# Gathering and Organising System Requirements

Massimo Felici

School of informatics

# 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

# VolBank: Requirements

1. To develop a system that will handle the registration of volunteers and the depositing of their time.

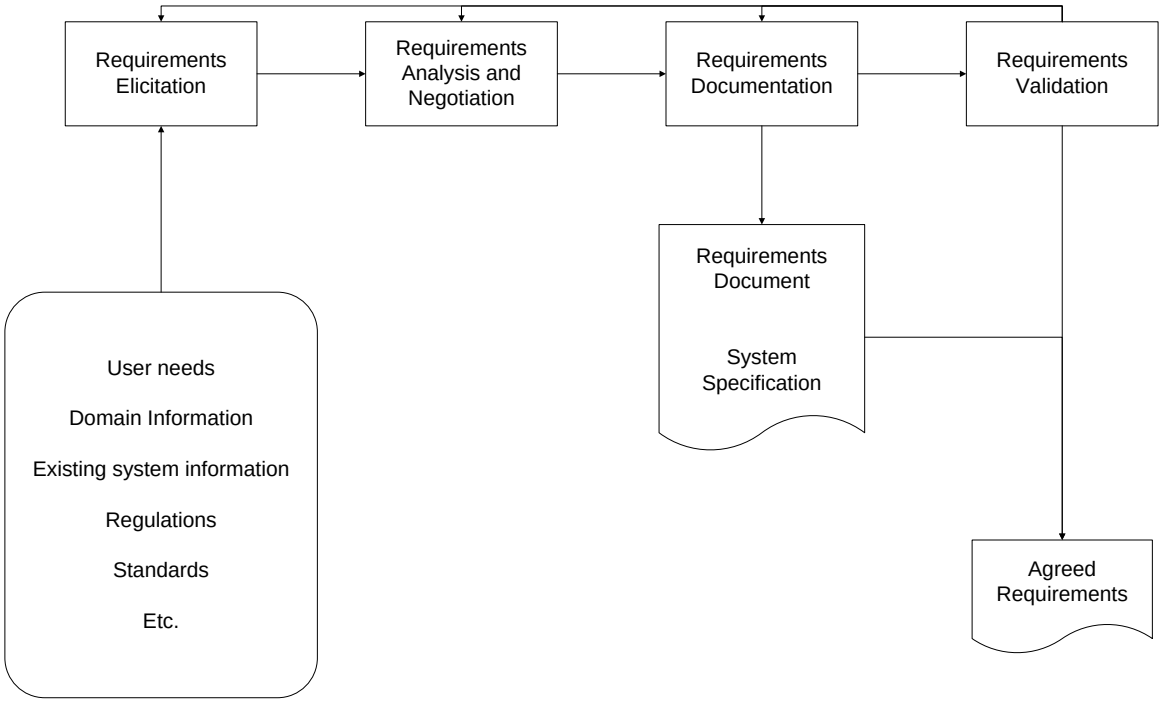
2. To handle recording of opportunities for voluntary activity.

3. To match volunteers with people or organisations that need their skills.

4. To generate reports and statistics on volunteers, opportunities an time deposited.

# Requirements Engineering Activities



# Slide 3: Requirements Engineering Activities

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; Modelling; Test definition

**Management:** Traceability; Attributes; Change/Evolution

The pattern, sequence and interaction of these activities is orchestrated by a Requirements Engineering Process.

# Slide 3: Requirements Engineering Activities

## Required Readings

- I. Sommerville. Integrated Requirements Engineering: A Tutorial. IEEE Software, January/February 2005, pp. 16-23.

## Suggested Readings

- I. Sommerville, P. Sawyer. Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997.
- G. Kotonya, I. Sommerville. Requirements Engineering: Processes and techniques. John Wiley & Sons, 1998.
- I. Sommerville. Software Engineering, Eighth Edition, Addison-Wesley 2007.
  - Chapter 6 on Software Requirements.
  - Chapter 7 on Requirements Engineering Processes.

# 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

# Slide 4: 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
- **Observations**
  Observing 'real world' work

Some requirements elicitation techniques find grounds in Ethnography - a technique from the social sciences. *Note that actual work processes often differ from formal prescribed processes.*

# VolBank: Elicitation

- **Goals** (why the system is being developed)
  An high level goal is to increase the amount of volunteer effort utilised by needy individuals and organisations
  Possible requirements in measurement and monitoring
- **Domain Knowledge**
  Some specific requirements, e.g., Safety and Security
- **Stakeholders**
  Volunteers, organisations, system administrators, needy people, operator, maintenance, manager
- **Operational Environment**
  Probably constrained by software and hardware in the office
- **Organisational Environment**
  Legal issues of keeping personal data, safety issues in 'matching'

# VolBank: Examples of Requirements

- **Volunteer** identifies:

  1. The need for security/assurance in contacting organisations

- **Management** identifies:

  2. The number of hours volunteered per month above a given baseline as the key metric

- **Operator** identifies:

  3. The need to change details when people move home
  4. The need to manage disputes when a volunteer is unreliable, or does bad work

# Requirements Analysis

- Discovers problems, incompleteness and inconsistencies in the elicited requirements
- A problem checklist may be used to support analysis
- **A Problem Checklist**
  - Premature design
  - Combined requirements
  - Unnecessary requirements
  - Requirements ambiguity
  - Requirements realism
  - Requirements testability

# Slide 7: Requirements Analysis

**Requirements Analysis involves:** Classification, Conceptual Modeling, Architectural Design and Requirements Allocation and Requirements Negotiation.

Requirements Analysis deals with large volume of requirements information, detects and resolves conflicts, scopes the system and defines interfaces with the environment, translates system requirements into software requirements and provides feedback to the stakeholders (in order to resolve conflicts through the negotiation process).

**A Problem Checklist:** Premature design, Combined requirements, Unnecessary requirements, Use of non-standard hardware, Conformance with business goals, Requirements ambiguity, Requirements realism, Requirements testability.

# Functional and Non-functional Requirements

**Functional Requirements:** *"These are statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do."*

**Non-functional Requirements:** *"These are constraints on the services. They include timing constraints, constraints on the development process and standards. Non-functional requirements often apply to system as a whole. They do not usualy just apply to individual system features or services."*

# Slide 8: Examples of Functional Requirements

Examples of *functional requirements* for a university library system used by students and faculty to order books and documents from other libraries.

1. The user shall be able to search either all of the initial set of databases or select a subset from it.

2. The system shall provide appropriate viewers for the user to read documents in the document store.

3. Every order shall be allocated a unique identifier, which the user shall be able to copy to the accounts permanent storage area.

# Slide 8: Non-functional Requirements

- **Non-functional requirements** (e.g., safety, security, usability, reliability, etc.) define the overall qualities or attributes of the resulting system

- **Constraints** on the product being developed and the development process

- ***Warnings: unclear distinction between non-functional and functional requirements***

**Suggested Readings**

- J. Boegh, S. De Panfilis, B. Kitchenham, A. Pasquini. A Method for Software Quality Planning, Control, and Evaluation. IEEE Software, March/April 1999, pp. 69-77.

# Slide 8: Metrics for Specifying Non-functional Requirements

| Property | Measure |
|---|---|
| Speed | Processed transactions/second |
| | User/Event response time |
| | Screen refresh time |
| Size | K bytes |
| | Number of RAM chips |
| | Lines of Code (LOC) |
| Ease of use | Training time |
| | Number of help frames |
| Reliability | Mean time to failure |
| | Probability of unavailability |
| | Rate of failure occurrence |
| | Availability |
| Robustness | Time to restart after failure |
| | Percentage of events causing failure |
| | Probability of data corruption on failure |
| Portability | Percentage of target-dependent statements |
| | Number of target systems |

# VolBank: Analysis and Classification

- **Functional Requirements**
  - The system allows a volunteer to be added to the register of volunteers. The following data will be recorded:...
- **Non-functional Requirements**
  - The system ensures confidentiality of personal data and will not release it to a third party
  - The system ensures the safety of all participants

# 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

# Other Requirements Activities

- **Constructing Specifications**

  System requirements definition: customer facing, at system level

  Software Requirements Specification: developer facing, at software level

- **Requirements Validation**

  define the acceptance test with stakeholders

- **Requirements Management**

  Manage requirements and maintain traceability

  Requirements change because the environment changes and there is a need to evolve

# Slide 11: Other Requirements Activities

There are four main types of traceability links with respect to their process relationships to requirements:

**Forward from requirements.** Responsibility for requirements achievement must be assigned to system components, such that accountability is established and the impact of requirements change can be evaluated.

**Backward to requirements.** Compliance of the system with requirements must be verified, and gold-plating (designs for which no requirements exist) must be avoided.

**Forward to requirements.** Changes in stakeholder needs, as well as in technical assumptions, may require a radical reassessment of requirements relevance.

**Backward from requirements.** The contribution structures underlying requirements are crucial in validating requirements, especially in highly political settings.

# Slide 11: Other Requirements Activities

## Suggested Readings

- M. Jarke. Requirements Tracing. Communications of the ACM, Vol. 41, No. 12, December 1998.

# VolBank: Conceptual Modelling

- 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, organisation, needs, etc.
- Start to consider some high level model of the overall workflow for the process using modelling tools

# How to organise requirements

*"The software requirements document (sometimes called the software requirements specification or SRS) is the official statement of what the system developers should implement. It should include both the user requirements for a system and a detailed specification of the system requirements. In some cases, the user and system requirements may be integrated into a single description. In other cases, the user requirements are defined in an introduction to the system requirements specification. If there are a large number of requirements, the detailed system requirements may be presented in a separate document."*

# Software Requirements Specification (SRS)

1. **PROJECT DRIVERS**
   - The Purpose of the Product
   - Scope
   - Stakeholders
2. **PROJECT CONSTRAINTS**
   - Development Environment
   - Deadlines
3. **FUNCTIONAL REQUIREMENTS**
   ***functional requirement 1:***
   ***...***
   ***functional requirement m:***
4. **NON-FUNCTIONAL REQUIREMENTS** (e.g., Reliability, Usability, Security, Maintainability, etc.)
   ***non-functional requirement 1:***
   ***...***
   ***non-functional requirement n:***
5. **PROJECT ISSUES**
   - Open Issues
   - Evolution (e.g., Requirements changes, Future Requirements, etc.)

# Slide 14: Software Requirements Specification (SRS)

*You may want to use the VOLERE template or the IEEE Std 830-1998 (tailored for your purposes) or any similar SRS template as support for your practical work. The VOLERE requirements shell and the IEEE standard provide guidance for writing requirements.*

## Required Readings

- IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998.

## Suggested Readings

- S. Robertson, J. Robertson. Mastering the Requirements Process. Addison-Wesley, 1999.

# VolBank: Design and Allocation

- **How do we allocate requirements?**
  - The system shall ensure the safety of all participants?
- **Further analysis to identify principal threats:**
  - Safety of the volunteer from hazards at the work site
  - Safety of the organisations from hazards of poor or inadequate work
  - Safety of people from volunteers with behavioural problems
  - ...
- **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
  4. ...

# VolBank: Negotiation

- **Safety** and **Privacy** requirements
  - may be inconsistent or conflicting
  - 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
  - **Prioritise**
  - It may be too much effort to implement a fault reporting system in the first release of the system

# Required Readings

- I. Sommerville. Integrated Requirements Engineering: A Tutorial. IEEE Software, January/February 2005, pp. 16-23.

- IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998.

# Suggested Readings

- I. Sommerville, P. Sawyer. Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997.
- G. Kotonya, I. Sommerville. Requirements Engineering: Processes and techniques. John Wiley & Sons, 1998.
- I. Sommerville. Software Engineering, Eighth Edition, Addison-Wesley 2007.
  - Chapter 6 on Software Requirements.
  - Chapter 7 on Requirements Engineering Processes.
- M. Jarke. Requirements Tracing. Communications of the ACM, Vol. 41, No. 12, December 1998.
- S. Robertson, J. Robertson. Mastering the Requirements Process. Addison-Wesley, 1999.
- J. Boegh, S. De Panfilis, B. Kitchenham, A. Pasquini. A Method for Software Quality Planning, Control, and Evaluation. IEEE Software, 16(2):69-77, 1999.

# 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

- It may be possible to use a more formal notation to capture some aspects of the system