Software Engineering Theory and Practice

School of Computing	
Title	Software Engineering Theory and Practice
Module Coordinator	Steven Ossont
Email	steven.ossont@port.ac.uk
Code	M30819
Moodle	https://moodle.port.ac.uk/course/view.php?id=11429

U30819: Software Engineering Theory and Practice

Topic: Software Requirements Specification (Requirements Engineering)

https://moodle.port.ac.uk/course/view.php?id=11429

Steven Ossont steven.ossont@port.ac.uk

Based on Ian Somerville, Software Engineering, 10th edition

Software Requirements Specification

Software Requirements Specification document must address/include:

- What is the purpose of your software/product?
- What are **you** building?
- Detailed requirements
- Customer/Stakeholder approval

> It is **not** a design document. As far as possible, it should be document of **what** the system should do rather than **how** the system should it should do it

Challenges in Requirements Gathering

- Scope
  - Boundaries of the system are ill-defined
  - Unnecessary information is given
- Volatility
  - Requirement evolve over time

Challenges in Requirements Gathering

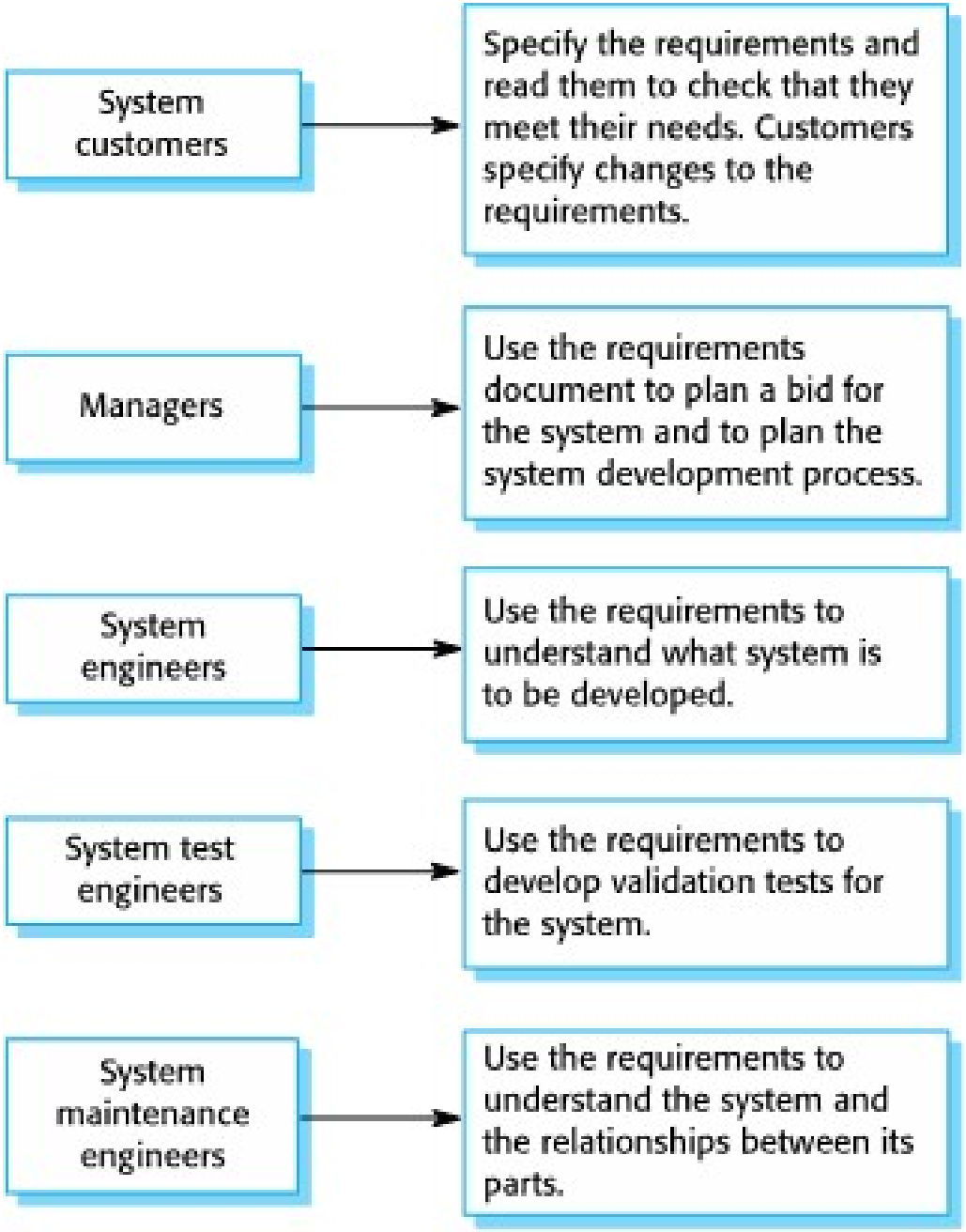
- Understanding
  - Users have incomplete understanding of their needs
  - Users overestimate the available technology and its capabilities
  - Analysts have poor knowledge of problem domain

Software Requirements Specification

> The process of writing down the user requirements and translating them into system requirements to produce a **Software requirements specification (SRS)**

- User requirements have to be understandable by end-users and customers who do not have a technical background.
- System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development

Users


System customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers
Use the requirements to understand what system is to be developed.
System test engineers
Use the requirements to develop validation tests for the system.
System maintenance engineers
Use the requirements to understand the system and the relationships between its parts.

The Software requirements specification

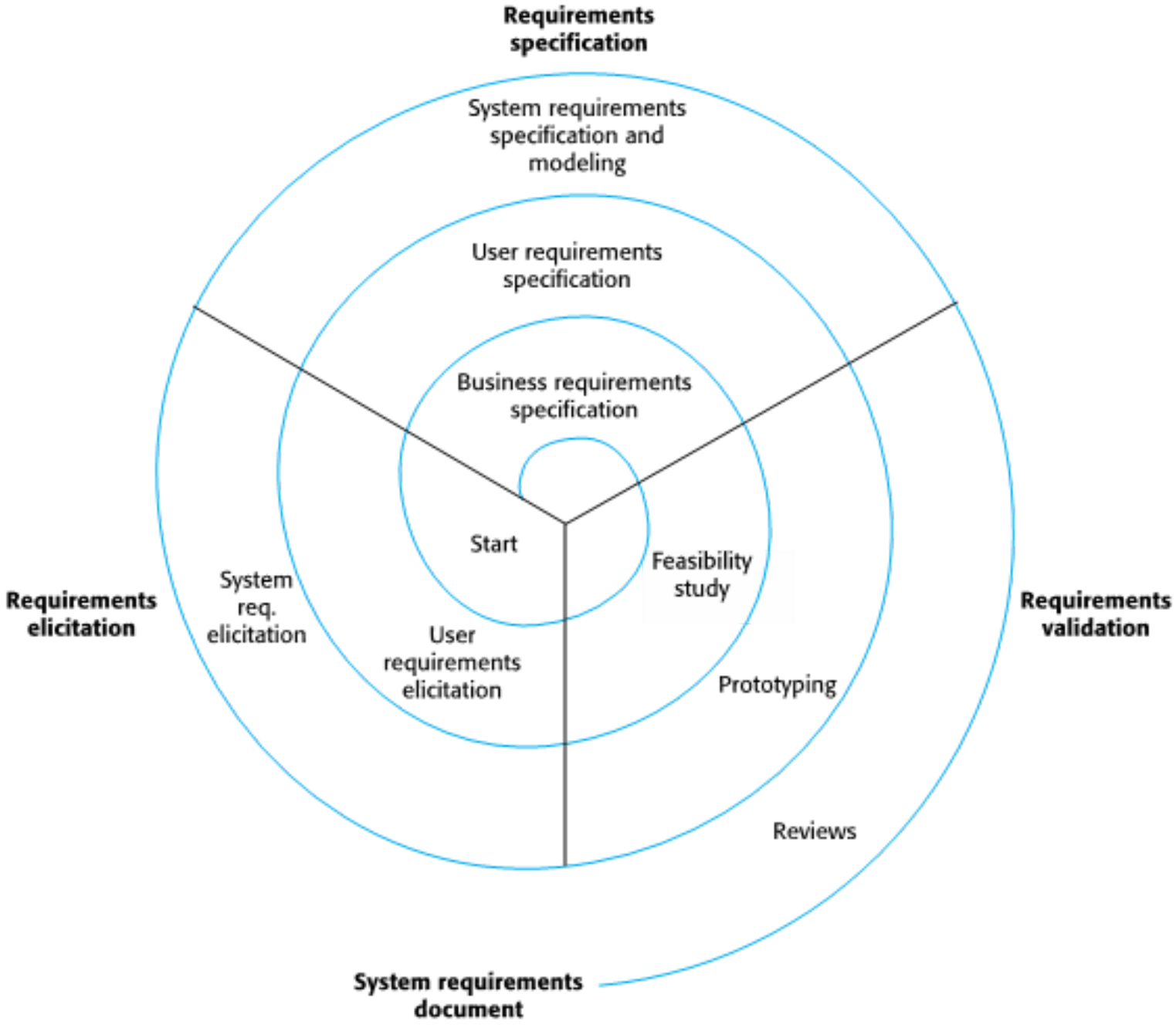
- Break down the problem
- Inform design decisions
- Prioritise requirements
- Identify validation strategies
- Support change
- Avoid conflict

Ideal Software Requirements Specification

The report should be:

- Clear
- Unambiguous
- Easy to understand
- Complete
- Consistent

Iterate..


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Guidelines for Writing Requirements

- Use a template for all requirements
- Use consistent language
- Use text highlighting/color to identify key parts of each requirement
- Avoid using jargon, abbreviations, or acronyms or make sure you explain all these in a glossary
- Include a rationale for each requirement – why has this been included?
- Include the source of each requirement

Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.

Example: Natural Language Specification

- 3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. (Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)
- 3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. (A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)

(Chapter 4 Requirements Engineering)

Problems of Natural Language Specification

- Lack of clarity
- Ambiguity
- Context dependency
  - Same words in different sentences have different meanings
  - Depends on the whole paragraph/page

Ways of writing a system requirements specification

Notation	Description
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.

Structured natural language Specification

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Structured natural language Specification

- Description of the function specified
- Description of inputs and their source
- Description of outputs and their source
- Information needed for running the function
- Description of action to be taken
- Description of consequences or side-effects of the function

Structured natural language

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

Structured natural language

Action
CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements
Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition
The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition r0 is replaced by r1 then r1 is replaced by r2.
Side effects None.

Ways of writing a system requirements specification

Notation	Description
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.

Hardware Description Language (HDL)

VHDL

```
library IEEE;
use IEEE.STD_Logic_1164, all;

entity LATCH_IF_ELSEIF is
  port (En1, En2, En3, A1, A2, A3: in std_logic;
        Y: out std_logic);
end entity LATCH_IF_ELSEIF;

architecture RTL of LATCH_IF_ELSEIF is
begin
   process (En1, En2, En3, A1, A2, A3)
   begin
      if (En1 = '1') then
        Y <= A1;
      elseif (En2 = '1') then
        Y <= A2;
      elseif (En3 = '1') then
        Y <= A3;
      end if;
   end process;

end architecture RTL;
```

Verilog

```
module LATCH_IF_ELSEIF  (En1, En2, En3, A1, A2, A3,Y);
    input En1, En2, En3, A1, A2, A3;
    output Y;

    reg Y;

    always @(En1 or En2 or En3 or A1 or A2 or A3)
       if (En1 == 1)
          Y =A1;
     else if (En2 == 1)
          Y = A2;
     else if (En3 == 1)
          Y = A3;

end module
```

Ways of writing a system requirements specification

Notation	Description
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML use case and sequence diagrams are commonly used.

Graphical Specification

- Entity-Relationship Diagrams (ERD)
- Data Flow Diagrams
- State Transition Diagrams
- Decision Tables
- Decision Trees

Diagram Specification: Entity-Relationship Diagrams

- Techniques that identifies a system’s entities and the relationship between those entities
  - Entities: people, places, items, events, concepts
  - Attributes: properties or descriptive qualities of an entity
  - Relationships: links between different entities

Diagram Specification: Decision Table

Technique for specifying complex if-then conditions

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 − r1)<(r1 − r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing r2 > r1 & ((r2 − r1) ≥ (r1 − r0))	CompDose = round ((r2 − r1)/4) If rounded result = 0 then CompDose = MinimumDose

Ways of writing a system requirements specification

Notation	Description
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don’t understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract

Mathematical Methods

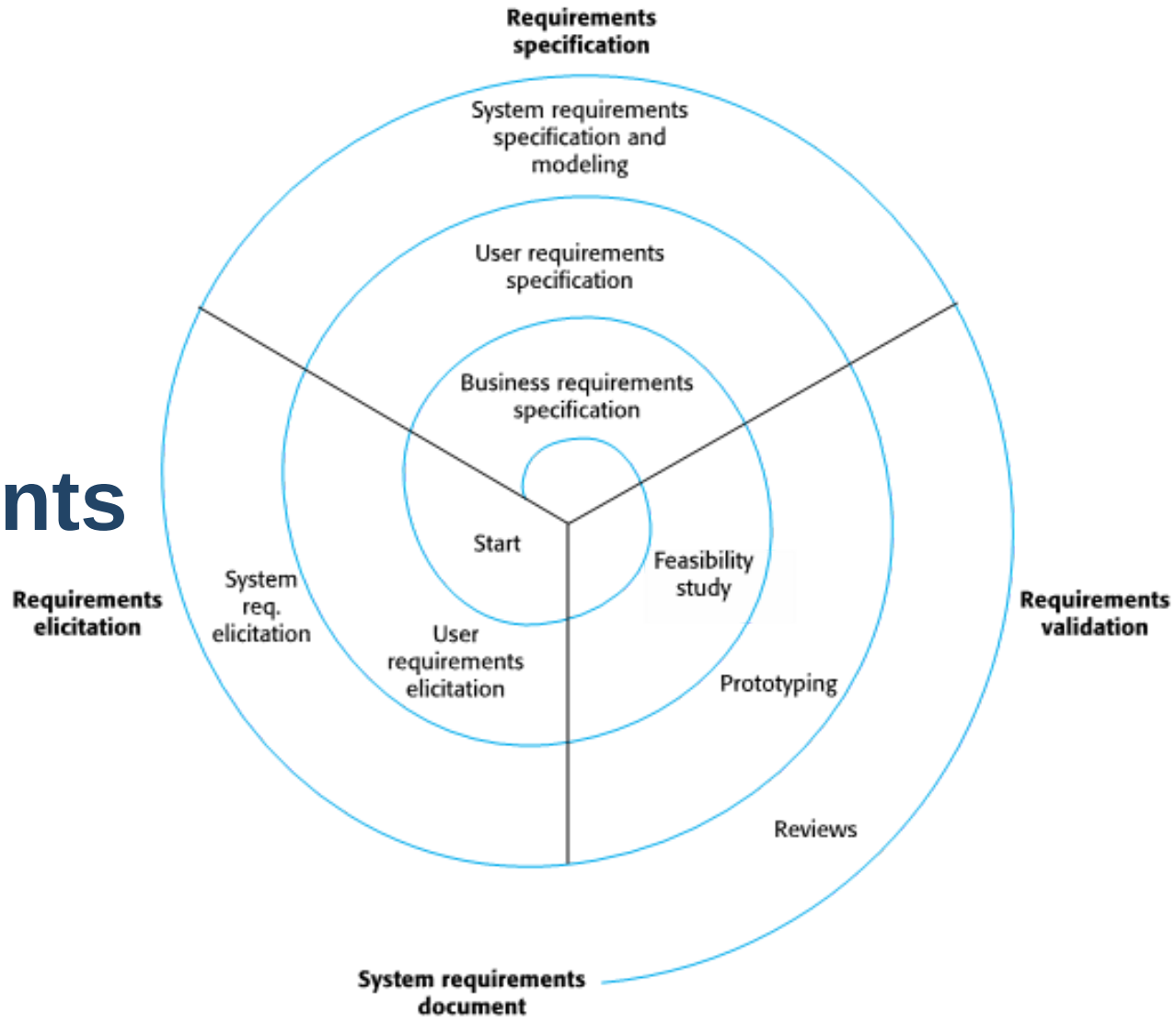
- Recurrence relations
- Axiomatic definition
- Formal specifications
- Implicit equations
- Regular expressions

Requirements vs Design

In principle, requirements should state what the system should do and the design should describe how it does this. **In practice, requirements and design are inseparable**

- A system architecture may be designed to structure the requirements
- The system may inter-operate with other systems that generate design requirements;
- The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
    - This may be the consequence of a regulatory requirement.

Requirements



Requirements Validation

> *checking that requirements actually define the system that the customer really wants* (Sommerville)

Requirements Validation

- The cost of fixing a requirement problem once the system in implemented is much higher than fixing design or coding errors
- Changing requirements triggers changes in design, coding, and a re-test of the entire project

Requirements Validation Checks

- Validity
  - Does the system support the customers needs?
- Consistency checks
  - requirements should not conflict
- Completeness checks
  - all functions and constraints need to be defined
- Realism checks
  - make sure the requirements can be implemented (Budget and technology)
- Verifiability
  - requirements should always be written so that they are **verifiable/testable**

Requirements Validation Techniques

- Requirements reviews
  - systematic analysis of requirements by a team of reviewers
- Prototyping
  - an executable model of the system is demonstrated to users/customers
- Test-case generation
  - derive tests for each of the requirements

Requirements reviews

Must be regular and involve both the Customer and developer. Reviews may be formal (with completed documents) or informal. Checks include:

- Verifiability
  - Is the requirement realistically testable?
- Comprehensibility
  - Is the requirement properly understood?
- Traceability
  - Is the origin of the requirement clearly stated?
- Adaptability
  - Can the requirement be changed without a large impact on other requirements?

Requirements Management

- Requirements change and evolve throughout the life-cycle of a project:
  - The technical/business environment of the system may change after installation
  - The customers are not always the users
  - For large systems with a diverse user community, the Software requirements specification is a compromise

Requirements Management

- The process of *understanding and controlling changes to system requirements*
  - Keep track of individual requirements
  - Maintain links between dependent requirements
  - Formal process for making changes and linking the changes to existing requirements
  - Manage requirements early in the process!

Questions?