Software Requirements

CSCE 247 - Lecture 2 - 01/16/2019

Today's Goals

- What are requirements?
- Understand the requirements problem
  - Why are requirements so important?
- Get a feel for the structure of a requirements document
  - What goes in there?
- Start to learn how to write "good" requirements
  - Clear and testable

Software Requirements

What is a requirement?

A **requirement** is a singular documented physical or functional need that a particular product must be able to perform.

"The software shall be able to calculate the sum of a column of integers."

A statement that identifies a necessary attribute, capability, characteristic, or quality of a system for it to have *value and utility to a stakeholder*.

Some Requirements...

From an infusion pump:

1. The application shall allow the Clinician to configure Patient information, Drug Information and Infusion parameters for basal, patient bolus and square bolus Infusion.
   - 1.1. The application shall allow the Clinician to enter new patient data only when the infusion pump is in Off or Paused State.
   - 1.2. The application shall perform all Patient and drug data validations after the Clinician enters the values.
   - 1.3. The application shall request clinician confirmation of the the patient and drug information before initiating infusion
   - 1.4. If Infusion is in progress during the configuration, the system shall continue infusion with existing parameters, until the new configuration is confirmed by the Clinician

Requirement Specification

A requirement's **specification** is a comprehensive technical description of how that requirement will be realized.

The set of specifications will fully describe what the software will do and how it will be expected to perform.

Specification Example

Requirement:

1. The user shall set a password to control account access.

Specification:

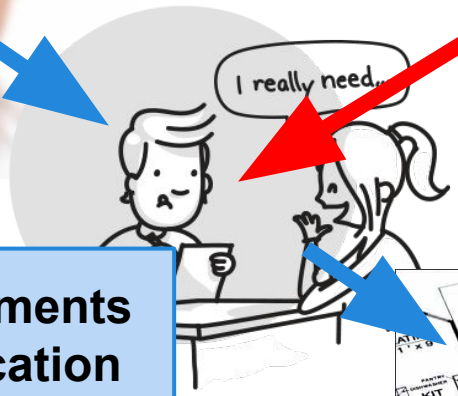
1.1. The password shall be at least eight and no more than sixteen symbols long.
1.2. The password shall contain at least one lower case and one upper case letter.
1.3. The password shall be hashed and stored in the password database (Req 3.4).

Specifying Requirements

- Requirements and their specifications are a description of what the system should do.
- They capture **what** the system is doing, not **how** it does it.
  - How = how the code is structured, what algorithms and data structures are used, etc.
  - (that is the design).
- Must be detailed enough to distinguish between the “right” and “wrong” system.

The Importance of Good Requirements

Concept Formation



Most of the problems sneak in here!

Requirements Specification

Implementation and Testing

Design

Importance of Requirements

“The single hardest part of building a software system is deciding precisely what to build… No other part of the work so cripples the resulting system if it is done wrong. No other part is more difficult to rectify later.”

- Fred Brooks

In other words:
“The spec was vague”

The Stakeholders

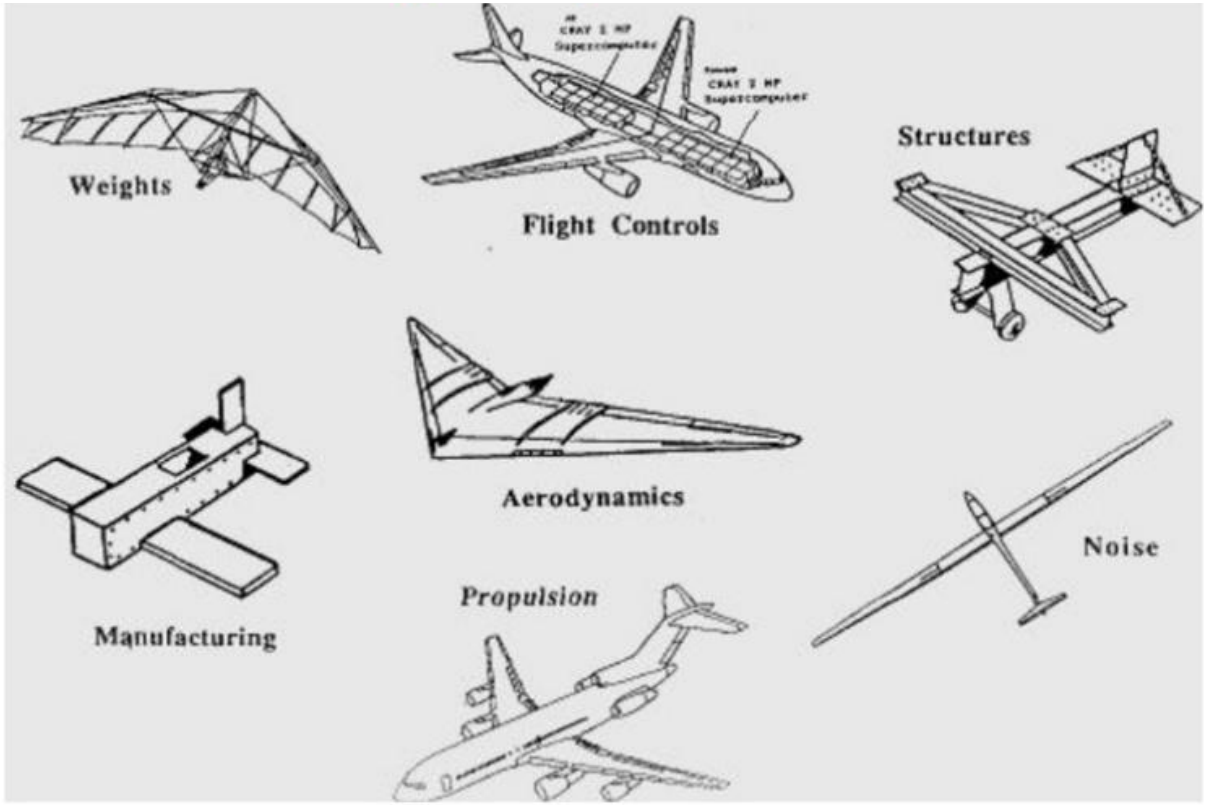
The specification process must involve all stakeholders:

- Clients
- Engineers
- Regulatory Agencies
- Users

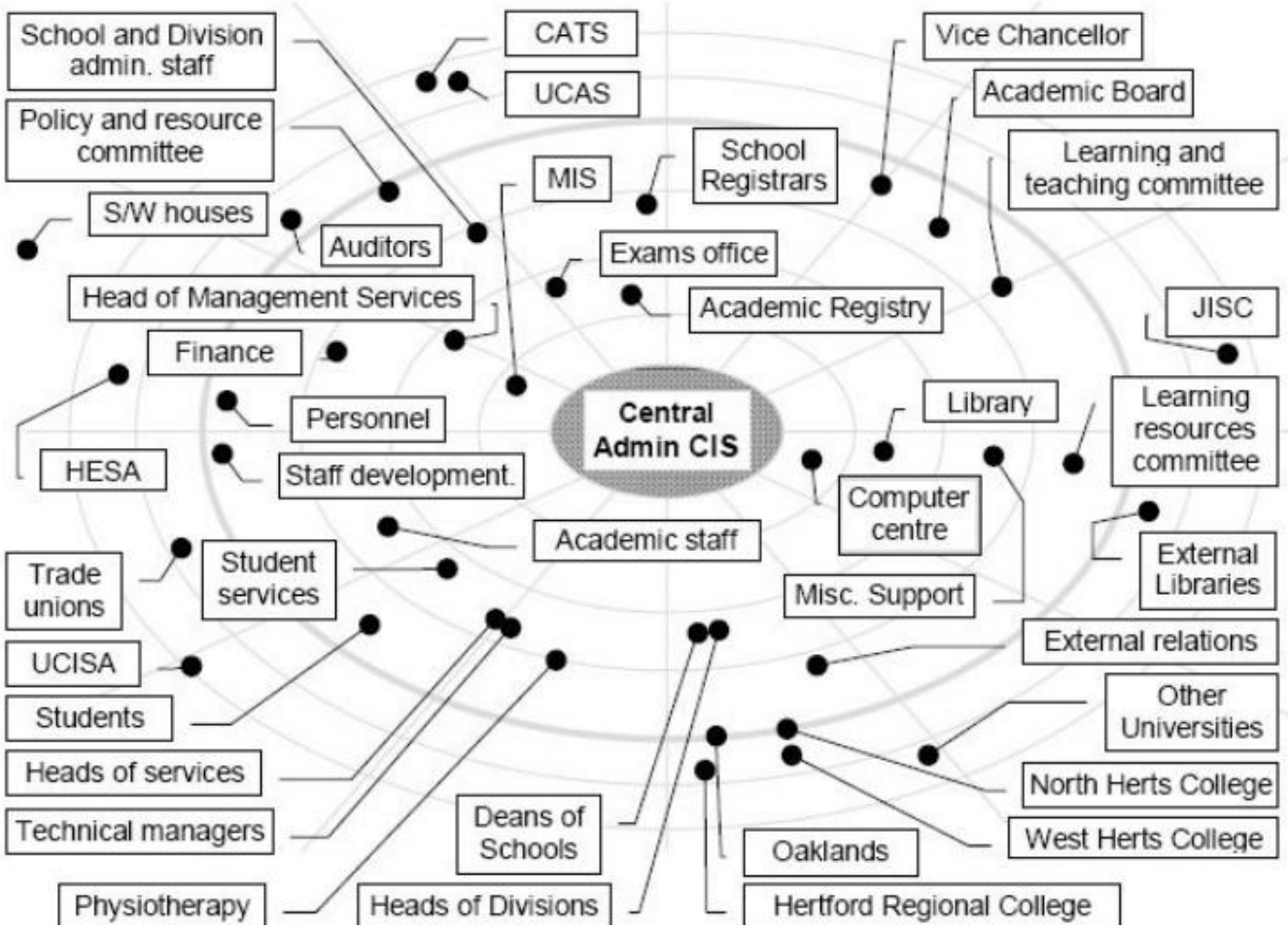
Do all of these want the same thing?

The Stakeholders

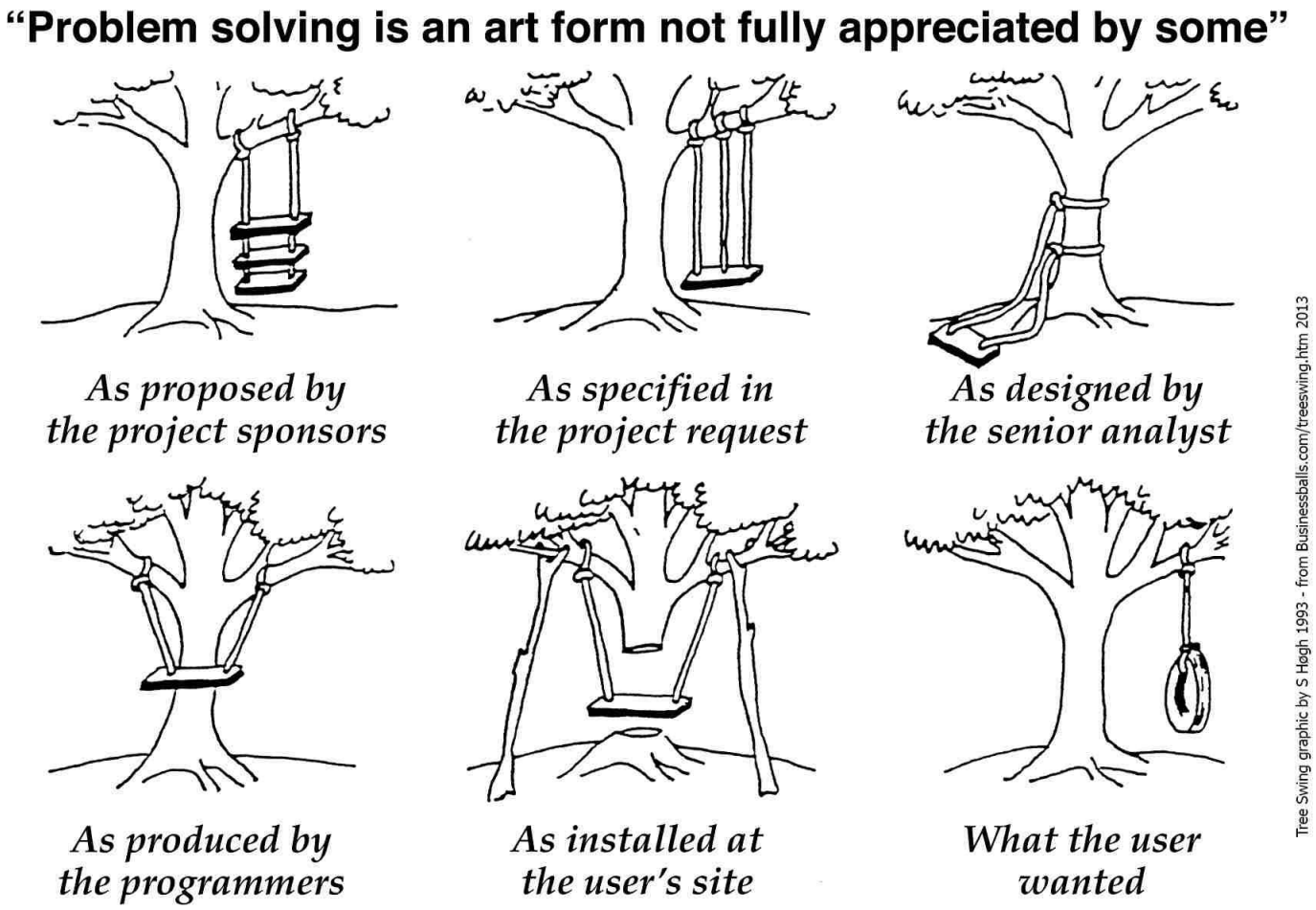
One requirement can have many meanings depending on the stakeholder's perspective.



There Can Be Many Stakeholders



Importance of Requirements



Importance of Requirements

The Engineering Argument:

- Engineering is about developing solutions to problems. A good solution can only be developed if *engineers understand the problem*.

The Economic Argument:

- Errors cost more to correct the longer they go undetected.

The Empirical Argument:

- Failure to understand and manage requirements is the biggest single cause of cost and schedule over-runs

Key Points

- Requirements capture what a proposed system should do, and the constraints they should follow.
- Poor requirements are the source of all evil.
- Requirements problems are the:
  - Most costly
  - Most difficult to correct

Types of Requirements

Types of Requirements

- **User Requirements**
  - Properties that the user wants to see out of the final system.
  - Typically reflect what the user wants the software to make possible in the real world.
    - “The user shall be able to book a concert ticket through the system.”
  - Usually high-level and lacking in technical detail.
  - Gathered from the user, and form the basis for technical requirements.

Types of Requirements

- **Hardware Requirements**
  - Define the properties expected from and constraints on the physical hardware in a system.
  - Expectations on storage space, CPU, network latency, sensor inaccuracy bounds, etc.
  - Example (from a pacemaker):
    - The Device Control Monitor shall either:
      - use an inductive telemetry wand to communicate with the pulse generator, maintaining consistent communication over the range of 0 cm to 5 cm between the wand and the pulse generator; or,
      - use some other medium, such as RF or ultrasound, that is safe and legal to use, for maintaining consistent telemetry with an implanted medical device.

Types of Requirements

- **Software Requirements**
  - Properties of and constraints on the software being executed on the system.
  - Bulk of requirements, typically.
  - Example (from a pacemaker):
    - If dynamic, the AV delay shall be determined individually for each new cardiac cycle based on the duration of previous cardiac cycles.
      - The previous cardiac cycle length is multiplied by a factor stored in device memory to create the dynamic AV delay.
      - The AV delay shall vary between a programmable maximum paced AV delay and a programmable minimum paced AV delay

Types of Requirements

- **System Requirements**
  - Define how software and hardware must work together to perform functionality.
  - Often related to interfaces, input, or output - how hardware can influence the software.
  - Example:
    - An automatic pacing threshold test shall be available on command of the DCM for both pulse width and amplitude measurements.
      - The user is instructed to terminate the test by removing the telemetry wand or selecting the ”Stop” button when loss of capture is observed.
      - The last six test results will be displayed on the screen and printed report.

Types of Requirements

Requirements can be **functional:**

- Related to the functionality of the system.
- Describe system services or actions.
- Can be used to distinguish correct output from incorrect.

or **non-functional**:

- Constraints on the system or the development process.
- Related to how functionality is performed, not whether the end result is correct.

Non-Functional Requirements

- Define system properties and constraints
  - Reliability, response time, storage requirements
  - Constraints on I/O device capability, system representations, etc.
- Process requirements may also be specified
  - Team organization, tool support, programming language, or development method.
- May be more critical than functional requirements.
  - If not met, the system is useless.

Non-Functional Requirement Types

- **Product Requirements**
  - Delivered product must behave in a certain manner: speed, reliability, disk space
- **Organizational Requirements**
  - Consequence of organizational policies and procedures: process, implementation requirements
- **External Requirements**
  - Arise from factors external to the product and development process: legislative requirements, communication protocols, interoperability standards.

Non-Functional Requirements

What are some examples of non-functional requirements?

- Product Requirements
  - Withdrawals must be available with no more than 30 minutes of maintenance time per day.
- Organizational Requirements
  - The system development process shall follow the SCRUM model.
- External Requirements
  - The system shall keep a log of transactions in the Microsoft Excel file format.

Writing Requirements

Writing Requirements

Normal Method:

Natural language statements, supplemented by diagrams and tables (like the examples so far).

This is universally understandable, but problems can arise.

Writing Requirements

What are some of the problems with natural language requirements?

- Lack of clarity
  - Precision is difficult without making the document hard to read, and it is hard to write unambiguously.
- Missing details
  - It is easy to forget to include details when using natural language.
- Requirements amalgamation
  - Several different requirements may be expressed together.

Level of Detail

The level of detail of requirements may range from a high-level abstract statement to a detailed mathematical functional specification.

This is inevitable: requirements serve dual functions:

- May be the basis for a bid for a contract
  - Therefore, open to interpretation.
- May be the basis for the contract itself
  - Therefore, must be defined in detail.

Capturing Good Requirements

Three common problems:

- Poorly written individual requirements
  - Ambiguous, contradictory, unclear.
- Untestable requirements
  - Cannot verify that the final system meets the requirements, as written.
- Poorly structured requirements documents
  - Unable to find necessary information.

Requirements Sample (NASA)

While acting as the bus controller, the C&C MDM CSCI shall set the e, c, w indicators identified in Table 3.2.16-2 for the corresponding RT to "failed" and set the failure status to failed for all RT's on the bus upon detection of transaction errors of selected messages to RTs whose 1553 FDIR is not inhibited in two consecutive processing frames within 100 milliseconds of detection of the second transaction error if: a backup BS is available, the BC has been switched in the last 20 seconds, the SPD card reset capability is inhibited, or the SPD card has been reset in the last 10 major (10 second) frames, and either

1. the transaction errors are from multiple RT's, the current channel has been reset within the last major frame, or
2. the transaction errors are from multiple RT's, the bus channel's reset capability is inhibited, and the current channel has not been reset within the last major frame.

Withdrawal Requirement

2.6: Withdrawal

If the card is accepted, the user has entered the correct PIN, and if there are sufficient funds in the account, the amount of cash shall be dispensed. If the card is invalid (in which case it should be ejected), the PIN does not match the one required for the card (in which case a tone shall sound and the user given the option to try again - the tries shall be limited to 3), or the balance is insufficient (in which case a tone shall sound and the user shall have the opportunity to enter a new amount) cash shall not be dispensed.

Withdrawal Requirement (Take 2)

2.6: The system shall support cash withdrawals by the user.

High-level requirement, defining a function of the system.

allowed if, and only if:

- The card can be validated (Req 2.7).
- The PIN is valid for the card (Req 2.8).
- The funds in the card account exceed the funds requested

Action to be performed if preconditions are met.

equest.

Conditions necessary to perform the action.

2.6.2: If a withdrawal is a
requested shall be dispensed.

Using Templates to Present Information

- Templates define a standard requirement structure.
- You should establish templates for the requirement descriptions.
  - Ensure readers are familiar with the document.
  - Acts as a checklist so that no sections are forgotten.
  - Makes it easy to find the needed information.

Suggested Template Items

- **Number:** A unique identifier.
- **Use Case:** Which use-case is this requirement linked to.
- **Introduction/Definition:** What is this requirement about?
- **Rationale:** Why does the requirement exist?
- **Source:** Who came up with the requirement?
- **Author:** Who wrote it down?
- **Inputs:** What are the necessary inputs to use this function?
- **Required Function:** What is the requirement?
- **Outputs:** What is output as a result of this function?
- **Related Requirements:** List of relevant requirements?
- **Conflicts:** Are there requirements in conflict with this one?
- **Support Material:** Docs, figures, tables, etc.
- **Test Cases:** How do we test this requirement?
- **Date:** When was this requirement modified last?
- **Priority:** How important is this requirement?

Withdrawal, Using Template (Still Bad)

Introduction: The most common action performed at an ATM is the withdrawal of funds. The ATM must support this functionality.

Rationale: Survey ABC-345 indicates that withdrawals are highly desirable. The success of the product hinges on successful withdrawal of funds.

Inputs: Card number, PIN, requested amount

Description: If the card is accepted, the user has entered the correct PIN, and there are sufficient funds in the account, the amount of cash shall be dispensed. If the card is invalid (in which case it should be ejected), the PIN does not match the one required for the card (in which case a tone shall sound and the user given the option to try again—the tries shall be limited to 3), or the balance is insufficient (in which case a tone shall sound and the user shall have the opportunity to enter a new amount) cash shall not be dispensed.

Outputs: Customer Receipt, Requested Amount of Money, Alarm

Persistent Changes: The Requested Amount will be deducted from the Customer Account.

Withdrawal, Using Template (Better)

Introduction: The most common action performed at an ATM is the withdrawal of funds. The ATM must support this functionality.

Rationale: Survey ABC-345 indicates that withdrawals are highly desirable. The success of the product hinges on successful withdrawal of funds.

Inputs: Card number, PIN, requested amount

Description: The user shall be able to withdraw funds from the ATM

- A withdrawal shall be allowed if and only if:
  - The Card Number can be validated (Req 35)
  - The PIN is valid for the card (Req 45)
  - The funds in the card account exceeds the Requested Amount requested in the withdrawal
- If a withdrawal is allowed, the Requested Amount of money shall be dispensed.

Outputs: Customer Receipt, Requested Amount of Money

Persistent Changes: The Requested Amount will be deducted from the Customer Account.

Related Requirements: 35, 45

The Requirements Specification Document

The Requirements Document

- The official statement of what work is required from the system developers.
- Should include both requirement definitions and specifications.
- NOT a design document - as much as possible, this defines what the system will do and not how it will be coded.
- Should also include tests that can show that the requirement is fulfilled.

The Requirements Document

What does the document need to 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?
- Performance
  - What is the speed, availability, response time, recovery time, etc of the software functions?

SRS Must Address...

- Quality Attributes
  - Things we desire out of the system.
    - It must be portable to other platforms.
    - It must be easy to maintain.
  - What are the considerations for portability, maintainability, security, etc?
  - How is quality measured for each attribute.
- Design/Implementation Constraints
  - Are there any required standards in effect, implementation language, policies for data management, resource limits, operating environment, etc?

SRS Should NOT Include

What shouldn't go into the SRS?

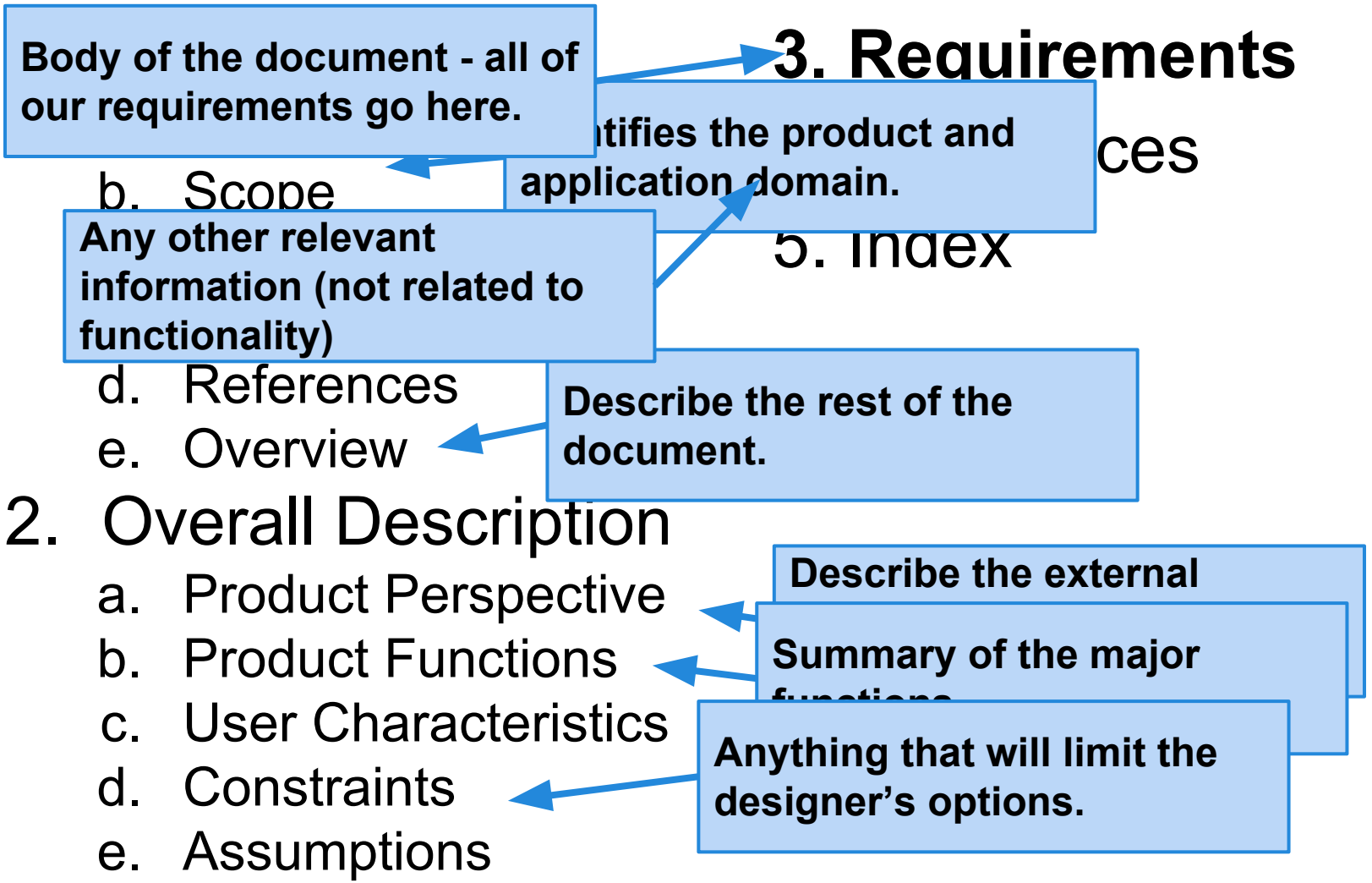
- Project Planning (cost, staffing, schedules, etc.)
  - Unrelated to the functionality of the system.
  - Requirements document should rarely change after completion - project planning continually changes.
- Product Assurance Plans
  - How you will assess quality (test plans, QA process, V&V, CMM)
  - Important considerations, but different audiences, different timelines.

SRS Should NOT Include

What shouldn't go into the SRS?

- Design Details
  - Requirements should be relatively stable before design commences.
  - Requirements and design have different audiences.
  - Analysis and design are different areas of expertise.
  - **Exception:** Unless there are constraints imposed on the design by the domain.

IEEE Document Structure



“Requirements”

How do we organize the “Requirements” section?

- Should be customized to fit the project.
- IEEE suggests three options.
- Separate sections for software, hardware, system requirements.
- Entire document (especially this section) needs to be structured to provide:
  - Understandability
  - Changeability
  - Other “ability” attributes

Requirements Outline

- **External Interface Requirements**
  - **User Interfaces**
  - **Hardware Interfaces**
  - **Software Interfaces**
  - **Communication Interfaces**
- Function
- Performa
- Design C
- Other Re

Constraints and definitions outlining how your system interacts with other systems and users.

“Specific Requirements” Outline

- External Interface Requirements
- **Functional Requirements**
  - **Three Example Structures**
- Perfo
- Desig
- Othe

Option 3: Organized by System Mode

- **Mode 1**
  - **Requirement 1.1**
  - **Requirement 1.2**
  - **…**
  - **Requirement 1.m**
- **Mode 2**
- **….**
- **Mode N**

Key Points

- The structure of the requirements document is of critical importance.
  - Tune to the audience.
  - Use a template to help organize.
    - Customize it to fit your needs
- Individual requirements contain more information than you may think.
  - Use templates to structure and clarify.
  - But, requirement must still be well-written.
    - Precise, avoid amalgamation, make distinction between functional/non-functional

Next Time

- Next week:
  - More on writing good, testable requirements.
  - Reading: Sommerville, chapter 4

- Team assignments due Monday night.
  - E-mail if you have a team, or if you want assigned to a team.