Requirements Engineering

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Administration

- SEOC1 Tutorials start in week 3
- SEOC1 Communications:
  - Mailing List: seoc1-students@inf.ed.acuk
  - Newsgroup: ed.inf.course.seoc1
  - Links in the SEOC1 course webpage:

  http://www.inf.ed.ac.uk/teaching/courses/seoc1.html

Software Requirements

- Main activities involved in Software Requirements engineering:
  - **Elicitation**: Identify sources; Elicit requirements
  - **Analysis**: Classify requirements; Model; Top-level architecture; Allocate requirements to components; Negotiate requirements
  - **Specification**: Requirements Definition Doc; Software Requirements Specification; Document Standards; Document Quality
  - **Validation**: Reviews; Prototypes; Modelling; Test definition
  - **Management**: Traceability; Attributes; Change/Evolution
- The pattern, sequence and interaction of these activities is orchestrated by a Requirements Engineering Process.

10 Top Reasons for Not Doing Requirements

1. We don't need requirements, we're using objects/java/web/...
2. The users don't know what they want
3. We already know what the users want
4. Who cares what the users want?
5. We don't have time to do requirements
6. It's too hard to do requirements
7. My boss frowns when I write requirements
8. The problem is too complex to write requirements
9. It's easier to change the system later than to do the requirements up front
10. We have already started writing code, and we don't want to spoil it

Volunteer Bank (VolBank)

1. To develop a system that will handle the registration of volunteers and the depositing of their time. To record:
   i. The details of volunteers, contact details, skills and needs
   ii. The time that each volunteer deposits in the system
   iii. To transfer from the web server details of volunteers and the time they are depositing.
2. To handle recording of opportunities for voluntary activity:
   i. Details of voluntary organisations
   ii. Needs of voluntary organisations
   iii. Needs of individuals (inc volunteers) for help

VolBank continued

3. To match volunteers with people or organisations that need their skills:
   i. Match volunteer with local opportunities
   ii. Match local opportunity with a team of volunteers
   iii. Record matches between volunteers and opportunities
   iv. Notify volunteers of a match
   v. Notify organisations of a match
   vi. Record if agreement is reached from a particular match
4. To generate reports and statistics on volunteers, opportunities an time deposited.

VolBank: Elicitation

- Identify potential sources of requirements:
  - **Goals** (why the system is being developed): high level goal is to increase the amount of volunteer effort utilized by needy individuals and organisations – suggests possible requirements in measurement and monitoring
  - **Domain Knowledge**: not much relevant here but in some areas e.g. safety – hazard analysis; security – vulnerability and threat analysis
  - **Stakeholders**: at least: volunteers, organisations, system administrators, needy people, operator, maintenance, manager
  - **Operating Environment**: may be constrained by existing software and hardware in the office
  - **Organisational Environment**: legal issues of keeping personal data, safety issues in "matching"

VolBank: Elicitation

- Approaches to eliciting requirements:
  i. **Interviews with stakeholders**:
     i. Operator identifies:
        a. The need to change details when people move home
        b. The need to manage disputes when a volunteer is unreliable, or does bad work
     ii. Female Volunteer identifies: the need for security/assurance in contacting organisations, …
     iii. Management identifies number of hours volunteered per month above a given baseline as the key metric
  ii. **Scenarios**: means to elicit the usual flow of work
  iii. **Prototypes**: mock-up using paper or powerpoint or software
  iv. **Facilitated Meetings**: professional group work
  v. **Observation**: observing "real world" work

VolBank: A Failed Match Scenario

- **Goal**: to handle failure of a match
- **Context**: the volunteer and organisation have been matched and a date for a preliminary meeting established
- **Resources**: time for volunteer and organisation
- **Actors**: volunteer, operator, organisation
- **Episodes**: The volunteer arrives sees the job to be done and decides (s)he cannot do it, organisation contacts operator to cancel the match and reorganise.
- **Exceptions**: volunteer fails to show up

VolBank: Requirements Analysis

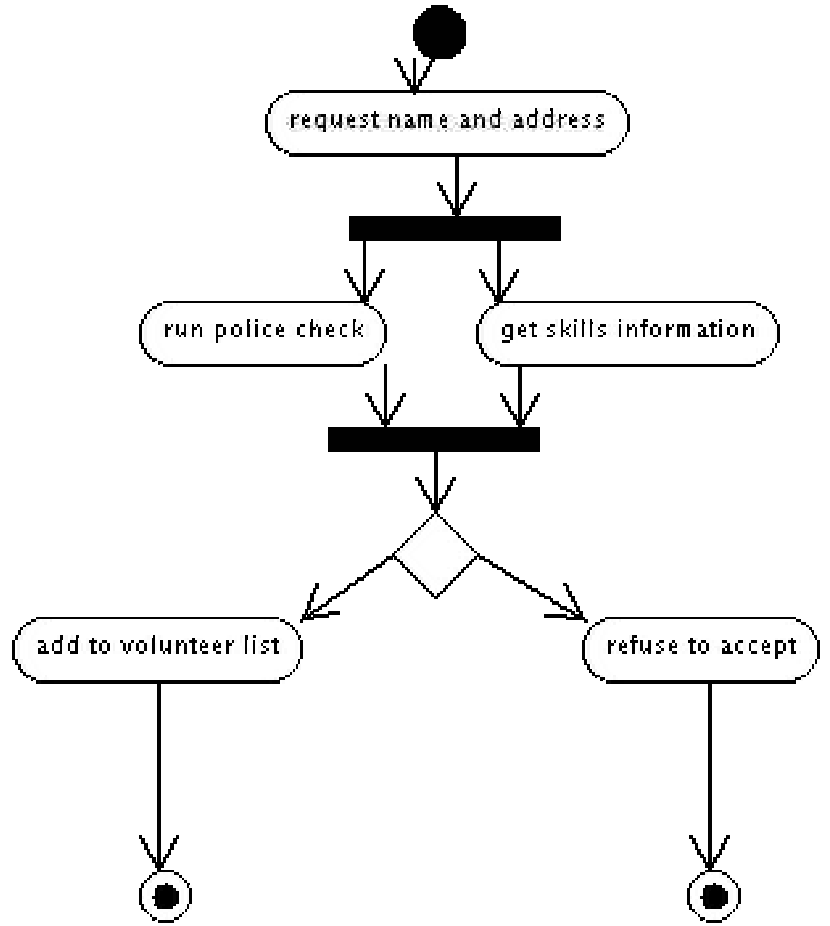
- Large volume of requirements information, need to analyse to
  - detect and resolve conflicts
  - scope the system and define interfaces with the environment
  - translate system requirements into software requirements.
- Analysis involves:
  - Classification
  - Conceptual Modelling
  - Architectural Design and Requirements Allocation
  - Requirements Negotiation

VolBank: Analysis continued

- **Classification**:
  - **Functional**: the system shall allow a volunteer to be added to the register of volunteers, the following data will be recorded: …
  - **Non-functional**:
    - The system shall ensure confidentiality of personal data and will not release it to a third party
    - The system shall ensure the safety of all participants
  - Process or product: the systems shall be developed under the relevant ISO 9001 standard.
  - Also priority of feature, scope, volatility

VolBank: Conceptual Modelling

- Begin to identify classes of object and their associations:
  - volunteer, contact details, match, skills, organisation, needs, etc.
- Start to consider some high level model of the overall workflow for the process using modelling tools.



An example of UML Activity Diagrams

VolBank: Design and Allocation

- How do we allocate requirements?
  - The system shall ensure the safety of all participants?
- Further analysis to determine principal threats:
  1. Safety of the volunteer from hazards at the work site
  2. Safety of the organisations from hazards of poor or inadequate work
  3. Safety of people from volunteers with behavioural problems
  4. …
- Design might allow us to allocate:
  - 1 to an information sheet
  - 2 to a rating component and procedures on allocating work
  - 3 to external police register
  - …

VolBank: Negotiation

- Safety and Privacy requirements may be inconsistent
  - need to modify one or both
  - Privacy: only authorised releases for safety checks will be permitted and there is a procedure for feeding back to the individual if a check fails.
- Some requirements may be achievable but only at great effort – attempt to downscale
  - it may be too much effort to implement a fault reporting system in the first release of the system

Other Activities

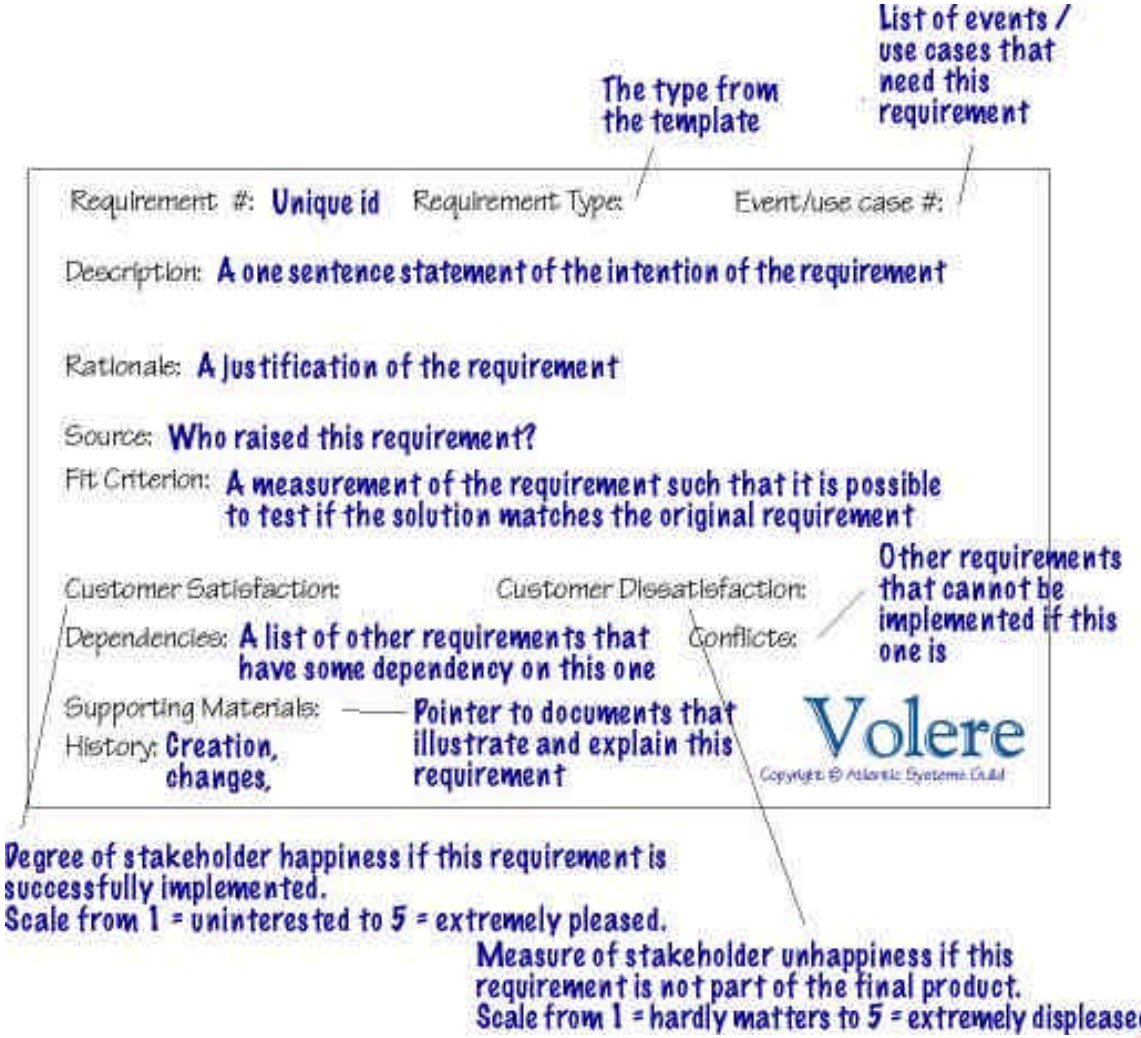
- Constructing specifications
  - System requirements definition: customer facing, at system level
  - Software Requirements Specification: developer facing, at software level.
- Requirements validation
  - key activity aim to get as much as possible
  - define the acceptance test with stakeholders.
- Requirements Management
  - requirements change because the environment changes and there is a need to evolve, need tools to manage the collection and maintain traceability.

How to organise requirements?

- Software Requirements Specification (SRS)
  - For example, see the **VOLERE** Template
- The SRS document is a structured documents that containing the identified requirements
- For instance, the VOLERE Template identifies the following major SRS parts:
  - PROJECT DRIVERS (e.g., The Purpose of the Product, Stakeholders, etc.)
  - PROJECT CONSTRAINTS (e.g., Costs)
  - FUNCTIONAL REQUIREMENTS
  - NON-FUNCTIONAL REQUIREMENTS (e.g., Usability, Performance, Operational, Maintainability, Portability, Safety, Reliability, Security, Cultural, etc.)
  - PROJECT ISSUES (e.g., Open Issues, Risks, Evolution, etc.)

How to collect requirements?

The VOLERE requirements shell provides a guide for writing requirements



References

- Requirements Engineering
  - Suzanne Robertson and James Robertson. Mastering the Requirements Process. Addison-Wesley, 1999.
  - Gerald Kotonya and Ian Sommerville. Requirements Engineering: Processes and Techniques. John Wiley, 1998.
  - Ian Sommerville and Pete Sawyer. Requirements Engineering: A good practice guide. John Wiley, 1997.
  - Dean Leffingwell and Don Widrig. Managing Software Requirements: A Use Case Approach. Addison-Wesley, Second Edition, 2003.
- Requirements Template
  - James Robertson and Suzanne Robertson. VOLERE: Requirements Specification Template. Edition 9, Atlantic Systems Guild.
- Look at the course resources (e.g., references, papers, presentations, webpages, etc.)

[web link from the SEOC1 webpage]

Reading/Activity

- Please read Chapter 2 – Software Requirements - of the **SWEBOK**. This provides a basic outline of the requirements process.
- Look at the course resources (e.g., references, papers, presentations, webpages, etc.)

  [web link from the SEOC1 webpage]
- Please read Chapter 3 – Use Cases - pages 25-46 of the UML book in preparation for the next lecture.
- Try running **Argo/UML** on one of the DICE machines - just type argo in a shell window.

Summary

- Requirements engineering is a very imprecise activity
- RE is the key activity to constructing good systems cheaply
  - poor requirements lead to very poor systems
- Issues are very wide ranging and negotiating agreement between all the stakeholders is hard
- In some application areas it may be possible to use a more formal notation to capture some aspects of the system (e.g., control systems, compilers, …)