RISK AND PLANNING FOR MISTAKES II

Eunsuk Kang

Required reading: "How Big Data Transformed Applying to College", Cathy O'Neil

LEARNING GOALS:

- Evaluate the risks of mistakes from AI components using the fault tree analysis (FTA)
- Design strategies for mitigating the risks of failures due to AI mistakes

RISK ANALYSIS

WHAT IS RISK ANALYSIS?

WHAT IS RISK ANALYSIS?

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

WHAT IS RISK ANALYSIS?

- What can possibly go wrong in my system, and what are potential impacts on system requirements?
- Risk = Likelihood * Impact

WHAT IS RISK ANALYSIS?

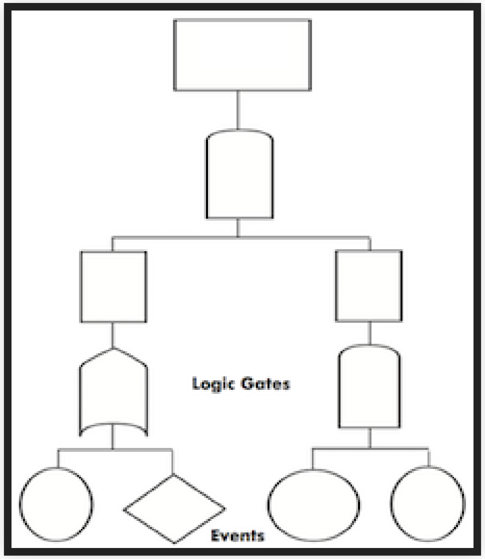
- What can possibly go wrong in my system, and what are potential impacts on system requirements?
- Risk = Likelihood * Impact
- A number of methods:
    - Failure mode & effects analysis (FMEA)
    - Hazard analysis
    - Why-because analysis
    - Fault tree analysis (FTA) <= Today's focus!
    - ...

RISKS?

- Lane assist system
- Credit rating
- Amazon product recommendation
- Audio transcription service
- Cancer detection
- Predictive policing

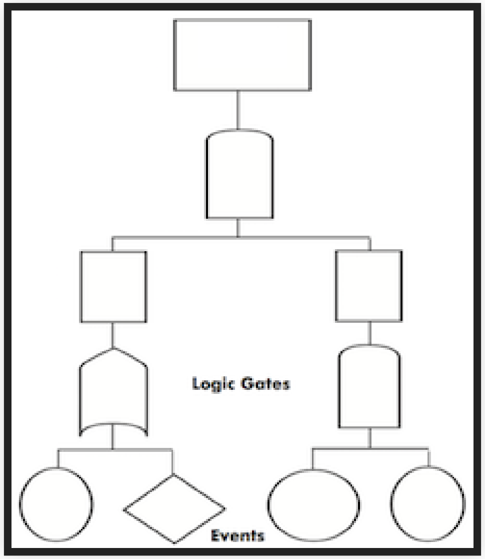
Discuss potential risks, including impact and likelyhood

FAULT TREE ANALYSIS (FTA)


Logic Gates
Events

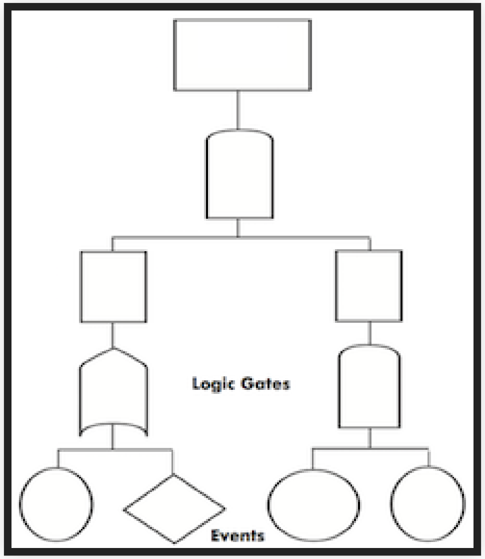
FAULT TREE ANALYSIS (FTA)

- Fault tree: A top-down diagram that displays the relationships between a system failure (i.e., requirement violation) and its potential causes.
    - Identify sequences of events that result in a failure
    - Prioritize the contributors leading to the failure
    - Inform decisions about how to (re-)design the system
    - Investigate an accident & identify the root cause



FAULT TREE ANALYSIS (FTA)

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



FAULT TREE ANALYSIS & AI

- Increaseingly used in automotive, aeronautics, industrial control systems, etc.,

FAULT TREE ANALYSIS & AI

- Increaseingly used in automotive, aeronautics, industrial control systems, etc.,
- AI is just one part of the system

FAULT TREE ANALYSIS & AI

- Increaseingly used in automotive, aeronautics, industrial control systems, etc.,
- AI is just one part of the system
- AI will EVENTUALLY make mistakes
    - Ouput wrong predictions/values
    - Fail to adapt to changing environment
    - Confuse users, etc.,

FAULT TREE ANALYSIS & AI

- Increaseingly used in automotive, aeronautics, industrial control systems, etc.,
- AI is just one part of the system
- AI will EVENTUALLY make mistakes
    - Ouput wrong predictions/values
    - Fail to adapt to changing environment
    - Confuse users, etc.,
- How do mistakes made by AI contribute to system failures? How do we ensure their mistakes do not result in a catastrophe?

FAULT TREES:: BASIC BUILDING BLOCKS

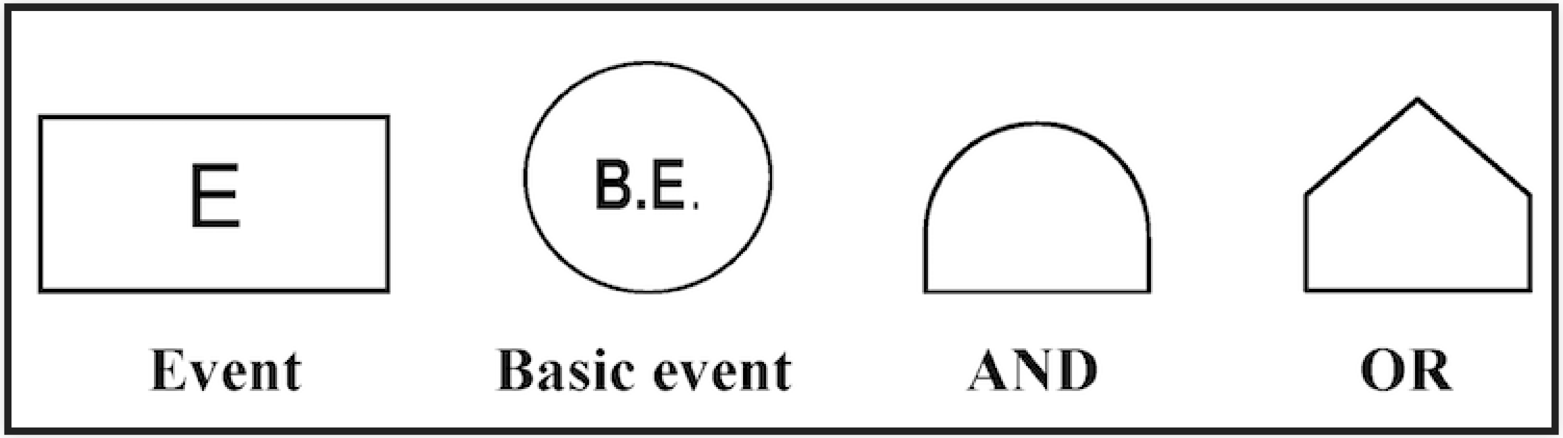
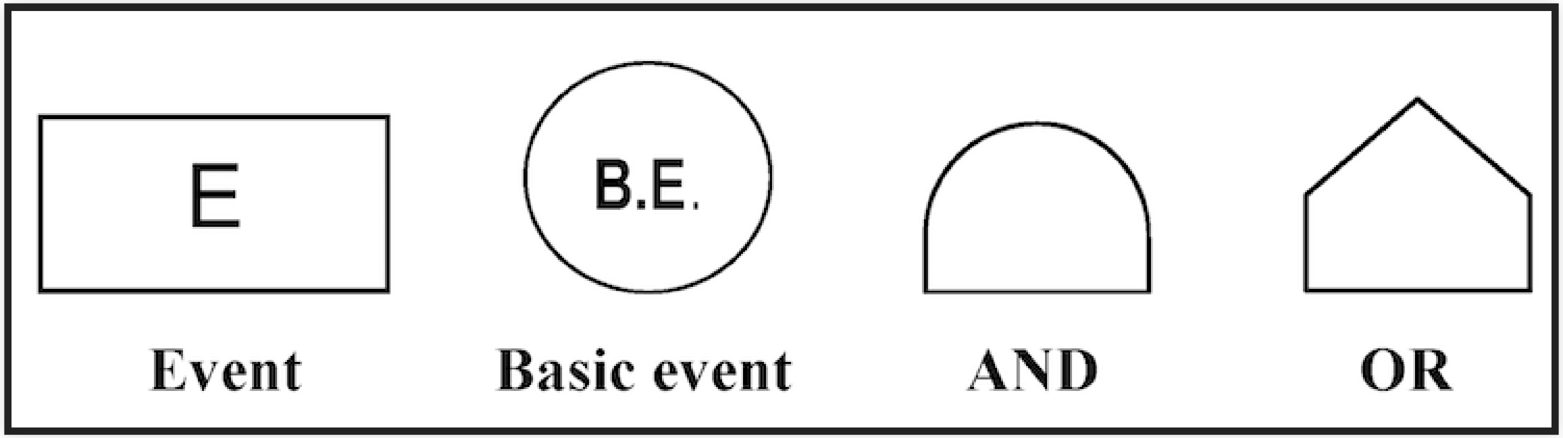

E
B.E.
Event
Basic event
AND
OR

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

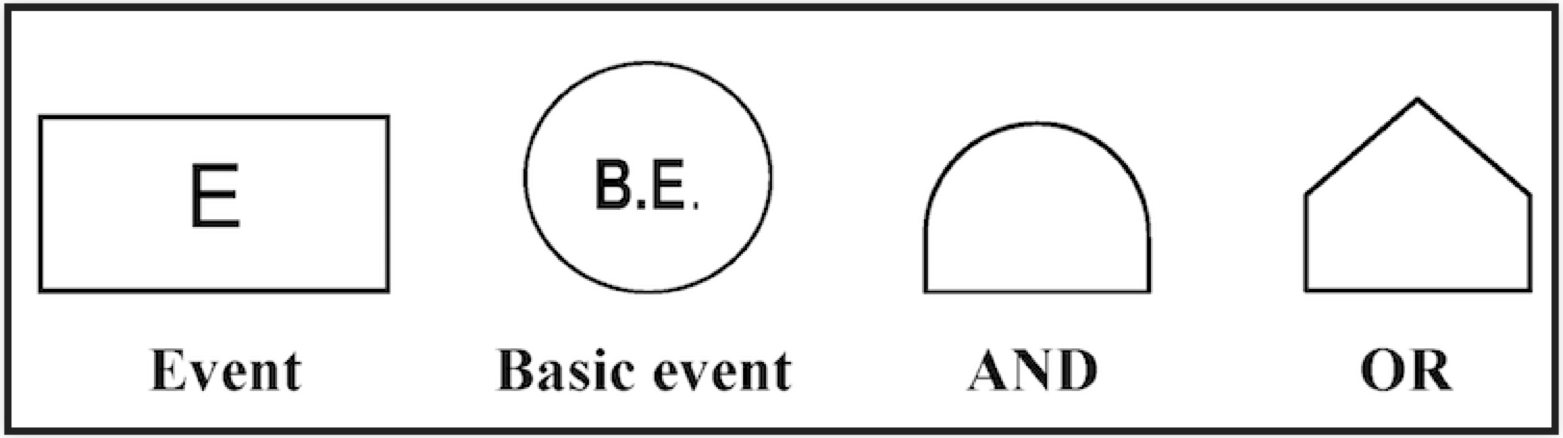
FAULT TREES:: BASIC BUILDING BLOCKS



- 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; leafs of the tree

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

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; leafs of the tree
- Gate: Logical relationship between an event & its immedicate subevents
  - AND: All of the sub-events must take place
  - OR: Any one of the sub-events may result in the parent event

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

FAULT TREE EXAMPLE

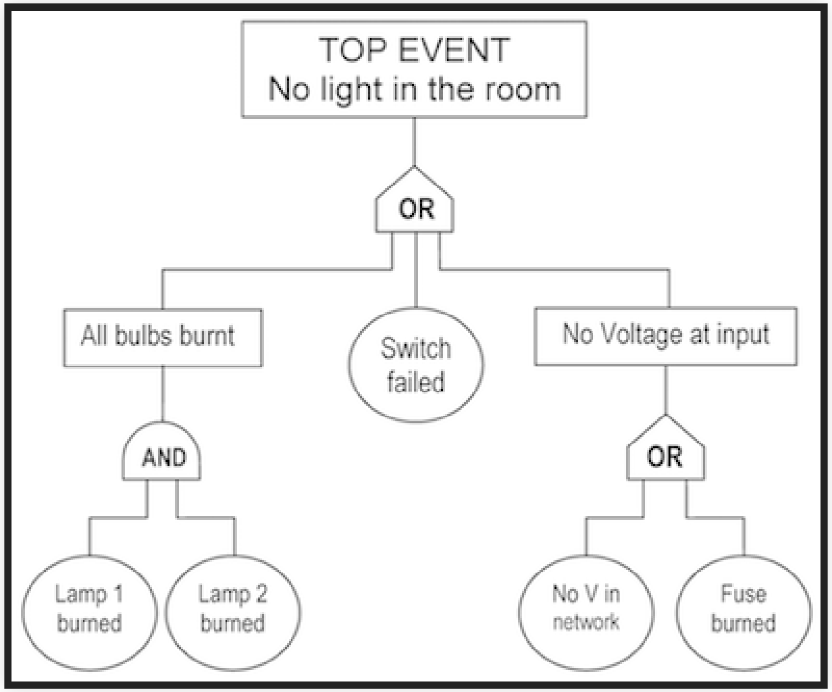
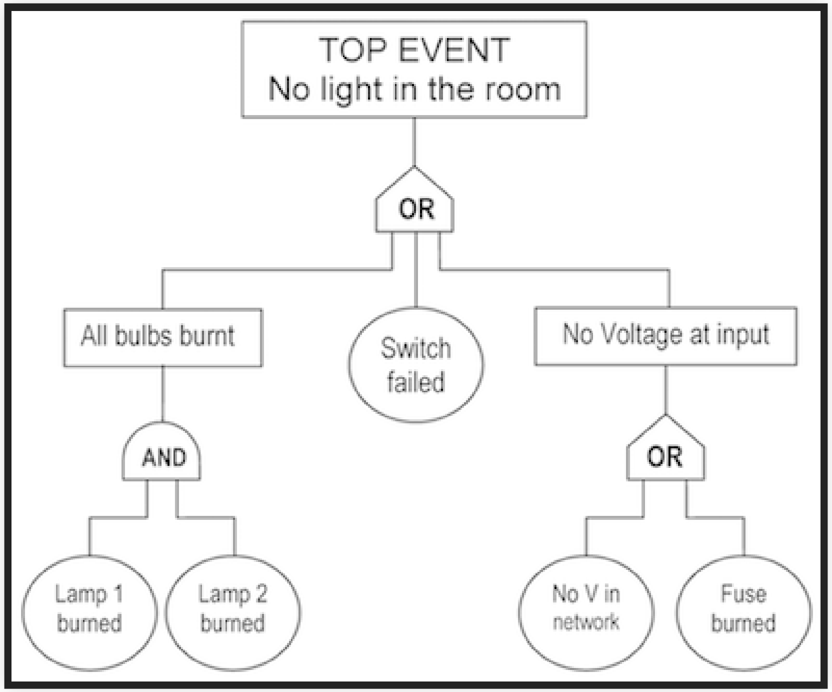

TOP EVENT
No light in the room
OR
All bulbs burnt
Switch failed
No Voltage at input
AND
OR
Lamp 1 burned
Lamp 2 burned
No V in network
Fuse burned

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

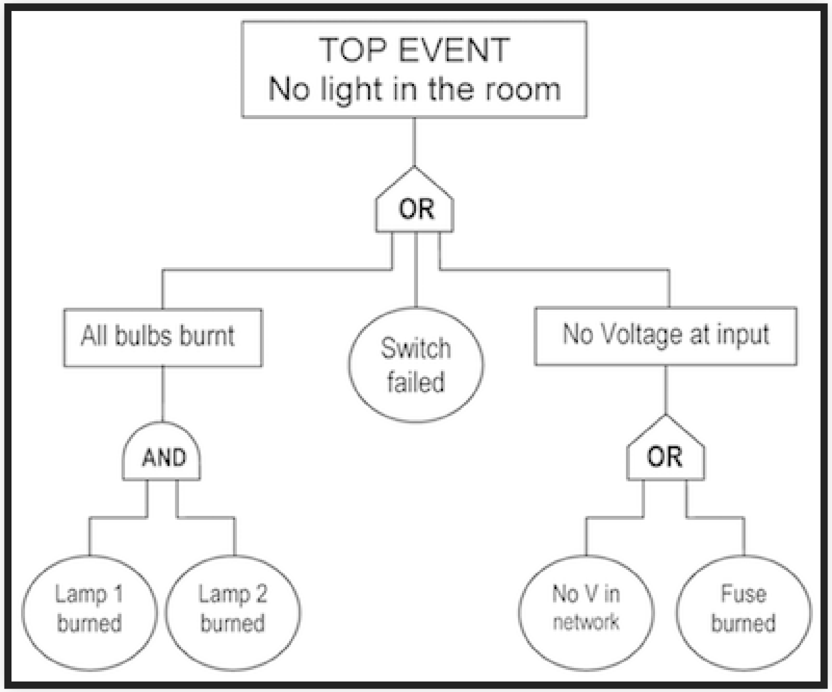
FAULT TREE EXAMPLE


TOP EVENT
No light in the room
OR
All bulbs burnt
Switch failed
No Voltage at input
AND
OR
Lamp 1 burned
Lamp 2 burned
No V in network
Fuse burned

- Every tree begins with a TOP event (typically a violation of a requirement)

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

FAULT TREE EXAMPLE


TOP EVENT
No light in the room
OR
All bulbs burnt
Switch failed
No Voltage at input
AND
OR
Lamp 1 burned
Lamp 2 burned
No V in network
Fuse burned

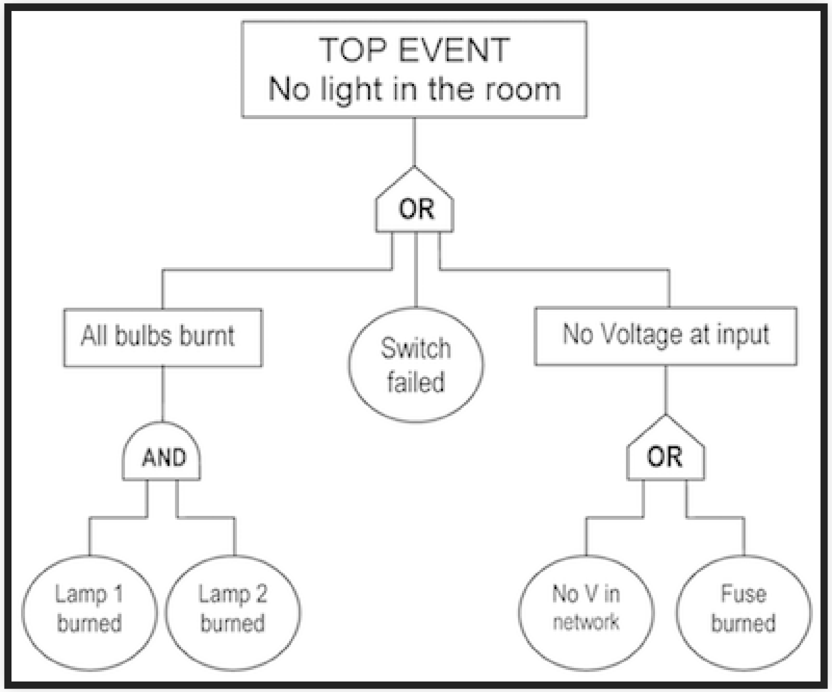
- Every tree begins with a TOP event (typically a violation of a requirement)
- Every branch of the tree must terminate with a basic event

Figure from *Fault Tree Analysis and Reliability Block Diagram* (2016), Jaroslav Menčík.

ANALYSIS

- What can we do with fault trees?
    - Qualitative analysis: Determine potential root causes of a failiure through *minimal cut set analysis*
    - Quantitative analysis: Compute the probablity of a failure

MINIMAL CUT SET ANALYSIS


TOP EVENT
No light in the room
OR
All bulbs burnt
Switch failed
No Voltage at input
AND
OR
Lamp 1 burned
Lamp 2 burned
No V in network
Fuse burned

- 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 be obtained by removing a basic event.
- Q. What are minimal cut sets in the above tree?

FAILURE PROBABILITY ANALYSIS

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 prob. of intermediate events through AND & OR gates
    - (Alternatively, as sum of prob. of minimal cut sets)

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 prob. 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

FTA PROCESS

1. Specify the system structure
    - Environment entities & machine components
    - Assumptions (ENV) & specifications (SPEC)

FTA PROCESS

1. Specify the system structure
    - Environment entities & machine components
    - Assumptions (ENV) & specifications (SPEC)
2. Identify the top event as a violation of REQ

FTA PROCESS

1. Specify the system structure
    - Environment entities & machine components
    - Assumptions (ENV) & specifications (SPEC)
2. Identify the top event as a violation of REQ
3. Construct the fault tree
    - Intermediate events can be derived from violation of SPEC/ENV

FTA PROCESS

1. Specify the system structure
   - Environment entities & machine components
   - Assumptions (ENV) & specifications (SPEC)
2. Identify the top event as a violation of REQ
3. Construct the fault tree
   - Intermediate events can be derived from violation of SPEC/ENV
4. Analyze the tree
   - Identify all possible minimal cut sets

FTA PROCESS

1. Specify the system structure
    - Environment entities & machine components
    - Assumptions (ENV) & specifications (SPEC)
2. Identify the top event as a violation of REQ
3. Construct the fault tree
    - Intermediate events can be derived from violation of SPEC/ENV
4. Analyze the tree
    - Identify all possible minimal cut sets
5. Consider design modifications to eliminate certain cut sets

FTA PROCESS

1. Specify the system structure
    - Environment entities & machine components
    - Assumptions (ENV) & specifications (SPEC)
2. Identify the top event as a violation of REQ
3. Construct the fault tree
    - Intermediate events can be derived from violation of SPEC/ENV
4. Analyze the tree
    - Identify all possible minimal cut sets
5. Consider design modifications to eliminate certain cut sets
6. Repeat

EXAMPLE: FTA FOR LANE ASSIST

- REQ: The vehicle must be prevented from veering off the lane.
- ENV: Sensors are providing accurate information about the lane; driver responses when given warning; steering wheel is functional
- SPEC: Lane detection accurately identifies the lane markings; the controller generates correct steering commands to keep the vehicle within lane

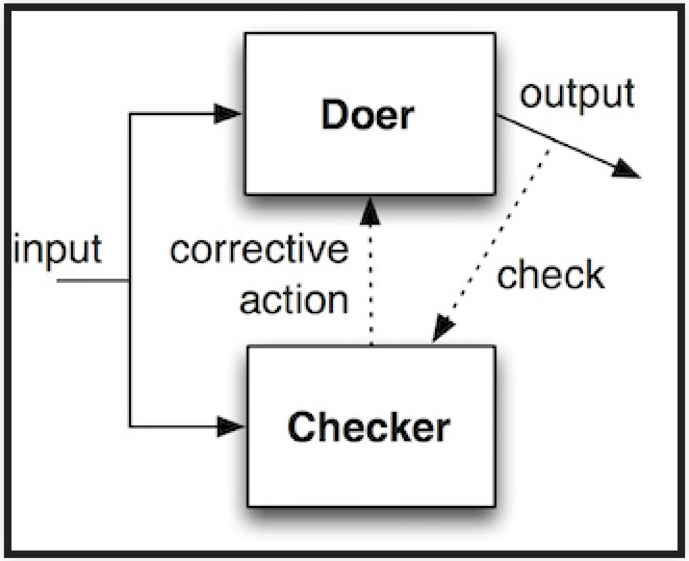
EXAMPLE: FTA FOR LANE ASSIST

MITIGATION STRATEGIES

ELEMENTS OF FAULT-TOLERANT DESIGN

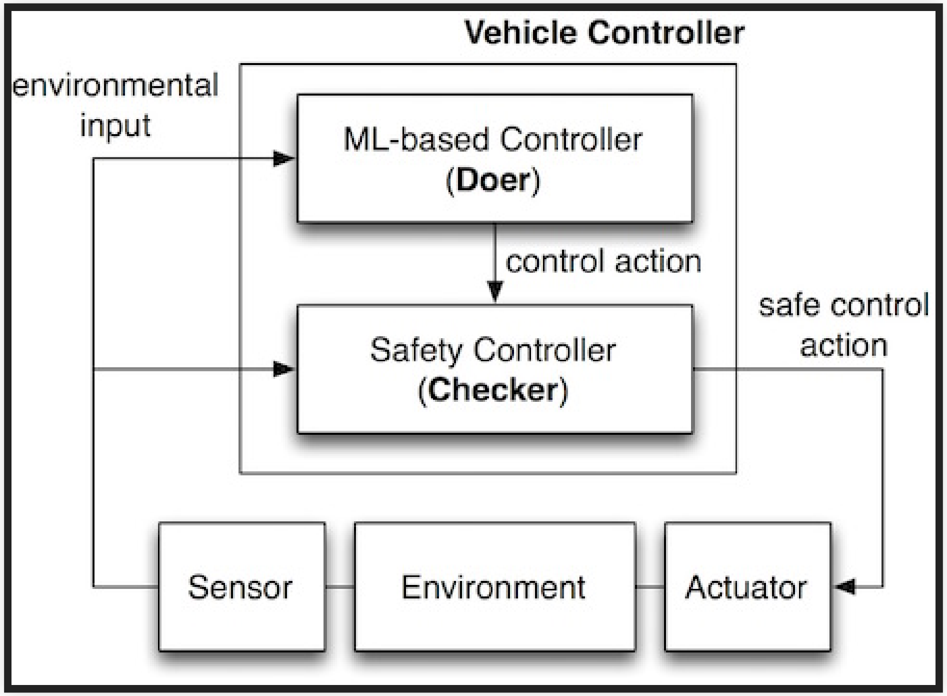
- **Assume**: Components will fail at some point
- **Goal**: Minimize the impact of failures
- **Detection**
  - Monitoring
- **Response**
  - Graceful degradation (fail-safe)
  - Redundancy (fail over)
- **Containment**
  - Decoupling & isolation

DETECTION: MONITORING


Doer
output
input
corrective action
check
Checker

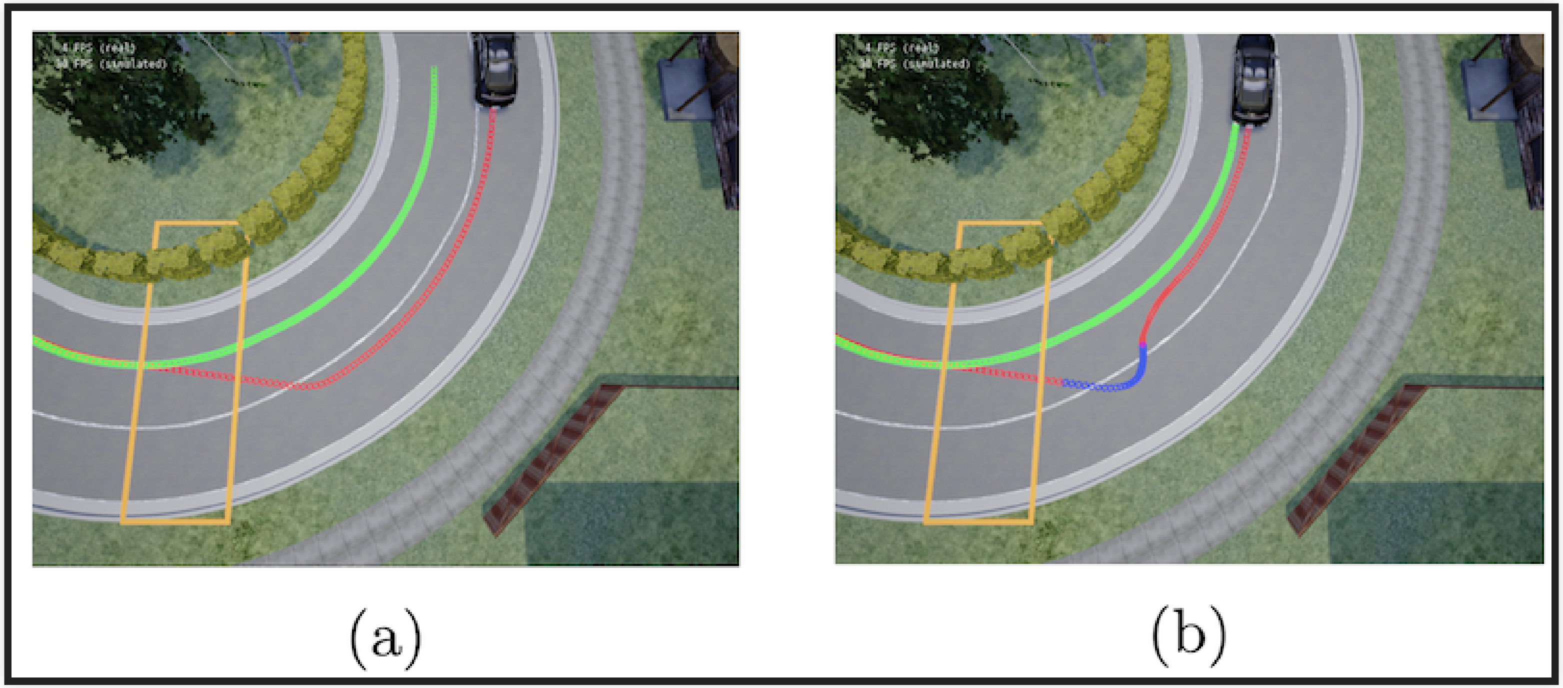
- **Goal**: Detect when a component failure occurs
- **Heartbeat** pattern
    - Periodically sends diagnostic message to monitor
- **Doer-Checker** pattern
    - Doer: Perform primary function; untrusted and potentially faulty
    - Checker: If doer output faulty, perform corrective action (e.g., default safe output, shutdown); trusted and verifiable

DOER-CHECKER EXAMPLE: AUTONOMOUS VEHICLE


Vehicle Controller
environmental input
ML-based Controller (**Doer**)
control action
Safety Controller (**Checker**)
safe control action
Sensor
Environment
Actuator

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

DOER-CHECKER EXAMPLE: AUTONOMOUS VEHICLE



- Yellow region: Slippery road, causes loss of traction
- ML-based controller (**doer**): Model ignores traction loss; generates unsafe maneuvering commands (a)
- Safe controller (**checker**): Overrides with safe steering commands (b)

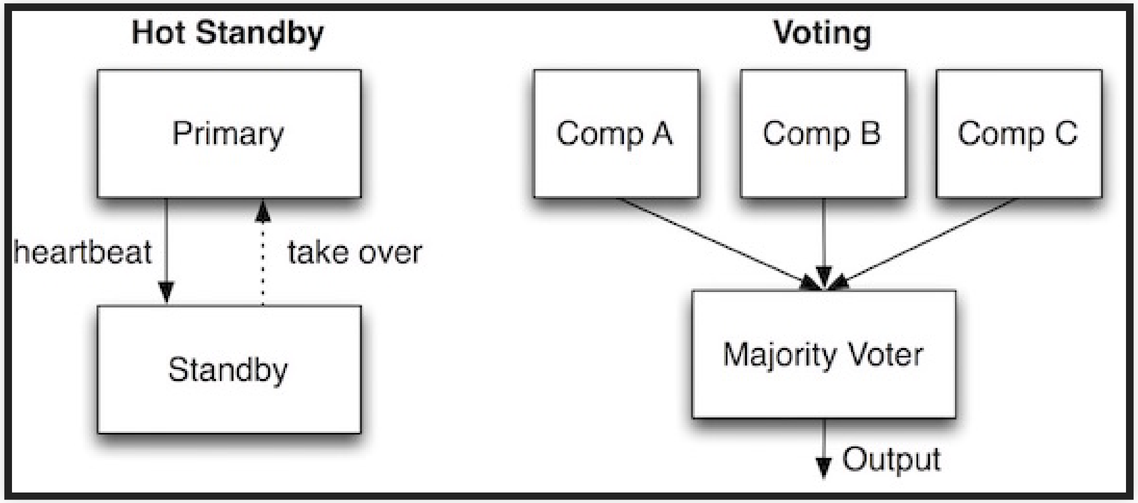
Runtime-Safety-Guided Policy Repair, Intl. Conference on Runtime Verification (2020)

RESPONSE: GRACEFUL DEGRADATION (FAIL-SAFE)

[redacted]

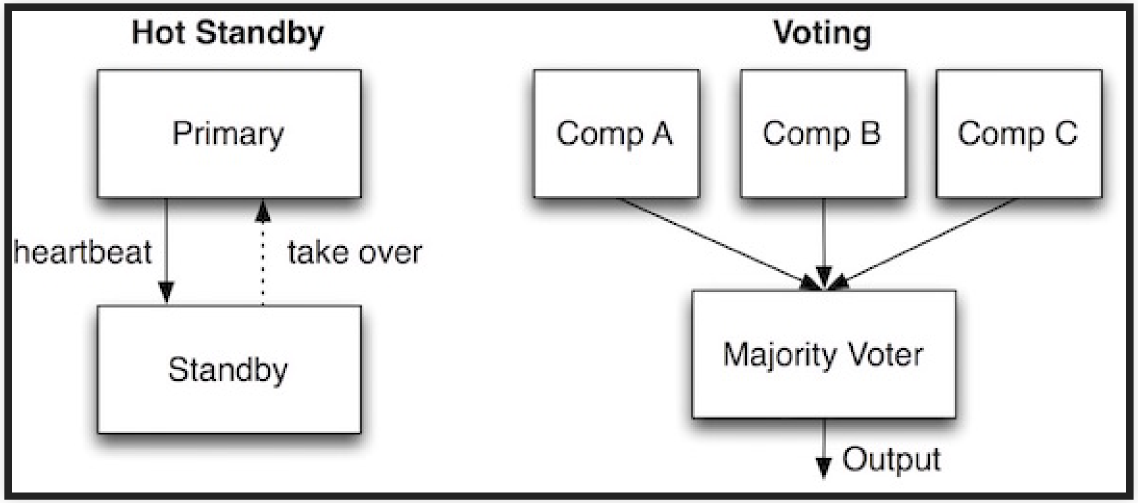
- **Goal**: When a component failure occurs, continue to provide safety (possibly at reduced functionality and performance)
- Relies on a monitor to detect component failures
- Example: Perception in autonomous vehicles
    - If Lidar fails, switch to a lower-quality detector; be more conservative
    - **But what about other types of ML failures? (e.g., misclassification)**

RESPONSE: REDUNDANCY (FAILOVER)


Hot Standby
Primary
heartbeat
take over
Standby
Voting
Comp A
Comp B
Comp C
Majority Voter
Output

- **Goal**: When a component fails, continue to provide the same functionality
- **Hot Standby**: Standby watches & takes over when primary fails
- **Voting**: Select the majority decision
- Caution: Do components fail independently?
    - Reasonable assumption for hardware/mechanical failures
    - **Q. What about software?**

RESPONSE: REDUNDANCY (FAILOVER)


Hot Standby
Primary
heartbeat
take over
Standby
Voting
Comp A
Comp B
Comp C
Majority Voter
Output

- **Goal**: When a component fails, continue to provide the same functionality
- **Hot Standby**: Standby watches & takes over when primary fails
- **Voting**: Select the majority decision
- Caution: Do components fail independently?
  - Reasonable assumption for hardware/mechanical failures
  - Software: Difficult to achieve independence even when built by different teams (e.g., N-version programming)
  - **Q. ML components?**

RESPONSE: HUMAN IN THE LOOP

Less forceful interaction, making suggestions, asking for confirmation

- 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 and often better with thin data
- AI for prediction, human for judgment?
- But:
    - 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

Examples?

Speaker notes

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

RESPONSE: UNDOABLE ACTIONS

Design system to reduce consequence of wrong predictions, allowing humans to override/undo

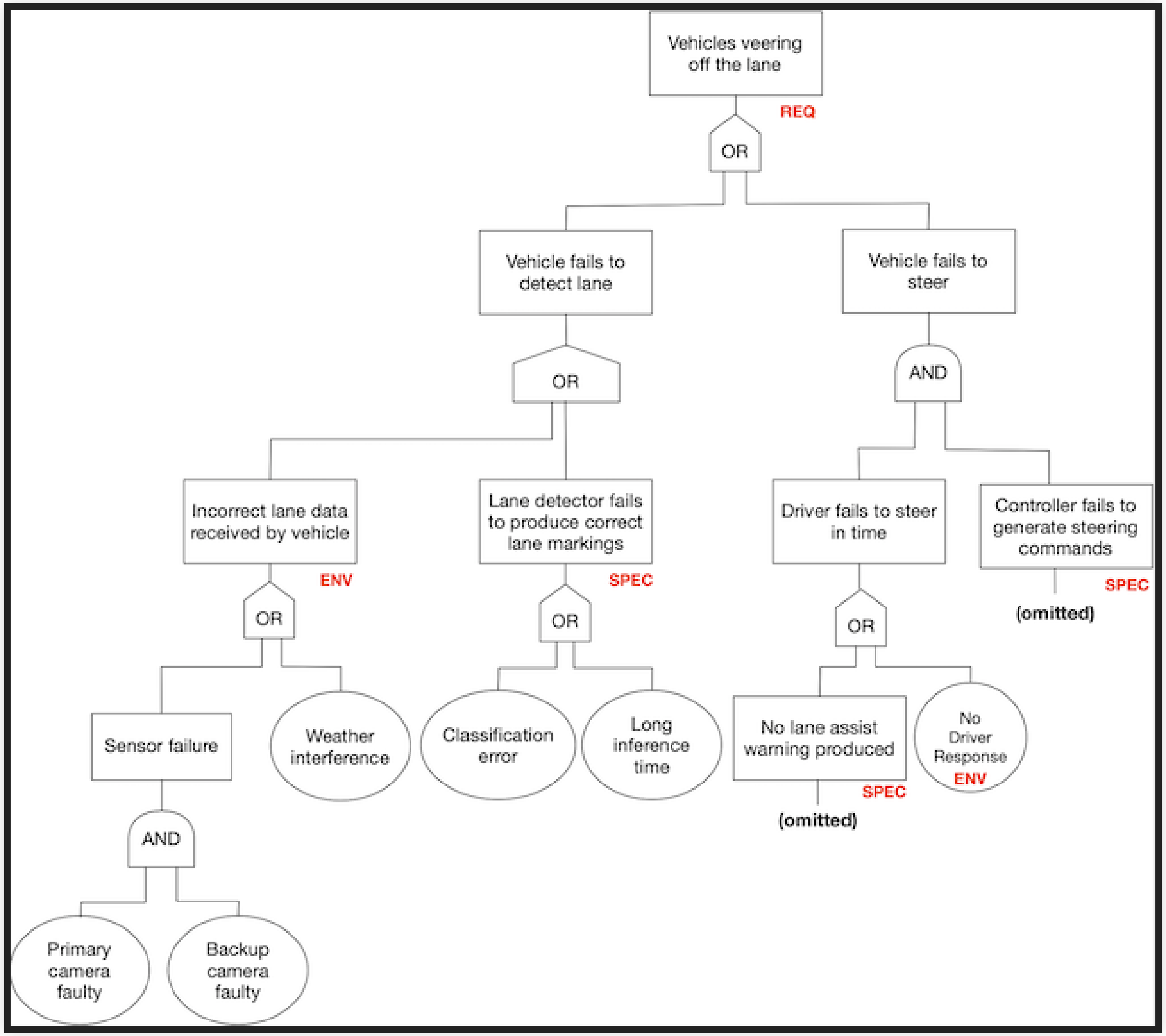
Examples?

Speaker notes

Smart home devices, credit card applications, Powerpoint design suggestions

EXAMPLE: LANE ASSIST

Q. Possible mitigation strategies?


Vehicles veering off the lane
REQ
OR
Vehicle fails to detect lane
Vehicle fails to steer
OR
AND
Incorrect lane data received by vehicle
ENV
Lane detector fails to produce correct lane markings
SPEC
Driver fails to steer in time
Controller fails to generate steering commands
SPEC
(omitted)
OR
OR
OR
Sensor failure
Weather interference
Classification error
Long inference time
No lane assist warning produced
SPEC
(omitted)
No Driver Response
ENV
AND
Primary camera faulty
Backup camera faulty

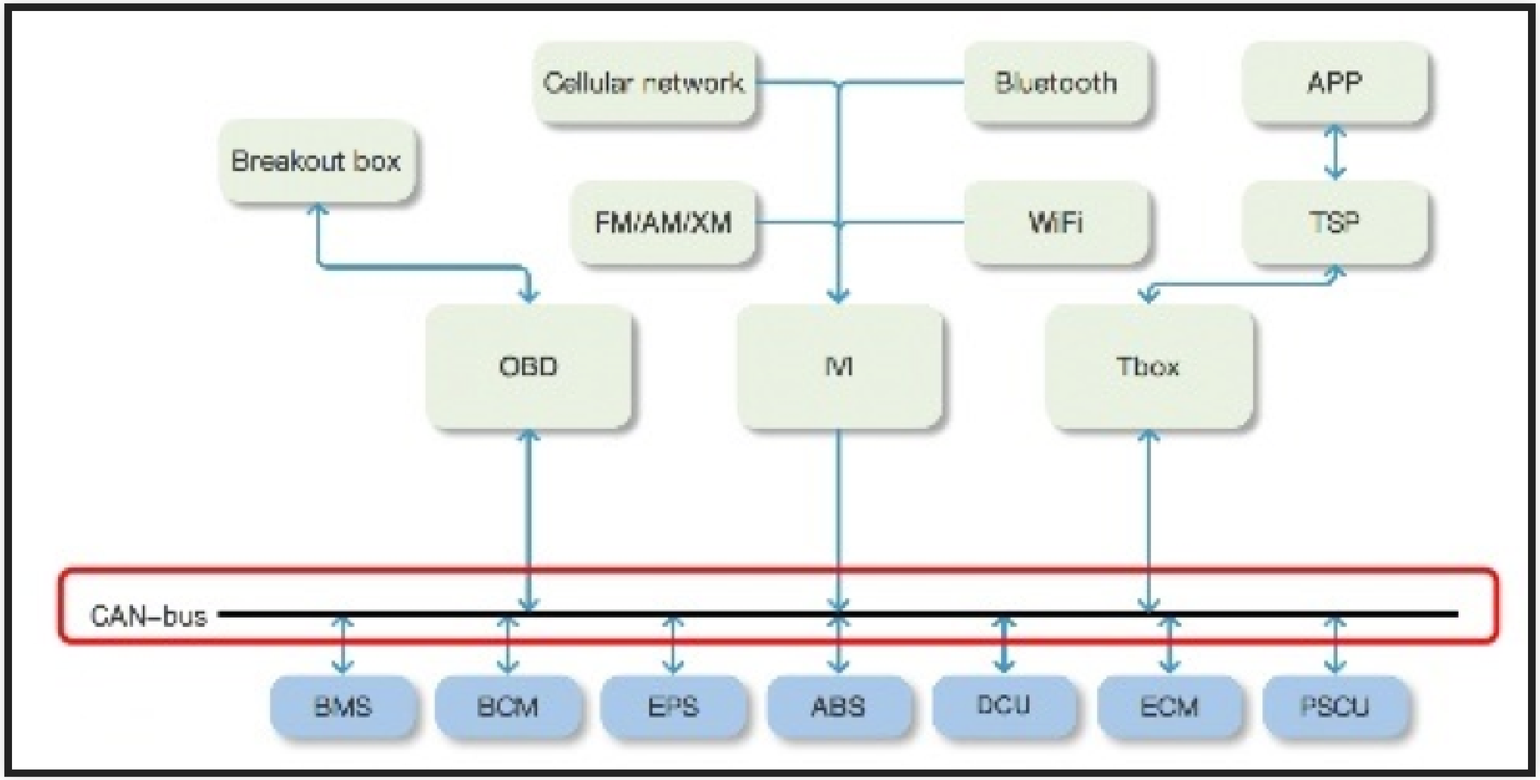
CONTAINMENT: DECOUPLING & ISOLATION

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

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 3 hours
- **Lesson**: Handle expected component faults; prevent propagation

POOR DECOUPLING: AUTOMOTIVE SECURITY


Cellular network
Bluetooth
APP
Breakout box
FM/AM/XM
WiFi
TSP
OBD
IVI
Tbox
CAN–bus
BMS
BCM
EPS
ABS
DCU
ECM
PSCU

- 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 (Stefan Savage, UCSD)

CONTAINMENT: DECOUPLING & ISOLATION

- Goal: Faults in a low-critical (LC) components should not impact high-critical (HC) components
- Apply the principle of least privilege
    - LC components should be allowed to access min. necessary functions
- Limit interactions across criticality boundaries
    - Deploy LC & HC components on different networks
    - Add monitors/checks at interfaces
- Is AI in my system performing an LC or HC task?
    - If HC, can we "demote" it into LC?
    - Alternatively, replace HC AI components with non-AI ones
    - **Q. Examples?**

SUMMARY

- Accept that ML components will make mistakes
- Use risk analysis to identify and mitigate potential problems
- Design strategies for detecting and mitigating the risks from mistakes by AI