CS 4387/5387
SOFTWARE INTEGRATION AND V&V

LECTURE 11
SOFTWARE TRACING

Outline

- Requirements Tracing
- Team assignment (Tracing)
- Team work (integration)
What is Traceability

- from SEI-93-TR-25, "Key Practices of the Capability Maturity Model, Version 1.1") SPE Sub AC10.2
  - The software requirements, design, code, and test cases are traced to the source from which they were derived and to the products of the subsequent software engineering activities

- IEEE Standard Glossary of Software Engineering Terminology
  - The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another.

What is Traceability - 2

- Traceability is used to track the relationship between each unique product-level requirement and its source.
  - a product requirement might trace from a business need, a user request, a business rule, an external interface specification, an industry standard or regulation, or to some other source.
  - Pre-SRS traceability

- Also used to track the relationship between each unique product-level requirement and the work products to which that requirement is allocated.
  - a single product requirement might trace to one or more architectural elements, detail design elements, objects/classes, code units, tests, user documentation topics, and/or even to people or manual processes that implements that requirement.
  - Post-SRS Traceability
Purpose of Traceability

- Demonstrate to the customer that the requested contents have been developed
- Facilitates communications, helping customer relationship management and commitment negotiation
- Ensure that all requirements are correct and included in the test plan and the test cases
- Ensure that developers are not creating features that have never been requested

Background Issues

- High costs associated with:
  - Rework
  - Maintenance
  - Changes
Background - Rework

- The cost of rework on large scale projects has typically consumed 40 to 50 percent of the total software development budget.


Background - Fundamental Principles of Change

- In software development, change is the only constant
- The very act of creating software causes discovery that will impact the process itself;
  - Customers want to modify requirements
  - Developers want to modify the technical approach
  - Management wants to modify the project approach
- Changes in software engineering projects rarely happen in a vacuum: there are impacts
Background - Change has Management Impacts

Product Impact
- Direct
- Ripple effects
- Performance effects

Plan Impact
- Size
- Effort
- Schedule
- Cost

Background - Change has Technical Impacts

- Implementing changes can be frustrating and costly when software developers are unsure which other requirements, design, code and test products may be affected
Traceability

- These issues can be magnified with missing or poor traceability

Consequences of No Traceability - 1

- Acceptance Testing Compromised
  - It is difficult to demonstrate that the product is ready for a rapid and easy acceptance test
  - Product goes into acceptance too early and remains in acceptance too long
Consequences of No Traceability - 2

- **Project Management Compromised**
  - When re-planning, the project leader cannot be sure which requirements have already been partially or completely developed and which can still be removed without extra work.
  - De-scoping of the project can generate more work through the removal of previous work.

Consequences of No Traceability - 3

- **Product Compromised**
  - Development is not performed in order of criticality (either importance to the customer or technical complexity).
  - Nice-to-have features are developed and implemented more fully than key features.
  - Software developers may spend a large amount of time creating a wonderful system that does not correspond to any requirements:
    - Cost overrun
    - Schedule slippage
Consequences of No Traceability - 4

- Maintenance is Difficult
  - tracing tests, code, and design to originating requirements becomes laborious
  - impact of changes on related work products and requirements is difficult
  - testing becomes overly complex

Forward Traceability

- Good traceability practices allow for bidirectional traceability, meaning that the traceability chains can be traced in both the forwards and backwards directions
- Forward traceability looks at:
  - Tracing the requirements sources to their resulting product requirement(s) to ensure the completeness of the product requirement specification. (Pre-SRS)
  - Tracing each unique product requirement forward into the design that implements that requirement, the code that implements that design and the tests that validate that requirement and so on. (Post-SRS)

The objective is to ensure that each requirement is implemented in the product and that each requirement is thoroughly tested.
**Backward Traceability**

- Tracing each unique work product (e.g., design element, object/class, code unit, test) back to its associated requirement.
  - Backward traceability can verify that the requirements have been kept current with the design, code, and tests.

- Tracing each requirement back to its source(s).

---

**Forward/Backward Traceability**

*Which is Validation and Which is Verification?*
Traceability: How

<table>
<thead>
<tr>
<th>Requirement Source</th>
<th>Product Requirements</th>
<th>HLD Section #</th>
<th>LLD Section #</th>
<th>Code Unit</th>
<th>UTS Case #</th>
<th>STS Case #</th>
<th>User Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Rule #1</td>
<td>R00120 Credit Card Types</td>
<td>4.1 Parse Mag Strip</td>
<td>4.1.1 Read Card Type</td>
<td>Read_Card_Type.c</td>
<td>UT 4.1.032</td>
<td>ST 120.023</td>
<td>Section 12</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4.1.2 Verify Card Type</td>
<td>Ver_Card_Type.c</td>
<td>UT 4.2.012</td>
<td>ST 120.035</td>
<td>Section 12</td>
</tr>
<tr>
<td>Use Case #132 step 6</td>
<td>R00230 Read Gas Flow</td>
<td>7.2.2 Gas Flow Meter Interface</td>
<td>7.2.2 Read Gas Flow Indicator</td>
<td>Read_Gas_Flow.c</td>
<td>UT 7.2.043</td>
<td>ST 230.002</td>
<td>Section 21.1.2</td>
</tr>
<tr>
<td></td>
<td>R00231 Calculate Gas Price</td>
<td>7.3 Calculate Gas price</td>
<td>7.3 Calculate Gas price</td>
<td>Cal_Gas_Price.c</td>
<td>UT 7.3.005</td>
<td>ST 231.001</td>
<td>Section 21.1.3</td>
</tr>
</tbody>
</table>

Requirements Traceability Matrix

- Implemented using a word processor, spreadsheet, database or special tool
- Each requirement work products or software work products created throughout the life cycle should be identified
What Should Be in the Matrix

- requirements
- architectural design
- detailed design
- source code/objects
- test plans/cases

- Project plans and documentation are important, but usually should reference the matrix and not be entries within it

Requirements Traceability Matrix Example

<table>
<thead>
<tr>
<th>Requirement Source</th>
<th>Product Requirements</th>
<th>HLD Section #</th>
<th>LLD Section #</th>
<th>Code Unit</th>
<th>UTS Case #</th>
<th>STS Case #</th>
<th>User Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Rule #1</td>
<td>R00130 Credit Card Types</td>
<td>4.1 Parse Mag Ship</td>
<td>4.1.1 Read Card Type</td>
<td>Read_Card_Type.c</td>
<td>UT 4.1.02</td>
<td>UT 4.1.03</td>
<td>ST 120.020 ST 120.021 ST 120.022</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT 4.1.03</td>
<td>UT 4.1.08</td>
<td>UT 4.1.043</td>
</tr>
<tr>
<td>4.1.2 Verify Card Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT 4.2.012</td>
<td>UT 4.2.013</td>
<td>ST 120.035 ST 120.036 ST 120.037</td>
</tr>
<tr>
<td>4.1.2 Verify Card Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT 4.2.016</td>
<td>UT 4.2.017</td>
<td>ST 120.038 ST 120.039 ST 120.040</td>
</tr>
<tr>
<td>4.1.2 Verify Card Type</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT 4.2.031</td>
<td>UT 4.2.045</td>
<td></td>
</tr>
<tr>
<td>Use Case #132 step 6</td>
<td>R00230 Read Gas Flow</td>
<td>7.2.2 Gas Flow Meter Interface</td>
<td>7.2.2 Read Gas Flow Indicator</td>
<td>Read_Gas_Flow.c</td>
<td>UT 7.2.040</td>
<td>UT 7.2.041</td>
<td>ST 230.002 ST 230.003</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>UT 7.2.044</td>
<td>UT 7.2.044</td>
<td>ST 230.004</td>
</tr>
<tr>
<td>7.3 Calculate Gas price</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ST 231.005</td>
<td>ST 231.006</td>
<td>ST 231.007</td>
</tr>
<tr>
<td>7.3 Calculate Gas price</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ST 231.007</td>
<td>ST 231.008</td>
<td>ST 231.009</td>
</tr>
</tbody>
</table>

Each requirement, each requirements source, and each work product element must have a unique identifier that can be used as a reference in the matrix.
Roles and Responsibilities

- **Project manager**
  - Ensures all required information is provided as needed
  - Review the traceability matrix for completeness

- **Requirements engineer**
  - Update requirements traceability matrix as needed, support analysis as needed

- **Test manager**
  - Provide mapping of requirements to test products

- **Designer**
  - Provide mapping of requirements to design products

Roles and Responsibilities - 2

- **Developer**
  - Provide mapping of requirements to development products

- **Software Configuration Management staff**
  - Provide baselined inputs to the matrix
  - Baseline outputs and put notes in project archive

- **QA staff**
  - Periodically participate in/audit Requirements Management and Configuration Management activities
  - Generate Quality Audit Report if needed
A Sample Procedure

Entry Criteria
- Creation of or changes to requirements, design, code or test products
- A numbering/naming convention has been defined
- An experienced, trained individual is designated as the responsible entity for tracing the requirements
- A team of trained personnel is designated to support this activity and sufficient resources are allocated

A Sample Procedure - 2

Inputs
- Requirements Traceability Matrix (if it exists)
- Appropriate phase work products
- Approved change request
A Sample Procedure - 3

- Steps
  - Initial Requirements
    1. Determine traceability strategy and initiate matrix
  - Change
    1. Use matrix to help determine related requirements, design, code, tests, etc.
    2. Perform impact analysis
  - Update matrix with available information/new products created as a result of changes
  - Review matrix

A Sample Procedure - 4

- Outputs
  - New or updated Requirements Traceability Matrix
Another Approach: Trace Tagging

- Each requirement, each requirement source and each work product element must have a unique identifier.

- Identifiers are used as tags in the subsequent work products to identify backwards tracing to the predecessor document:
  - Software Design Specification (SDS) includes tags that identify the requirements implemented by each uniquely identified design element.
  - Unit Test Specification (UTS) includes trace tags that trace back to the design elements that each test case verifies.
  - Source code units include trace tags back to design elements.
  - Integration test cases with tags back to architecture elements.
  - System test cases with trace tags back to requirements.

Trace Tagging

- The system shall cancel the transaction if at any time prior to the actual dispensing of gasoline, the cardholder requests cancellation.

<table>
<thead>
<tr>
<th>SDS Identifier</th>
<th>SRS Tag</th>
<th>Component Name</th>
<th>Component Description</th>
<th>Type</th>
<th>Etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.01.032</td>
<td>R00104</td>
<td>Cancel_Transaction</td>
<td>Cancel transaction when the customer presses cancel button</td>
<td>Module</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Case #</th>
<th>SDS Identifier</th>
<th>SRS Tag</th>
<th>Test Case Name</th>
<th>Inputs</th>
<th>Expected Result</th>
<th>Etc</th>
</tr>
</thead>
<tbody>
<tr>
<td>23476</td>
<td>7.01.032</td>
<td>Cancel</td>
<td>Cancel Before_PIN_Entry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23477</td>
<td>7.01.032</td>
<td>Cancel</td>
<td>Cancel After_Invalid_PIN_Entry</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23476</td>
<td>7.01.032</td>
<td>Cancel</td>
<td>Cancel_Adjust_Start_Pump_Gas</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Advantage/Disadvantage?
Traceability Summary

- Change is practically inevitable in software projects
- Changes have impacts
- Those impacts must be analyzed
- Traceability makes it possible to evaluate changes to a system as it is being developed, and after it is deployed.
- Traceability implementation requires the commitment of a cross functional team
  - Requirements analyst/engineer: initiate requirements traceability and document the original tracing of the product requirements to their source
  - System and software architects create the high-level design, those practitioners add their information to the traceability documentation
  - Developers doing low-level design, code and unit testing add additional traceability information for the elements they create
  - Same for integration, system and alpha, beta and acceptance testers

Team Assignment

- Develop a traceability matrix (forward tracing) for the DH System
- Product artifacts to use:
  - Customer Need Statement and HLRD
  - SRS
  - Architecture components
    - Module view
    - C&C view