17-423/723: Software System Design

Design Reviews

Feb 23, 2026

Logistics

- M2 due on Friday (Feb 27)
- No recitation this week; project work time

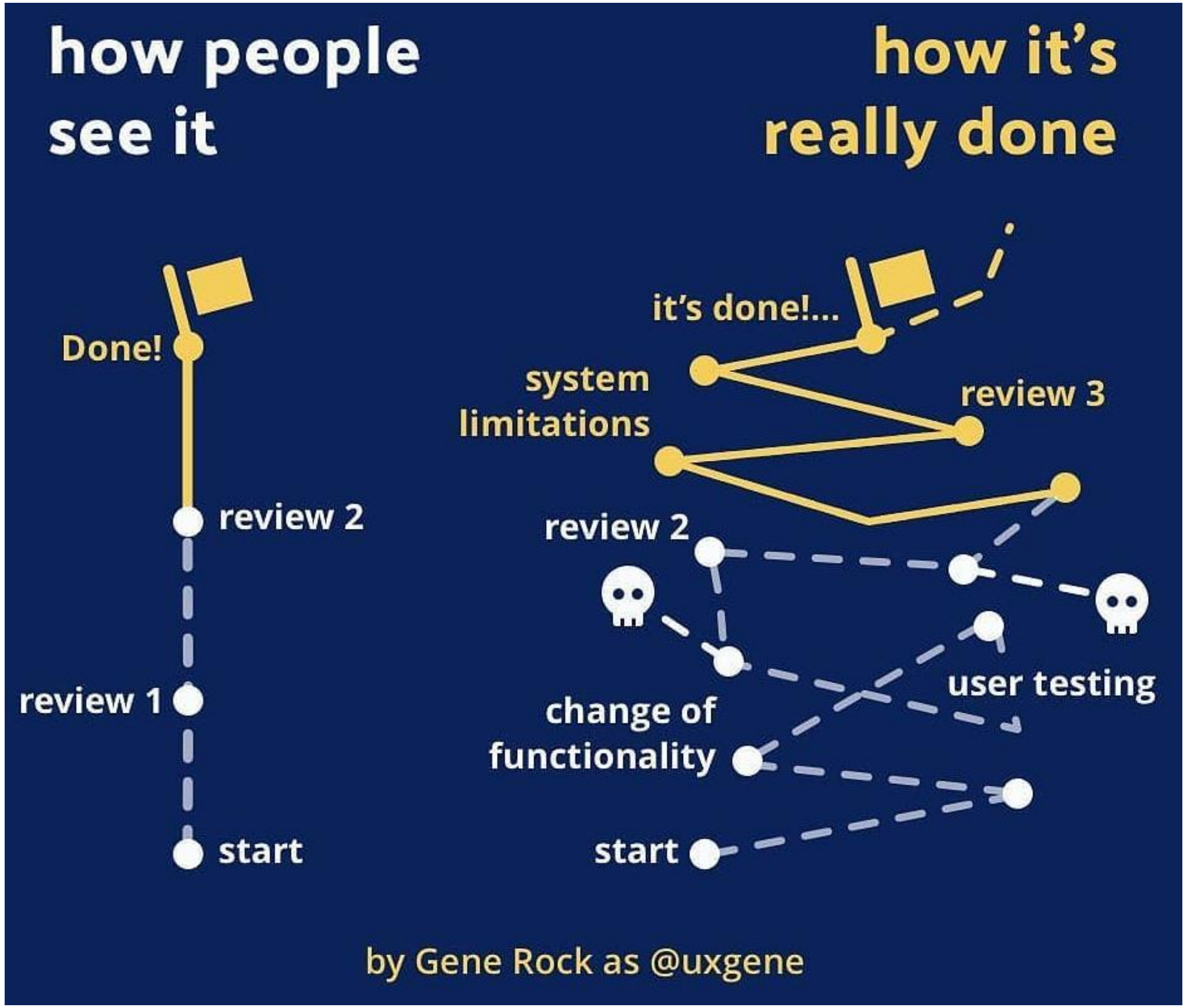
Leaning Goals

- Devise and document an argument for ***why*** the design achieves a desired function or quality attribute
- Distinguish between intellectual vs. statistical control and explain the roles that they play in assurance cases
- Review and identify weaknesses in an existing design argument

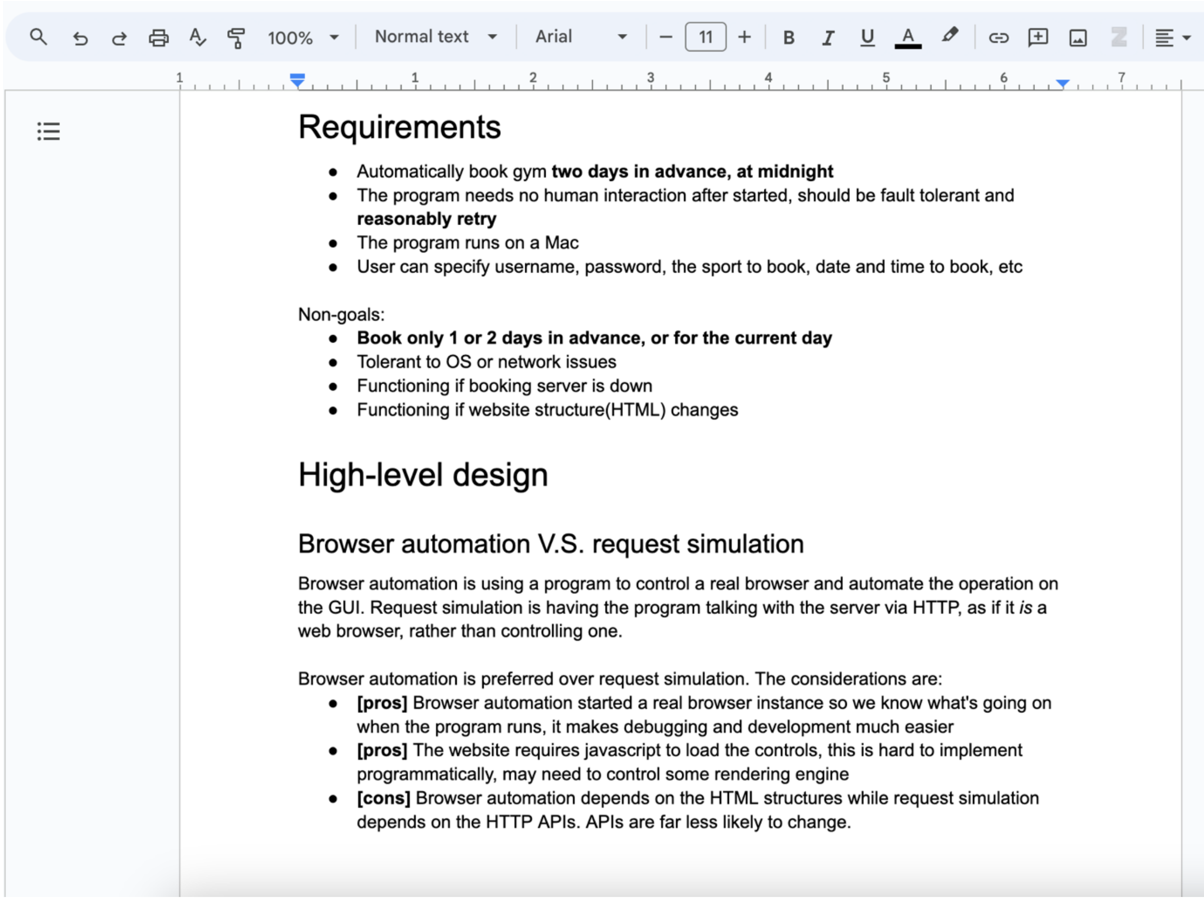
Design Review

- An activity for evaluating a design against system requirements
  - Check whether a product (designed or implemented) achieves its expected functionality and quality attributes
  - Identify potential issues to be addressed
- An important part of a software development process in practice
- **Not the same as code review!**
  - Design review: Focus is on higher-level design decisions
  - Code review: Focus is on the quality of the source code (e.g., correctness, readability, security vulnerabilities, etc.,)

Design Reviews in Practice



Design Reviews in Practice: Google



Requirements

- Automatically book gym **two days in advance, at midnight**
- The program needs no human interaction after started, should be fault tolerant and **reasonably retry**
- The program runs on a Mac
- User can specify username, password, the sport to book, date and time to book, etc

Non-goals:

- **Book only 1 or 2 days in advance, or for the current day**
- Tolerant to OS or network issues
- Functioning if booking server is down
- Functioning if website structure(HTML) changes

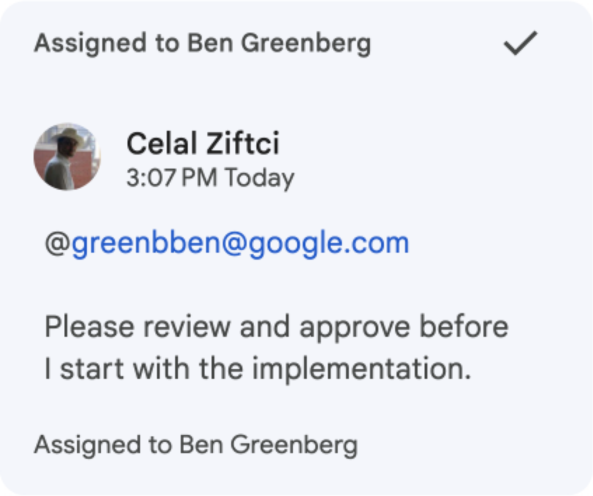
High-level design

Browser automation V.S. request simulation

Browser automation is using a program to control a real browser and automate the operation on the GUI. Request simulation is having the program talking with the server via HTTP, as if it *is* a web browser, rather than controlling one.

Browser automation is preferred over request simulation. The considerations are:

- **[pros]** Browser automation started a real browser instance so we know what's going on when the program runs, it makes debugging and development much easier
- **[pros]** The website requires javascript to load the controls, this is hard to implement programmatically, may need to control some rendering engine
- **[cons]** Browser automation depends on the HTML structures while request simulation depends on the HTTP APIs. APIs are far less likely to change.



Assigned to Ben Greenberg ✓

Celal Ziftci
3:07 PM Today

@greenbben@google.com

Please review and approve before I start with the implementation.

Assigned to Ben Greenberg

- Widely performed at Google
- Design docs are written using Google Docs
- Stakeholders leave comments directly on the docs

Improving Design Reviews at Google.
Ziftci & Greenberg. IEEE/ACM ASE (2023).

Request for Comments (RFC)

- A common type of document used for design proposals and reviews
- Describes a design proposal/decision, why it is needed (i.e., goals), how it works, and alternatives considered
- Frequently used by technical committees for network protocols and standards (e.g., HTTP, TCP/IP, OAuth…)
- But also used within tech organizations to document and review major design/product proposals
- Examples

Challenges with Design Reviews

REVIEWS 10 LINES OF CODE: FINDS 10 ISSUES.
REVIEWS 500 LINES OF CODE: “LOOKS GOOD TO ME”
made with mematic

Documenting for Design Reviews

- Code is a poor abstraction for understanding why/how design works
- To facilitate a design review, design decisions must be documented
- We have already discussed different notations for documenting a design:
  - Context (domain) models
  - Component diagrams
  - Data models
  - Sequence diagrams
- But these notations don’t explicitly say **why** the design decisions were made, and **how** they support the system in achieving desired quality

Today’s Class

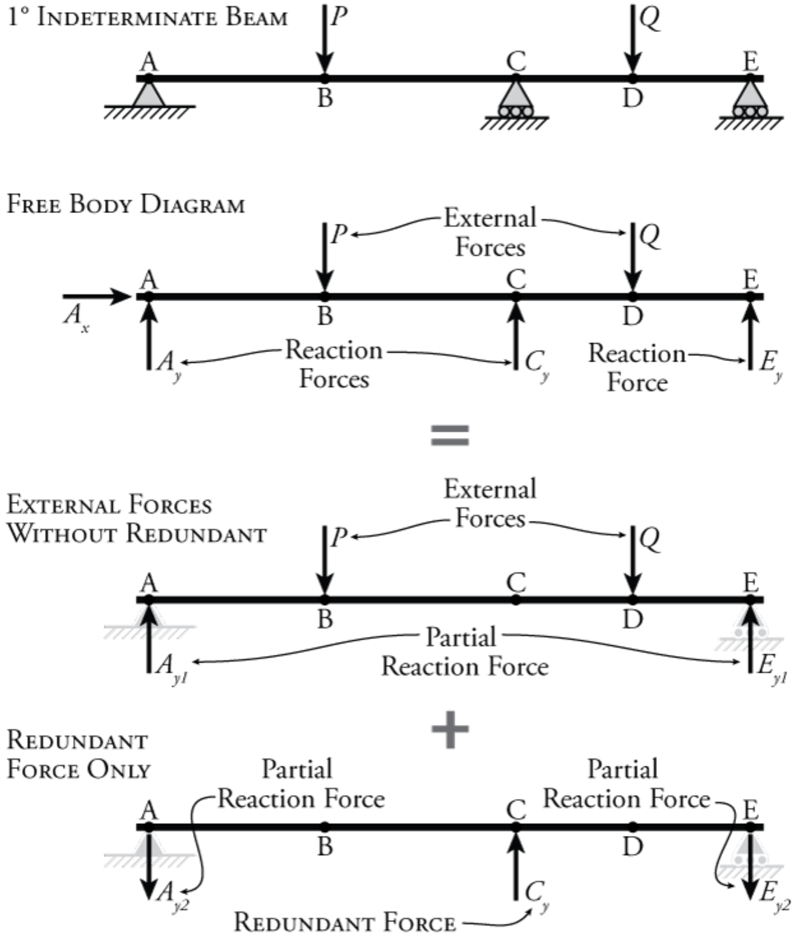
- **Design argumentation**: Devising and documenting an explicit **argument** for why the system design achieves its expected functionality
- **Design review**: Identifying weaknesses in the argument & suggesting ways to improve the design

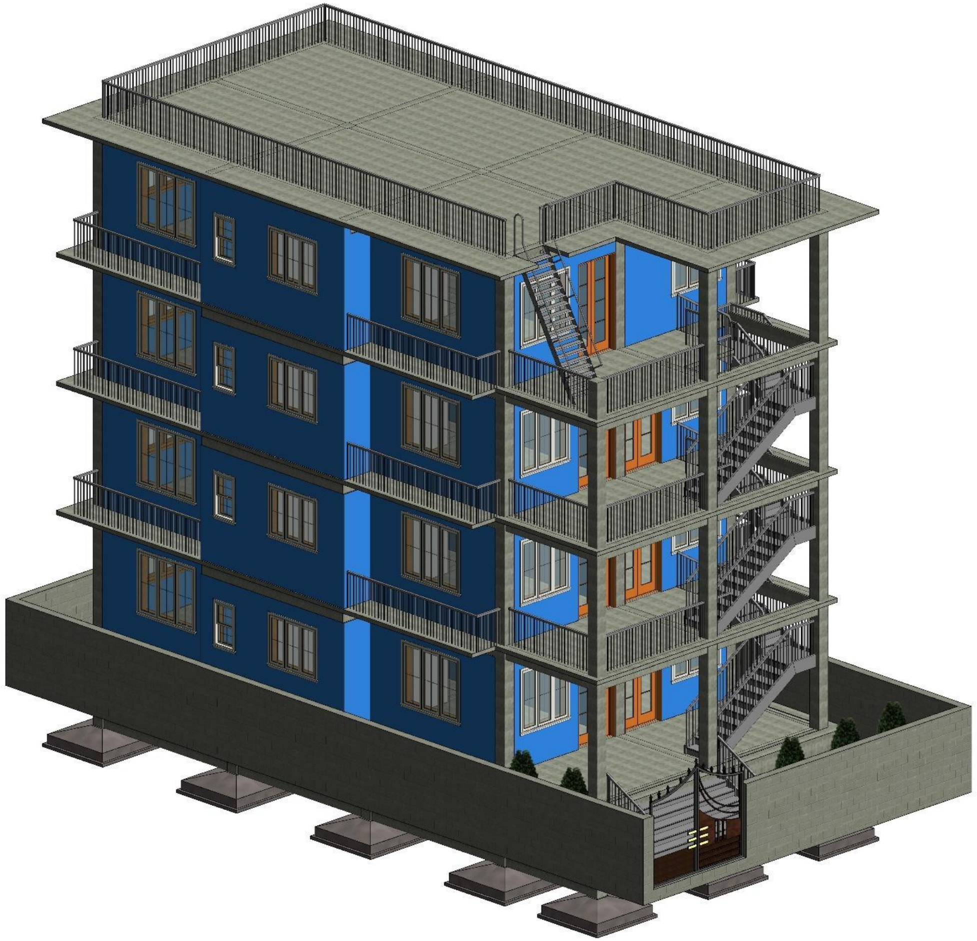
Design Arguments

Today’s Reading: Intellectual vs. Statistical Control

- Coined by George Fairbanks (software engineer & design educator at Google; former CMU grad)
- Two approaches to gaining confidence that your system “works” (i.e., it satisfies its functional & QA requirements)
- **Intellectual control**: Developer has a mental model to understand and explain why and how the system works (without running it)
- **Statistical control**: Developer obtains confidence through generating empirical evidence (through testing, static analysis, etc.,)
- **Q. Which one of these is more prominent in practice?**
- **Q. What are the risks of relying exclusively on one form of control?**

Arguing why your design works



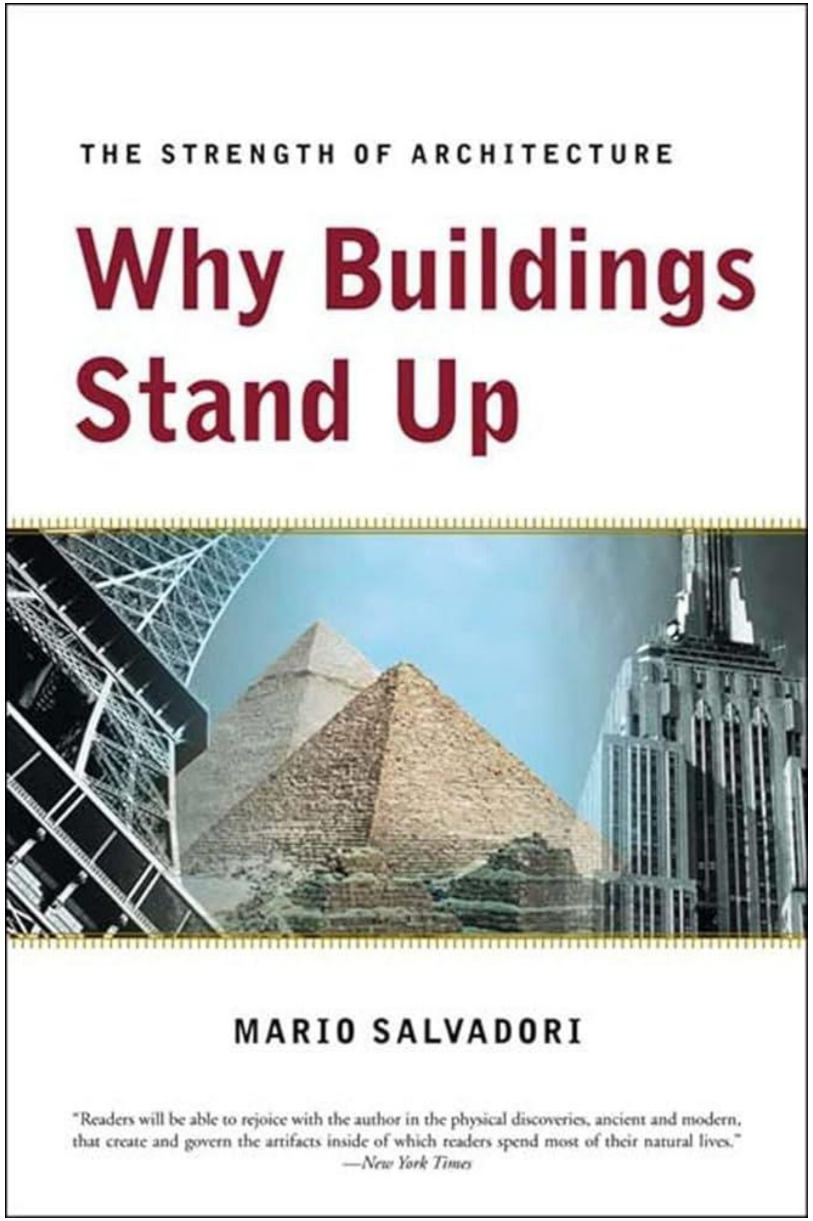


But in software…

“Software is like a cathedral; first we build it, and then we pray.” – Sam Redwine

Design Argumentation

- **Goal**: Argue “why my design works”
- An argument is often implicit and incomplete in the designer’s mind
- If you can’t produce a strong argument, how do you know that your system works?
- Allow another person to review & identify weaknesses in the argument
- **One approach**: Assurance case
    - **Assurance**: The process of demonstrating that the system will function and satisfy its quality attributes as intended

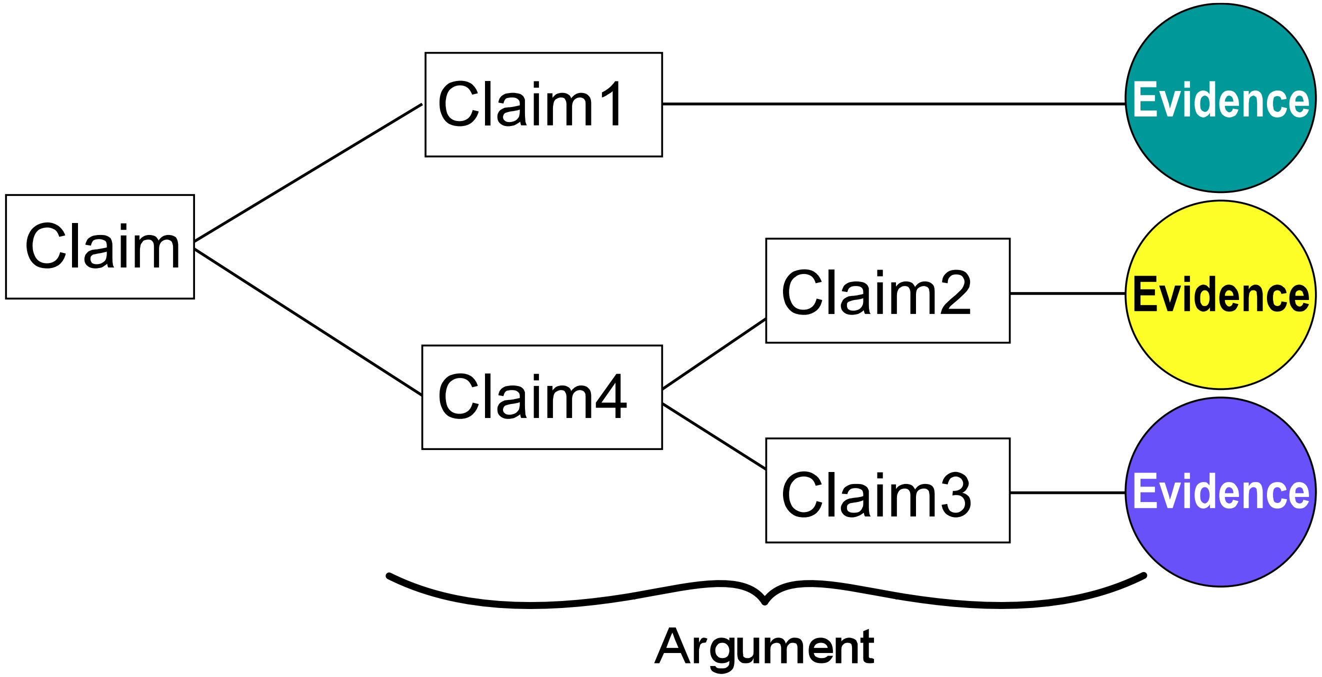


"Readers will be able to rejoice with the author in the physical discoveries, ancient and modern, that create and govern the artifacts inside of which readers spend most of their natural lives."
—*New York Times*

Assurance Case

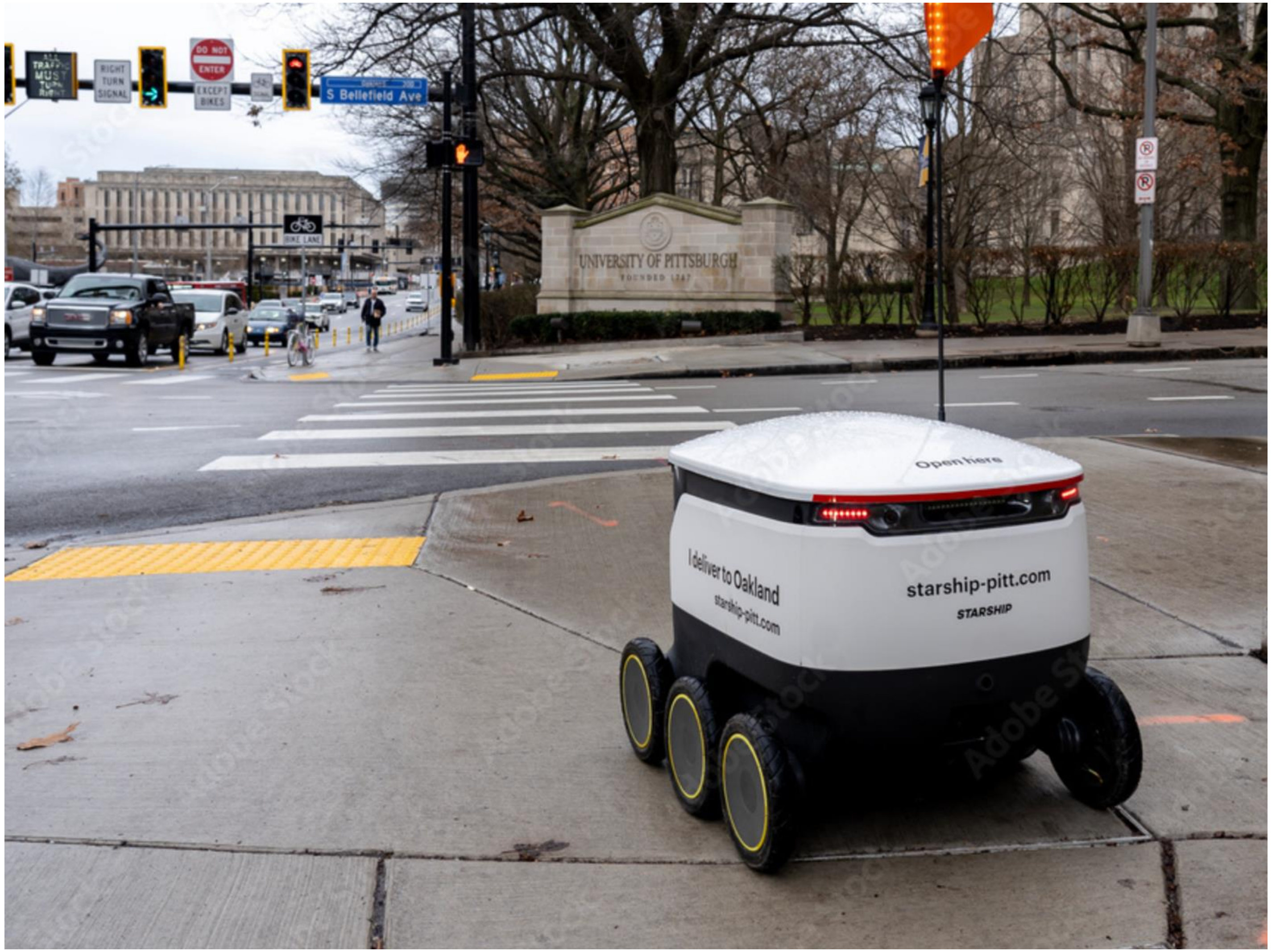
- An explicit argument that a system achieves a desired requirement, along with supporting evidence
- **Claim:** A statement about a piece of functionality or quality attribute of the system
- **Argument**: A top-level claim decomposed into multiple, hierarchical **subclaims**
- **Evidence**: A documented piece of evidence that supports a leaf subclaim
  - Results of testing, software analysis, formal verification, inspection, expert opinions, architecture design
  - Must be **auditable** & **verifiable** independently by a third party

Assurance Case: Structure



IF 🟢 **THEN** Claim1; **IF** 🟡 **THEN** Claim2; **IF** 🟣 **THEN** Claim3;
IF Claim2 **and** Claim3 **THEN** Claim4; **IF** Claim1 **and** Claim4 **THEN** Claim

Example: Sidewalk Delivery Robot



Building an Assurance Case

1. Identify a **top-level claim** to demonstrate: A statement about a piece of desired functionality or a quality attribute
    - The intrusion detection system notifies the homeowner in time when a stranger appears around the house (**functionality**)
    - The movie streaming app delivers its content at 1080p resolution with less than 1 second buffering event (**performance**)
    - The stock tracker app can be extended with new types of output format without impacting the rest of functionality (**changeability**)
    - The sidewalk robot avoids collision with pedestrians (**safety**)

Assurance Case: Delivery Robot

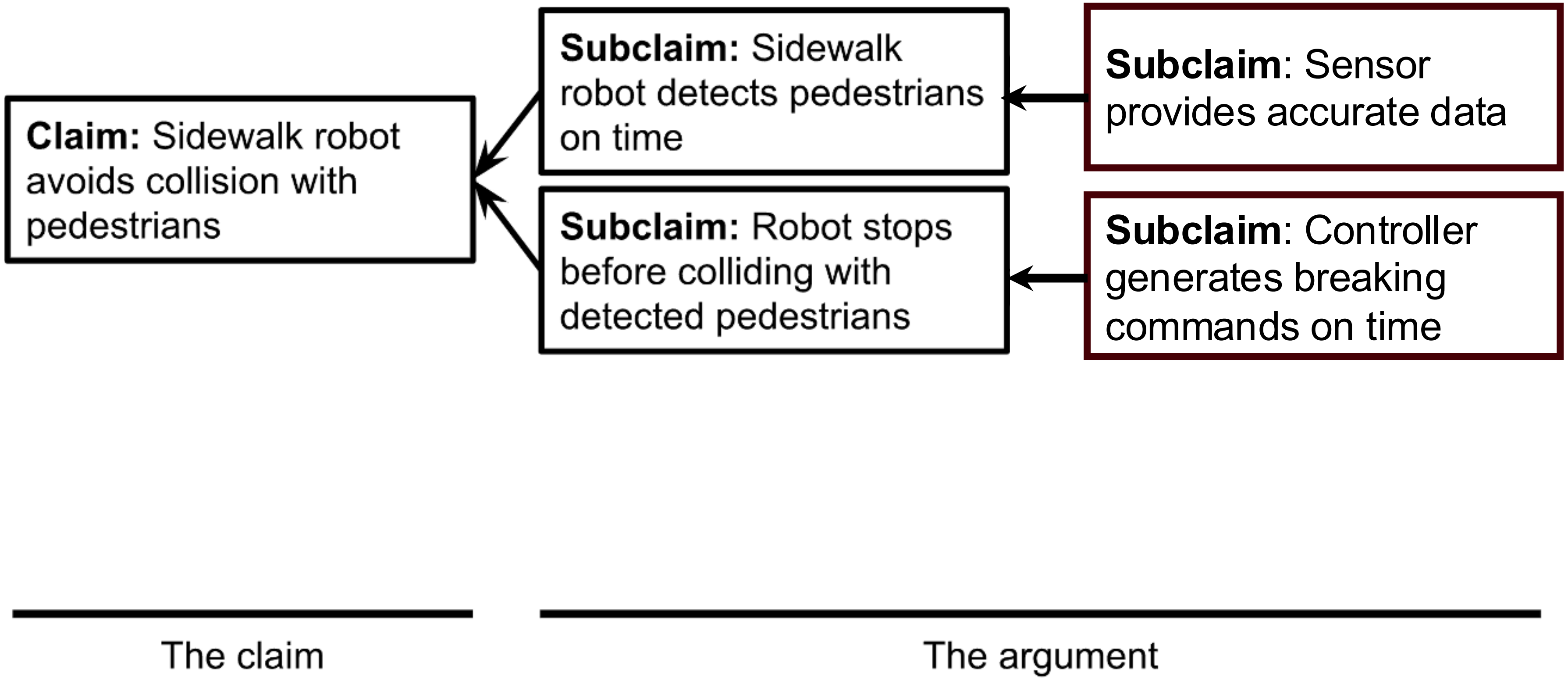
Claim: Sidewalk robot avoids collision with pedestrians

The claim

Building an Assurance Case

1. Identify a **top-level claim** to demonstrate: A statement about a piece of desired functionality or a quality attribute
2. Identify one or more **subclaims** to support a higher-level claim.
   - Logically, "If all the subclaims hold, then their parent claim also holds”
   - Each subclaim can, in turn, be decomposed into further subclaims
   - Each leaf-level subclaim describes (1) the responsibility of a software component or (2) an assumption about a domain entity

Assurance Case: Delivery Robot



Claim: Sidewalk robot avoids collision with pedestrians

Subclaim: Sidewalk robot detects pedestrians on time

Subclaim: Robot stops before colliding with detected pedestrians

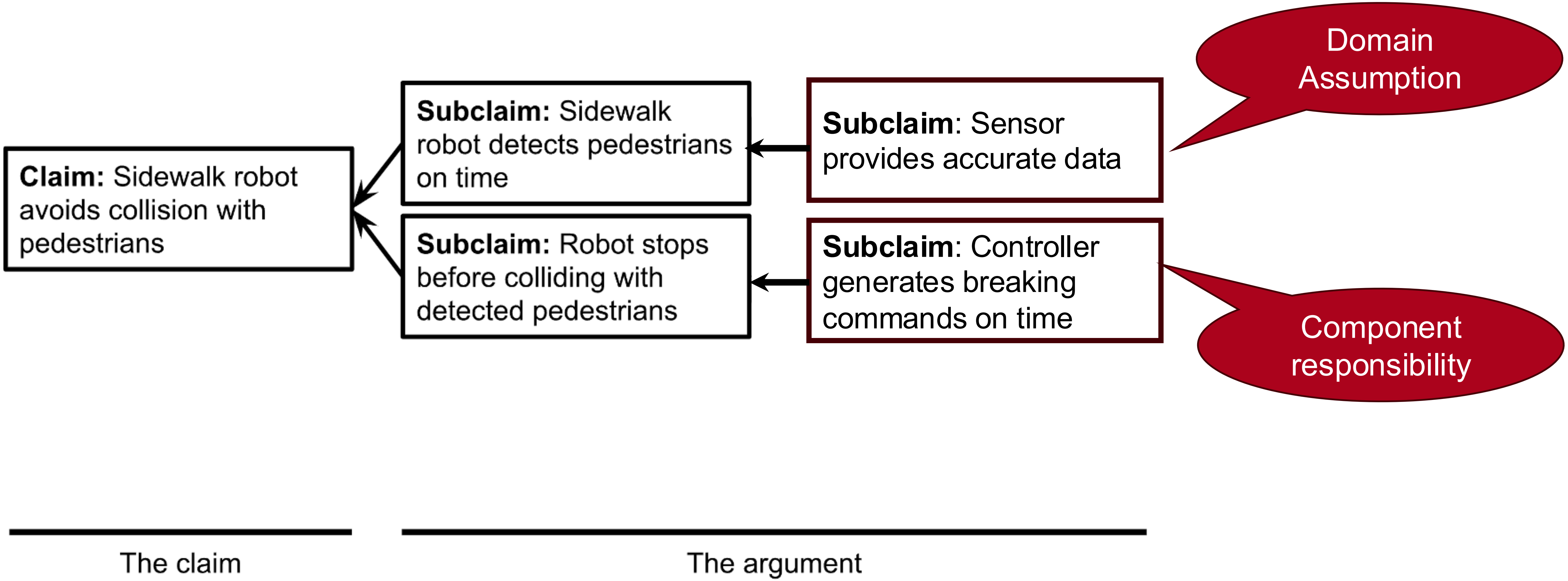
Subclaim: Sensor provides accurate data

Subclaim: Controller generates breaking commands on time

The claim

The argument

Assurance Case: Delivery Robot



Claim: Sidewalk robot avoids collision with pedestrians

Subclaim: Sidewalk robot detects pedestrians on time

Subclaim: Robot stops before colliding with detected pedestrians

Subclaim: Sensor provides accurate data

Subclaim: Controller generates breaking commands on time

Domain Assumption

Component responsibility

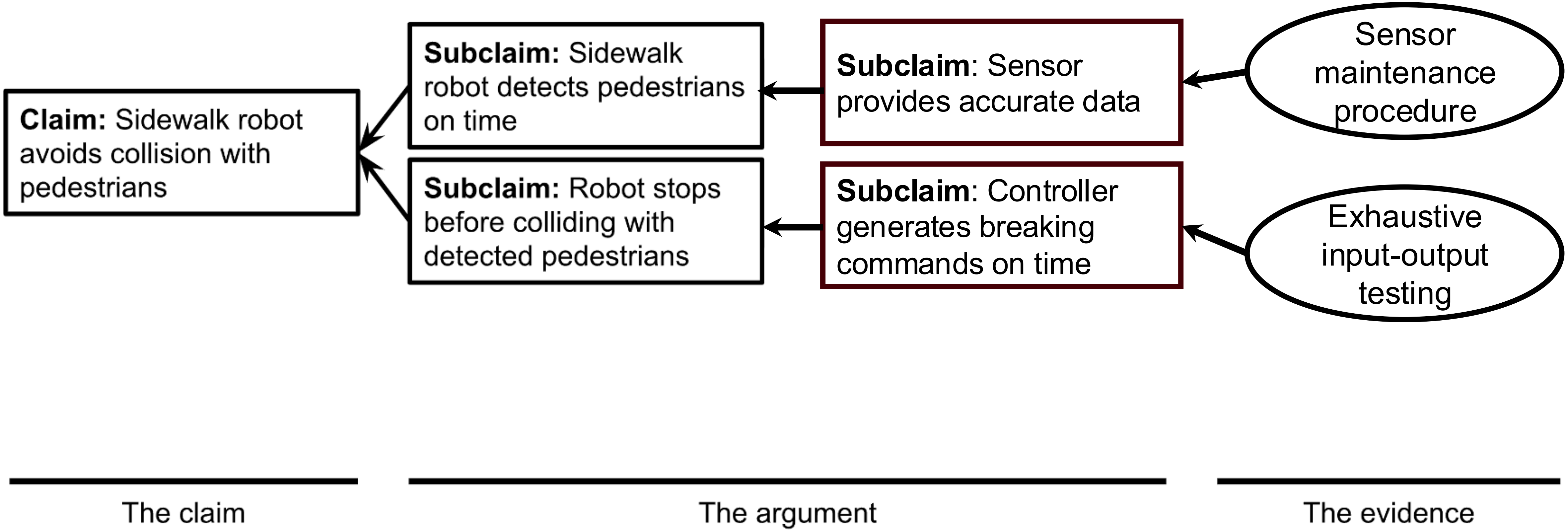
The claim

The argument

Building an Assurance Case

1. Identify a **top-level claim** to demonstrate: A statement about a piece of desired functionality or a quality attribute
2. Identify one or more **subclaims** to support a higher-level claim.
3. For each leaf-level subclaim, provide a piece of evidence to support the claim
   - **Results of testing or program analysis** (e.g., “The app successfully handled stress testing with 1,000 user requests per second”)
   - **Design decisions** (e.g., ”Backup servers are deployed in case the primary one fails” or “An interface is used to hide details about the format of a stock quote from its clients”)
   - **Empirical data** (e.g., “Based on historical data, the battery is expected to last 3 months before failing”)
   - **Procedures** (e.g., “The battery is replaced regularly by the user”)

Assurance Case: Delivery Robot



Claim: Sidewalk robot avoids collision with pedestrians

Subclaim: Sidewalk robot detects pedestrians on time

Subclaim: Robot stops before colliding with detected pedestrians

Subclaim: Sensor provides accurate data

Subclaim: Controller generates breaking commands on time

Sensor maintenance procedure

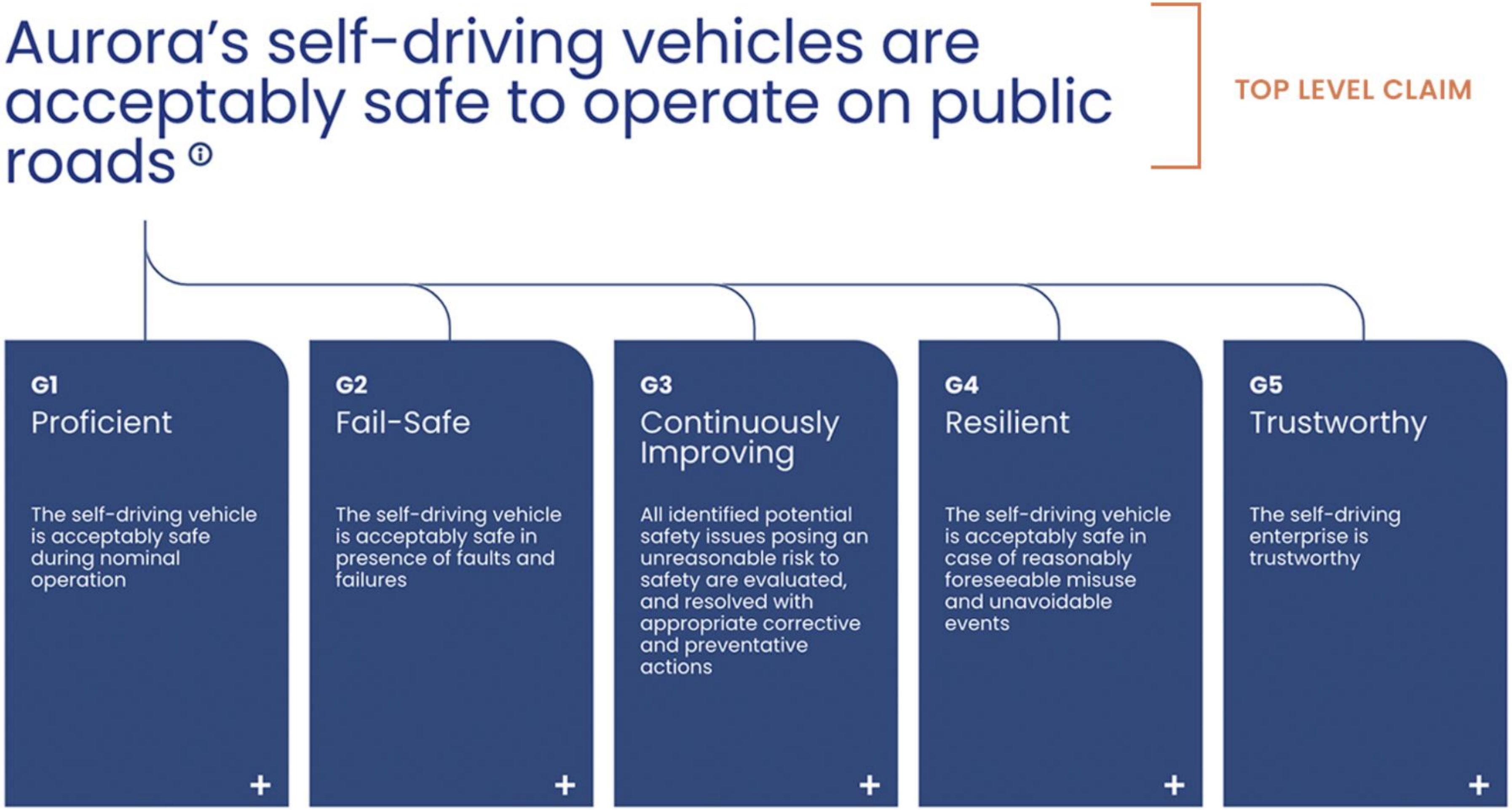
Exhaustive input-output testing

The claim

The argument

The evidence

Assurance Cases in Practice



Aurora's self-driving vehicles are acceptably safe to operate on public roads ⓘ

TOP LEVEL CLAIM

G1
Proficient

The self-driving vehicle is acceptably safe during nominal operation

+

G2
Fail-Safe

The self-driving vehicle is acceptably safe in presence of faults and failures

+

G3
Continuously Improving

All identified potential safety issues posing an unreasonable risk to safety are evaluated, and resolved with appropriate corrective and preventative actions

+

G4
Resilient

The self-driving vehicle is acceptably safe in case of reasonably foreseeable misuse and unavoidable events

+

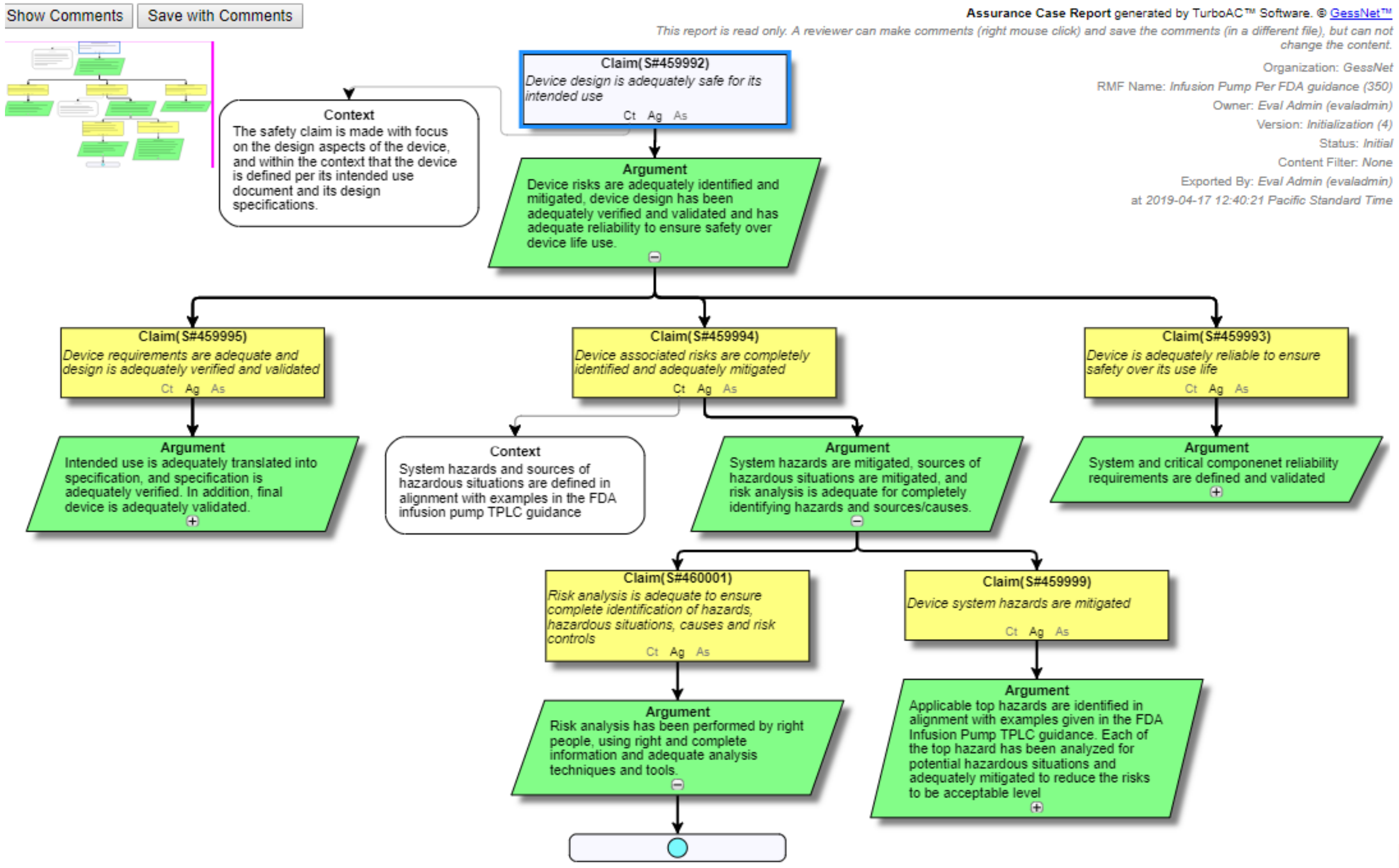
G5
Trustworthy

The self-driving enterprise is trustworthy

+

https://safetycaseframework.aurora.tech/

Assurance Cases in Practice



Assurance Case Report generated by TurboAC™ Software. © GessNet™

This report is read only. A reviewer can make comments (right mouse click) and save the comments (in a different file), but can not change the content.

Organization: *GessNet*

RMF Name: *Infusion Pump Per FDA guidance (350)*

Owner: *Eval Admin (evaladmin)*

Version: *Initialization (4)*

Status: *Initial*

Content Filter: *None*

Exported By: *Eval Admin (evaladmin)*

at *2019-04-17 12:40:21 Pacific Standard Time*

Claim(S#459992)
Device design is adequately safe for its intended use

Context
The safety claim is made with focus on the design aspects of the device, and within the context that the device is defined per its intended use document and its design specifications.

Argument
Device risks are adequately identified and mitigated, device design has been adequately verified and validated and has adequate reliability to ensure safety over device life use.

Claim(S#459995)
Device requirements are adequate and design is adequately verified and validated

Argument
Intended use is adequately translated into specification, and specification is adequately verified. In addition, final device is adequately validated.

Claim(S#459994)
Device associated risks are completely identified and adequately mitigated

Context
System hazards and sources of hazardous situations are defined in alignment with examples in the FDA infusion pump TPLC guidance

Argument
System hazards are mitigated, sources of hazardous situations are mitigated, and risk analysis is adequate for completely identifying hazards and sources/causes.

Claim(S#460001)
Risk analysis is adequate to ensure complete identification of hazards, hazardous situations, causes and risk controls

Argument
Risk analysis has been performed by right people, using right and complete information and adequate analysis techniques and tools.

Claim(S#459999)
Device system hazards are mitigated

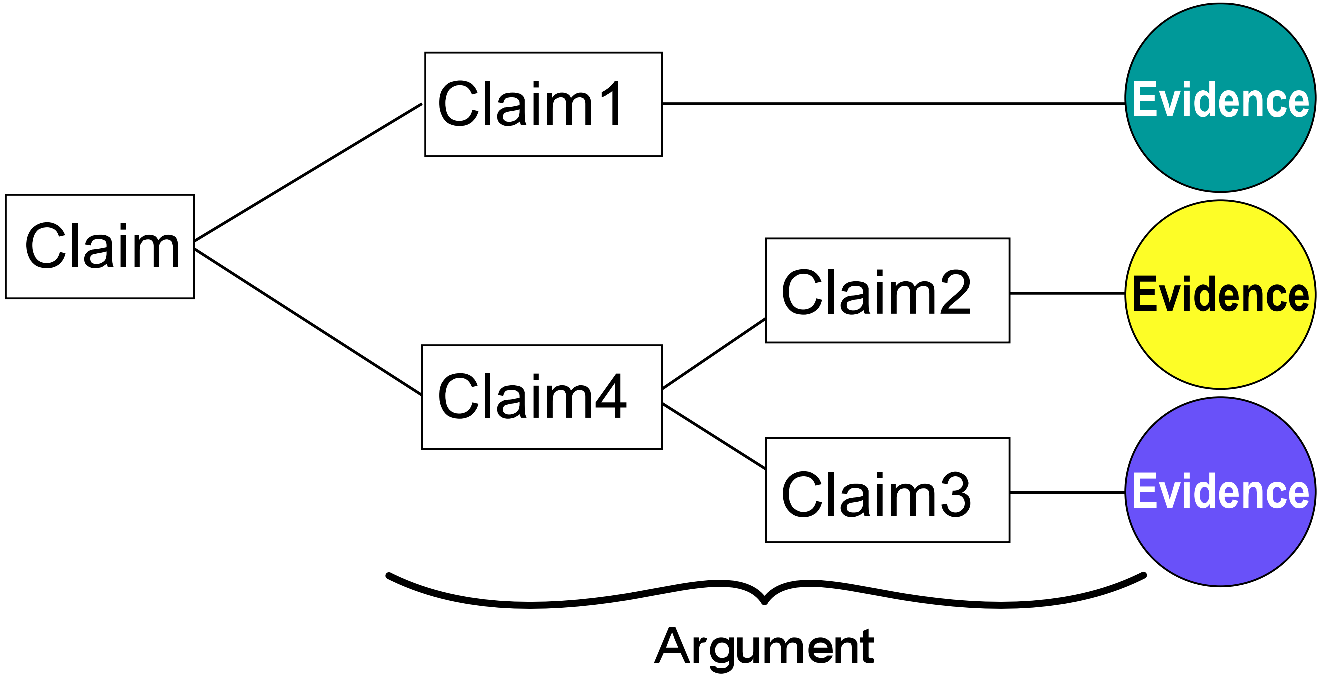
Argument
Applicable top hazards are identified in alignment with examples given in the FDA Infusion Pump TPLC guidance. Each of the top hazard has been analyzed for potential hazardous situations and adequately mitigated to reduce the risks to be acceptable level

Claim(S#459993)
Device is adequately reliable to ensure safety over its use life

Argument
System and critical componenet reliability requirements are defined and validated

Introduction of Assurance Case Method and its Application in Regulatory Science. Fubin Wu (2019).

Intellectual vs. Statistical Control Revisited



Q. What roles do intellectual and statistical control play in building an assurance case?

Assurance Case: Benefits & Limitations

- Provides an explicit structure to a design argument
  - Encourages the designer to articulate why their design works
  - Easier to navigate, inspect, and refute for reviewers
  - Provides traceability between system-level claims & low-level evidence
- Challenges and pitfalls
  - **Completeness**: How do I know whether it's missing any subclaims?
  - Effort in constructing the case & evidence: How much evidence is enough?
  - System evolution: If system changes, must also recreate the case & evidence
- Recall: **Risk-driven design**!
  - Build an assurance case for the most important functionalities or quality attributes

Exercise: Assurance Case for IntelliGuard

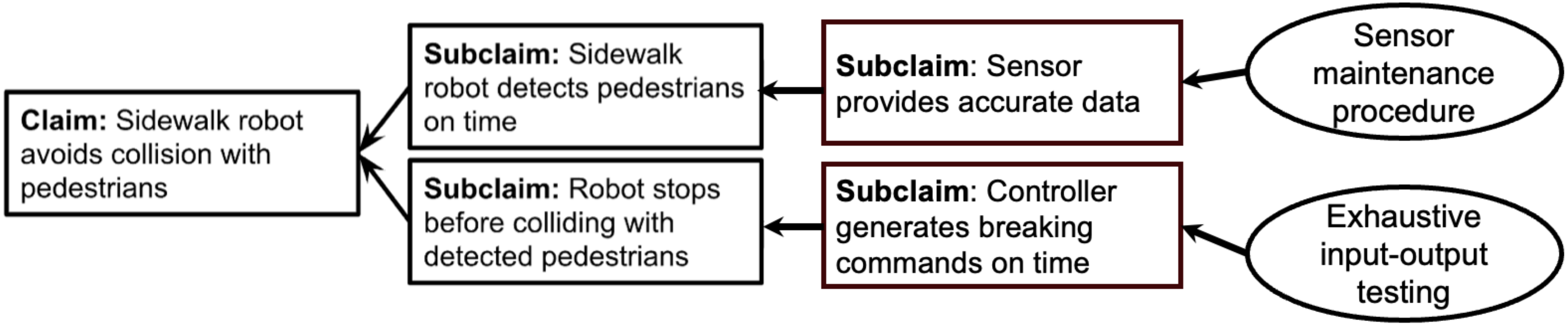
- Recall **IntelliGuard** from HW1
- Break into groups; pick one person's design from HW1
- For that design, develop an assurance case for the following top-claim: "The intrusion detection system notifies the homeowner in time when a stranger appears around the house"
- For evidence, include **hypothetical** pieces of evidence that you would include (assuming you had implemented & tested the system)
- Make sure the assurance case is legible; you will share it with your classmates later

Design Review

Design Review

- **Goals**
  - Improve the quality of a design by identifying and addressing flaws or weaknesses
  - Communicate and align the understanding of the design with other teams and stakeholders of the system
  - Indicate that the product is ready for release or the next phase of development
  - Track changes and improvements to the system design over time
- There are no ”standard” practices or methods for design reviews
- We will discuss how we can use an assurance case to drive a design review process

Criteria for Reviewing an Assurance Case

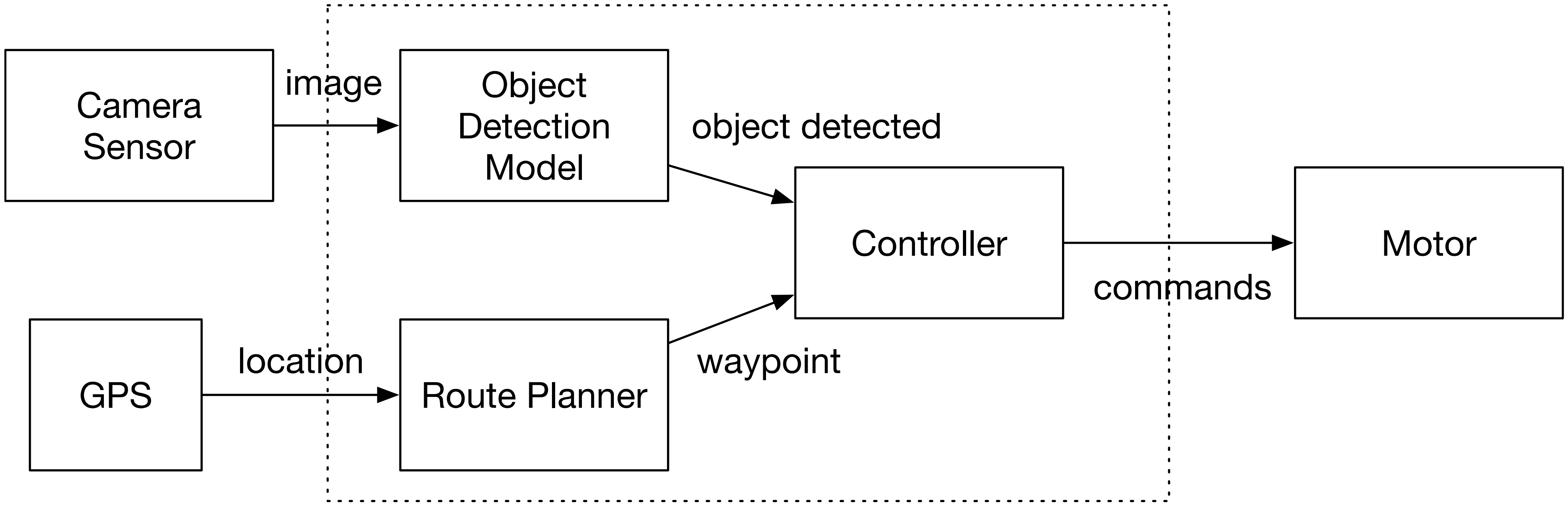

Claim: Sidewalk robot avoids collision with pedestrians
Subclaim: Sidewalk robot detects pedestrians on time
Subclaim: Robot stops before colliding with detected pedestrians
Subclaim: Sensor provides accurate data
Subclaim: Controller generates breaking commands on time
Sensor maintenance procedure
Exhaustive input-output testing

- **Soundness**: Do the subclaims imply their parent claim? Are there any missing subclaims?
- **Validity**: Is the evidence strong enough to support a leaf claim? Can the evidence be independently verified (e.g., by re-running the test cases)?

Adversarial Thinking

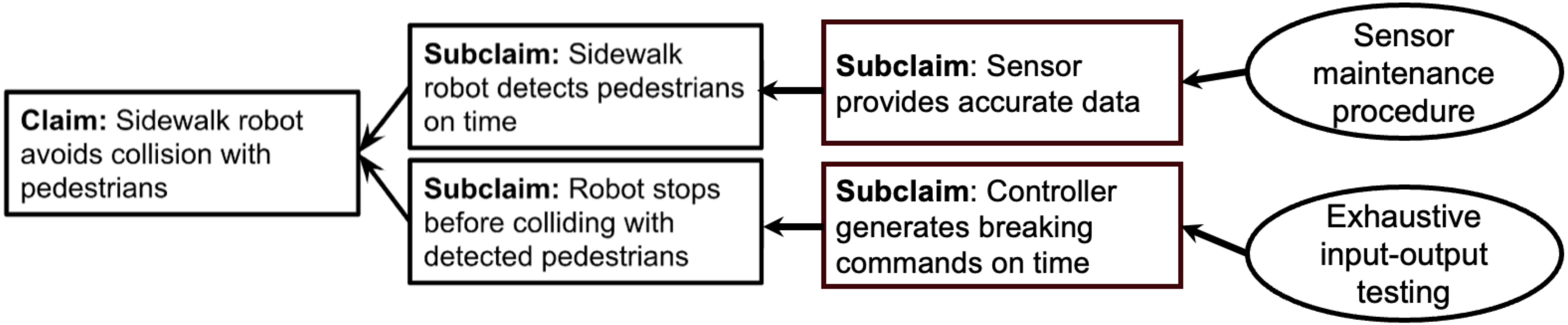
- Think like an “attacker”, not the designer of the system
- As a reviewer, your goal is to **invalidate** the argument; i.e., show how the system may fail to satisfy the claim in certain scenarios
- For each **leaf subclaim**: Think of a scenario where it fails to hold due to insufficient evidence (**validity flaw**)
- For each **non-leaf subclaim**: Think of a scenario where all its children subclaims hold but it does not (**soundness flaw**)

Component Diagram for Delivery Robot



- Consider domain entities & components that are involved in achieving the desired functionality/quality attribute
- **Q. Is there an assumption or responsibility missing from the argument? Should that be added as a new subclaim?**

Criteria for Reviewing an Assurance Case



Claim: Sidewalk robot avoids collision with pedestrians

Subclaim: Sidewalk robot detects pedestrians on time

Subclaim: Robot stops before colliding with detected pedestrians

Subclaim: Sensor provides accurate data

Subclaim: Controller generates breaking commands on time

Sensor maintenance procedure

Exhaustive input-output testing

- **Q. Is there an assumption or responsibility missing from the argument? Should that be added as a new subclaim?**

Reviewing Evidence

- **For testing & program analysis reports**: Re-run the tests or analysis under the identical conditions (if possible) and compare the output. Attempt to identify inputs that produce an incorrect output (i.e., invalidate the subclaim).
- **For design decisions**: Review the design document (e.g., component diagram) and the code to ensure that the documented decisions are implemented properly in the system.
- **For procedures**: Check that the procedure is trustworthy; often requires domain knowledge!
- **For empirical data**: Apply proper statistical methods to ensure the validity of the presented data

Sample Review Comments

- **Soundness flaw**: The subclaim “Sensor provides accurate data” is not enough to ensure “Sidewalk robot detects pedestrians on time”, since the object detection model may fail to detect a pedestrian even if it’s given an accurate image from the sensor.
- **Validity flaw**: There is not enough evidence to support the subclaim “Sensor provides accurate data”. Sensor might fail during deployment between maintenance procedures and cause the robot to ignore a pedestrian.

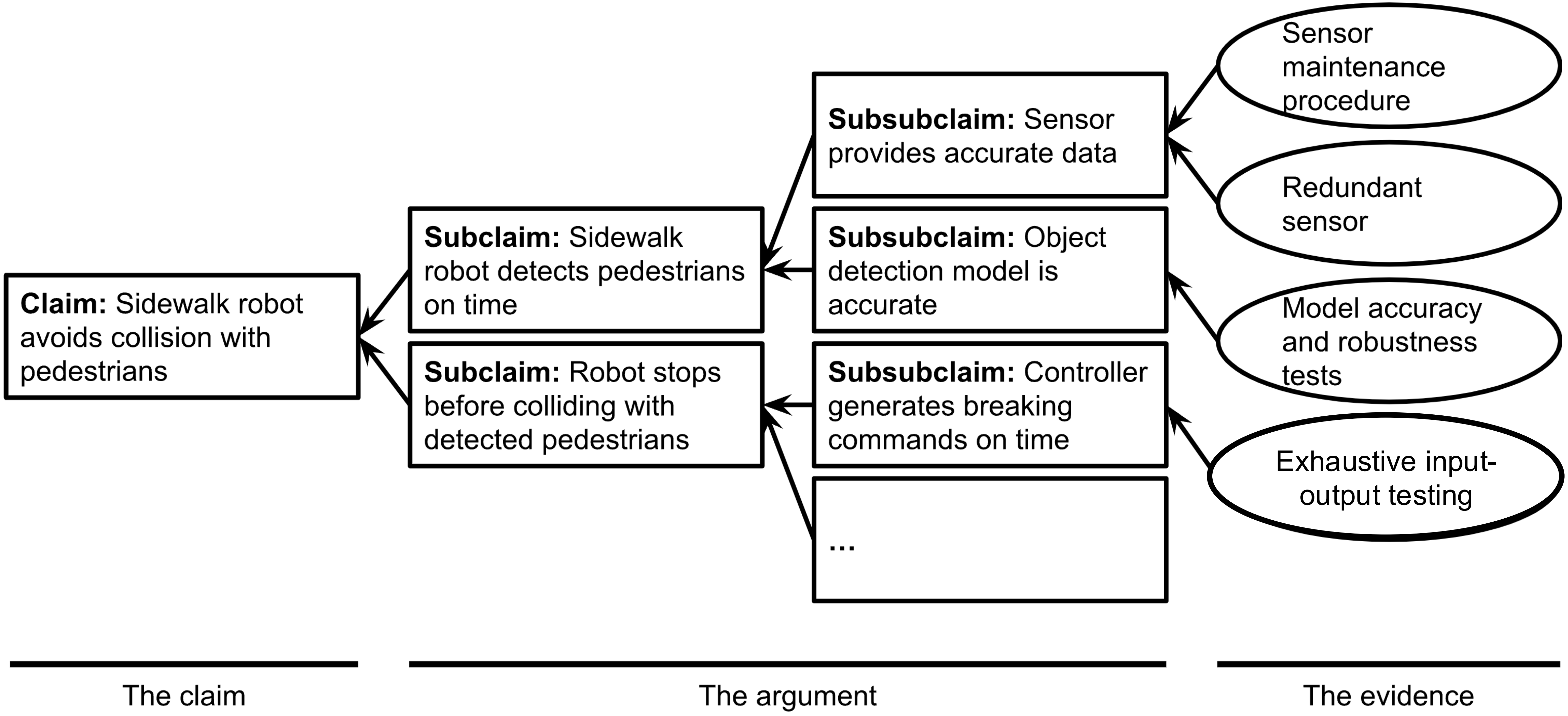
Responding to Review Feedback

- **Be open to feedback!** The goal is to improve the design, not to argue that you are right (no matter what)
- **But refute the feedback when appropriate!** It is possible that the reviewer misunderstood the design/argument. Explain why the feedback is incorrect.
- **Do nothing but put on backlog**: The identified flaws might not be significant enough to be addressed now, but can be revisited later
- **Improve the argument**
- **Improve the design, if the former is not possible.**
- Send the revised assurance case back for a further review; repeat until no more feedback

Improving the Argument

- For each **leaf subclaim**: A scenario where it fails to hold (due to insufficient evidence)
  - Add additional pieces of evidence to support the subclaim
- For each **non-leaf subclaim**: A scenario where all its children subclaims hold but it does not
  - Add a new subclaim(s) to ensure that the parent claim is implied by its children
- If no further evidence or subclaim can be added to fix the argument, then a valid argument does not exist – the design itself must be fixed!

Improved Assurance Case for Delivery Robot


Claim: Sidewalk robot avoids collision with pedestrians
Subclaim: Sidewalk robot detects pedestrians on time
Subclaim: Robot stops before colliding with detected pedestrians
Subsubclaim: Sensor provides accurate data
Subsubclaim: Object detection model is accurate
Subsubclaim: Controller generates breaking commands on time
…
Sensor maintenance procedure
Redundant sensor
Model accuracy and robustness tests
Exhaustive input-output testing
The claim
The argument
The evidence

Exercise: Assurance Case for IntelliGuard

- Take (1) an assurance case and (2) a component diagram for IntelliGuard from another group
- Review the assurance case and identify potential flaws with respect to soundness and validity
- Discuss the flaws identified by the other group: (1) refute if they are not flaws or (2) devise ways to improve the argument or design to address those flaws

Design Review: Tips

- Be constructive! The goal is to help improve the design, not to shoot it down
- Don’t nitpick; look for larger problems that could lead to significant risks for the project
- Take a risk-driven approach! Focus on claims about most important functionalities or quality attributes
- Recruit outsiders (e.g., customers, engineers from another team) for review, to reduce bias
- Keep a record of suggestions from the reviewers; track which of those suggestions have been implemented
- Do design reviews regularly, after each project milestone or iteration

Intellectual vs. Statistical Control & AI

- **Q. How might AI-assisted development affect the level of control that we (human developers) have over our system?**

MY AI JUST WROTE THIS 10000 LINES CODE IN 2 MINUTES
NOW I WILL SPEND 2 DAYS DEBUGGING IT
imgflip.com

Summary

- Exit ticket!