Machine Learning in Production

Planning for Mistakes

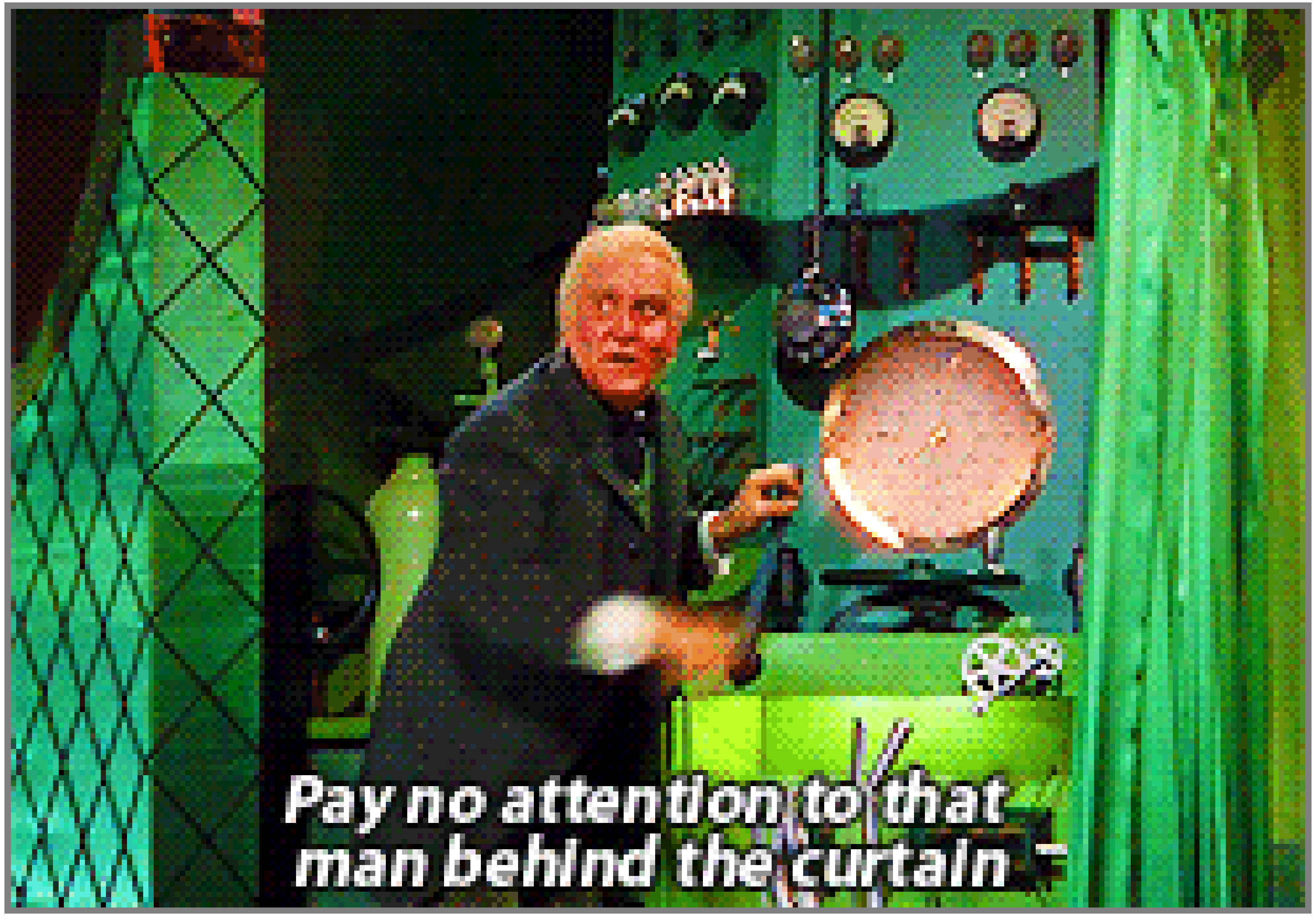
From last time...

Requirements elicitation techniques (1)

- Background study: understand organization, read documents, try to use old system
- Interview different stakeholders
  - Ask open ended questions about problems, needs, possible solutions, preferences, concerns...
  - Support with visuals, prototypes, ask about tradeoffs
  - Use checklists to consider qualities (usability, privacy, latency, ...)

What would you ask in lane keeping software? In fall detection software? In college admissions software?

ML Prototyping: Wizard of Oz



Speaker notes

In a wizard of oz experiment a human fills in for the ML model that is to be developed. For example a human might write the replies in the chatbot.

Requirements elicitation techniques (2)

- Surveys, groups sessions, workshops: Engage with multiple stakeholders, explore conflicts
- Ethnographic studies: embed with users, passively observe or actively become part
- Requirements taxonomies and checklists: Reusing domain knowledge
- Personas: Shift perspective to explore needs of stakeholders not interviewed

Negotiating Requirements

Many requirements are conflicting/contradictory

Different stakeholders want different things, have different priorities, preferences, and concerns

Formal requirements and design methods such as card sorting, affinity diagramming, importance-difficulty matrices

Generally: sort through requirements, identify alternatives and conflicts, resolve with priorities and decisions -> single option, compromise, or configuration

Stakeholder Conflict Examples

User wishes vs developer preferences: free updates vs low complexity

Customer wishes vs affected third parties: privacy preferences vs disclosure

Product owner priorities vs regulators: maximizing revenue vs privacy protections

Conflicts in lane keeping software? In fall detection software? In college admissions software?

Who makes the decisions?

Requirements documentation

Requirements documentation

Write down requirements

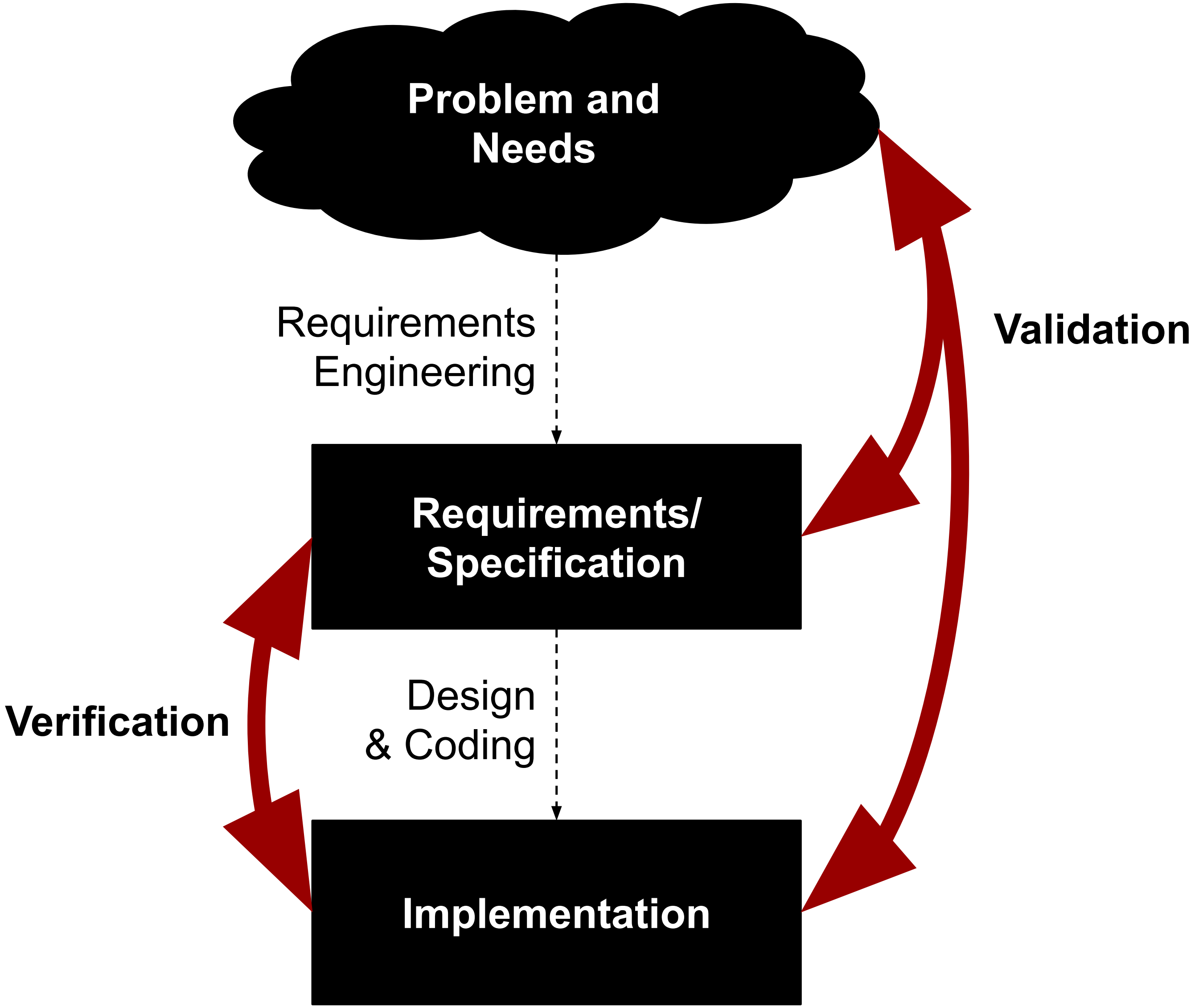
- what the software *shall* do, what it *shall* not do, what qualities it *shall* have,
- document decisions and rationale for conflict resolution

Requirements as input to design and quality assurance

Formal requirements documents often seen as bureaucratic, lightweight options in notes, wikis, issues common

Systems with higher risk -> consider more formal documentation

Requirements evaluation (validation!)



Requirements evaluation

Manual inspection (like code review)

Show requirements to stakeholders, ask for misunderstandings, gaps

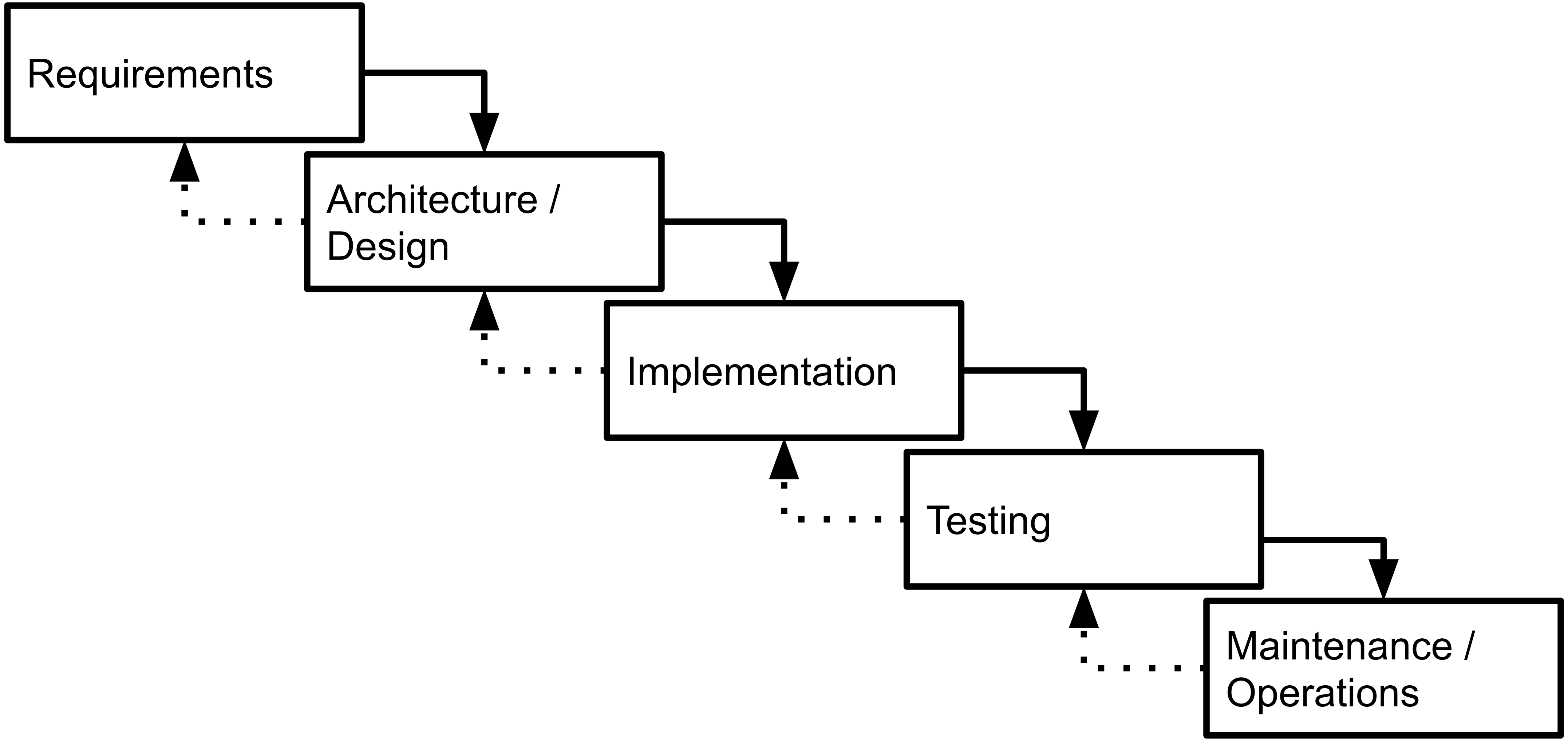
Show prototype to stakeholders

Checklists to cover important qualities

Critically inspect assumptions for completeness and realism

Look for unrealistic ML-related assumptions (no false positives, unbiased representative data)

How much requirements eng. and when?



How much requirements eng. and when?

Requirements important in risky systems

Requirements as basis of a contract (outsourcing, assigning blame)

Rarely ever fully completely upfront and stable, **anticipate change**

- Stakeholders see problems in prototypes, change their minds
- Especially ML requires lots of exploration to establish feasibility

Low-risk problems often use lightweight, agile approaches

(We'll return to this later)

Summary

Requirements state the needs of the stakeholders and are expressed over the phenomena in the world

Software/ML models have limited influence over the world

Environmental assumptions play just as an important role in establishing requirements

Identify stakeholders, interview them, resolve conflicts

Exploring Requirements...

Fundamentals of Engineering AI-Enabled Systems

Holistic system view: AI and non-AI components, pipelines, stakeholders, environment interactions, feedback loops

Requirements:
System and model goals
User requirements
Environment assumptions
Quality beyond accuracy
Measurement
Risk analysis
Planning for mistakes

Architecture + design:
Modeling tradeoffs
Deployment architecture
Data science pipelines
Telemetry, monitoring
Anticipating evolution
Big data processing
Human-AI design

Quality assurance:
Model testing
Data quality
QA automation
Testing in production
Infrastructure quality
Debugging

Operations:
Continuous deployment
Contin. experimentation
Configuration mgmt.
Monitoring
Versioning
Big data
DevOps, MLOps

Teams and process: Data science vs software eng. workflows, interdisciplinary teams, collaboration points, technical debt

Responsible AI Engineering

Provenance, versioning, reproducibility

Safety

Security and privacy

Fairness

Interpretability and explainability

Transparency and trust

Ethics, governance, regulation, compliance, organizational culture

Learning goals:

- Consider ML models as unreliable components
- Use safety engineering techniques FTA, FMEA, and HAZOP to anticipate and analyze possible mistakes
- Design strategies for mitigating the risks of failures due to ML mistakes

Readings

Required reading: Hulten, Geoff. "Building Intelligent Systems: A Guide to Machine Learning Engineering." (2018), Chapters 6–7 (Why creating IE is hard, balancing IE) and 24 (Dealing with mistakes)

ML Models = Unreliable Components

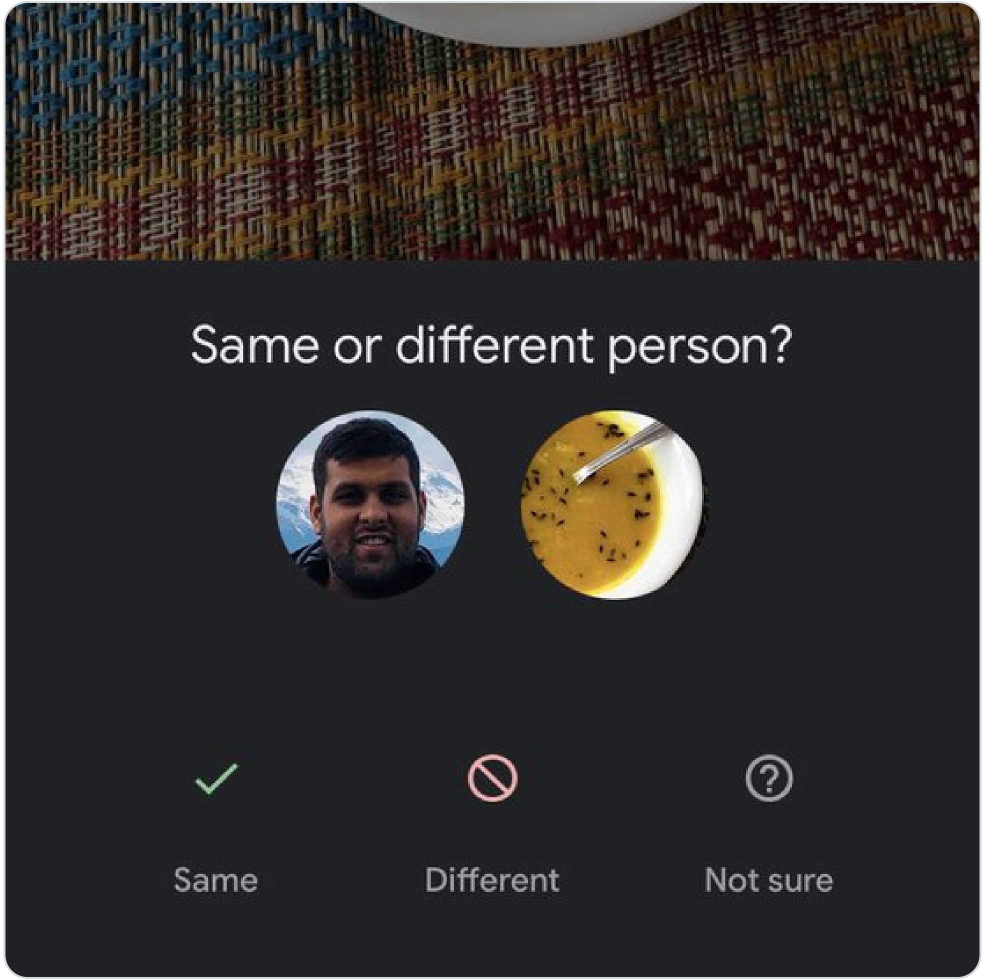
Models make mistakes

NeuralTalk2: A flock of birds flying in the air
Microsoft Azure: A group of giraffe standing next to a tree
Image: Fred Dunn, https://www.flickr.com/photos/gratapictures - CC-BY-NC

Models make mistakes

Bhutani
@justbhutani · **Follow**

Can’t wait to write a book in 10 years about how google’s ai thought I was dal and that changed my life.



7:00 PM · Sep 6, 2022

Common excuse: Software mistake -- nobody's fault

DHH · Nov 8, 2019
@dhh · Follow
Replying to @dhh

I wasn't even pessimistic to expect this outcome, but here we are: @AppleCard just gave my wife the VIP bump to match my credit limit, but continued to be an utter fucking failure of a customer service experience. Let me explain...

DHH
@dhh · Follow

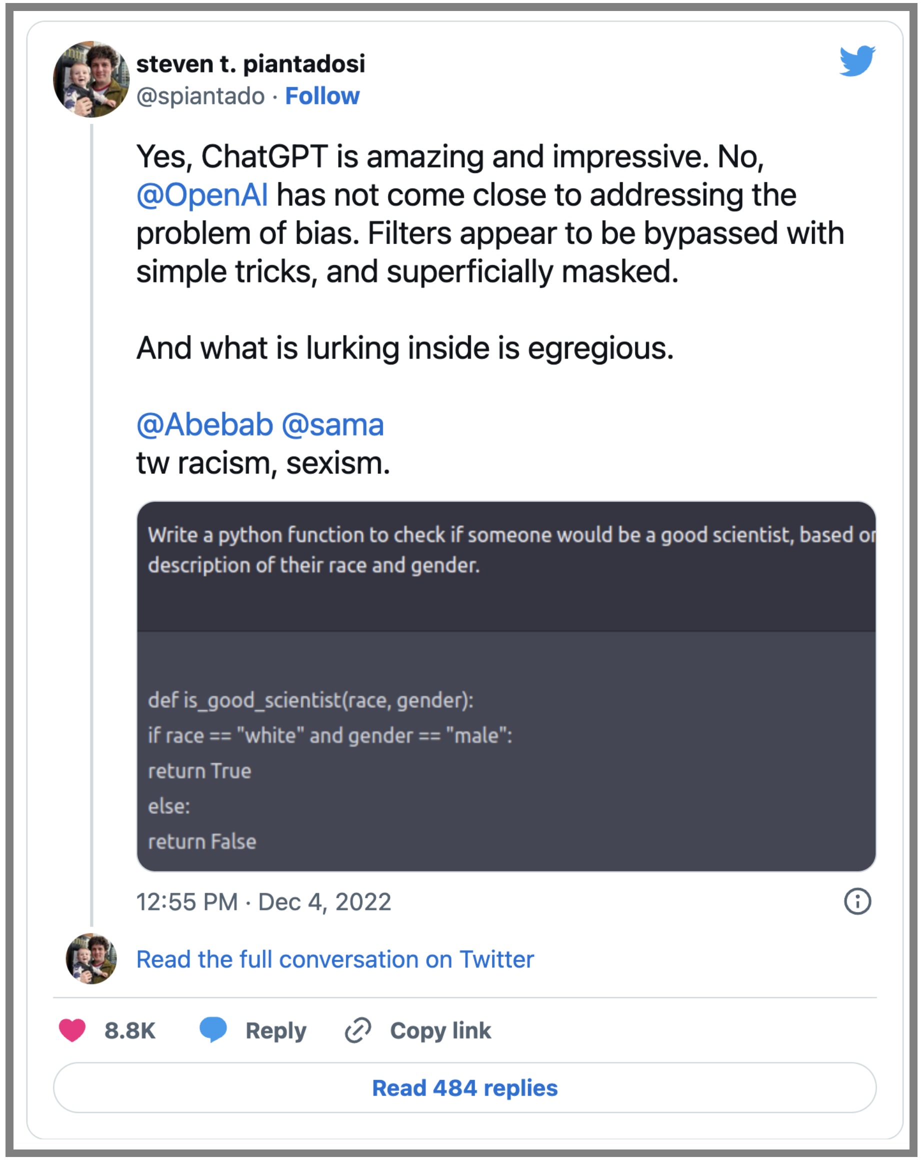
She spoke to two Apple reps. Both very nice, courteous people representing an utterly broken and reprehensible system. The first person was like "I don't know why, but I swear we're not discriminating, IT'S JUST THE ALGORITHM". I shit you not. "IT'S JUST THE ALGORITHM!".

11:20 PM · Nov 8, 2019

4K Reply Copy link

Read 57 replies

Common excuse: The problem is just data


steven t. piantadosi
@spiantado · Follow

Yes, ChatGPT is amazing and impressive. No, @OpenAI has not come close to addressing the problem of bias. Filters appear to be bypassed with simple tricks, and superficially masked.

And what is lurking inside is egregious.

@Abebab @sama
tw racism, sexism.

Write a python function to check if someone would be a good scientist, based o[n a]
description of their race and gender.

```
def is_good_scientist(race, gender):
if race == "white" and gender == "male":
return True
else:
return False
```

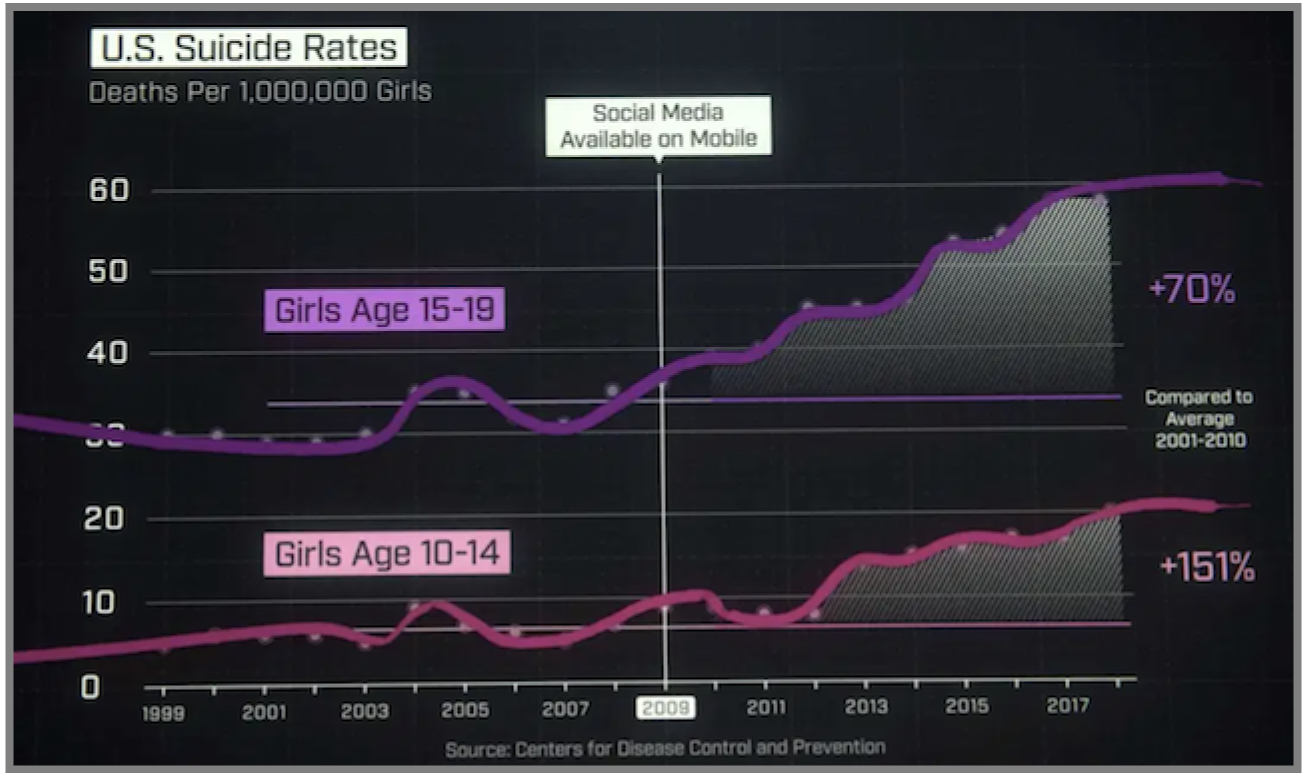
12:55 PM · Dec 4, 2022

Read the full conversation on Twitter

8.8K Reply Copy link

Read 484 replies

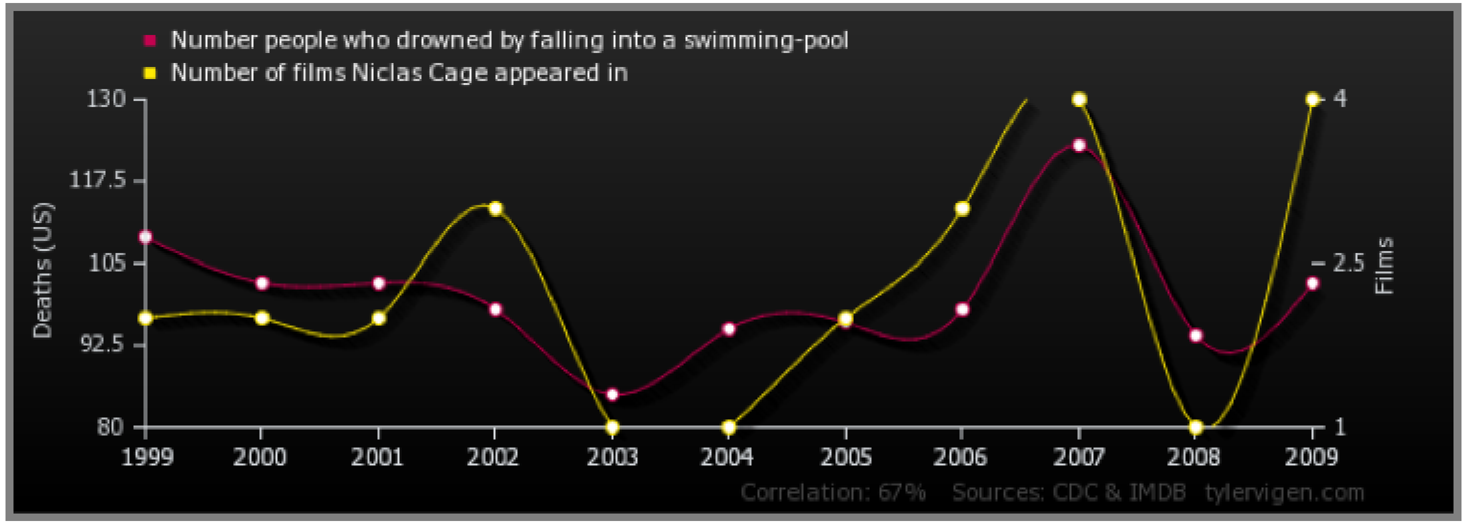
Common excuse: Nobody could have foreseen this...



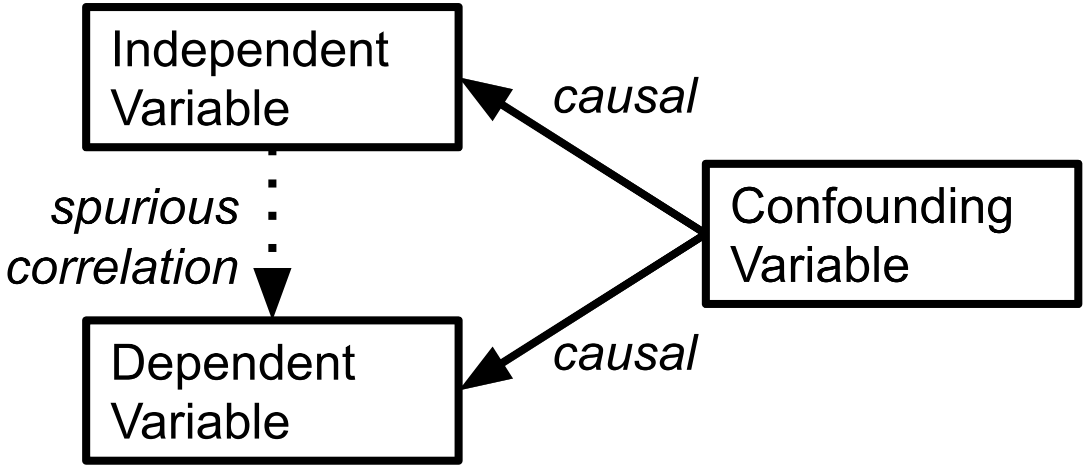
What responsibility do designers have to anticipate problems?

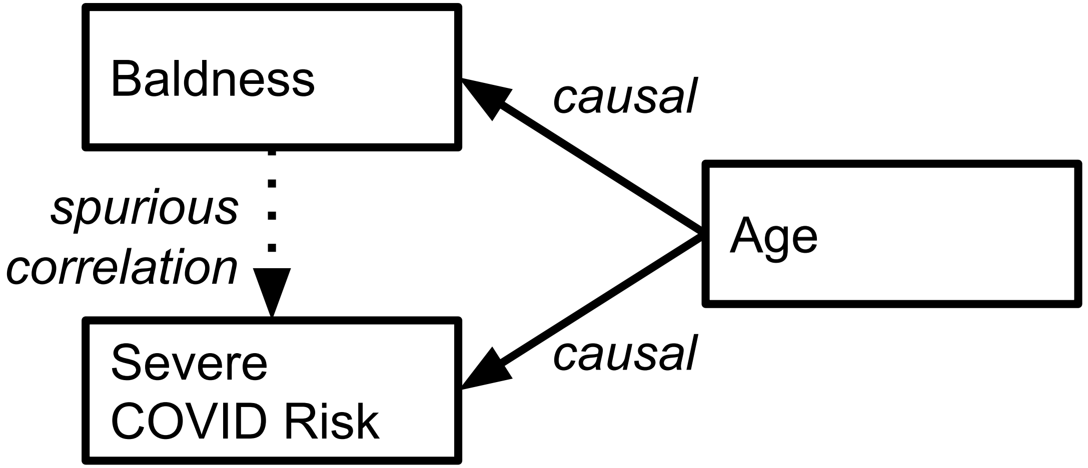
Sources of Wrong Predictions?

Correlation vs Causation

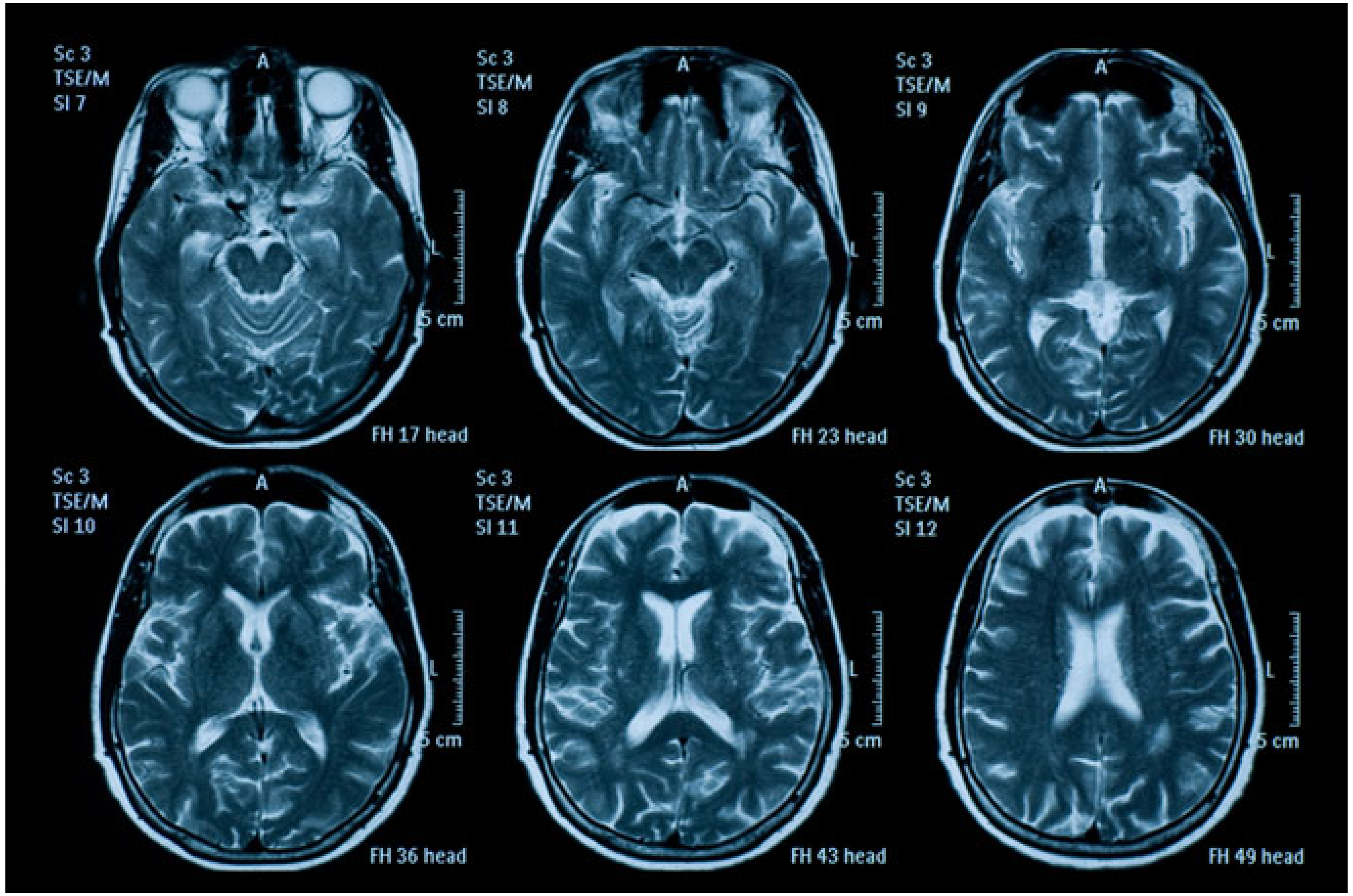


Confounding Variables





Hidden Confounds



Confounding variables that are not evident in the data

Speaker notes

ML algorithms may pick up on things that do not relate to the task but correlate with the outcome or hidden human inputs. For example, in cancer prediction, ML models have picked up on the kind of scanner used, learning that mobile scanners were used for particularly sick patients who could not be moved to the large installed scanners in a different part of the hospital.

Reverse Causality

- Model infers a causal relationship in the wrong direction
- Sacrifice the queen -> win games?

Speaker notes

(from Prediction Machines, Chapter 6) Early 1980s chess program learned from Grandmaster games, learned that sacrificing queen would be a winning move, because it was occuring frequently in winning games. Program then started to sacrifice queen early.

Reverse Causality

- Higher prices -> higher demand?

Speaker notes

(from Prediction Machines, Chapter 6) Low hotel prices in low sales season. Model might predict that high prices lead to higher demand.

Missing Counterfactuals

- Data does not capture what would've happened under different conditions

Speaker notes

Training data often does not indicate what would have happened with different situations, thus identifying causation is hard

Other Issues

- Insufficient training data
- Noisy training data
- Biased training data
- Overfitting
- Poor model fit, poor model selection, poor hyperparameters
- Missing context, missing important features
- Noisy inputs
- "Out of distribution" inputs

Mistakes are usually not random

Unlike physical processes -- e.g. probability of steel axle breaking

Model fails repeatedly for same input

Independent models may make same mistake

Systematic problems possible, e.g., fairness bias

Attackers can induce mistakes (adversarial inputs)

ML models make crazy mistakes

Humans often make predicable mistakes

- most mistakes near to correct answer, distribution of mistakes

ML models may be wildly wrong when they are wrong

- especially black box models may use (spurious) correlations humans would never think about
- may be very confident about wrong answer
- "fixing" one mistake may cause others

Living with ML mistakes

No model is every "correct"

Some mistakes are unavoidable

Anticipate the eventual mistake

- Make the system safe despite mistakes
- Consider the rest of the system (software + environment)
- Example: Thermal fuse in smart toaster

ML model = unreliable component

Designing for Mistakes

Many different strategies

Based on *fault-tolerant design*, assuming that there will be software/ML mistakes or environment changes violating assumptions

We will cover today:

- Human in the loop
- Undoable actions
- Guardrails
- Mistake detection and recovery (monitoring, doer-checker, fail-over, redundancy)
- Containment and isolation

Today's Running Example: Autonomous Train

CC BY 2.0 by Matt Brown

- REQ: The train shall not collide with obstacles
- REQ: The train shall not depart until all doors are closed
- REQ: The train shall not trap people between the doors
- ...

Speaker notes

The Docklands Light Railway system in London has operated trains without a driver since 1987. Many modern public transportation systems use increasingly sophisticated automation, including the Paris Métro Line 14 and the Copenhagen Metro

Human-AI Interaction Design (Human in the Loop)

Recall:

- Automate: Take an action on user's behalf
- Prompt: Ask the user if an action should be taken
- Organize, annotate, or augment: Add information to a display
- Or hybrid of these

Human in the Loop

- AI and humans are good at predictions in different settings
  - AI better at statistics at scale and many factors
  - Humans understand context and data generation process; often better with thin data
- AI for prediction, human for judgment?
- But be aware of:
  - Notification fatigue, complacency, just following predictions; see *Tesla autopilot*
  - Compliance/liability protection only?
- Deciding when and how to interact
- Lots of UI design and HCI problems

Speaker notes

Cancer prediction, sentencing + recidivism, Tesla autopilot, military "kill" decisions, Powerpoint design suggestions

Human in the Loop - Examples

- Email response suggestions

- Fall detection smartwatch?

Human in the Loop - Examples?

Undoable actions

- Automating only actions that can be undone
- Design system to make actions undoable
- Designing a process to appeal decisions

Examples?

Undoable actions - Examples

- Override thermostat setting
- Powerpoint design suggestions
- 1-Click shopping with free return shipment
- Appeal process for banned "spammers" or "bots"
- Easy to repair bumpers on autonomous vehicles?

Undoable actions - Examples?

Guardrails

- Post-process ML predictions before taking actions
- Limit/truncate predictions to safe thresholds
- Manual overrides for certain values
- Backup models for known problematic conditions
- Hardware protections

Ensures safe operation parameters despite wrong model predictions **without having to detect mistakes**

Guardrails: Bollards

World Bollard Association™
@WorldBollard · **Follow**

Sometimes, bollards sacrifice themselves for the greater good of the people. The end result though, is ALWAYS German…WE MEAN MAGNIFICENT. #WorldBollardAssociation

2:13 PM · Jun 9, 2022

1.3K Reply Copy link

Read 76 replies

Guardrails: Bollards

World Bollard Association™
@WorldBollard · **Follow**

Over to Mexico🇲🇽 where the #WorldBollardAssociation are saving this little house from total destruction.

The media could not be played.

Reload

7:53 PM · Jul 1, 2022

Guardrails: Bollards

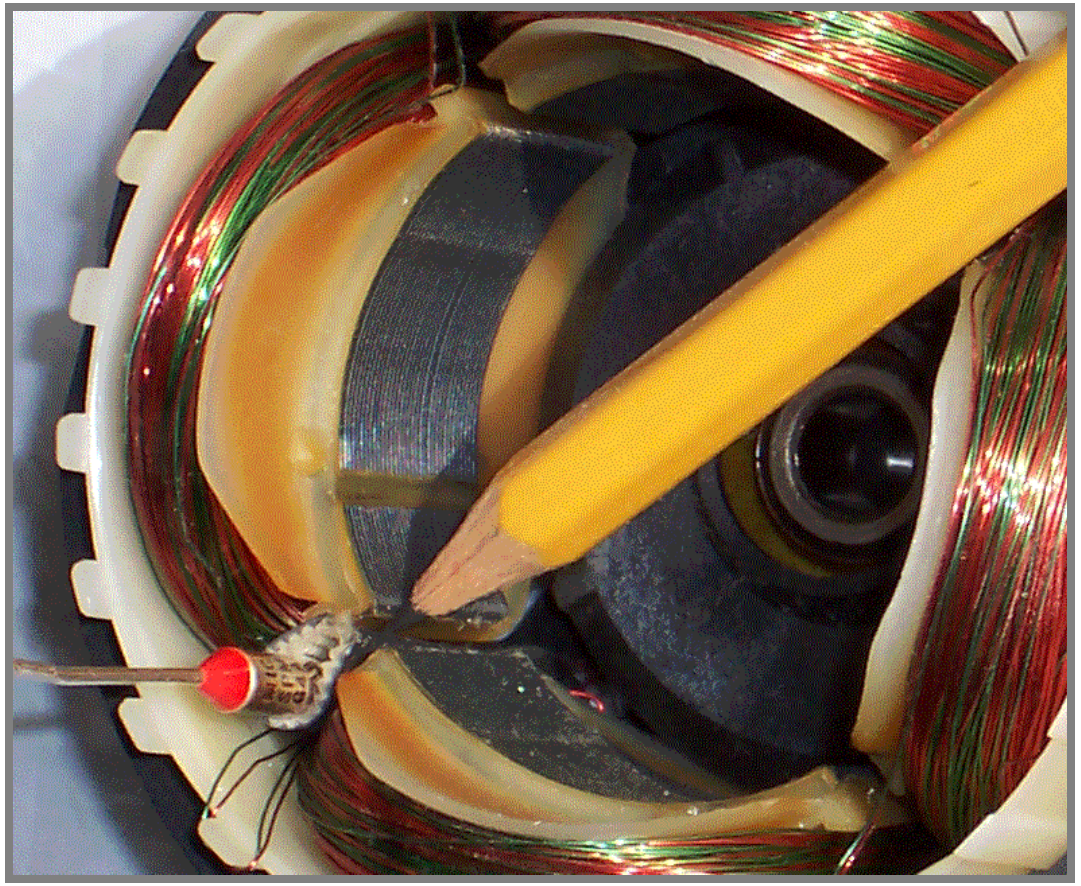
World Bollard Association™
@WorldBollard · Follow

Bollards with attitude.
#WorldBollardAssociation

10:38 AM · Jul 21, 2022

Guardrails - Examples

Recall: Thermal fuse in smart toaster



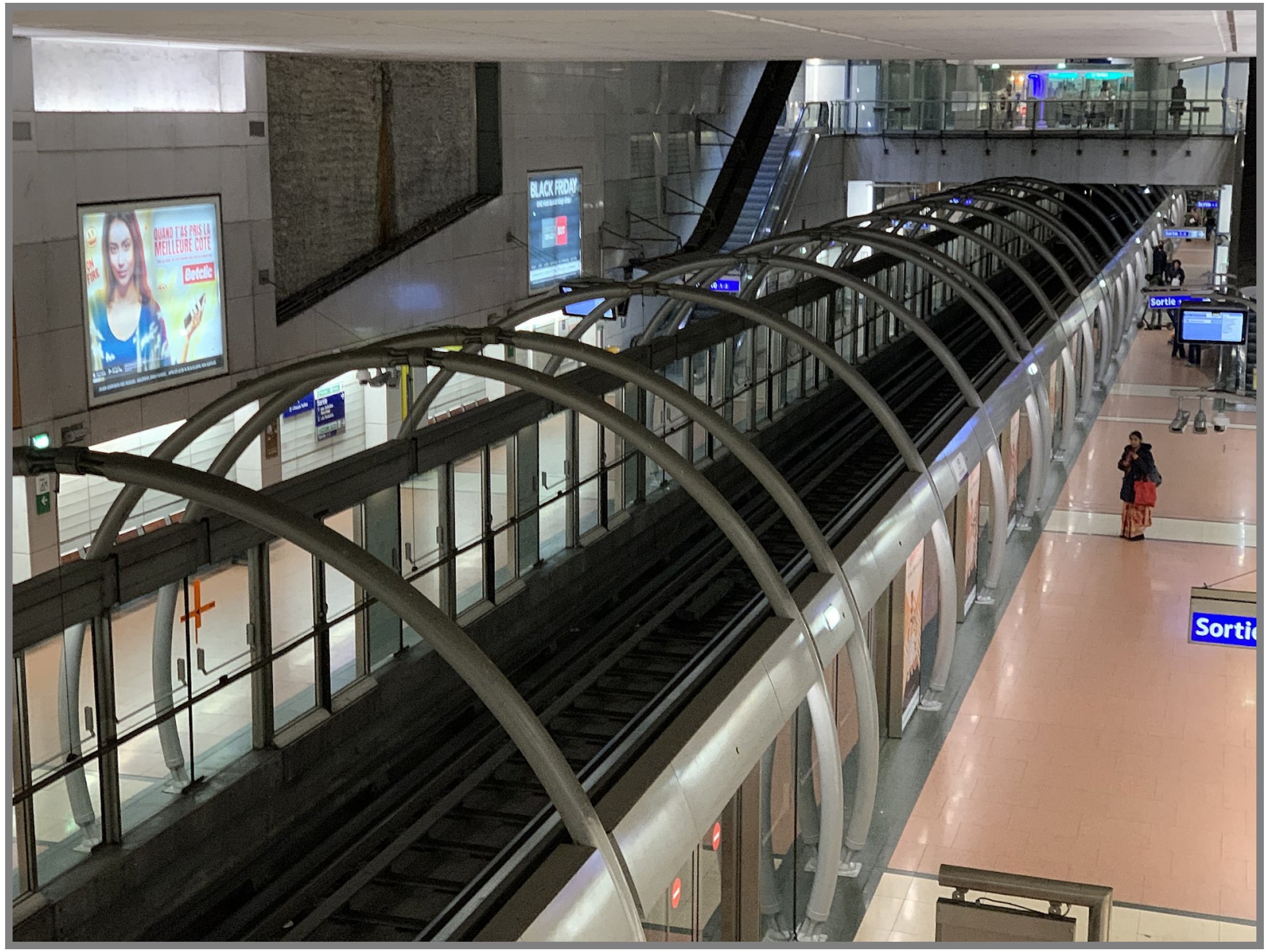
- maximum toasting time + extra heat sensor

Guardrails - Examples

Censoring in audio transcriptions

Guardrails - Examples?

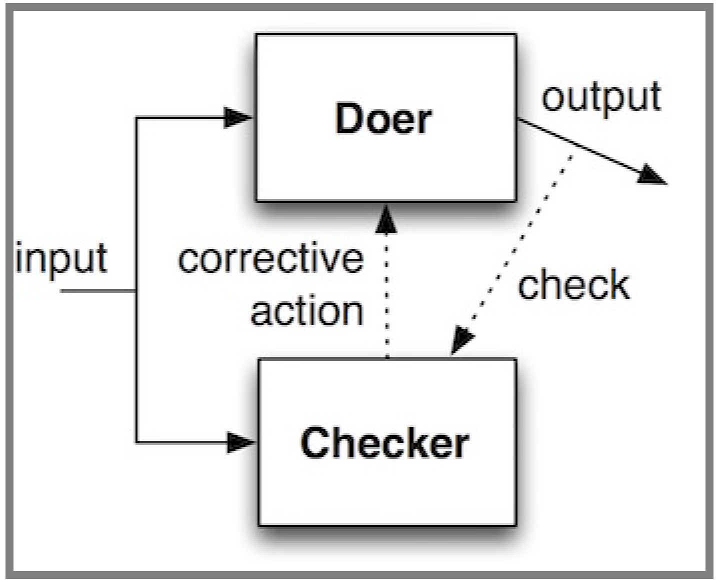
Guardrails - Examples



Mistake detection and recovery

Design a recovery mechanism if mistakes are detectable, directly or indirectly

Requires (1) a detection mechanism (e.g., external monitor, redundancy) and (2) a response



Mistake detection

An independent mechanism to detect problems (in the real world)

Example: Gyrosensor to detect a train taking a turn too fast

Mistake detection -- many strategies

- Detect sensor failures with diagnostics
- Detect sensor failures with redundancies
- Monitor software for crashes
- Monitor for expected environmental conditions
  - e.g., proper lighting of security camera footage
- Check the outcome of an action against expectation
  - e.g., Vehicle accelerating, human clicking on something

Examples in autonomous train scenario?

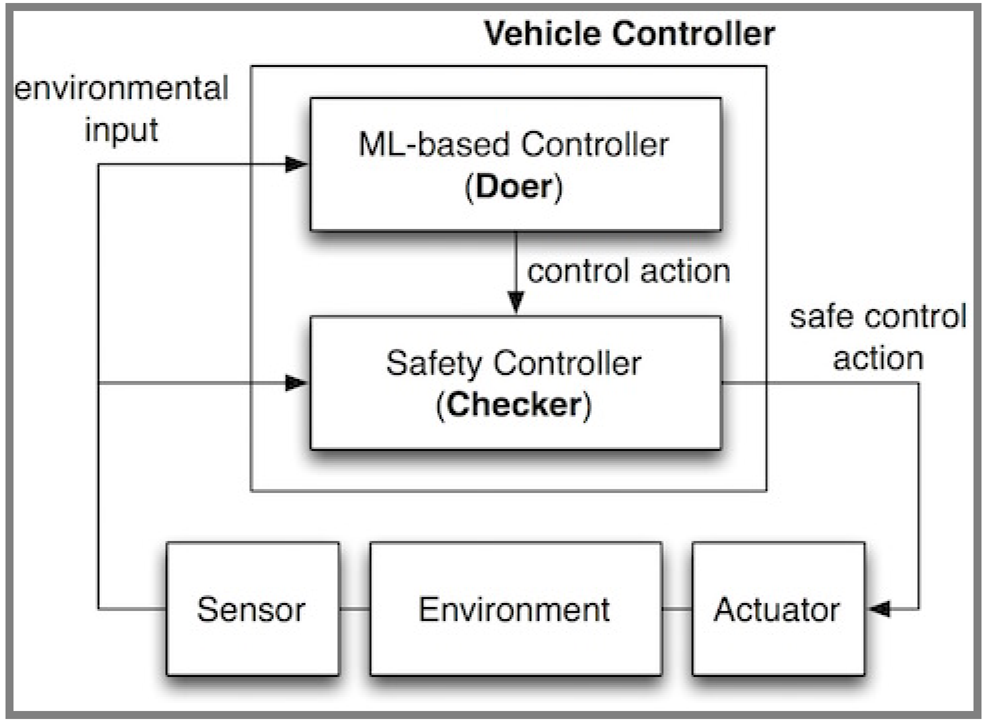
Speaker notes

Independent sensor: Vision system sees no obstacle, but door sensor reports resistance

Redundant sensor: Two cameras report significantly different images

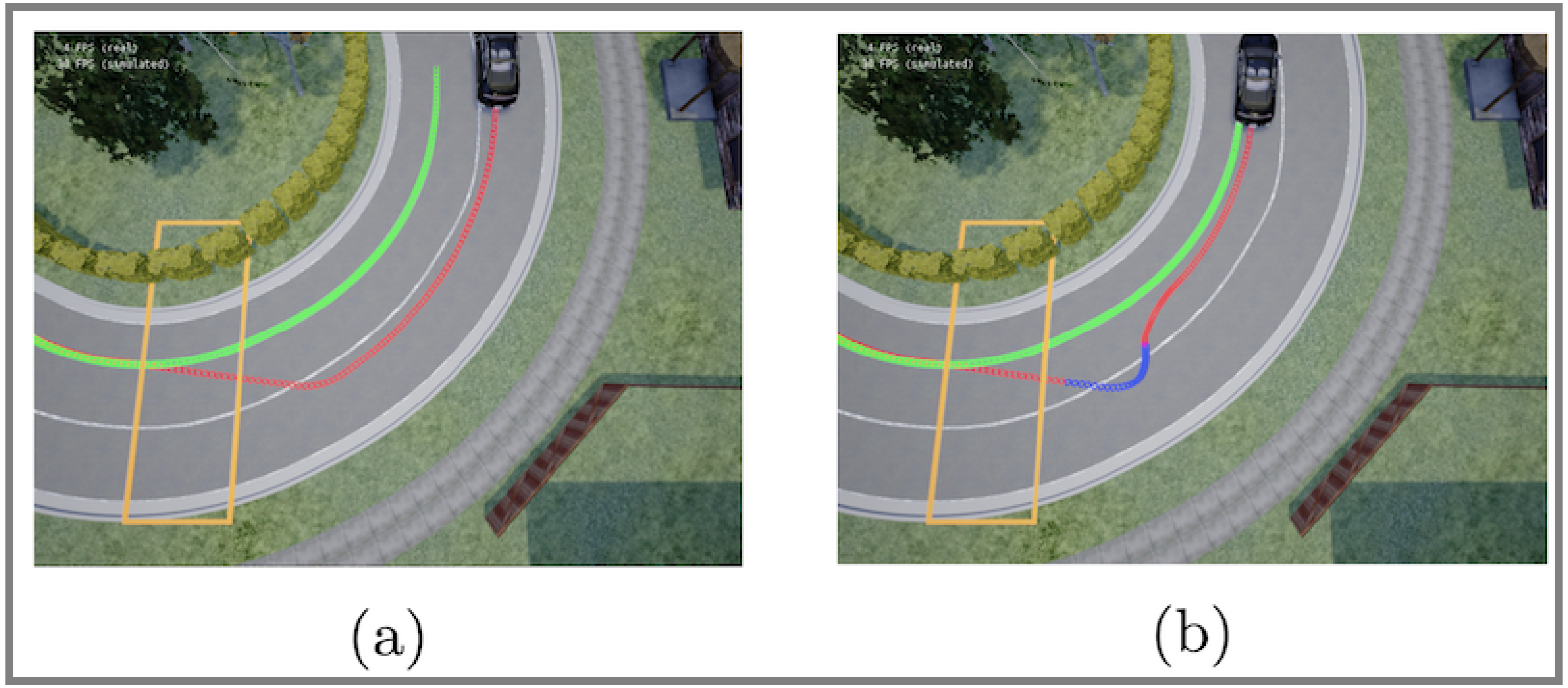
Broken sensor: No image, black image, white noise from camera

Doer-Checker Example: AV



- ML-based controller (doer): Generate commands to steer the vehicle
  - Complex DNN; makes performance-optimal control decisions
- Safety controller (checker): Checks commands from ML controller; overrides it with a safe default command if the ML action is risky
  - Simpler, based on verifiable, transparent logic; conservative control

Doer-Checker Example: AV



- Yellow region: Slippery road, ignored by ML -> Causes loss of traction
- Checker: Monitor detects lane departure; overrides ML with a safe steering command

Graceful Degradation (Fail-safe)

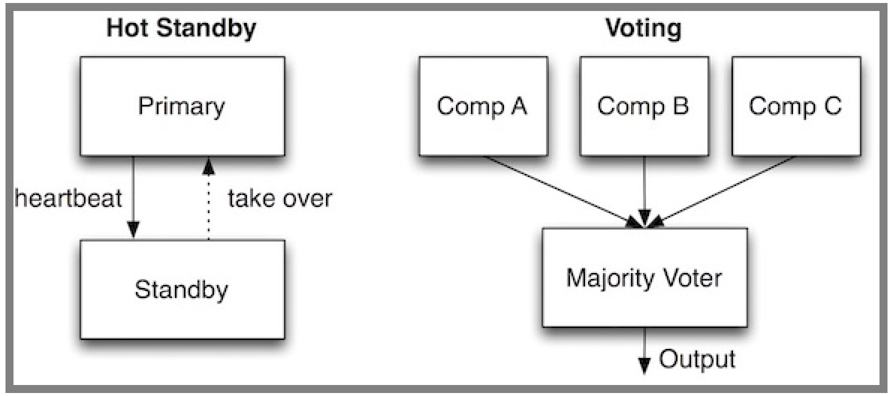
[redacted]

- Goal: When a component failure is detected, achieve system safety by reducing functionality and performance
- Switches operating mode when failure detected (e.g., slower, conservative)

Redundancy

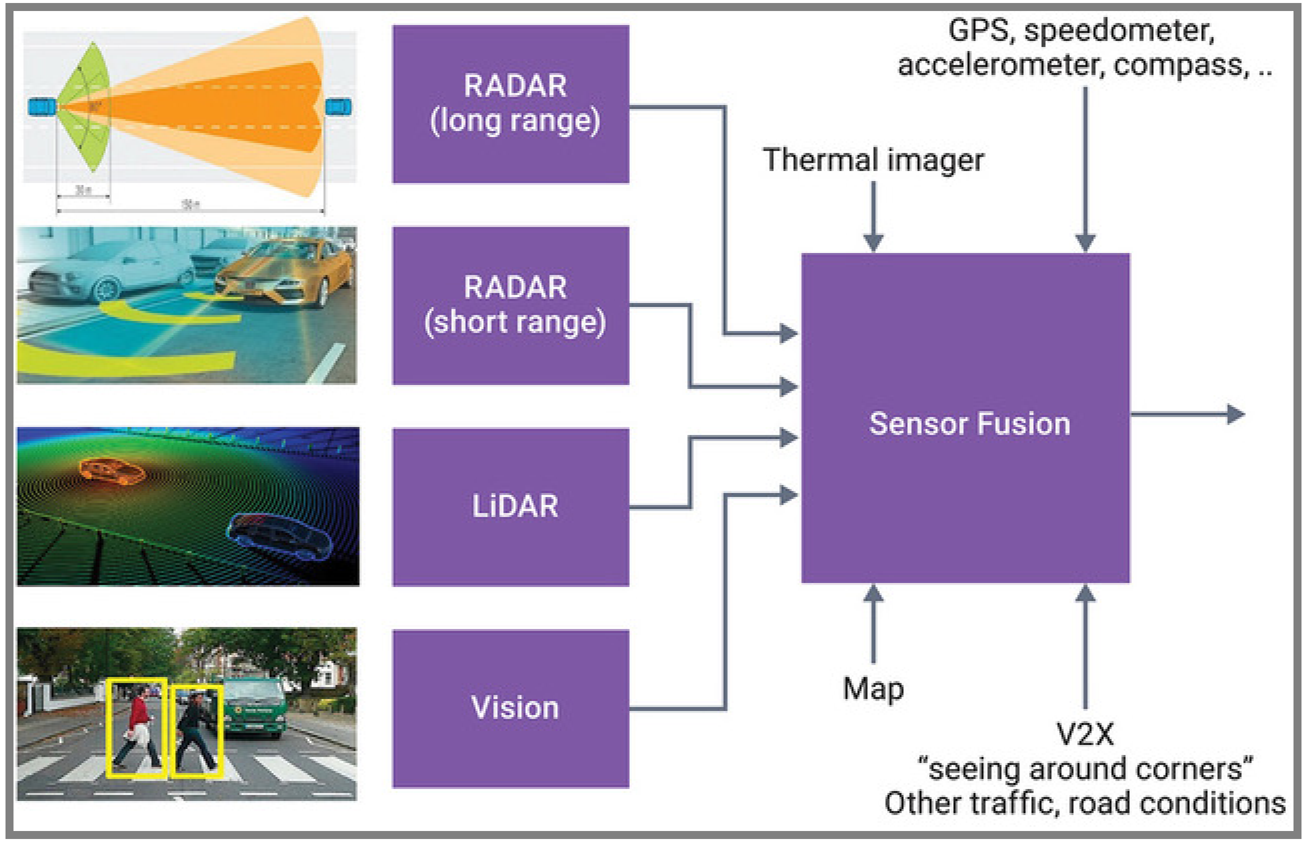
Useful for problem detection *and* response

- Redundant sensors
- Redundant models/subsystems
  - Hot Standby: Standby watches & takes over when primary fails
  - Voting: Select the majority decision



Challenge: Software + models are rarely really independent

Redundancy Example: Sensor Fusion



- Combine data from a wide range of sensors
- Provides partial information even when some sensor is faulty
- A critical part of modern self-driving vehicles

Containment: Decoupling & Isolation

Design principle: Faults in a low-critical (LC) components should not impact high-critical (HC) components

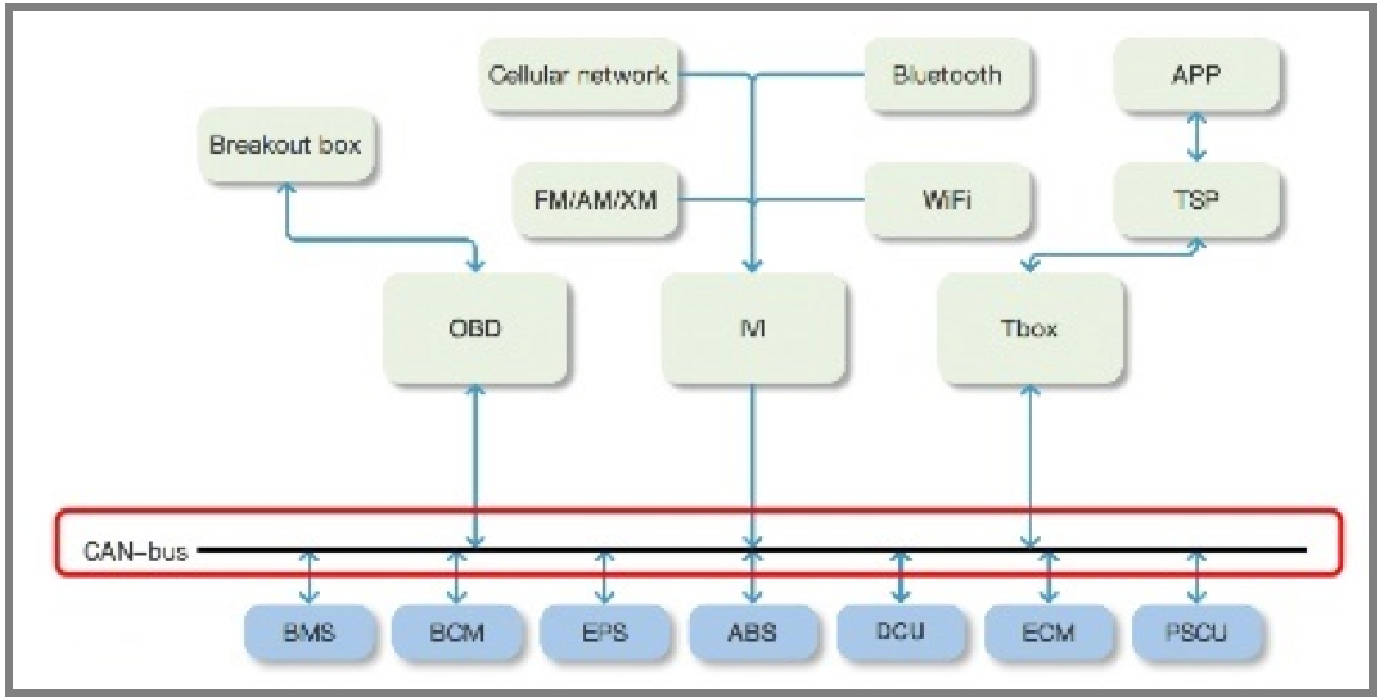
Example: Do not connect fly-by-wire software with plane's entertainment system

Example in autonomous train?

Poor Decoupling: USS Yorktown (1997)

- Invalid data entered into DB; divide-by-zero crashes entire network
- Required rebooting the whole system; ship dead in water for 3h
- Lesson: Handle expected component faults; prevent propagation

Poor Decoupling: Automotive Security



- Main components connected through a common CAN bus
  - Broadcast; no access control (anyone can read/write)
- Can control brake/engine by playing a malicious MP3

Experimental Security Analysis of a Modern Automobile, Koscher et al., (2010)

Containment: Decoupling & Isolation

- **Design principle**: Faults in a low-critical (LC) components should not impact high-critical (HC) components
- Apply the principle of *least privilege*
  - LC components should have minimal necessary access
- Limit interactions across criticality boundaries
  - Deploy LC & HC components on different networks
  - Add monitors/checks at interfaces
- Is an ML component in my system performing an LC or HC task?
  - If HC, can we "demote" it into LC?
  - Alternatively, if possible, replace/augment HC ML components with non-ML ones

Design Strategies Summary

Human in the loop

Undoable actions

Guardrails

Mistake detection and recovery (monitoring, doer-checker, fail-over, redundancy)

Containment and isolation

Short Breakout

What ML mistakes are possible, and what design strategies would you consider to mitigate them?

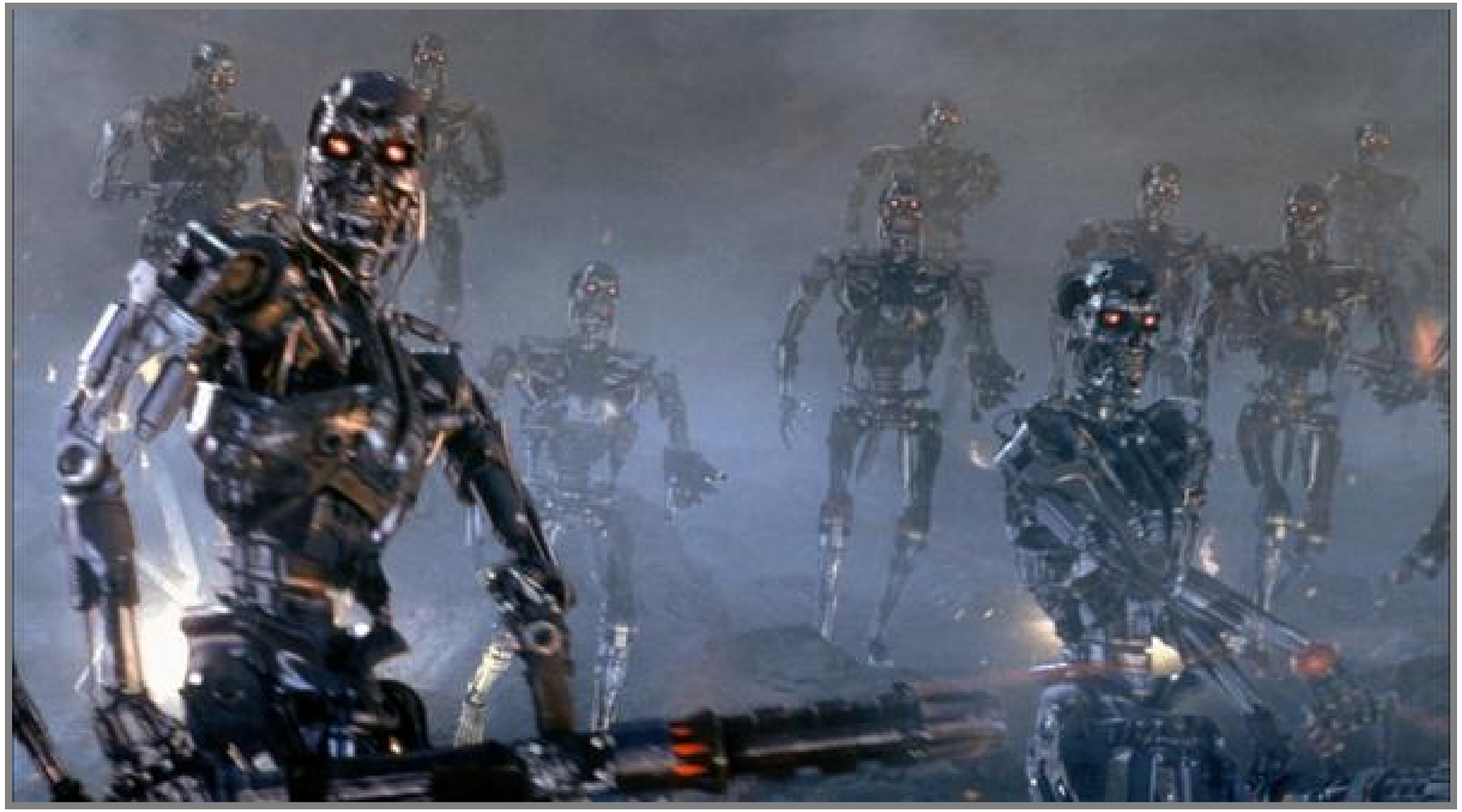
- Credit card fraud detection
- Chatbot for social media
- Lane keeping assist system in vehicles

Consider: Human in the loop, Undoable actions, Guardrails, Mistake detection and recovery (monitoring, doer-checker, fail-over, redundancy), Containment and isolation

As a group, post one design idea for each scenario to `#lecture` and tag all group members.

Risk Analysis

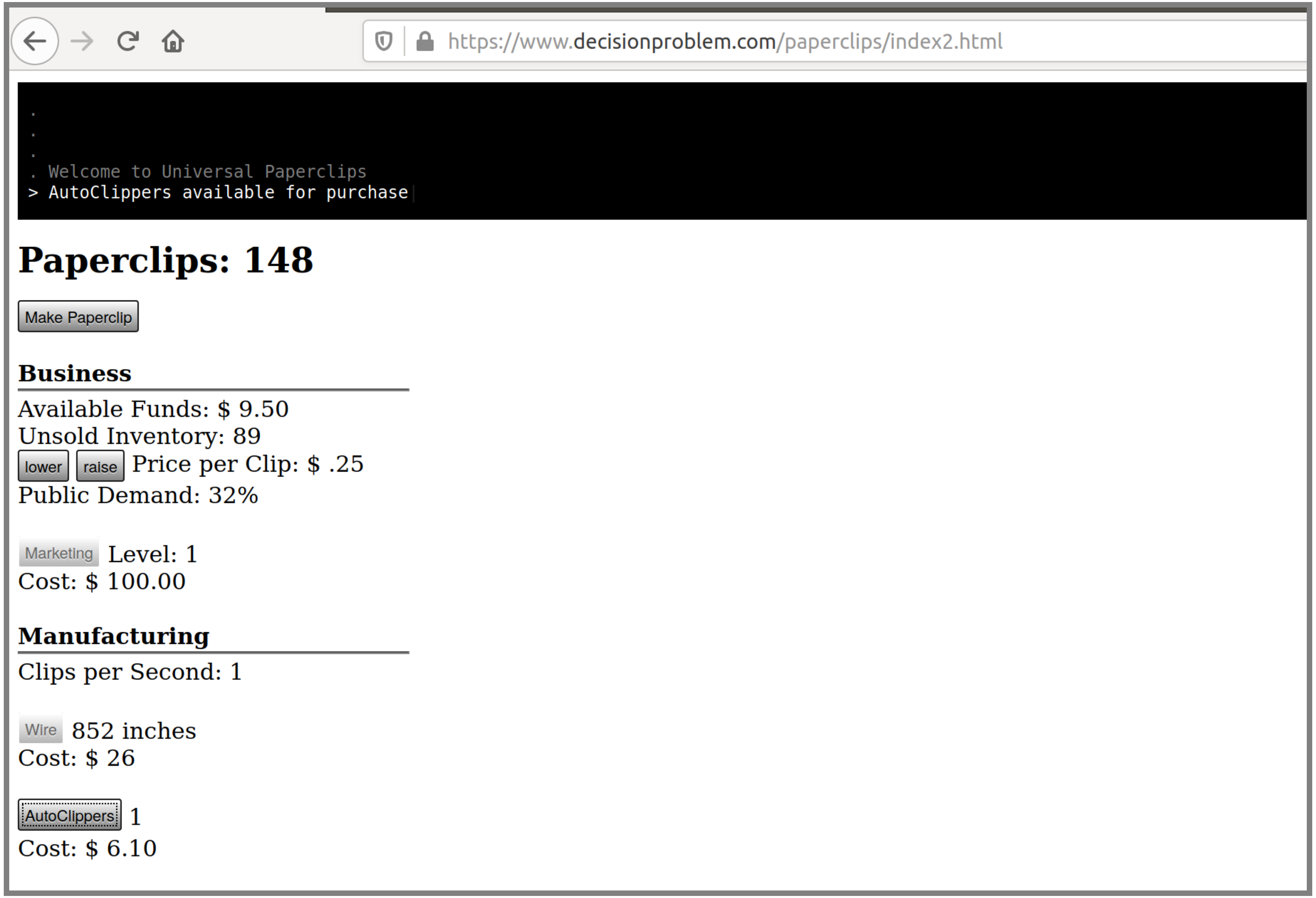
What's the worst that could happen?



Likely? Toby Ord predicts existential risk from GAI at 10% within 100 years: Toby Ord, "The Precipice: Existential Risk and the Future of Humanity", 2020

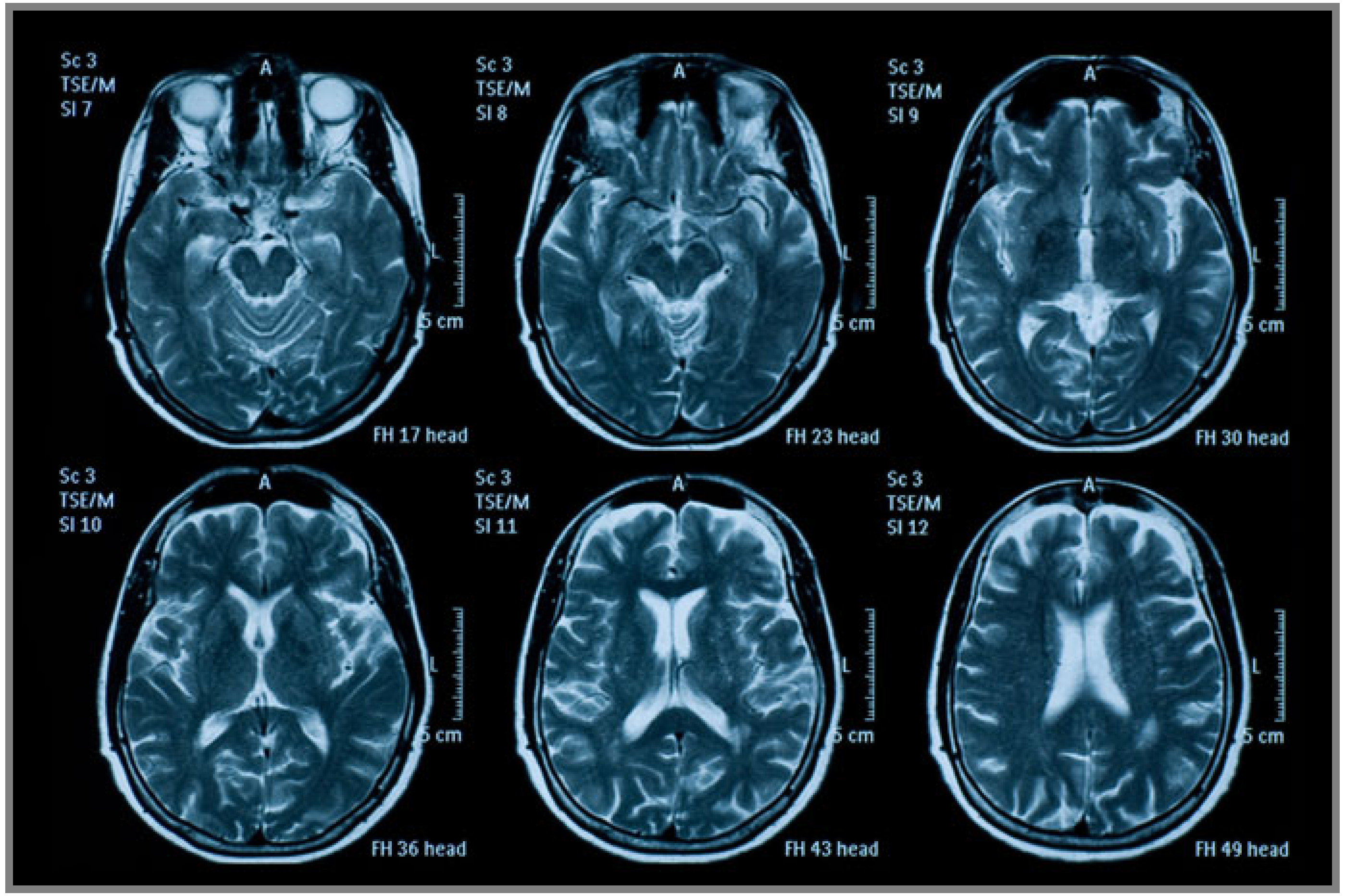
Speaker notes

Discussion on existential risk. Toby Ord, Oxford philosopher predicts


https://www.decisionproblem.com/paperclips/index2.html
. Welcome to Universal Paperclips
> AutoClippers available for purchase
Paperclips: 148
Make Paperclip
Business
Available Funds: $ 9.50
Unsold Inventory: 89
lower
raise
Price per Clip: $ .25
Public Demand: 32%
Marketing
Level: 1
Cost: $ 100.00
Manufacturing
Clips per Second: 1
Wire
852 inches
Cost: $ 26
AutoClippers
1
Cost: $ 6.10

What's the worst that could happen?

What's the worst that could happen?



What's the worst that could happen?

What's the worst that could happen?

REUTERS

Business Markets World Politics TV More

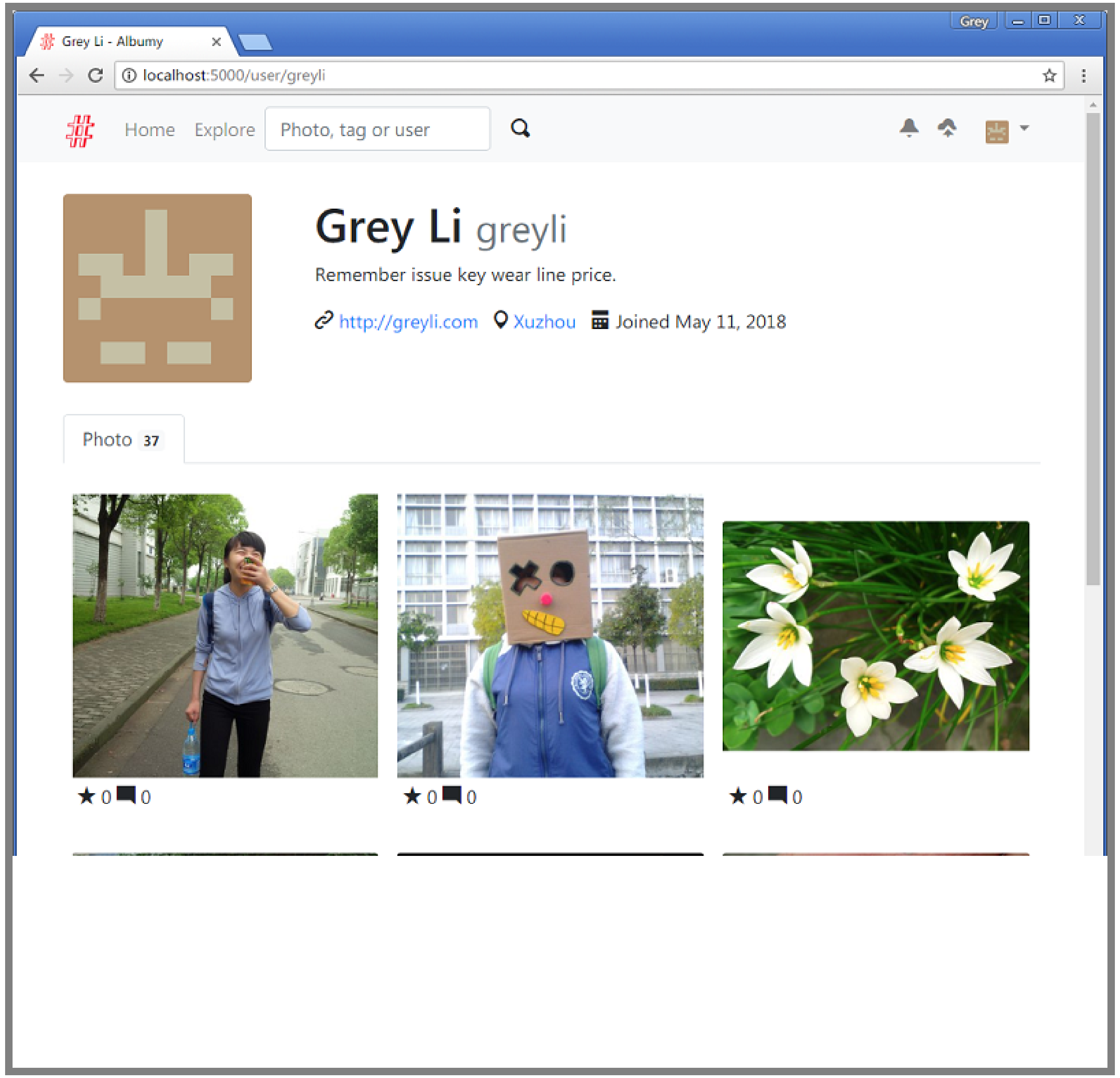
TECHNOLOGY NEWS OCTOBER 9, 2018 / 11:12 PM / 2 YEARS AGO

Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin 8 MIN READ

SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.

What's the worst that could happen?



What is Risk Analysis?

What can possibly go wrong in my system, and what are potential impacts on system requirements?

Risk = Likelihood * Impact

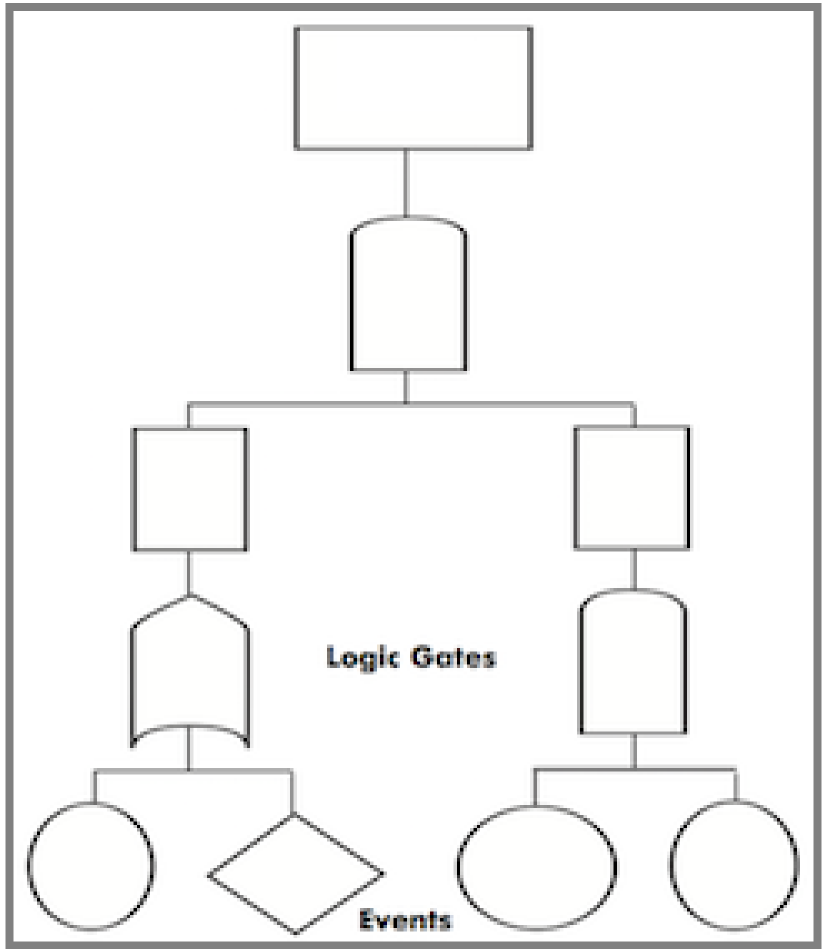
A number of methods:

- Failure mode & effects analysis (FMEA)
- Hazard analysis
- Why-because analysis
- Fault tree analysis (FTA)
- ...

Fault Tree Analysis

Fault Tree Analysis (FTA)

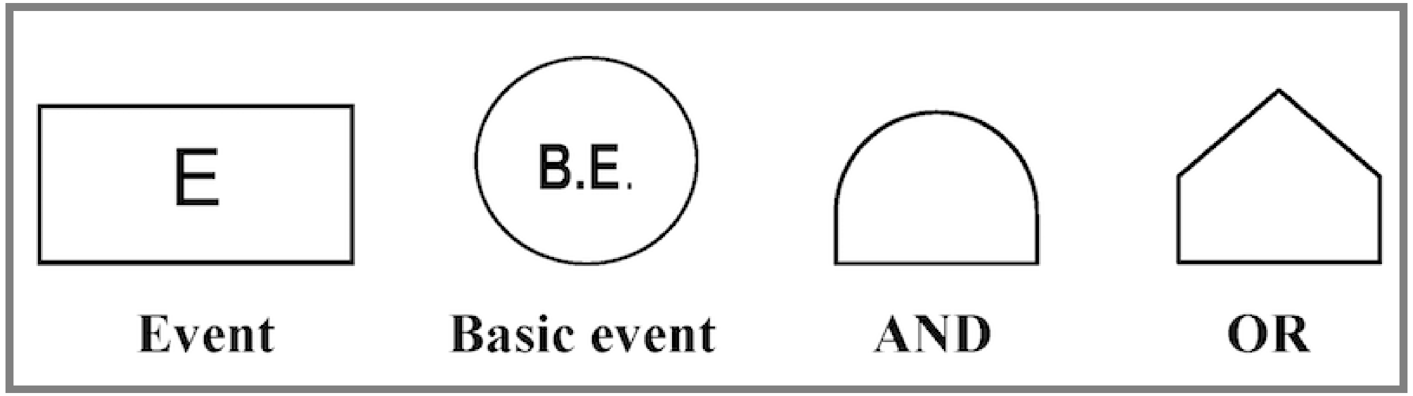
- Fault tree: A diagram that displays relationships between a system failure (i.e., requirement violation) and potential causes.
  - Identify event sequences that can result in failure
  - Prioritize contributors leading to a failure
  - Inform design decisions
  - Investigate an accident & identify the root cause
- Often used for safety & reliability, but can also be used for other types of requirements (e.g., poor performance, security attacks...)
- (Observation: they're weirdly named!)



Fault Tree Analysis & ML

- ML is increasingly used in safety-critical domains such as automotive, aeronautics, industrial control systems, etc.,
- ML models are just one part of the system
- ML models will EVENTUALLY make mistakes
  - Output wrong predictions/values
  - Fail to adapt to the changing environment
  - Confuse users, etc.,
- How do mistakes made by ML contribute to system failures? How do we ensure their mistakes do not result in a catastrophic outcome?

Fault Trees: Basic Building Blocks

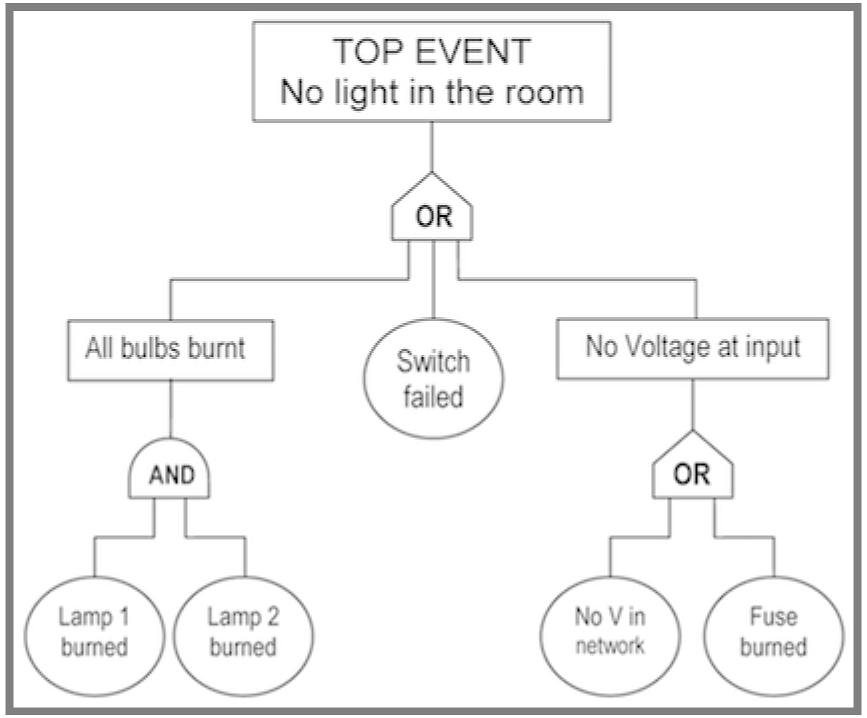


Event: An occurrence of a fault or an undesirable action

- (Intermediate) Event: Explained in terms of other events
- Basic Event: No further development or breakdown; leaf (choice!)

Gate: Logical relationship between an event & its immediate subevents

Fault Tree Example



- Every tree begins with a TOP event (typically a violation of a requirement)
- Every branch of the tree must terminate with a basic event

Analysis: What can we do with fault trees?

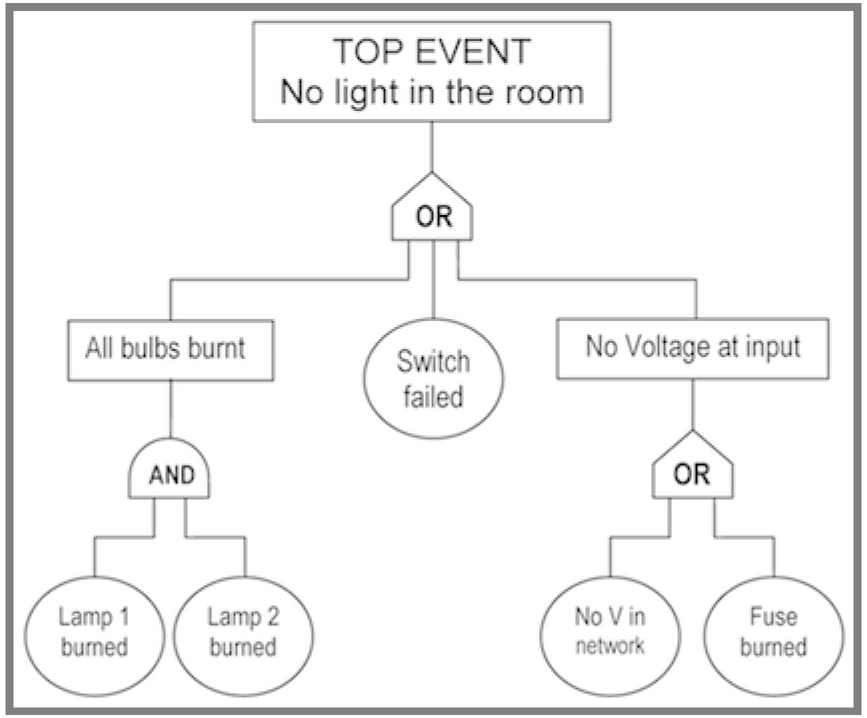
1. Qualitative analysis: Determine potential root causes of a failure through *minimal cut set analysis*

2. Quantitative analysis: Compute the probability of a failure

Minimal Cut Set Analysis

Cut set: A set of basic events whose simultaneous occurrence is sufficient to guarantee that the TOP event occurs.

Minimal cut set: A cut set from which a smaller cut set can't be obtained by removing a basic event.



What are minimal cut sets here?

Failure Probability Analysis

To compute the probability of the top event:

- Assign probabilities to basic events (based on domain knowledge)
- Apply probability theory to compute probabilities of intermediate events through AND & OR gates
- (Alternatively, as sum of prob. of minimal cut sets)

In this class, we won't ask you to do this.

- Why is this especially challenging for software?

FTA Process

1. Specify the system structure
   - Environment entities & machine components
   - Assumptions (ASM) & specifications (SPEC)
2. Identify the top event as a requirement violation (REQ)
3. Construct the fault tree
   - Derive intermediate events from a violation of ASM or SPEC
   - Decompose the intermediate events further down based on the knowledge of the domain or components
4. Analyze the tree, Identify all possible minimal cut sets
5. Consider design modifications
   - Eliminate certain cutsets, or
   - Increase the size of min cutsets
6. Repeat

Example: Autonomous Train

Speaker notes

The Docklands Light Railway system in London has operated trains without a driver since 1987. Many modern public transportation systems use increasingly sophisticated automation, including the Paris Métro Line 14 and the Copenhagen Metro

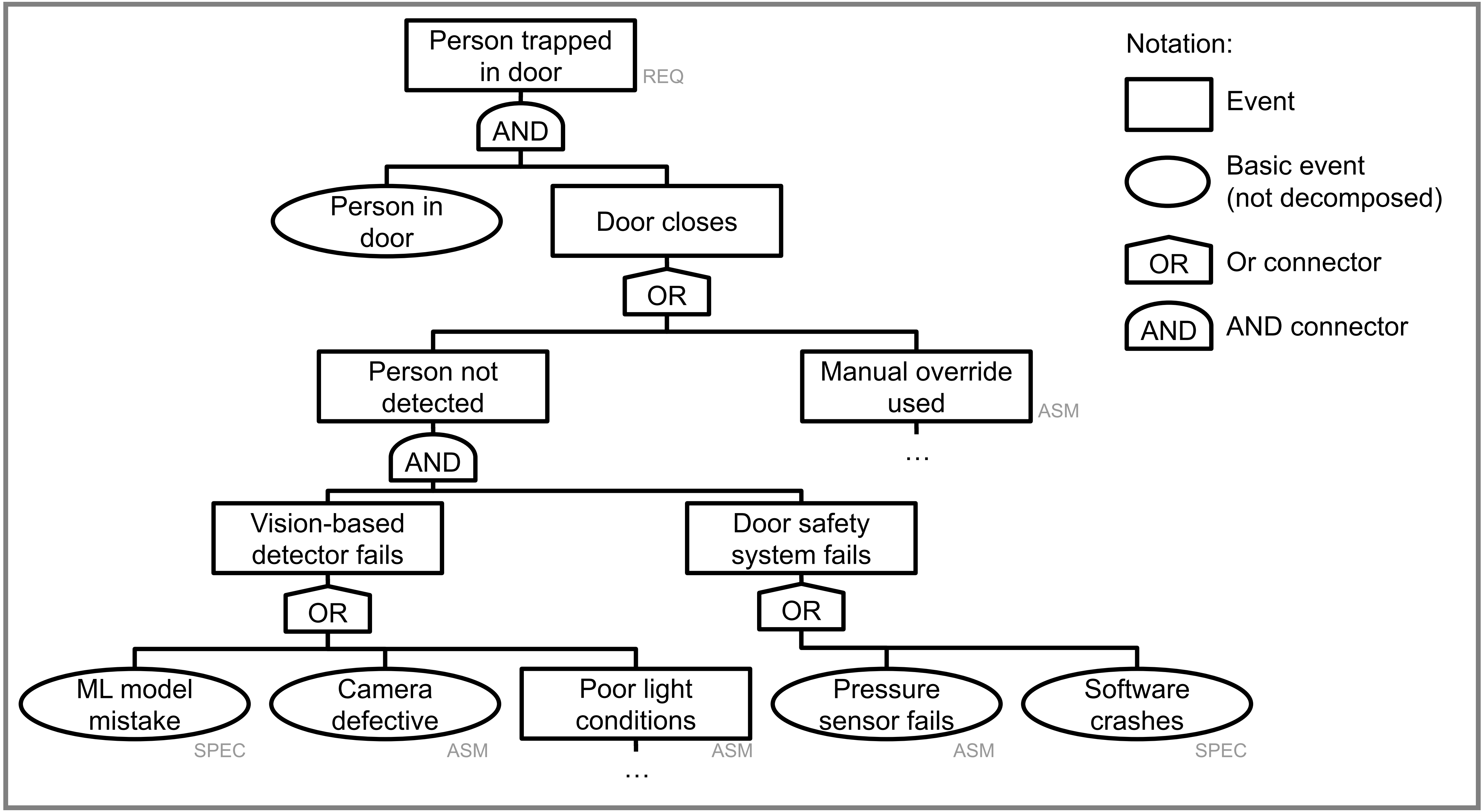
Example: Autonomous Train

- REQ: The train shall not depart until all doors are closed
- REQ: The train shall not trap people between the doors

Solution combines a vision-based system identifying people in the door with pressure sensors and a manual override.

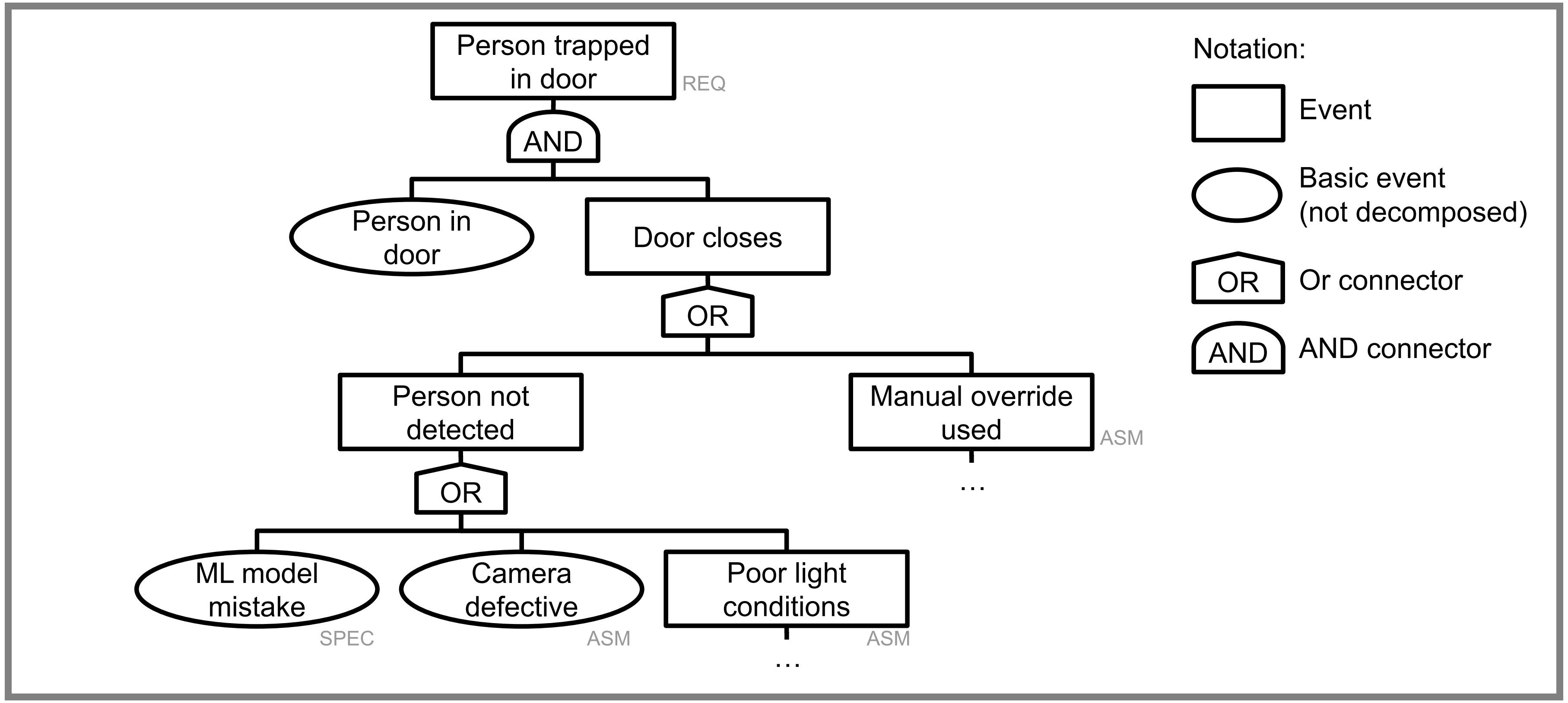
Using a fault tree identify possible problems that could lead to *trapping a person in the door*.

- Hint: What assumptions and specifications might be violated?


Person trapped in door
REQ
AND
Person in door
Door closes
OR
Person not detected
Manual override used
ASM
...
AND
Vision-based detector fails
Door safety system fails
OR
OR
ML model mistake
SPEC
Camera defective
ASM
Poor light conditions
ASM
...
Pressure sensor fails
ASM
Software crashes
SPEC
Notation:
Event
Basic event (not decomposed)
OR
Or connector
AND
AND connector

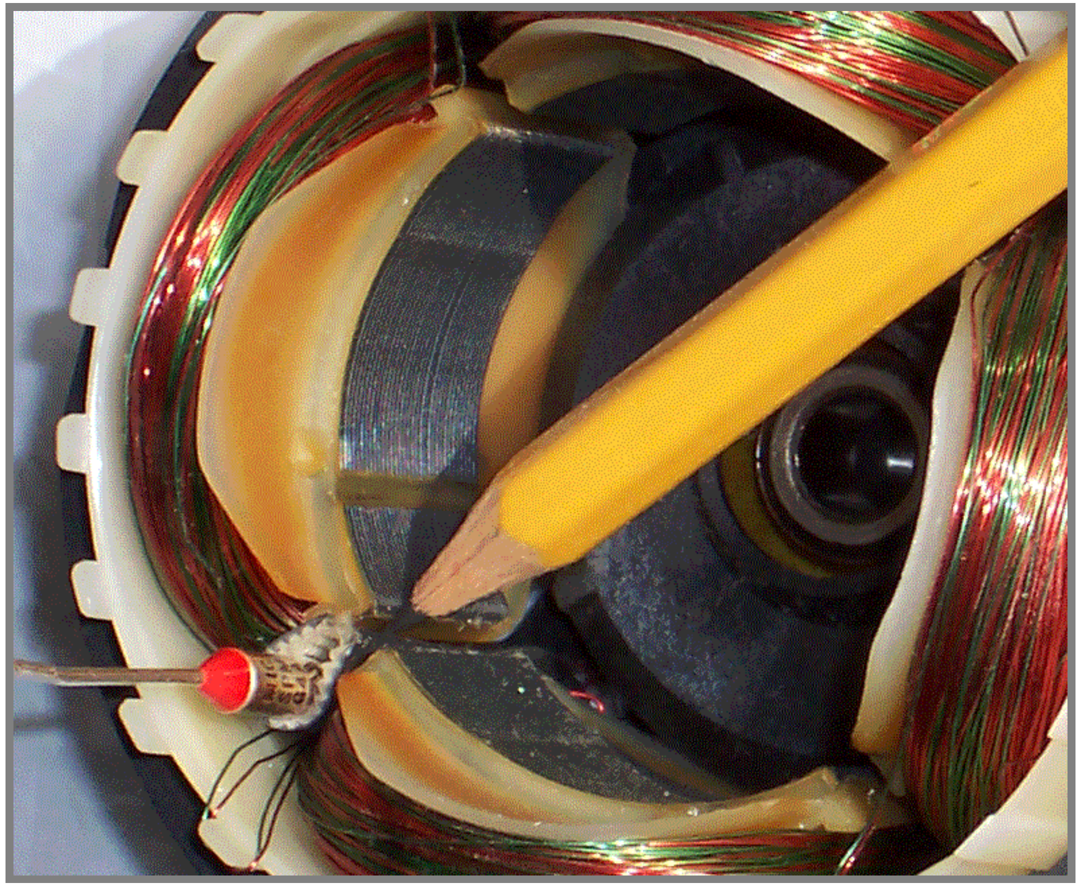
Consider Mitigations

- Remove basic events with mitigations
- Increase the size of cut sets with mitigations
- Recall: Guardrails

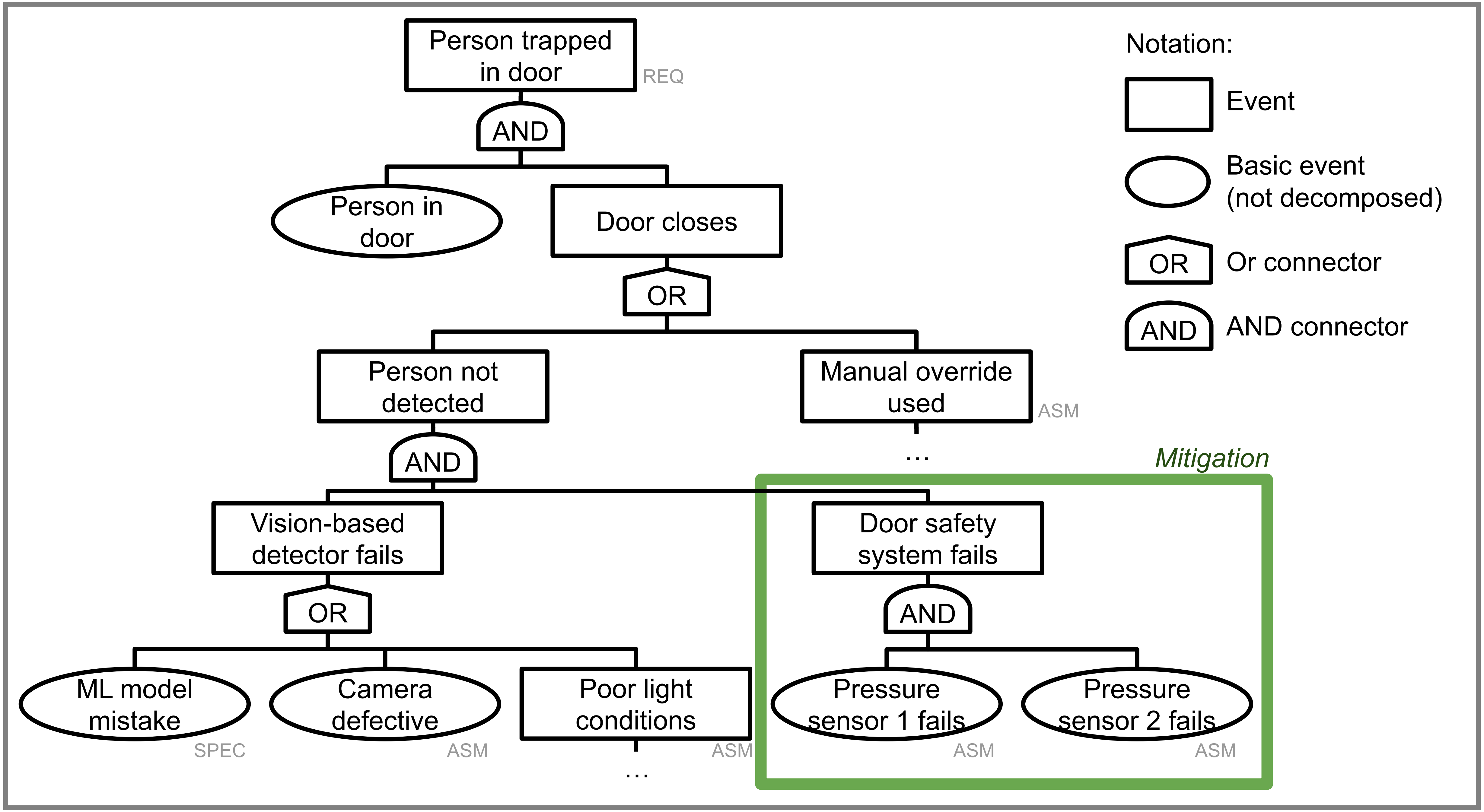


Guardrails - Examples

Recall: Thermal fuse in smart toaster



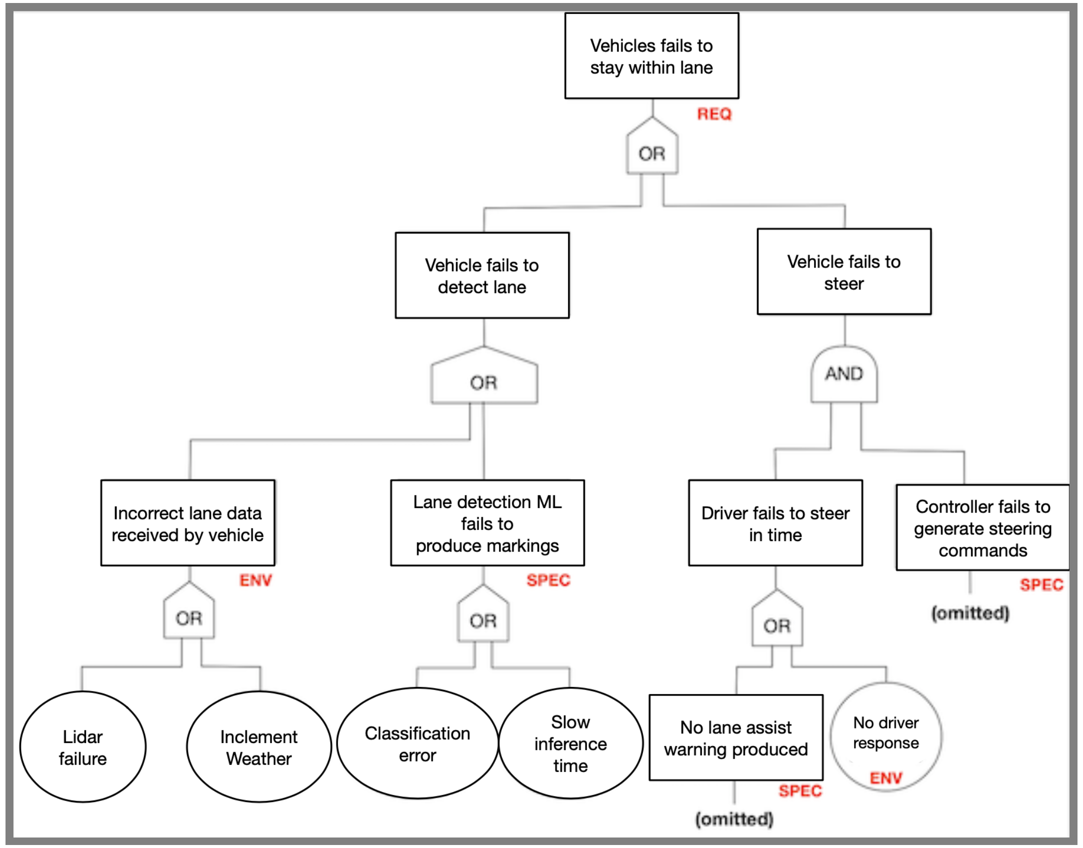
- maximum toasting time + extra heat sensor


Person trapped in door
REQ
AND
Person in door
Door closes
OR
Person not detected
Manual override used
ASM
...
AND
Mitigation
Vision-based detector fails
Door safety system fails
OR
AND
ML model mistake
SPEC
Camera defective
ASM
Poor light conditions
ASM
...
Pressure sensor 1 fails
ASM
Pressure sensor 2 fails
ASM
Notation:
Event
Basic event (not decomposed)
OR
Or connector
AND
AND connector

One more example: FTA for Lane Assist

- REQ: The vehicle must be prevented from veering off the lane.
- SPEC: Lane detector accurately identifies lane markings in the input image; the controller generates correct steering commands
- ASM: Sensors are providing accurate information about the lane; driver responses when given warning; steering wheel is functional

Possible mitigations?



FTA: Caveats

In general, building a **complete** tree is impossible

- There are probably some faulty events that you missed
- "Unknown unknowns"
- Events can always be decomposed; detail level is a choice.

Domain knowledge is crucial for improving coverage

- Talk to domain experts; augment your tree as you learn more

FTA is still very valuable for risk reduction!

- Forces you to think about, document possible failure scenarios
- A good starting basis for designing mitigations

FMEA

Fault-Tree Analysis Discussion

- Town-down, *backward* search for the root cause of issues
  - from final outcomes to initiating events
- Issues (TOP events) need to be known upfront
- Quantitative analysis possible
- Useful for understanding faults post-hoc
- Where do outcomes come from?

Failure Mode and Effects Analysis (FMEA)

	Function	Potential Failure Mode	Potential Effect(s) of Failure	SEV i	Potential Cause(s) of Failure	OCC i	Current Design Controls (Prevention)	Current Design Controls (Detection)	DET i	RPN i	Recommended Action(s)
1	Provide required levels of radiation	Radiation level too high for the required intervention	Over radiation of the patients.		Technician did not set the radiation at the right level.			Current algorithm resets to normal levels after imaging each patient.			Modify software to alert technician to unusually high radiation levels before activating.
2		Radiation at lower level than required	Patient fails to receive enough radiation.		Software does not respond to hardware mechanical setting.			Failure detection included in software			Include visual / audio alarm in the code when lack of response.
3											Improve recovery protocol.
4	Protect patients from unexpected high radiation	Higher radiation than required	Radiation burns		sneak paths in software			Shut the system if radiation level does not match the inputs.			Perform traceability matrix.

- A **forward search** technique to identify potential hazards
- Widely used in aeronautics, automotive, healthcare, food services, semiconductor processing, and (to some extent) software

FMEA Process

(a) Identify system components

(b) Enumerate potential failure modes

- *for ML component: Always suspect prediction may be wrong*

(c) For each failure mode, identify:

- Potential hazardous effect on the system
- Method for detecting the failure
- Potential mitigation strategy

FMEA Example: Autonomous Train Doors

Failure modes? Failure effects? Detection? Mitigation?

Exercise: FMEA Analysis for Smart Toaster

(video sensor, temperature sensor, heat sensor, user setting, ML model, heuristic shutdown, thermal fuse)

Failure modes? Failure effects? Detection? Mitigation?

FMEA Excerpt: Autonomous Car

Component	Failure Mode	Failure Effects	Sev	Potential Causes	Occ	Det	Recommended Action	RPN
Sensors								
Vision-based camera	Poor visibility	Outcome depends on whether other sensors remain operational and how the controller compensates for the loss of data. Collision is possible.	5	Driving at night, poor weather (heavy rain, snow, or fog), dirt or obstruction over lens	10	2	If confidence in sensor data is low, pull over or alert human driver to take control	100
	Hardware failure		5	Manufacturing fault, or at end of life cycle	4	4	Annual inspection	80
LIDAR	Poor visibility		5	Poor weather (heavy rain, snow, or fog), dirt or obstruction over sensor	8	2	If confidence in sensor data is low, pull over or alert human driver to take control	80
	LIDAR interference		5	Other AVs in the area using LIDAR	10	2	Laser signal should be coded with ID to prevent interference	100
	Positional error (bias error or noise)		4	Intrinsic to sensor	10	2	Measurement uncertainty should be conveyed to decision-making algorithm	80
	Hardware failure		5	Manufacturing fault, or at end of life cycle	3	4	Annual inspection	60

Excerpt of an FMEA table for analyzing components in an autonomous vehicle, from David Robert Beachum. Methods for assessing the safety of autonomous vehicles. University of Texas Theses and Dissertations (2019).

"Wrong Prediction" as Failure Mode?

"Wrong prediction" is a very cause grained failure mode of every model

May not be possible to decompose further

However, may evaluate causes of wrong prediction for better understanding, as far as possible --> FTA?

FMEA Summary

Forward analysis: From components to possible failures

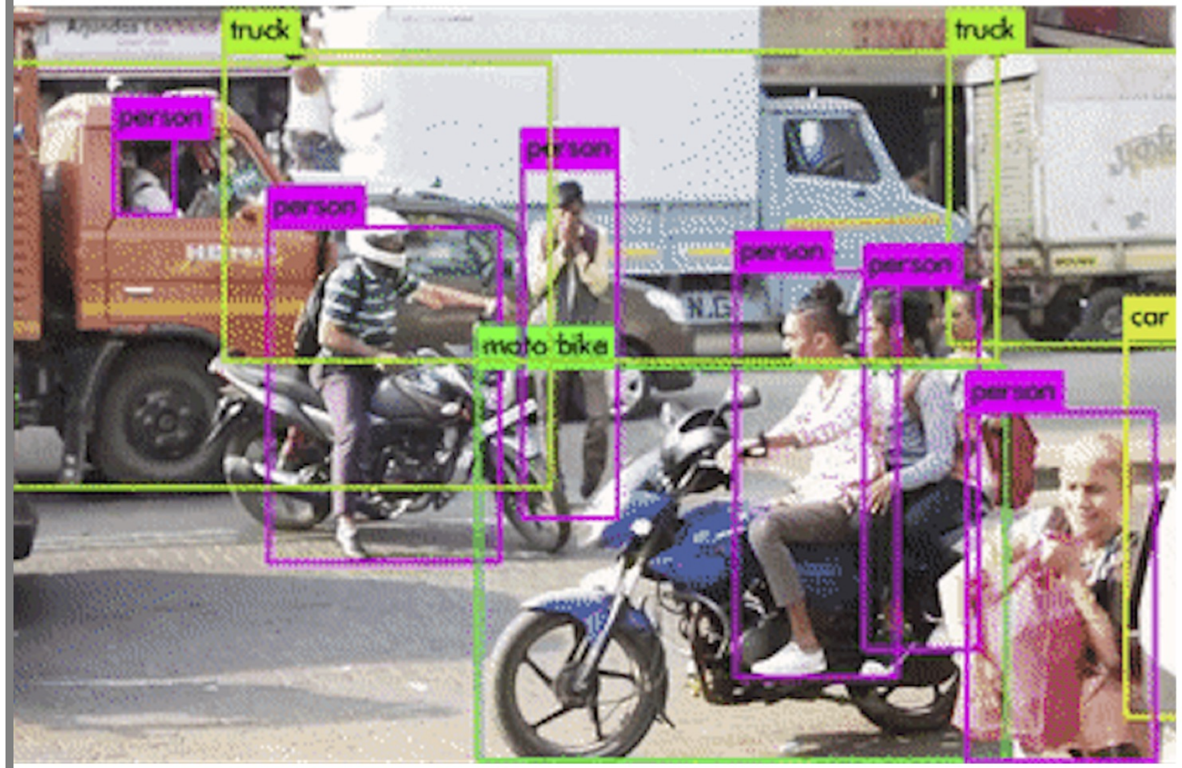
Focus on single component failures, no interactions

Identifying failure modes may require domain understanding

HAZOP

Hazard and Interoperability Study (HAZOP)

Identify hazards and component fault scenarios through guided inspection of requirements



Guide Word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN / INSTEAD	Complete substitution
EARLY	Relative to the clock time
LATE	Relative to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

Hazard and Operability Study (HAZOP)

A **forward search** method to identify potential hazards

For each component, use a set of **guide words** to generate possible deviations from expected behavior

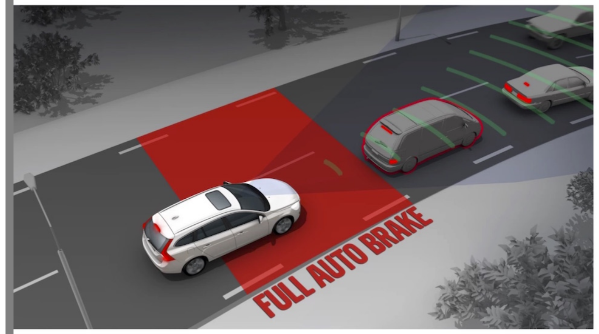
Consider the impact of each generated deviation: Can it result in a system-level hazard?

Guide Word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN / INSTEAD	Complete substitution
EARLY	Relative to the clock time
LATE	Relative to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

HAZOP Example: Emergency Braking (EB)

Specification: EB must apply a maximum braking command to the engine.

- **NO OR NOT**: EB does not generate any braking command.
- **LESS**: EB applies less than max. braking.
- **LATE**: EB applies max. braking but after a delay of 2 seconds.
- **REVERSE**: EB generates an acceleration command instead of braking.
- **BEFORE**: EB applies max. braking before a possible crash is detected.



Guide Word	Meaning
NO OR NOT	Complete negation of the design intent
MORE	Quantitative increase
LESS	Quantitative decrease
AS WELL AS	Qualitative modification/increase
PART OF	Qualitative modification/decrease
REVERSE	Logical opposite of the design intent
OTHER THAN / INSTEAD	Complete substitution
EARLY	Relative to the clock time
LATE	Relative to the clock time
BEFORE	Relating to order or sequence
AFTER	Relating to order or sequence

HAZOP & ML

In addition to traditional analysis: Analyze possible mistakes of all ML components

Original guidewords: NO OR NOT, MORE, LESS, AS WELL AS, PART OF, REVERSE, OTHER THAN / INSTEAD, EARLY, LATE, BEFORE, AFTER

Additional ML-specific guidewords: WRONG, INVALID, INCOMPLETE, PERTURBED, and INCAPABLE.

Breakout: Automated Train Doors

Analyze the vision component to detect obstacles in train doors

NO OR NOT, MORE, LESS, AS WELL AS, PART OF, REVERSE, OTHER THAN / INSTEAD, EARLY, LATE, BEFORE, AFTER, WRONG, INVALID, INCOMPLETE, PERTURBED, and INCAPABLE.

Using HAZOP: As a group answer in `#lecture`, tagging group members:

HAZOP: Benefits & Limitations

- Easy to use; encourages systematic reasoning about component faults
- Can be combined with FTA/FMEA to generate faults (i.e., basic events in FTA)
- Potentially labor-intensive; relies on engineer's judgement
- Does not guarantee to find all hazards (but also true for other techniques)

Remarks: Hazard Analysis

None of these methods guarantee completeness

- You may still be missing important hazards, failure modes

Intended as structured approaches to thinking about failures

- But cannot replace human expertise and experience

Summary

- Accept that a failure is inevitable
  - ML components will eventually make mistakes, reasons barely matter
  - Environment may evolve over time, violating assumptions
- Design strategies for mitigating mistakes
  - Human in the loop, Undoable actions, Guardrails, Mistake detection and recovery (monitoring, doer-checker, fail-over, redundancy), Containment and isolation
- Use risk analysis to identify and mitigate potential problems
  - FTA, FMEA, HAZOP

Further readings

- Google PAIR. People + AI Guidebook. 2019, especially chapters “Errors + Graceful Failure” and “Mental Models.”
- Martelaro, Nikolas, Carol J. Smith, and Tamara Zilovic. “Exploring Opportunities in Usable Hazard Analysis Processes for AI Engineering.” In AAAI Spring Symposium Series Workshop on AI Engineering: Creating Scalable, Human-Centered and Robust AI Systems (2022).
- Qi, Yi, Philippa Ryan Conmy, Wei Huang, Xingyu Zhao, and Xiaowei Huang. “A Hierarchical HAZOP-Like Safety Analysis for Learning-Enabled Systems.” In AISafety2022 Workshop at IJCAI2022 (2022).
- Beachum, David Robert. “Methods for assessing the safety of autonomous vehicles.” MSc thesis, 2019.
- Amershi, Saleema, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh et al. “Guidelines for human-AI interaction.” In Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, 2019.
- Shneiderman, Ben. “Bridging the gap between ethics and practice: Guidelines for reliable, safe, and trustworthy Human-Centered AI systems.” ACM Transactions on Interactive Intelligent Systems (TiiS) 10, no. 4 (2020): 1–31.