Software Engineering – Fall 2015
(CSC 4350/6350)
TR. 5:30 pm – 7:15 pm

Rao Casturi
09/22/2015
http://cs.gsu.edu/~ncasturi1
Recap - Requirement Elicitation
Recap – Requirement Elicitation

• Understanding the “System”

• Requirement Elicitation has 2 phases
  – Requirement Specifications (Functional and Non Functional Requirements)
    • Natural Language
    • Focused around the communication with the client & user community
  – Analysis Model (Dynamic, Analysis Object Model)
    • More of formal or semi formal object models
    • Communication between Developers and technical teams

• Concepts: (5)
  – Functional (Need to have)
  – Non-Functional (Not directly related to the System)
  – Realism, Tractability, Verifiability
  – Complete, Unambiguous
  – From scratch, reengineering, just interface
Goal of this activity is to capture all the “**Shall**” statement sentences from the requirement documents as project requirements as a tabular form.

End of this activity the “Requirements Trace Matrix” is a deliverable.

Why is this activity important?
- **Scope Creep**
- Foundation for next phases (Use Cases, Categories, Classes, Methods)
- Cost & Time lines of project
- Future development (Incremental) Issues
Recap - Requirement Elicitation Activities:

- **Main Theme in this phase**
  - Identify the Actors
  - Identify Scenarios
  - Identify Use Case
  - Refine the Use Cases
  - Identify the Relations
  - Terminology documentation
  - Identify Non-Functional Requirements

- **RAD (Requirement Analysis Document) Work product**
Recap – RTM (Requirements Traceability Matrix)

First set of **“Shall”** identified with a unique line entry

- **Req. Entry Number**
- **Paragraph Number**
- **Requirement sentence**

Some Definitions

- **Shall** statement is a conceptual requirement for the complete system development
- **Requirements Trace Matrix (RTM)** an organized list of system requirements gathered during the initial phase from the system specification or requirement documents.
- **Derived Requirements** where the “Shall” can’t be seen but can still be a requirement for the overall project or system

<table>
<thead>
<tr>
<th>Entry</th>
<th>Paragraph</th>
<th>Requirements Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>The SEM shall monitor all occupied living quarters</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>There shall be a total of 48 living quarters</td>
</tr>
<tr>
<td>3</td>
<td>2.1</td>
<td>There shall be 3 sensors in each living quarters</td>
</tr>
<tr>
<td>4</td>
<td>2.1</td>
<td>There shall be 1 sensor for each environment condition</td>
</tr>
</tbody>
</table>

Pre-agreed Types/Categories

- Software Requirement (SW)
- Hardware Requirement (HW)
- Derived Requirement (DR)
- Nice to have Requirement (NTH)
- Software Constraint (SWC)
- Functional (F) Non Functional (NF)

Only One per Project

Requirements Traceability Matrix

GSU: Software Engineering - CSC4350/6350 - Rao Casturi
Requirement Elicitation – Management
Putting it all together

➤ RAD (Requirement Analysis Document)

1. Introduction
2. Current System
3. Future System
4. System Model
5. High Level Time Line Diagrams
6. Appendix (Proto type, Screen Shots etc)
7. Glossary
Requirement Traceability Matrix

- Simple Tabular view of the requirements collected
- Living document over the project life

```
<table>
<thead>
<tr>
<th>Entry #</th>
<th>Para #</th>
<th>HCC Requirements Traceability Matrix</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0</td>
<td>The SEM shall monitor all occupied living quarters.</td>
<td>SW</td>
</tr>
<tr>
<td>2</td>
<td>2.0</td>
<td>There shall be a total of 48 living quarters.</td>
<td>HW, SWC</td>
</tr>
<tr>
<td>3</td>
<td>2.1</td>
<td>Air Pressure, Temperature, and Oxygen % shall be obtained once per minute.</td>
<td>SW, P</td>
</tr>
<tr>
<td>4</td>
<td>2.1</td>
<td>There shall be three sensors in each living quarter.</td>
<td>HW, SWC</td>
</tr>
<tr>
<td>5</td>
<td>2.1</td>
<td>There shall be one sensor for each environmental condition.</td>
<td>HW, SWC</td>
</tr>
<tr>
<td>6</td>
<td>2.2</td>
<td>A current value representing a deviation &gt;= 1% but &lt;2% from the nominal value shall cause the window in the panel to be lit.</td>
<td>SW</td>
</tr>
<tr>
<td>7</td>
<td>2.2</td>
<td>A current value representing a deviation &gt;= 2% but &lt; 3% from the nominal value shall cause the window in the panel to be lit and flash two times per second.</td>
<td>SW</td>
</tr>
<tr>
<td>8</td>
<td>2.2</td>
<td>A current value representing a deviation &gt;= 3% from the nominal value shall cause the window in the panel to be lit and flash four times per second and sound an audible alarm.</td>
<td>SW</td>
</tr>
<tr>
<td>9</td>
<td>2.2</td>
<td>There shall be one audible alarm.</td>
<td>HW, SWC</td>
</tr>
<tr>
<td>10</td>
<td>3.0</td>
<td>The panel shall accommodate 48 annunciators arranged in six rows.</td>
<td>SW⁴</td>
</tr>
<tr>
<td>11</td>
<td>3.0</td>
<td>Each annunciator shall be mapped to a unique living quarter.</td>
<td>HW, SWC</td>
</tr>
<tr>
<td>12</td>
<td>3.0</td>
<td>Each annunciator shall accommodate three windows.</td>
<td>SW⁵</td>
</tr>
</tbody>
</table>
```

Source: Use Cases combined with BOOCH OMT UML – Putnam Texel & Charles Williams
Different Flavors of Software Development Methodologies
Software Development Methodologies

- Water fall (Traditional)
- Agile
- XP
- Scrum
Agile Methodology

• Goal: Release the working solution to end users – Fast Pace

• Incremental approach of Software Specification, Development and Delivery

• Suited for projects where requirements change rapidly during the development process

• Cut down on Process Bureaucracy and Documentation which may not be used by the end user
# Agile Philosophy

<table>
<thead>
<tr>
<th>Idea</th>
<th>Description of the Idea</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Involvement</td>
<td>• Clients should be closely involved throughout the development process.</td>
</tr>
<tr>
<td></td>
<td>• Clients are there to prioritize any new requirements and to evaluate released iterations.</td>
</tr>
<tr>
<td>Incremental Product Delivery</td>
<td>• Software is developed and released in a small chunks with the a clear direction from the Client</td>
</tr>
<tr>
<td>People over Process</td>
<td>• People strength is played all the time.</td>
</tr>
<tr>
<td></td>
<td>• Give the individual opportunity to show the skill to be used in the specific iteration</td>
</tr>
<tr>
<td>Change is inevitable</td>
<td>• Expect the change and be flexible to incorporate the change</td>
</tr>
<tr>
<td>Simplicity</td>
<td>• Think big but make it small to manage</td>
</tr>
</tbody>
</table>
Which one is a better pick? (Agile or Planned)

1. Do we need a detailed specification documents before moving to design and implementation?
2. Can your client carve out sufficient time for review and feedback?
3. Size of the system? Large?
4. Location of the team?
5. What type of system? Real Time?
6. System Life?
7. Top down organization structure?
8. What is your team skill set?
9. Any outside regulations?
XP – Extreme Programing

• Same concept of Agile – Incremental but done at extreme pace 1 day!
# XP Characteristics

<table>
<thead>
<tr>
<th>Idea</th>
<th>Description of the Idea</th>
</tr>
</thead>
</table>
| **Pair Programming**  | • Developers work in pairs  
                          • Code Check                                                                         |
| **Refactoring**       | • Continuous improvements to code                                                       |
| **Collective ownership** | • No single owner for the knowledge  
                          • No separate QA                                                                 |
| **Test Case – Development** | • First develop unit test cases before code can be developed                            |

Big on TESTING and PAIR Programming
Agile Project Management - SCRUM

- 3 Phases
- Sprint Cycle is usually 2 to 4 weeks
- “Scrum Master”
Questions ?