CS4500 SOFTWARE DEVELOPMENT

REQUIREMENTS

Michael Weintraub
Spring, 2016
Unit Objective

• Understand what requirements are
• Understand how to acquire, express, validate and manage requirements
“Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.”

- Attributed to Donald Knuth
## Definitions

<table>
<thead>
<tr>
<th>A thing demanded or obligatory</th>
<th>Dictionary.com</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>Constraints</strong>, <strong>demands</strong>, <strong>necessities</strong>, <strong>needs</strong>, or <strong>parameters</strong> that must be met or satisfied, usually within a certain timeframe.</td>
<td></td>
</tr>
<tr>
<td>2. <strong>Marketing</strong>: A <strong>standard</strong> of <strong>benefit</strong>, <strong>cost</strong>, timeliness, and <strong>value</strong> of a <strong>product</strong> or <strong>service</strong> as expressed or perceived by a <strong>customer</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

| Software Requirements is a field within **software engineering** that deals with establishing the needs of stakeholders that are to be solved by software. The IEEE Standard Glossary of Software Engineering Technology defines a software requirement as: |
|-------------------------------|----------------|
| 1. A condition or capability needed by a user to solve a problem or achieve an objective. |
| 2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. |
| A documented representation of a condition or capability as in 1 or 2. |

*A requirement is a description of a system feature, capability, or constraint and should **focus on what** a system should do **rather than how** it should or could be done.*
Expressing a Requirement in English– RFC 2119

In many standards track documents several words are used to signify the requirements in the specification. These words are often capitalized. This document defines these words as they should be interpreted in IETF documents. Authors who follow these guidelines should incorporate this phrase near the beginning of their document:

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Note that the force of these words is modified by the requirement level of the document in which they are used.

1. **MUST**  This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.

2. **MUST NOT**  This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.

3. **SHOULD**  This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

4. **SHOULD NOT**  This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

5. **MAY**  This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does include a particular option MUST be prepared to interoperate with another implementation which does not include the option (except, of course, for the feature the option provides.)
Other Ways to Express Requirements

• Ad hoc diagrams or models
• Formal Specifications
  – Mathematical Notations
    • Functional or Declarative statements
  – Well defined modeling paradigms
    • Unified Modeling Language (UML)
    • Finite State Machines
    • Petri Nets

Anything will do, provided the language used is sufficiently descriptive, and everyone involved can understand it and use it effectively.
The Challenge With English

It’s expressive, intuitive, and universal

It may be vague and ambiguous, and statements are open to reader interpretation

The assignment SHALL be due on the date assigned.

At day/mon/year hh:mm is pretty clear

At day/mon/year is not

I saw a man on the hill with a telescope (5 valid interpretations)
Requirements Set The Stage For Success

Use

• Defines the commitment between the clients and the delivery organization, along with budget and schedule
  – May be set upfront or evolve incrementally
  – 60% of projects face complex domains (Dr. Dobb’s, State of the Union Survey, 2014)

• Defines the acceptance criteria
  – ~45% of project failures involve requirements phase issues (Chaos Study)
    • Incomplete requirements (13%)
    • Lack of user involvement (12%)
    • Changing specifications (9%)
    • Unrealistic expectations (10%)

Risks

• Each individual understands the same statement differently

• Clear identification of **Minimal Required Functionality**
  – Tabula Rasa Disease
    • *Real versus perceived needs*
  – Marketing doing architecture/design
  – Technology not appreciating difficulty, explicit or implied

• Perceived versus Actual Agreements

Don’t lose sight that at the end of the day, it’s about what gets delivered. But if you don’t know where you are going, it’s hard to aim right. And then the project is called *research*. 
Defining What The System Should Do

• User Requirements
  – English statements of what the system is expected to provide to system users and the constraints under which the system must operate
  – The system must generate a report every day listing who came to class.

• System Requirements
  – “Detailed” descriptions of system functions, services, and operational constraints

1. At 10pm local time, a report listing every NEU student who appeared in the MSD classroom MUST BE generated.
2. For each student in the report, the listing SHALL include the student’s name, myNEU ID, time entered, time left, and a listing of materials turned in
3. Only the professor and TA’s may (SHALL?) see this report
4. The report SHALL be delivered via email

A UML representation of the requirements for a requirements system
Organizing Requirements

**Functional**
- Describes what the system should do
  - Specifies the **FUNCTIONALITY** the system should deliver
    - At the right level of detail
    - So the client(s) clearly understands the meaning and intent
    - As well as the people who will design, implement and maintain the system

**Non-Functional**
- Describes everything not function related
  - Performance
  - Reliability
  - Cost/Resource
  - Scaling
  - Organizational aspects
  - Environmental
  - Regulatory
  - Safety
  - Security
Keys to Requirements Engineering Processes

- Engage the clients closely
- Find what users want and what works for them
- Model the knowledge
- Document!!

Reflect on what you learn
Requirements Engineering Process

Stage 1: Data (knowledge) Collection Techniques:
- Interviews
- Questionnaires
- Concurrent Protocols
- Observation
- Card Sorting
- Carrying out a role myself

Stage 2: Data (knowledge) Analysis:
- Organization Details
- User Task Information
- Good & Bad aspects of current systems

Stage 3: Data Modelling
- TKS for Tasks within Roles

Stage 4: Requirements Elicitation
- Requirement SET: 1

Stage 5: Refinement to Requirements, (based on participation):
- Requirement SET: 2

Stage 6: Agreed set of prioritised Requirements, (based on participation):
- Requirement SET: 3


Source: Sommerville, *Software Engineering, 10th Ed*, Fig 4.6
Acquiring Requirements

Approach
• Work closely with stakeholders
  – To learn about the application domain
  – To identify needed services and features
  – Constraints or restrictions
• Organize and sort the findings
• Prioritize

But stakeholders…
• May not know exactly what they want
• Their words have specialized (and not understood) meanings
• Have implicit knowledge assumptions
• May contradict each other
• May play politics
• Change their minds
• React to changes in the environment (internal or external influences)
Interviewing to Gather Needs

- Closed Interviews
  - predefined questions
- Open
  - no agenda

- Time consuming for the domain experts
  - They have a “day” job
- Vision for system may not be aligned across people/organizations
- Fraught with jargon and assumptions about domain understanding
- Building trust takes time
- Interviewers bring their own biases
Ethnography to Understand the Context

• Observes operational processes
  – Immersion in the working environment

• Describes how the system fits into the greater environment
  – Describes how things *actually work* rather than idealized processes
  – Changes the focus from the individual to the system
Are These Good or Bad Requirements?

1. The system SHALL work with any browser.
2. The system MUST respond quickly to user clicks.
3. The system SHALL store user information in an Oracle database including name, DOB, address...
4. The user MUST have Firefox installed.
5. The system should never crash.
6. The system MUST never be hacked.

- Browser list not testable
- Quickly not quantified
- Why is Oracle important here?
- Not a system requirement
- Not feasible or testable
- Not feasible or testable
Good or Bad?

1. The system SHALL enforce a 6% sales tax on all orders being shipped to Mass.
2. The system SHALL automatically calculate sales taxes based on relevant sales tax laws.
3. Sales SHALL be alerted as to which contracts will be expiring within the upcoming 90 days.
4. Every order SHALL be given a unique identifier.
5. All data stored in the filesystem MUST be encrypted with 128-bit AES encryption.
6. Every user SHOULD read the “Mythical Man Month” before using the system.
7. The fantasy league payout WILL be 30% of the difference in score of the next India-Pakistan test match.
8. The fantasy league payout WILL be 30% of the difference in score of the previous India-Pakistan test match.
9. The client MUST be available on iOS and Android.
10. All network communications SHOULD be done using http.
Storytime

A Description Of How Things Might Work As a Series of Scenarios

1. What system and users expect when the scenario starts
2. The normal flow of events
3. What can go wrong and how each is handled
4. What is happening in parallel
5. The system state at the end of the scenario

• Can be text or charts
Agreeing on Requirements is a Negotiation

Identifying the *Crowd

Identify the IN Crowd
- Inaccurate
- Incomplete
- Infeasible
- Inconsistent

Identify the UN Crowd
- Unclear (ambiguous)
- Unnecessary
- Untestable

Goodenov Principle

There is a diminishing return on perfection. There will be a point when something is sufficient.

Prioritize the Need

Establishing traceability

No requirements set is ever perfect or complete. Changes are inevitable. It’s how you manage change that often determines success.

Put another way:

You’ve got to know when to hold ’em
Know when to fold ’em
Know when to walk away
And know when to run…
The Guide for Subsequent Activity
- But remember, there will be feedback from the field

Contains
- List of functional requirements
- List of non-functional requirements
- Interface requirements
- Hardware specifications
- Essential domain knowledge
- High level system models
  - Conceptual
  - Use cases

Many tools can work
- Excel, Word, Sharepoint
- Rational
- ReqPro

We will be using confluence
Validity Checks

- Do the requirements meet the *real* needs?
- Do any requirements conflict?
- Is the requirements set complete?
- Are the requirements realistic / feasible?
  - Technically realistic
  - Budget realistic
- Are the requirements verifiable?
  - Can tests be defined so one can demonstrate the system satisfies the requirement?
If the problem is complex enough, it will likely never be completely described.

Even if the system is deployed, changes come.
- The environment changes.
- Once used, new priorities or requirements come to light.
- Compromises get exposed; priorities change.
Traceability

Insuring that a requirement gets implemented

Identifying the impact of change
Change Management

Change is Inevitable

- Create a Change Proposal
  - Is the proposed change valid? Clear?

- Assess the Impact
  - Using tracing data, estimate the impact to the design, test, and deployment elements
  - Costs are estimated
  - May be decided upon by a Change Board

- Sign-off/Change Orders

- Implement the change