Requirements Management

Gunter Mussbacher, University of Ottawa

Based on material from:
Kotonya & Sommerville, Z. Zhang, IBM and Telelogic,
# Table of Contents

- Introduction to Requirements Management
- Traceability
- Baselines
- Change Management
- Requirements Management Tools

• A factor present in every successful project and absent in every unsuccessful project is sufficient attention to requirements.\(^1\)

---

\(^1\) Suzanne & James Robertson, “Requirements-Led Project Management”, Addison-Wesley, 2004
THE PROJECT REQUIREMENTS ARE FORMING IN MY MIND

NOW THEY'RE CHANGING... CHANGING... CHANGING... OKAY. NO, WAIT... CHANGING... CHANGING... DONE.

NATURALLY, I WON'T BE SHARING ANY OF THESE THOUGHTS WITH ENGINEERING.

I BUDGETED FOR SOME GOONS TO BEAT IT OUT OF YOU.

E-Mail: SCOTTADAMS@AOL.COM  ©1994 United Feature Syndicate
Introduction to Requirements Management
Why Do Requirements Change?

• Change in software development: as inevitable as difficult to control!
  • Better understanding: new requirements become apparent
  • Everything else is changing…
    • Business
    • Context
    • Technologies
    • Markets
    • …

• Possible responses to change
  • Add, modify, or remove requirements
Some Problems Due to Changing Requirements

- Requirements changing towards the end of development without any impact assessment

- Unmatched/outdated requirements specifications causing confusion and unnecessary rework

- Time spent coding, writing test cases or documentation for requirements that no longer exist
Requirements Management

• A systematic approach to eliciting, organizing, and documenting the requirement of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system.¹

¹ Leffingwell & Widrig 1999, p.16
Requirements management includes all activities intended to maintain the integrity and accuracy of expected requirements:

- Manage changes to **agreed** requirements
- Manage changes to **baseline** (increments)
- Keep project plans **synchronized** with requirements
- Control **versions** of individual requirements and versions of requirements documents
- Manage **relationships** between requirements
- Managing the **dependencies** between the requirements document and other documents produced in the systems engineering process
- Track requirements **status**
Requirements Management Activities (2)

- Change control
  - Proposing changes
  - Analyzing impact
  - Making decisions
  - Updating requirements documents
  - Updates plans
  - Measuring requirements volatility

- Version control
  - Defining a version identification scheme
  - Identifying requirements document versions
  - Identifying individual requirement versions

- Requirements status tracking
  - Defining a possible requirement statuses
  - Recording the status of each requirement
  - Reporting the status distribution of all requirements

- Requirements tracing
  - Defining links to other requirements
  - Defining links to other system elements

Source: Wiegers, 1999
Requirements Development (RD) and Management (RM)

Marketing, Customers, Management
Requirements

Analyze, Document, Review, Negotiate

Baselined Requirements

Current baseline
Revised baseline

Marketing, customers, management
Requirements changes
Requirements change process

Project changes

Project Environment

Source: Wiegers, 1999
Changes lead to a need for management
There is no management without:
  • Traceability
  • Baselines enabling comparisons
From a practical point of view, there is no traceability or management without appropriate tools

In theory, practice and theory are similar...
But in practice they are different 😊
Requirements Change Factors (1)

- Requirements errors, conflicts, and inconsistencies
  - May be detected at any phase (when requirements are analyzed, specified, validated, or implemented)

- Evolving customer/user knowledge of the system
  - When the requirements are developed, customers/users simultaneously develop a better understanding of what they really need

- Technical, schedule, or cost problems
  - Difficult to plan and know everything in advance
  - We may have to revisit the list of requirements and adapt it to the current situation
Requirements Change Factors (2)

- Changing customer priorities, new needs
  - May be caused by a change in the system environment (technological, business, political...), i.e., the context
  - Business and strategic goals may change
  - May be caused by the arrival of a new competitor
  - Laws and regulations may change
  - Collaborating systems may change
  - May also be caused by technology changes in the enterprise (migration to a new operating system, DBMS...)
  - May be caused by organizational changes (organizational structure, business processes, employees...)

Requirements Volatility

• Requirements continuously change
  • While the requirements are being elicited, analyzed, specified, and validated and after the system has gone into service

• Some requirements are usually more subject to change than others
  • Stable requirements are concerned with the essence of a system and its application domain
    • Derived from the client’s principal business activities or the domain model
    • They change more slowly than volatile requirements
    • E.g., a hospital will always have doctors, nurses, patients…
  • Volatile requirements are specific to the instantiation of the system in a particular environment for a particular customer at a particular time
    • E.g., in a hospital, we can think of requirements related to the policies of the government health system
Types of Volatile Requirements

• Mutable requirements
  • These are requirements which change because of changes to the environment in which the system is operating

• Emergent requirements
  • These are requirements which cannot be completely defined when the system is specified but which emerge as the system is designed and implemented

• Consequential requirements
  • These are requirements which are based on assumptions about how the system will be used
    • Once the system is in place, some of these assumptions will be wrong

• Compatibility requirements
  • These are requirements which depend on other equipment, technology, or processes
### Expectations of Requirements Management (1)

- **Identification** of individual requirements

- **Traceability** from highest level requirements to implementation
  - Established via links through a requirements database
  - Links between requirements and design models, tests, code…
  - Coverage and consistency analysis
  - What are the traceability policies? What types of links? From where? To where?

- **Impact assessments** of proposed changes
  - Analysis tools let you see which other requirements (and other linked artifacts) will be affected by a change
Expectedations of Requirements Management (2)

- **Controlled access** to current project information
  - A shared database ensures that all users are working with current data (consistency, parallel access)
  - A central repository allows all users to see the information that they need to see (visibility)

- **Change control**
  - Change proposal system implements controlled process for managing change
  - How do we collect, document, and address changes?

- **Deployment of required tool support**
  - To help manage requirements change
Identification of Requirements

• It is essential for requirements management that every requirement has a unique identification
  • The most common approach is requirements numbering based on chapter/section in the requirements document
• There are several problems with this approach
  • Numbers cannot be unambiguously assigned until the document is complete
  • Assigning chapter/section numbers is an implicit classification of the requirements ➔ may mislead readers of the document into thinking that the most important relationships are with requirements in the same section
## Requirements Identification Techniques

- **Dynamic renumbering**
  - Some word processing systems allow for automatic renumbering of paragraphs and the inclusion of cross references.
  - As you reorganise your document and add new requirements, the system keeps track of the cross references and automatically renumbers your requirements depending on its chapter, section, and position within the section.

- **Database record identification**
  - When a requirement is identified, it is entered in a requirements database and a database record identifier is assigned which is then used for all subsequent references to the requirement.

- **Symbolic identification**
  - Requirements can be identified by giving them a symbolic name which is associated with the requirement itself (e.g., SEC1, SEC2, SEC3… may be used for requirements which relate to system security).
BTW, Requirements Have Attributes!

- Apart from an identifier, requirements should have attributes that establish context and background, and go beyond the requirement description
- For **filtering**, **analysis**, **metrics**…
  - Creation date, Last update, Author, Stakeholders (Owners / Source)
  - Version number
  - Status, Priority, Importance, Stability
  - Rationale, Comments
  - Acceptance criteria
  - Subsystem / Product release number
  - …
- The more complex the project, the richer the attributes…
- Many attributes are predefined in RM tools, others are defined by requirements engineers as required by the project
## Requirements Attributes

- **Classes and attributes of a requirements management database**

<table>
<thead>
<tr>
<th><strong>SYS_MODELS</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Model: MODEL</td>
</tr>
<tr>
<td>Description: TEXT</td>
</tr>
<tr>
<td>Next: MODEL</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>REQUIREMENT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier: TEXT</td>
</tr>
<tr>
<td>Statement: TEXT</td>
</tr>
<tr>
<td>Date_entered: DATE</td>
</tr>
<tr>
<td>Date_changed: DATE</td>
</tr>
<tr>
<td>Sources: SOURCE_LIST</td>
</tr>
<tr>
<td>Rationale: REQ_RATIONALE</td>
</tr>
<tr>
<td>Status: STATUS</td>
</tr>
<tr>
<td>Dependents: REQ_LIST</td>
</tr>
<tr>
<td>Is_dependent_on: REQ_LIST</td>
</tr>
<tr>
<td>Model_links: SYS_MODELS</td>
</tr>
<tr>
<td>Comments: TEXT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>SOURCE_LIST</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>People: TEXT</td>
</tr>
<tr>
<td>Documents: TEXT</td>
</tr>
<tr>
<td>Reqs: REQ_LIST</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>REQ_RATIONALE</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Rationale: TEXT</td>
</tr>
<tr>
<td>Diagrams: GRAPHIC</td>
</tr>
<tr>
<td>Photos: PICTURE</td>
</tr>
</tbody>
</table>

- **Select only the necessary attributes!**
### DOORS – Objects and Attributes

#### Introduction

**Traceability**
- Baselines
- Change Management
- Requirements Management Tools

#### Requirements Management Tools

<table>
<thead>
<tr>
<th>User requirements for passenger car</th>
<th>v1.0</th>
<th>v2.0</th>
<th>v2.5</th>
<th>v3.0</th>
<th>Verification Method</th>
<th>Test Results</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users shall be able to travel 2000 kilometers without the need for any form of additional fuel.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to travel 2000 kilometers without the need for any form of additional fuel.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.3.3 Stopping</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Test</td>
<td>Pass</td>
<td>Medium</td>
</tr>
<tr>
<td>Users shall be able to stop safely</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Test</td>
<td>Pass</td>
<td>Medium</td>
</tr>
<tr>
<td>Users shall be able to stop with the vehicle maintaining a straight track over the stopping distance when the steering is maintained to within ( \pm 10% ) of a straight line by the user.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Test</td>
<td>Pass</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>3.1.4 Fuel economy</strong></td>
<td>Test</td>
<td>Fail</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to obtain fuel consumption better than that provided by the 95% of cars built in 1996.</td>
<td>Test</td>
<td>Pass</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to accelerate from 0 to 100 Kilometers per hour in 10 seconds.</td>
<td>Test</td>
<td>Fail</td>
<td>Very High</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to accelerate from 0 to 100 Kilometers per hour in 8 seconds.</td>
<td>Test</td>
<td>Pass</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.6 Safety</strong></td>
<td>Test</td>
<td>Pass</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to travel in safety in accordance with the Road Research Laboratories Safety standards dated 1 January 1993.</td>
<td>Test</td>
<td>Pass</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users shall be able to travel at the same level of safety as provided by the best 10% of cars being developed to be built in 1998.</td>
<td>Test</td>
<td>Pass</td>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.6 Noise levels</strong></td>
<td>Analysis</td>
<td>Fail</td>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Requirements Statuses

• Help manage the requirement lifecycle
  • Their number and nature depend on the process in place

• Example of a set of statuses:
  • Proposed: by some stakeholder
  • Approved: part of baseline, committed to implement
  • Rejected: after evaluation
  • Implemented: designed and implemented
  • Verified: Relevant tests have passed
  • Deleted: Removed from list

• RM includes amongst its tasks the tracking of the status of all requirements during the project
Another essential aspect of requirements management

- Every version of a requirement needs to be uniquely identified
- The last version of a requirement must be available to all team members
- Changes need to be documented and clearly communicated
- A version identifier must be updated with every change to the requirement

Requirements documents should include

- A revision history: changes, dates, by whom, why...
- Standard markers for revisions (e.g., strikethrough or underlined text, coloring, line markers…)

Version control tool may be used

- To store and manage the revision history
- To store justifications (to add, modify, delete, reject a requirement)
Traceability
Traceability?

- "Can I ask you some questions?"

- "By all means."

- "Okay. Well, for starters I'll have who, what, when and where and then wither, whence and wherefore for a follow-up, and then one bit side-order of why."

Traceability Quotes (1)

• Requirements traceability refers to the ability to describe and follow the life of a requirement, in both forwards and backwards direction (i.e., from its origins, through its development and specification, to its subsequent deployment and use, and through all periods of ongoing refinement and iteration in any of these phases).¹

• A software requirements specification is traceable if the origin of each of its requirements is clear and if it facilitates the referencing of each requirement in future development or enhancement documentation.²

• Traceability gives essential assistance in understanding the relationships that exist within and across software requirements, design, and implementation.³

• A link or relationship defined between entities.⁴

Traceability Quotes (2)

- Traceability is often mandated by contracts and standards.\(^1\)
  - E.g., military and aerospace
- One cannot manage what cannot be traced.\(^2\)
- Traceability information helps assess the impact of changes to requirements, connecting these requirements as well as requirements for other representations of the system.\(^3\)
- Traceability is a property of a system description technique that allows changes in one of the three system descriptions – requirements, specifications, implementation – to be traced to the corresponding portions of the other descriptions. The correspondence should be maintained through the lifetime of the system.\(^4\)

Importance of Traceability (1)

- Requirements cannot be managed effectively without requirements traceability
  - A requirement is traceable if you can discover who suggested the requirement, why the requirement exists, what requirements are related to it, and how that requirement relates to other information such as systems designs, implementations and user documentation
Importance of Traceability (2)

• Benefits of traceability
  • Prevents losing knowledge
  • Supports the verification process (certification, localization of defects)
  • Impact analysis
  • Change control
  • Process monitoring (e.g., missing links indicate completion level)
  • Improved software quality (make changes correctly and completely)
  • Reengineering (define traceability links is a way to record reverse engineering knowledge)
  • Reuse (by identifying what goes with a requirement: design, code…)
  • Risk reduction (e.g., if a team member with key knowledge leaves)
Traceability Difficulties

- Various stakeholders require different information
- Huge amount of requirements traceability information must be tracked and maintained
- Manual creation of links is very demanding
  - Likely the most annoying problem
- Specialized tools must be used
- Integrating heterogeneous models/information from/to different sources (requirements, design, tests, code, documentation, rationales...) is not trivial

- Requires organizational commitment (with an understanding of the potential benefits)
Backward and Forward Traceability (1)

- **Backward traceability**
  - To previous stages of development
  - Depends upon each element explicitly referencing its source in earlier documents

- **Forward traceability**
  - To all documents spawned by a document
  - Depends upon each element in the document having a unique name or reference number

Source of figure: Kotonya and Sommerville
Backward and Forward Traceability (2)

- Top to bottom from requirements’ point of view
  - Forward-to traceability
    - Links other documents (which may have preceded the requirements document) to relevant requirements
    - Help validation
    - Help evaluate which requirements are affected by changes to users’ needs
  - Forward-from traceability
    - Links requirements to the design and implementation components
    - Help assure that all requirements have been satisfied
Backward and Forward Traceability (3)

- Bottom to top from requirements’ point of view
  - Backward-to traceability
    - Links design and implementation components back to requirements
    - Help determine why each item is designed/implemented
  - Backward-from traceability
    - Links requirements to their sources in other documents or people
    - Help validation
    - Help evaluate how changes to requirements impact stakeholders needs
### Types of Traceability (1)

- **Requirements – source traceability**
  - Links requirements with a person or document
- **Requirements – rationale traceability**
- **Requirements – requirements traceability**
  - Links requirements with other requirements which are, in some way, dependent on them
- **Requirements – architecture traceability**
  - Links requirements with the subsystems where these requirements are implemented (particularly important where subsystems are being developed by different subcontractors)
- **Requirements – design traceability**
  - Links requirements with specific hardware or software components in the system which are used to implement the requirement
Types of Traceability (2)

- Requirements – interface traceability
  - Links requirements with the interfaces of external systems which are used in the provision of the requirements
- Requirements – feature traceability
- Requirements – tests traceability
  - Links requirements with test cases verifying them (used to verify that the requirement is implemented)
- Requirements – code traceability
  - Generally not directly established, but can be inferred
Representation – Traceability Table

- Show the relationships between requirements or between requirements and other artifacts
- Table can be set up to show links between several different elements
- Backward and forward traceability
- Difficult to capture different types of links

<table>
<thead>
<tr>
<th>User Requirement</th>
<th>Functional Requirement</th>
<th>Design Element</th>
<th>Code Module</th>
<th>Test Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-28</td>
<td>catalog.query.sort</td>
<td>Class Catalog</td>
<td>catalog.sort()</td>
<td>search.7</td>
</tr>
<tr>
<td>UC-29</td>
<td>catalog.query.import</td>
<td>Class Catalog</td>
<td>catalog.import(), catalog.validate()</td>
<td>search.12, search.13, search.14</td>
</tr>
</tbody>
</table>
Representation – Traceability Matrix

• Define links between pairs of elements
  • E.g., requirements to requirement, use case to requirement, requirement to test case…

• Can be used to defined relationships between pairs
  • E.g., specifies/is specified by, depends on, is parent of, constrains…

• More amenable to automation than traceability table

<table>
<thead>
<tr>
<th></th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>R4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Traceability matrices become more of a problem when there are hundreds or thousands of requirements as the matrices become large and are sparsely populated.

A simplified form of a traceability matrix may be used where, along with each requirement description, one or more lists of the identifiers of related requirements are maintained.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Depends-on</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1</td>
<td>R3, R4</td>
</tr>
<tr>
<td>R2</td>
<td>R5, R6</td>
</tr>
<tr>
<td>R3</td>
<td>R4, R5</td>
</tr>
<tr>
<td>R4</td>
<td>R2</td>
</tr>
<tr>
<td>R5</td>
<td>R6</td>
</tr>
</tbody>
</table>
Example – DOORS Links

- A relationship between two objects in the DOORS database is established using a link
  - One source object and one destination object
  - Links can be followed in either direction
  - Possible to have many links between the same two objects
    - Links also have types and attributes!
DOORS – Creating and Accessing Links

1.1 Use Cases

3. Install PAS
   The CPC selects "Install PAS" from the installation menu of the cell phone. The system installs the PAS software.

5. Save photo
   The CPC selects a photo stored locally on the cell phone. See Enter Photo Details (Common).

6. Update PAS with all local photos
   The CPC selects the "Update Photos" from the PAS menu.

7. Change photo name
   The system saves the new photo name to who to send the photo. The system removes the PAS from the cell phone.

12. 1.2 Common Use Cases

14. Enter Photo Details (Common)
   The CPC enters a name for the photo. The CPC selects one of the following categories: portrait, landscape, animals, architecture, miscellaneous.

13. Save Photo (Common)
   The system checks whether the format of the photo conforms to Industry Standard "Photo-enabled Cell Phone 123.01". The system checks whether the size of the photo is less than or equal to 5MB. If the photo passes both checks, the system saves the photo, the name of photo, the category of the photo, and the current date in PAS. Otherwise, the system displays an error message for the CPC.

15. Select Photo (Common)
   The CPC selects a photo from a list of photo names or thumbnails. The CPC may switch between the list of photo names and the thumbnails.
DOORS – Exploring Traceability Links

![Traceability Explorer](image)

Object 02: PAS - InstallPAS
- 1.1: PAS.InstallPAS
- 1.2: PAS.UsePAS
- 1.3: PAS.RemovePAS
- 1.4: PAS.ViewNotification
- 1.5: PAS.CellPhoneClients
- 1.6: PAS.CellPhoneClients

Object 03: InstallPAS
- 3: InstallPAS
DOORS – Hierarchical Link View
DOORS – Types of Analysis

• **Impact** Analysis
  • Analyze out-links (which objects in other modules are affected when this module is changed)

• **Traceability** Analysis
  • Analyze in-links (changes in these objects will affect this module)

• May involve multiple levels of objects/documents
DOORS – Traceability and Software Artefacts

DOORS:
Requirements Management & Traceability

TAU/Architect & TAU/Developer:
System Modeling & Code Generation

SYNERGY/Change:
Work Orders

SYNERGY/CM:
Engineering Tasks

 ActiveCM:
Controlled Code Modules
DOORS – Analysis with Wizard

Orphans indicate missing links
What are Suspect Links?

If documents are linked ...

... a change by this user here...

... shows up as a warning flag to this user here.

- Proactively know when changes effect your requirements
- Suspect link indicates that element may have been affected by a change
- Help ensure ripple effects of changes are considered
Suspect Links

- Visible as indicators or with change notes (added/deleted)
Traceability Planning

- Planning traceability? Yes, just like any other project!
  - Who are the stakeholders?
  - What are the needs (analysis, reports…)?
    - Useful, measurable, feasible objectives
  - Definition of links and attributes
    - Can some be inferred automatically?
  - Process (who collects and when to collect traceability information)
    - Roles, access
    - Data/link input and updates
    - Exceptional situations (e.g., lack of time)
  - Representations, queries, tools
  - ...
  - Traceability policies define what and how traceability information should be maintained
Factors to Consider during Planning (1)

- Number of requirements
  - The greater the number of requirements, the more the need for formal traceability policies
- Expected system lifetime
  - More comprehensive traceability policies should be defined for systems which have a long lifetime
- Maturity level of organization
  - Detailed traceability policies are more likely to be implemented and used properly in a cost-effective way in organizations which have a higher level of process maturity
- Size of project and team
  - The larger the project or team, the greater the need for formal traceability policies
Factors to Consider during Planning (2)

- **Type of system**
  - Critical systems such as hard real-time control systems or safety-critical systems need more comprehensive traceability policies than non-critical systems

- **Additional constraints from customer**
  - E.g., compliance to military standard

- **Traceability links should be defined by whoever has the appropriate information available**

<table>
<thead>
<tr>
<th>Link Source Object Type</th>
<th>Link Target Object Type</th>
<th>Information Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>System requirement</td>
<td>Software requirement</td>
<td>System engineer</td>
</tr>
<tr>
<td>Use case</td>
<td>Functional requirement</td>
<td>Requirements analyst</td>
</tr>
<tr>
<td>Functional requirement</td>
<td>Functional requirement</td>
<td>Requirements analyst</td>
</tr>
<tr>
<td>Functional requirement</td>
<td>Test case</td>
<td>Test engineer</td>
</tr>
<tr>
<td>Functional requirement</td>
<td>Software architecture element</td>
<td>Software architect</td>
</tr>
<tr>
<td>Functional requirement</td>
<td>Other design elements</td>
<td>Designer or Developer</td>
</tr>
<tr>
<td>Business rule</td>
<td>Functional requirement</td>
<td>Requirements analyst</td>
</tr>
</tbody>
</table>
Modeling Traceability

• The types of links to use (and their attributes) must be defined for different types of requirements
  • It is a design problem!

• May be modeled with a UML class diagram (metamodel)
  • Object types (classes)
  • Object attributes (attributes)
  • Structure of folders, modules, objects
    • Stereotypes, composition…
  • Link types (associations)
    • Satisfies, tests, refines, contains, contributes to, threatens, justifies…
  • Link attributes (association classes)
  • …
Example – UCM Models Imported in DOORS

- Associations describe internal links
Example – UCM External Links in DOORS

- User Requirements (Use Cases)
  - Satisfies
  - UCMs (User Level)
  - Acceptance Tests
  - Tests

- System Requirements
  - Satisfies
  - UCMs (System Level)
  - Functional Tests
  - Tests

- Software Requirements (Subsystem)
  - Satisfies
  - Conventional links
  - External UCM links

Introduction  Traceability  Baselines  Change Management  Requirements Management Tools
Example – Automatic Link Generation (1)

• Important to minimize the manual effort for link creation

• From system requirements to user-level UCMs to user reqs.
• From tests to system-level UCMs to system requirements
Types of Traceability Links

- Note the types of links in the previous examples, as well as the types of objects they relate
  - Satisfies, Tests
  - Refines, References, Contains...
- Others could be created
  - Contributes, Contradicts, Justifies, Depends on...

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Design</th>
<th>Code (software)</th>
<th>Documentation</th>
<th>Test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1. Braking distance &lt;50 m when speed ~90 km/h</td>
<td>![Design Image]</td>
<td>public ABS control(String args[]) throws Exception { Class c = null; if (args.length == 1) {</td>
<td>Braking: The driver should push brakes sharply to the utmost.</td>
<td>Braking test:</td>
</tr>
<tr>
<td>5.2. Absorbers should be electronically controlled.</td>
<td>![Design Image]</td>
<td>}</td>
<td>...</td>
<td>- on dry asphalt;</td>
</tr>
</tbody>
</table>

- on slippery roads
- on bumpy roads
Other Example of Traceability Links

- Change request modifies Business requirement
- Business requirement drives specification of System requirements, use case, external interface, quality attribute
- System requirements modifies Business rule
- Business rule is origin of Software functional requirement
- Software functional requirement is origin of Architecture, user interface, or functional design
- Architecture, user interface, or functional design is verified by Integration test and Code
- Integration test is verified by Unit test
- Code is implemented in System test
- System test lead to creation of Project plan task
Cardinality of Traceability Links

- Depending on the traceability information, the cardinality of traceability links may be
  - One-to-one
    - E.g., one design element to one code module
  - One-to-many
    - E.g., one functional requirement verified by multiple test cases
  - Many-to-many
    - E.g., a use case may lead to multiple functional requirement, and a functional requirement may be common to several use cases
Advice for DOORS Links

• Direction of links
  • From the most concrete to the most abstract
  • To avoid access rights issues
  • To make use of the integrated analysis routines of DOORS

• Link Modules
  • One module per type of link
  • NEVER use default module (should not even be offered)
  • To avoid maintenance problems
  • Specific types facilitate analysis and filtering
Baselines
Baseline

• Non-modifiable (read-only) version of a document
  • Describes a moment in time
  • May include multiple documents at the same time
• Enables document comparison and management
• Comes with a change history for the document
  • Information on objects, attributes, and links created, deleted, or edited since the creation of the baseline
  • Often also contains information on user sessions (when the document was opened, by whom…)
• Requires access control

• It is advisable to establish a baseline for a new document that is imported into the document management system
  • In order not to lose any changes
Baseline for Requirements

- Represents the set of functional and non-functional requirements that the development team has committed to implement in a specific release.
- Before going into the baseline, the requirements should be reviewed and approved by stakeholders.
- Once in the baseline, all changes should follow a defined change control process.

Baseline

- Different viewpoints
- No formal documents
- Always changing
- Shared understanding
- Configuration management
- Change management
Baseline Usage

- Baselines may be
  - Created
    - Complete image of requirements state at a given time
  - Deleted
  - Visualized
    - Possibility to go back
  - Compared
    - To see changes since a certain time
  - Copied
  - Signed
    - For authorization, contract
DOORS – Baseline Compare

1 Introduction
These are the initial user requirements. Follow this Internet link to view the 1.1 Schedule
This module contains the user requirements baseline for September 2002.

2 User types
This section describes the nature of the user requirements.

2.1 New Nationalities
The car will be used in the countries.

2.2 User sizes
The car shall be suitable for two kilograms to 100 kilograms.

3 Requirements
This section contains the user requirements.

3.1 Capability Requirements
3.1.1 Carrying Capacity
3.1.1.1 Number of People
Four average size adults shall be able to travel in comfort as defined in the standard produced in 1999.

3.2 User sizes
The car shall be suitable for two kilograms to 100 kilograms.

3 Change Management
The car will be used in the countries.

4 Requirements
This section contains the user requirements.

4.1 Capability Requirements
4.1.1 Carrying Capacity
4.1.1.1 Number of People
Four average size adults shall be able to travel in comfort as defined in the standard produced in 1999.

Change History
Previous Baseline
Current Version
**DOORS – Module Compare**

- Change analysis between versions

<table>
<thead>
<tr>
<th>User requirements for SUV 4x4</th>
<th>Comparison with User Requirements version: 2.1 (1998)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Users shall be able to operate the clutch, if fitted, in standard footwear.</td>
<td>Users shall be able to operate the clutch, if fitted, in standard footwear.</td>
</tr>
<tr>
<td>4.1.1.7 Gears</td>
<td>4.1.1.17 Gears</td>
</tr>
<tr>
<td>Users shall be able to operate gears, if fitted, with minimal effort.</td>
<td>Users shall be able to operate gears, if fitted, with minimal effort.</td>
</tr>
<tr>
<td>4.1.8 Distance</td>
<td>4.1.88 Distance</td>
</tr>
<tr>
<td>Users shall be able to travel 1600 kilometers without the need for any form of additional fuel.</td>
<td>Users shall be able to travel 10600 kilometers without the need for any form of additional fuel.</td>
</tr>
<tr>
<td>4.1.9 Equipment malfunction</td>
<td>4.1.99 Equipment malfunction</td>
</tr>
<tr>
<td>Users shall be able to be aware of equipment malfunction within 10 seconds of the malfunction occurring.</td>
<td>Users shall be able to be aware of equipment malfunction within 10 seconds of the malfunction occurring.</td>
</tr>
</tbody>
</table>

Username: Wendy  Exclusive edit mode
Change Management
Change Management (1)

• The more things change…

• If you see change not as an enemy, but as a welcome friend, you will secure the most valuable prize of all – the future…
Change Management (2)

- Concerned with the procedures, processes, and standards which are used to manage changes to a system requirements.
- Change management policies may cover:
  - The change request process and the information required to process each change request.
  - The process used to analyse the impact and costs of change and the associated traceability information.
  - The membership of the body that formally considers change requests.
  - Software support (if any) for the change control process.

- A change request may have a status as well as requirements:
  - E.g., proposed, rejected, accepted, included...
Some requirements problem is identified
  - Could come from an analysis of the requirements, new customer needs, or operational problems with the system
  - The requirements are analysed using problem information and requirements changes are proposed
The proposed changes are analysed
  - How many requirements (and, if necessary, system components) are affected? Roughly how much would cost, in both time and money?
The change is implemented
  - A set of amendments to the requirements document or a new document version is produced (of course this should be validated with whatever normal quality checking procedures are in place)
Change Request Form

• Proposed changes are usually recorded on a change request form which is then passed to all of the people involved in the analysis of the change

• Change request forms may include
  • Date, Customer, Requester, Product including version
  • Description of change request including rationale
  • Fields to document the change analysis
  • Signature fields
  • Status
  • Comments
customers may misunderstand requirements and their context and suggest unnecessary changes

with the help of traceability information

risk is too high

negotiations with customers

cost/time required for the implementation of change is too high/long
## Different Management Aspects

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Traceability</th>
<th>Baselines</th>
<th>Change Management</th>
<th>Requirements Management Tools</th>
</tr>
</thead>
</table>

**• Change Management**
- How does a customer submit change requests?
- How is this request being monitored, prioritized, and implemented?

**• Configuration Management**
- Versioning, labelling, and tracking code and other components during the development cycle of software

**• Release Management**
- Defines how and when different hardware and software will be made available together as a product
Tool Support for Change Management

• May be provided through requirements management tools or through configuration management tools

• Tool facilities may include
  • Electronic change request forms which are filled in by different participants in the process
  • A database to store and manage requests
  • A change model which may be instantiated so that people responsible for one stage of the process know who is responsible for the next process activity
  • Electronic transfer of forms between people with different responsibilities and electronic mail notification when activities have been completed
  • Electronic signatures
  • Discussion forums
  • In some cases, direct links to a requirements database
Example – DOORS Change Proposal System (1)

- The Change Proposal System (CPS) allows people to access DOORS modules and to propose changes (without immediately changing the modules)
- This allows for feedback and the application of changes in a controlled manner
- An administrator controls the visibility of data as well as who is allowed to propose change requests

- DOORS can also be integrated with SYNERGY
  - Version/change management system
Example – DOORS Change Proposal System (2)

- Manage change; no surprises

Changes from all users including DOORSnet

Read-only user submits “Change Proposal”

Changes reviewed on-line

Submitted

In Review

Further Research

On Hold

Next Release

Accepted

Rejected

E-mail
• Standard DOORS module
Select a SYNERGY change request

2.2 Better handling of Requirements Changes

2.2.1 User Workspace in DOORS

- When opening a module and when selecting a RCR, the user shall see in DOORS all changes already proposed against the RCR.
- DOORS shall highlight which requirements have been changed using the revision box (Red Color).
- At any time during last editor session the user shall have the possibility to record all the changes he/she is proposing against a RCR.
- A user shall be able to exit a DOORS session without having finished his work and later come back and restore his edition.
Perform appropriate changes
• Changes managed by SYNERGY/Change

![Image of DOORS/SYNERGY interface](image-url)
• Once approved, the change request can be applied to DOORS
Requirements Management Tools
What Kind of Tool Do We Need?

• Different companies will use different tools, which may or may not be tailored to the requirements management task
  
  • Word processor (Microsoft Word with templates…)
  • Spreadsheet (Microsoft Excel…)
  • Industrial-strength, commercial RM tools
    • IBM/Telelogic DOORS, IBM Requisite Pro, Borland CaliberRM…
  • Internal tools
    • GenSpec (Hydro-Quebec)…
  • Open source RM tools
    • OSRMT: http://sourceforge.net/projects/osrmt
  • Bug tracking tools (free or not)
    • Bugzilla…
  • Collaboration tools (free or not)
    • TWiki…
What Should We Look For in a Tool?

- Types/attributes for requirements and links
- Specifications and models
- Version and change management
- Database repository
- Traceability
- Analysis (impact, completeness, style, differences…)
- Automatic inspection of requirements (according to rules)
- Visualization and reports

- Requirements document generation
- Monitoring of requirements statuses
- Access control
- Import/export
- Communication with stakeholders
- Scripting language (for automation)
- Reuse of requirements, models, projects
- …
RM Tool Architecture – Example

- **Introduction**
- **Traceability**
- **Baselines**
- **Change Management**
- **Requirements Management Tools**

**RM Tool Architecture – Example**

- **NL requirements document**
  - **Req. converter**
  - **Requirements database**
  - **WP linker**
  - **Change control system**
  - **Requirements report**
  - **Report generator**
  - **Traceability report**
  - **Traceability support system**
  - **Req. browser**
  - **Req. query system**
Requirements Management Implies Integration!

- Introduction
- Traceability
- Baselines
- Change Management
- Requirements Management Tools

Diagram:

- Project-tracking tool
- Test management tool
- Change-request tool
- Requirements management tool
- Version control tool
- Design modeling tool
- Project estimation tool
**Approaches – Document or Database? (1)**

- Requirements have to be stored in such a way that they can be accessed easily and related to other requirements.

- **Document (e.g., Word)**
  - Easy to use, easy to access, simple training
  - Requirements are all stored in the requirements document
  - It is easy to produce the final requirements document
Approaches – Document or Database? (2)

• Database (e.g., DOORS)
  • Good for management, controlled access, links, analysis, reports
  • Good query and navigation facilities
  • Support for change and version management
  • But: hard (and costly) to configure, manage, and use; link between the database and the requirements document must be maintained (final requirements document must be generated)

• Ideally: Target the benefits of both
  • E.g., DOORS and RequisitePro offer integrations with Word (import/export) as well as document-oriented views (for the “look and feel”…)

Introduction                Traceability                Baselines                Change Management                Requirements Management Tools
How About Evolving the Process Itself?

• Evolution of requirements types
  • Adding / modifying / deleting
    • Attributes
    • Link types
    • Requirements status
    • ...

• Evolution of change management
  • Adding / modifying / deleting
    • Attributes
    • Lifecycle status
    • Forms
    • ...

Introduction  Traceability  Baselines  Change Management  Requirements Management Tools
Thinking About Getting a RM Tool?

- The list of potential criteria and the list of products can be rather long…
  - See the INCOSE study:
    http://www.incose.org/ProductsPubs/Products/rmsurvey.aspx
    - About 25 tools and 80 criteria, with explanations

- Which are relevant to you, in your context (project, organization…)?
  - Need a goal model to define criteria and for assessment!
<table>
<thead>
<tr>
<th>ID</th>
<th>Car System Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR12</td>
<td>1.2 Control car</td>
</tr>
<tr>
<td>SR13</td>
<td>1.2.1 Switch on car</td>
</tr>
<tr>
<td>SR14</td>
<td>The car shall be able to discriminate which authorized people shall be able to switch on and operate the car.</td>
</tr>
<tr>
<td>SR15</td>
<td>1.2.2 Control speed</td>
</tr>
<tr>
<td>SR16</td>
<td>The car shall have a foot mechanism to control the speed of the car.</td>
</tr>
<tr>
<td>SR17</td>
<td>The speed control shall be infinitely variable from zero to maximum speed.</td>
</tr>
<tr>
<td>SR173</td>
<td>The speed of the car shall be controllable by automatic means.</td>
</tr>
</tbody>
</table>
DOORS – Multi-User Editing

- Make required edits, and unlock to allow others access

### Requirements

<table>
<thead>
<tr>
<th>ID</th>
<th>Car user requirements parsed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>UR118</td>
<td>The car will be used in the following countries: UK, USA, Northern Europe, Eastern Europe, Japan.</td>
</tr>
<tr>
<td>UR5</td>
<td>2.2 User sizes</td>
</tr>
<tr>
<td>UR6</td>
<td>People come in all shapes and sizes. The car must be suitable for people maximum and minimum sizes 1.8m to 2m weighing 35 kilograms to 120 kilograms.</td>
</tr>
<tr>
<td>UR7</td>
<td>3 Requirements</td>
</tr>
<tr>
<td>UR8</td>
<td>3.1 Capability Requirements</td>
</tr>
<tr>
<td>UR9</td>
<td>3.1.1 Carrying Capacity</td>
</tr>
<tr>
<td>UR10</td>
<td>3.1.1.1 Number of people</td>
</tr>
<tr>
<td>UR12</td>
<td>Four average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 50% of cars produced in 1993. The top level of cars are those in the price range £13,000 to £30,000 at 1993 prices. Five average size adults shall be able to travel in comfort for a period of 3 hours. Two average size adults and 3 average size children shall be able to travel in comfort for a period of 3 hours. This could be accomplished with a three seat arrangement. Users shall have easy entry and exit.</td>
</tr>
</tbody>
</table>

[Image of a software interface showing requirements and a shareable edit mode]
DOORS – Integration with UML 2.0

- Linkable UML 2.0 diagrams and element objects, via the Analyst plug-in (Tau G2 UML 2.0 editor)
DOORS – Integration with URN

- Linkable URN diagrams and element objects, via the DXL export plug-in for jUCMNav
TWiki Overview

- A generic Wiki tool (TWiki.org)
  - Promotes collaboration
  - Database-driven
  - Access and version control
  - Forms and queries
  - State-based workflows (processes)
  - Text and graphics
  - Lightweight, extensible (plug-in architecture)

- Example of Forms and Queries
  - Requirements: http://cserg0.site.uottawa.ca/twiki/bin/view/ProjetSEG/UCMNavRequirements
  - Library: http://cserg0.site.uottawa.ca/twiki/bin/view/UCM/UCMVirtualLibrary
  - Use Cases: http://cserg0.site.uottawa.ca/seg/bin/view/CSI4900/UseCases
## Current Requirements

- Add a new requirement: [Req] Create

(Requirement name format is ReqNameOfRequirement)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Type</th>
<th>Importance</th>
<th>Priority</th>
<th>Status</th>
<th>Dependencies</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Req Action</strong></td>
<td><strong>Redo Many</strong></td>
<td>Functional</td>
<td>Optional</td>
<td>Low</td>
<td>Implemented</td>
<td><strong>Req Action Undo Many</strong></td>
<td>13 Mar 2005 - 19:58</td>
</tr>
<tr>
<td>Req Action</td>
<td><strong>Redo One</strong></td>
<td>Functional</td>
<td>Optional</td>
<td>Low</td>
<td>Implemented</td>
<td><strong>Req Action Undo One</strong></td>
<td>13 Mar 2005 - 20:01</td>
</tr>
<tr>
<td><strong>Req Action</strong></td>
<td><strong>Undo Many</strong></td>
<td>Functional</td>
<td>Optional</td>
<td>Medium</td>
<td>Implemented</td>
<td><strong>Req Action Undo One</strong></td>
<td>13 Mar 2005 - 20:01</td>
</tr>
<tr>
<td><strong>Req Action</strong></td>
<td><strong>Undo One</strong></td>
<td>Functional</td>
<td>Mandatory</td>
<td>Urgent</td>
<td>Implemented</td>
<td></td>
<td>13 Mar 2005 -</td>
</tr>
</tbody>
</table>
Twiki – Requirement Example

This is an important feature that may impact how the model is maintained. Does EMF help here?

-- DanielAmyot - 30 Jan 2005

When using GEF/EMF, every action performed creates a Command object. You create the Command and its inverse if you ever need to bring the model back to its original state. This command object is stored in some stack somewhere and the redo/undo functions are handled by the framework. Etienne's network editor already has this functionality. (I don't know how many commands are contained in the stack.

-- JasonKealey - 03 Feb 2005

<table>
<thead>
<tr>
<th>Req Name</th>
<th>Req Action Undo One</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>jUCMNav SHALL allow the user to undo the latest transformation or action on the UCM model.</td>
</tr>
<tr>
<td>Type</td>
<td>Functional</td>
</tr>
<tr>
<td>Importance</td>
<td>Mandatory</td>
</tr>
<tr>
<td>Priority</td>
<td>Urgent</td>
</tr>
<tr>
<td>Status</td>
<td>Implemented</td>
</tr>
<tr>
<td>Author</td>
<td>DanielAmyot</td>
</tr>
<tr>
<td>Dependencies</td>
<td></td>
</tr>
<tr>
<td>Verification</td>
<td></td>
</tr>
<tr>
<td>Approach</td>
<td></td>
</tr>
<tr>
<td>Test Cases</td>
<td>Test Undo Redo</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>Req Name</td>
<td>text</td>
</tr>
<tr>
<td>Description</td>
<td>textarea</td>
</tr>
<tr>
<td>Type</td>
<td>select</td>
</tr>
<tr>
<td>Importance</td>
<td>select</td>
</tr>
<tr>
<td>Priority</td>
<td>select</td>
</tr>
<tr>
<td>Status</td>
<td>select</td>
</tr>
<tr>
<td>Author</td>
<td>text</td>
</tr>
<tr>
<td>Dependencies</td>
<td>text</td>
</tr>
<tr>
<td>Verification Approach</td>
<td>text</td>
</tr>
<tr>
<td>Test Cases</td>
<td>text</td>
</tr>
</tbody>
</table>
Using TWiki…

• We have:
  • Requirement types description with configurable statuses & attributes
  • Bidirectional links (WikiWords)
  • Configurable requests, filtering, reports
  • Access control and version management (showing differences)
  • Change management (again with forms, process, etc.)
  • Discussions, attachment of documents/images
  • Export (HTML)
  • Scripting language (Perl)

• But do we really have:
  • Graphical view of traceability?
  • Editable tables (à la Excel/Word)?
  • Baselines? Tool integration? Imports? Analysis?
IBM Requisite Pro

- Keep your team on track
- 3 interfaces - work the way you want
- Document centric or database centric - your choice

Microsoft Word

Database

Web
✓ User defined requirement types
✓ User defined attributes
✓ User defined filters (views)
✓ Saved views
IBM Requisite Pro – Integration

- IBM Rational TestManager
- Testers view current state of requirements from their tool

- IBM Rational XDE and IBM Rational Rose, Rational Software Architect and Rational Software Modeler
- Developers view current state of requirements from their tool