Mastering Requirements Management with Use Cases
Module 2: Introduction to RMUC
Objectives

- Define key requirements management terms.
- Identify contributing factors to project success and project failure.
  - Describe how requirements management increases the chances of project success.
- Describe qualities of requirements sets.
  - Verifiable, traceable, unambiguous.
- Describe the RUP® requirements management workflow, roles, and artifacts.
Some Familiar Situations...

- **We built everything that you requested!**
- **Yes, but it is not what I want.**
- **Why didn't you tell us you wanted that feature?**
- **No one asked me!**

- **Hmm? I guess they mean for it to work this way.**
- **I have an idea for a great, new feature. Can you throw it in?**
- **No problem!**
Introduction to RMUC: Overview

Needs

Features

Software Requirements

Test Procedures

Design

User Doc

Problem

Solution Space

The system to be built

Traceability
Definitions

♦ Requirement
  ▪ A condition or capability to which the system must conform.

♦ Requirements management
  ▪ A systematic approach to:
    • Eliciting, organizing, and documenting requirements.
    • Establishing and maintaining agreement between customer/user and the project team on the changing requirements.
What Do Software Requirements Specify?

- Inputs
- Functions
- Non-Functional Requirements (E.g. Performance)

System

Outputs

Design Constraints

(E.g. Environments)
## Definitions

### Stakeholder Request

A **solution-independent** expression of a stakeholder’s desired state in the solution or subject domain.

### Feature

An externally observable service by which the system directly fulfills one or more stakeholder requests.

### Software Requirements

- **Functional Requirement**
  
  A requirement that specifies, from a black-box perspective, how the solution interacts with the outside world.

- **Non-Functional Requirement**
  
  A requirement that expresses, from a black-box perspective, the quality attributes of the solution.

### Constraint

A limitation on the design of the system or the process used to build the system.
Examples from the Course Registration System

Stakeholder Request
- Need less administrative overhead for registration.
- Professors need immediate access to student grades.

Feature
A tree browser provides a way to view student information, by semester by class.

Software Requirement

*Functional*
- The use case starts when the student selects the “register for course” command. The system displays the list of available courses…

*Non-Functional*
- 99% of 24/7 availability (3.65 days downtime per year)

Constraint
- Operate on the College DEC VAX Main Frame.
Characterization of Definitions

Based on Leffingwell & Widrig, 1999
Requirements Exist at Many Levels

- Stakeholder Needs
- Product or System Features
- Software Requirements
- Design Spec
- Test Procedures
- Documentation Plans
Requirements Management Is Not Easy Because…

- **Requirements:**
  - Are not always obvious.
  - Come from many sources.
  - May not always be easy to express clearly in words.
  - Relate to one another and to other deliverables of the software engineering process.
  - Have unique properties or property values.
  - Change.
  - Are difficult to control in large numbers.
Management Requires Strategy

RM Plan

---

RUCS10:
RM Plan
Effective Requirements Management

- Maintain a clear statement of the requirements with:
  - Good quality requirements
  - Applicable attributes for each requirement type
  - Traceability to other requirements and other project artifacts

The GOAL is to deliver **quality** products **on time and on budget** that meet the customer’s **real needs**.
What is a “Quality Product”?

- Quality: *Old Concepts*
  - Satisfies requirements documents.
  - Passed system test.
  - Development adhered to process.
  - Activity-based paradigm

- Quality: *Modern Concept*
  - Understand all stakeholder needs.
  - Continually assess all artifacts to see if needs are met.
  - Results-based paradigm
## Dimensions of Quality

### Components of FURPS+

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functionality</strong></td>
<td>Feature set capabilities, security, generality</td>
</tr>
<tr>
<td><strong>Usability</strong></td>
<td>Human factors aesthetics, consistency, documentation</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td>Frequency/severity of failure, recoverability, predictability, accuracy, MTBF</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>Speed efficiency, resource usage, throughput, response time</td>
</tr>
<tr>
<td><strong>Supportability</strong></td>
<td>Testability, Adaptability, Compatibility, Serviceability, Localizability</td>
</tr>
</tbody>
</table>

Grady, 1992
On Time and On Budget

How much work can we do?

Resources

Scope

Budget

Time
Meet the Customer’s Real Needs

Feature 1: The system must...
Feature 2: The system must...
Feature 3: The system shall...
Feature 4: The system shall...
Feature 5: The system must...
Feature 6: The system shall…
Feature 7: The system must...
Feature n: The system shall...

How do we know what the needs are?
How do we determine priority?
What is in the baseline?

How do we determine priority?

Time

Project Start Date
Target Release Date
Requirements Enable Agreement

Customer
User Community

System
To Be Built

Requirements
Verification

Surrogate
Goal

The Goal

Adapted from Al Davis
What Factors Contribute to Project Success?

Project Success Factors

- **28%** of projects completed on time and on budget.
- **49%** of projects overran original estimates.
  - Time overrun averaged 63%.
  - Cost overrun averaged 45%.
- **23%** of projects canceled before completion.

### The CHAOS Ten

<table>
<thead>
<tr>
<th>Rank</th>
<th>Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Executive Management Support</td>
</tr>
<tr>
<td>2</td>
<td>User Involvement</td>
</tr>
<tr>
<td>3</td>
<td>Experienced Project Manager</td>
</tr>
<tr>
<td>4</td>
<td>Clear Business Objectives</td>
</tr>
<tr>
<td>5</td>
<td>Minimized Scope</td>
</tr>
<tr>
<td>6</td>
<td>Standard Software Infrastructure</td>
</tr>
<tr>
<td>7</td>
<td>Firm Basic Requirements</td>
</tr>
<tr>
<td>8</td>
<td>Formal Methodology</td>
</tr>
<tr>
<td>9</td>
<td>Reliable Estimates</td>
</tr>
<tr>
<td>10</td>
<td>Other</td>
</tr>
</tbody>
</table>

Standish Group, '01 (www.standishgroup.com)
Size Is Important

Success by Project Size

Success Rate (%)

Project Size ($)

Standish Group, ‘99 (www.standishgroup.com)
The High Cost of Requirement Errors

The 1-10-100 Rule

Relative cost to repair errors:
When introduced vs. when repaired.

“All together, the results show as much as a 200:1 cost ratio between finding errors in the requirements and maintenance stages of the software lifecycle.”

Average cost ratio 14:1
Grady 1989

Boehm 1988

Requirements Time
Design
Coding
Unit Test
Acceptance Test
Maintenance
Help Projects Succeed

- **Problem analysis**
  - Understand the problem.
  - Gain stakeholder agreement.
  - Clear statement of business objectives.

- **Requirements elicitation**
  - Identify who will use the system (actors).
  - Elicit how the system will be used (use cases).

- **Requirements management**
  - Specify requirements completely.
  - Manage expectations, changes, and errors.
  - Control scope creep.
  - Enlist all team members.
Involve the Whole Team in Requirements

- Developers, Testers, and Writers
  - Help develop requirements management practices.
  - Monitor adherence to practices.
  - Verify elicitation process.
  - Document requirements.
  - Participate in requirements reviews.
  - Participate in or chair a Change Control Board (CCB).
  - Review traceability outcomes.
  - Verify quality, testability, and completeness.
Value Diminishes as Quality is Compromised
Qualities of Software Requirements Sets

- Correct
  - Is a true statement of something the system must do.
- Complete
  - Describes all significant requirements of concern to the user.
- Consistent
  - Does not conflict with other requirements.
- Unambiguous
  - Is subject to one and only one interpretation.

Qualities of Software Requirements Sets (cont.)

- **Verifiable**
  - Can be tested cost effectively.

- **Ranked for importance and stability**
  - Can be sorted based on customer importance and stability of the requirement itself.

- **Modifiable**
  - Changes do not affect the structure and style of the set.

- **Traceable**
  - The origin of each requirement can be found.

- **Understandable**
  - Comprehended by users and developers.

Qualities of a Requirement: Verifiable

A requirement is verifiable if:

- There exists some finite, cost-effective process with which a person or machine can check that the product meets the requirement.

- The system supports up to 1,000 simultaneous users.
- The system shall respond to an arbitrary query in 500 msec.
- The color shall be a pleasing shade of green.
- The system shall be available 24 x 7.
- The system shall export view data in comma-separated format, according to the IEEE specification.

Are these requirements verifiable? If not, what is a better way to state them?
Qualities of a Requirement: Traceable

Stakeholder Requests

Features

Use Cases

Supplementary
Qualities of a Requirement: Unambiguous

- A requirement is unambiguous if it has only one interpretation.

"A shall do B to C"

ref - IEEE 830
Exercise: Exploring Ambiguity

*Mary had a little lamb.*

In the space below, write (or draw) your detailed understanding of what this sentence means.

Gause & Weinberg, 1989
Explore Ambiguity: Dictionary Definitions

had - Past of have

have - 1a: To hold in possession as property
  4a: To acquire or get possession of: OBTAIN (best to be had)
  4c: ACCEPT; to have in marriage
  5a: To be marked or characterized by (have red hair)
  10a: To hold in a position of disadvantage or certain defeat
  10b: TRICK, FOOL (been had by a partner)
  12: BEGET, BEAR (have a baby)
  13: To partake of (have dinner)
  14: BRIBE, SUBORN (can be had for a price)

lamb - 1a: A young sheep esp. less than one year old or without permanent teeth
  1b: The young of various other animals (as smaller antelopes)
  2a: A person as gentle or weak as a lamb
  2b: DEAR, PET
  2c: A person easily cheated or deceived especially in trading securities
  3a: The flesh of lamb used as food
<table>
<thead>
<tr>
<th>Have</th>
<th>Lamb</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1a</td>
<td>1a</td>
<td>Mary owned a little sheep under one year of age or without permanent teeth.</td>
</tr>
<tr>
<td>4a</td>
<td>1a</td>
<td>Mary acquired a little sheep under one year of age or without permanent teeth.</td>
</tr>
<tr>
<td>5a</td>
<td>1a</td>
<td>Mary is the person who owned a little sheep under one year of age or without permanent teeth.</td>
</tr>
<tr>
<td>10a</td>
<td>1a</td>
<td>Mary held a little sheep under one year of age or without permanent teeth in a position of disadvantage.</td>
</tr>
<tr>
<td>10b</td>
<td>1a</td>
<td>Mary tricked a little sheep under one year of age or without permanent teeth.</td>
</tr>
<tr>
<td>12</td>
<td>1b</td>
<td>Mary gave birth to a young antelope.</td>
</tr>
<tr>
<td>12</td>
<td>2a</td>
<td>Mary is (or was) the mother of a particular small, gentle person.</td>
</tr>
<tr>
<td>13</td>
<td>3a</td>
<td>Mary ate a little of the flesh of lamb.</td>
</tr>
<tr>
<td>14</td>
<td>2c</td>
<td>Mary bribed a small person trading in securities who was easily cheated.</td>
</tr>
</tbody>
</table>
Explore Ambiguity: An Observation

Understandability

Ambiguity

The sweet spot
What Is NOT in a Requirement?

- **Design**
  How to accomplish the requirements.
  Design Model specifies components of a system or their interfaces with other sub-components.

- **Verification**
  How you know the requirements have been met.
  Test Suite contains a set of test scripts and test logs.

- **Project Data**
  When the requirements are met.
  Software Development Plan specifies schedules, verification and validation plans, and configuration management plans.

Adapted from Alan Davis
RUP: A Framework for Requirements Management
Requirements Discipline: Workflow Details

- New System
- Existing System
- Analyze the Problem
  - Incorrect problem
- Understand Stakeholder Needs
  - Addressing correct problem
- Define the System
- Manage the Scope of the System
  - Can't do all the work
- Refine the System Definition
  - Work in scope
- Manage Changing Requirements
  - New Input
Roles and Artifacts

- System Analyst
- Stakeholder Requests
- Requirements Management Plan
- Glossary
- Requirements Attributes
- Supplementary Specifications
- Vision
- Use-Case Model
- Storyboard
- Requirements Specifier
- Use Case
- Software Requirements Specification
- Use-Case Package
- Software Requirement
<table>
<thead>
<tr>
<th>Question</th>
<th>Artifact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where is the problem defined?</td>
<td>Vision</td>
</tr>
<tr>
<td>Where are the stakeholders and users listed?</td>
<td>Supplementary Spec</td>
</tr>
<tr>
<td>Where are the environments and platforms identified?</td>
<td>Use Case Specs</td>
</tr>
<tr>
<td>Where are the non-functional requirements located?</td>
<td></td>
</tr>
<tr>
<td>Where are the use cases maintained?</td>
<td></td>
</tr>
<tr>
<td>Where is the common terminology stored?</td>
<td>Glossary</td>
</tr>
<tr>
<td>Where are the stakeholder Needs/Requests captured?</td>
<td>Stakeholder Requests</td>
</tr>
</tbody>
</table>

What Artifacts Are Used to Manage Requirements?
One Discipline – Many Paths (One Possible Configuration)

The focus changes from phase to phase and iteration to iteration.
1. What is a requirement?
2. What is requirements management?
3. What factors contribute to project success?
4. What team members are involved in requirements management and how?
5. How would you explain the 1-10-100 rule?
6. Why would you use a development case?
7. What are some requirement characterizations?
8. What are some qualities of requirements?