Machine Learning in Production

Gathering Requirements

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

- Understand the role of requirements in ML-based systems and their failures
- Understand the distinction between the world and the machine
- Understand the importance of environmental assumptions in establishing system requirements

Readings

Required reading: Jackson, Michael. "The world and the machine." In Proceedings of the International Conference on Software Engineering. IEEE, 1995.

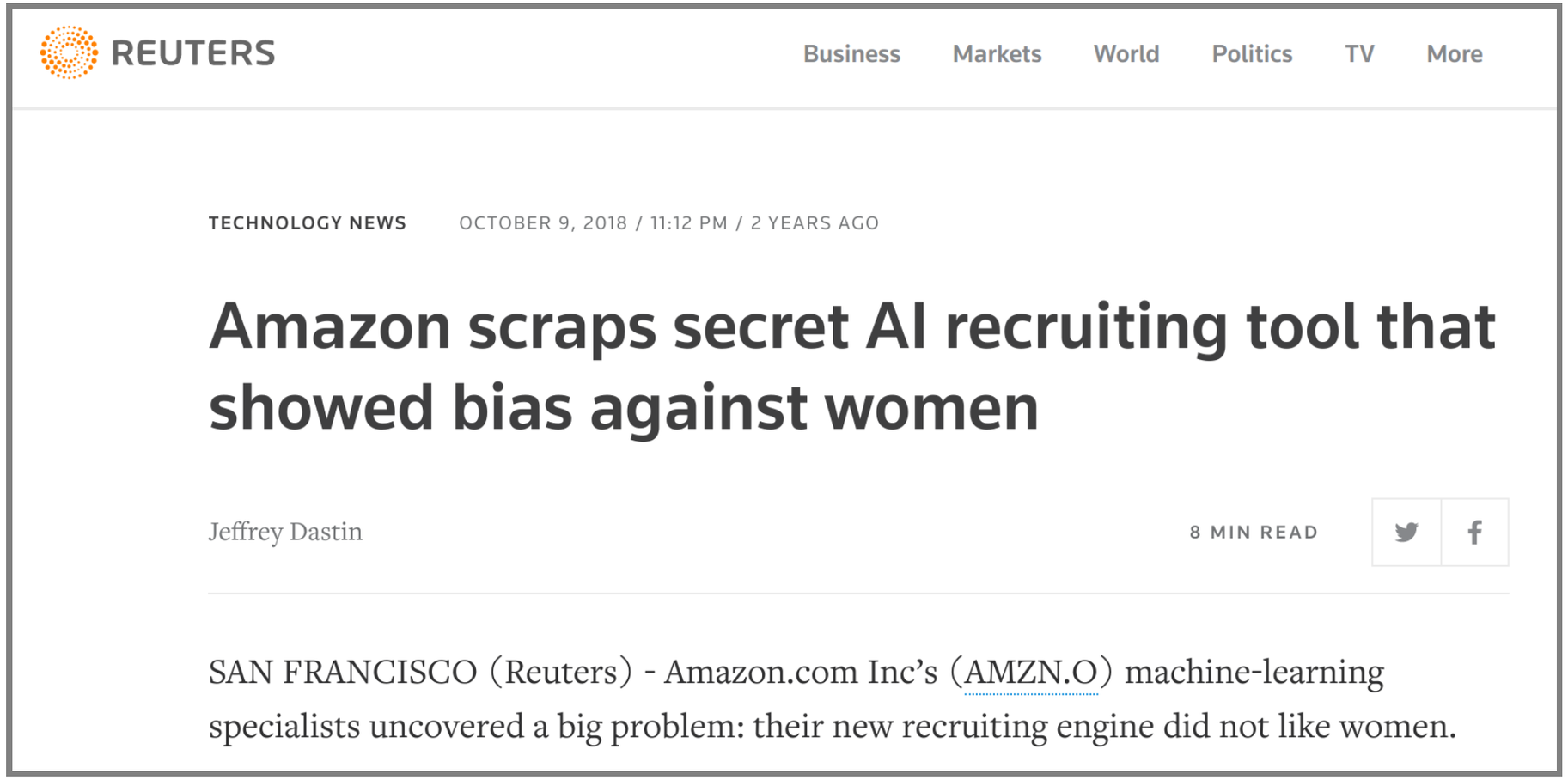
Going deeper: Van Lamsweerde, Axel. Requirements engineering: From system goals to UML models to software. John Wiley & Sons, 2009.

Failures in ML-Based Systems

Facial Recognition in ATM

Q. What went wrong? What is the root cause of the failure?

Automated Hiring



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.

Q. What went wrong? What is the root cause of the failure?

Autopilot in Vehicles

U.S. NEWS

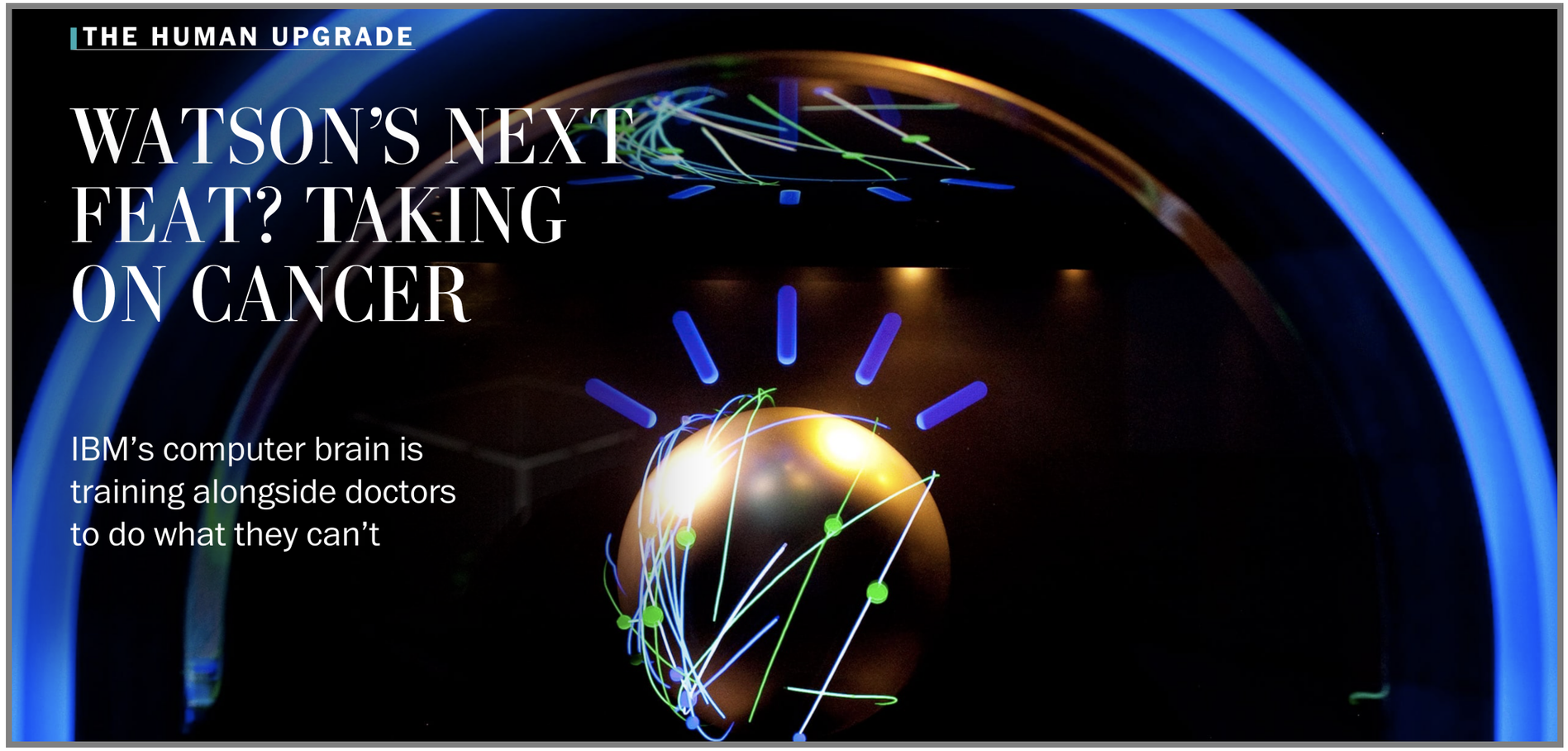
Tesla driver slept as car was going over 80 mph on Autopilot, Wisconsin officials say

Q. What went wrong? What is the root cause of the failure?

IBM Watson


PIENSE
THINK
सोचिए
ΣΚΕΨΟΥ
DENKE
PENSER
$300,000
Who is Stoker?
(I FOR ONE WELCOME OUR NEW COMPUTER OVERLORDS)
$1,000
$1,000,000
Who is Bram Stoker?
$ 17,973
$200,000
WHO IS BRAM STOKER?
$5600

IBM Watson



Washington Post, 06/2015

IBM Watson

future tense

How IBM's Watson Went From the Future of Health Care to Sold Off for Parts

BY LIZZIE O'LEARY JAN 31, 2022 • 9:00 AM

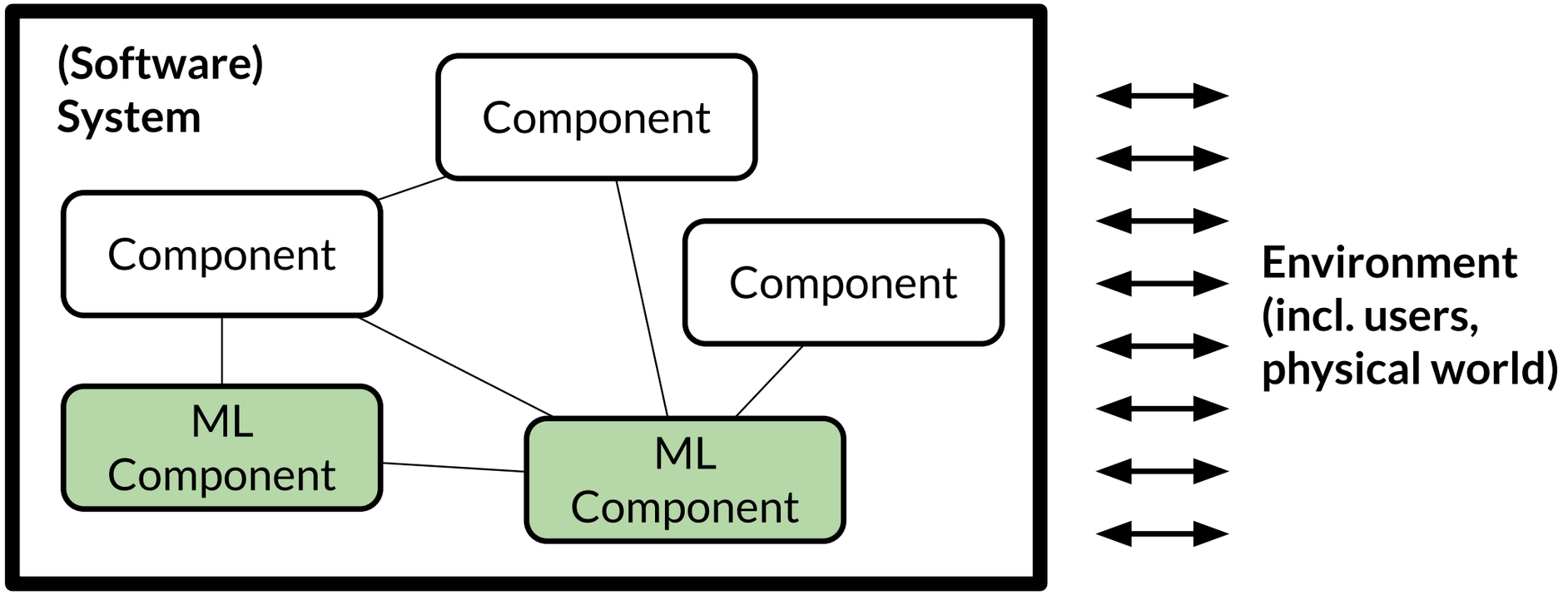
"We got concerns from them that the recommendations that it was giving were just not relevant...it would suggest a particular kind of treatment that wasn't available in the locality in which it was making the recommendation, or the recommendation did not at all square with the treatment protocols that were in use at the local institution..."

Slate, 01/2022

Risks in ML-based Systems

What went wrong? What were the root causes of failures in these systems? Was the quality of an ML model to blame?

Reminder: ML in a System



Machine learning is a component within a system

Need to also understand other parts and environment

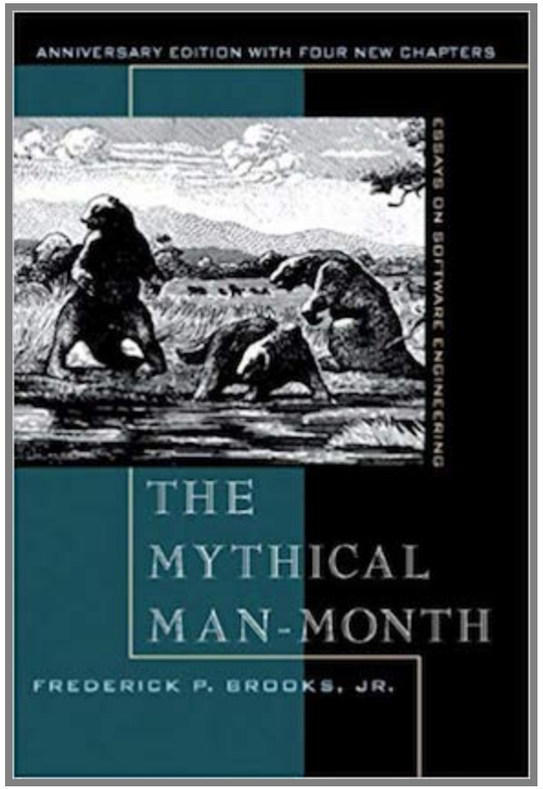
Software Requirements

Software Requirements

- Describe what the system should do, in terms of the services that it provides and their qualities (safety, reliability, performance…)
- Gathered through discussions with stakeholders (customers, domain experts, marketing team, industry regulators…)

Importance of Requirements

"The hardest single part of building a software system is deciding precisely what to build...No other part of the work so cripples the resulting system if done wrong." -- Fred Brooks, Mythical Man Month (1975)

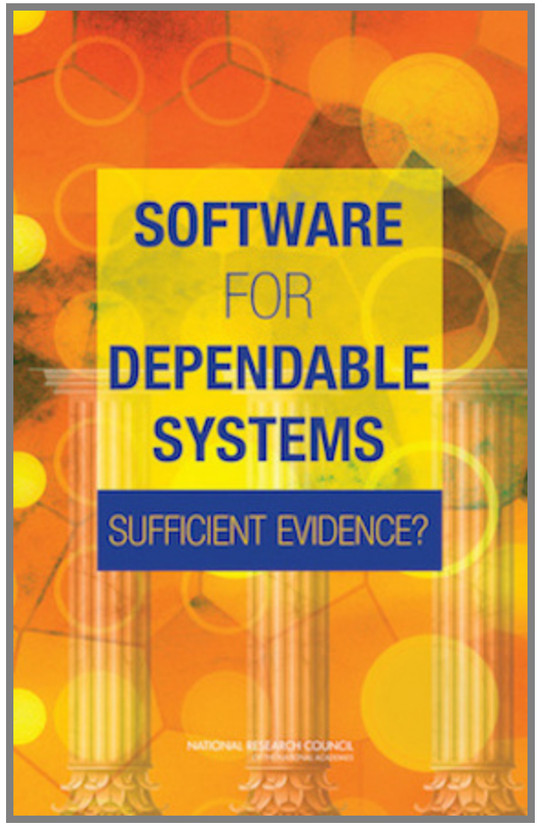


Importance of Requirements

An investigation of software-related failures by the National Research Council in the US (2007)

Bugs in code account only for 3% of fatal software accidents

Most failures due to **poor understanding of requirements** or usability issues



Urge to start coding...

Developers tend to focus on writing code...

Often ignore requirements...

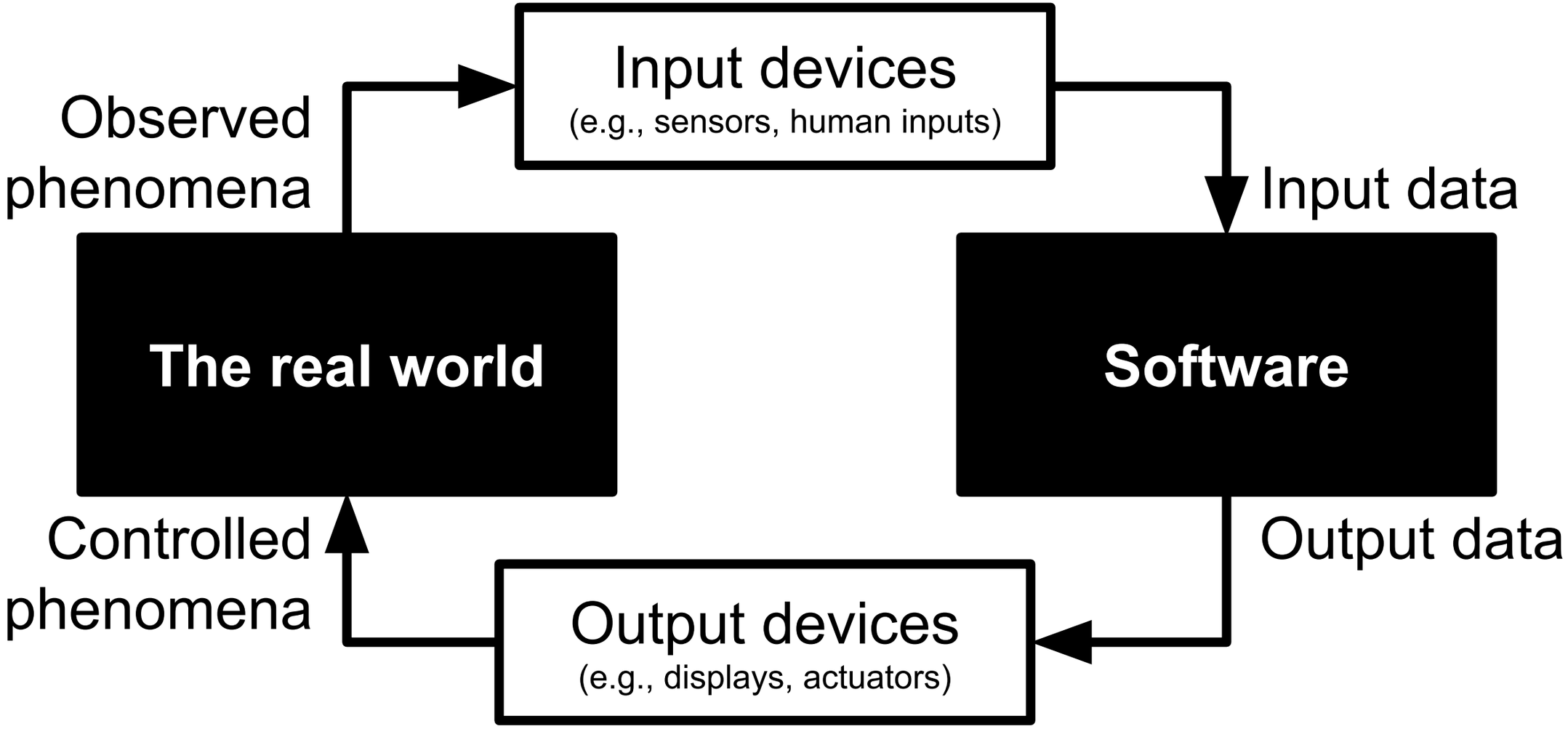
Too much effort, busywork only, distracts from coding...

Facing costly problems later... (built the wrong system)

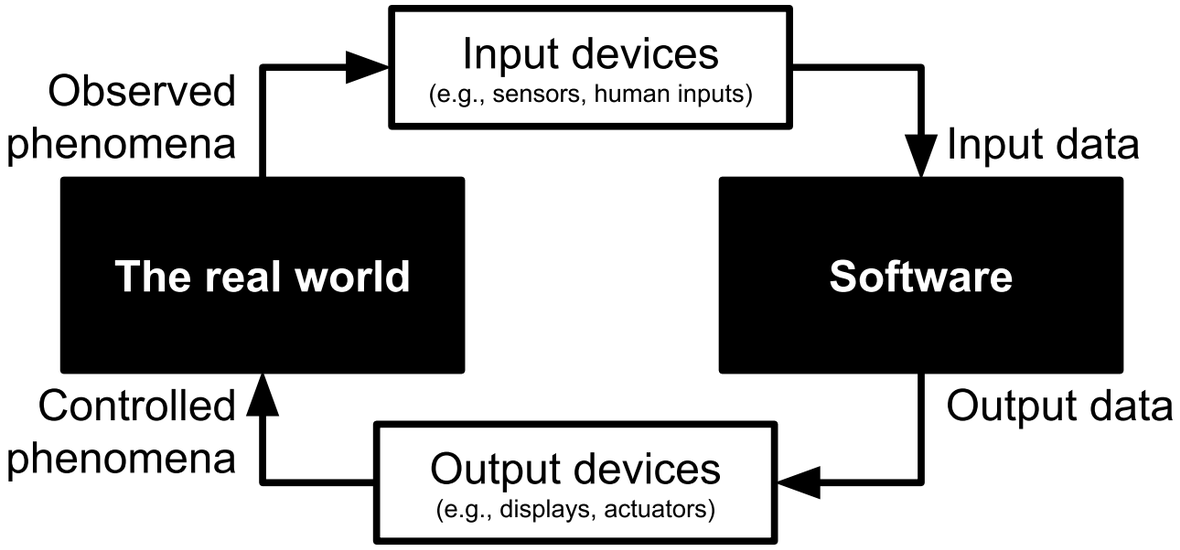
Requirements & Design: **Think before coding**

Untangling Requirements

Machine vs World



Machine vs World



Machine vs World: Fall Detection

- What are elements of the environment?
- What are the goals/requirements of the software in the real world?

(Smartwatch-based fall detection and emergency response)

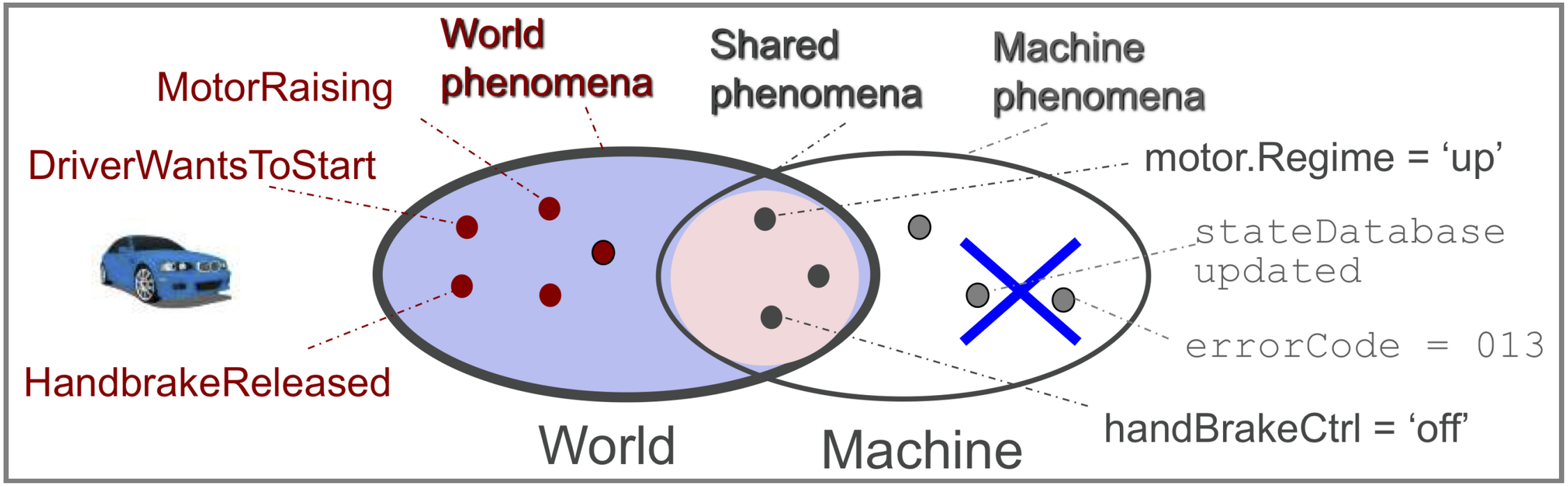
Machine vs World: Lane Keeping Assist

What are the goals/requirements of the software in the real world?

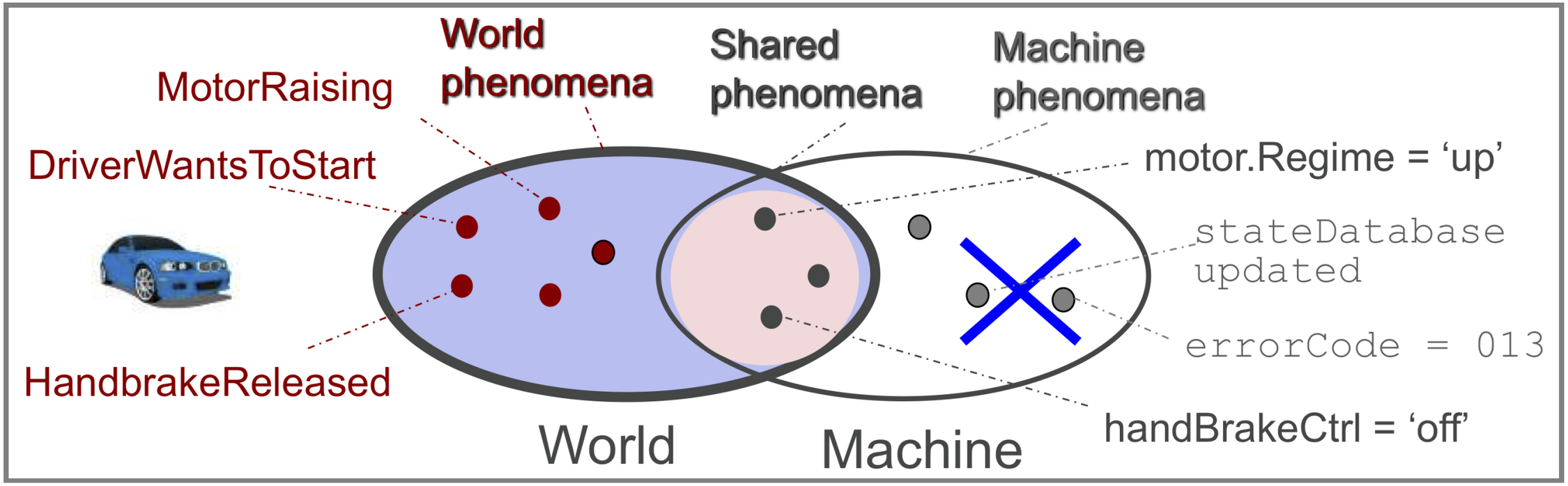
Speaker notes

Requirement: The vehicle must be prevented from veering off the lane.

Shared Phenomena



Shared Phenomena

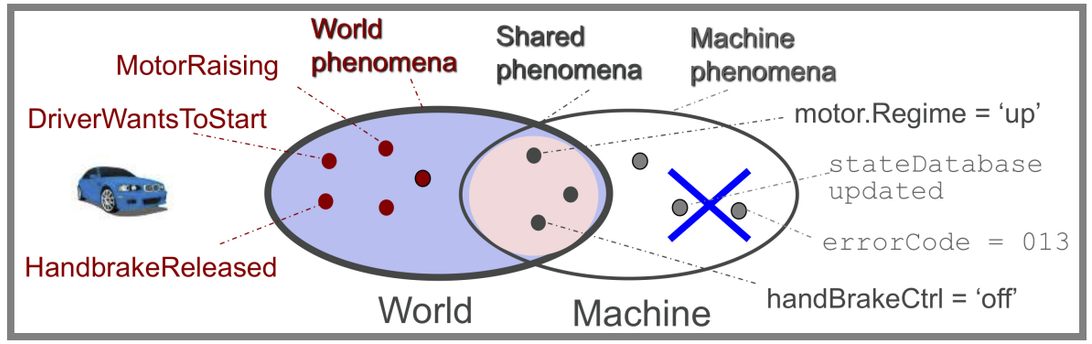


Requirement vs Specification

- System Requirement (REQ): What the system must ensure, in terms of desired effects on the environment
- Software Specification (SPEC): What software must implement, expressed over the shared phenomena
- Assumptions (ASM): What's assumed about the behavior/properties of the environment; bridges the gap between REQ and SPEC

Formally: $ASM \wedge SPEC \models REQ$

Shared Phenomena

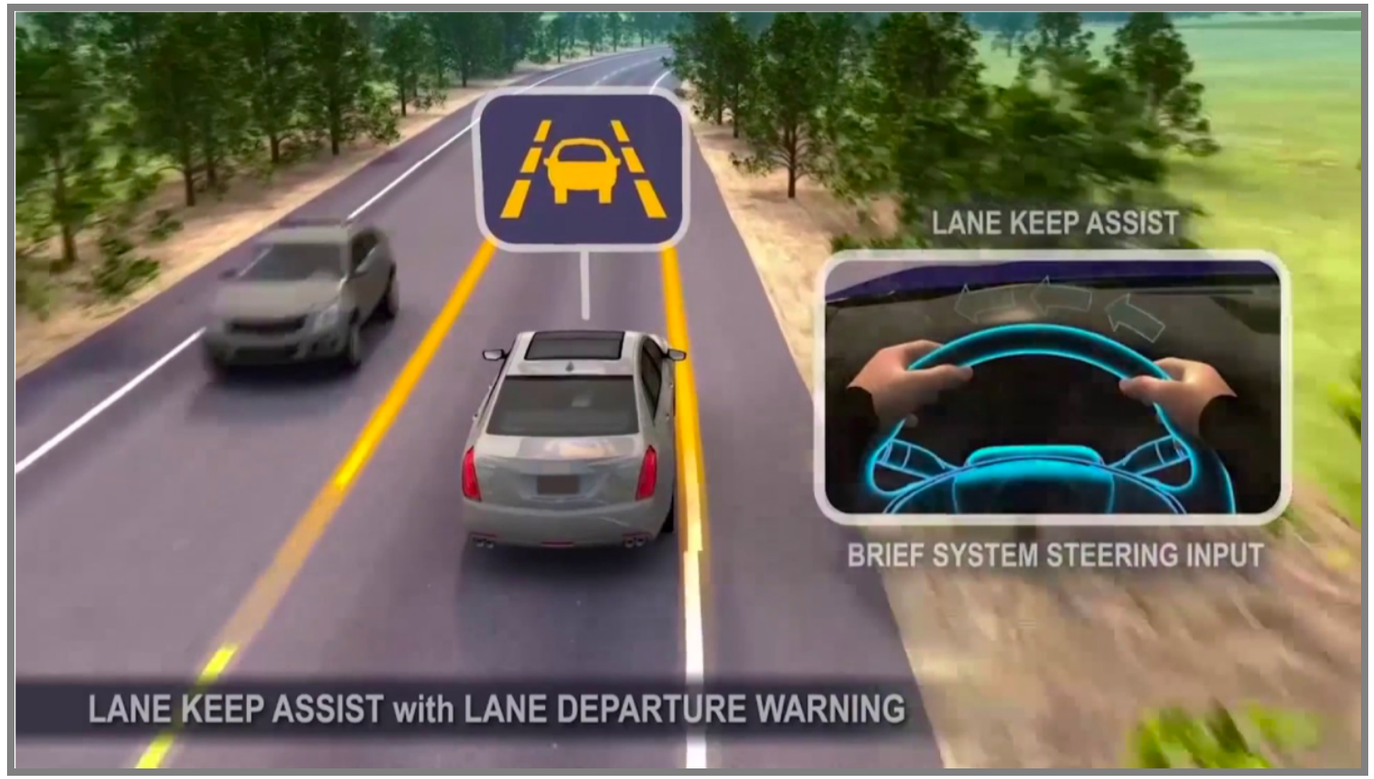


Requirements (REQ) are expressed only in terms of world phenomena

Assumptions (ASM) expressed in terms of world & shared phenomena

Specifications (SPEC) are expressed in terms of shared phenomena

Lane Assist Specification



- Requirement (REQ): The vehicle must be prevented from veering off the lane.
- Specification (SPEC): ??

Breakout: Lane Assist Assumptions

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

Discuss with your neighbor to come up with 2-3 assumptions

Example Assumptions for Lane Assist

Sensors are providing accurate information about the lane

Driver responses when given warning

Steering wheel is functional

...

What could go wrong?

Recall: ASM ∧ SPEC ⊨ REQ

1. Wrong, inconsistent or infeasible requirements (REQ)
2. Missing or incorrect environmental assumptions (ASM)
3. Wrong or violated specification (SPEC)
4. Inconsistency in assumptions & spec (ENV ∧ SPEC = False)

Example each for lane assist?

Lufthansa 2904 Runaway Crash

Lufthansa 2904 Runaway Crash

Reverse thrust: Decelerates plane after landing

REQ: Reverse thrust is enabled if and only if plane is on the ground

SPEC: Reverse thrust is enabled if and only if wheel is turning

- *if (a) 6.3 tons of weight are sensed on each landing gear or (b) sensors indicate the wheels are turning faster than 72 knots*

ASM: Wheel is turning if and only if plane on the ground

ASM: High amounts of weight are only on both landing gears if the plan is on the ground

Lufthansa 2904 Runaway Crash

Lufthansa 2904 Runaway Crash

REQ: Reverse thrust is enabled if and only if plane is on the ground

SPEC: Reverse thrust is enabled if and only if wheel is turning

ASM: Wheel is turning if and only if plane on the ground

On that day, runway was wet due to rain!

- Wheel fails to turn, even though the plane is on the ground (assumption violated)
- Pilot attempts to enable RT; overridden by the software
- Plane goes off the runway and crashes!

Assumption Violations in ML-based Systems (1)

Assumptions about correctness of model predictions?

Assumptions of human behavior? Interaction with the system?

Assumptions about training data?

Assumptions about stability of data? about reliability of sensors? reliability of human input?

Assumption Violations in ML-based Systems (1)

Unrealistic or missing assumptions

- e.g., poorly understood effect of weather conditions on sensor accuracy, missing pedestrian behavior

Concept drift

- Environment evolves over time; underlying distribution changes
- e.g. user's preferences on products
- (More on this in the data quality lecture)

Assumption Violations in ML-based Systems (2)

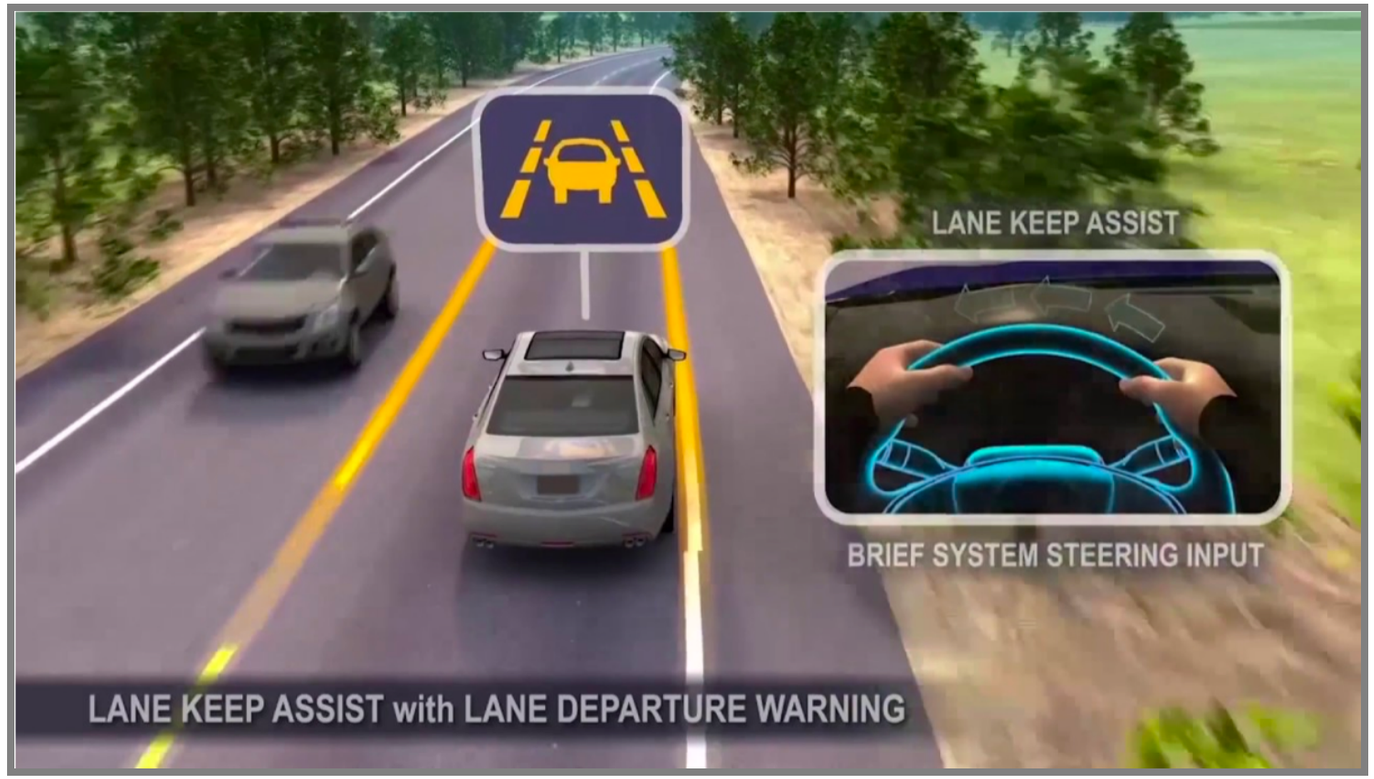
Adversaries

- A malicious actor deliberately tries to violate assumptions
- e.g., adversarial attacks on stop signs
- (More in the security lecture)

Feedback loops

- System repeatedly acts on and changes the environment over time; earlier assumptions may cease to hold
- e.g., predictive policing

Recall: Lane Assist



- REQ: The vehicle must be prevented from veering off the lane.
- ASM: Sensors are providing accurate information about the lane; driver responses when given warning; steering wheel is functional

What could go wrong in lane assist?

REQ: The vehicle must be prevented from veering off the lane.
ASM: Sensors are providing accurate information about the lane; driver responses when given warning; steering wheel is functional

Missing or incorrect environmental assumptions (ASM)?

- Concept drift? Adversaries?

Wrong or violated specification (SPEC)?

Process for Establishing Requirements

1. Identify environmental entities and machine components
2. State a desired requirement (REQ) over the environment
3. Identify the interface between the environment & machine
4. Identify the environmental assumptions (ENV)
5. Develop specifications (SPEC) that are sufficient to establish REQ
6. Check whether ENV $\wedge$ SPEC $\models$ REQ
7. If not, go back to the beginning & repeat

Breakout Session: Fall detection

As a group, post answer to `#lecture` and tag group members:

Requirement: ...
Assumptions: ...
Specification: ...
What can go wrong: ...

For completeness: Behavioral vs quality requirements

Behavioral requirements (functional requirements)

- What the system shall do
- How inputs and outputs relate
- ... typically clear 'correctness' specifications

Quality requirements (non-functional requirements)

- How the system should operate and be built
- Development budget and deadlines
- Code quality, maintainability, extensibility requirements
- Latency, scalability, throughput requirements
- Safety, security, fairness requirements
- Usability requirements
- ... all require measurement

Takeaway

- Software/ML models alone cannot fulfill system requirements
  - They are just one part of the system, and have limited control over the environment
- Environmental assumptions are just as critical in achieving requirements
  - If you ignore/misunderstand these, your system may fail or do poorly (no matter how good your model is)
  - Identify and document these assumptions as early as possible!
  - Some of the assumptions may be violated over time as the environment changes -- Monitor these assumptions and adjust your specification accordingly

Gathering and Negotiating Requirements

Understanding requirements is hard

- Customers don't know what they want until they see it
- Customers change their mind ("no, not like that")
- Descriptions are vague
- It is easy to ignore important requirements (privacy, fairness)
- Focused too narrowly on needs of few users
- Engineers think they already know the requirements
- Engineers are overly influenced by technical capability
- Engineers prefer elegant abstractions

Examples?

See also Jackson, Michael. "The world and the machine." In Proceedings of the International Conference on Software Engineering. IEEE, 1995.

Start with Stakeholders…

Stakeholders: *all persons and entities who have an interest in a project or who may be affected by the project*

Not only direct customers and users, also affected people, owners, developers, operators, regulators, …

All may have needs, preferences, or concerns…

Start creating a list of all possible stakeholders

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

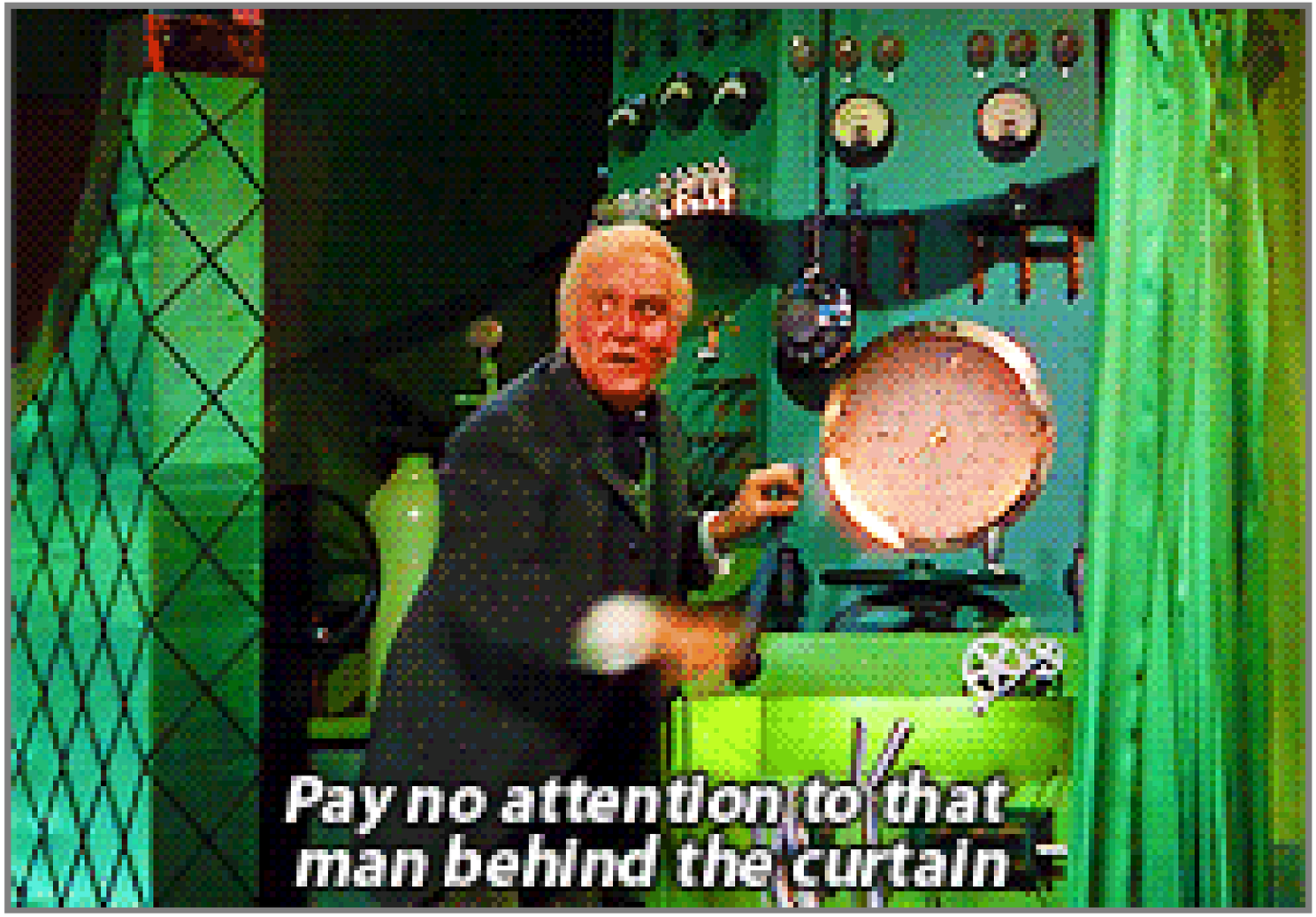
Requirements elicitation techniques

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



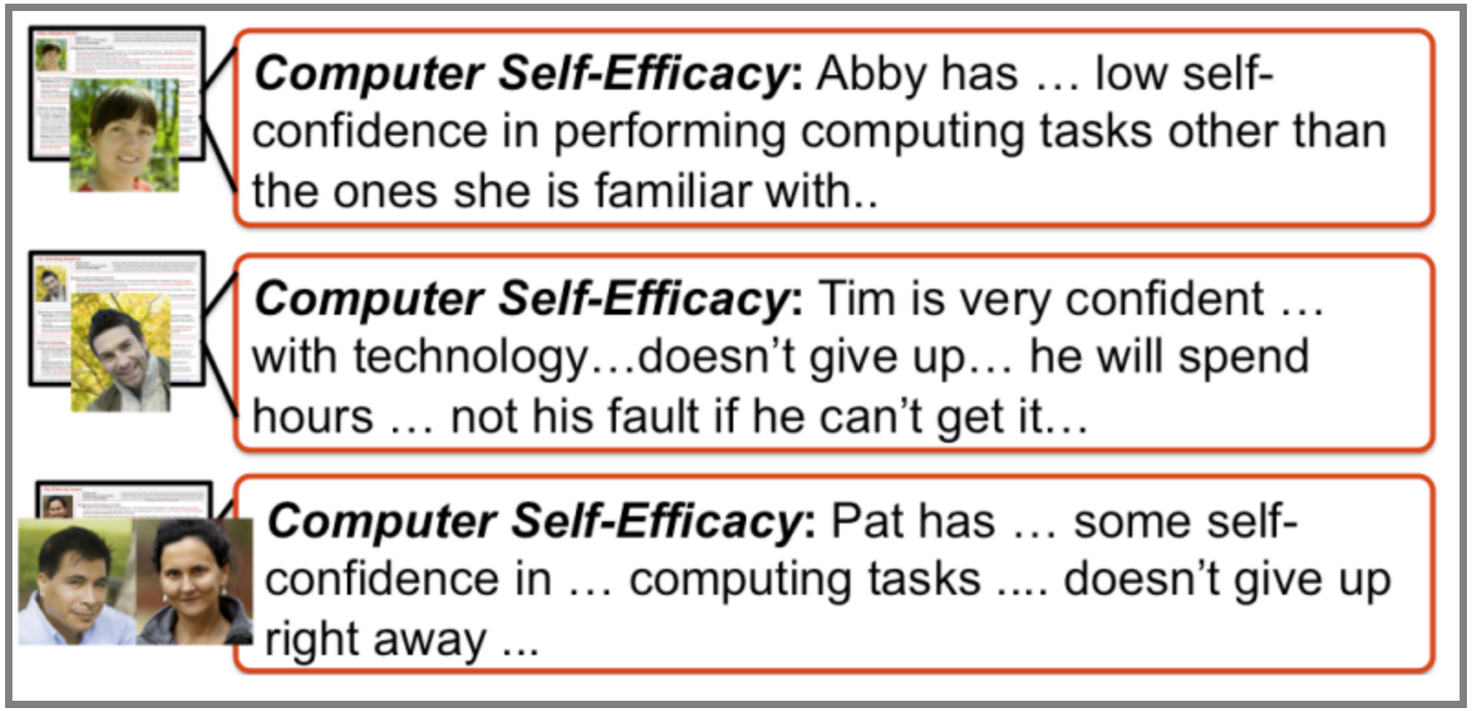
Speaker notes

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

Personas in GenderMag



See examples and details http://gendermag.org/foundations.php

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

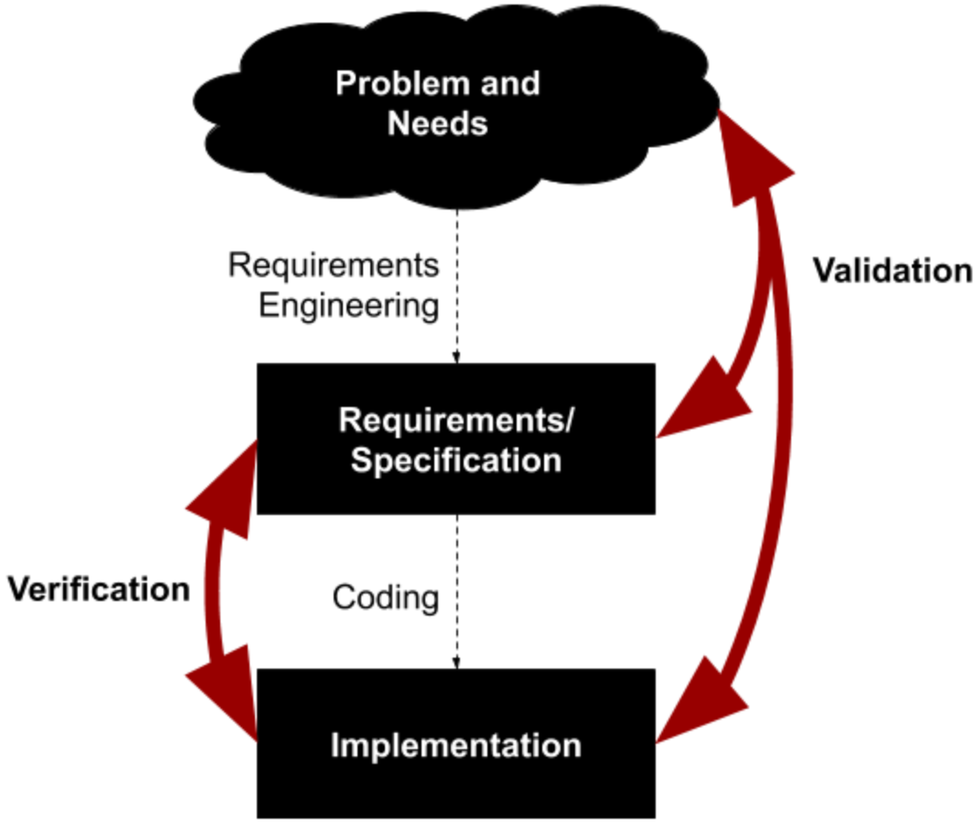
- what the software *shall* do, what it *shall* not do, what qualities it *shall* have,
- document decisions and rationale for conflict resolution

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

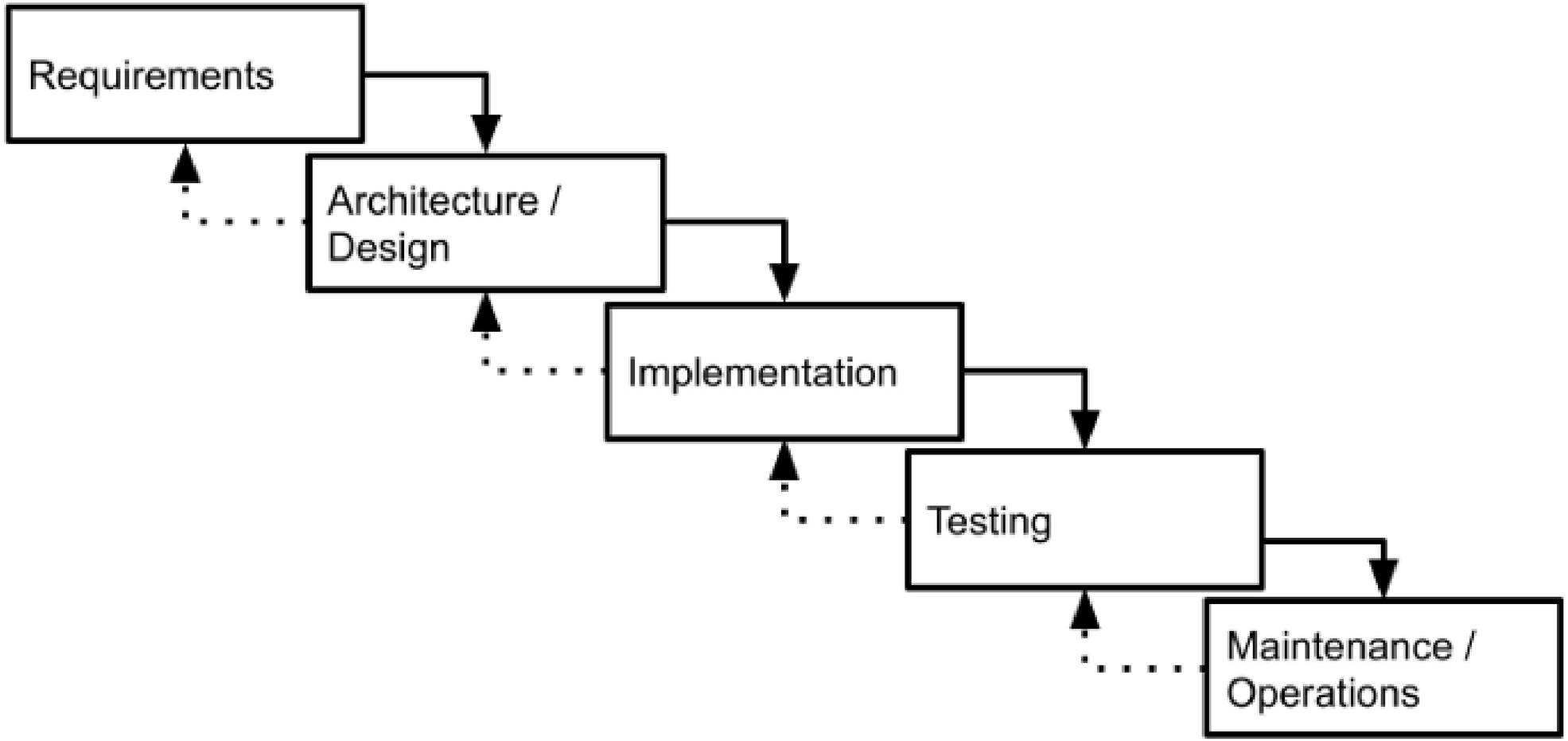
Show prototype to stakeholders

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

Further Reading

- Van Lamsweerde, Axel. Requirements engineering: From system goals to UML models to software. John Wiley & Sons, 2009.
- Vogelsang, Andreas, and Markus Borg. "Requirements Engineering for Machine Learning: Perspectives from Data Scientists." In Proc. of the 6th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), 2019.
- Rahimi, Mona, Jin LC Guo, Sahar Kokaly, and Marsha Chechik. "Toward Requirements Specification for Machine-Learned Components." In 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), pp. 241-244. IEEE, 2019.
- Kulynych, Bogdan, Rebekah Overdorf, Carmela Troncoso, and Seda Gürses. "POTs: protective optimization technologies." In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 177-188. 2020.
- Wiens, Jenna, Suchi Saria, Mark Sendak, Marzyeh Ghassemi, Vincent X. Liu, Finale Doshi-Velez, Kenneth Jung et al. "Do no harm: a roadmap for responsible machine learning for health care." Nature medicine 25, no. 9 (2019): 1337-1340.
- Bietti, Elettra. "From ethics washing to ethics bashing: a view on tech ethics from within moral philosophy." In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 210-219. 2020.
- Guizani, Mariam, Lara Letaw, Margaret Burnett, and Anita Sarma. "Gender inclusivity as a quality requirement: Practices and pitfalls." IEEE Software 37, no. 6 (2020).