Requirements and Specifications (Part 2/2)

Martin Kellogg

Requirements and Specifications (Part 2)

Today's agenda:

- **Reading Quiz**
- Requirements elicitation
- Formal specifications
- A special, totally-unpredictable surprise

Reading quiz: reqs and specs part 2

Q1: **TRUE** or **FALSE**: The author argues that formal specifications are good for showing your code matches your expectations but very bad for showing your expectations match your needs.

Q2: In the project's _____, you will show what your project will look like, once it is complete, without actually running any code.

A. Deployment Demo
B. Preliminary Demo
C. Chinese Room Demo
D. Wizard-of-Oz Demo

Reading quiz: reqs and specs part 2

Q1: **TRUE** or **FALSE**: The author argues that formal specifications are good for showing your code matches your expectations but very bad for showing your expectations match your needs.

Q2: In the project's _____, you will show what your project will look like, once it is complete, without actually running any code.

A. Deployment Demo
B. Preliminary Demo
C. Chinese Room Demo
D. Wizard-of-Oz Demo

Reading quiz: reqs and specs part 2

Q1: **TRUE** or **FALSE**: The author argues that ~~formal specifications~~ **TESTS** are good for showing your code matches your expectations but very bad for showing your expectations match your needs.

Q2: In the project's _____, you will show what your project will look like, once it is complete, without actually running any code.

A. Deployment Demo
B. Preliminary Demo
C. Chinese Room Demo
D. Wizard-of-Oz Demo

Reading quiz: reqs and specs part 2

Q1: **TRUE** or **FALSE**: The author argues that ~~formal specifications~~ **TESTS** are good for showing your code matches your expectations but very bad for showing your expectations match your needs.

Q2: In the project's _____, you will show what your project will look like, once it is complete, without actually running any code.

A. Deployment Demo
B. Preliminary Demo
C. Chinese Room Demo
D. Wizard-of-Oz Demo

Requirements and Specifications (Part 2)

Today's agenda:

- Reading Quiz
- **Requirements elicitation**
- Formal specifications
- A special, totally-unpredictable surprise

Requirements and Specifications (Part 2)

Today's agenda:

- Reading Quiz
- **Requirements elicitation**
- Formal specifications
- A special, totally-unpredictable surprise

Announcements:

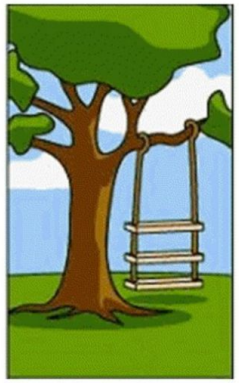
- Project groups were emailed to you last night
- IP1 grading will be done soon (hopefully today)

Requirements

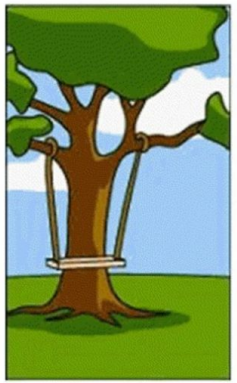
- The document I gave you during the GroupThink game was an example of a set of **requirements**

Requirements

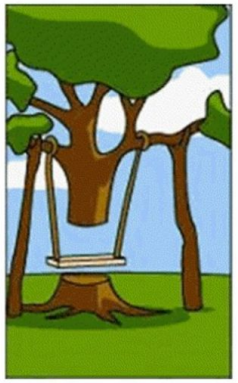
- The document I gave you during the GroupThink game was an example of a set of **requirements**
  - **where** do requirements come from?
  - **what kinds** of requirements are there?
  - **why** is this related to specification?



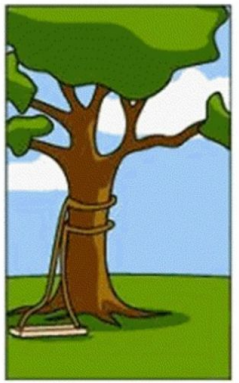
How the customer explained it



How the project leader understood it

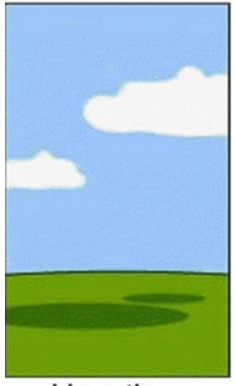


How the engineer designed it

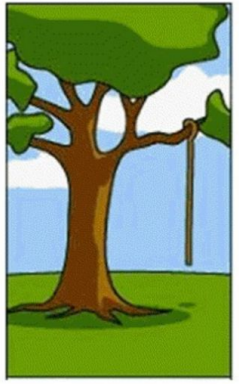


How the programmer wrote it

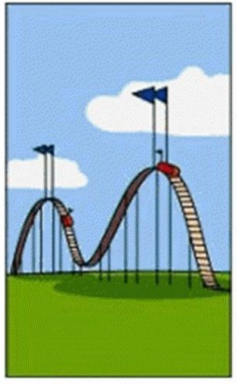
How the sales executive described it



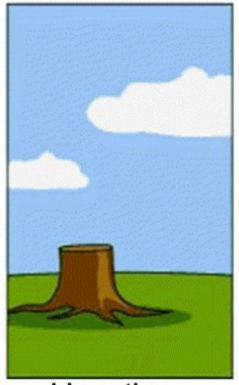
How the project was documented



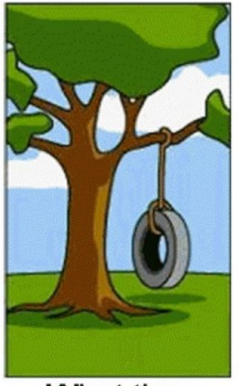
What operations installed



How the customer was billed



How the help desk supported it



What the customer really needed

Requirements elicitation

- Option 1: **users tell** developers what they want
  - Client determines the problem and the solution
  - Requirements might be formally provided in the form of a contract or statement of work
  - Client might provide all requirements, or just some subset (e.g., “must be HIPAA compliant”)

Requirements elicitation

- Option 1: **users tell** developers what they want
  - Client determines the problem and the solution
  - Requirements might be formally provided in the form of a contract or statement of work
  - Client might provide all requirements, or just some subset (e.g., “must be HIPAA

Not always possible: clients often **don’t know what they want**

Requirements elicitation

- Option 2: **direct research**
  - Interview users, ask questions about their problems, propose potential solutions, examine those solutions
  - Embed your client in your design team, or better yet, become an **anthropologist** in your client's environment
  - Build requirements documents that demonstrate your understanding of the requirements, iterate

Requirements elicitation

- Option 2: **direct research**
  - Interview users, ask questions about their problems, propose potential solutions, examine those solutions
  - Embed your client in your design team, or better yet, become an **anthropologist** in your client's environment
  - Build requirements documents that demonstrate your understanding of the requirements, iterate
  - Empowers your team with **credibility and authority**

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

- Input, Output
- Interface
- Response to events
- etc.

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

- Input, Output
- Interface
- Response to events
- etc.

Properties of a good functional spec:

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

- Input, Output
- Interface
- Response to events
- etc.

Properties of a good functional spec:

- **Completeness**: All requirements are documented

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

- Input, Output
- Interface
- Response to events
- etc.

Properties of a good functional spec:

- **Completeness**: All requirements are documented
- **Consistency**: No conflicts between requirements

What kinds of requirements are you eliciting?

Definition: a ***functional specification*** is a description of what a system should do that doesn't specify how the system should do it

- Input, Output
- Interface
- Response to events
- etc.

Properties of a good functional spec:

- **Completeness**: All requirements are documented
- **Consistency**: No conflicts between requirements
- **Precision**: No ambiguity in requirements

What kinds of requirements are you eliciting?

- But, not all specs are functional!

What kinds of requirements are you eliciting?

- But, not all specs are functional!

Definition: A ***quality requirement*** specifies not the functionality of the system, but the manner in which it delivers that functionality

What kinds of requirements are you eliciting?

- But, not all specs are functional!

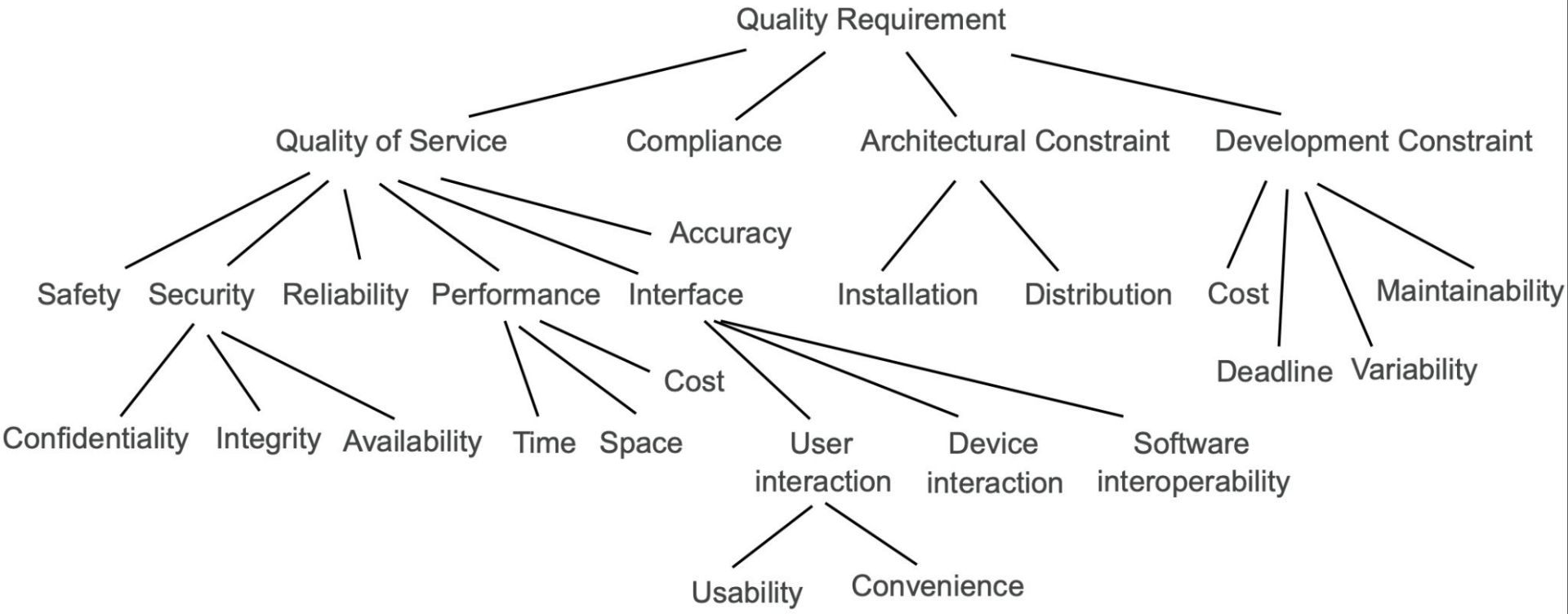
Definition: A ***quality requirement*** specifies not the functionality of the system, but the manner in which it delivers that functionality

Quality requirements can be more important than functional requirements:

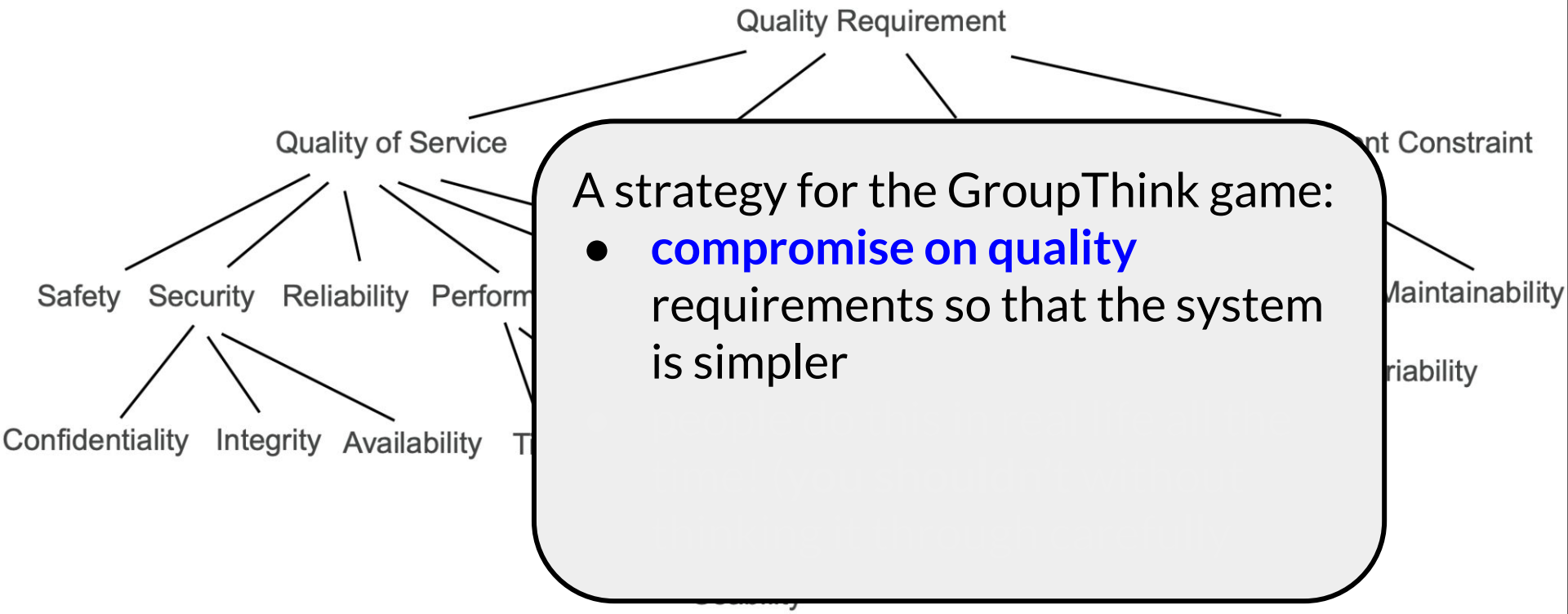
- Can work around missing functionality
- Low-quality system may be unusable

Examples of quality requirements

Examples of quality requirements



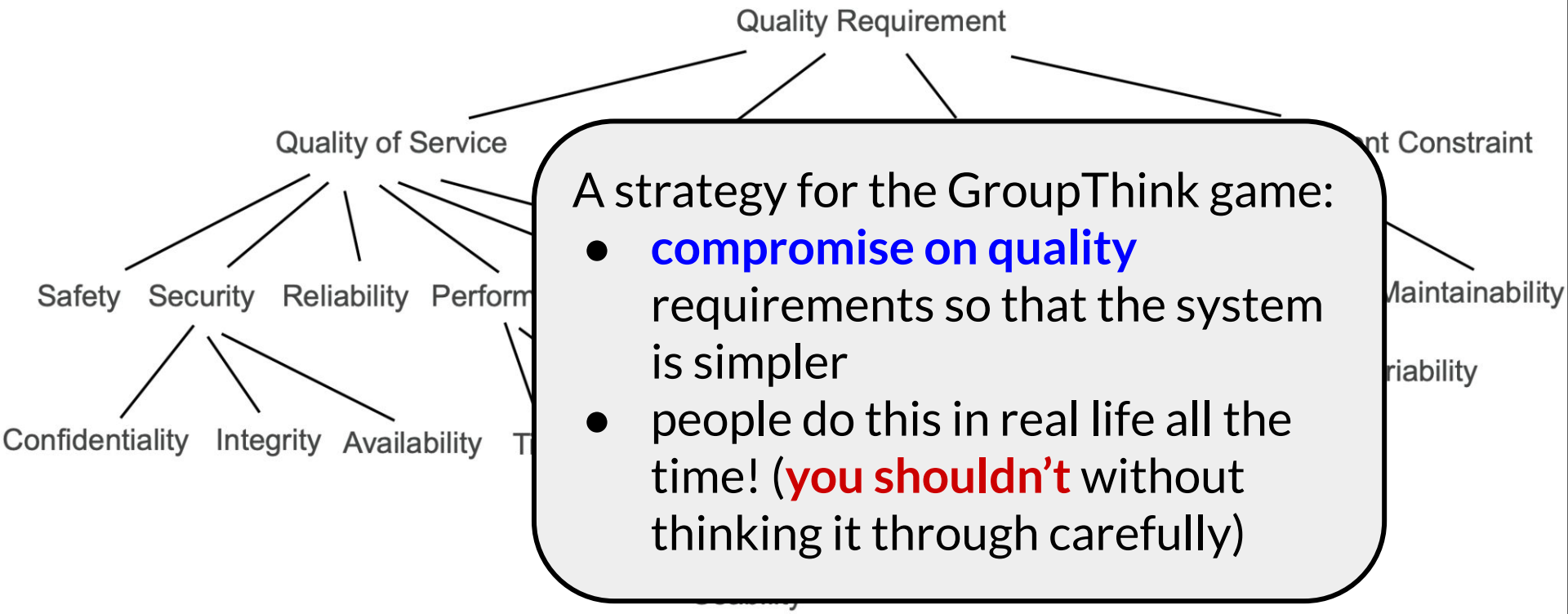
Examples of quality requirements



A strategy for the GroupThink game:

- **compromise on quality** requirements so that the system is simpler

Examples of quality requirements



A strategy for the GroupThink game:

- **compromise on quality** requirements so that the system is simpler
- people do this in real life all the time! (**you shouldn't** without thinking it through carefully)

Can we have all the quality requirements?

- Who is going to ask for a slow, inefficient, unmaintainable system?

Can we have all the quality requirements?

- 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

Can we have all the quality requirements?

- 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?

Can we have all the quality requirements?

- 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?

Trade-offs!

Expressing Quality Requirements

- Requirements serve as **contracts**: they should be **testable/falsifiable**

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

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

Expressing Quality Requirements

- Requirements serve as **contracts**: they should be **testable/falsifiable**
- An **informal** goal is a general [illegible] or "high security")
  - May still be helpful to dev[illegible] intentions of the system users
- A **verifiable** non-functional requirement is a statement using some measure that can be objectively tested

Advice: when possible, make your quality requirements verifiable

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.”

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.”

Requirements and Specifications (Part 2)

Today's agenda:

- Reading Quiz
- Requirements elicitation
- **Formal specifications**
- A special, totally-unpredictable surprise

How to write specifications

Many ways:

How to write specifications

Many ways:

- as English prose (e.g., the group think game's spec)
- as semi-structured English prose (e.g., user stories)
- as structured English (***formal specification***)
- as an executable formal specification

How to write specifications

Many ways:

- as **English prose** (e.g., the group think game's spec)
- as semi-structured English prose (e
- as structured English (*formal speci*
- as an executable formal specificat

Technical writing courses should cover how to do this

How to write specifications

Many ways:

- as English prose (e.g., the group think game's spec)
- as semi-structured English prose (e.g., **user stories**)
- as structured English (***formal specification***)
- as an executable formal specification

User stories

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

User stories

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

- also requires a ***condition of satisfaction***, which is the measurement you will use to decide if the user story has been completed

User stories: examples

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

User stories: examples

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

- “As a **computer user**, I want to **backup my entire hard drive** so that **my files are safe**”

User stories: examples

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

- “As a **computer user**, I want to **backup my entire hard drive** so that **my files are safe**”
- “As a **typical computer user**, I want to **specify folders to backup**, so that **my most important files are safe**”

User stories: examples

“As a <**role**>, I can <**capability**>, so that I can <**receive benefit**>.”

- “As a **computer user**, I want to **backup my entire hard drive** so that **my files are safe**”
- “As a **typical computer user**, I want to **specify folders to backup**, so that **my most important files are safe**”
- “As a **power user**, I want to **specify subfolders and filetypes NOT to backup**, so that **my backup doesn’t fill up with things that I don’t need to preserve**”

Writing user stories: INVEST principles

User stories should be:

Writing user stories: INVEST principles

User stories should be:

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

How to write specifications

Many ways:

- as English prose (e.g., the group think game's spec)
- as semi-structured English prose (e.g., user stories)
- as **structured English** (***formal specification***)
- as an executable formal specification

Formal specifications

Definition: a ***formal specification*** is a model of a system's design expressed in a formal language

Formal specifications

Definition: a ***formal specification*** is a model of a system's design expressed in a formal language

- today's reading mostly was focused on formal specifications
- formal specifications are common in some safety-critical domains (e.g., aerospace, automotive software)
- to build one, you typically need to invest in learning a formal specification language (e.g., TLA+, which is the only one I've seen used in industry)

Formal specifications

Definition: a ***formal specification*** is a model of a system's design expressed in a formal language

- today's reading mostly was
- formal specifications are c
  (e.g., aerospace, automotiv
- to build one, you typically r
  specification language (e.g.
  used in industry)

ns

n

This class doesn't cover formal specifications in any detail, but you should be aware of their existence: writing a model of your system (in any spec language, or none at all) is a good way to catch **design errors**.

How to write specifications

Many ways:

- as English prose (e.g., the group think game's spec)
- as semi-structured English prose (e.g., user stories)
- as structured English (***formal specification***)
- as an **executable formal specification**

Executable formal specifications

It is sometimes possible to ***refine*** a formal specification into a program.

- such specifications are usually written in a special-purpose programming language ("***interactive proof assistant***")
- allows you to write proofs that directly apply to your executable code
- much, much more labor-intensive to develop than a standard software project
- area of active research!

Requirements and Specifications (Part 2)

Today's agenda:

- Reading Quiz
- Requirements elicitation
- Formal specifications\
- **A special, totally-unpredictable surprise**

Requirements and Specifications (Part 2)

Today's agenda:

- Reading Quiz
- Requirements elicitation
- Formal specifications\
- ~~**A special, totally-unpredictable surprise**~~ **GroupThink Specification Exercise, round 2**

Re-form your groups from Wednesday

You have (time left in class - 15) minutes discuss the spec again

After that, we'll play another round of the game, with a new rule:

- Answers that contradict the specification count for zero, even if you all answer together
- You'll only have 30 seconds to answer each question

If you were not in class on Wednesday, join whichever group is nearest to where you're sitting.

Takeaways: requirements and specifications

- Make sure you build the right thing (spend time gathering requirements)
- Specifications can help to:
  - increase understanding of system requirements between engineers and customer
  - document what the system does/will do
  - improve code quality
- Writing good specifications and getting everyone to understand them is hard and therefore worth spending time on