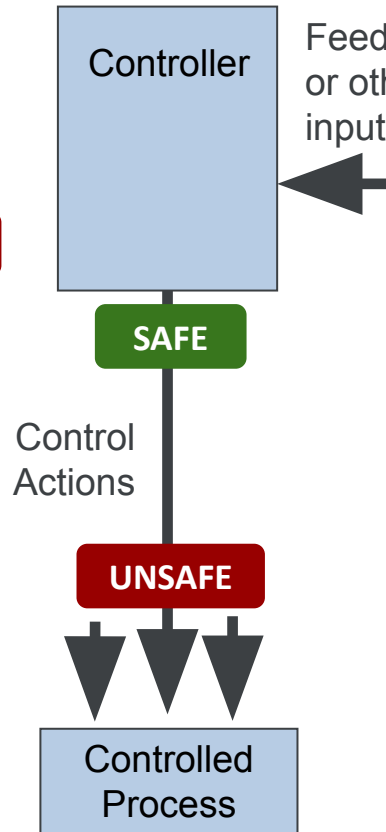
## Class 1



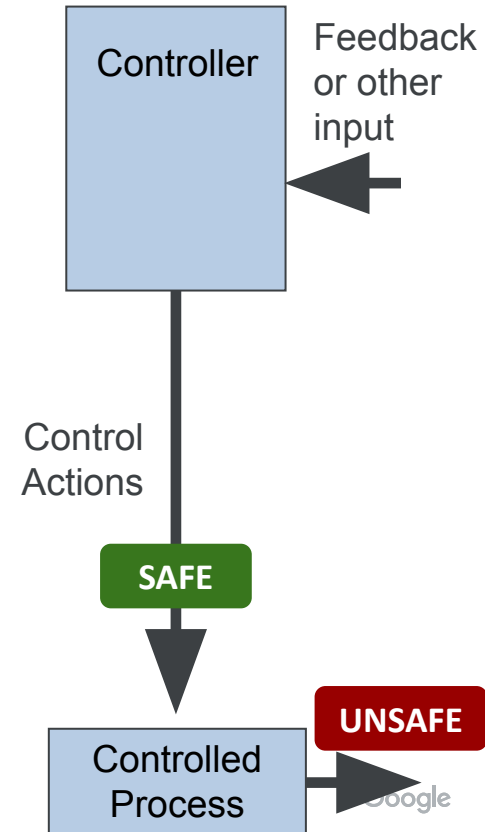
## Class 2



## Class 3



## Class 4



# Breakout 3: STPA Scenario Classes

UCA:

<Controller> does not provide <Control Action> when <Context>

## Class 1 Scenario:

### Unsafe Controller Behavior

- <Controller> does not provide <Control Action> when <Context>, and
- <Controller> received feedback correctly indicating <Context>

## Class 2 Scenario:

### Unsafe Feedback/Information

- Feedback received by <Controller> does not adequately indicate <Context>, and
- <Context> is true

## Class 3 Scenario:

### Unsafe Control Execution

- <Controller> provides <Control Action> when <Context>, and
- <Process> does not receive <Control Action> when <Context>

## Class 4 Scenario:

### Unsafe Process Behavior

- <Process> receives <Control Action> when <Context>, and
- <Process> does not respond by <Control Action Response>

UCA-4:

**SRE does not provide Rollback when current version is causing client impact**

Controller = SRE

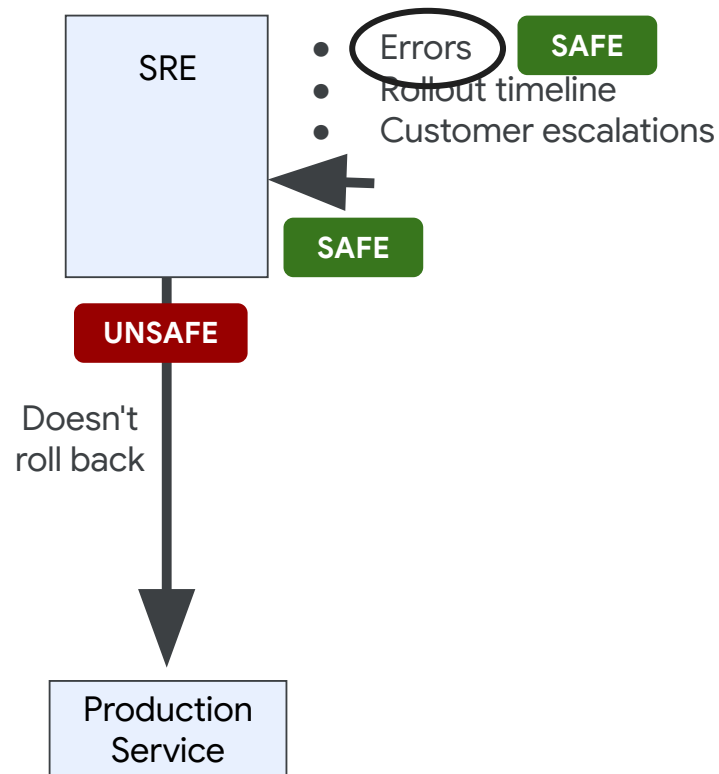
Control Action = Rollback

Context = Current version is causing client impact

**Deliverable:** Write the four scenario classes for our UCA above

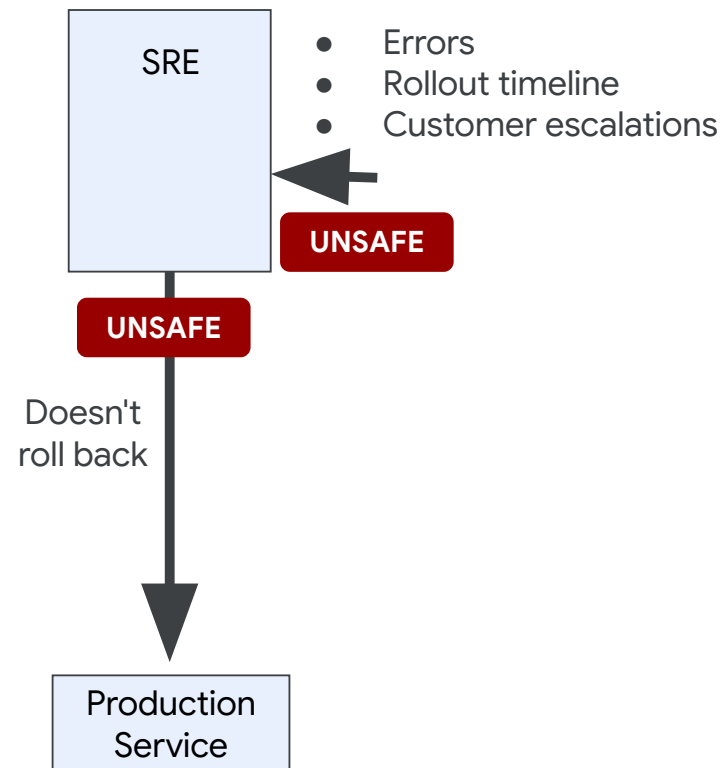
## Step 4: Class 1 formal scenario (Unsafe Controller Behavior)

- **SRE** does not provide **roll back** when **the current version is causing client impact**
- **SRE** received feedback correctly indicating **client impact**
- **SRE** received **errors** feedback correctly indicating **client impact**



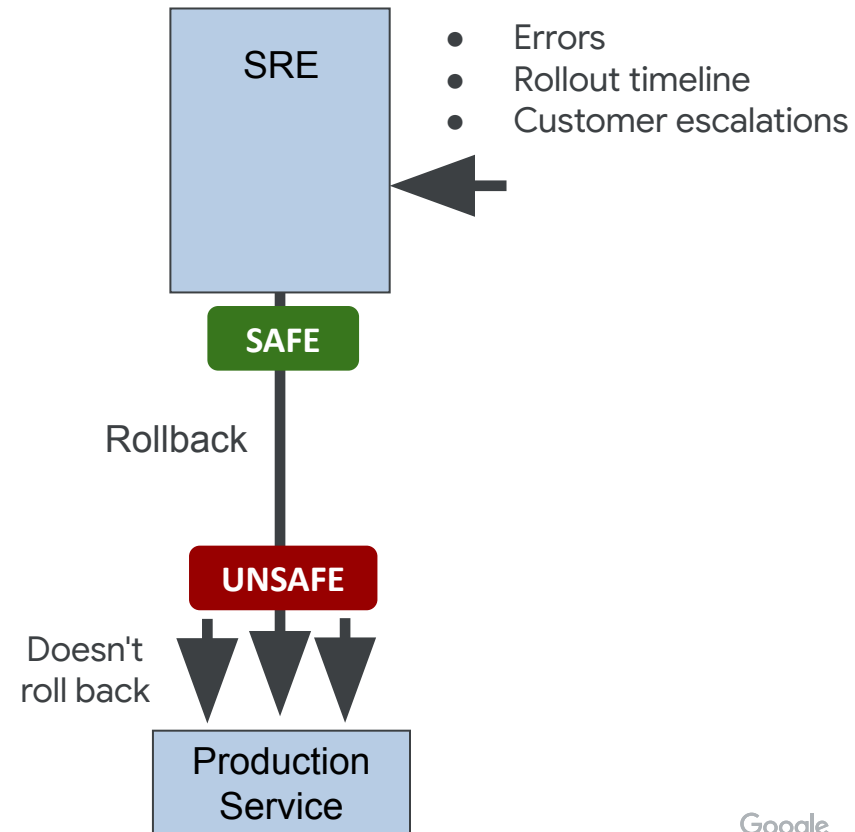
## Step 4: Class 2 formal scenario (Unsafe Feedback / Information)

- Feedback received by **SRE** does not adequately indicate **there is client impact**.
- **Current version of production service is causing client impact.**



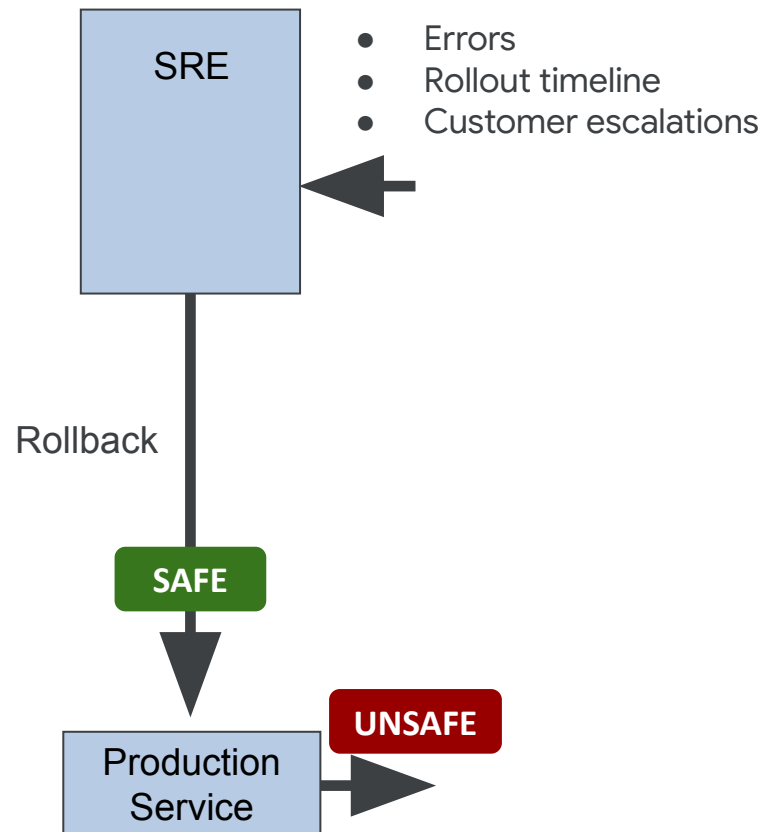
## Step 4: Class 3 formal scenario (Unsafe Control Execution)

- **SRE** provides **roll back** when **current version of production service is causing client impact**.
- However, **production service** does not receive **roll back** when **there is impact**.



## Step 4: Class 4 formal scenario (Unsafe Process Behavior)

- **Production service** receives **roll back**, when **current version is causing impact**.
- **Production service** does not respond by **rolling back**.



# STPA Scenario Classes

UCA: SRE does not provide Rollback when current version is causing client impact

## Class 1 Scenario:

### **Unsafe Controller Behavior**

- *SRE does not provide Rollback when current version is causing client impact*
- *SRE received feedback correctly indicating client impact*

## Class 2 Scenario:

### **Unsafe Feedback/Information**

- *Feedback received by SRE does not adequately indicate there is client impact, and*
- *Current version is causing client impact*

## Class 3 Scenario:

### **Unsafe Control Execution**

- *SRE provides Rollback when there is client impact, and*
- *Production Service does not receive Rollback*

## Class 4 Scenario:

### **Unsafe Process Behavior**

- *Production Service receives Rollback when there is client impact, and*
- *Production Service does not respond by rolling back current version*

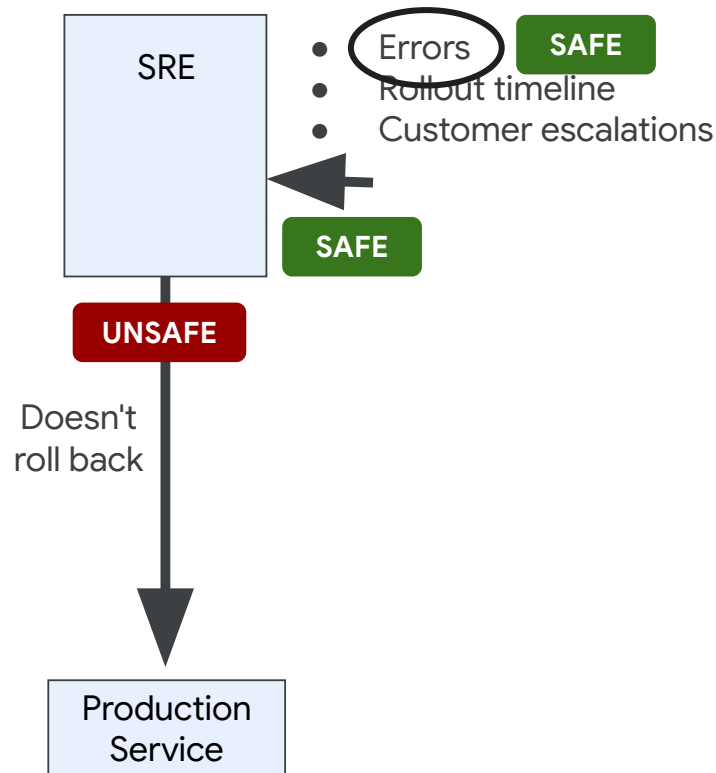
**Answers shown**

# Step 4: Refined Loss scenarios, Class 1

## Class 1: Controller Behavior

Our UCA:

SRE does not provide Rollback when current version is causing client impact

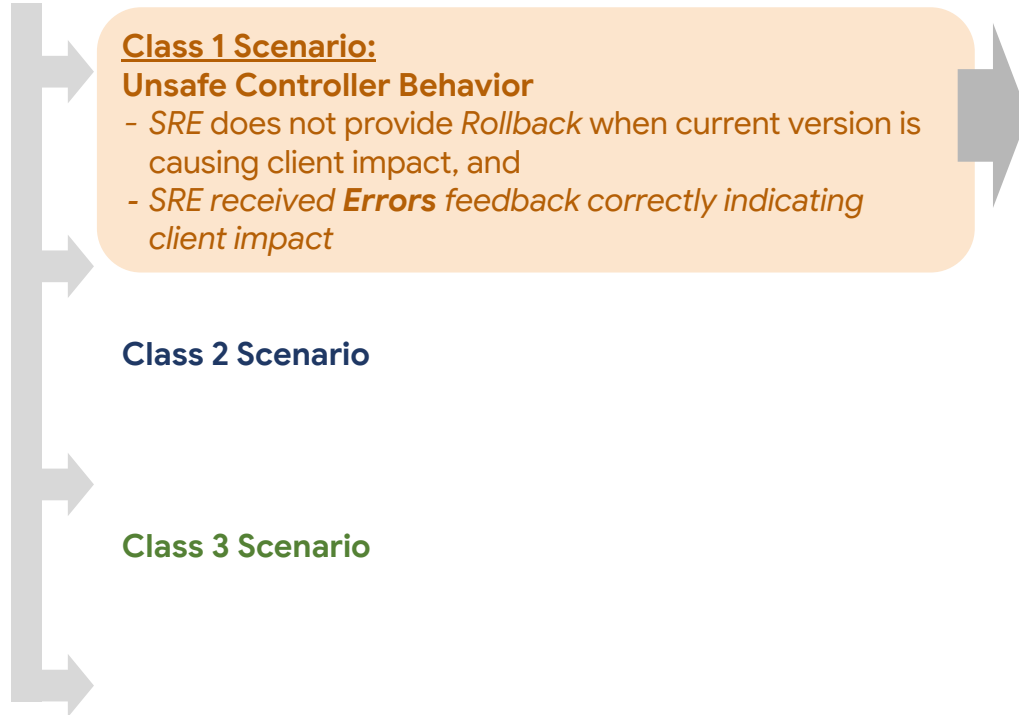


## Step 4: Class 1 formal scenario

- **SRE** does not provide **roll back** when **the current version is causing client impact**
- **SRE** received **errors** feedback correctly indicating **client impact**

# Breakout 4: Unsafe Controller Behavior

UCA: SRE does not provide Rollback when current version is causing client impact



**Class 1 Scenario:**  
**Unsafe Controller Behavior**  
 - SRE does not provide *Rollback* when current version is causing client impact, and  
 - SRE received **Errors** feedback correctly indicating client impact

Class 2 Scenario

Class 3 Scenario

Class 4 Scenario

Examples from past projects  
(not a checklist)

# Refined Scenarios

Responsibilities  
 <Controller> by design is responsible for always assigning \_\_\_ to every \_\_\_ (even if \_\_\_\_.)

Decision Making  
 If \_\_\_, then <Controller> may select <Control Action> to \_\_\_\_

Mental Model  
 PM-1: <Controller> incorrectly believes \_\_\_\_\_ because ...

Interpretation of inputs  
 <Controller> will interpret <feedback/input> as an indicator of \_\_\_, leading to an underestimate.

Controller States / Modes  
 If <Controller> is in \_\_\_ mode/state, it will continue to <Control Action> using alternate input \_\_\_\_.

Other Inputs  
 Although <Input> is correct, the feedback from \_\_\_ may be incorrect and may cause <Controller> to \_\_\_\_.

## Step 4: Class 1 refined scenarios

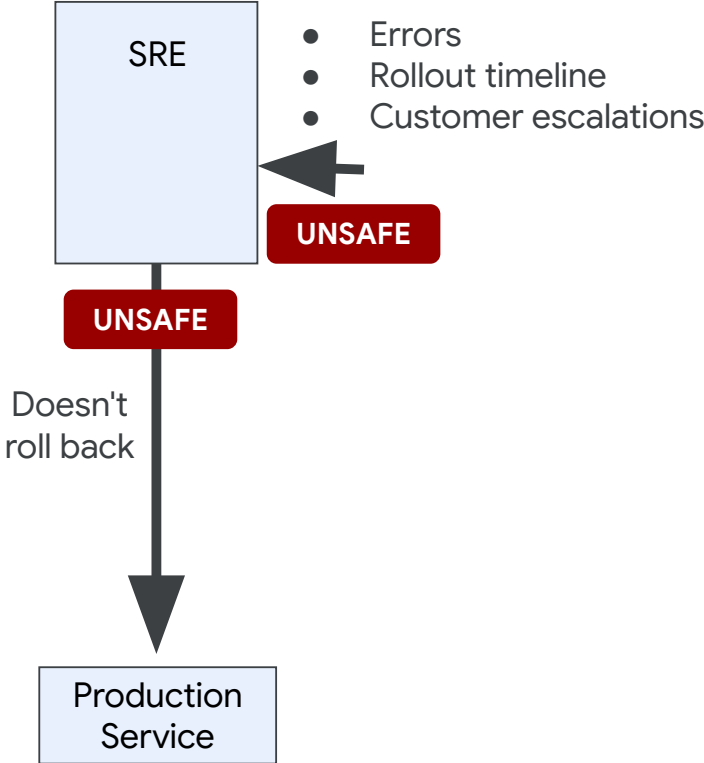
- **SRE does not provide roll back when the current version is causing client impact**
  - Let's pick **Errors** and say that this feedback is correct
- SRE sees errors, but believes current version is not causing client impact because ...
  - There are no customer escalations
  - Engineering culture within the organization favors development velocity over rolling back to mitigate an incident
  - The SRE doesn't have enough confidence in the success of the rollback (e.g. issues with the rollback system tooling itself, whether old version is compatible, whether the rollback will cause additional problems)

# Step 4: Refined Loss scenarios - Class 2

## Class 2: Unsafe Feedback

**Our UCA:**

**SRE does not provide Rollback when current version is causing client impact**



## Step 4: Class 2 formal scenario

- Feedback received by **SRE** does not adequately indicate **there is client impact**.
- **Current version of production service is causing client impact.**

# Breakout 5: Unsafe Feedback

UCA: SRE does not provide Rollback when current version is causing client impact



**Class 1 Scenario:**

**Unsafe Controller Behavior**

- *SRE does not provide Rollback when current version is causing client impact*
- *SRE received feedback correctly indicating client impact*

**Class 2 Scenario:**

**Unsafe Feedback/Information**

- *Feedback received by SRE does not adequately indicate there is client impact, and*
- *Current version is causing client impact*

**Class 3 Scenario**

**Class 4 Scenario**

# Refined Scenarios

**Why? Because:**

- 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.

Examples from past projects (not a checklist)

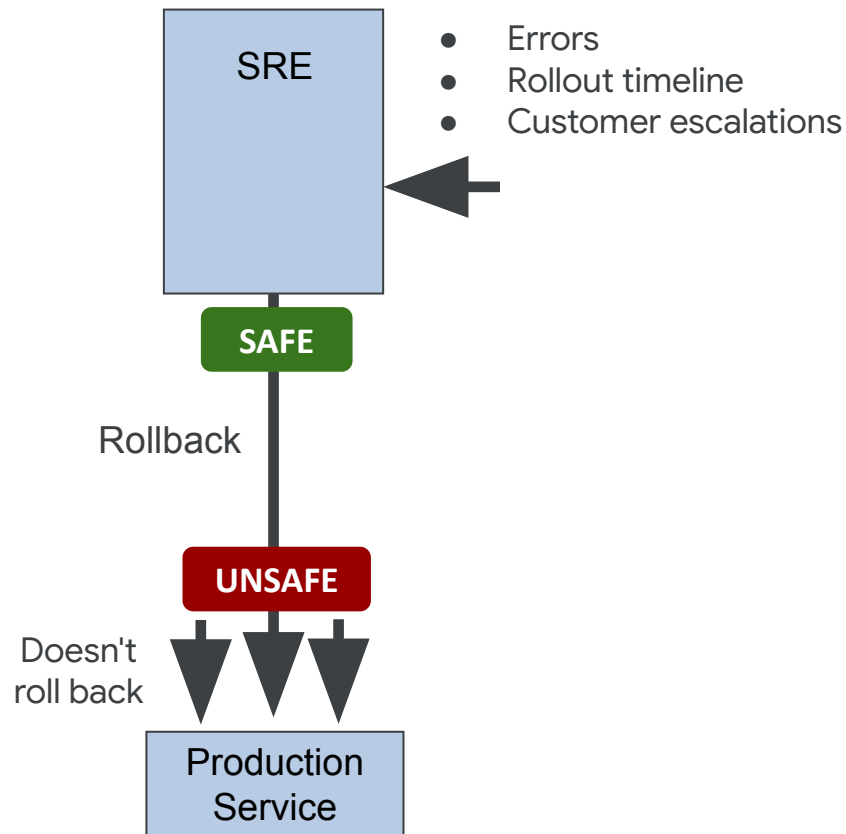
## Step 4: Class 2 refined scenarios

- **SRE does not provide roll back. Feedback to SRE (errors) does not adequately indicate there is client impact.**
  - Monitoring infrastructure is down
  - Monitoring has been incorrectly configured, e.g. monitoring backend responses but not frontend responses
  - Monitoring uses "synthetic traffic" or probes to simulate client requests and check for errors, but the issue impacts other paths not tested by the probes

## Step 4: Refined loss scenarios - Class 3

### Class 3: Control Execution

**SRE does provide Rollback when current version is causing client impact**



## Step 4: Class 3 formal scenario

- **SRE provides Rollback when current version of production service is causing client impact.**
- **However, production service does not receive Rollback when there is impact.**

# Breakout 6: Unsafe Control Execution

UCA: SRE does not provide Rollback when current version is causing client impact

## Class 1 Scenario:

### Unsafe Controller Behavior

- *SRE does not provide Rollback when current version is causing client impact*
- *SRE received feedback correctly indicating client impact*

## Class 2 Scenario:

### Unsafe Feedback/Information

- *Feedback received by SRE does not adequately indicate there is client impact, and*
- *Current version is causing client impact*

## Class 3 Scenario:

### Unsafe Control Execution

- *SRE provides Rollback when there is client impact, and*
- *Production Service does not receive Rollback*

## Class 4 Scenario

# Refined Scenarios

## 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.*

Examples from real projects (not a checklist)

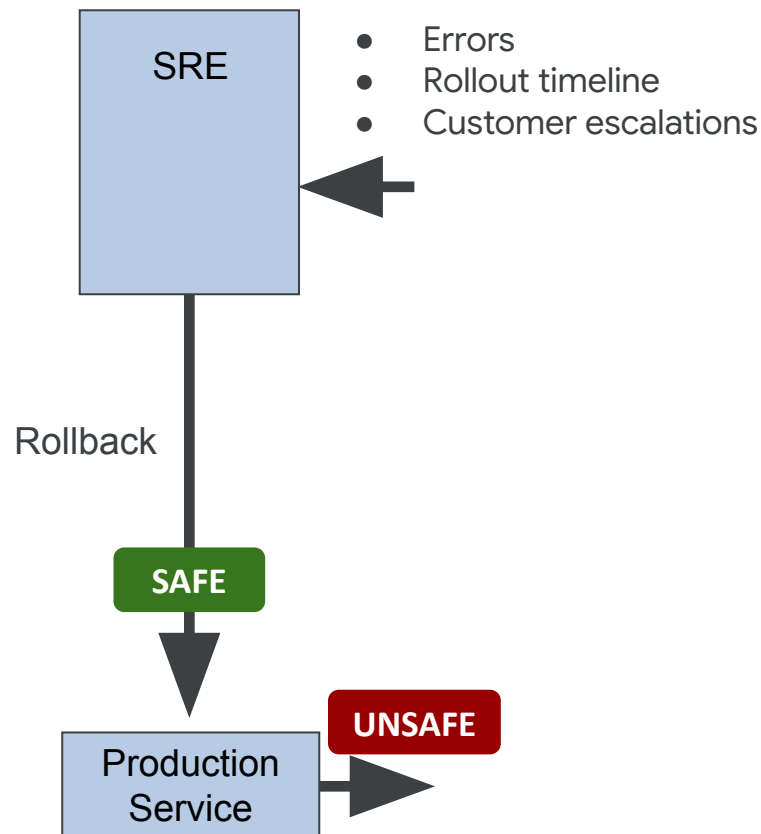
## Step 4: Class 3 refined scenarios

- **SRE *provides* roll back. However, production service *does not receive* roll back.**
  - There is a production freeze in effect, blocking all changes to production
  - There is a deployment gate specifically blocking the previous version
  - More than one rollback is being commanded, and one takes precedence over the other
  - Deployment is down, impacting both rollouts and rollbacks

# Step 4: Refined loss scenarios - Class 4

## Class 4: Controlled Process Behavior

**SRE does provide Rollback when current version is causing client impact**



## Step 4: Class 4 formal scenario

- **Production service** receives **roll back**, when **current version is causing impact**.
- **Production service** does not respond by **rolling back**.

# Breakout 7: Unsafe Process Behavior

UCA: SRE does not provide Rollback when current version is causing client impact

## Class 1 Scenario:

### **Unsafe Controller Behavior**

- *SRE does not provide Rollback when current version is causing client impact*
- *SRE received feedback correctly indicating client impact*

## Class 2 Scenario:

### **Unsafe Feedback/Information**

- *Feedback received by SRE does not adequately indicate there is client impact, and*
- *Current version is causing client impact*

## Class 3 Scenario:

### **Unsafe Control Execution**

- *SRE provides Rollback when there is client impact, and*
- *Production Service does not receive Rollback*

## Class 4 Scenario:

### **Unsafe Process Behavior**

- *Production Service receives Rollback when there is client impact, and*
- *Production Service does not respond by rolling back current version*

# Refined Scenarios

Examples from real projects (not a checklist):

## Why?

- <Process> requires \_\_\_\_, which is unavailable
- <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.

## Step 4: Class 4 refined scenarios

- **SRE *provides* roll back. Production service *receives* roll back, but roll back does not occur.**
  - The SRE does not have permissions to modify production
  - Production deployment requires approval from another engineer
  - Deployment doesn't have enough resources (CPU, RAM) to execute the rollback
  - Rollback requires something that is no longer available, e.g. a specific database version that no longer exists.

## Step 4: Class 4 refined scenarios

- **SRE *provides* roll back. Production service *receives* roll back, but roll back does not occur.**
  - Production service is designed to do sensibility checks before allowing a rollback. If the production state doesn't match what was expected, rollback isn't allowed.
    - This could happen if an SRE has applied a manual emergency fix to production
    - The system detects that the current production state doesn't match its internal record, and assumes it should not overwrite the emergency fix (anti-stomping mode).

# Wrap-up

- STPA is gaining acceptance in the software industry.
  - SRE book 2nd edition will include chapter on STPA & CAST
- STPA takes less time: We found 7 critical system design flaws in 26 engineer hours
  - Design already approved, ready to ship.
  - Without STPA, would have cost us weeks/months (in testing or post-launch)
  - Prevented major outage from impacting all users in a region (0 affected users with STPA)
- STPA finds real cases: Several times, STPA has identified critical flaws that led to a major outage—weeks before it happened!



AI generated image created by Gemini in March 2026

Thank you!

# Appendix



**This incident never happened,  
because we used STPA to find and fix  
the causal design flaws!**

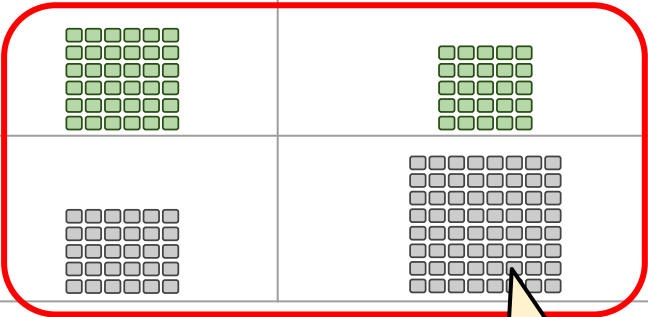


Image source: <https://www.pexels.com/photo/cellphone-on-a-stand-with-navigation-map-application-open-7738879/>



# Cost\* of Defects for Road Disruptions

	Design	Implementation	Testing	Production
When are failure modes <u>introduced</u> ?				
When are failure modes <u>found</u> ?				
Opportunity <u>cost</u> of fixes				

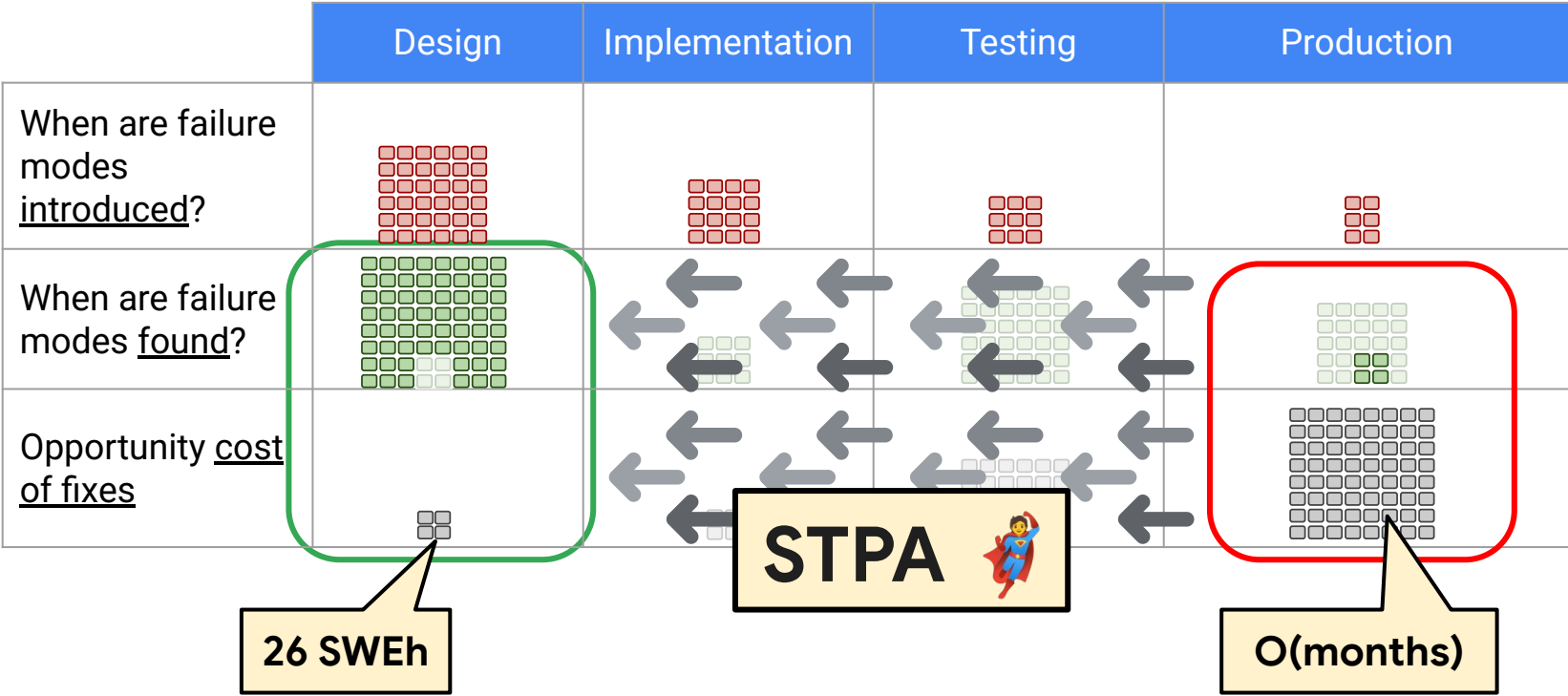


O(months)

\* Coarse characterization, adapted for Google from System Safety and STPA Class Materials, John Thomas, 2021  
Data from "ROI Analysis of the System Architecture Virtual Integration Initiative", SEI, 2018



# Cost\* of Defects for Road Disruptions



\* Coarse characterization, adapted for Google from System Safety and STPA Class Materials, John Thomas, 2021  
Data from "ROI Analysis of the System Architecture Virtual Integration Initiative", SEI, 2018