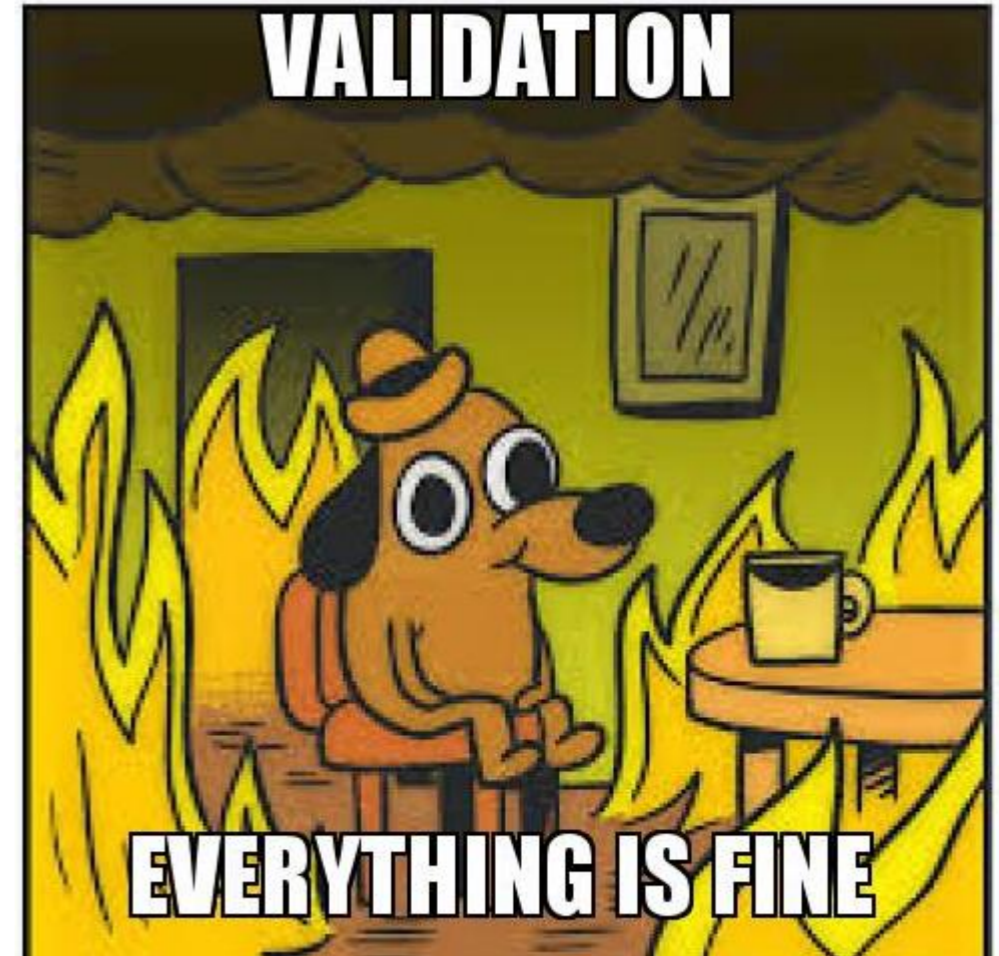
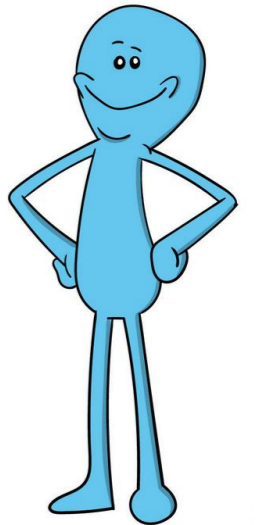


Requirements, Validation, and Risk



Review: Requirements

- **Requirements** are what the customer or stakeholder wants from software
 - **Requirements** describe what the system *will do*, **not** how to do it
 - “The system shall apply brakes if 6300kg load is detected.”
 - *Not* “The system will use a raspberry pi to actuate the Rolls Royce brake pad”
 - (that’s **implementation bias**)
 - **Requirements** are **measurable**
- **Communication is hard**
 - Misunderstandings of requirements cost the most
 - **Requirements and specification** sit at the **interface** between machine (system) and environment (world)
- **Upholding** a specification doesn’t immunize you from system **failure**



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

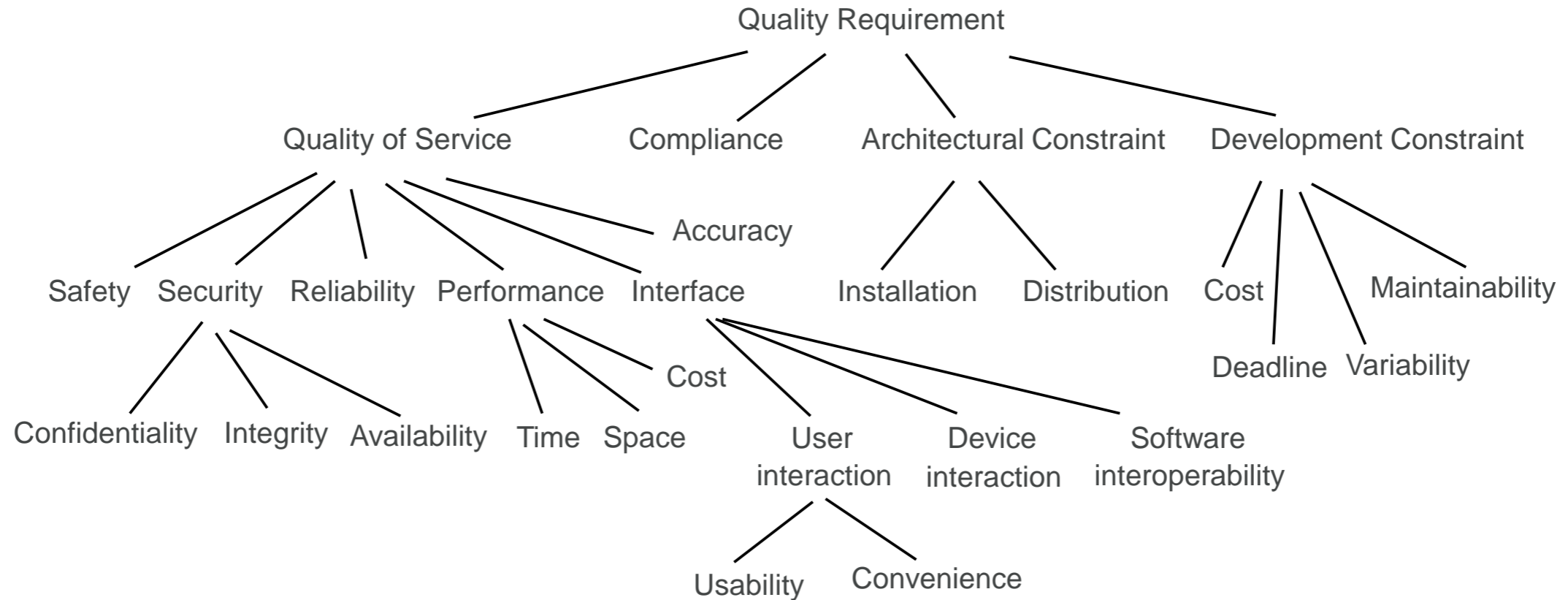
- **Quality requirements** specify *not* the functionality of the system, but *the manner in which* it delivers that functionality
- Can be more critical than functional requirements
 - Can work around missing functionality
 - Low-quality system may be unusable
- Examples?

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



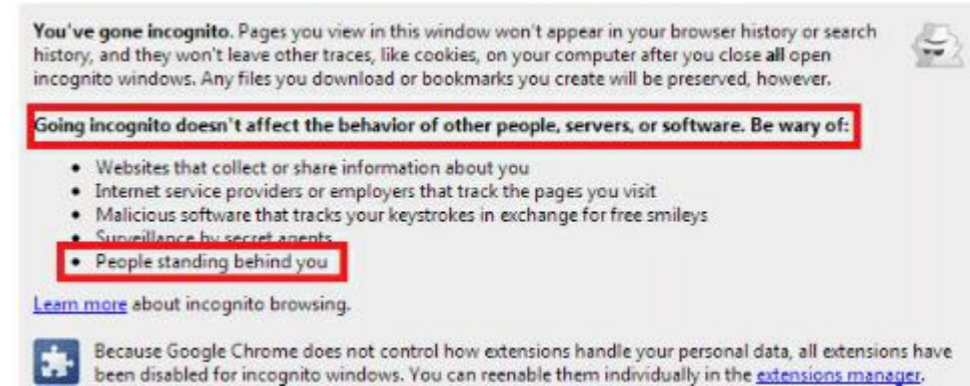
What are some of these for “selling videos on the web”?

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

- **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 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 Steps (Iterative)

- Identifying stakeholders
- Domain understanding
- Requirements elicitation (interviews, ...)
- Evaluation and agreement (conflicts, prioritization, risks, ...)
- Documentation and specification
- Consolidation and quality assurance (what?)



Target Qualities for RE Processes

- Completeness of objectives, requirements, assumptions
- Consistency of RD items
- Adequacy of requirements, assumptions, domain props
- Unambiguity of RD items
- Measurability of requirements, assumptions
- Pertinence of requirements, assumptions
- Feasibility of requirements
- Comprehensibility of RD items
- Good structuring of the RD
- Modifiability of RD items
- Traceability of RD items (where did we see this before?)

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"`.

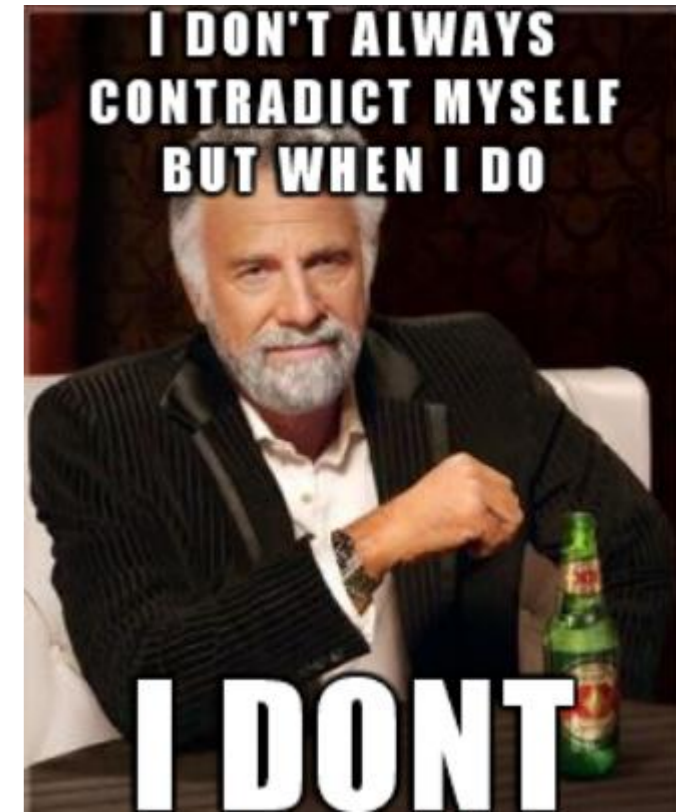
```
1
2 function reverseString(str) {
3   if (str === 'hello') {
4     return 'olleh';
5   }
6   if (str === 'Howdy') {
7     return 'ydwoH';
8   }
9   if (str === 'Greetings from Earth') {
10    return 'htraE morf sgniteerG';
11  }
12 }
13
14 reverseString("hello");
15
```

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
 - 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?)

Trivia: Woodworking

- This type of joinery uses a series of trapezoidal “pins” in one board that interlock with another board to resist being pulled apart. It is believed to predate written history, with examples in the tombs of Chinese emperors and entombed with First Dynasty Egyptian mummies.



Trivia: Musical Instruments

- This Austrian piano manufacturer famously produces a 97-key grand piano that is over 9 feet long (290 cm).
- The extra keys extend the keyboard down to C0, below the standard A1 on normal 88-key pianos, allowing 8 full octaves of pitch.



Psychology: Belief

- What factors influence our belief in a statement?
 - “You only use 10 percent of your brain. Eating carrots improves your eyesight. Vitamin C cures the common cold. Crime in the United States is at an all-time high.”
- We would like factors such as “evidence” or “validity” to be dominant
- Today we consider “repetition” and “ease”

Psychology: Belief

- Subjects were asked to rate how certain they were that 60 statements were true or false
 - “Zachary Taylor was the first president to die in office.” “Lithium is the lightest of all metals.” “The largest museum in the world is the Louvre in Paris.”
- Critically, subjects gave ratings on **three successive occasions** at two week intervals

Psychology: Illusory Truth Effect

- For both true and false statements, there was a **significant increase** in the validity judgments for the repeated statements (and no change for the non-repeated ones)
- [Lynn Hasher, David Goldstein, Thomas Toppino. Frequency and the Conference of Referential Transparency. J. Verbal Learning and Verbal Behavior, 1977.]

Psychology: Illusory Truth Effect

- Participants were exposed to false news stories portrayed as true news stories. After a five week delay, participants who had read the false experimental stories rated them as more truthful and more plausible than participants who had not been exposed to the stories. In addition, there was evidence of the **creation of false memories** for the source of the news story. *Participants who had previously read about the stories were more likely to believe that they had heard the false stories from a source outside the experiment.* These results suggest that repeating false claims will not only increase their believability but may also result in source monitoring errors. [Danielle Polage. Making up History: False Memories of Fake News Stories. Europe J. Psychology, 2012.]

Psychology: Illusory Truth Effect

- “When people make judgments about the truth of a claim, related but nonprobative information rapidly leads them to believe the claim: an effect called “truthiness”. ... Across all experiments, easily pronounced names trumped difficult names. Moreover, the effect of pronounceability produced truthiness for claims attributed to those names.”
[People with Easier to Pronounce Names Promote Truthiness of Claims. PLOS ONE, 2014.]
- Implications for SE? Process and requirements decisions are made based on evidence and claims. Who said: “Slogans should be persistently repeated until the very last individual has come to grasp the idea.”

One-Slide Summary:

Requirements Elicitation and Validation

- **Requirements elicitation** relies on communication with **stakeholders**. This includes identifying relevant parties, understanding the domain, interviews, and the exploration of alternatives. Requirements often conflict.
- **Validation** checks the correctness of requirements; **verification** checks the correctness of software.
- **Risk** includes both the likelihood and the consequence of failure.

Requirements Elicitation

- **Requirements elicitation** is the process of identifying system requirements through communication with stakeholders. Typically:

Step 1. Identify stakeholders

Step 2. Understand the domain

- Analyze artifacts, interact with stakeholders

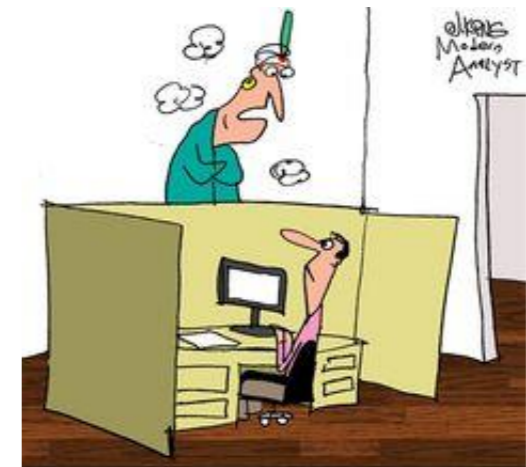
Step 3. Discover the real needs

- Interview stakeholders, resolve conflicts

Step 4. Explore alternatives to address needs

Stakeholder

- A **stakeholder** is any person or group who will be affected by the system, directly or indirectly
 - **Customers**, other parts of your own organization, regulatory bodies, etc.
- Stakeholders may disagree
- Requirements process should involve negotiation to resolve conflicts
- (We will return to conflicts)



*"Again this year, you get one wish...
but please don't waste it on
something even I can't grant, like
clear business requirements."*

Stakeholder Analysis

- Common criteria for **identifying** relevant stakeholders include:
- Relevant positions in the organization
- Effective role in **making decisions** about the system
- Level of domain expertise
- Exposure to perceived problems
- Influence in system **acceptance**
- Personal objectives and conflicts of interest

NASA Example of Stakeholders

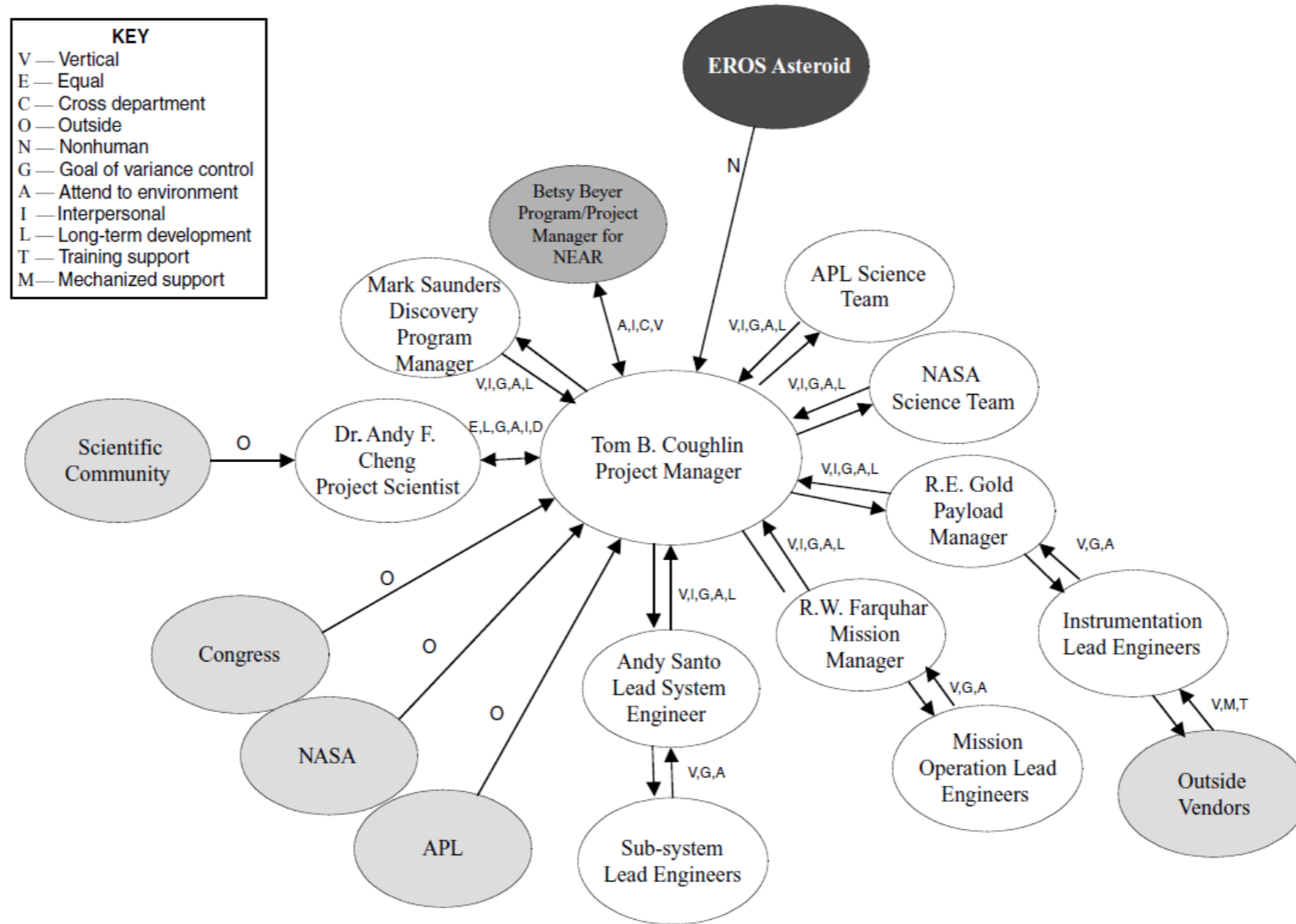


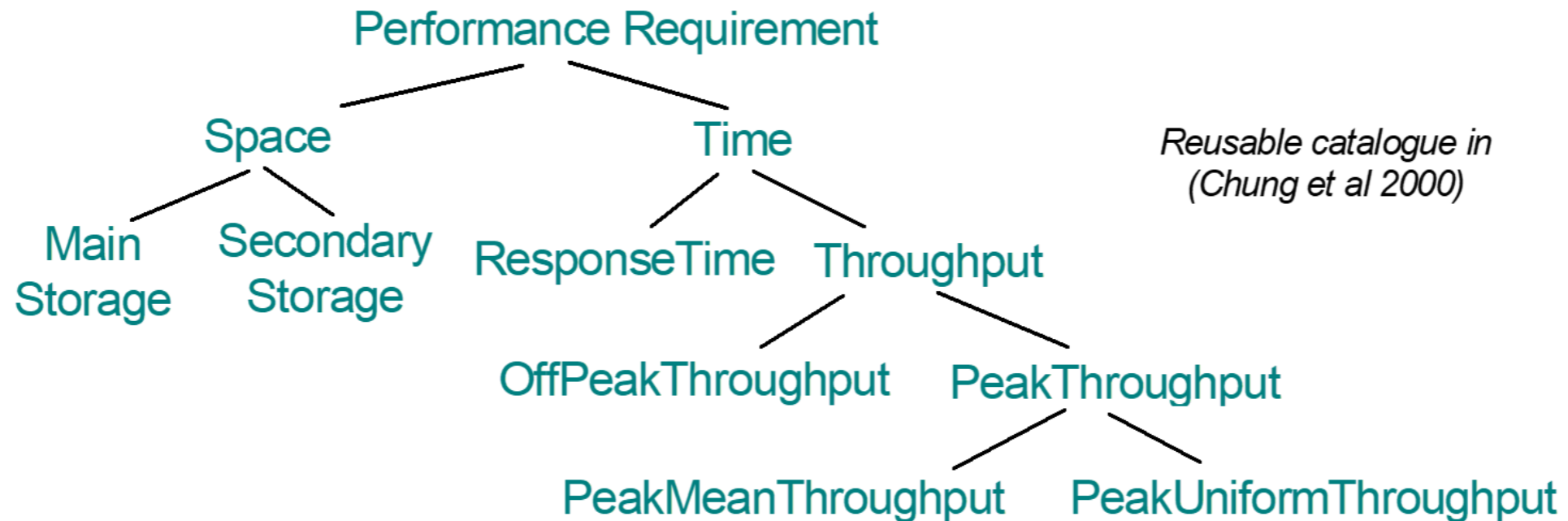
FIGURE 6-3 Role network for National Aeronautics and Space Administration (NASA's) Near Earth Asteroid Rendezvous project.

Step 2: Understanding the Domain

- **Content analysis** involves learning about the system domain
 - Books, articles, wikipedia, etc.
- This often focuses on the system to be built or replaced
 - How does it work? What are the problems? Are there manuals? Bug reports?
- But it also involves the organization
- And reusing knowledge from other systems

Domain-Independent Checklist

- Consider the list of qualities (from the previous lecture) and select relevant ones
 - Privacy, security, reliability, etc.
 - Even “performance” can be complicated:



Step 3: Discover Real Needs via Interviews

- Having identified stakeholders of interest and information to be gathered ...
- Conduct an **interview**



Step 3: Discover Real Needs via Interviews

- Having identified stakeholders of interest and information to be gathered ...
- Conduct an **interview**
 - This can be structured or unstructured, individual or group, etc.
 - It may even be a simple phone call
- Record and transcribe interview
- Report important finding
- Check validity of report with interviewee

Requirements Interview Advice

- Get basic facts about the interviewee before (role, responsibilities, ...)
- Review interview questions before interview
- Begin concretely with specific questions, proposals: work through prototype or scenario
- Be open-minded; explore additional issues that arise naturally, but stay focused on the system
- Contrast with current system or alternatives
 - Explore conflicts and priorities
- Plan for follow-up questions



TECH SUPPORT

The only industry where, despite the fact that the customer is always right, the customer is always wrong.

Example: Identifying Problems (1)

- What problems do you run into in your day-to-day work? Is there a standard way of solving it, or do you have a workaround?
 - Why is this a problem? How do you solve the problem today? How would you ideally like to solve the problem?
- **Keep asking follow-up questions** (“What else is a problem for you?”, “Are there other things that give you trouble?”) for as long as the interviewee has more problems to describe

Example: Identifying Problems (2)

- So, as I understand it, you are experiencing the following problems/needs ...
 - Describe the interviewee's problems and needs in your own words: often **you do not share** the same image. It is very very common to not understand each other even if at first you think you do.
- Just to confirm, have I correctly understood the problems you have with the current solution?
 - Are there any other problems you're experiencing? If so, what are they?

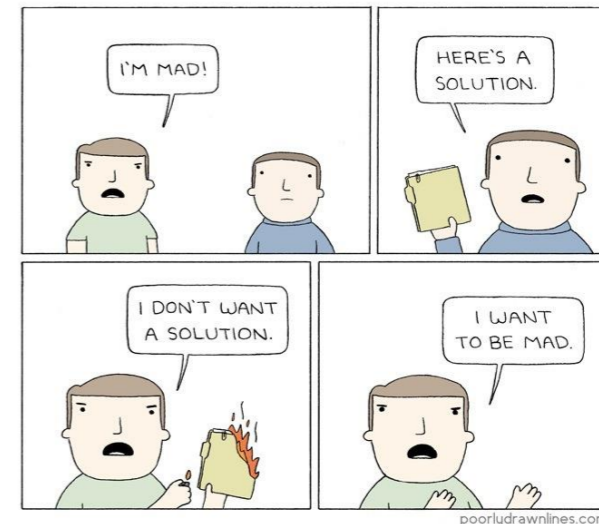
Interview Tradeoffs

- Strengths

- Reveal what stakeholders do, feel, prefer
- How they interact with the system
- Challenges with current systems

- Weaknesses

- Subjective, yield inconsistencies
- Hard to capture domain knowledge
- Organizational issues, such as politics
- Hinges on interviewer skill



Capturing and Synthesizing

- We acquire requirements from many sources
 - Elicit from stakeholders
 - Extract from policies or other documentation
 - Synthesize from above: estimation and invention
- Stakeholders do not always know what they want (!)
 - Be faithful to stakeholder needs and expectations
 - Anticipate additional needs and risks
 - Validate that “additional needs” are necessary or desired

Observation and Ethnography

- Observe people using their current system
- **Passive**: no interference with task performers
 - Watch from outside, record (notes, video), edit transcripts, interpret
 - Protocol analysis: they concurrently explain it
- **Active**: you get involved in the task, even become a team member
- Ethnographic studies, over long periods of time, discover emergent properties of social group involved

Analogy: Ethnography



- (Dr. Margaret Mead in Samoa, 1975)

Mead vs. Freeman (1)

- In her popular 1928 book, *Coming of Age in Samoa*, Mead presented Samoan culture in a particular light
 - Hypothesis: adolescence is a function of surrounding culture
 - Other societies don't shun certain behaviors
 - Based on observations, interviews, ethnographic studies, etc.
- Mead almost certainly had a political agenda (she was a progressive, etc.)
 - But that did not make her wrong

Mead vs. Freeman (2)

- In **1983**, Freeman's *Margaret Mead and Samoa: The Making and Unmaking of an Anthropological Myth*, suggested that Mead was just gullible. Two of her informants had been lying: “Never can giggly fibs have had such far-reaching consequences in the groves of Academe.”
 - This significantly discredited her work
- It seemed his follow-on interviews found very different results. How could that be?

Mead vs. Freeman (3)

- Basically, *Freeman* was lying
- In 1996, Orans used Mead's own notes to show that “such humorous fibbing could not be the basis of Mead's understanding. Freeman asks us to imagine that the joking of two women ... was of more significance than the detailed information she had collected throughout her fieldwork.”

Mead vs. Freeman (4)

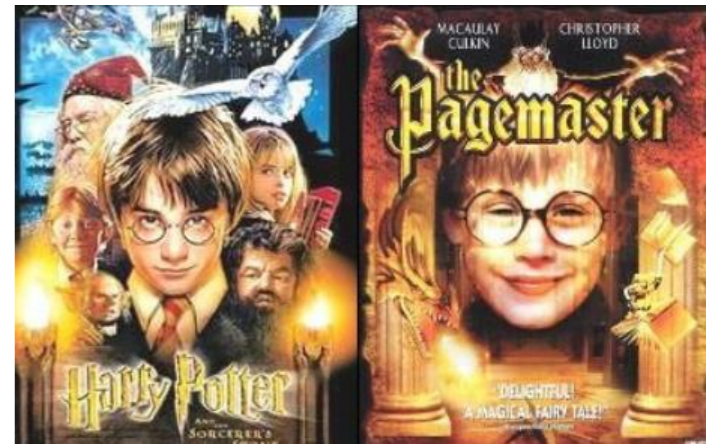
- In 2011, Shankman used Freeman's own notes and found that his interviews were conducted in problematic ways:
 - One interviewee felt cast in a negative light
 - Freeman told the interviewees: “the purpose of the interview is to correct the lies Mead wrote in her book—lies that insult you all.”

Mead vs. Freeman (5)

- Ultimately, ethnography is complicated
 - Personal views of each individual
 - Does the observation we're making influence the outcome of the ethnographic study?
 - Will others conducting information cast questions in a biased way?
- SE Implication: Gathering requirements with ethnographic studies is hard.

Identifying Conflicts: Inconsistencies

- **Terminology** clash: same concept named differently in different statements
 - e.g., library: “borrower” vs. “patron”
- **Designation** clash: same name for different concepts in different statements
 - e.g., “user” for “library user” vs. “library software user”
- **Structure** clash: same concept structured differently in different statements
 - e.g., “latest return date” as time point (e.g. Fri 5pm) vs. time interval (e.g. Friday)



Conflict Strength

- In a **strong conflict**, statements are not satisfiable together
 - e.g., “participant constraints may not be disclosed to anyone else” vs. “the meeting initiator must know participant constraints”
- In a **weak conflict (divergence)**, statements are not satisfiable together under **some** boundary condition
 - e.g., “patrons shall return borrowed copies within X weeks” vs “patrons may keep borrowed copies as long as needed” contradicts only if “needed $> X$ ”

Resolving Conflicts



- “No Silver Bullet”
(this is why they pay you)
- For Terminology, Designation and Structural conflicts: **build a glossary**
- For Weak and Strong Conflicts: negotiation is typically required
 - If the cause is different stakeholder objectives, it must be resolved outside of RE
 - If the cause is quality desires (e.g., “Good, cheap, on-time: pick two”), you **explore quality tradeoffs**

Step 4: Explore Alternatives

- Alternative solutions and tradeoffs are typically presented via **prototypes**, **mockups** or **storyboards**
- Mockups can be low- or high-fidelity
- Rapid prototypes can be throw-away (designed to learn about the problem, not for actual use) or evolutionary (intended to be incorporated into the final product)
- Stories detail **who** the players are, **what** happens to them, **how** it happens, why it happens, and what could go wrong

Informality

- Storyboards and mockups definitely do exist, but are often informal and incomplete



Exploration

- Humans are better at **recognizing and evaluating** solutions than facing blank pages
- Mockups and prototypes explore uncertainty in requirements
 - Validate that we have the right requirements
 - Get feedback on a candidate solution
 - “I'll know it when I see it.”
- Stories illuminate the system by walking through real or hypothetical sequences

Requirements Documentation

- Formal **standards** for writing down requirements exist (e.g., “may” vs. “must”) but are not a focus for this course
- They vary by domain and company (e.g., startup vs. established)



At last, he has found the famous Requirements Document dating back to the Traditional Age.

Requirements Elicitation: Reminder

- **Requirements elicitation** is the process of identifying system requirements through communication with stakeholders. Typically:

Step 1. Identify stakeholders

Step 2. Understand the domain

- Analyze artifacts, interact with stakeholders

Step 3. Discover the real needs

- Interview stakeholders, resolve conflicts

Step 4. Explore alternatives to address needs

Requirements for Requirements?

- Correct
- Consistent
- Unambiguous
- Complete
- Feasible
- Relevant
- Testable
- Traceable



Verification and Validation

- **Validation** is the task of determining if the **requirements** are correct
 - Are the requirements complete? Do they reflect the client's problem? Are they consistent?
- **Verification** is the task of determining if the **software** is correct (e.g., by testing)
 - Does the software satisfy the specification?
 - Is the specification correct with respect to the requirements, assuming the domain properties hold?

Approaches Validation

- Interviews
- Reading
- Walkthroughs
- Prototypes
- Scenarios
- Checklists
- Modeling

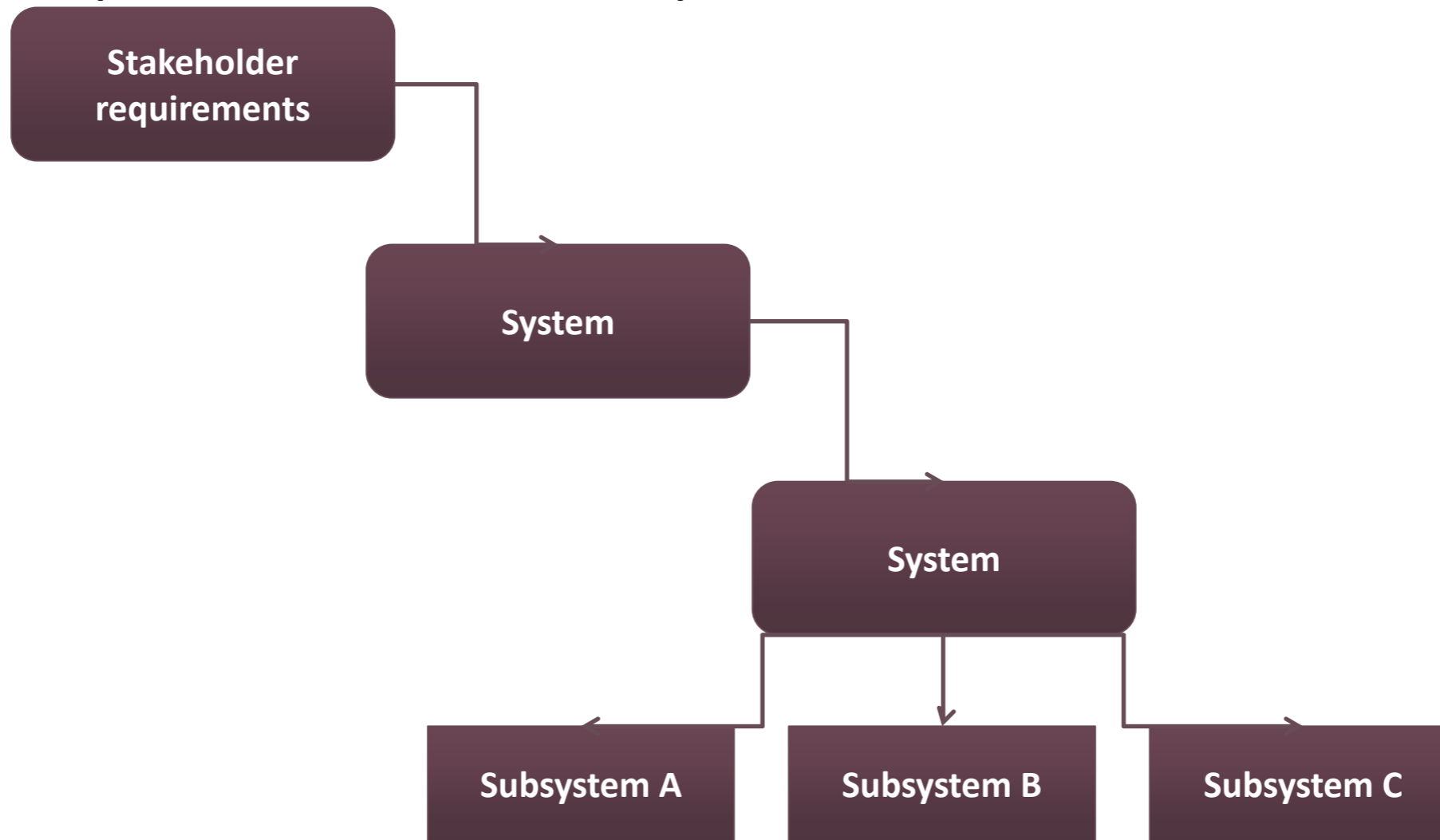
Verification

- Testing
- Mathematical proofs
- Simulation
- Static analysis
- Dynamic analysis
- Checks for unreachable states or transitions (model checking)

Decomposition

- We recursively **decompose** a system, from the highest level of abstraction (stakeholder requirements) into lower-level subsystems and implementation choices
- This decomposition establishes **traceability**, which identifies relationships between requirements and implementations
- Traceability is important for verification and when requirements **change**
- Decomposition helps both validate and verify

Decomposition Example

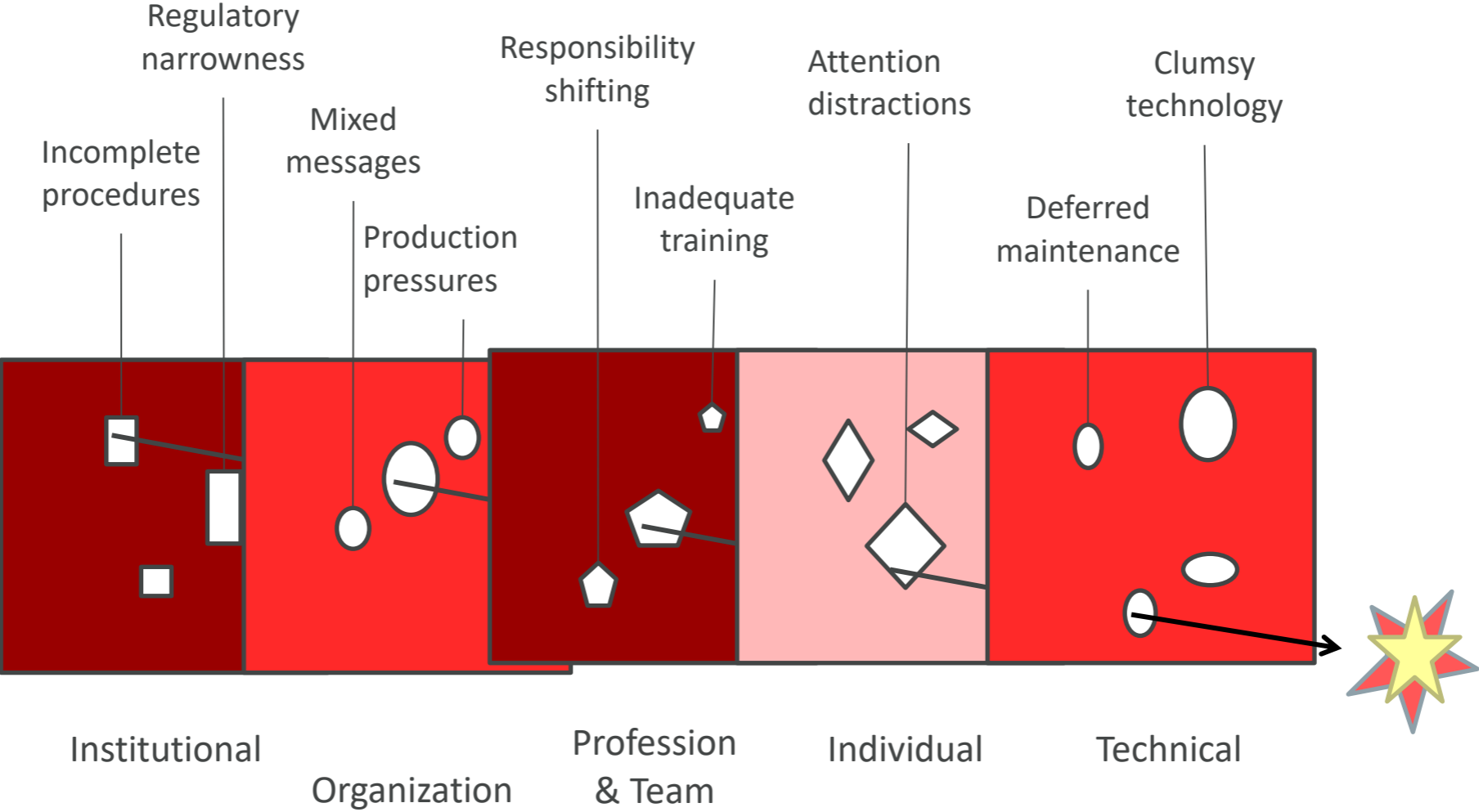


Risks



- A **risk** is an uncertain factor that may result in a loss of satisfaction of a corresponding objective
- For example:
 - The system delivers a radiation overdose to patients (Therac-25, Theratron-780)
 - Medication administration record (MAR) knockout (provided inaccurate medication plans hospital-wide)
 - Premier Election Solutions vote-dropping “glitch”

Swiss Cheese Model



Modified from Reason, 1999, by R.I. Crook

Risk Assessment

- Risk consists of multiple parts:
 - The likelihood of failure
 - The negative consequences or impact of failure
 - In advanced models: the causal agent and weakness
- Mathematically,
- **Risk = Likelihood · Impact**



Example: CVSS V2.10 Scoring

- The Common Vulnerability Scoring System consists of:
 - 6 base metrics (access vector, complexity, confidentiality impact, ...)
 - 3 temporal metrics (exploitability, remediation, ...)
 - 5 environmental metrics; all qualitative ratings (collateral damage, ...)
- $\text{BaseScore} = \text{round_to_1_decimal}(((0.6 * \text{Impact}) + (0.4 * \text{Exploitability}) - 1.5) * f(\text{Impact}))$
- $\text{Impact} = 10.41 * (1 - (1 - \text{ConfImpact}) * (1 - \text{IntegImpact}) * (1 - \text{AvailImpact}))$
- $\text{Exploitability} = 20 * \text{AccessVector} * \text{AccessComplexity} * \text{Authentication}$
- $f(\text{Impact}) = 0$ if $\text{Impact} = 0$, 1.176 otherwise

Example: DO-178b

Aviation Failure Impact Categories

- *No effect* – failure has no impact on safety, aircraft operation, or crew workload
- *Minor* – failure is noticeable, causing passenger inconvenience or flight plan change
- *Major* – failure is significant, causing passenger discomfort and slight workload increase
- *Hazardous* – high workload, serious or fatal injuries
- *Catastrophic* – loss of critical function to safely fly and land

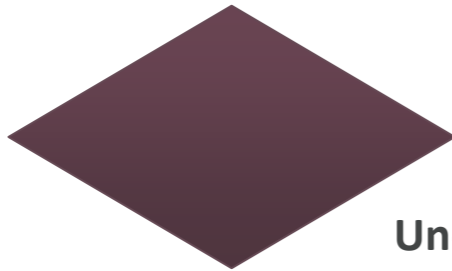
Fault Tree Analysis

- **Fault tree analysis** is a top-down technique to model, reason about, and analyze risk
- A fault tree analysis decomposes a particular type of failure into constituent potential causes and probabilities
- It defines the scope of system responsibilities and identifies unacceptable risk conditions that should be mitigated

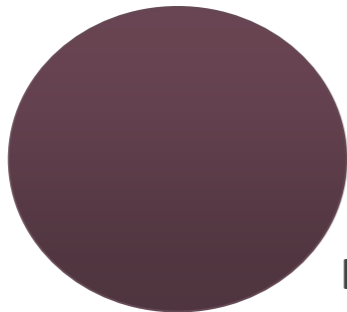
Fault Tree Diagrams



Top-level or intermediate event



Undeveloped event



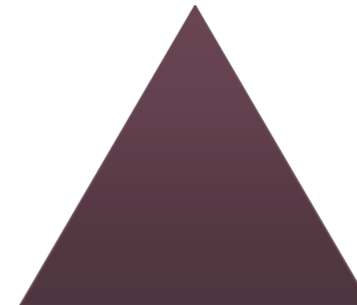
Basic event



Or gate

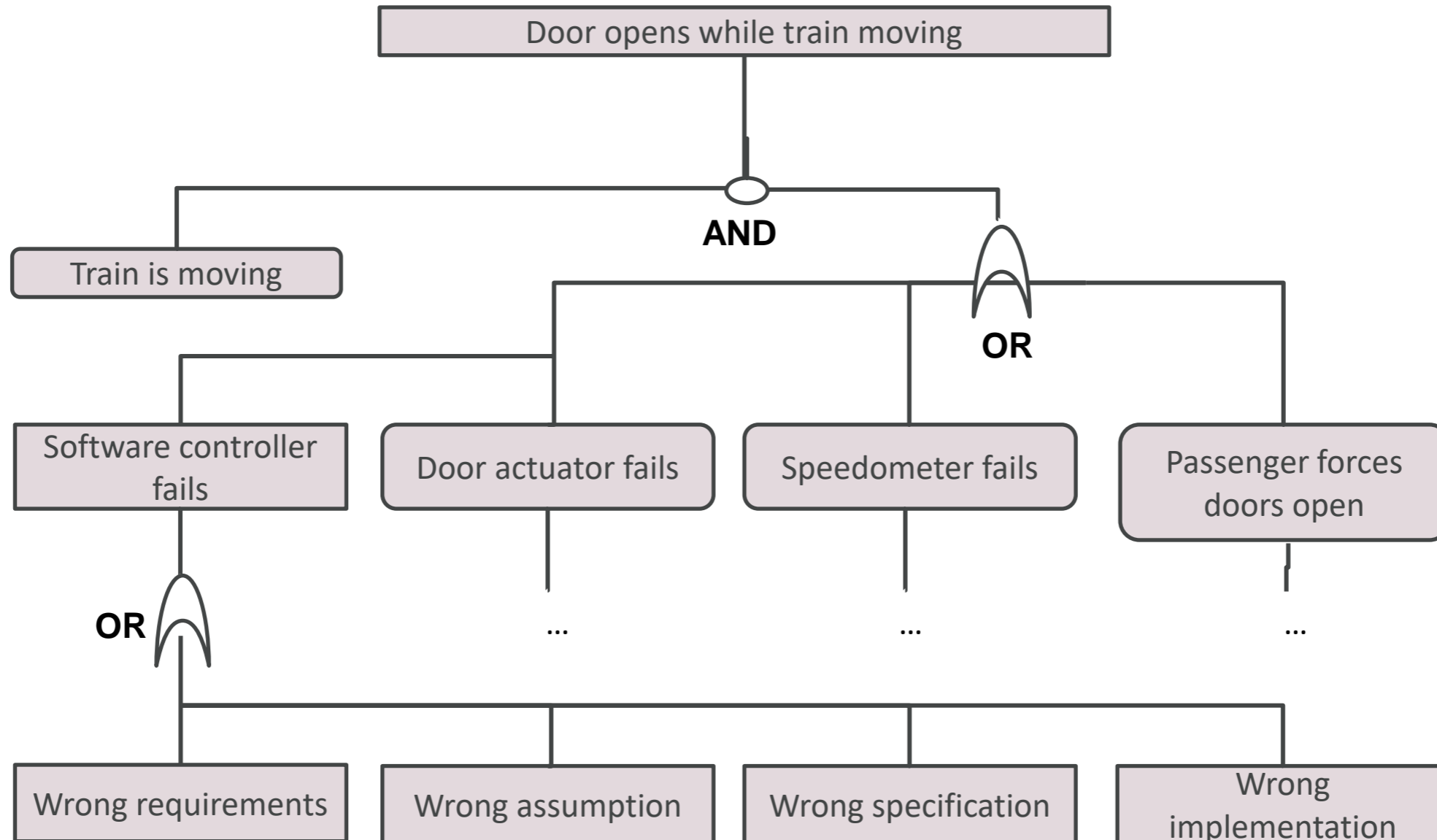


And gate



Transfer gate

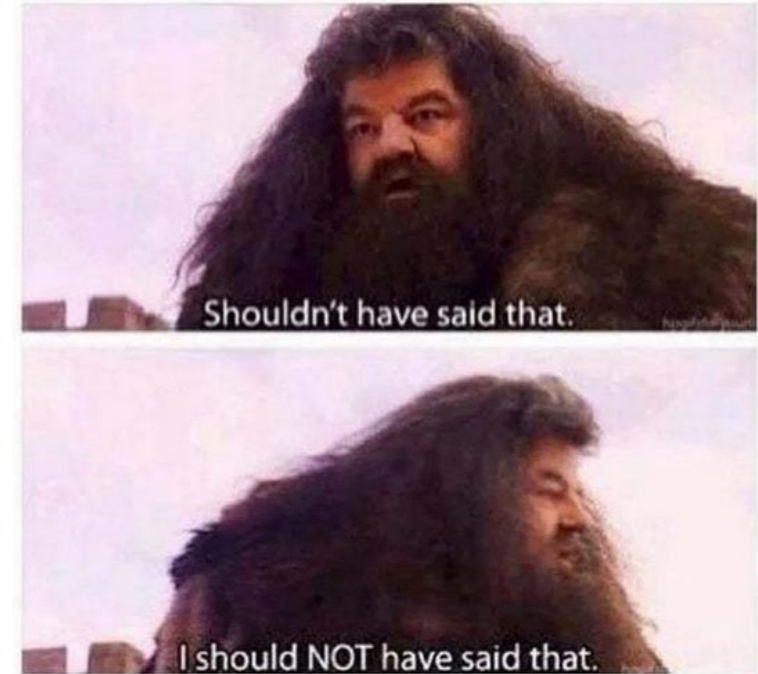
Example Fault Tree to Quantify Risk



When they don't respond to a risky text within 2 minutes

Risk Response Strategies

- **Accept** the risk: for low likelihood or low impact risks, or where the cost of mitigation is too high
- **Transfer** the risk: push the risk outside the system boundary
- **Mitigate** the risk: introduce active countermeasures
 - Reduce likelihood of failure; reduce severity of impact; change *ors* to *ands*!
- **Avoid** the risk: redesign so that risk cannot occur



Questions?

