Requirements Engineering

Massimo Felici

JCMB-1402 0131 650 5899

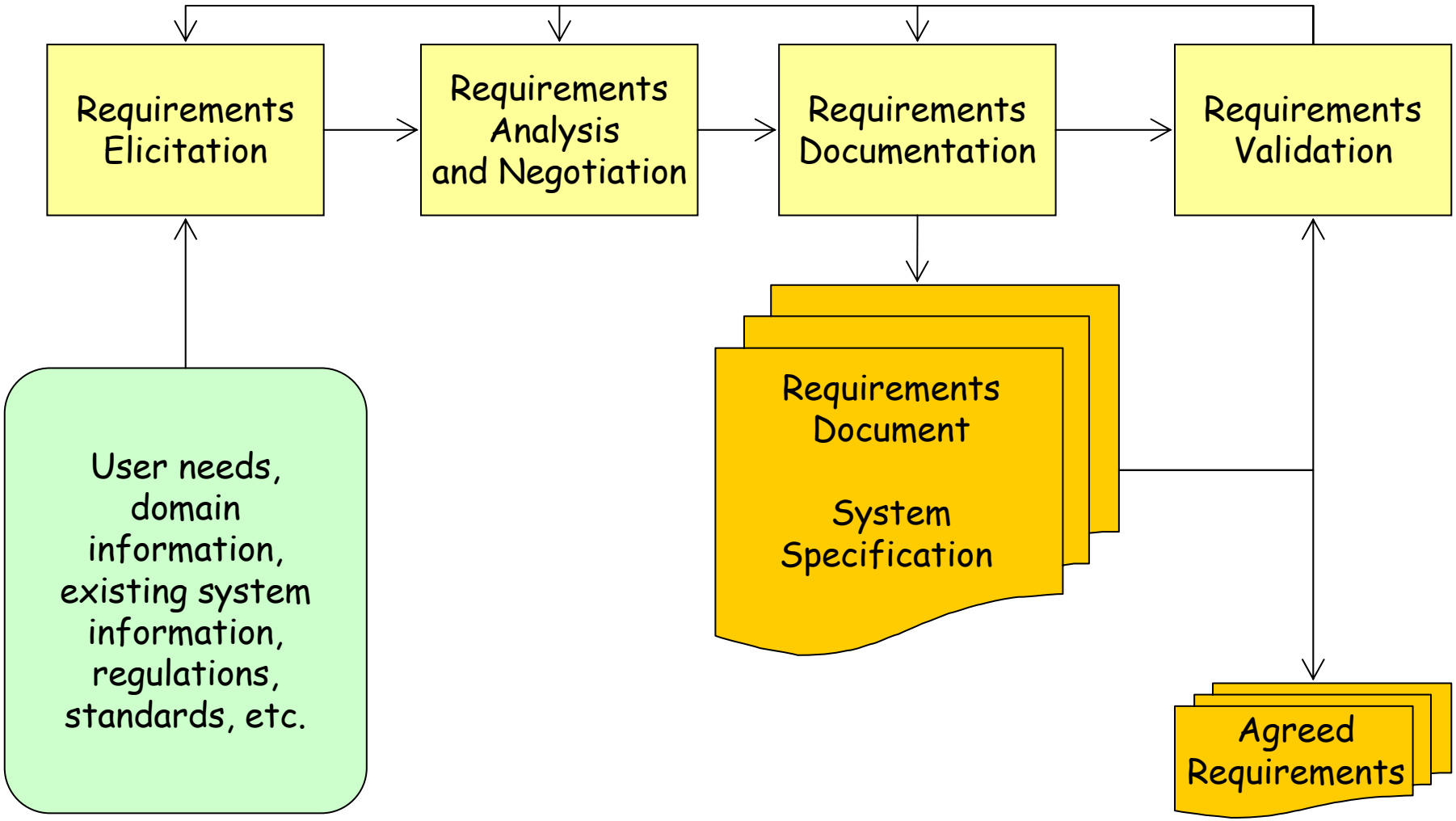
1BP-G04 0131 650 4408

mfelici@inf.ed.ac.uk

10 Top Reasons for Not Doing Requirements

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

Requirements Engineering Activities



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

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

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

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**

Other 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**

VolBank - Volunteer Bank

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 organizations that need their skills.
4. To generate reports and statistics on volunteers, opportunities an time deposited.

VolBank: Elicitation

- **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
  - Possible requirements in measurement and monitoring
- **Domain Knowledge**
  - Some specific requirements, e.g., Safety and Security
- **Stakeholders**
  - volunteers, organizations, system administrators, needy people, operator, maintenance, manager
- **Operational Environment**
  - Probably constrained by software and hardware in the office
- **Organizational 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 organizations, …
- **Management** identifies:
  1. The number of hours volunteered per month above a given baseline as the key metric
- **Operator** identifies:
  1. The need to change details when people move home
  2. The need to manage disputes when a volunteer is unreliable, or does bad work

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 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

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

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 organizations 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
  - …

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

How to organize requirements?

- **Software Requirements Specification (SRS)**
  - The SRS document is a structured documents that containing the identified requirements
- The **VOLERE Template** identifies the following SRS main 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.)

Requirements Engineering Practices

Examples of Requirements Engineering practices are:

- **Define a standard document structure**
  - For example, tailor a standard requirements specification template to your needs
- **Identify requirements uniquely**
  - For example, number each requirements specified in the requirements documentation

Readings

- **Requirements Specification Template**
  - J. Robertson, S. Robertson. VOLERE: Requirements Specification Template. Edition 10.1, Atlantic Systems Guild.
- I. Sommerville. Integrated Requirements Engineering: A Tutorial. IEEE Software, January/February 2005, pp. 16-23.
- 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.

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.
- 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.
- I. Sommerville. Software Engineering, Eighth Edition, Addison-Wesley 2007.
  - Chapter 6 on Software Requirements
  - Chapter 7 on Requirements Engineering Processes

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, …)