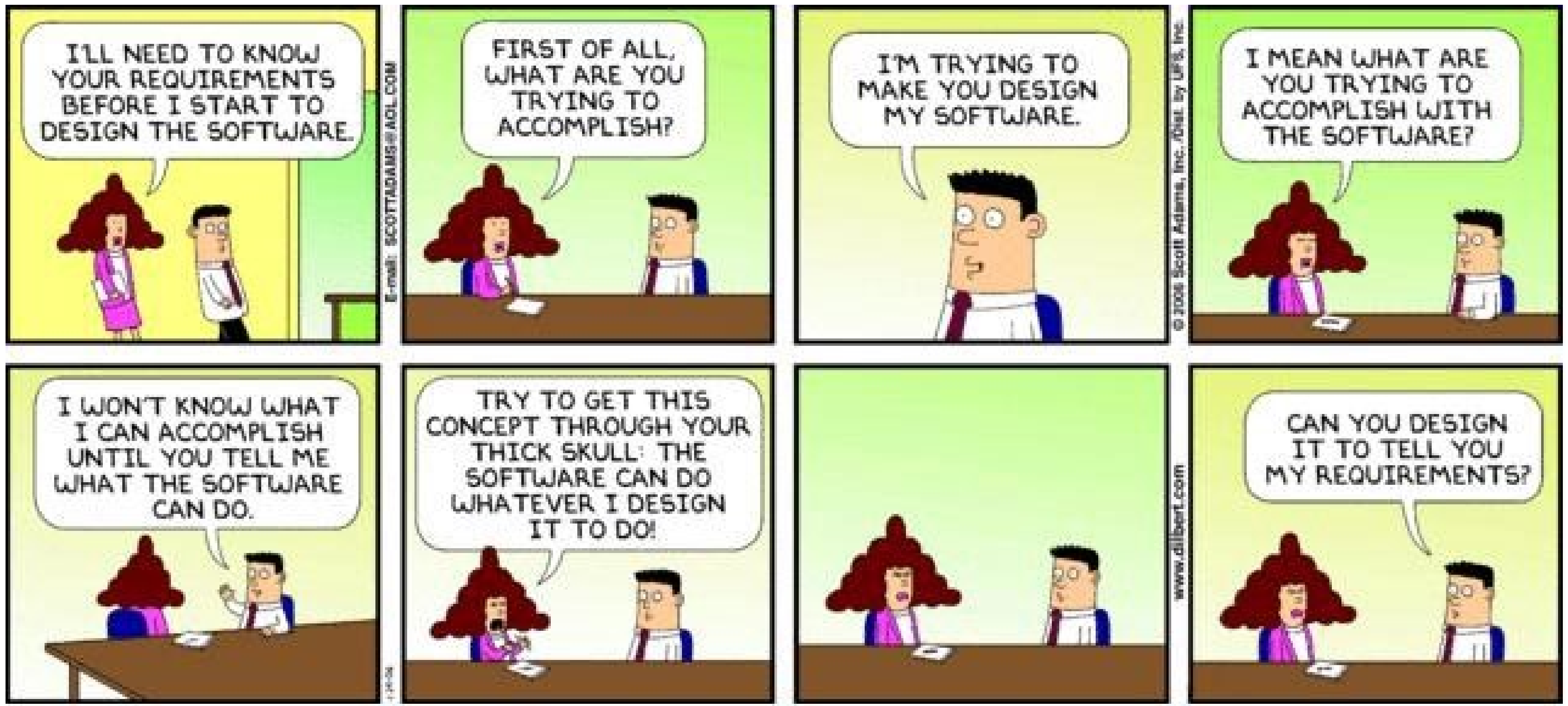
Requirements and Specifications


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
E-mail: SCOTTADAMS@AOL.COM
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

Review: Delta Debugging

- Delta Debugging is an *automatic* algorithm that
  - Takes a set of “changes” as input (call it X)
    - Changes to source code (git commit diffs)
    - Changes to input (line or character diffs to an input file)
    - Changes to schedule (timings or # instructions executed per thread)
  - Takes an “Interesting(x)” function
    - Interesting is “true” if applying diffs in x leads to program failure
    - … or if feeding input x leads to failure
    - … or if feeding schedule x leads to race conditions
  - Produces a *one-minimal set* as output wrt Interesting
- Delta Debugging helps you minimize test inputs!

Delta Debugging Algorithm

DD(P, $\{c_1, \dots, c_n\}$) =

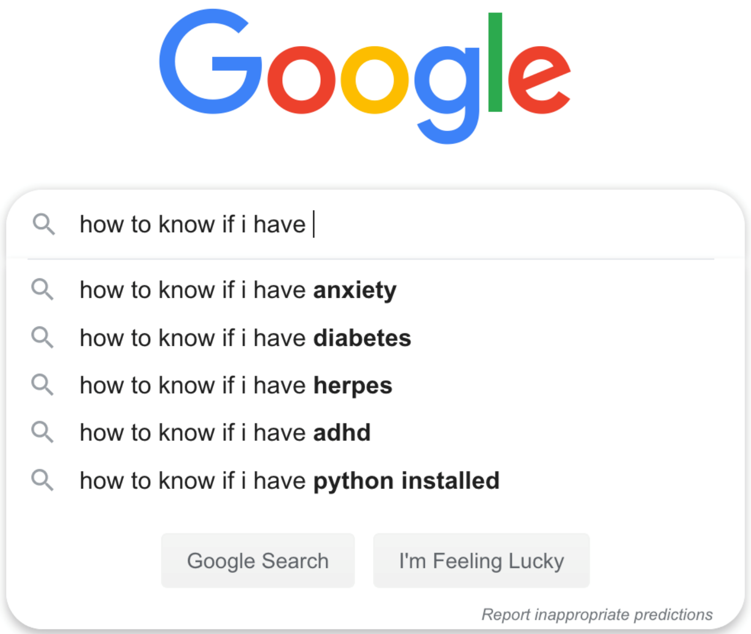
- if $n = 1$ then return $\{c_1\}$
- let P1 = $\{c_1, \dots c_{n/2}\}$
- let P2 = $\{c_{n/2+1}, \dots, c_n\}$
- if Interesting($P \cup P1$) = Yes then return DD(P,P1)
- if Interesting($P \cup P2$) = Yes then return DD(P,P2)
- else return DD($P \cup P2$, P1) $\cup$ DD($P \cup P1$, P2)

Assumptions

- All three key assumptions are questionable
- Interesting is **Monotonic**
  - Interesting(X) → Interesting(X $\cup$ {c})
- Interesting is **Unambiguous**
  - Interesting(X) & Interesting(Y) → Interesting(X $\cap$ Y)
- Interesting is **Consistent**
  - Interesting(X) = Yes or Interesting(X) = No
  - (Some formulations: Interesting(X) = Unknown)

Moving on to Requirements! The Story So Far ...

- **Quality assurance** is critical to software engineering
- OK, so we want to build a **quality** product
- What are we *supposed* to be building, again?

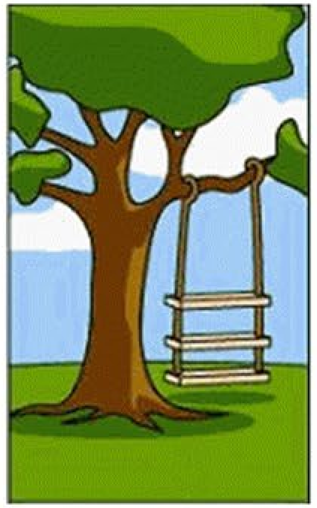


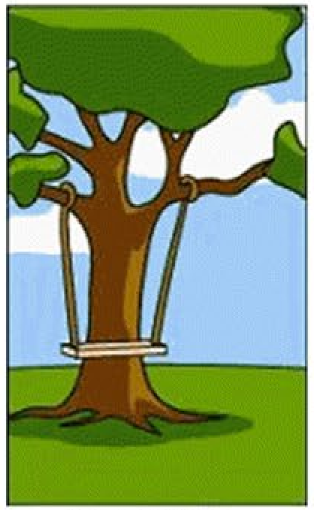
how to know if i have

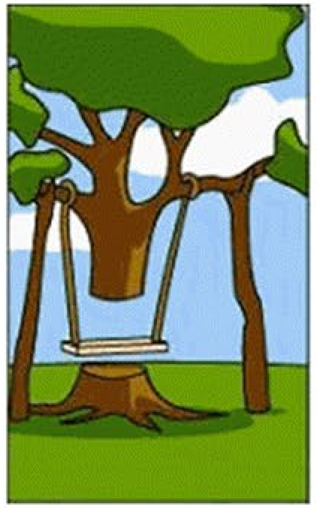
how to know if i have **anxiety**
how to know if i have **diabetes**
how to know if i have **herpes**
how to know if i have **adhd**
how to know if i have **python installed**

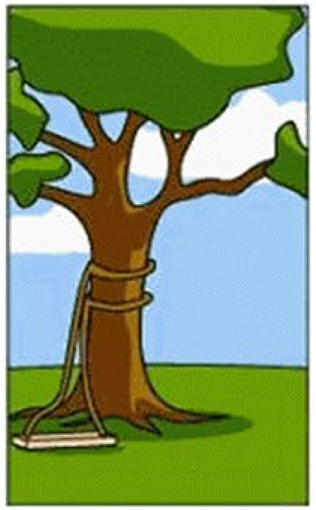
One-Slide Summary

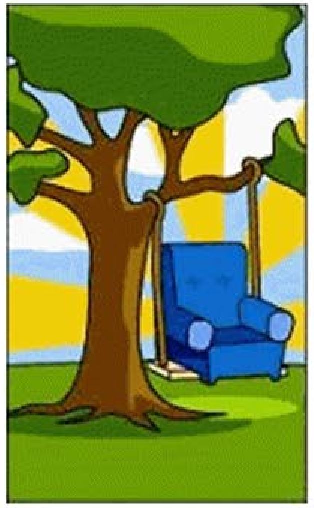
- **Requirements** articulate the relationship and interface between a desired system and its environment.
- **Requirement Engineering** is the process of identifying, eliciting, analyzing, specifying, validating and managing the needs and expectations of stakeholders for a software system.
- We distinguish between **functional** and **quality** (or non-functional) requirements. Both should be stated in measurable ways.
- Requirements can describe variables, inputs, outputs, and assumptions between them.
- We distinguish between informal statements and **verifiable** requirements.

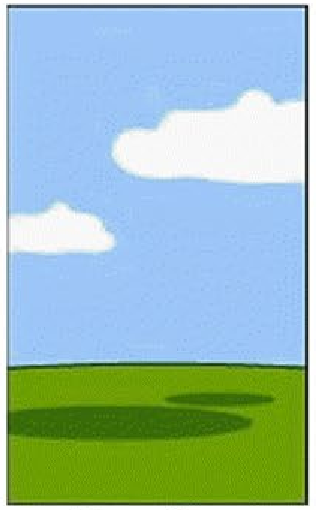

How the customer explained it

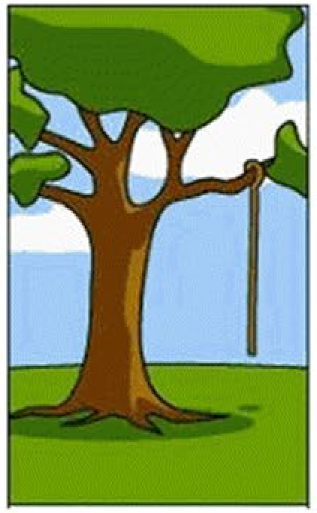

How the project leader understood it

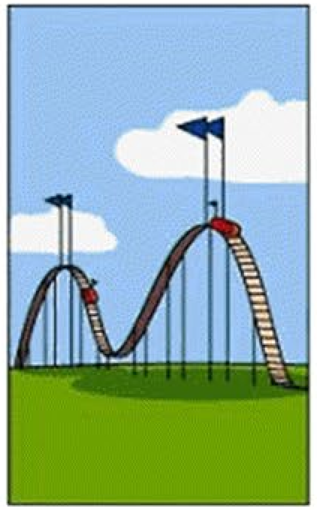

How the engineer designed it

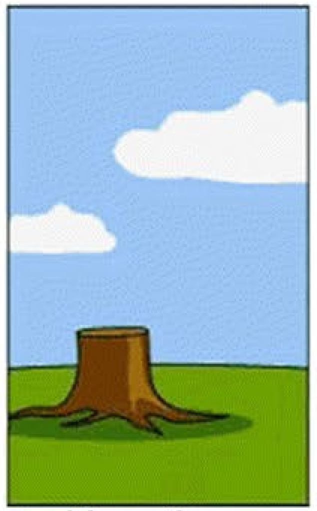

How the programmer wrote it

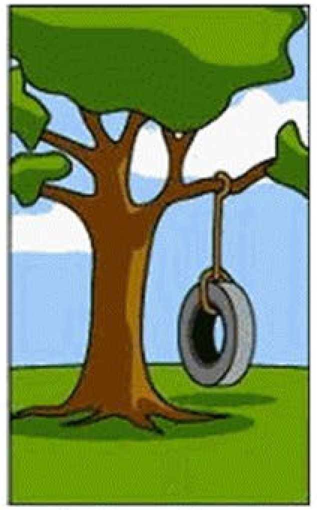

How the sales executive described it


How the project was documented


What operations installed


How the customer was billed


How the help desk supported it

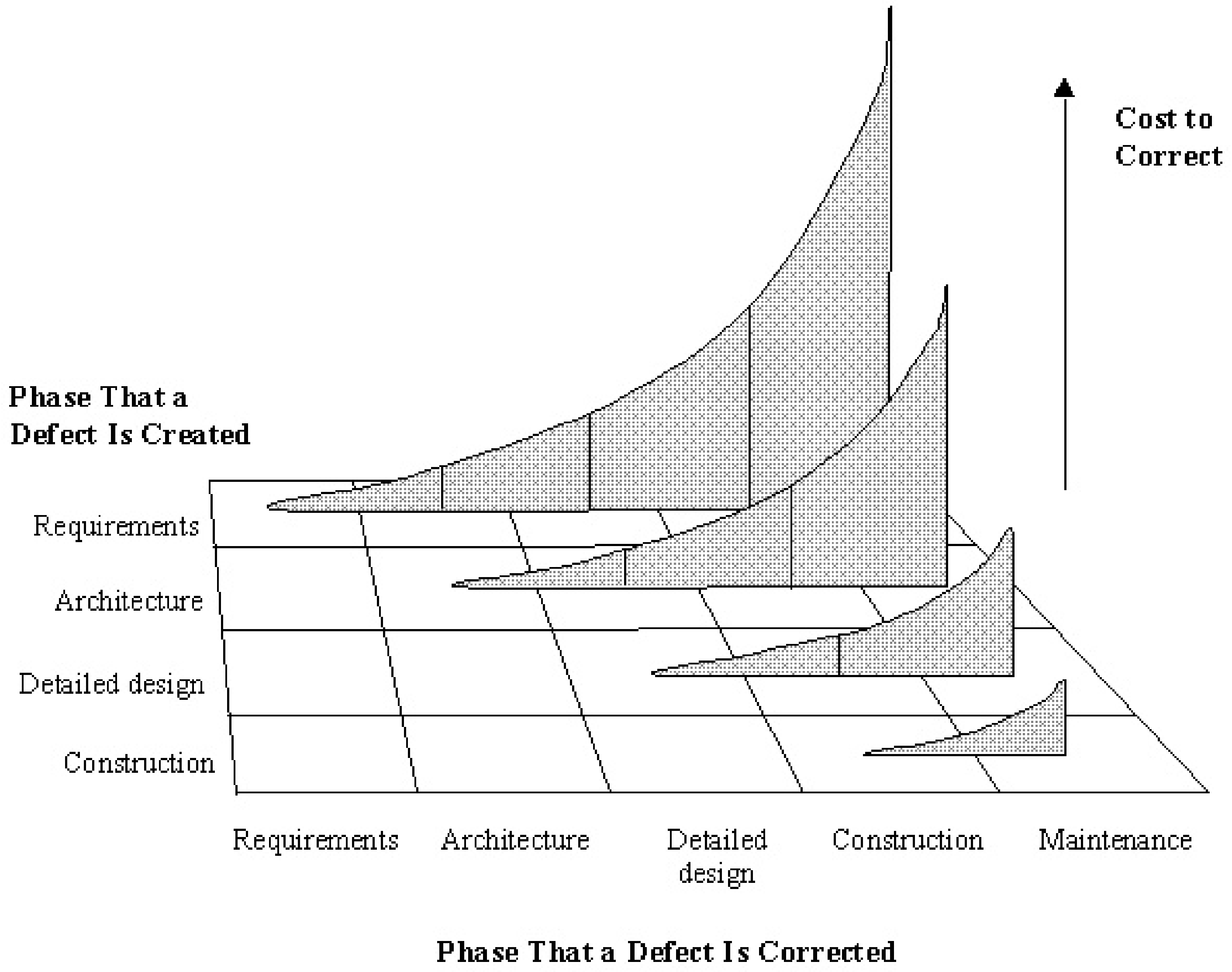

What the customer really needed

Requirements

- **Requirements** say what the system will do, *not* how it will do it
- “The hardest single part of building a software system is deciding precisely **what to build**. No other part of the conceptual work is as difficult as establishing the detailed technical requirements … No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later.”

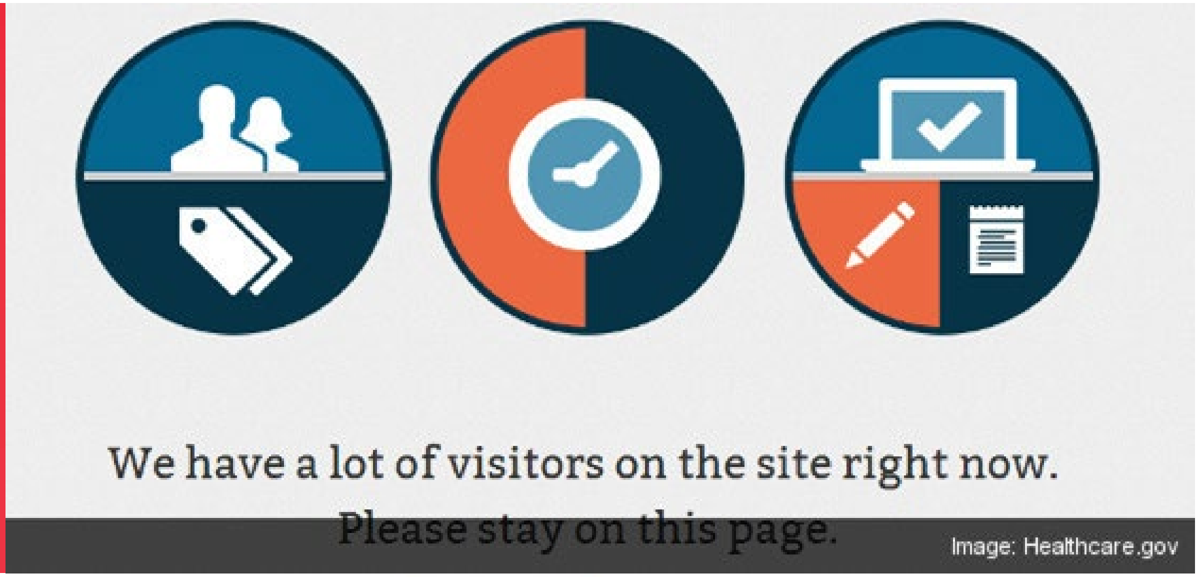
  — Fred Brooks

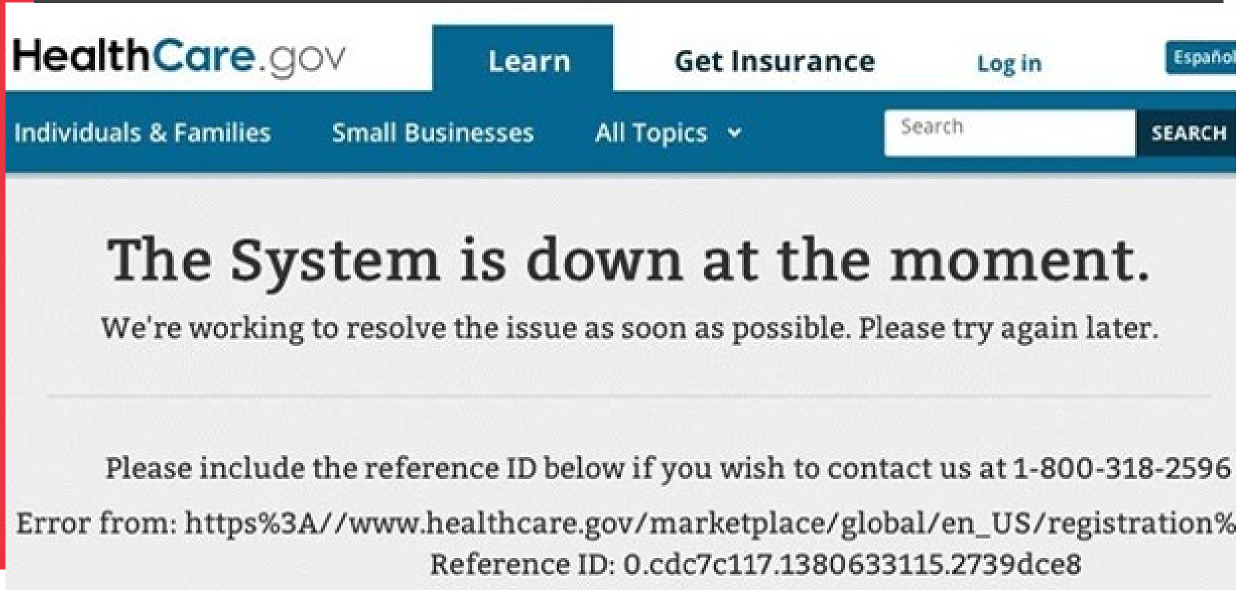
“Difficult to Rectify Later”



Healthcare.gov

MARCH 10, 2014
Ukraine's Future / Minimum-Wage Wars / Wes Anderson
TIME
CODE RED_
Inside the nightmare launch of HealthCare.gov and the team that figured out how to fix it
BY STEVEN BRILL
time.com


We have a lot of visitors on the site right now.
Please stay on this page.



The System is down at the moment.

We're working to resolve the issue as soon as possible. Please try again later.

Please include the reference ID below if you wish to contact us at 1-800-318-2596

Error from: https%3A//www.healthcare.gov/marketplace/global/en_US/registration%
Reference ID: 0.cdc7c117.1380633115.2739dce8

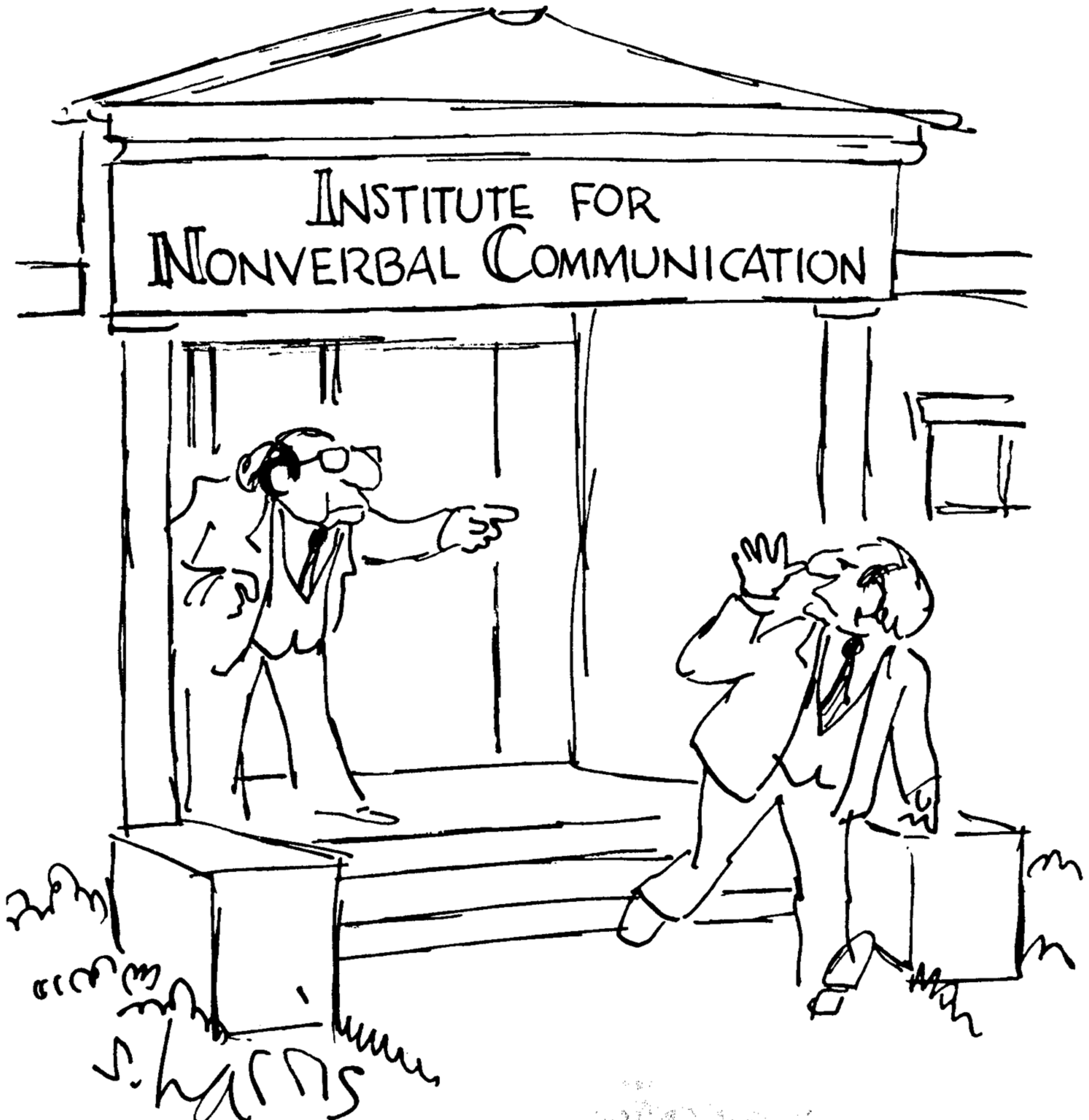
What is Past is Prologue

- A 1994 survey of 8000 projects at 350 companies found: 31% of projects canceled before completed; 9% of projects delivered on time, within budget in large companies, 16% in small companies. (Similar results reported since.)
- Largest Causes:
  - Incomplete requirements (13.1%)
  - Lack of user involvement (12.4%)
  - Lack of resources (10.6%)
  - Unrealistic expectations (9.9%)
  - Lack of executive support (9.3%)

$\longleftarrow$ No “programmers were too inept” or “we forgot how AVL trees work”

Communication Problem

- Goal: figure out what *should* be built
  - Express those ideas so that the *correct* thing is built



Requirements Brainstorming Example

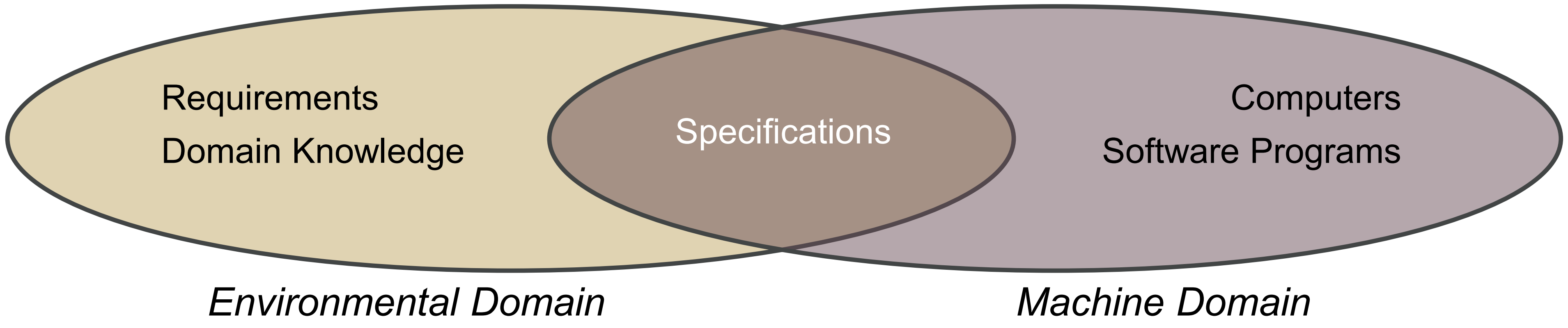
- You are paying someone to write software for “selling videos on the web”
- Your thoughts on …
  - How fast should we deliver content, at what quality, for what price?
  - “Nice to have” functionality
  - Required functionality
  - Expected qualities
  - Involved subproblems

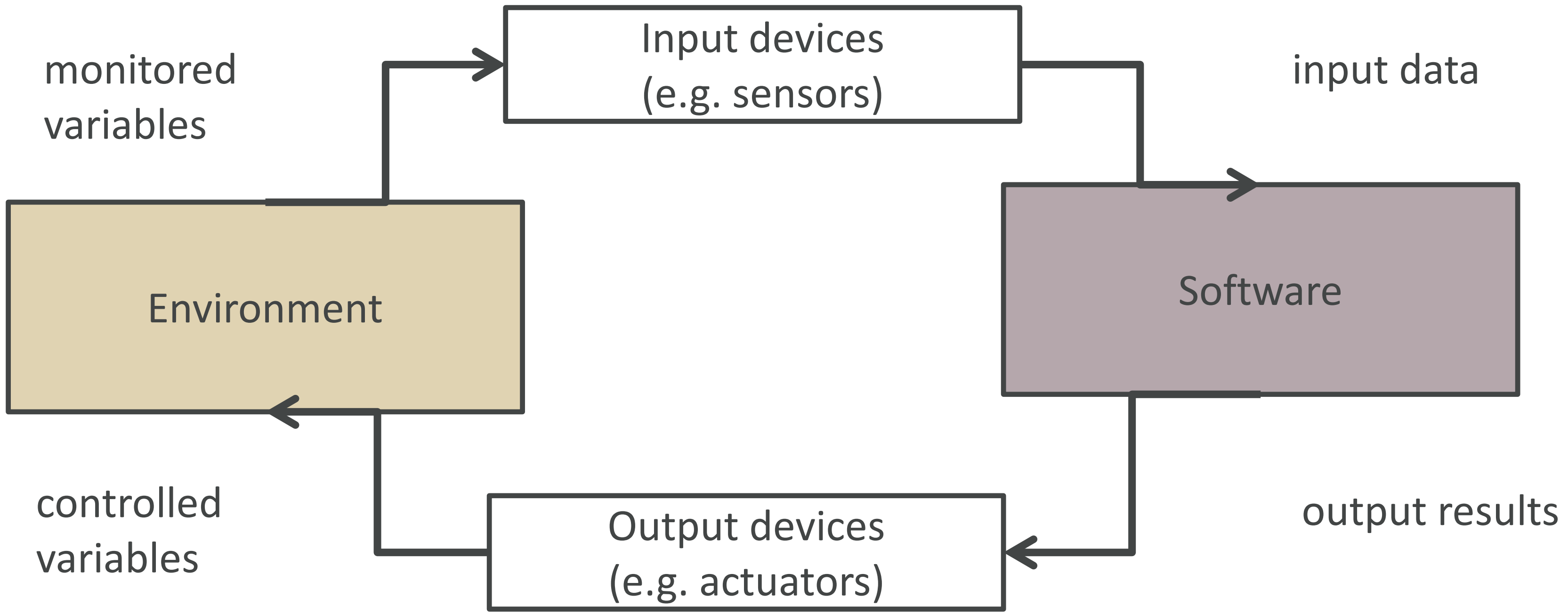
Use clothing to keep warm during winter

Winter has arrived, but you have decided to cut down on your bills this year by lowering the thermostat. Although your feet will soon feel chilly

PROBABLY BAD NEWS.COM

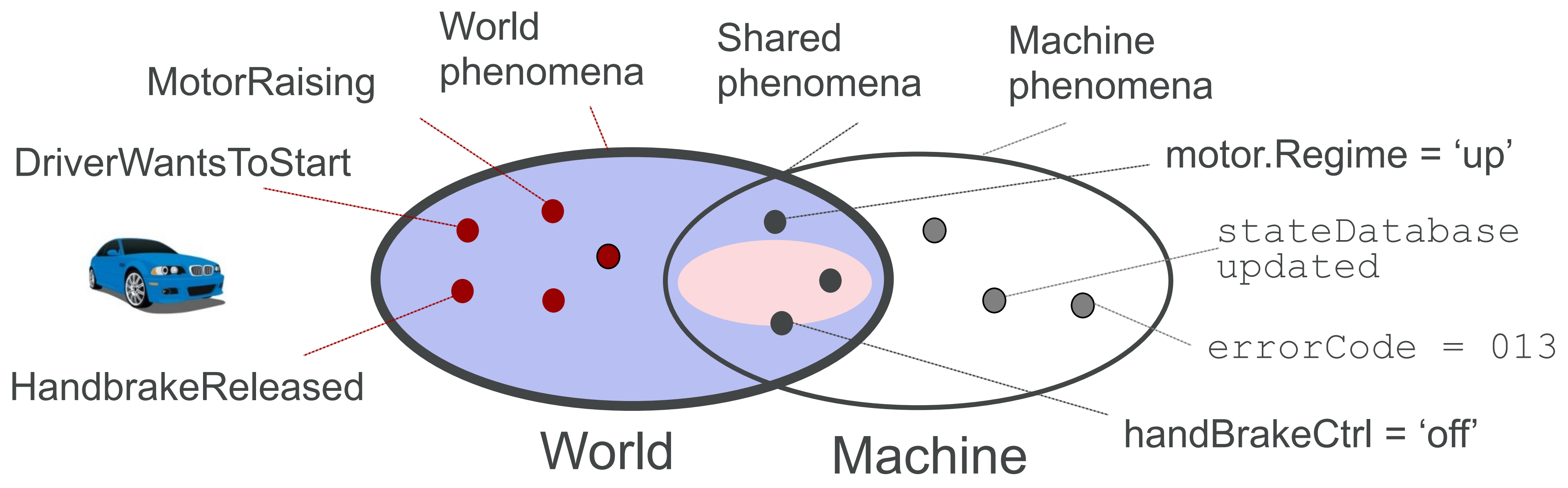
Environment vs. Machine





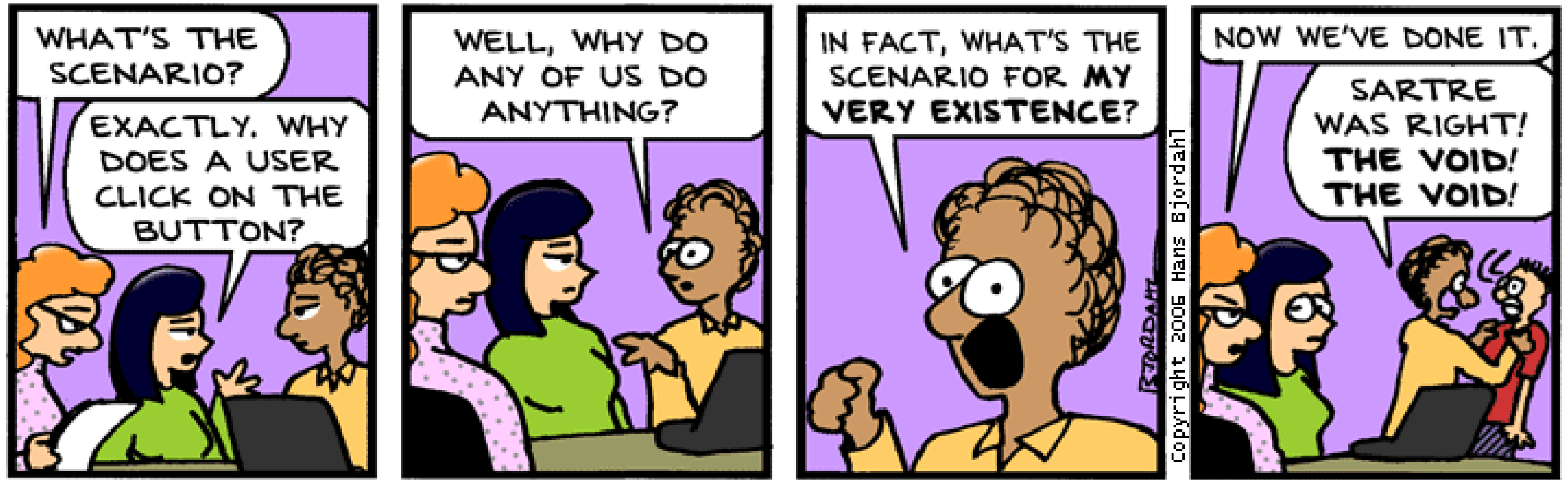
Pamela Zave & Michael Jackson, "Four Dark Corners of Requirements Engineering,"

Environment vs. Machine
Example: Automobile



Delving into Requirements: System, Software, Assumptions

- System requirements: relationships between monitored and controlled variables
- Software requirements: relationship between inputs and outputs
- Domain properties and assumptions state relationships between those


WHAT'S THE SCENARIO?
EXACTLY. WHY DOES A USER CLICK ON THE BUTTON?
WELL, WHY DO ANY OF US DO ANYTHING?
IN FACT, WHAT'S THE SCENARIO FOR MY VERY EXISTENCE?
Copyright 2006 Hans Bjordahl
NOW WE'VE DONE IT.
SARTRE WAS RIGHT! THE VOID! THE VOID!

Bug Bash by Hans Bjordahl http://www.bugbash.net/

Lufthansa Flight 2904: Sep 14, 1993

Lufthansa Flight 2904

- There are two “on ground” conditions:

1. Each shock absorber bears a load of 6300 kgs
2. Both wheels turn at 72 knots (83 mph) or faster

- Ground spoilers activate for conditions 1 or 2
- Reverse thrust only activates for condition 1 on both main landing gears
- Wheel brake activation depends upon the rotation gain and condition 2

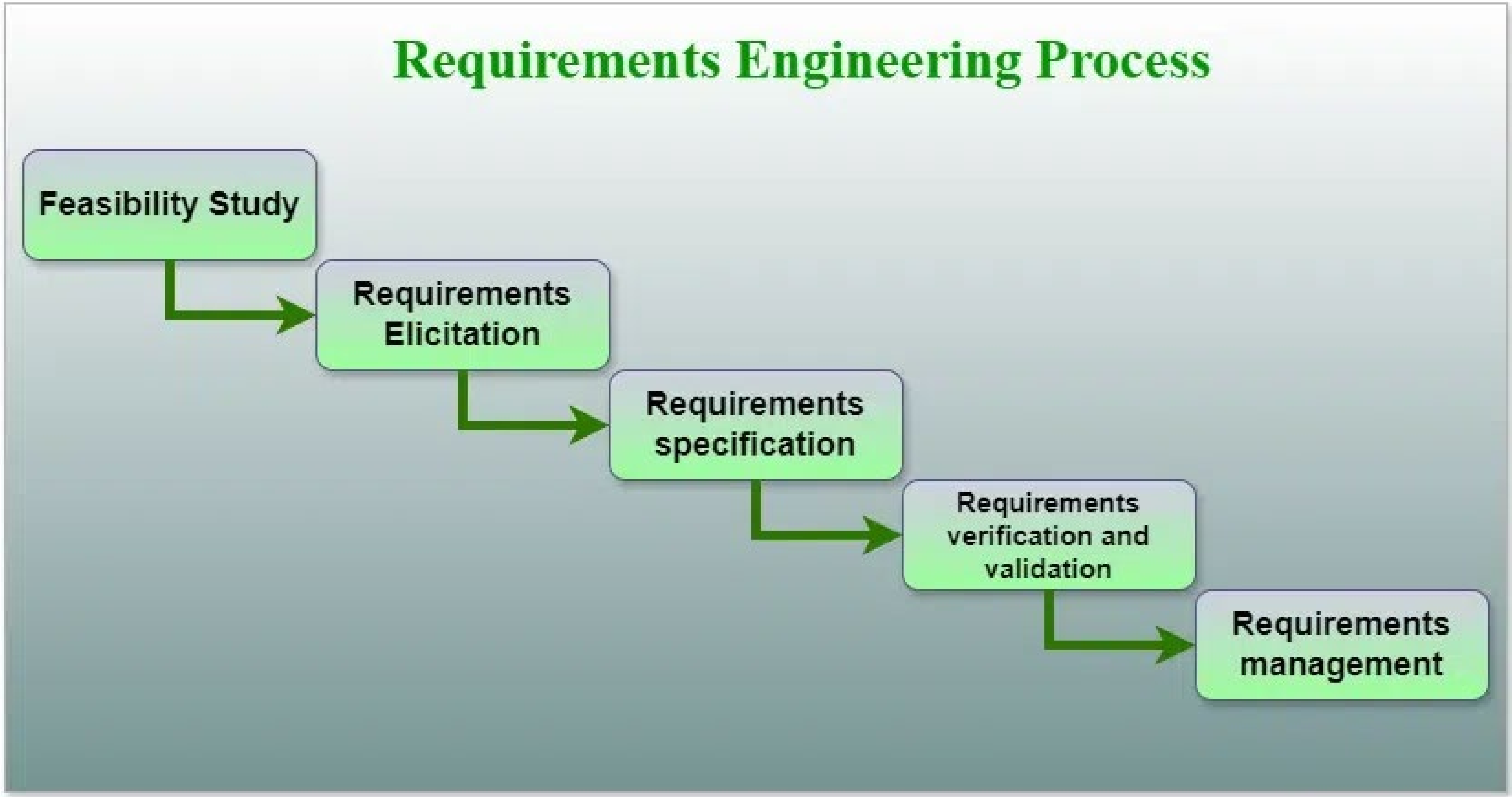
Why Didn't it Stop?
(2 died, 56 injured)

- There is no way for the pilots to override the software decision manually
- The thrust reversers are only activated if the first condition is true
- The braking system was not activated
  - Condition 1 was not fulfilled because the plane landed inclined (to counteract crosswind). Thus the required pressure on both landing gears was not reached.
  - Condition 2 was not fulfilled due to a hydroplaning effect on the wet runway.

Requirements Engineering

- **Knowledge acquisition**: how to capture relevant detail about a system
  - Is the knowledge complete and consistent?
- **Knowledge representation**: once captured, how do we express it most effectively
  - Express it for whom?
  - Is it received consistently by different people?
- You may sometimes see a distinction between the requirements *definition* and the requirements *specification*

Requirements Engineering: Typical Process


Requirements Engineering Process
Feasibility Study
Requirements Elicitation
Requirements specification
Requirements verification and validation
Requirements management

Feasibility Study

- Technical Feasibility
  - Whether there are correct required resources and technologies that will be used for project development.
  - The technical skills and capabilities of the technical team
- Operational Feasibility
  - How easy the product will be to operate and maintain after deployment.
- **Economic Feasibility**
  - Cost and benefit of the project
    - Includes all required costs for final development hardware and software resources required, design and development costs operational costs, and so on
    - Whether the project will be beneficial in terms of finance for the organization or not

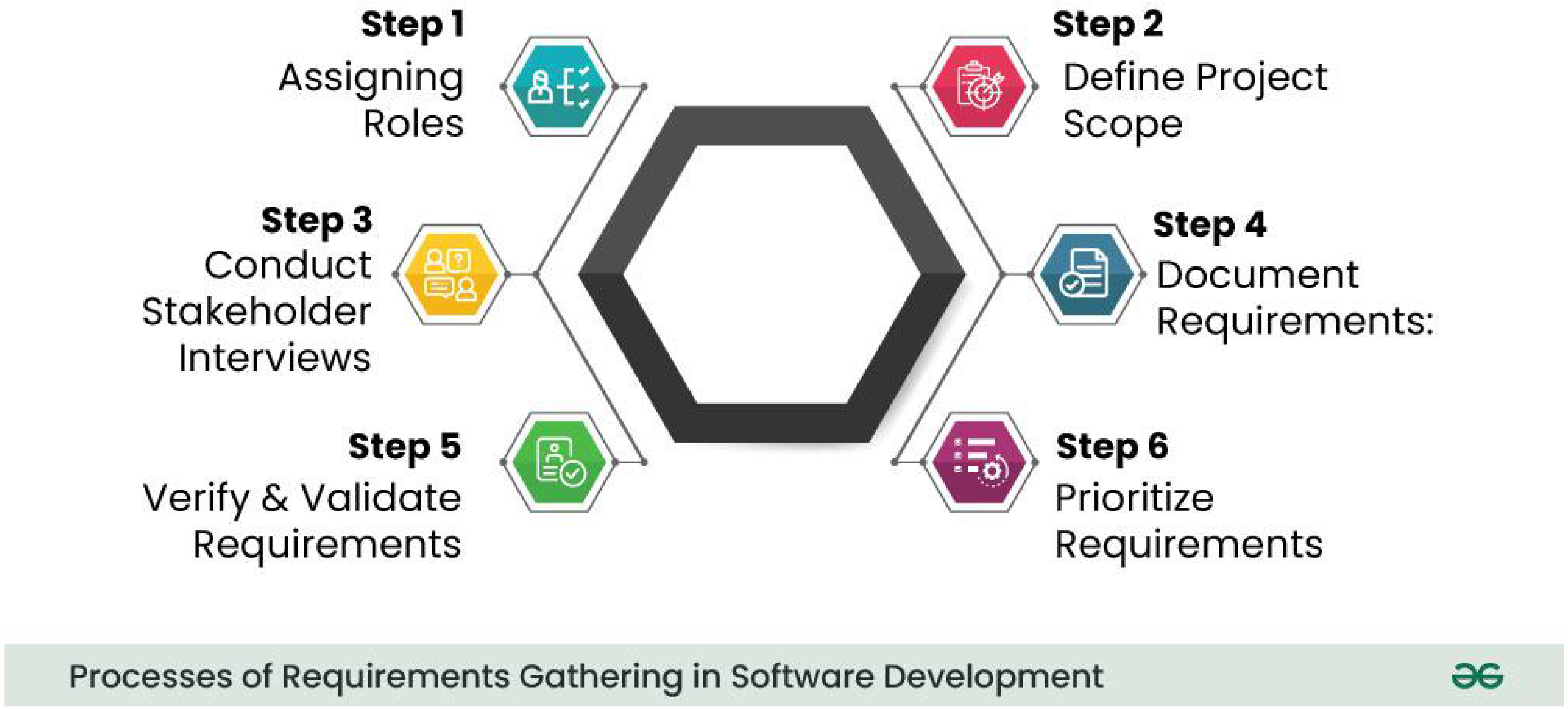
Requirements Elicitation ?= Requirements Gathering (I say, YES)

- **Gain knowledge** about the project domain and requirements
  - Stakeholder identification and analysis
    - Stakeholder: who will be affected by the system, directly or indirectly (e.g., users, clients, project team, managers, etc.)
    - Understand the needs, expectations, and influence of each stakeholder
    - Process: interviews, surveys, workshops, prioritize requirements and manage conflicting interests

Requirements Elicitation ?= Requirements Gathering (I say, YES)

- **Gain knowledge** about the project domain and requirements
  - Problem definition
    - Engage the stakeholders in discussion to uncover or articulate the core problems or opportunities
  - Requirement extraction
    - Gather detailed requirements from stakeholders
    - Process: interviews, surveys, observations, or brainstorming sessions
  - Requirement documentation
    - Document gathered requirements in a **structured format.**
    - **Process:** Create requirements documents, use cases, user stories, or prototypes to capture and communicate requirements effectively.
  - **Validation and verification:**
    - Ensure that gathered requirements are accurate, complete, and consistent.
    - **Process:** Conduct reviews, walkthroughs, or use validation tools to verify that the requirements meet the defined criteria, prototyping to get feedback

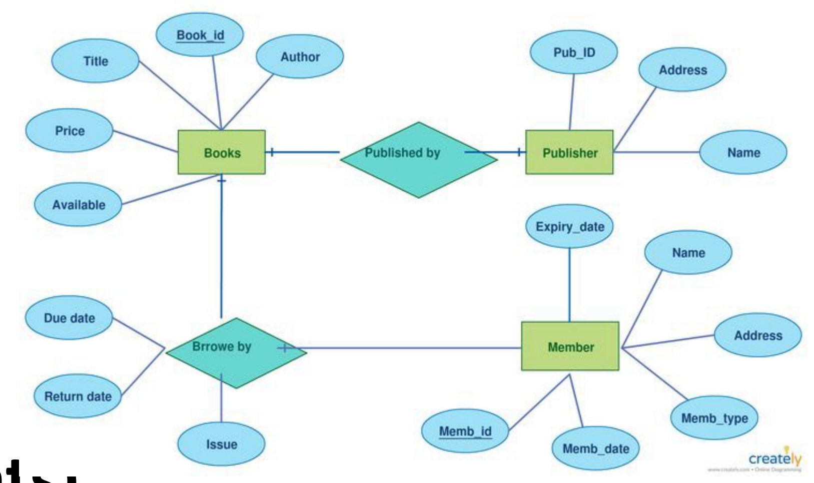
Requirement Elicitations ?= Requirements Gathering (I say, YES)

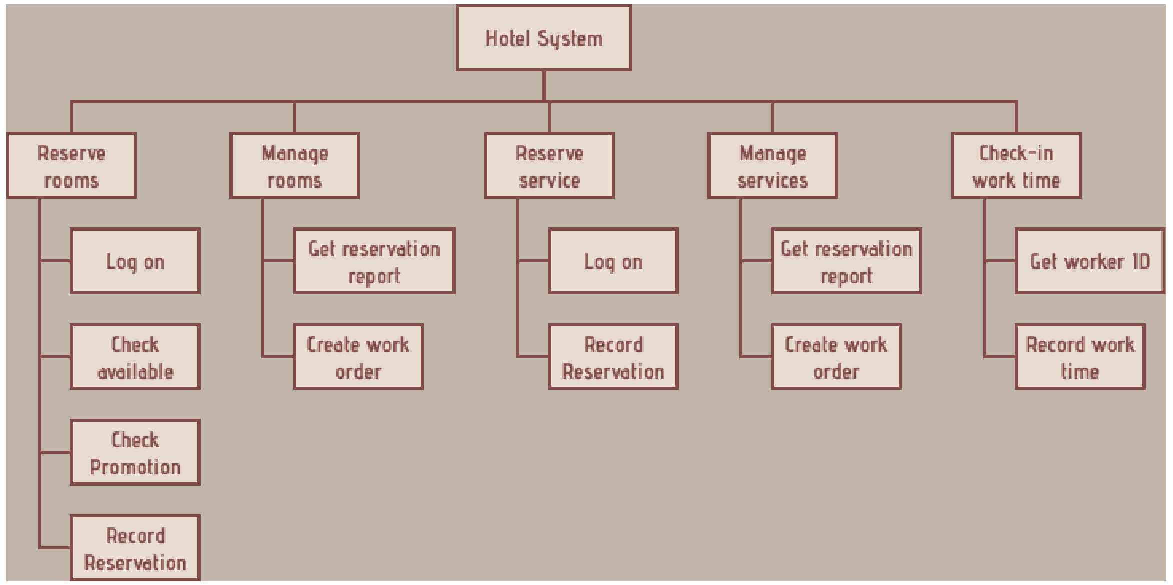

Step 1
Assigning Roles
Step 2
Define Project Scope
Step 3
Conduct Stakeholder Interviews
Step 4
Document Requirements:
Step 5
Verify & Validate Requirements
Step 6
Prioritize Requirements
Processes of Requirements Gathering in Software Development

Requirements Specification

- Produce formal software requirement models
  - **Functional requirements**
  - **Non-functional requirements (quality requirements)**
- Models
  - Entity relationship diagrams (ERDs)
  - Dataflow diagrams (DFDs)
  - Function decomposition diagrams (FDDs)
  - …

E-R Diagram for Library Management System





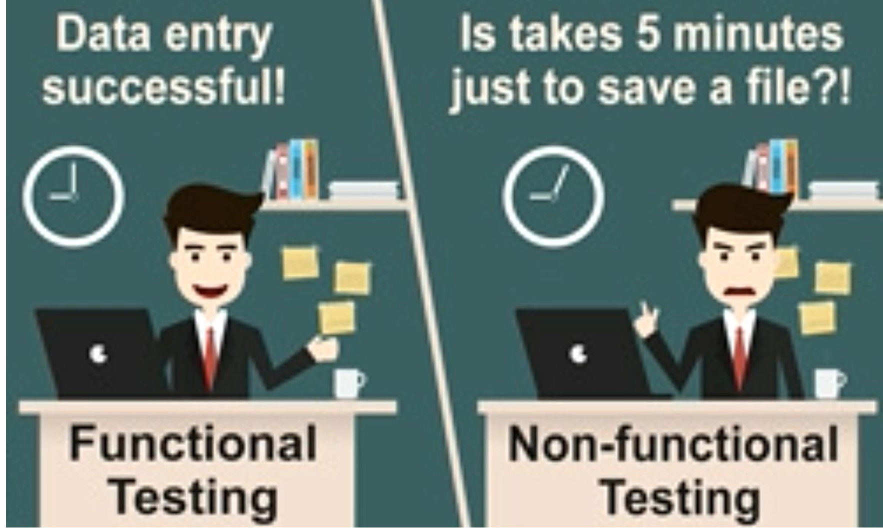
Functional Requirements

- **Functional requirements** describe what the machine should do (“get the right answer”)
  - Input, Output
  - Interface
  - Response to events
- Criteria
  - Completeness: All requirements are documented
  - Consistency: No conflicts between requirements
  - Precision: No ambiguity in requirements

Quality (nonfunctional) Requirements

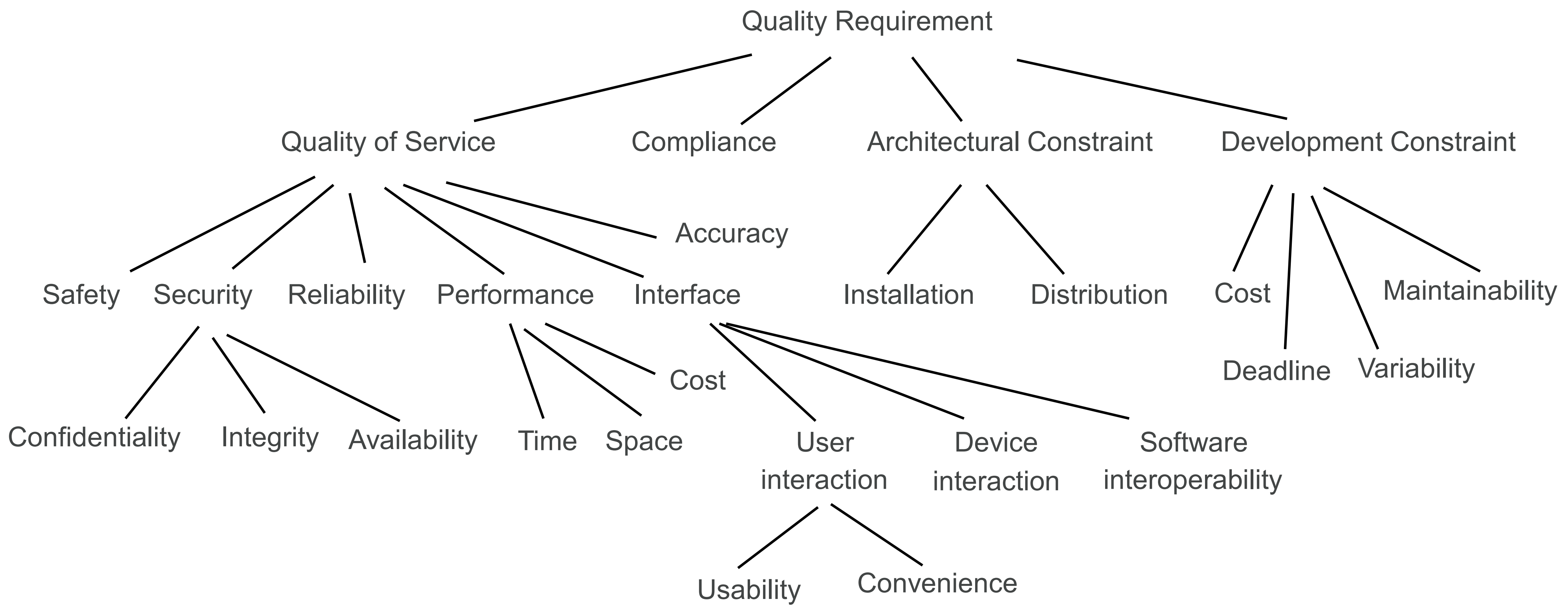
- **Quality requirements** specify *not* the functionality of the system, but *the manner in which* it delivers that functionality
- Can be more critical than functional requirements
  - Can work around missing functionality
  - Low-quality system may be unusable
- Examples?

Framing the Question



- Who is going to ask for a slow, inefficient, unmaintainable system?
- A better way to think about quality requirements is as **design criteria to help choose between alternative implementations**
- The question becomes: *to what extent* must a product satisfy these requirements to be acceptable?

Quality Requirement Examples



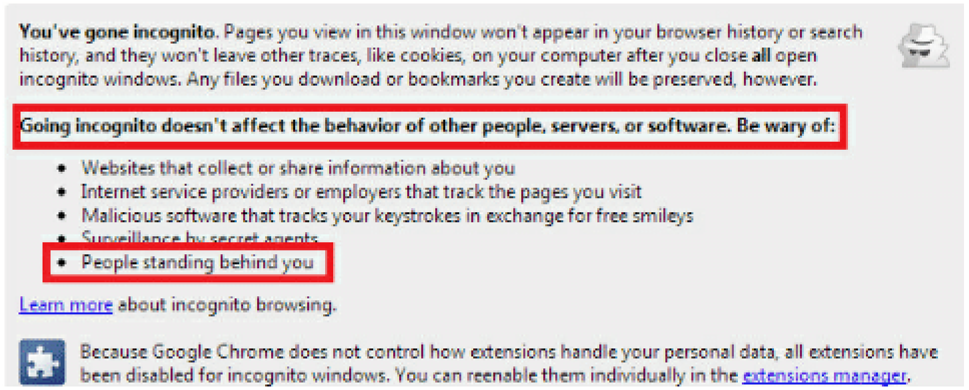
Expressing Quality Requirements

- Requirements serve as contracts: they should be testable/falsifiable
- An **informal goal** is a general intention (e.g., "ease of use" or "high security")
  - May still be helpful to developers as they convey the intentions of the system users
- A **verifiable** non-functional requirement is a statement using some measure that can be objectively tested

Informal vs. Verifiable Example

- **Informal goal**: “the system should be easy to use by experienced controllers, and should be organized such that user errors are minimized.”
- **Verifiable non-functional requirement**: “Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day, on average.”

Quality Requirement Examples


You've gone incognito. Pages you view in this window won't appear in your browser history or search history, and they won't leave other traces, like cookies, on your computer after you close **all** open incognito windows. Any files you download or bookmarks you create will be preserved, however.

Going incognito doesn't affect the behavior of other people, servers, or software. Be wary of:

- Websites that collect or share information about you
- Internet service providers or employers that track the pages you visit
- Malicious software that tracks your keystrokes in exchange for free smileys
- Surveillance by secret agents
- People standing behind you

Learn more about incognito browsing.

Because Google Chrome does not control how extensions handle your personal data, all extensions have been disabled for incognito windows. You can reenable them individually in the extensions manager.

- Confidentiality requirement: A non-staff patron may never know which books have been borrowed by others
- Privacy requirement: The calendar constraints of a participant may never be disclosed to other invited participants without consent
- Integrity requirement: The return of book copies shall be encoded correctly and by library staff only
- Availability requirements: A blacklist of bad patrons shall be made available at any time to library staff. Information about train positions shall be available at any time to the vital station computer.

Quality Requirement Examples

- Reliability req: The train acceleration control software shall have a mean time between failures on the order of 100 hours
- Accuracy req: A copy of a book shall be stated as available by the loan software if and only if it is actually available on the library shelves. The information about train positions used by the train controller shall accurately reflect the actual position of trains up to 4 meters at most. The constraints used by the meeting scheduler should accurately reflect the real constraints of invited participants.
- Performance req: Responses to bibliographical queries shall take less than 2 seconds. Acceleration commands shall be issued to every train every 3 seconds. The meeting scheduler shall be able to accommodate up to 9 requests in parallel. The new e-subscription facility should ensure a 30% cost saving.

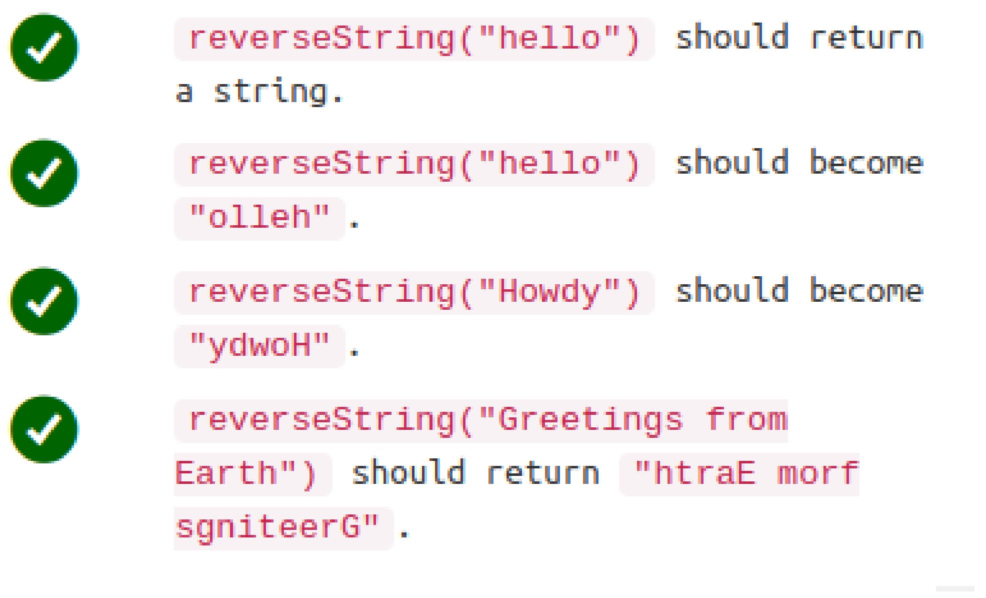
Requirements Verification and Validation

- Verification
  - The set of tasks that ensures the software correctly implements a specific function
  - Checking that the requirements are complete, consistent, and accurate.
- Validation
  - A different set of tasks that ensures that the software that has been built is traceable to customer requirements
  - Checking that the requirements meet the needs and expectations of the stakeholders.
- Verification and Validation is an iterative process that occurs throughout the software development life cycle.
  - Testing!

Requirement Management

- Analyzing, documenting, tracking, prioritizing, and agreeing on the requirement and controlling the communication with relevant stakeholders.
- “the changing nature of requirements. ”
- Activities:
  - **Tracking and controlling changes**
  - **Version control**
  - **Traceability:** linking the requirements to other elements of the development process, such as design, testing, and validation.
  - **Communication**
  - **Monitoring and reporting:** monitoring the progress of the development process and reporting on the status of the requirements.

What Could Go Wrong?

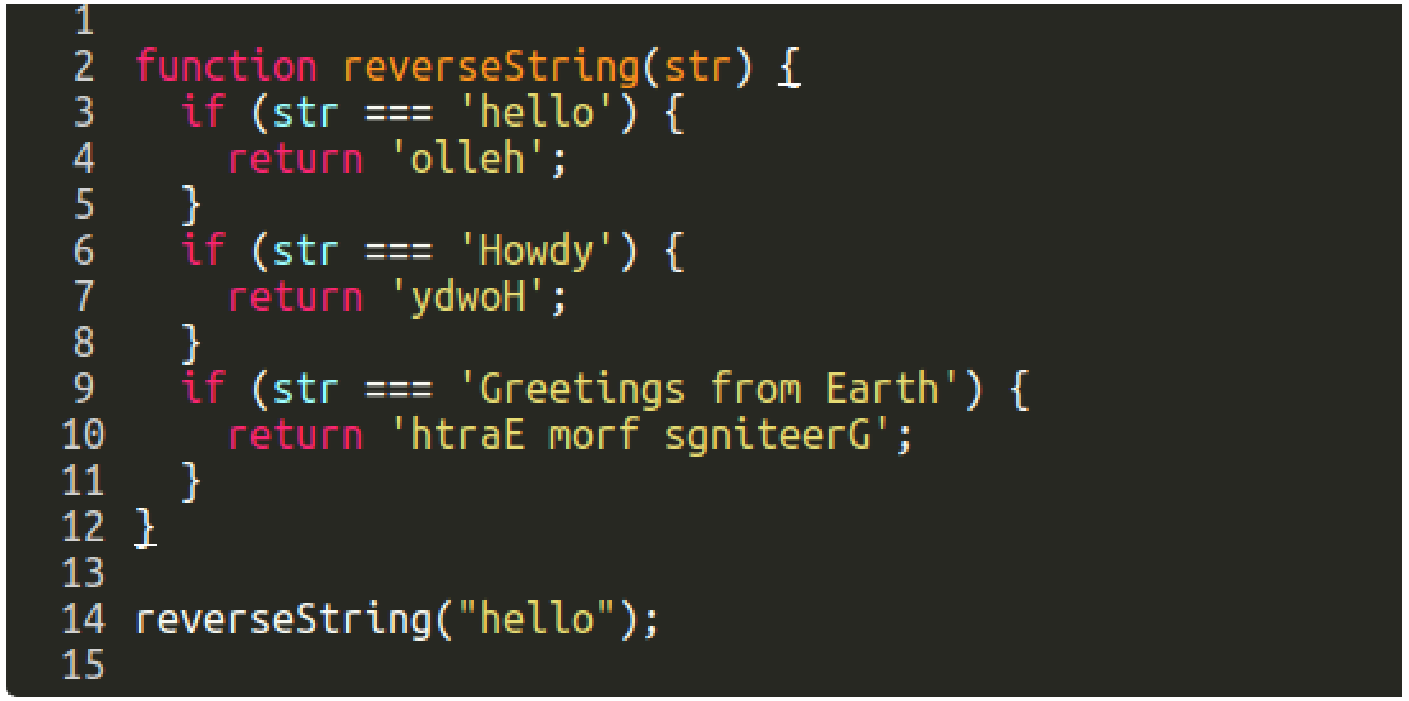


✓ `reverseString("hello")` should return a string.

✓ `reverseString("hello")` should become `"olleh"`.

✓ `reverseString("Howdy")` should become `"ydwoH"`.

✓ `reverseString("Greetings from Earth")` should return `"htraE morf sgniteerG"`.



```javascript

function reverseString(str) {
  if (str === 'hello') {
    return 'olleh';
  }
  if (str === 'Howdy') {
    return 'ydwoH';
  }
  if (str === 'Greetings from Earth') {
    return 'htraE morf sgniteerG';
  }
}

reverseString("hello");

```

Types of RE Errors and Flaws

- **Omission**
- **Contradiction**
- **Inadequacy**
- **Ambiguity**
- Unmeasurability
- Noise, overspecification
- Unfeasibility (wishful thinking)
- Unintelligibility
- Poor structuring, forward references
- Opacity

Omission and Contradiction

- Omission: problem/world feature not stated by any RD item (RD: requirement document)
  - e.g., no req about state of train doors in case of emergency stop
- Contradiction: RD items stating a problem/world feature in an incompatible way
  - "All doors must always be kept closed between platforms"
  - *and* "All doors must be opened in case of emergency stop"

Inadequacy and Ambiguity

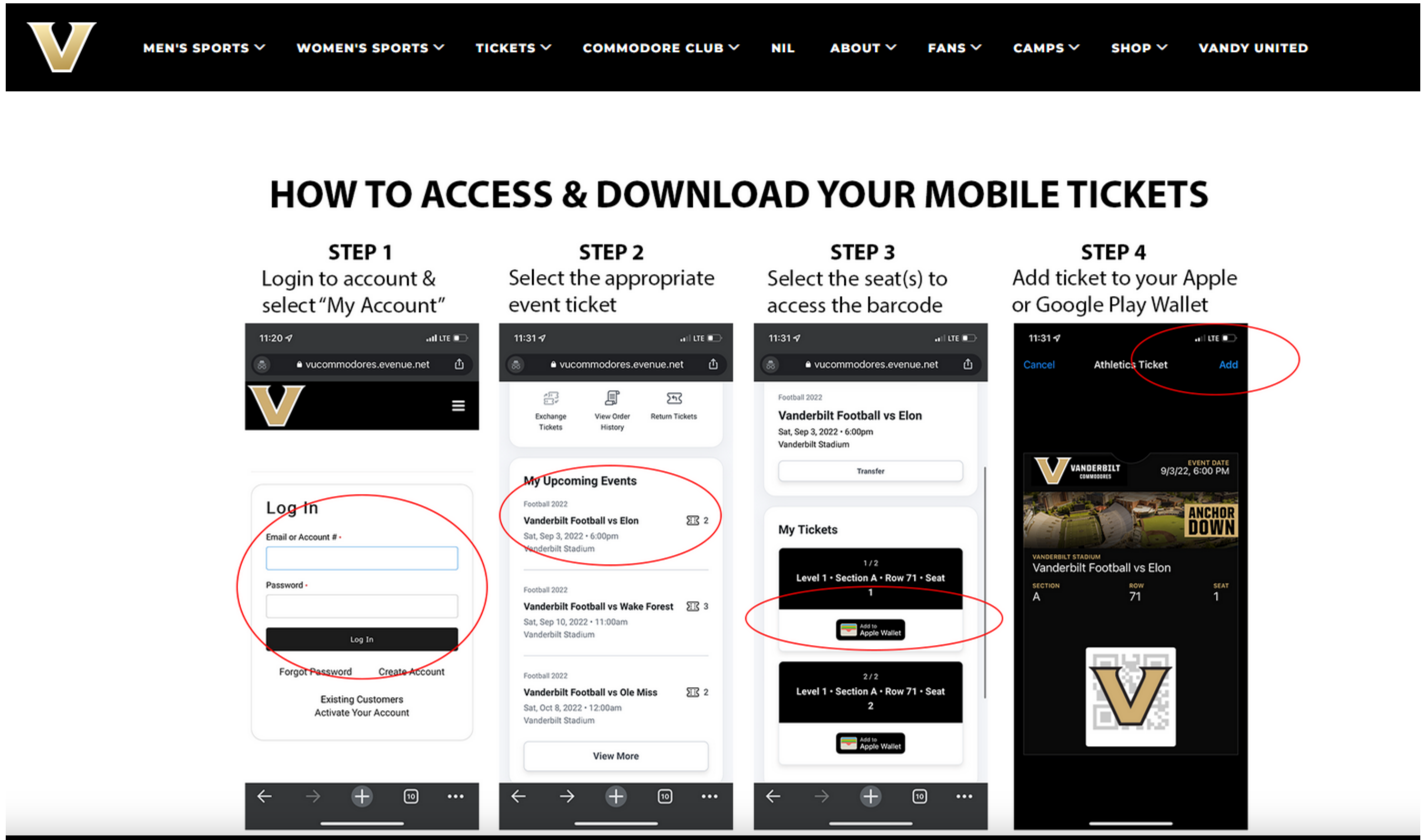
- Inadequacy: RD item not clearly stating a problem/world feature (“I need more to go on”)
  - “Panels inside trains shall display all flights served at next stop”
    - (Which panels? Which trains? Display how? What does “served” mean? *Flights* vs. Trains?)
- Ambiguity: RD item allowing a problem/world feature to be interpreted in different ways
  - “All doors shall be opened as soon as the train is stopped at platform”
    - (When do you start opening the doors?)

RD example

- https://www.indeed.com/career-advice/career-development/software-requirements-document-example
- https://www.cse.msu.edu/~cse435/Handouts/SRSExample-webapp.doc
- Many other examples can be found online

Requirements Engineering: Case Study

- Develop an online ticketing system
  - Kind of similar to the Vanderbilt Ticketing system, but you need to allow more clients (i.e., more than just Vanderbilt) and have both PC and mobile end.
  - https://vucommodores.com/mobile-ticketing/



HOW TO ACCESS & DOWNLOAD YOUR MOBILE TICKETS

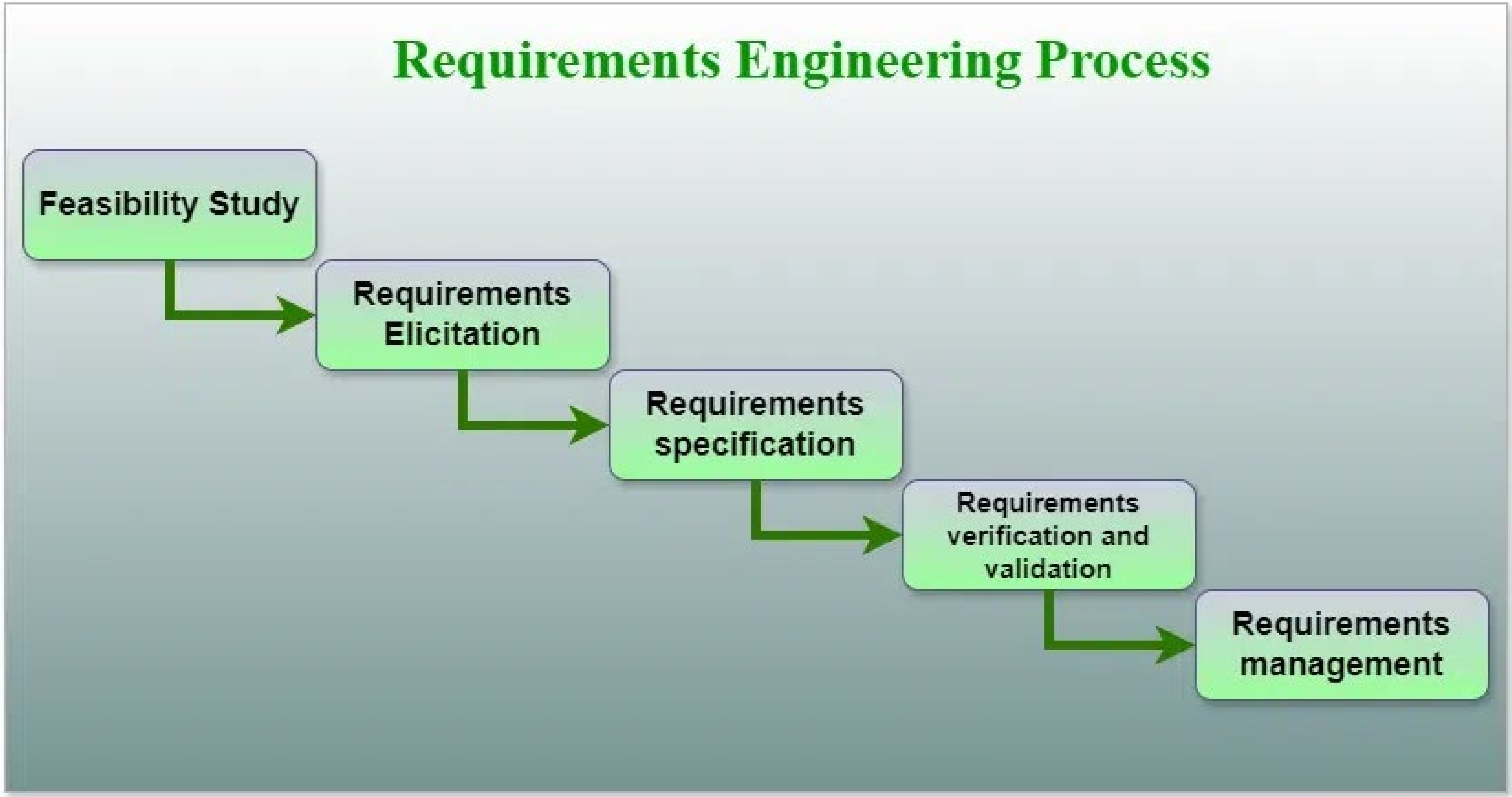
STEP 1
Login to account & select "My Account"

STEP 2
Select the appropriate event ticket

STEP 3
Select the seat(s) to access the barcode

STEP 4
Add ticket to your Apple or Google Play Wallet

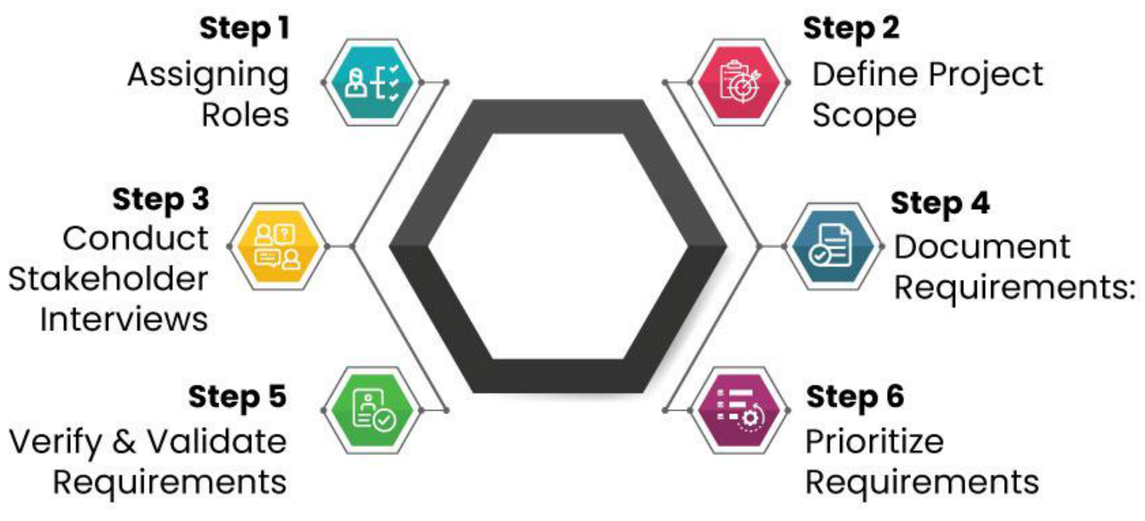
Requirements Engineering: Typical Process


Requirements Engineering Process
Feasibility Study
Requirements Elicitation
Requirements specification
Requirements verification and validation
Requirements management

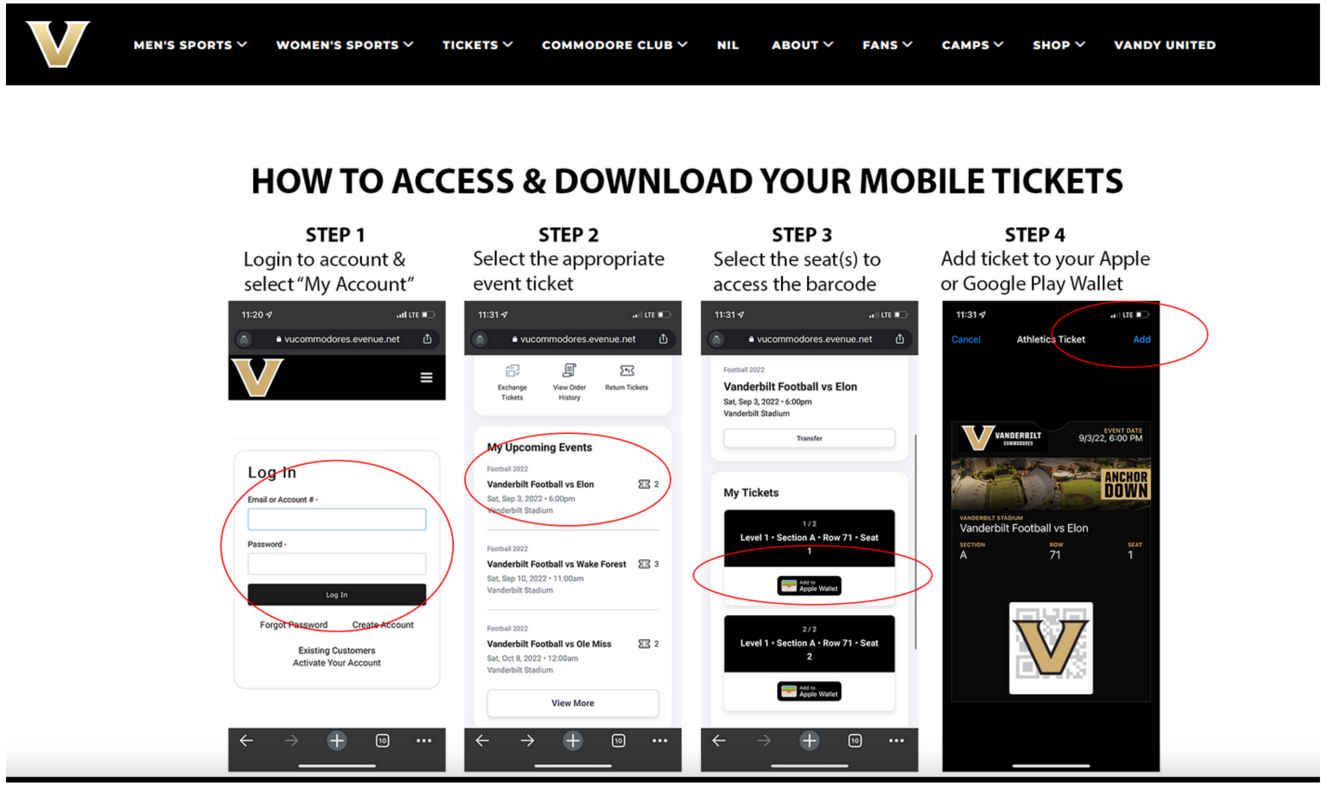
Requirement Elicitation: Case Study

Stakeholders:

<span>Product Manager</span>, users, clients (e.g., Taylor Swift's team), developer, operational team leader, designer, sponsor, development manager, legal team, marketing team, custom service team, product owner, etc.



Processes of Requirements Gathering in Software Development



HOW TO ACCESS & DOWNLOAD YOUR MOBILE TICKETS

STEP 1
Login to account & select "My Account"

STEP 2
Select the appropriate event ticket

STEP 3
Select the seat(s) to access the barcode

STEP 4
Add ticket to your Apple or Google Play Wallet

Quiz-alternative: in-class case study

- Write down your teammates’ NAME, VUID, stakeholder role
  - E.g., Yu Huang, huany47, client