Requirements
Organisation, Analysis and Evolution

Software Requirements and Design
CITS 4401
Lecture 20

Organising Requirements

Viewpoints

- **Interactor viewpoints:** people or other systems that interact directly with the system
- **Indirect viewpoints:** stakeholders who do not use the system, but influence reqs
- **Domain viewpoints:** constraints of the domain that influence reqs
- e.g. ATM customer, the bank’s database
- e.g. bank management and security people
- e.g. standards for inter-bank communication
Use Cases

- 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.
- A user scenario is a sequence of events that a user performs in order to perform some function within the system.

Categorising Requirements

- Classifying and categorising requirements is a non-trivial task
- The Mobilearn project adapted database version of Volere shell and template
- DB allows flexible and ad-hoc categorisations of requirements

Mobilearn Requirements Lessons Learned

1. provide a flexible mechanism for organising requirements,
2. plan ahead for the RE process, and
3. use the ‘waiting room’ for requirements that can not be implemented now
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 (e.g. Z), temporal logic, state machines

Resolving Conflicts
Boehm’s Win-Win Spiral

Multi-stakeholder involvement with coordination and collaboration based on
i) **Win Conditions** capture the desired objectives of the individuals
ii) **Conflict/Risk/Uncertainty specifications** (CRU’s) capture the conflicts between win conditions and their associated risks and uncertainties.
iii) **Points of Agreement (POA’s)** capture the agreed upon set of conditions which satisfy stakeholder win conditions and also define the system objectives.

Win-Win Model

1. identify next-level stakeholders
2. identify their win conditions
3. reconcile win conditions
4. evaluate product and process alternatives; resolve risks
5. define next level of product and process
6. validate next level of product and process
7. review & commitment; return to 1

W-W Negotiation Process

[Diagram showing the process of maintaining equilibrium through negotiation steps]

Feasibility Studies
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 Feasibility Studies

- 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 for the Feasibility Studies

- 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?
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
  - SCM, traceability tables, good record keeping, metrics

Definitions

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

Techniques

- SCM, traceability tables, good record keeping, metrics

Consequences of reqs change

- Depends when in the life cycle the requirements change
- Best case – review requirements specification
- Worst case – changes to requirements, 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

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

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 Tables

- 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

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)

References

  - Section 7.2.5 “Specification”
  - Section 27.1 “Software Configuration Management”
  - Section 28.1 “Formal Methods - Basic concepts”
  - Section 7.1 “Feasibility Studies”