Software Requirement Engineering

Gerson Sunyé
University of Nantes
gerson.sunye@univ-nantes.fr
http://sunye.free.fr

Agenda

- Introduction
- Activities
    - Elicitation
    - Analysis
    - Specification
    - Validation
    - Management
- Specification example

Introduction

Software Requirements

- Once the domain analysis is done, it is time to specify the system boundaries.

- Often, of the domain model is reused by the requirements: clients want to keep their business "as is".

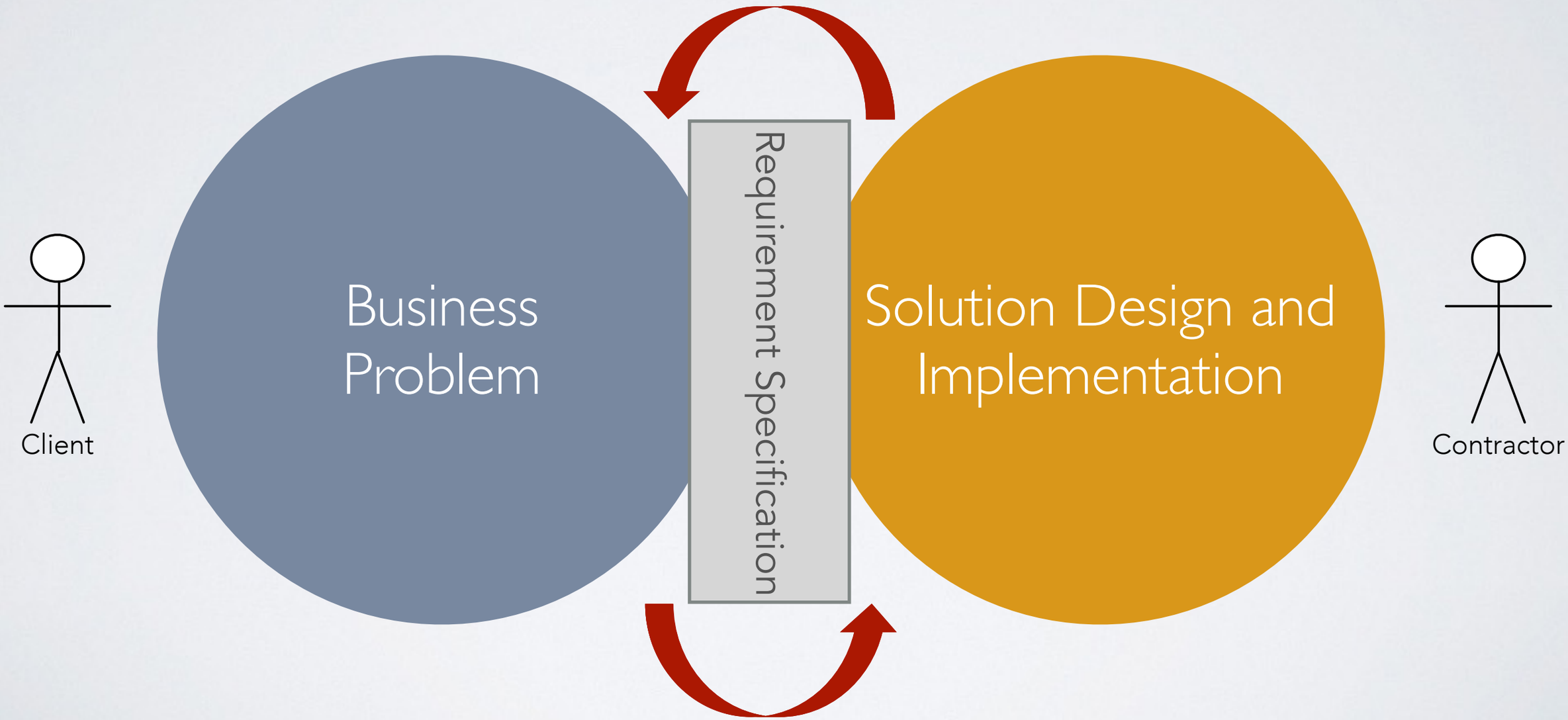
- In other few cases, clients want to evolve their business. In these case, requirements specify the system as the client want it "to be".

Requirement Engineering

"A requirement is the capacity and the conditions to which the system (and more broadly, the project) must comply."

"Requirement engineering is the rigorous application of scientific principles and techniques for requirement development, communication, and management."

The Boundary of two Worlds



Building the Right System Right

"Building the right system"

"Building the system right"

A Difficult Activity

The first difficulty of specifying requirements is to find, to communicate, and to remember what is really necessary and to specify it in a clear and understandable manner for clients and contractors.

Project Success Factors

1. User involvement.
2. Executive management support.
3. Clear statement of requirements.
4. Proper planning.
5. Realistic expectations
6. Smaller project milestones.
7. Competent staff
8. Ownership.
9. Clear vision and objectives.
10. Hard-working and focused staff.

[Standish Group's Chaos Report (1995)]

Project Challenged Factors

1. Lack of user input.
2. Incomplete requirements and specifications.
3. Changing requirements and specifications.
4. Lack of executive support.
5. Technology incompetence.
6. Lack of resources.
7. Unrealistic expectations.
8. Unclear objectives.
9. Unrealistic time frames.
10. New technology.

[Standish Group's Chaos Report (1995)]

Why is it so Hard?

- Cobb's Paradox:
  "*We know why projects fail, we know how to prevent their failure--so why do they still fail*?"
  (Martin Cobb, secretary of the treasury board of Canada, 1996)

- Requirement engineering bridges the gap between:

  - Real-world problem: complex, messy, unstable, subjective, political, etc.

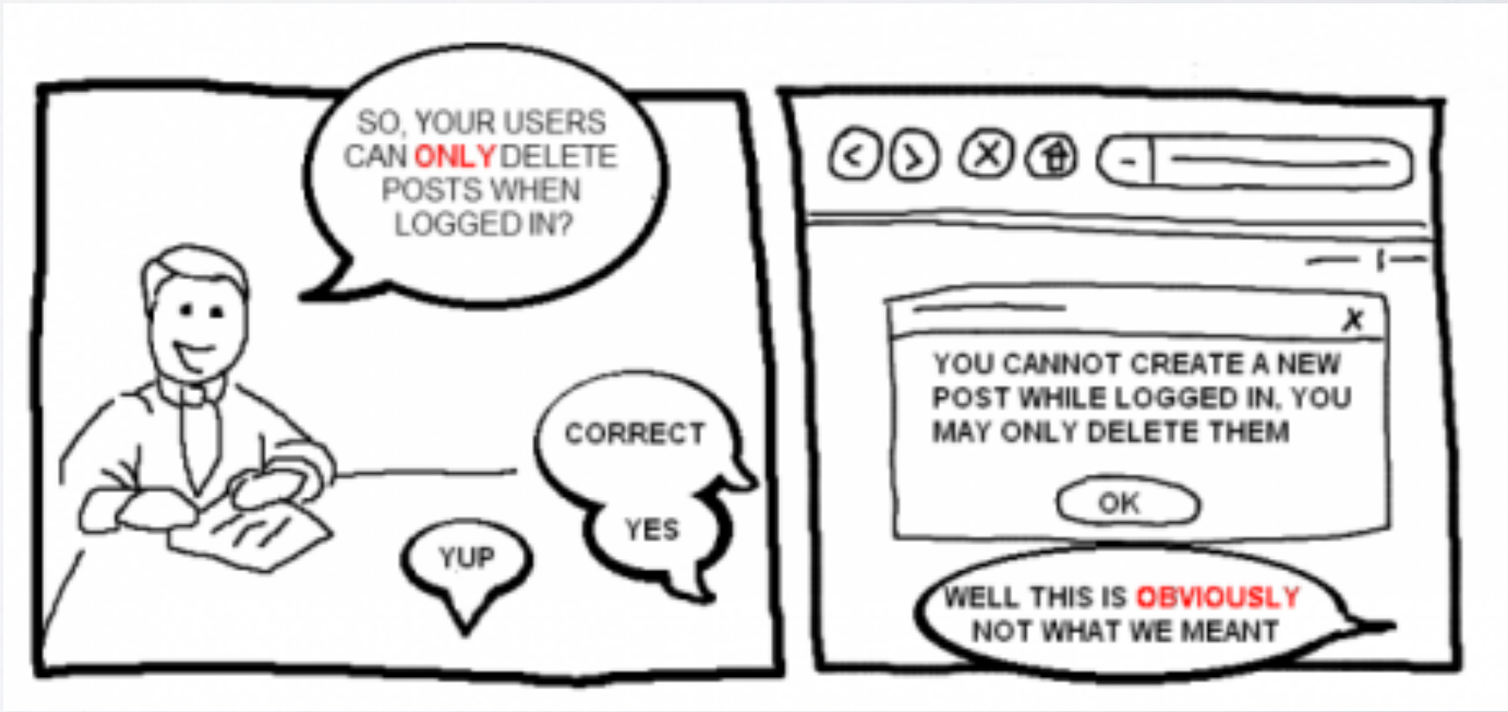
  - Computer programs: the opposite.

Why is it so Hard?

- Inherent difficulties:
  - Numerous and distributed stakeholders.
  - Vary and conflicting goals.
  - Goals may not be explicit.
  - Difficult to articulate requirements.
  - Particular, inconsistent terminology.

A Human Activity

- The context of Requirement Engineering is a human activity system and the clients are people.
- Requirement Engineering uses techniques for eliciting and modeling, drawn on cognitive and social sciences:
  - Cognitive Psychology.
  - Anthropology.
  - Sociology.
  - Linguistics.



Requirement Engineering Goals

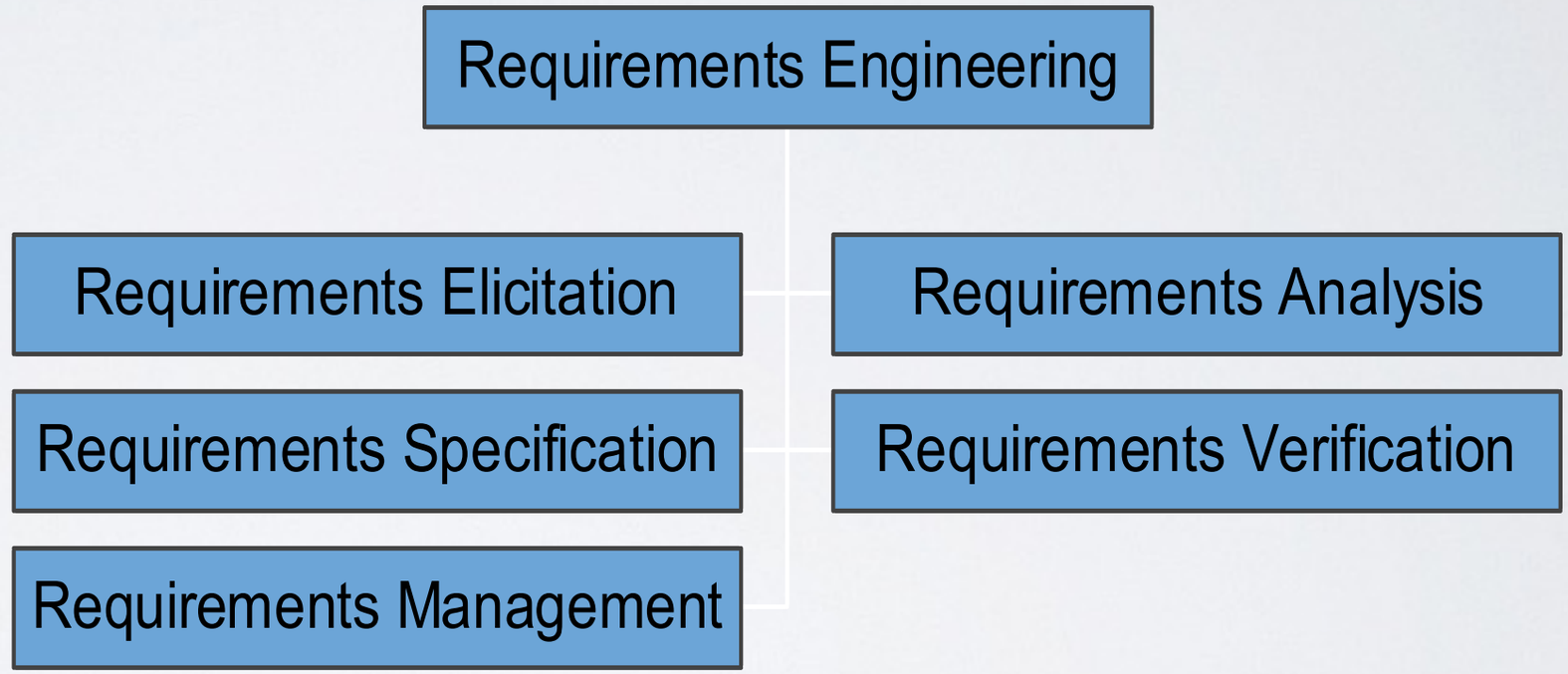
- Identify stakeholders and their needs.
- Create documents/models/databases for:
- Analyze, manage, and affect priorities.
- Communicate and validate with stakeholders.
- Define what the contractor must build.
- Verify that it was done correctly.
- Cope with future changes.

Typical Deliverables

- A Software Requirement Specification document containing:
  - A domain model (classes, data dictionary, etc.).
  - A set of functional requirements (use case descriptions).
  - A set of non-functional requirements: performance, etc.
  - Known platform constraints: hardware, operating system, etc.
  - Project and planning constraints: budget, deadlines, personal, etc.

Requirement Engineering Activities

- Elicitation
- Analysis
- Specification
- Verification
- Management



Requirement Elicitation

Discovering Requirements

- Requirements elicitation is the process of discovering the requirements for a system by communicating with customers, system users and others who have a stake in the system development.

Requirement Elicitation Goals

- Identify relevant sources of requirements (stakeholders).
- Determine what information is needed.
- Analyze the gathered information, and look for implications, inconsistencies, or unresolved issues
- Confront the contractor the understanding of the requirements with the source.
- Synthesize appropriate statements of the requirements.

Requirements to Elicit

- Boundaries
  - Identify the high level boundaries of the system.
  - Stakeholders and Use Cases depend on boundaries.
- Stakeholders
  - Customer or Clients.
  - Developers.
  - Users (current and future).
- Goals
  - Eliciting High level goals early in development.
- Tasks
  - When it is difficult to articulate user requirements

Elicitation Techniques (1/2)

- Traditional
  - Introspection
  - Existing documents analysis.
  - Data analysis.
  - Interviews
    - Open-ended
    - Structured
  - Surveys / Questionnaires
  - Meetings
- Group elicitation
  - Brainstorming
  - RAD (Rapid Application Development)
  - JAD (Joint Application Design)
- Prototyping
  - Throwaway (close-ended).
  - Evolutionary (breadboard).
  - Incremental.
  - Extreme.

Elicitation Techniques (2/2)

- Model-Driven
  - Goal-based methods – KAOS & I
  - Scenario-based methods - CREWS
- Cognitive
  - Task analysis
  - Protocol analysis
  - Laddering
  - Card sorting
  - Repertory grids
- Contextual
  - Alternative to traditional and cognitive techniques
  - Ethnographic technique – Participant observation

Modeling and Analyzing Requirements

Creating Descriptions for Interpretation and Validation

- Requirement modeling is the process of creating abstract descriptions that are amenable to interpretation and validation.
    - Enterprise organization.
    - Information (data).
    - Functional behavior.
    - Business domain.
    - Non-functional properties

Modeling Requirements

- Enterprise Modeling
  - Understanding client organization's structure
  - Business rules
  - Goals, tasks and responsibilities
  - Data
- Data Modeling
  - Understand, manipulate, and manage large volume of information.
  - How to represent real world phenomenon into system information

Modeling Requirements

- Behavioral Modeling
    - Dynamic or functional behavior of stakeholders and system, both existing (as is) and required (to be).
- Domain Modeling
    - Abstract description of the environment in which the system will operate
    - Requirements reuse within a domain.
- Modeling non-functional properties
    - Difficult to express in a measurable way.
    - Difficult to analyze.
    - Properties of a system as a whole.

Analyzing Requirement Models

- Requirements Animation
- Automated Reasoning
- Analogical and Case-based Reasoning
- Knowledge based Critiquing
- Consistency Checking

Different Languages

- Natural language
  - No special training required
  - Ambiguous, verbose, vague, obscure ...
  - No automation
- Ad hoc notation (bubbles, squares and arrows)
  - No special training required
  - No syntax formally defined meaning not clear, ambiguous
  - No automation
- Semi-formal notation (URN, UML, SysML...)
  - Syntax (graphics) well defined
  - Partial common understanding, reasonably easy to learn
  - Partial automation
  - Meaning only defined informally
  - Still a risk of ambiguities
- Formal notation (Logic, SDL, Petri nets, FSM ...)
  - Syntax & semantics defined
  - Great automation (analysis and transformations)
  - More difficult to learn & understand

Requirement Specification

Writing Clear Requirements

- Requirement specification describes clearly and accurately each of the essential requirements (functions, performance, design constraints, and quality attributes) of the system and its external interfaces.

Requirements are Complete Sentences

- Each requirement must first form a complete sentence
  - Not a bullet list of buzzwords, list of acronyms, or sound bites on a slide
- Each requirement contains a subject and predicate
  - Subject: a user type (watch out!) or the system under discussion
  - Predicate: a condition, action, or intended result
    - Verb in predicate: "shall" / "will" / "must" to show mandatory nature; "may" / "should" to show optionality
      - MUST, REQUIRED or SHALL: mean that the definition is an absolute requirement of the spec.
      - MUST NOT or SHALL NOT: absolute prohibition
      - SHOULD or RECOMMENDED: think twice about not doing it!
      - SHOULD NOT or NOT RECOMMENDED: think twice about doing it!
      - MAY or OPTIONAL: truly optional
- The whole requirement provides the specifics of a desired end goal or result
- Contains a success criterion or other measurable indication of the quality

Requirements are Clear, Simple, and Unambiguous

- Look for the following characteristics in each requirement (Amyot)
  - Feasible (not wishful thinking)
  - Needed (provides the specifics of a desired end goal or result)
  - Testable (contains a success criterion/ other measurable indication of quality)
  - Clear, unambiguous, precise, one thought
  - Prioritized
  - ID
- Another set of criteria (Pohl, Lucas-Hirz)
  - Complete: No missing information
  - Atomic: Express one and only one requirement
  - Traceable: Source, evolution, impact, effective use
  - Correct: needed
  - Unambiguous: exactly one meaning
  - Consistent: consistent with respect to the terminology
  - Verifiable: testable
  - Up to date: reflect the actual status

Tips for Writing

- Write short sentences. Use the active voice (no more than 22 words).
- To know whether your requirement is explicit enough, read it from the developer and the tester points of view.
- Write requirements that can be unitary tested. Avoid "and/or" statements that compose multiple requirements.
- Keep a consistent and homogeneous level of details.
- Avoid redundancy. May ease the reading but not maintenance. May lead to inconsistencies.

[Wiegers 2006]

Writing Pitfalls to Avoid

- Never describe how the system is going to achieve something (over-specification), always describe what the system is supposed to do
    - Refrain from designing the system
        - Danger signs: using names of components, materials, software objects, fields & records in the user or system requirements
- Designing the system too early may possibly increase system costs
- Do no mix different kinds of requirements (e.g., requirements for users, system, and how the system should be designed, tested, or installed)
- Do not mix different requirements levels (e.g., the system and subsystems)
    - Danger signs: high level requirements mixed in with database design, software terms, or very technical terms

"What" is better than "HOW"

- Bad:
  "The system shall use Microsoft Outlook to send an email to the customer with the purchase confirmation.

- Good:
  "The system shall inform the customer that the purchase is confirmed."

Avoid Ambiguities

- Never build in let-out or escape clauses
  - Requirements with let-outs or escapes are dangerous because of problems that arise in testing
    - Danger signs: if, but, when, except, unless, although
      - These terms may however be useful when the description of a general case with exceptions is much clearer and complete that an enumeration of specific cases
- Avoid ambiguity
  - Write as clearly and explicitly as possible
  - Ambiguities can be caused by:
    - The word or to create a compound requirement
    - Poor definitions (giving only examples or special cases)
    - The words etc., …and so on (imprecise definition)
  - Do not use vague indefinable terms
    - Many words used informally to indicate quality are too vague to be verified
    - Danger signs: user-friendly, flexible, approximately, easy, as much as possible

Keep it Simple

- Do not make multiple requirements
  - Keep each requirement as a single sentence
  - Conjunctions are danger signs: and, or, with, also
- Do not ramble
  - Long sentences with arcane language
  - References to unreachable documents (traceability issues)
- Do not speculate
  - There is no room for "wish lists" – Things that somebody probably wants
  - Danger signs: vague subject type and generalization words such as usually, generally, often, normally, typically
- Do not express suggestions or possibilities
  - Suggestions that are not explicitly stated as requirements are invariably ignored
  - Danger signs: may, might, should, could, perhaps, probably
- Avoid wishful thinking
  - Wishful thinking means asking for the impossible (e.g., 100% reliable, safe, handle all failures, fully upgradeable)

Templates for Specifications

- Different templates for requirements specifications
  - IEEE 830-1998 and 233-1998 - Standard for Software Requirements Specifications
- Describes the content and qualities of a good software requirements specification (SRS)
  a. Correct;
  b. Unambiguous;
  c. Complete;
  d. Consistent;
  e. Ranked for importance and/or stability;
  f. Verifiable;
  g. Modifiable;
  h. Traceable.

The IEEE 830-1998 Template

1. Introduction
    1. Purpose
    2. Document Conventions
    3. Intended Audience and Reading Suggestions
    4. Product Scope
    5. References
2. Overall Description
    1. Product Perspective
    2. Product Functions
    3. User Classes and Characteristics
    4. Operating Environment
    5. Design and Implementation Constraints
    6. User Documentation
    7. Assumptions and Dependencies
3. External Interface Requirements
    1. User Interfaces
    2. Hardware Interfaces
    3. Software Interfaces
    4. Communications Interfaces
4. System Features
    1. System Feature 1
    2. System Feature 2 (and so on)
5. Other Nonfunctional Requirements
    1. Performance Requirements
    2. Safety Requirements
    3. Security Requirements
    4. Software Quality Attributes
    5. Business Rules
6. Other Requirements

Requirement Verification and Validation

Requirement Verification

- Checks consistency of the software requirements specification artifacts and other software development products (design, implementation, ...) against the specification

Requirement Validation

"Validation is the process of establishing that the requirements and models elicited provide an accurate account of stakeholder requirements."

- Checks that the right product is being built
- Ensures that the software being developed (or changed) will satisfy its stakeholders
- Checks the software requirements specification against stakeholders goals and requirements
- Two main difficulties:
  - Question of truth and what is knowable ("Everybody lies." [Dr. House #101]).
  - Reaching agreement among different stakeholders.

Requirement Management and Evolution

Management and Traceability

- Requirement Management
  "The ability, not only to write requirements, but also to do so in a form that is readable and traceable by many, in order to manage their changes over time ."

- Requirement Traceability
  "The ability to describe and follow the lifecycle of a requirement in both forwards and backwards directions."

Managing Requirements

- The fundamental goal is to keep project within costs, within budget, and to meet customers needs.
- Estimate cost of system based on requirements.
- Control the volatility of the requirements.
- Manage the requirements configuration of the system
- Negotiate requirement changes
- Re-estimate cost of the system when requirements change.

Requirement Evolution

- Adding requirements
    - Changing stakeholder needs
    - Missed in initial analysis
- Requirement Scrubbing – Removing requirements
    - Usually only during development, to forestall cost and schedule overruns
- Manage inconsistency in requirements
- Managing changing requirements
    - Continued requirement elicitation
    - Evaluation of Risk
    - Evaluation of systems in the environment

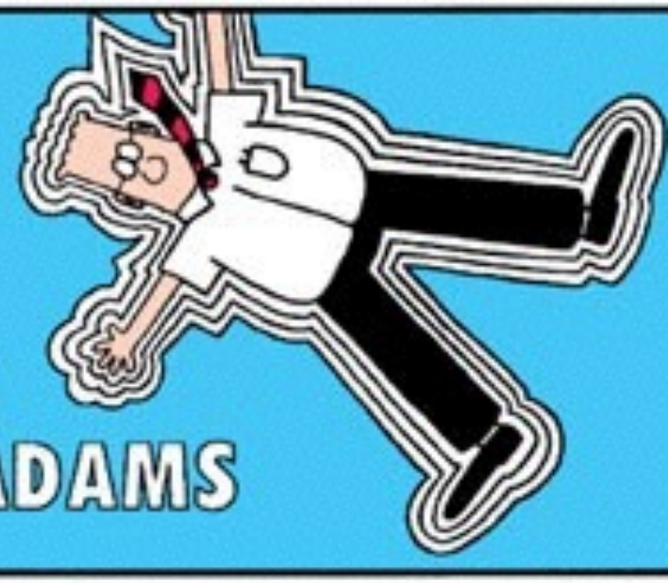
Product Families

- Increasingly important form of development activity
- Range of software product that share similar requirements and architectural characteristics, yet differ in certain key requirements
- Research issue : Identifying the core requirements for the architectures that are:
  - Stable in the presence of change
  - Flexible enough to be customized and adapted to changing requirements

Conclusion

DILBERT®

BY SCOTT ADAMS



Conclusion

- Requirements cannot be limited to system functionality specification.
- They must also focus on modeling indicative and optional properties of the environment
- It is utopian to try to specify consistent and comprehensive requirements.

References

- Karl E. Wiegers, "More about Software Requirements", Microsoft Press, 2006

- Bashar Nuseibeh and Steve Easterbrook. "Requirement Engineering – A Roadmap"

- "Requirements Engineering A good practice guide," Ian Sommerville and Pete Sawyer, John Wiley and Sons, 1997