Requirements Prioritization
Requirements Management
Requirements Traceability and Variability

Nicolas Sannier
* EDF R&D – STEP, 6 Quai Watier BP49
  78401 Chatou, France
  nicolas.sannier@edf.fr

** Inria, Campus Universitaire de Beaulieu,
  35042, Rennes Cedex, France
  nicolas.sannier@inria.fr
Where does this lecture come from?

- Make of work, experiences, slides, stuff borrowed from / inspired by / …
  - Orlena “Olly” Gotel
  - Scott Adams
  - Daniel Amyot
  - Benoit Baudry
  - Steve Easterbrook
  - Daniel Lucas-Hirtz
  - Gunter Mussbacher
  - And so many others
Summary from Last Session

• Key terms about RE Modeling, Verification and Validation
  – Modeling is one abstract perspective of the world
    • Different models for different tasks and moments …
      – Structural diagrams: modeling domains
      – Behavior diagrams: modeling interactions, modeling one system behavior
      – Goal modeling: refining from goals to requirements, detect and negotiate conflicts
      – NFR framework for non functional requirements
        » Making NFRs measurable, and analyzable
  – Verification and Validation is about evaluating whether:
    • We’re doing the right System/Specification/process … (validation)
    • We’re making the System/Specification/process… right (verification)
    • Is iterative all along the lifecycle and all along the different steps
    • Techniques: simple checks, reviews, model-based (formal) V&V …
What is the program for today?

• Introduction
• Requirements Negotiation and Prioritization
• Requirements Management
  – Requirements change!
  – Requirements Attributes
  – Change Management
• Requirements Traceability
  – What means traceability
  – Traceability Matrix
  – Computing Traceability using Natural Language Processing
• Requirements Variability
  – Introduction to Software/System Product Lines
  – Requirements and Feature Modeling
Dilbert on Changing Requirements

ENGINEERING LIAISON
TELL ME YOUR PROJECT STATUS AND I'LL TRANSLATE FOR OUR CLIENTS.

THE PROJECT WILL NEVER BE COMPLETED BECAUSE OUR IDIOT CLIENTS CHANGE THE REQUIREMENTS EVERY OTHER DAY.

I'LL JUST SAY YOU'RE DRUNK.

THE PROJECT REQUIREMENTS ARE FORMING IN MY MIND.

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

• Too many requirements for too less resources
  – Establishing priorities

• “Changing requirements is as certain as death and taxes”
  – Standish group CHAOS Reports

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

• Priorities in requirements lead to a need for requirements management
• Changes in requirements lead to a need for requirements management
  – Versioning, traceability, …
REQUIREMENTS NEGOTIATION AND PRIORITIZATION
Conflicts on Requirements

- Possible conflicts to be resolved among stakeholders
  - Between supplier and customers about costs, benefits, risks
  - Power struggle within customer organization
  - Conflicts with other projects about resources
  - Conflicting goals, features, requirements
  - ...

- In practice, causes are frequently intermixed.

- Potential resolution strategies:
  - Agreement
  - Compromise
  - Voting
  - Definition of variants
  - Mediation
  - Arbitration
  - Goal analysis (think GORE, KAOS, GRL!)
Conflicts and Negotiations

• Conflict resolution hence often involves *negotiation*
  – Negotiating a coherent set of requirements is not easy
  – But it is one task of the requirements analyst
  – Difficult to satisfy everyone, to achieve all goals…

• First, detect when requirements are inconsistent (think goal modeling)
• Then, convince all stakeholders to understand the point of view of each other
  – Have each party explain what they believe the *other party* wants and why

• Reach an *agreement* on a *coherent set* of requirements that meets the needs of as many stakeholders as possible
  – Analyze each party’s goals, find solutions that do not conflict but ideally support everybody’s goals
Priorities

- Different stakeholders have different goals and different priorities
- Some stakeholders’ decisions carry more weight than others

- Companies often lack systematic data, metrics, and technologies to support the prioritization process
  - Often done manually, informally, on an ad-hoc basis
  - Difficult to establish and communicate

- Attitude!
  - "No need for priorities, we can do everything in the specification!“
    - Yes, but when and at what cost?
  - Suddenly, when the deadline is fast approaching, some requirements are put aside in order to deliver something on time...
Priorities: The 20-80 rule

• 20% of functionalities provide 80% of revenues
  – Think of MS Word…
• 80% of users use only 20% of the system
• The remaining 80% of functionalities offer a lower return on investment while adding delays, development costs, maintenance costs...

• How to find the most useful and beneficial 20% of functionalities?
Dilbert on Negotiating and Setting Priorities

I saved a fortune by personally negotiating the contract for our new ERP system.

You bought outdated hardware and forgot several components that are required.

And I like software with my hardware, but that's just me.

As you requested, I sorted the future product features into three priorities.

Let me know which group I should stop working on.

This is the part where you pretend to add value.

Essential critical must-have.
You said Requirements Prioritization?

- Requirements prioritization is also referred to as **triage**
- Need to decide which requirements really matter or on those that need to be implemented in the current release
- Need for compromise, negotiation, priorities
- Prioritization is needed because there will almost always be the need for trade-offs (e.g., required functionality vs. time and resources)

- Must help:
  - Make **acceptable** tradeoffs among goals of value, cost, time-to-market
  - Allocate resources based on importance of requirements to the project as a whole (project planning)
  - Determine when a requirements should become part of the product
  - Offer the right product!
Criteria for Requirements Triage

- Must be simple and fast, for industry adoption
- Must yield accurate and trustworthy results
- Must consider issues such as
  - The value of requirements to stakeholder (maximize)
  - The cost of implementation (minimize)
  - Time to market (to minimize)

- Important to agree on requirements granularity
  - E.g., use cases, features, detailed functional requirements
Requirements Categories

- Determine criteria, granularity, scale dimensions
- Frequently used:
  - Urgency
    - High (mission critical requirement; required for next release)
    - Medium (supports necessary system operations; but could wait if necessary)
    - Low (a functional or quality enhancement; would be nice to have someday)
  - Importance
    - Essential (product unacceptable unless these requirements are satisfied)
    - Conditional (would enhance the product, but the product is acceptable)
    - Optional (functions that may or may not be worthwhile)

You are in big big big trouble!
Make reasonable choices and triage

Mandatory: 98
Strongly wished: 15
Optional: 7

LET ME KNOW WHICH GROUP I SHOULD STOP WORKING ON.

ESSENTIAL CRITICAL MUST-HAVE

You are in big big big trouble!
Make reasonable choices and triage
Requirements Ranking

• Rank requirements from 1 to N
• In theory, the best way to decide
• In practice, it’s quickly unfeasible
  – May work for few requirements
  – Impossible for large projects
  – Lots of requirements interdependencies

In theory, practice and theory are similar…
But in practice they are different 😊
Priority: Evaluate Trade-off Value vs Cost

- High Priority
- Average Priority
- To avoid!
Evaluate Values and Costs

- Calculate return on investment by
  - Assessing the value of each requirement
    - In terms of benefits or penalties whether present or missing
  - Assessing the cost of each requirement
    - Time to develop, resources needed, …
  - Calculating the cost-value trade-offs

- Difficulties:
  - Hard to calculate absolute value/cost
  - Relative value/cost figures are easier to obtain
  - Interdependent requirements difficult to treat individually
  - Inconsistencies or conflicts in priorities assigned by individual stakeholders
Wieger’s Prioritization
K. Wiegers, “First Things First”

• Semi-quantitative analytical approach to requirements prioritization based on value, cost, and risk
• Relies on estimation of relative priorities of requirements
  – Dimensions
    • Relative benefit (for having requirement)
    • Relative penalty to stakeholder (if requirement is not included)
    • Relative cost (to implement requirement)
    • Relative risk (technical and other risks)
  – Each dimension is given a value on a given scale (e.g., 0..9)
  – Dimensions have relative weights
• Formula used to derive overall priority
  – priority = (value%) / ((cost% * cost weight) + (risk% * risk weight))
• Still limited by ability to properly estimate
  – Requires adaptation and calibration

Value% (R_i) = Benefit(R_i) × WeightBenefit + Detriment(R_i) × WeightDetriment
Priority(R_i) = Value% (R_i) / (Cost%(R_i) × WeightCosts + Risk% (R_i) × WeightRisk)
# Wieger’s Prioritization

K. Wiegner, “First Things First”

<table>
<thead>
<tr>
<th>Feature</th>
<th>2</th>
<th>1</th>
<th>Total Value</th>
<th>Value %</th>
<th>Cost %</th>
<th>Risk %</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Query status of a vendor order</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>8.4</td>
<td>4.8</td>
<td>1</td>
<td>1.345</td>
</tr>
<tr>
<td>2. Generate a Chemical Stockroom inventory report</td>
<td>9</td>
<td>7</td>
<td>25</td>
<td>16.2</td>
<td>11.9</td>
<td>3</td>
<td>0.987</td>
</tr>
<tr>
<td>3. See history of a specific chemical container</td>
<td>5</td>
<td>5</td>
<td>15</td>
<td>9.7</td>
<td>7.1</td>
<td>2</td>
<td>0.957</td>
</tr>
<tr>
<td>4. Print a chemical safety datasheet</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3.2</td>
<td>2.4</td>
<td>1</td>
<td>0.833</td>
</tr>
<tr>
<td>5. Maintain a list of hazardous chemicals</td>
<td>4</td>
<td>9</td>
<td>17</td>
<td>11.0</td>
<td>9.5</td>
<td>4</td>
<td>0.708</td>
</tr>
<tr>
<td>6. Modify a pending chemical request</td>
<td>4</td>
<td>3</td>
<td>11</td>
<td>7.1</td>
<td>7.1</td>
<td>2</td>
<td>0.702</td>
</tr>
<tr>
<td>7. Generate an individual laboratory inventory report</td>
<td>6</td>
<td>2</td>
<td>14</td>
<td>9.1</td>
<td>9.5</td>
<td>3</td>
<td>0.646</td>
</tr>
<tr>
<td>8. Search vendor catalogs for a specific chemical</td>
<td>9</td>
<td>8</td>
<td>26</td>
<td>16.9</td>
<td>16.7</td>
<td>8</td>
<td>0.586</td>
</tr>
<tr>
<td>9. Check training database for hazardous chemical training record</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>6.5</td>
<td>9.5</td>
<td>2</td>
<td>0.517</td>
</tr>
<tr>
<td>10. Import chemical structures from structure drawing tools</td>
<td>7</td>
<td>4</td>
<td>18</td>
<td>11.7</td>
<td>21.4</td>
<td>7</td>
<td>0.365</td>
</tr>
</tbody>
</table>

| Totals                                       | 54  | 46  | 154         | 100     | 100    | 33     | 100      |
## Volere Prioritisation Spreadsheet

**Copyright © The Atlantic Systems Guild 2002**

<table>
<thead>
<tr>
<th>Requirement/Product Use Case/Feature</th>
<th>Number</th>
<th>Factor - score out of 10</th>
<th>%Weight applied</th>
<th>Factor - score out of 10</th>
<th>%Weight applied</th>
<th>Factor - score out of 10</th>
<th>%Weight applied</th>
<th>Factor - score out of 10</th>
<th>%Weight applied</th>
<th>Total Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Value to Customer</td>
<td>40</td>
<td>Value to Business</td>
<td>20</td>
<td>Minimise Implementation Cost</td>
<td>10</td>
<td>Ease of Implementation</td>
<td>30</td>
<td>Priority Rating</td>
</tr>
<tr>
<td>Requirement 1</td>
<td>1</td>
<td>2</td>
<td>0.8</td>
<td>7</td>
<td>1.4</td>
<td>3</td>
<td>0.3</td>
<td>8</td>
<td>2.4</td>
<td>4.9</td>
</tr>
<tr>
<td>Requirement 2</td>
<td>2</td>
<td>2</td>
<td>0.8</td>
<td>8</td>
<td>1.6</td>
<td>5</td>
<td>0.5</td>
<td>7</td>
<td>2.1</td>
<td>5</td>
</tr>
<tr>
<td>Requirement 3</td>
<td>3</td>
<td>7</td>
<td>2.8</td>
<td>3</td>
<td>0.6</td>
<td>7</td>
<td>0.7</td>
<td>4</td>
<td>1.2</td>
<td>5.3</td>
</tr>
<tr>
<td>Requirement 4</td>
<td>4</td>
<td>6</td>
<td>2.4</td>
<td>8</td>
<td>1.6</td>
<td>3</td>
<td>0.3</td>
<td>5</td>
<td>1.5</td>
<td>5.8</td>
</tr>
<tr>
<td>Requirement 5</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0.1</td>
<td>3</td>
<td>0.9</td>
<td>4</td>
</tr>
<tr>
<td>Requirement 6</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>1.2</td>
<td>6</td>
<td>0.6</td>
<td>5</td>
<td>1.5</td>
<td>6.9</td>
</tr>
<tr>
<td>Requirement 7</td>
<td>7</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>0.6</td>
<td>6</td>
<td>0.6</td>
<td>7</td>
<td>2.1</td>
<td>4.9</td>
</tr>
<tr>
<td>Requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Pairwise Prioritization, AHP, …

• Finding scores and weights is difficult and subjective
• Potential solution: pairwise comparison
  – Which requirements (A or B) is more important:  A  <<  <  =  >  >>  B
• Benefits
  – Indicates what is important to the client
  – Identifies requirements of high value and low cost (priority!)
  – Identifies requirements of low value and high cost (likely to be removed)
  – Has already been used to assist numerous corporate and government decision makers
• New problems
  – Large number of pairs – pairwise comparison can be tedious
    • Solved using transitivity and other tricks!
    • Mathematical optimization of the number of pairs to be considered
  – Many dependencies between requirements
    • Can actually be used to further reduce the # of pairs
  – Complexity handled by grouping requirements per features → Fewer pairs to check
## Pairwise Prioritization, AHP, ...

<table>
<thead>
<tr>
<th>Level</th>
<th>Order of magnitude</th>
<th>Managing a complete set of interdependencies...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small-Scale</td>
<td>~10 reqs</td>
<td>requires small effort.</td>
</tr>
<tr>
<td>Medium-Scale</td>
<td>~100 reqs</td>
<td>is feasible but requires large effort.</td>
</tr>
<tr>
<td>Large-Scale</td>
<td>~1000 reqs</td>
<td>is practically unfeasible, but feasible among small bundles of requirements.</td>
</tr>
<tr>
<td>Very Large-Scale</td>
<td>~10000 reqs</td>
<td>among small bundles of requirements is unfeasible in practice.</td>
</tr>
</tbody>
</table>

[Regnell et al, REFSQ2008]
Analytic Hierarchy Process (AHP)
Developed by Karlsson and Ryan (1997) based on work by Saaty (early 1970)
see also http://en.wikipedia.org/wiki/Analytic_Hierarchy_Process

• Use cost-value diagrams to analyze and discuss candidate requirements
• Useful for requirements triage and release planning (but also applicable in many other situations where complex decisions are to be made)

• Basic procedure for rating a set of criteria
  – Develop pairwise comparison matrix of each criterion
  – Normalize the matrix
  – Average the value of each row to get corresponding rating
• Criterion ratings are then used to evaluate different potential decisions

<table>
<thead>
<tr>
<th>RATING</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Equally preferred</td>
</tr>
<tr>
<td>3</td>
<td>Moderately preferred</td>
</tr>
<tr>
<td>5</td>
<td>Strongly preferred</td>
</tr>
<tr>
<td>7</td>
<td>Very strongly preferred</td>
</tr>
<tr>
<td>9</td>
<td>Extremely strongly preferred</td>
</tr>
</tbody>
</table>
Pairwise Prioritization, AHP, …

Example

• Suppose two criteria, cost and quality, for product A & B
  – The cost for A is $60 and the quality is above average.
  – The cost for B is $15 and the quality is right at average.
• Which product do you choose?

<table>
<thead>
<tr>
<th>COST</th>
<th>QUALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>A 1</td>
<td>1/7</td>
</tr>
<tr>
<td>B 7</td>
<td>1</td>
</tr>
</tbody>
</table>

B very strongly preferred

Reverse of preference

• The matrix describes:
  – the price of B is very strongly preferred over A
  – A is only moderately preferred over B
Pairwise Prioritization, AHP, …
Example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1/3</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1/2</td>
<td>5</td>
</tr>
</tbody>
</table>

Normalize the matrix
First add up all the values in each column

Next the values in each column are divided by the corresponding column sums

Note: the values in each column add up to 1
Pairwise Prioritization, AHP, …

Example

Average the value of each row to get corresponding scores

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Row Average</th>
</tr>
</thead>
</table>

A has highest rank
Pairwise Prioritization, AHP, …

Example

<table>
<thead>
<tr>
<th>Req</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Value</strong></td>
<td>12%</td>
<td>6%</td>
<td>2%</td>
<td>3%</td>
<td>27%</td>
<td>9%</td>
<td>1%</td>
<td>17%</td>
<td>11%</td>
<td>12%</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>20%</td>
<td>2%</td>
<td>1%</td>
<td>5%</td>
<td>20%</td>
<td>13%</td>
<td>2%</td>
<td>9%</td>
<td>13%</td>
<td>15%</td>
</tr>
</tbody>
</table>
Pairwise Prioritization, AHP, …
With Multiple Different Stakeholders

• Each client is unique!
• Each stakeholder group may have a different weight
  – Process uses a weighting criteria to consider each individual stakeholder group

• Example (stakeholders M1 to M10 are different markets)
  – Revenue last release
  – Profit last release
  – Number of sold licenses last release
  – Predictions of the above criteria for the coming release
  – Number of contracts lost to competitors
  – Number of potential customer with nil licenses to date
  – Size of total market segment
  – Growth potential
  – …
AHP, …
With Multiple Different and weighted Stakeholders

Before adjustment based on stakeholders importance

After adjustment based on stakeholders importance
REQUIREMENTS MANAGEMENT
Things change! And so Are Requirements

- “If you can see change not as an enemy, but as a welcome friend, you will secure the most valuable prize of all – the future…” - Unknown
- “If you are not riding the wave of change, you will find yourself beneath it.” – Unknown
- Carrying heavy loads and historical human progress
Requirements Change Factors

• Requirements errors, conflicts, and inconsistencies
  – May be detected at any phase (analysis, specification, validation, development)

• 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

• 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
  – May be caused by organizational changes
  – Laws and regulations may change
Requirements Change
Increasing and Evolving International Nuclear Standards

- New technologies appearing
  - From hardwired logic to software systems
  - From nuclear specific systems to general purpose Customs-off-the-shelf (COTS)
- Safety classification evolution
  - Important to Safety / Not Important to Safety
    - 1E / not 1E – A, B, C, NC …
Requirements Change
Increasing US Healthcare Regulations

- HIPAA
- CCHIT
- HITECH (ARRA)
- MU Stage 1
- MU Stage 2


Additional Various State Laws

... and each of these laws evolves too (multiple versions!)

Health Insurance Portability and Accountability Act
Certification Commission for Health Information Technology
Health Information Technology for Economic and Clinical Health Act
American Recovery and Reinvestment Act (ARRA)
Meaningful use
Requirements Change
Feature Survival Chart  C. Wnuk, B. Regnell and L. Karlsson (REV’2008)

- Evolution of requirements all along the project’s life
- Green: in-scope features
  - Light green: primary
  - Dark green: secondary
- Red: out of scope feature
- Grey: undecided
Why Do Requirements Change?

Reasons for scope exclusion

- Stakeholder business decision
- Lack of resources
- Portfolio changes
- Replaced or renamed
- Already implemented
- Unclear feature or requirements
- Dependent of supplier support

Percentages

Reasons for descope
Impacts of 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
  - Pretty hard with small time-to-market
You Said 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.
A Few Words on Baseline Requirements

• Baseline: non-modifiable (read-only) version of a document
  – Describes a moment in time
  – May include multiple documents at the same time
    – Is the shared vision upon the related concern
• 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
A Few Words on Baseline 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
Requirements Management Activities

- 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**

### 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
Objectives of Requirements Management

- **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
- **Impact assessments** of proposed changes
  - Analysis tools let you see which other linked artifacts will be affected by a change
- **Controlled access** to current project information
  - A shared database ensures that all users are working with current data
  - A central repository allows all users to see the information that they need to see
- **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
Identifying 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
  – Dynamic renumbering
    • Some word processing systems allow for automatic renumbering of paragraphs and the inclusion of cross references
  – 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)
Requirements Also 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

Example: Requirement 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

- Question!!!
  - Find a suitable modeling approach to describe requirements statuses lifecycle
Change Management

• 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...
Change Management Process

- Once 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)
customers may misunderstand requirements and their context and suggest unnecessary changes

- Rejected request

with the help of traceability information

- Risk is too high

- Rejected request

- Accepted change

- Rejected request

- Rejected request

- Accepted change

consequential changes may be unacceptable to user/customer

negotiations with customers

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

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

• Configuration Management
  – Versioning, labelling, and tracking during the development cycle of software

• Release Management
  – Defines how and when different hardware and software will be made available together as a product

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

• Tool facilities may include
  – Electronic change request forms
  – A database to store and manage requests
  – A change model which may be instantiated (responsibility and process)
  – Electronic signatures, notifications, Discussion forums
REQUIREMENTS TRACEABILITY
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)”
O. Gotel and A. Finklestein, 1994

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
IEEE 830, 1998

Traceability is often mandated by contracts and standards
– E.g., nuclear energy, military, automotive (recently), and aerospace
There is Traceability Everywhere
Requirements Traceability
Why is it Important?

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

- Benefits of traceability
  - 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)
  - Risk reduction (e.g., if a team member with key knowledge leaves)
  - Supports the verification process (certification, localization of defects and conflicts…)
Requirements Traceability
Why is it Important?

• Verification and Validation
  – assessing adequacy of test suite
  – assessing conformance to requirements
  – assessing completeness, consistency, impact analysis
  – assessing over- and under-design
  – investigating high level behavior impact on detailed specifications
  – detecting requirements conflicts
  – checking consistency of decision making across the lifecycle

• Maintenance
  – Assessing change requests
  – Tracing design rationale

• Document access
  – ability to find information quickly in large documents

• Process visibility
  – ability to see how the software was developed
  – provides an audit trail

• Management
  – change management
  – risk management
  – control of the development process
Requirements Traceability
Why is it Difficult?

• Cost
  – very little automated support
  – Manual creation of links is very demanding
    • Likely the most annoying problem
  – Specialized tools must be used
  – full traceability is very expensive and time-consuming

• Delayed gratification
  – the people defining traceability links are not the people who benefit from it
    • development vs. V&V
  – much of the benefit comes late in the lifecycle
    • testing, integration, maintenance

• Size and diversity
  – Huge range of different document types, tools, decisions, responsibilities,…
  – Various stakeholders require different information
  – No common schema exists for classifying and cataloging these
  – In practice, traceability concentrates only on baselined requirements

Beware:
• Trace what deserve to (or shall) be traced;
• Define the traceability objectives;
• Trade-off cost/benefits
Traceability Current State of Practice

• Coverage:
  – links from requirements forward to designs, code, test cases,
  – links back from designs, code, test cases to requirements
  – links between requirements at different levels

• Traceability process
  – Assign each sentence or paragraph a unique id number
  – Manually identify linkages
  – Use manual tables to record linkages in a document
  – Use a traceability tool (database) for project wide traceability

• Tool then offers ability to
  – follow links
  – find missing links
  – measure overall traceability
Backward and Forward Traceability

- Backward traceability (why ?)
  - To previous stages of development
  - Depends upon each element explicitly referencing its source in earlier documents

- Forward traceability (how ?)
  - To all documents spawned by a document
  - Depends upon each element in the document having a unique name or reference number
Different Kinds of Traceability Links

- Requirements – source traceability
- Requirements – rationale traceability
- Requirements – requirements traceability
  - Links with other requirements which are dependent on them
- Requirements – architecture traceability
  - Links with the subsystems where these requirements are implemented
- Requirements – design traceability
  - Links with specific hardware or software components
- Requirements – interface traceability
  - Links with the interfaces of external systems
- Requirements – feature traceability
- Requirements – tests traceability
  - Links with test cases verifying them
- Requirements – code traceability
  - Generally not directly established, but can be inferred
How to Trace?
Traceability Tables

- 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></td>
<td></td>
<td></td>
<td></td>
<td>search.8</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</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>search.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>search.14</td>
</tr>
</tbody>
</table>
How to Trace?
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>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>
</tr>
<tr>
<td>R2</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>
</tr>
<tr>
<td>R4</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>
</tr>
<tr>
<td>R6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source artefacts</th>
<th>Goal 1</th>
<th>Goal 2</th>
<th>Goal 3</th>
<th>Goal 4</th>
<th>Goal 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>R2</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>
</tr>
<tr>
<td>R4</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>
</tr>
<tr>
<td>R6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Target artefacts</th>
<th>Goal 1</th>
<th>Goal 2</th>
<th>Goal 3</th>
<th>Goal 4</th>
<th>Goal 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>satisfies</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario 2</td>
<td></td>
<td>based_on</td>
<td>conflicts</td>
<td>satisfies</td>
<td></td>
</tr>
<tr>
<td>Scenario 3</td>
<td></td>
<td></td>
<td>satisfies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario 4</td>
<td>conflicts</td>
<td></td>
<td>satisfies</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scenario 5</td>
<td></td>
<td></td>
<td>satisfies</td>
<td></td>
<td>based_on</td>
</tr>
</tbody>
</table>
How to Trace?
Traceability in Plain Text / Lists

• Textual (hypertext) Reference
  – Not very robust against changes

• 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>
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)
  – …
Modeling Traceability
Traceability Cardinality

• 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
Computing Traceability
(Cleland et al., 2010 - Gotel & Morris, 2011 - Sannier & Baudry, 2012, Jeanneret et al., 2011)

• Three dimensions
  – Artifact to track (requirements, concepts, …)
  – Artifact signature: how can we recognize it (signs, fingerprints, signature elements)
  – Artifact tracks: signature elements lefts in the domain under study

• One approach: Indexing and Information Retrieval (searching for signature elements)

• From Requirements to code
  – Extract meaningful terms in requirements and retrieve them in the classes signatures

• Between different documents
  – Wrapping requirements using Topic detection algorithms, Clustering Algorithms
Computing Traceability
(Cleland et al., 2010 - Gotel & Morris, 2011 - Sannier & Baudry, 2012, Jeanneret et al., 2011)

• Lots of false positive results
  – Approaches are conservatives
  – Better safe than sorry (must have all positives to maintain traceability)

• The larger the reference corpus is:
  – The more information is scattered
  – The more the system provides false positives
  – The more information misses precision (from 15 to 30%)

• Research objectives
  – Providing smaller (but still complete) solution research space
    • Clustering hypothesis (good (resp. bad) candidates tend to get close to other
ten good (resp. bad) ones)
    • Modeling and Typing to remove noisy elements
REQUIREMENTS VARIABILITY AND SOFTWARE/SYSTEM PRODUCT LINES
Introduction: variability = Complexity

33 optional independent features

a unique variant for every person on this planet
Introduction: variability = Complexity

320 optional independent features

more variants than estimated atoms in the universe
Feature Models:
A De Facto Standard for Variability Modeling

• Central to many software product line approaches
  – More than 1000 citations of [Kang et al., 1990] per year
  – Generative programming [Czarnecki et al., 2000]
• Research effort
  – Formal Semantics [Schobbens et al., 2007]
  – Automated Reasoning Techniques [Batory et al., 2005], [Czarnecki et al., 2007],
    [Mendonca et al., 2009], [Thuem et al., 2009], [Janota et al., 2010], [Benavides et al., 2010]
• Tools
  – Commercial: pure::variants, Gears
  – Academic: fmp, FeatureIDE, FaMa, SPLOT, TVL, etc.
You Said Feature Modeling?

• … about identifying
  – common features of concepts
  – variable features of concepts
  – and their dependencies
• and documenting them in a coherent model, called a feature model
• Feature modeling is a core activity of important domain-engineering methods
• Features are a convenient way to think about commonalities and differences between family members
  – e.g. "MP3 player" – some phones have it, some don't
• Features are usually coarse-grained abstractions
  – Abstract from detailed requirements
  – Abstract from design/implementation details
• An old but still vague notion
You said feature?

• [Kang et al.] “a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems”

• [Kang et al.] “distinctively identifiable functional abstractions that must be implemented, tested, delivered, and maintained”

• [Eisenecker and Czarnecki]. “anything users or client programs might want to control about a concept”

• [Bosch et al.] “A logical unit of behavior specified by a set of functional and non-functional requirements.”

• [Chen et al.] “a product characteristic from user or customer views, which essentially consists of a cohesive set of individual requirements”

• [Batory] “an elaboration or augmentation of an entity(s) that introduces a new service, capability or relationship”
What’s IN a feature?

• Confusion as to what a feature generally represents (i.e., black box view)
• Redefine “Feature” as a 3-tuple of requirements, domain assumptions and specifications
  – Requirements (R)
    • The purpose of the system (optative): “The SnakeEye System shall notify the police of the presence of burglars”
  – Domain assumptions (W)
    • The given behavior of the environment (indicative): ‘Burglars cause movement when they break in’ and ‘There is a movement sensor monitoring the house, connected to the system’
  – Specifications (S)
    • What the system should do to satisfy R given W holds (optative): ‘Alert the police if the sensor captures movement’
Feature Models

- Describe the common and variable features (characteristics) of a system under study
- **Hierarchy:** rooted tree
- **Variability:** mandatory, optional, Groups: exclusive or inclusive features, Cross-tree constraints

- A feature model is a synthetic and complete hierarchy of features with variability
  - The primary purpose of the hierarchy is to organize features into multiple levels of increasing detail
- Variability defines what the allowed combinations of features are.
  - Variability in a feature model is expressed through different mechanisms.
  - describes a set of valid feature combinations.
Feature Models

- Represents all the possible configurations
- Hierarchy + Variability = set of configurations

![Feature Model Diagram]

- Medical Image
  - Modality Acquisition
    - MRI
    - PET
  - Format
    - DICOM
    - Nifti
  - Anonymized

T1 or T2 excludes Anonymized
DICOM implies Anonymized
PET or Anonymized

Optional
Mandatory
Xor-Group
Or-Group
Feature Models

- Represents all the possible configurations
- Hierarchy + Variability = set of (UN)VALID configurations (tools that can reason on it)

Question: Is this configuration valid?

Around 30 reasoning operations on Features
Satisfiability (NP-complete)
Configuration Checking
Dead features detection

[Schobbens et al., 2007]  [Benavides et al., 2010]