Introduction to Software Engineering

Requirement engineering – part I

Philippe Lalanda

Philippe.lalanda@imag.fr

http://membres-liglab.imag.fr/lalanda/

Purpose of this lecture

- Define the notion of requirement
- Present the different forms of requirements
- Introduce requirement engineering

Preliminary definitions

- Contract

  A legally binding document agreed upon by the customer and supplier. This includes the technical and organizational requirements, cost, and schedule for a product.

- Customer

  The person(s) who pays for the product and usually decides the requirements.

- Supplier/provider

  The person(s) who produces a product for a customer.

- User

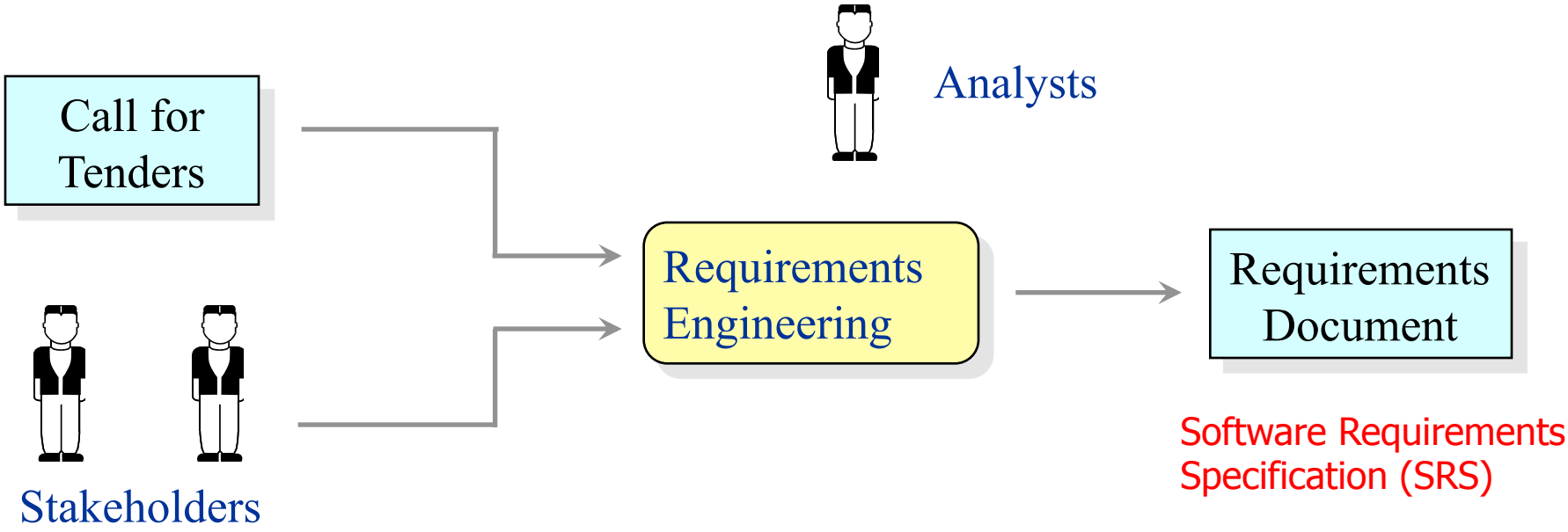
  The person(s) who operates or interacts directly with the product. The user(s) and the customer(s) are often not the same person(s).

Outline

- Introduction
- Requirements
- Functional requirements
- Quality requirements
- Requirement engineering
- Conclusion

Requirements engineering

- Objectives
  - Find out the needs and constraints of customers
  - Specify them in a dedicated document (SRS)



Call for tenders

- The first expression of the customer's needs and constraints
  - Written by the customer
  - Sizes and forms are very diverse
  - Often high-level, unstructured requirements

- Basis for a bid for a contract
  - Must be open for interpretation (to encourage competing bids)
  - Must be open for different solutions

Stakeholders

- All the people having an interest in the project
  - Something is at stake for them!
- Many people are concerned
  - Customer side: users, domain experts, technical managers, marketing managers, sales men …
  - Supplier side: sales men, development teams, architects, technical managers, strategist …

Requirements origin

- Requirements always come from the customer side
- They may be written (specified) by
  - one or more representatives of the supplier
  - one or more representatives of the customer
  - or by both.

Requirements document (SRS)

- A complete, thorough description of all necessary requirements for project development
  - An official project deliverable
  - Basis for contract between clients and contractors
  - Basis for the future development phases
  - Usually a big document!

- Must be of high quality and approved
  - No room for interpretation!
  - Will be used in many situations

Importance of requirements

- Impacts of requirements
  - Legal impact
    - basis of the contract between customer and supplier
  - Economic impact
    - cost of correcting wrong requirements
    - Relevance of the marketed product
  - Social impact
    - wrong requirements may cause disasters
  - Usage impact
    - acceptance or rejection of a software

Importance of requirements

- Requirements influence all the software activities
  - Architecture
  - detailed design
  - test definition
  - acceptance, …
- More and more, tests are prepared during the requirement engineering phase
  - agile programming for instance

The requirements issue

- It is very difficult to formulate a complete and consistent set of requirements
  - Clients and contractors speak a different language
  - Sometimes clients do not know exactly what they want
  - Contractors have to deal with big masses of information
  - Stakeholders with different viewpoints (even conflicts)
  - Constantly shifting compromise
  - Clients needs and wishes evolve
  - Some problems can never be fully understood and understanding evolves during the system development

Outline

- Introduction
- Requirements
- Functional requirements
- Quality requirements
- Requirement engineering
- Conclusion

What is a requirement ?

- Requirements express what customers want or need
  - A « need » is something mandatory that we must have
  - A « want » is nice to have but not always mandatory

- A requirement can be
  - a goal,
  - a provided function,
  - a quality,
  - A property (domain / organization),
  - a constraint

Provided functions - 1

- Usually called functional requirements
- A function is a service provided to users or an internal task of the software
- Description of a function includes
  - Name and purpose
  - Manipulated concepts (data)
  - Inputs and outputs
  - Expected behavior
  - Constraints
  - Expected HMI

Provided functions - 2

- Examples (library software)
  - The software has to manage books borrowing
  - The checkout function starts by reading the subscriber card
  - A subscriber has to pay 20 euros per year
  - A subscriber is defined by his name, age, etc.
  - All needed information have to be displayed in a single window

Qualities - 1

- Usually called non functional requirements
- External or internal software qualities of
  - the provided functions
  - the global system
- They must be quantified (to be evaluated)

Qualities - 2

- Examples
  - Book checkout must be made in less than 1 minute
  - Any function must be done in less than 2 minutes
  - Backtracking must always be possible when borrowing a book
  - The software system must be available 6 days a week
  - The software system may be portable …

Constraints - 1

- A constraint under which a software operates or is developed

Process

- Methods and tools
- Standards …

Domain

- Usage
- Regulation (law) …

Development

- COTS (OS, middleware)
- Language …

Context

- Existing applications
- People …

Constraints - 2

- Examples
  - The system must use Oracle for persistency functions
  - UML must be used for the modeling phase (design)
  - An architectural design document must be provided
  - Java annotations may be used for development
  - No historic is maintained for subscribers
  - The system must interface with legacy systems “A”

Multiple abstraction levels

- Requirements can range from a high-level abstract statements to detailed mathematical functional specification
  - Several abstraction levels

- Example of categorization
  - Goal requirements
  - High level requirements
  - Detailed requirements

Examples

- Goal
  - The system must deal with copyright management
- High-level requirement
  - 1. The system must keep track of all data related to copyright
- Detailed requirements
  - 1.1 For every request, the user has to fill in a form with his names, id, and his precise demand
  - 1.2 Forms have to be kept for 5 years
  - 1.3 Forms have to be indexed by requesters, required documents, providers
  - 1.4 All requests have to be logged
  - 1.5 For copyright documents, authors will be paid every months

Requirements characteristics (wikipedia)

Unitary	A requirement addresses one and only one thing
Complete	A requirement is fully stated in one place with no missing information
Consistent	A requirement does not contradict any other requirement and is fully consistent with authoritative documentation.
Atomic	A requirement does not contain conjunctions
Traceable	A requirement meets all or part of a business need as stated by stakeholders and authoritatively documented

Requirements characteristics (wikipedia)

Current	A requirement has not been made obsolete by the passage of time
Feasible	A requirement can be implemented within the constraints of the project
Unambiguous	A requirement is subject to only one interpretation. No jargon, acronyms, esoteric verbiage.
Specify Importance	A requirement must specify a level of importance.
Verifiable	The implementation of arequirement can be determined. Inspection, demonstration, test, analysis …

Understandability

- Requirements must be understood by both the customers and the suppliers
  - To agree on the contract
  - To allow discussions with the users all along the development cycle

Traceability

- Many links to be maintained
  - Between requirements of different abstract levels
    - Goals -> functions -> sub functions
  - Between requirements at same level of abstraction
    - Related goals or related functions
  - Between requirements and implementation
  - Between requirements and tests
  - …

Some fun

- Once upon a time a car manufacturer decided to reduce its costs.
- Examining the requirements, someone noticed that a car was designed to resist rainy winds of 200 km/h from the back, which led to important manufacturing c
- It was decided to s this requirement, which led to cheaper trunks.
- In autumn, car vendors noticed water in the trunks. It turned out that cars were transported by Express trains (speed of more than 200 km/h!).

LOST TRACEABILITY

Outline

- Introduction
- Requirements
- Functional requirements
- Quality requirements
- Requirement engineering
- Conclusion

Requirement

- A requirement is an expression of desired behaviour
- A requirement deals with
  - objects or entities
  - the state they can be in
  - functions that are performed to change states or object characteristics

Object, function and state – from Davis, 93

- A functional requirement relates to …
  - an object
    - A client is identified by his name, age and address
  - or a function
    - A client can borrow up to 5 books
  - or to a state
    - A book is available, borrowed or lost
  - or both

Objects – from Davis, 93

- An object is a clearly defined entity
  - Corresponds to a concept
  - Only concepts related to the software are to be considered → not a domain analysis
- Requirements must define the objects
  - Name and meaning
  - Structure
  - Scope

Objects - 2

- Examples
  - A client is defined by his name, age, address
  - A book is defined by its title and author(s)

  - A client has to be ten or more (constrain)

Functions – from Davis, 93

- A function is a clearly defined activity changing objects (state or characteristics)
  - A function can be visible (a service) or not (internal process or task)
  - Only activities realized by the software are to be considered → not a domain analysis
- Requirements must define the functions
  - Name and meaning
  - Interfaces and manipulated data
  - behavior
  - Demanded resources …

Functions – 2

- Examples
  - Book checkout
    - To borrow a book, a subscriber has first to give his library card and then the different books he desires
  - Book check-in
  - Displaying subscriber information
    - When displaying clients information, id is given first, on the top of the window
  - Records about books borrowed must be saved every night

States – from Davis, 93

- A state characterizes the situation of an entity
  - Can be expressed as a predicate
  - Can change over time
  - Influences the behavior of the entity
- Requirements must define the states
  - All possible states (first state and end state in particular)
  - Transitioning events
  - Possible properties

States - 2

- Examples
  - A book can be available, borrowed or lost
  - A client is characterized by a number of borrowed books
  - A client is characterized by his situation regarding the payment of the bill

Objects, functions, states

- Many requirements define relationships between objects, functions and states
  - A client can borrow a book when he has paid his bill and has less than 5 borrowed books

- Analysis methods concentrate on a single aspect
  - Object
  - Function
  - state

Outline

- Introduction
- Requirements
- Functional requirements
- Software qualities
- Requirement engineering
- Conclusion

Software qualities

- Also called non functional requirements
- External or internal
  - System level or function level
- Hard to elicit and verify
  - Many interpretations are possible
  - Sources of conflicts
- It is necessary to quantify these requirements
  - With metrics that can be measured, tested, …

Example

- First formulation
  - The system must be easy to use for an experienced controller and must be organized to limit the number of errors
- Re formulated
  - A controller with more than 5 years of experience must be able to use the system after a 2-hour training. After that, the number of error per day must not exceed 2.

Performance

- Quality perceived by users (external)
- Requirements to be specified
  - Number of transactions per second
  - Arrival rate of inputs
  - Refreshing time
  - Response time for a given pattern of events
  - What to do when expected quantities are exceeded
    - Failure, ignorance of additional inputs, degraded services

Usability

- Quality perceived by users (external)
- Requirements to be specified
  - Provided HMI
  - Error messages
  - keyboard shortcut
  - Backtrack possibilities
  - Techniques to help users and to improve confidence
  - Amount of expected training
  - Facilities to avoid misuse

Availability and reliability

- Quality perceived by users (external)
- Requirements to be specified
  - Max. number of bug per Kline during integration
  - Min. duration with[illegible]m
  - Is there a ti[illegible]perform maintenance?
  - Maximum time allowed for restarting the system
  - Must backup copies be stored at a different location?
  - Must the system detect and isolate faults?

HARD TO ASSESS

Security

- Quality perceived by users (external)
- Requirements to be specified
  - Must access to the system or information be controlled?
  - Should each user's data isolated from the data of each other?
  - Should user programs be isolated from other programs and from the operating system?
  - Should precautions be taken against theft or vandalism?

Maintainability

- Quality perceived by engineers (internal)
- Requirements to be specified
  - When and in what ways might the system be changed in the future?
  - How easy sh[illegible] features to the system?
  - How easy should it be to port the system from one platform (computer, OS) to another?

HARD TO ASSESS

Synthesis

- It is often just not possible to guarantee a software quality
  - Security?
  - Availability?
- Example of Internet sites
  - Full security and 100% availability is demanded!

Outline

- Introduction
- Requirements
- Functional requirements
- Software quality
- Requirement engineering
- Conclusion

Requirement engineering

- A sub discipline of software engineering concerned with establishing the functions, qualities and constraints of software systems

- Requirements engineering is a process that continues through the lifetime of a system
  - requirements are subject to change
  - new requirements must be elicited and documented and existing requirements managed over the lifetime of the system

RE phases

Analysis	Identifying new requirements and stakeholder conflicts
Definition and negotiation	Checking requirements and resolving stakeholder conflicts
Specification	Documenting the requirements in a requirements document (SRS)
Validation	Checking that the documented requirements and models are consistent and meet stakeholder needs
Management	Managing changes to the requirements as the system is developed and put into use

Analysis

- It starts with high level investigation
  - what are the goals?
  - what are the constraints?
  - what are the current tools or processes in place?
- Can lead to a feasibility study
  - Tools and techno available?
  - In line with existing systems, strategy?
- Only when these requirements are well understood can functional requirements be developed

Analysis - continued

- The purpose is to gather information about the software
  - Understand the domain and computing environment
  - Identify requirements and conflicts
- Analysis is an opening phase
  - Imply as many stakeholders as possible
  - Avoid pre conceived ideas
  - Beware of self censure
  - Beware of the apparent simplicity of goals and needs

Requirements definition

- The purpose is to confront stakeholders with possible requirements and establish a list of valid requirements
  - Comparison of alternative options
  - Resolution conflicts
  - Negotiation of best tradeoffs
  - Get a shared agreement
- It is a closing phase
  - Group stakeholders
  - Reduce requirements to a stable core

Requirements specification

- The purpose is to clearly describe the software to produce
  - Documentation in form understandable by all parties
- It is a synthesis phase
  - Requirements writing
  - Requirements structuration (type, abstraction level)
  - Verification of consistence, completeness, …
  - Possible priority assignment
  - Validation

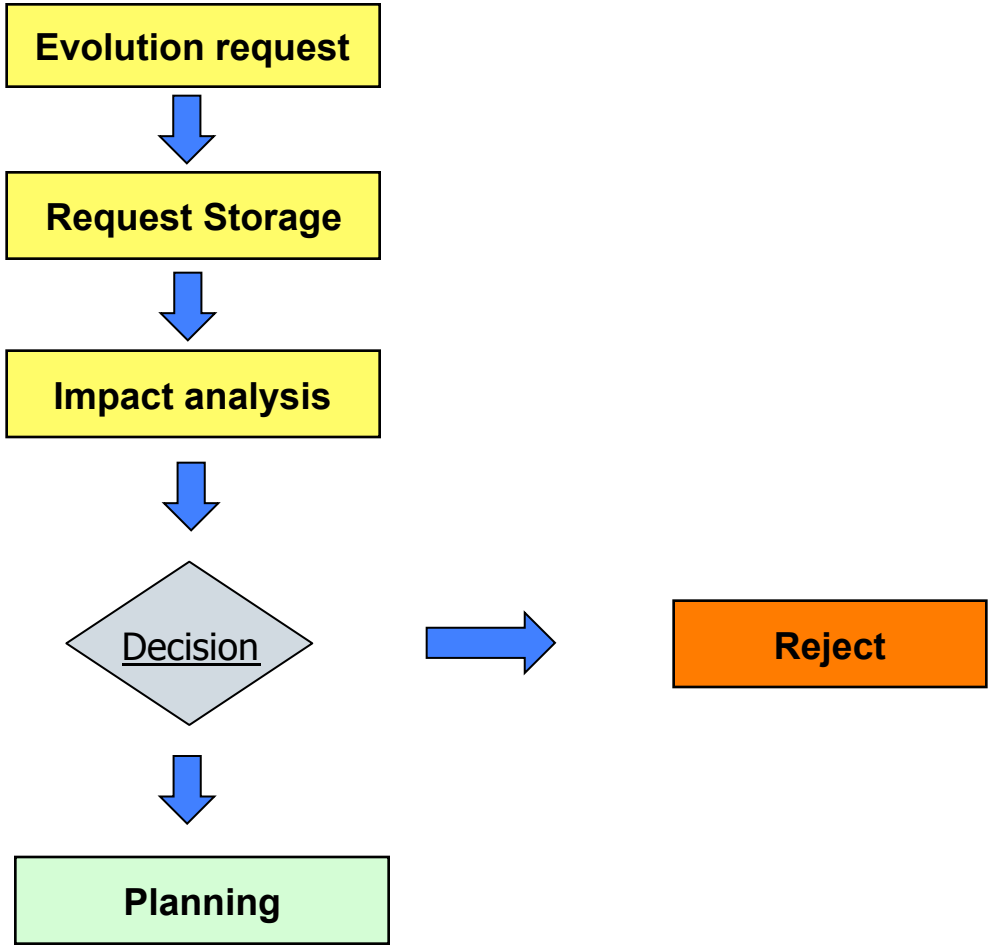
Requirements validation

- The purpose is to verify that the written requirements really express the clients goals
  - Completeness
  - Consistency
  - Adequacy
  - Precision
  - Relevance
  - Understandability
  - Good structuring
  - Modifiability
  - Traceability
  - Measurability

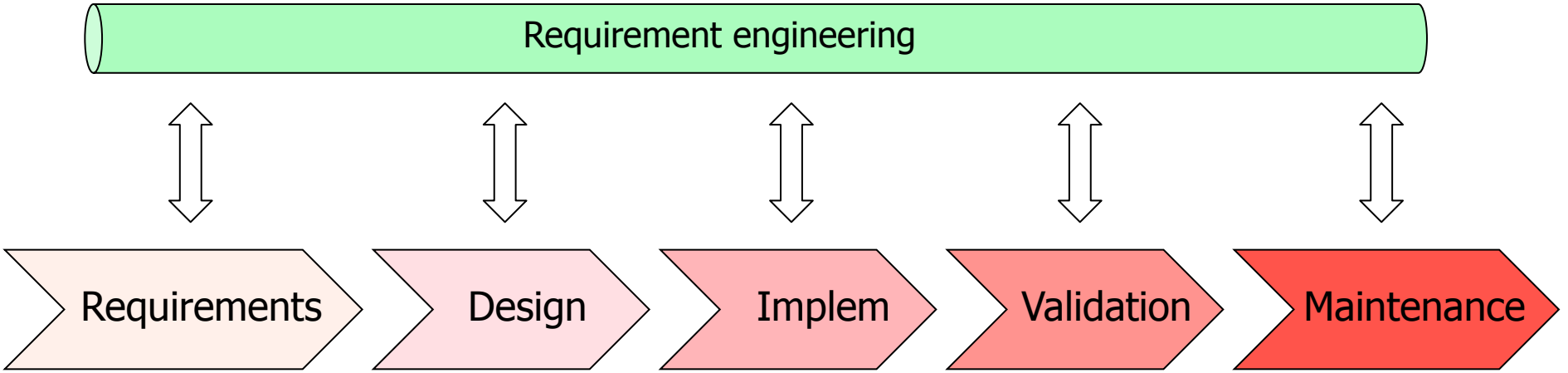
Requirements management

- No matter how hard you try, requirements cannot be fully defined at the beginning of the project.
- Some requirements will change
  - because they simply weren't extracted or known
  - because internal or external forces at work affect the project in mid-cycle (new needs, new techno, new stakeholder, etc.)
- Flexibility is a prime condition for success
  - An evolution management process is needed
  - Tools needed (especially for traceability)

Requirements evolution process



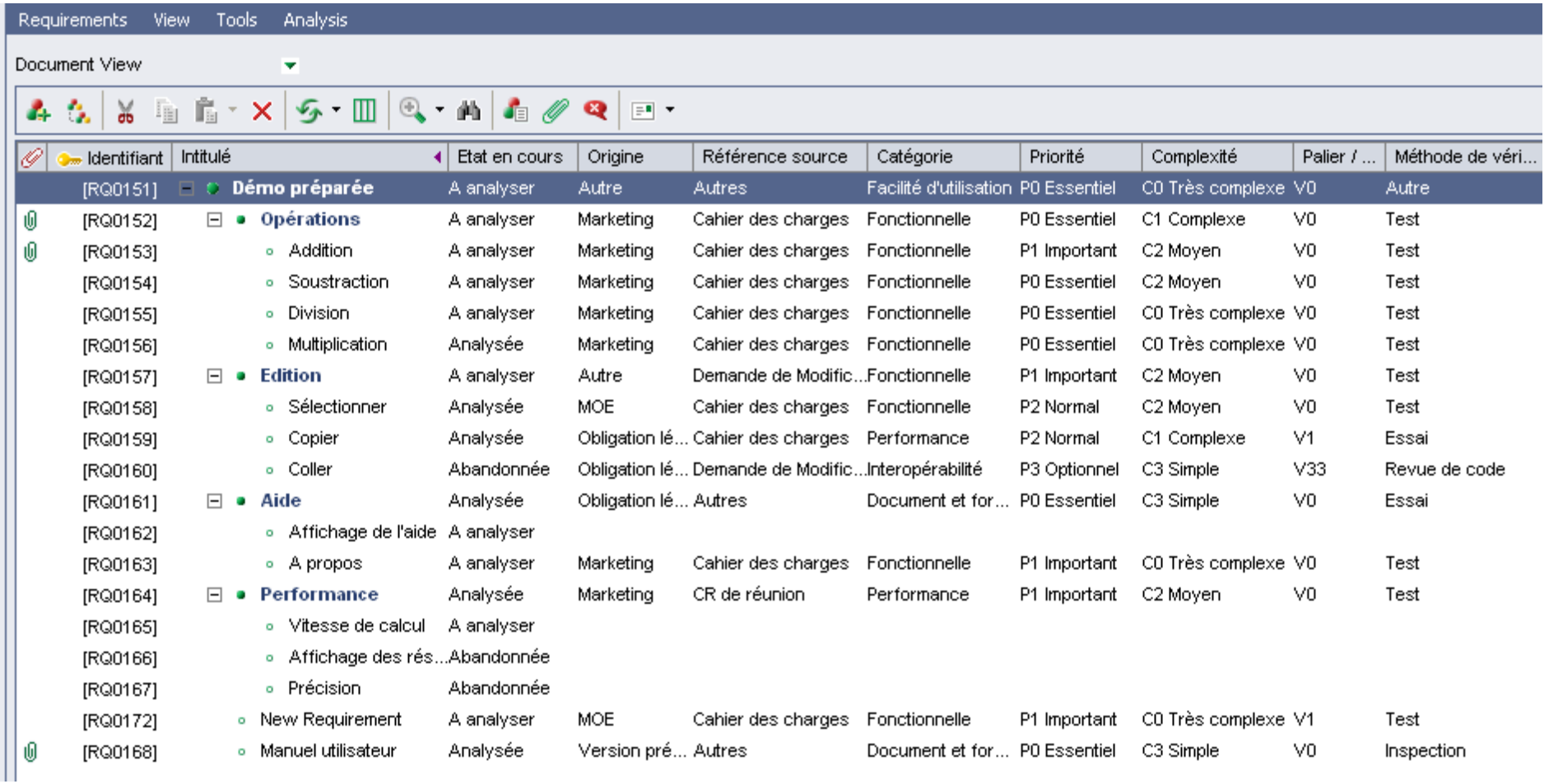
Requirements and lifecycle



Capabilities of a RE tool

- Requirements creation
  - Requirement definition with all properties
- Traceability to requirements origins
  - Links to external documents
- Links to use-case based tools (UML for ex.)
- Links to test tools
- Requirement document generation
- Coverage verification (to UML or test tools)

Tool example



	Identifiant	Intitulé	Etat en cours	Origine	Référence source	Catégorie	Priorité	Complexité	Palier / ...	Méthode de véri...
	[RQ0151]	**Démo préparée**	A analyser	Autre	Autres	Facilité d'utilisation	P0 Essentiel	C0 Très complexe	V0	Autre
📎	[RQ0152]	**Opérations**	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C1 Complexe	V0	Test
📎	[RQ0153]	Addition	A analyser	Marketing	Cahier des charges	Fonctionnelle	P1 Important	C2 Moyen	V0	Test
	[RQ0154]	Soustraction	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C2 Moyen	V0	Test
	[RQ0155]	Division	A analyser	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C0 Très complexe	V0	Test
	[RQ0156]	Multiplication	Analysée	Marketing	Cahier des charges	Fonctionnelle	P0 Essentiel	C0 Très complexe	V0	Test
	[RQ0157]	**Edition**	A analyser	Autre	Demande de Modific...	Fonctionnelle	P1 Important	C2 Moyen	V0	Test
	[RQ0158]	Sélectionner	Analysée	MOE	Cahier des charges	Fonctionnelle	P2 Normal	C2 Moyen	V0	Test
	[RQ0159]	Copier	Analysée	Obligation lé...	Cahier des charges	Performance	P2 Normal	C1 Complexe	V1	Essai
	[RQ0160]	Coller	Abandonnée	Obligation lé...	Demande de Modific...	Interopérabilité	P3 Optionnel	C3 Simple	V33	Revue de code
	[RQ0161]	**Aide**	Analysée	Obligation lé...	Autres	Document et for...	P0 Essentiel	C3 Simple	V0	Essai
	[RQ0162]	Affichage de l'aide	A analyser							
	[RQ0163]	A propos	A analyser	Marketing	Cahier des charges	Fonctionnelle	P1 Important	C0 Très complexe	V0	Test
	[RQ0164]	**Performance**	Analysée	Marketing	CR de réunion	Performance	P1 Important	C2 Moyen	V0	Test
	[RQ0165]	Vitesse de calcul	A analyser							
	[RQ0166]	Affichage des rés...	Abandonnée							
	[RQ0167]	Précision	Abandonnée							
	[RQ0172]	New Requirement	A analyser	MOE	Cahier des charges	Fonctionnelle	P1 Important	C0 Très complexe	V1	Test
📎	[RQ0168]	Manuel utilisateur	Analysée	Version pré...	Autres	Document et for...	P0 Essentiel	C3 Simple	V0	Inspection

Best known tools

- DOORS (Telelogic)
- RequisitePro (IBM/Rationale)
- Analyst Pro (Goda Software)

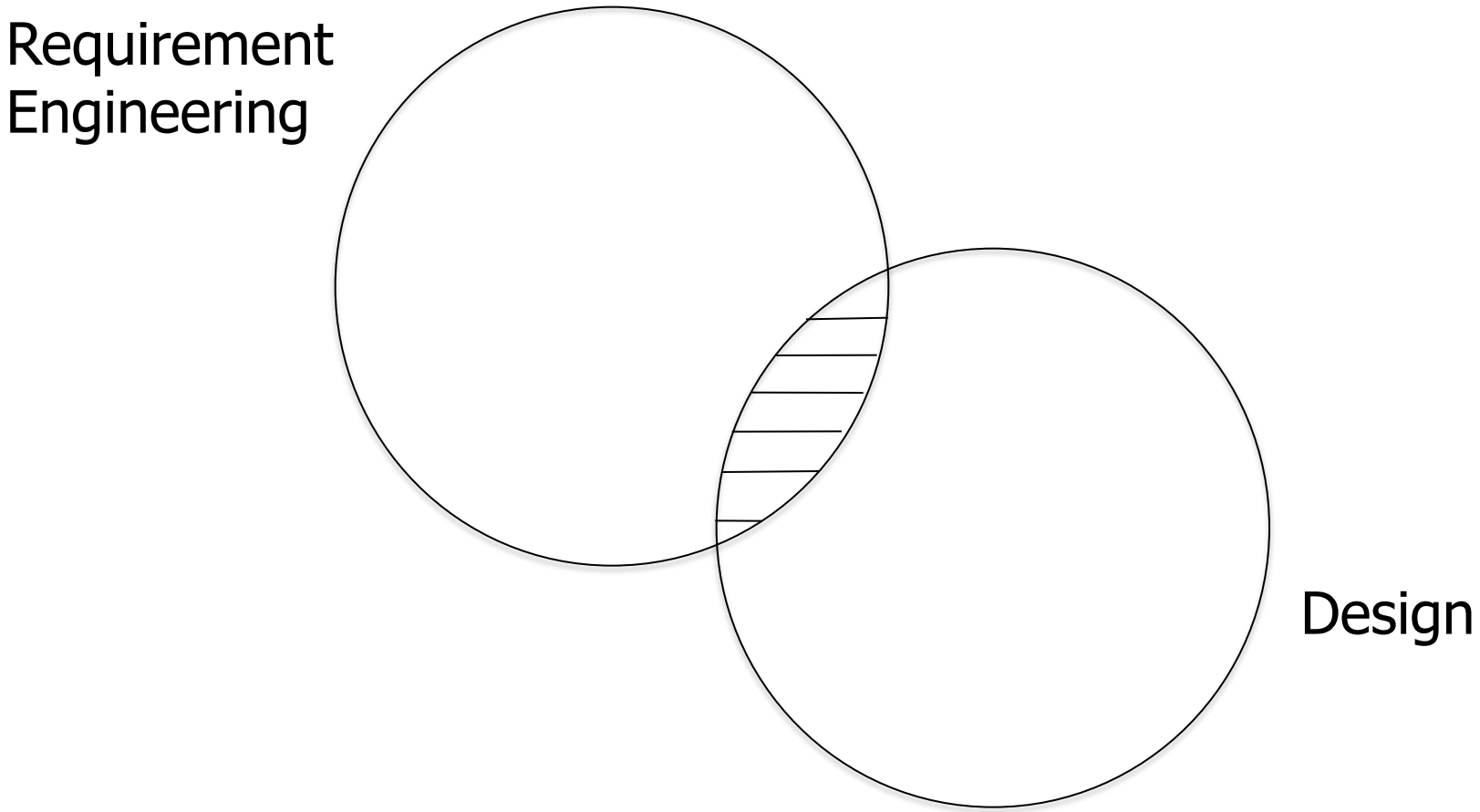
Outline

- Introduction
- Requirements
- Functional requirements
- Software quality
- Requirement engineering
- Conclusion

Purpose of requirements

Make sure that there is no room for the software supplier to get it wrong!

RE and design



Problem: RE is not popular !

- Not glamorous
- Frustrating
  - Users don't know what they want before seeing it
- Hard
  - Complex, tedious, long
- Changing
- Not immediately rewarded
  - Managers don’t like to wait for the code

Conclusion

Suddenly, a heated exchange took place between the king and the moat contractor. © Gary Larson