TODAY IS THE THREE-YEAR ANNIVERSARY OF OUR FIRST MEETING TO DISCUSS PROJECT REQUIREMENTS.

AND WE'RE STILL DISCUSSING REQUIREMENTS. DOES ANYONE ELSE SEE A PROBLEM HERE?

WHEN YOU'RE DONE, CAN WE TALK ABOUT REQUIREMENTS?
Organising Requirements
Viewpoints

- **Interactor viewpoints:** people or other systems that interact directly with the system
  
  - e.g. ATM customer, the bank’s database

- **Indirect viewpoints:** stakeholders who do not use the system, but influence reqs
  
  - e.g. bank management and security people

- **Domain viewpoints:** constraints of the domain that influence reqs
  
  - e.g. standards for inter-bank communication
Use Cases

- It is a sequence of steps that an actor performs in order to perform some function within the system.
- An actor is a role played by a user or any other system that interacts with the system.
- A powerful technique for identifying requirements.
- One of the foundation techniques used in Object-oriented analysis methodologies such as UML (Unified Modeling language), Objectory and Booch OOA/OOD.
Use Case Example

Source: http://epf.eclipse.org/wikis/
Structured Documents

- helps to ensure that all relevant questions and areas have been addressed

- Example:
  - Volere Requirements Shell
Volere

- Volere (Voh-lair-ray) the Italian verb to want, or to wish

- Volere is the umbrella that covers the collection of requirements templates, processes, books, consulting and training

- www.volere.co.uk
Volere Requirements Shell

The requirements shell is a guide to writing a requirement

<table>
<thead>
<tr>
<th>Requirement #: Unique Id</th>
<th>Requirement Type</th>
<th>Event/use case #:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Description:</th>
<th>A one sentence statement of the intention of the requirement</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Rationale:</th>
<th>A justification of the requirement</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Source:</th>
<th>Who raised this requirement?</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Fit Criterion:</th>
<th>A measurement of the requirement such that it is possible to test if the solution matches the original requirement</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Customer Satisfaction:</th>
<th>Customer Dissatisfaction:</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Dependencies:</th>
<th>A list of other requirements that have some dependency on this one</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supporting Materials:</th>
<th>History: Creation, changes, deletions, etc.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Degree of stakeholder happiness if this requirement is successfully implemented. Scale from 1 = uninterested to 5 = extremely pleased.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Measure of stakeholder unhappiness if this requirement is not part of the final product. Scale from 1 = hardly matters to 5 = extremely displeased.</th>
</tr>
</thead>
</table>

**List of events/use cases that need this requirement**
Volere Shell Example

Requirement #: 75  Requirement Type: 9  Event/BUC/PUC #: 7, 9

Description: The product shall record all the roads that have been treated

Rationale: To be able to schedule untreated roads and highlight potential danger

Originator: Arnold Snow - Chief Engineer

Fit Criterion: The recorded treated roads shall agree with the drivers’ road treatment logs and shall be up to date within 30 minutes of the completion of the road’s treatment

Customer Satisfaction: 3  Customer Dissatisfaction: 5

Dependencies: All requirements using road and scheduling data

Supporting Materials: Work context diagram, terms definitions in section 5

History: Created February 29, 2010
Volere Categories

PROJECT DRIVERS
1. The Purpose of the Product
2. Client, Customer and other Stakeholders
3. Users of the Product

PROJECT CONSTRAINTS
4. Mandated Constraints
5. Naming Conventions and Definitions
6. Relevant Facts and Assumptions

FUNCTIONAL REQUIREMENTS
7. The Scope of the Work
8. The Scope of the Product
9. Functional and Data Requirements

NON-FUNCTIONAL REQUIREMENTS
10. Look and Feel Requirements
11. Usability Requirements
12. Performance Requirements
13. Operational Requirements
14. Maintainability and Portability Requirements
15. Security Requirements
16. Cultural and Political Requirements
17. Legal Requirements

PROJECT ISSUES
18. Open Issues
19. Off-the-Shelf Solutions
20. New Problems
21. Tasks
22. Cutover
23. Risks
24. Costs
25. User Documentation and Training
26. Waiting Room
27. Ideas for Solutions

Categorising Requirements

- **Categorising** requirements is a **non-trivial** task
- **MOBIlearn** project adapted **database** version of **Volere** shell and template
- DB allows flexible and ad-hoc categorisations of requirements

**ABSTRACT**: This work reports on our experience of eliciting and managing requirements on a large European-based multinational project, whose purpose is to create a system to support learning using mobile technology. The project used an adapted database version of the **Volere shell and template**. We provide details about the project, describe the Volere tools, and explain how and why we used a flexible categorisation scheme to manage the requirements. Finally, we discuss three **lessons learned**: (1) provide a flexible mechanism for organising requirements, (2) plan ahead for the RE process, and (3) use the 'waiting room'.
Detecting Conflicts
Use Order of Priority

- Determines the degree of importance of each requirement to the customer.
- There may not be enough time or resources to implement all requirements, so the most critical should be implemented first.
- Helps to identify conflicting requirements.
- Can help you plan successive releases of a product by identifying which requirements should be done first, and which should be left to successive releases.
Formal Methods

- Construct a mathematical model of the requirements
- Use logical analysis to verify properties and identify inconsistencies
- Most methods have tool support and some have automatic analysis
- Popular models include 1st order logic, set theory (eg. Z), temporal logic, state machines
Resolving Conflicts
Boehm’s Win-Win Spiral Model

Source: Using the WinWin Spiral Model: A Case Study (Boehm et al. 1998)
Win-Win Spiral Model Benefits

- **Flexibility.** Teams can adapt to risks and uncertainties, such as a rapid project schedule and changing team composition.

- **Trust enhancement.** The model provided a means for growing trust among the project stakeholders, enabling them to evolve from adversarial to supportive and cooperative.

- **Customer wins** by getting the system that satisfies most of their needs.

- **Developer wins** by working on a realistic and achievable system (budgets and deadlines).
Feasibility Studies

**INPUT**
- set of preliminary business requirements
- an outline description of the system
- how the system is intended to support business processes

**OUTPUT**
- a report recommending whether or not it is worth carrying on with the requirements engineering and system development process
Questions for the FS

- Does the system *contribute* to the overall *objectives of the organisation*?
- Can the system be *implemented* using *current technology* and within given *cost* and *schedule* constraints?
- Can the system be *integrated* with other systems already in place?
More questions

- **How** would the organisation cope if this system were **not implemented**?
- **What** are the **problems with current processes**?
- **How** would the **new system alleviate** these problems?
- **Does** the system require **technology** that has **not previously** been used in the organisation?
- **What** must be **supported** by the system? **What need not be supported** by the system?
Requirements
Evolution
Issues

- Issues
  - requirements inevitably change, but why and to what effect?

- Definitions
  - a classification of requirements according to the types of change which may occur

- Techniques
  - Software configuration management (SCM), traceability tables, good record keeping, metrics
Reasons for reqs change

- User gains better understanding of the requirements from the requirements elicitation, analysis and validation process
- New ways of working result from the introduction of the SW system itself
- Changes to the environment of the organisation
- Changes to systems or processes within an organisation
Consequences of reqs change

- Depends when in the life cycle the requirements change
- Better early: review requirements specification
- Worse late: design, implementation, tests and documentation

- Modular design can minimise changes
Two Classes of Requirements

- **Enduring Requirements**
  - Derive from an organisation’s core activity
  - Relate directly to the problem domain
  - Relatively stable

- **Volatile Requirements**
  - Derive from the environment of the system
  - Likely to change during development or afterwards
Classes of Volatile Requirements

- Emergent
  - from improved understanding of the problem

- Consequential
  - as a result of using the delivered system

- Mutable
  - from changes to the environment of the organisation

- Compatibility
  - from changes to processes within the organisation
Traceability Table (Matrix)

- Uniquely number all requirements
- Identify specific aspects of the system or its environment classified by, for example,
  - **Features**: important customer observable system/product features
  - **Source**: of each requirement
  - **Dependency**: how requirements are related to one another
  - **Subsystems**: governed by a requirement
  - **Interface**: relation to internal and external interfaces
## Traceability Table Examples

<table>
<thead>
<tr>
<th>Feature</th>
<th>Requirement 1</th>
<th>Requirement 2</th>
<th>...</th>
<th>Requirement n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature 1</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature 2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature 3</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Feature m</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A Database of Requirements

- Manage requirements as a live repository
- Manage traceability tables
- Record rationale (reasons for a requirement)
- Record sources (where req. comes from)
- Record rejected requirements
- Identify volatile requirements (so they can be traced later)