Requirements Engineering

Massimo Felici

Room 1402, JCMB, KB

0131 650 5899

mfelici@inf.ed.ac.uk

Administration

- SEOC1 Tutorials begin in week 3

SEOC1 Communications

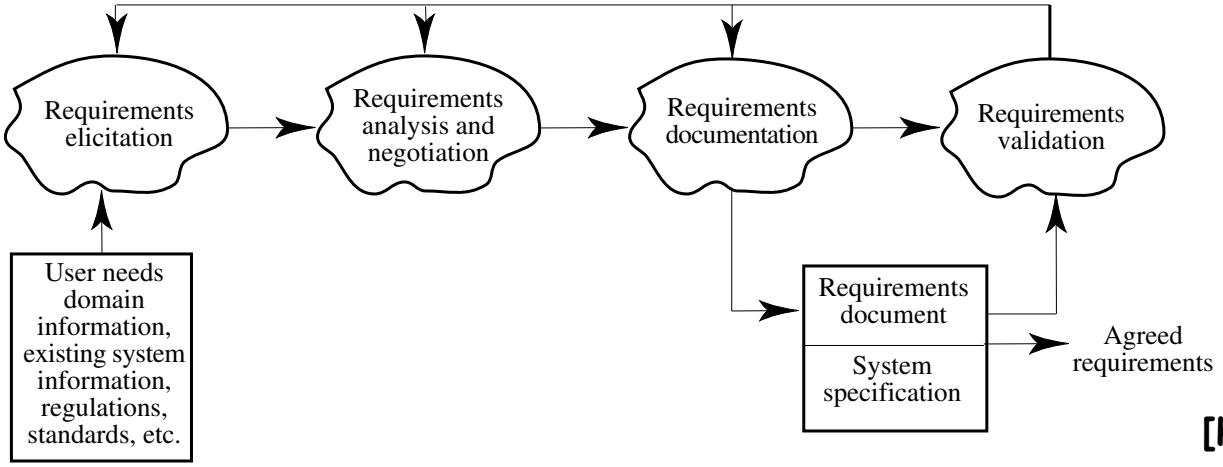
- **Mailing List**: **seoc1-students@inf.ed.ac.uk**
- **Newsgroup**: **eduni.inf.course.seoc1**
- **SEOC1 course webpage**:

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

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

Requirements Engineering Activities



[Kotonya and Sommerville, 1998]

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

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 organizations
   ii. Needs of voluntary organizations
   iii. Needs of individuals (inc volunteers) for help
3. To match volunteers with people or organizations 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 organizations 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.

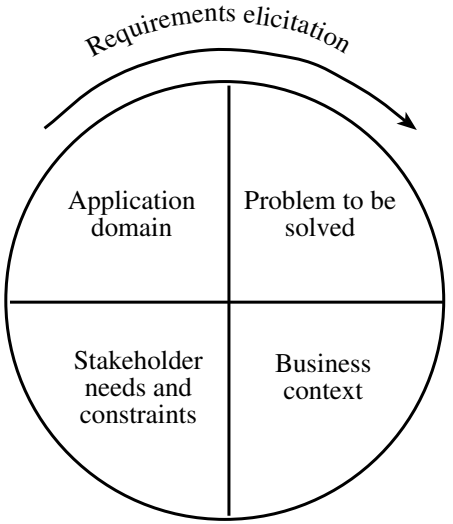
VolBank: Elicitation

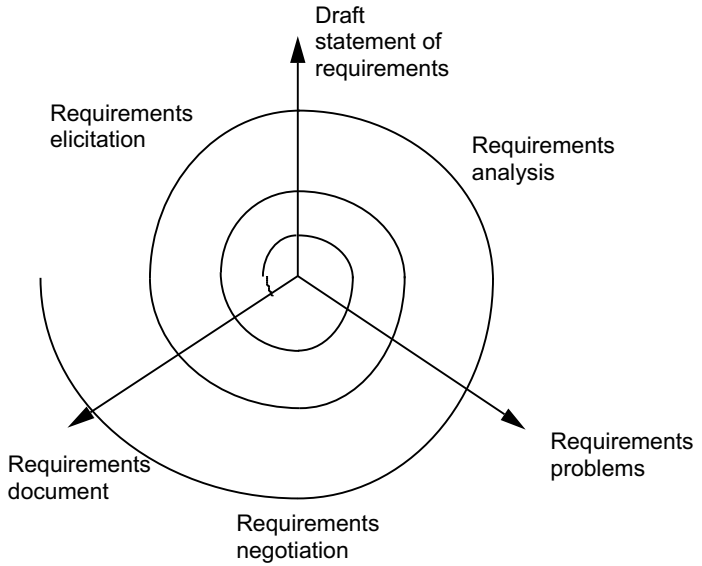
Identify potential sources of requirements

- **Goals** (why the system is being developed):
  - An high level goal is to increase the amount of volunteer effort utilized by needy individuals and organizations
  - It 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**:
  - volunteers, organizations, system administrators, needy people, operator, maintenance, manager
- **Operating Environment**:
  - may be constrained by existing software and hardware in the office
- **Organizational Environment**:
  - legal issues of keeping personal data, safety issues in "matching"

Requirements Elicitation Activities

- **Application domain understanding**
  - Application domain knowledge is knowledge of the general area where the system is applied
- **Problem understanding**
  - The details of the specific customer problem where the system will be applied must be understood
- **Business understanding**
  - You must understand how systems interact and contribute to overall business goals
- **Understanding the needs and constraints of system stakeholders**
  - You must understand, in detail, the specific needs of people who require system support in their work





[Kotonya and Sommerville, 1998]

Requirements Elicitation Techniques

- **Interviews with stakeholders**
  - Close/Open (Structured/Unstructured), Facilitated Meetings (e.g., professional group work)
- **Scenarios**
  - Elicit the "usual" flow of work
  - Are stories which explain how a system might be used
  - Expose possible system interactions and reveal system facilities which may be required
- **Prototypes**
  - mock-up using paper, diagrams or software
- **Observation**
  - Observing "real world" work
  - **Ethnography** is a technique from the social sciences
  - Actual work processes often differ from formal, prescribed processes

VolBank

Examples of requirements

A. Operator identifies:
   i. The need to change details when people move home
   ii. The need to manage disputes when a volunteer is unreliable, or does bad work

B. Volunteer identifies: the need for security/assurance in contacting organizations, …

C. Management identifies number of hours volunteered per month above a given baseline as the key metric

VolBank: A Failed Match Scenario

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

Requirements Analysis

- Discovers **problems**, **incompleteness** and **inconsistencies** in the elicited requirements
  - Large volume of requirements information
  - Detect and resolve conflicts
  - Scope the system and define interfaces with the environment
  - Translate system requirements into software requirements
  - Feedback to the stakeholders to resolve them through the negotiation process
- Involves:
  - Classification
  - Conceptual Modeling
  - Architectural Design and Requirements Allocation
  - Requirements Negotiation
- A problem checklist may be used to support analysis

A Problem Checklist

- Premature design
- Combined requirements
- Unnecessary requirements
- Use of non-standard hardware
- Conformance with business goals
- Requirements ambiguity
- Requirements realism
- Requirements testability

Non-functional Requirements

- Non-functional requirements (e.g., safety, security, usability, reliability and performance) define the overall qualities or attributes of the resulting system
- Constraints on the product being developed and the development process
  - E.g., the systems shall be developed under the relevant ISO 9001 standard

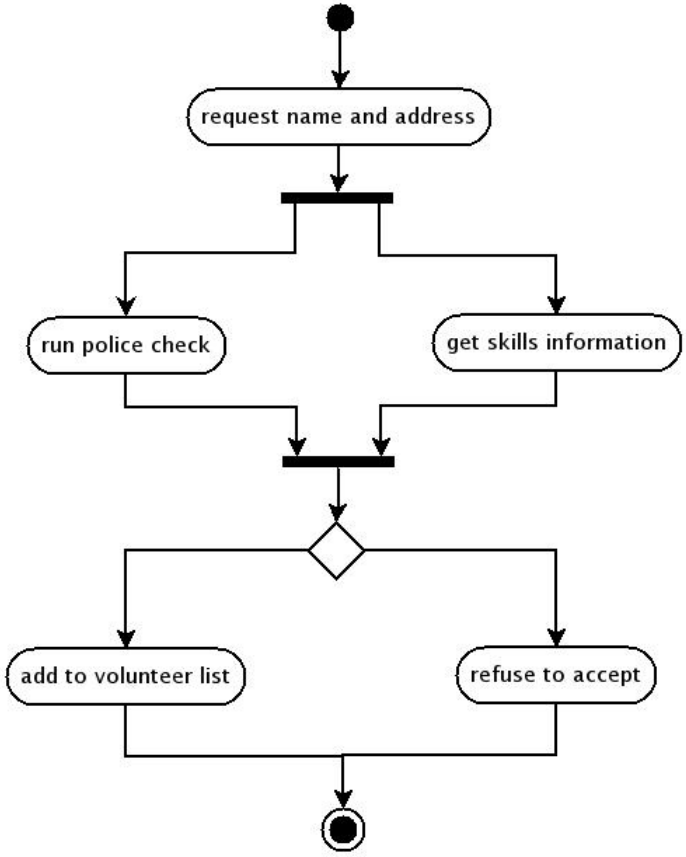
VolBank Analysis and Classification

- **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

VolBank: Conceptual Modeling

- Process of requirements engineering is usually guided by a requirements method
- Requirement methods are systematic ways of producing system models
- System models important bridges between the analysis and the design process
- Begin to identify classes of object and their associations:
  - volunteer, contact details, match, skills, organization, needs, etc.
- Start to consider some high level model of the overall workflow for the process using modeling 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 organizations 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** or **conflicting**
  - need to modify one or both
  - Privacy: only authorized 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
  - **Prioritize**
  - 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
  - Tools to manage requirements and maintain traceability

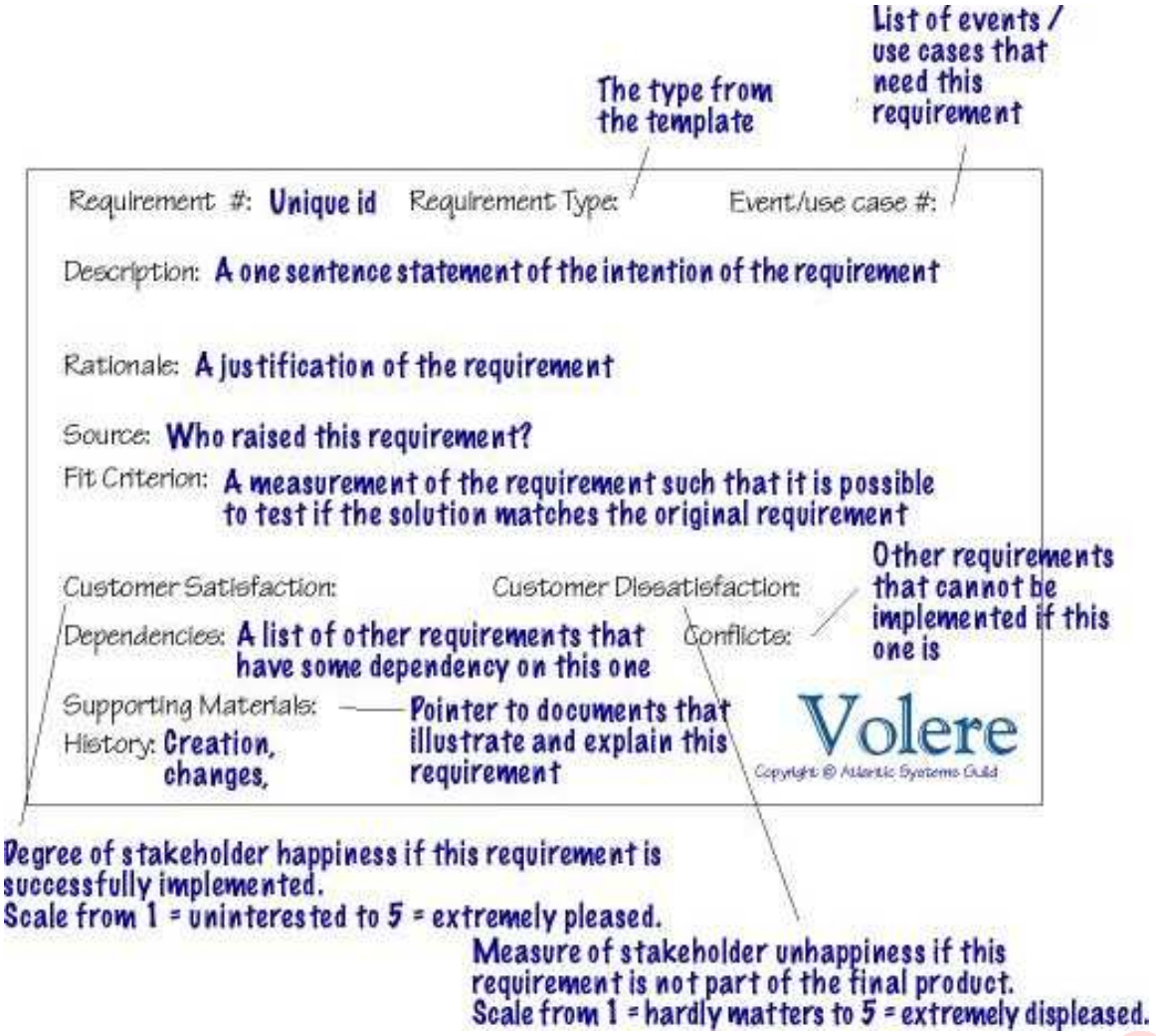
How to organize requirements?

- **Software Requirements Specification (SRS)**
  - 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.)

[SEOC1 Resource webpage]

How to collect requirements?

The VOLERE requirements shell provides a guide for writing requirements



Requirements Engineering References

- 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.

Reading/Activity

- Please read
  - Chapter 2 – Software Requirements - of the **SWEBOK**. It provides a basic outline of the requirements process.

    [SEOC1 Resource webpage]

  - **Requirements Template**: James Robertson and Suzanne Robertson. **VOLERE: Requirements Specification Template**. Edition 9, Atlantic Systems Guild.

    [SEOC1 Resource webpage]

  - **Chapter 3 – Use Cases** - pages 25-46 of the UML book in preparation for the next lecture.
  - Gary Cernosek and Eric Naiburg. **The Value of Modeling**. Rational Software, Copyright IBM Corporation 2004. This paper provides a brief technical discussion of software modeling.

    [SEOC1 Resource webpage]

- Run **Argo/UML** on one of the DICE machines
  - just type "**argouml**" in a shell window

Summary

- **Requirements engineering**
  - Involves diverse activities
  - Supports the construction of quality systems
- **Issues** are very wide ranging
  - Poor requirements lead to very poor systems
  - 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, …)