



Requirements elicitation

Software Engineering

2004-2005

Marco Scotto (Marco.Scotto@unibz.it)



Content

➤ **Introduction**

- Requirements engineering
- Requirements elicitation
- Requirements gathering
- Software requirements specification
- “Good” requirements
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts



Introduction

- Deciding precisely what to build is most important and most difficult
- Requirements are often buried under layers of assumptions, misconceptions, and politics
- Thorough understanding and constant communication with customers are essential



Content

- *Introduction*
- **Requirements engineering**
- Requirements elicitation
- Requirements gathering
- Software requirements specification
- “Good” requirements
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts



Requirements engineering

- Development, specification, and validation of requirements
- Elicitation and modeling
- Elicitation
 - Fact-finding, communication, and fact-validation
 - Output: requirements document
 - Understood by customers unambiguously
- Modeling (based on requirements document)
 - Representation and organization
 - Requirements in a form understood by software engineers unambiguously



Content

- *Introduction*
- *Requirements engineering*
- **Requirements elicitation**
- Requirements gathering
- Software requirements specification
- “Good” requirements
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts



Stakeholder

- Key representative of the groups who ...
 - Have vested interest in the system to be developed
 - Have direct and indirect influence on the requirements
- Examples: customers who pay, users who use, and technicians who maintain
- Each stakeholder has different perspectives and needs which have to be captured
- Involvement:
 - Spread throughout development life cycle (agile process)
 - at front-end of life cycle (plan-driven process)



Types of requirements

- Functional requirements
 - Services provided, reaction to specified inputs, behaviour in specified circumstances
- Non-functional requirements
 - *User-visible* properties relating to system as a whole
 - Security, privacy, usability, reliability, availability, and performance
 - Defects are expensive and hard to fix
- Constraints
 - Imposed by client, restricting implementation
 - *No direct effect on users' view* of system

e.g. programming language, development platform



Security & privacy

- Protection's focus: what, from whom, and for how long
- Examining the security and privacy policies of the organization
- Privacy policy
 - Privacy rights of users and information usage
- Security policy
 - Interaction of internal and external users, computer architecture topology, location of computer assets
- Reasons for engineers to understand the policies
 - Able to ask right questions early
 - Able to spot inconsistencies between policies and requirements



10 problems of requirements elicitation

1. The boundary of the system is ill-defined.
2. Unnecessary design information may be given.
3. Stakeholders have incomplete understanding of their needs.
4. Stakeholders have poor understanding of computer capabilities and limitations.
5. Software engineers have poor knowledge of problem domain.
6. Stakeholder and software engineers speak different languages.
7. "Obvious" information is omitted.
8. Different stakeholders have conflicting views.
9. Requirements are vague and untestable, such as "user friendly" and "robust".
10. Requirements are volatile and change over time.



Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- **Requirements gathering**
- Software requirements specification
- “Good” requirements
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts



9 info gathering techniques (1/5)

First 6 techniques: initial requirements capture

➤ Interviews

- Structured interview
 - Pre-determined questions and clear planned agenda
 - Questions: open-ended (stakeholders say what they want) or closed ended (multiple choice, ranking, rating)
- Unstructured interview
 - No questions prepared (free discussion)

➤ Observation

- Passive
 - no interruption to or direct involvement in business activities or via studies audio/video recordings
- Active
 - Participation and/or becoming part of the team





9 info gathering techniques (2/5)

- Examining existing documents and artifacts
 - Any form, automation, and policies
- Joint application design (JAD) sessions
 - Guide users and relevant experts through defining requirements, process, data models, and mock-ups
 - 6 roles of the JAD participants in a session:
 - Executive sponsor: supports or pays the project
 - Facilitator: moderates the meeting
 - Project leader: leader of the development team
 - Participants: stakeholders and engineers
 - Scribe: records and publishes proceedings
 - Development team members: “the quiet guys at the back”



9 info gathering techniques (3/5)

➤ Groupware

- Software tool for distributed requirements gathering
- Supports communication through video and audio conferencing, interactive chat, and email

➤ Questionnaires

- Reaching a wide range of people
- Obtaining honest, anonymous input
- Hard to analyze open-ended questions
- Less control over results



9 info gathering techniques (4/5)

Last 3 techniques:

- During development process
- For collecting feedbacks & additional requirements

➤ Prototypes

- Partially-developed demonstration system
- For interactions with stakeholders
- Paper prototype or automated prototype



9 info gathering techniques (5/5)

➤ Customer focus groups

- Reviewing interim results
- Obtaining feedback on quality and effectiveness of the system
- Documentation of requirements changes
- Prioritization on future work



➤ On-site customer

- Customer or stakeholder available nearby
- Providing valuable clarification and feedbacks as soon as the need arises



Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- *Requirements gathering*
- **Software requirements specification**
- "Good" requirements
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts




SRS

- Means for documenting requirements for relatively large projects with fairly stable requirements
- Templates
 - Often adopted by organizations as a standard form to specify requirements
 - Easier for readers to understand
- Other forms of requirements documentation
 - Based on use cases
 - Based on user stories (agile approach)



Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- *Requirements gathering*
- *Software requirements specification*
- **"Good" requirements**
- Requirements validation review
- Requirements volatility
- Requirements engineering and the three beasts



Properties of good requirements (1/4)

➤ Understandable

- No confusion and misunderstanding
 - domain-specific language and terms confuse developers
 - Technical terms confuse stakeholders
- Using short, declarative statements
- Examples, figures, and tables for clarification


➤ Non-prescriptive

- Stating what customer wants, not how programmer will do it



Properties of good requirements (2/4)

- Concise
 - Facilitating customer's validation of requirements
 - Prevents developers from skimming through info
 - Use KISS principle
- Consistent language
 - "Shall" statement → a "contract" or mandatory
 - "Should"/"may" statement → desirable but optional
- Consistent
 - No contradiction between requirements
- Correct and complete
 - Exhaustive list of requirements



Properties of good requirements

(3/4)

➤ Unambiguous → testable

- Writing test cases during requirements elicitation
 - Involve customers early
- Specify a quantitative description for each adverb and adjective
- Replace pronouns with specific names of entities
- Every noun is defined in exactly one place in the requirement document

➤ Traceable

- Requirements assigned with unique identifiers
- Easing the future reference to requirements



Properties of good requirements (4/4)

- Ranked for importance and stability
 - Should be decided together by team and stakeholders
 - Requirements negotiation process for determining:
 - Realistic priorities
 - How likely a requirement will change
- Feasible
 - Infeasible requirements found in elicitation phase
 - To be explained by stakeholder immediately
 - Infeasible requirements found in analysis phase
 - Stakeholder notified and requirements document updated





Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- *Requirements gathering*
- *Software requirements specification*
- *“Good” requirements*
- **Requirements validation review**
- Requirements volatility
- Requirements engineering and the three beasts



Requirements validation review

- Neutral and formal meetings
- Ensuring the document clearly and accurately reflect actual requirements
- Validation checklists used as reminder of what to look for in SRS
- **Realistic** about the number of requirements that can be reviewed in one meeting before the team gets tired





A sample checklist for a cell phone

- ✓ It turns on and off
- ✓ It sends and receive emails
- ✓ It sends and receives SMSs
- ✓ It sends and received MMSs
- ✓ It sends and receives calls
- ✓ It takes pictures
- ✓ It lets you review the pictures
- ✓ It traces meeting
- ✓ It records contacts
- ✓ It reproduces MP3
- ✓ It records videos
- ✓ It plays video
- ✓ It works with UMTS networks
- ✓ It connects to the Internet via GPRS
- ✓ It connects to the Internet via UMTS
- ✓ It supports bluetooth
- ✓ It supports infrared
- ✓ It receives FM radios

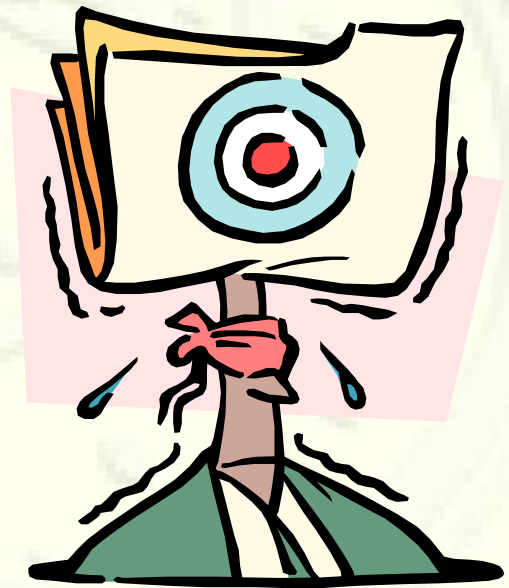


Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- *Requirements gathering*
- *Software requirements specification*
- *“Good” requirements*
- *Requirements validation review*
- **Requirements volatility**
- Requirements engineering and the three beasts

Requirements volatility

- Describing amount of change in requirements between the beginning and end of project
- Over time ...
 - Users' needs may mature due to increased knowledge about the system
 - Users may shift to new set of needs due to unforeseen pressures





Iterative requirements formulation

- Re-examining requirements with stakeholders periodically
- Allow requirements to evolve over time
- Some efficiency is lost when changes are allowed
- Wrong assumptions detected & corrected faster
- One-time requirements formulation
 - Poor practice
 - Getting what stakeholders want *initially*, but not what they *actually* want



Taming requirements volatility

- Change control board
 - A group of managers, clients, and developers together to decide the fate of proposed changes
 - Trade-off between rejecting changes and possible dissatisfaction of stakeholders with the product later
- Using a defined methodology for requirements analysis and modeling & frequent communication with customers
 - Requirements less volatile
- Scope creep must be controlled
 - Don't give customers complete freedom in *redefining* requirements



Content

- *Introduction*
- *Requirements engineering*
- *Requirements elicitation*
- *Requirements gathering*
- *Software requirements specification*
- *“Good” requirements*
- *Requirements validation review*
- *Requirements volatility*
- **Requirements engineering and the three beasts**



Requirements engineering & the three beasts

- Uncertainty
 - Difficult to formulate and document accurately and completely the desired system; volatility
- Irreversibility
 - Poor requirements are usually deeply embedded in the system; a lot of rework due to cascading effect
- Complexity
 - Having to deal with different stakeholders with different perspectives
- The beasts can be tamed by good requirement engineering practices