# Capstone Computing Project

# Requirements Specifications

# Requirements Specifications

- Why we need to write specifications
  - Purpose and audience
  - Choosing an appropriate size and formality
- Desiderata for Specifications
  - Properties of good specifications
  - Typical problems
  - What not to include
- Structure of a requirements document
  - IEEE standard

# Software Requirements Specification

- How do we communicate the Requirements to others?
  - It is common practice to capture them in an SRS
    - But an SRS doesn't need to be a single paper document...

- Purpose
  - Communication
    - explains the application domain and the system to be developed
  - Contractual
    - May be legally binding!
    - Expresses agreement and a commitment
  - Baseline for evaluating the software
    - supports testing, V&V
    - "enough information to verify whether delivered system meets requirements"
  - Baseline for change control

- Audience
  - Customers & Users
    - interested in system requirements…
    - …but not detailed software requirements
  - Systems (Requirements) Analysts
    - Write other specifications that inter-relate
  - Developers, Programmers
    - Have to implement the requirements
  - Testers
    - Have to check that the requirements have been met
  - Project Managers
    - Have to measure and control the project

# Appropriate Specification

- ➲ **Consider two different projects:**
  - A) Tiny project, 1 programmer, 2 months work
    - programmer talks to customer, then writes up a 2-page memo
  - B) Large project, 50 programmers, 2 years work
    - team of analysts model the requirements, then document them in a 500-page SRS

| | *Project A* | *Project B* |
|---|---|---|
| *Purpose of spec?* | Crystalizes programmer's understanding; feedback to customer | Build-to document; must contain enough detail for all the programmers |
| *Management view?* | Spec is irrelevant; have already allocated resources | Will use the spec to estimate resource needs and plan the development |
| *Readers?* | **Primary**: Spec author; **Secondary**: Customer | **Primary**: programmers, testers, managers; **Secondary**: customers |

# A complication: Procurement

- An 'SRS' may be written by…
  - …the procurer:
    - SRS is really a call for proposals
    - Must be general enough to yield a good selection of bids…
    - …and specific enough to exclude unreasonable bids
  - …the bidders:
    - SRS is a proposal to implement a system to meet the CfP
    - must be specific enough to demonstrate feasibility and technical competence
    - …and general enough to avoid over-commitment
  - …the selected developer:
    - reflects the developer's understanding of the customer's needs
    - forms the basis for evaluation of contractual performance
  - …or by an independent RE contractor!

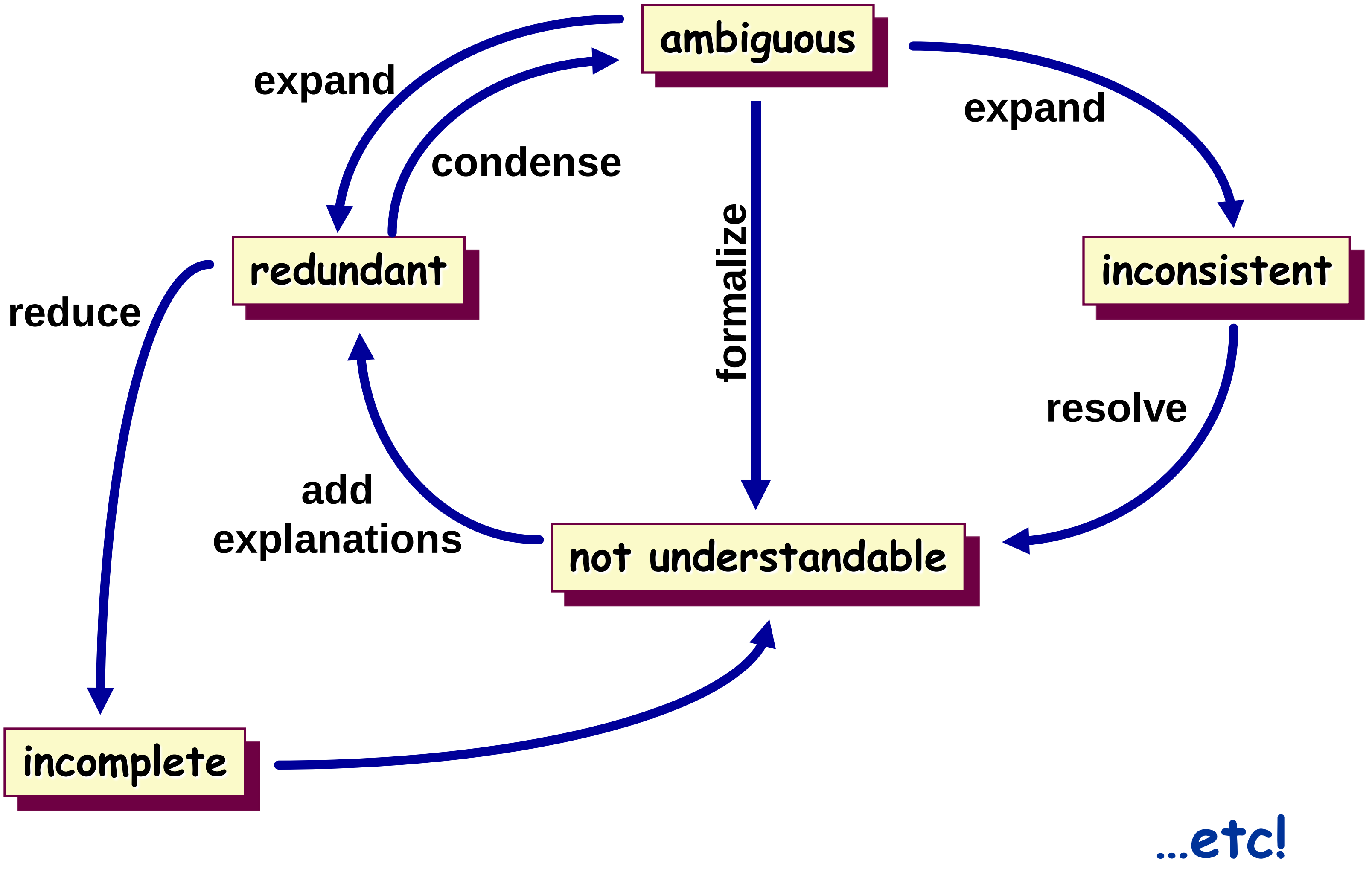
- Choice over what point to compete the contract
  - Early (conceptual stage)
    - can only evaluate bids on apparent competence & ability
  - Late (detailed specification stage)
    - more work for procurer; appropriate RE expertise may not be available in-house
  - IEEE Standard recommends SRS jointly developed by procurer & developer

# Desiderata for Specifications

***Source:*** *Adapted from IEEE-STD-830-1998*

- **Valid (or "correct")**
  - Expresses the real needs of the stakeholders (customers, users,…)
  - Does not contain anything that is *not* "required"
- **Unambiguous**
  - Every statement can be read in exactly one way
- **Complete**
  - All the things the system must do…
  - …and all the things it must not do!
  - Conceptual Completeness
    - E.g. responses to all classes of input
  - Structural Completeness
    - E.g. no TBDs!!!
- **Understandable (Clear)**
  - E.g. by non-computer specialists
- **Consistent**
  - Doesn't contradict itself
  - Uses all terms consistently
- **Ranked**
  - Indicates relative importance / stability of each requirement
- **Verifiable**
  - A process exists to test satisfaction of each requirement
- **Modifiable**
  - Can be changed without difficulty
    - Good structure and cross-referencing
- **Traceable**
  - Origin of each requirement is clear
  - Labels each requirement for future referencing

# There is no Perfect SRS!



# Use Appropriate Notations

## ➲ Natural Language?

- "The system shall report to the operator all faults that originate in critical functions or that occur during execution of a critical sequence and for which there is no fault recovery response."
  *(this is adapted from a real NASA spec for the international space station)*

## ➲ Or a decision table?

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Originate in critical functions? | **F** | **T** | **F** | **T** | **F** | **T** | **F** | **T** |
| Occur during critical sequence? | **F** | **F** | **T** | **T** | **F** | **F** | **T** | **T** |
| No fault recovery response? | **F** | **F** | **F** | **F** | **T** | **T** | **T** | **T** |
| **Report to operator?** | | | | | | | | |

# SRS Contents

- Software Requirements Specification should address:
  - Functionality.
    - What is the software supposed to do?
  - External interfaces.
    - How does the software interact with people, the system's hardware, other hardware, and other software?
    - What assumptions can be made about these external entities?
  - Performance.
    - What is the speed, availability, response time, recovery time of various software functions, and so on?
  - Attributes.
    - What are the portability, correctness, maintainability, security, and other considerations?
  - Design constraints imposed on an implementation.
    - Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) and so on?

# SRS should not include…

- Project development plans
  - E.g. cost, staffing, schedules, methods, tools, etc
    - Lifetime of SRS is until the software is made obsolete
    - Lifetime of development plans is much shorter
- Designs
  - Requirements and designs have different audiences
  - Analysis and design are different areas of expertise
    - i.e. requirements analysts shouldn't do design!
  - *Except where application domain constrains the design*
    - e.g. limited communication between different subsystems for security reasons.

# Typical mistakes

- **Noise**
  - text that carries no relevant information to any feature of the problem.
- **Silence**
  - a feature that is not covered by any text.
- **Over-specification**
  - text that describes a detailed design decision, rather than the problem.
- **Contradiction**
  - text that defines a single feature in a number of incompatible ways.
- **Ambiguity**
  - text that can be interpreted in at least two different ways.
- **Forward reference**
  - text that refers to a terms or features yet to be defined.
- **Wishful thinking**
  - text that defines a feature that cannot possibly be validated.
- **Requirements on users**
  - Cannot require users to do certain things, can only assume that they will
- **Jigsaw puzzles**
  - distributing key information across a document and then cross-referencing
- **Duckspeak requirements**
  - Requirements that are only there to conform to standards
- **Unnecessary invention of terminology**
  - E.g. 'user input presentation function'
- **Inconsistent terminology**
  - Inventing and then changing terminology
- **Putting the onus on the developers**
  - i.e. making the reader work hard to decipher the intent

# Organizing the Requirements

- Need a logical organization for the document
  - IEEE standard offers different templates
- Example Structures - organize by…
  - …External stimulus or external situation
    - e.g., for an aircraft landing system, each different type of landing situation: wind gusts, no fuel, short runway, etc
  - …System feature
    - e.g., for a telephone system: call forwarding, call blocking, conference call, etc
  - …System response
    - e.g., for a payroll system: generate pay-cheques, report costs, print tax info;
  - …External object
    - e.g. for a library information system, organize by book type
  - …User type
    - e.g. for a project support system: manager, technical staff, administrator, etc.
  - …Mode
    - e.g. for word processor: page layout mode, outline mode, text editing mode, etc
  - …Subsystem
    - e.g. for spacecraft: command&control, data handling, comms, instruments, etc.

# IEEE Standard for SRS

***Source:*** *Adapted from IEEE-STD-830-1998*

**1 Introduction**
- Purpose
- Scope
- Definitions, acronyms, abbreviations
- Reference documents
- Overview

**2 Overall Description**
- Product perspective
- Product functions
- User characteristics
- Constraints
- Assumptions and Dependencies

**3 Specific Requirements**

**Appendices**

**Index**

Identifies the product, & application domain

Describes contents and structure of the remainder of the SRS

Describes all external interfaces: system, user, hardware, software; also operations and site adaptation, and hardware constraints

Summary of major functions, e.g. use cases

Anything that will limit the developer's options (e.g. regulations, reliability, criticality, hardware limitations, parallelism, etc)

All the requirements go in here (i.e. this is the body of the document). IEEE STD provides 8 different templates for this section

# IEEE STD Section 3 (example)

***Source:*** *Adapted from IEEE-STD-830-1998*

## 3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.2 Hardware Interfaces

3.1.3 Software Interfaces

3.1.4 Communication Interfaces

## 3.2 Functional Requirements

*this section organized by mode, user class, feature, etc. For example:*

3.2.1 Mode 1

*3.2.1.1 Functional Requirement 1.1*

…

3.2.2 Mode 2

*3.2.1.1 Functional Requirement 1.1*

…

...

3.2.2 Mode n

...

## 3.3 Performance Requirements

*Remember to state this in measurable terms!*

## 3.4 Design Constraints

*3.4.1 Standards compliance*

*3.4.2 Hardware limitations*

*etc.*

## 3.5 Software System Attributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.5.5 Portability

## 3.6 Other Requirements

# Summary

- Requirements Specs have several purposes:
  - Communication
  - Contractual
  - Basis for Verification
  - Basis for Change Control
- Requirements Specs have several audiences:
  - Technical and non-technical
- Good Specs are hard to write
  - Complete, consistent, valid, unambiguous, verifiable, modifiable, traceable…
- Project needs vary
  - The amount of effort put into getting the spec right should depend on the possible consequences of requirements errors