Introduction to Software Engineering

Requirement engineering – part I

Philippe Lalanda

Philippe.lalanda@imag.fr

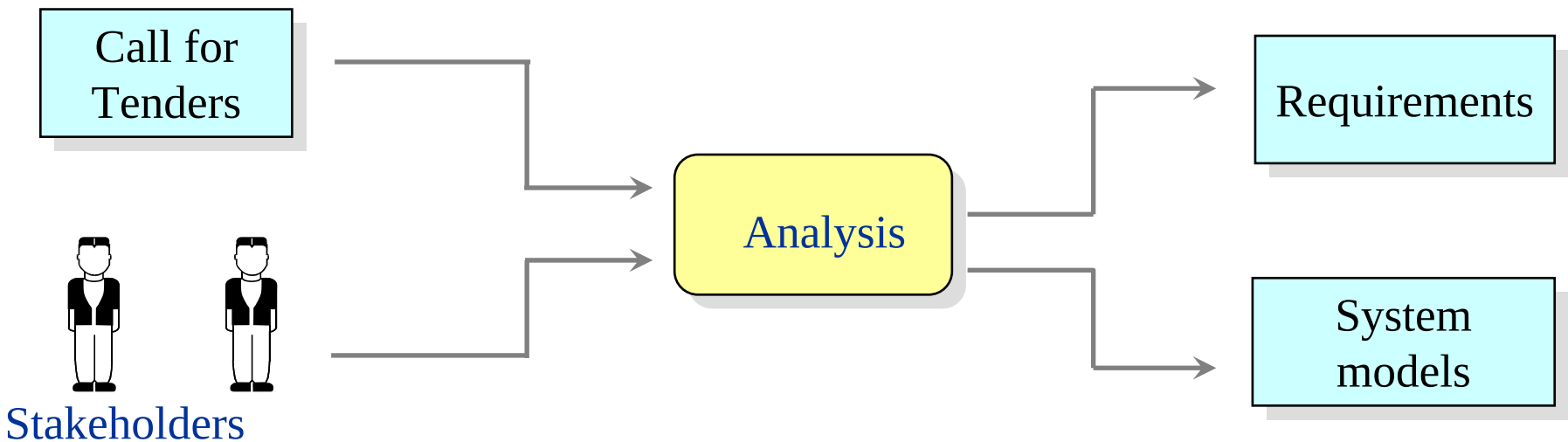
http://membres-liglab.imag.fr/lalanda/

Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

Requirements

- Objectives
  - Establish what the customer requires
  - Specify these requirements



Different requirements

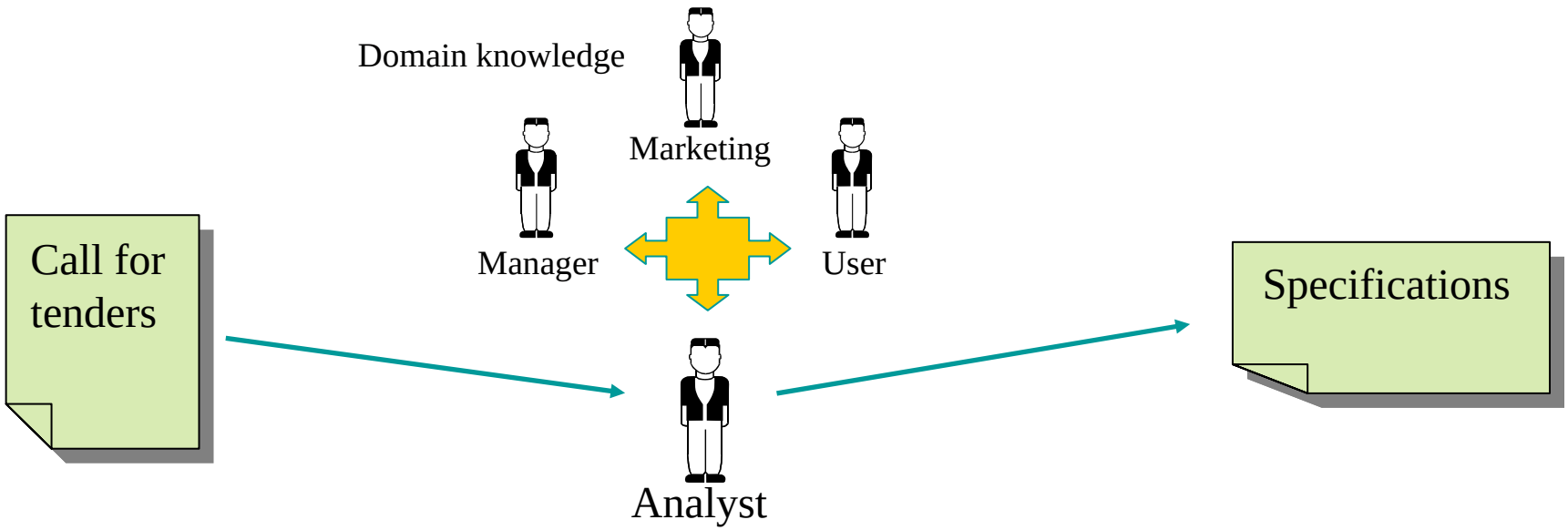
- Requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation
  - May be the basis for the contract itself - therefore must be defined in detail
  - Both these statements may be called requirements

Call for Tenders

Requirements

Requirements origin

- Requirements come from the client side
  - Users, domain experts, managers, marketing,…
- Re-formulated by the contractors
  - Analyst job



Actors

- Multiple parties, different background
  - managers and decision makers
  - domain experts
  - clients and users
  - analysts
  - architect and developers
- Conflicting viewpoints

Importance of requirement engineering

- Impacts of requirement engineering
  - Legal impact
    - it forms the basis of the contract between client and contractor
  - Economic impact
    - cost of correcting wrong requirements
  - Social impact
    - wrong requirements may cause disasters
  - Usage impact
    - acceptance or rejection of a software

Importance of requirement engineering

- Requirements influence all the software production activities
  - Architecture
  - detailed design
  - test definition
  - acceptance, …
- In UML, use cases drive test definition

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
  - RE is bugged by internal conflicts
  - Constantly shifting compromise in the requirements
  - Clients needs and wishes evolve
  - Analysts need human and technical skills
  - Some problems can never be fully understood and understanding evolves during the system development

Possible flaws

- A wide variety of flaws can occur
  - incompleteness
  - inconsistency
  - Inadequacy
  - ambiguity
  - unintelligibility
  - poor structure
  - over specification

Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

What is a requirement ?

- A requirement is a capability or a constraint
  - A service provided by the software
  - A constraint under which it operates and is developed
- Requirements express
  - everything the customers want
  - everything that is necessary to the production of the software

Beware

- Beware of the saying “what versus how”

Capabilities (functional requirement)

- Provided services
  - The description of a function or its behavior
  - A general property of the system
  - Expected HMI, …
- Examples
  - The software has to manage the library loan system
  - A subscriber has to pay 20 euros every year
  - The software can present the list of books borrowed by a given subscriber, …

Constraint (non functional requirement)

- A constraint under which a software operates and is developed

Quality

- Performance
- Reliability
- Usability
- Maintainability

Development

- COTS (OS, middleware, …)
- Methods
- Tools
- Standards

Domain

- Usage
- Regulation (law)

Multiple abstraction levels

- Range from a high-level abstract statements to detailed mathematical functional specification
- Three levels of abstraction
  - Domain requirements
  - User requirements
  - System requirements

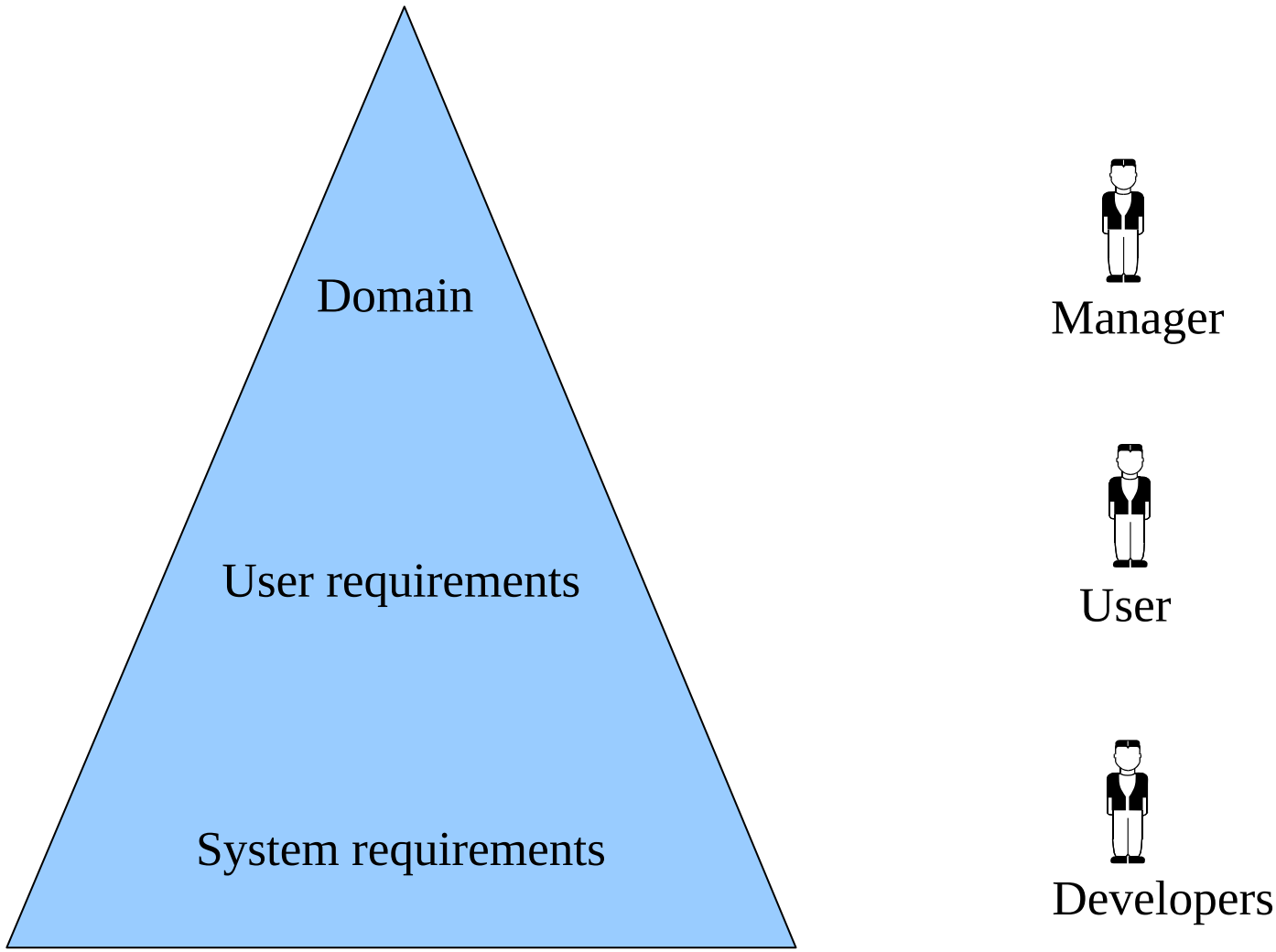
Domain / user / system requirements

- Domain requirements
  - Properties of the domain influencing the target software
  - Written for customers
- User requirements
  - A statement in natural language plus diagrams of the services the system provides and its operational constraints.
  - Written for customers
- System requirements
  - A structured document setting out detailed descriptions of the system services.
  - Written as a contract between client and contractor

Examples

- Domain requirement
  - The system must deal with copyright management
- User requirement
  - 1. The system must keep track of all data related to copyright
- System 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

Multiple abstraction levels



Requirements features

- A requirement has to be
  - Concise
  - Non ambiguous
  - Understandable by both the user and the contractor
  - Structured
  - Verifiable
    - Requirements are written in a way compatible with verification methods under use

Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

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 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 system concept, not an implementation concept
  - Only concepts related to the software are to be considered → not a domain analysis
- Requirements define objects and their scope
  - A client is defined by his name, age, address
  - A client has to be ten or more
  - A book is defined by its title and author(s)

Functions – from Davis, 93

- A function is a clearly defined activity
  - Tasks, services, processes
  - Only activities realized by the software are to be considered → not a domain analysis
- Requirements define functions in details
  - The software can display clients information
  - When displaying clients information, id is given first, on the top of the window
  - To borrow a book, a client has first to give his library card and then the different books he desires

States – from Davis, 93

- A state characterizes the situation of an entity (an object, a function, the system)
  - Can be expressed as a predicate
  - Influences the behavior of the entity
  - Temporal aspect
    - A state can be transient
    - A state can capture an historic
- Requirements define states in detail
  - 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

- Some 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 ongoing loans

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

Questions to be asked – from Pfleeger

- Functionality
  - What will the system do?
  - When will the system do it?
  - Are there several modes of operation?
  - What are the good reactions to possible stimuli?
- Data
  - What should be the format of data?
  - Must any data be retained for any period of time?

Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

Non functional requirements

- Qualities relate to global properties
- Hard to elicit and hard to verify
  - Many interpretations are possible
- It is necessary to quantify these requirements
  - With metrics that can be measured, tested, …

Quantification

- NF requirements relate to global properties which are hard to verify (in first formulation)
  - Not precise, many interpretations are possible
    - “The system must be easy to use”
  - Source of conflicts
    - “The system has to be robust and efficient”
- 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

- Questions to be asked
  - Are there constraints on execution speed, response time or throughput?
  - What efficiency measures will apply to resource usage and response time?
  - How much data will flow through the system?
  - How often will data be received or sent?

Performance

- Examples of requirements
  - Number of transactions per second
  - Refreshing time
  - Response time for a given pattern of events
- To be précised
  - For each input, the range of values
  - Arrival rate of inputs
  - What to do when expected quantities are exceeded
    - Failure, ignorance of additional inputs, degraded services

Usability

- Questions to be asked
  - What kind of training will be required for each type of user?
  - How easy should it be for a user to understand and use the system?
  - How difficult should it be for a user to misuse the system?

Usability

- Example of requirements
  - Interfaces
  - Error messages
  - Techniques needed to help users and improve confidence
- There are today tools to rapidly prototype interfaces

Reliability and availability

- Must the system detect and isolate faults?
- What is the prescribed meantime between failures?
- Is there a maximum time allowed for restarting the system after a failure?
- How often will the system be backed up?
- Must backup copies be stored at a different location?
- Should precaution be taken against fire damage?

Availability

- Examples of requirements
  - Max. number of bug per Kline during integration
  - Min. duration without a problem

Security

- 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

- Will maintenance merely correct bugs or will it also include improving the system?
- When and in what ways might the system be changed in the future?
- How easy should it be to add features to the system?
- How easy should it be to port the system from one platform (computer, OS) to another?

Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

Requirements traceability

- Many links to be maintained
  - Between requirements and original needs
  - Between abstract requirements and derived requirements
  - Between requirements and implementation
  - Between dependant requirements (same abstract level)
  - Between requirements and tests

- Note: requirements evolution is unavoidable

Some fun

- Un jour un constructeur automobile décida de réduire les coûts sur l'un de ses modèles phares.
- Une équipe se pencha sur les spécifications du modèle et chercha des axes de réduction des coûts.
- Quelqu'un s'avisa que le modèle était conçu pour résister à un vent arrière, avec de la pluie, de 200km/h (exigence produit) ce qui entrainait des coûts de fabrication importants.
- On décida donc de changer cela en allégeant la fermeture du coffre à bagage situé à l'arrière (exigence composant).
- Ce n'est qu'à l'automne, chez les concessionnaires, qui trouvaient de l'eau dans les coffres, que l'on s'avisa que les voitures étaient acheminées par train Express (exigence partie prenante).

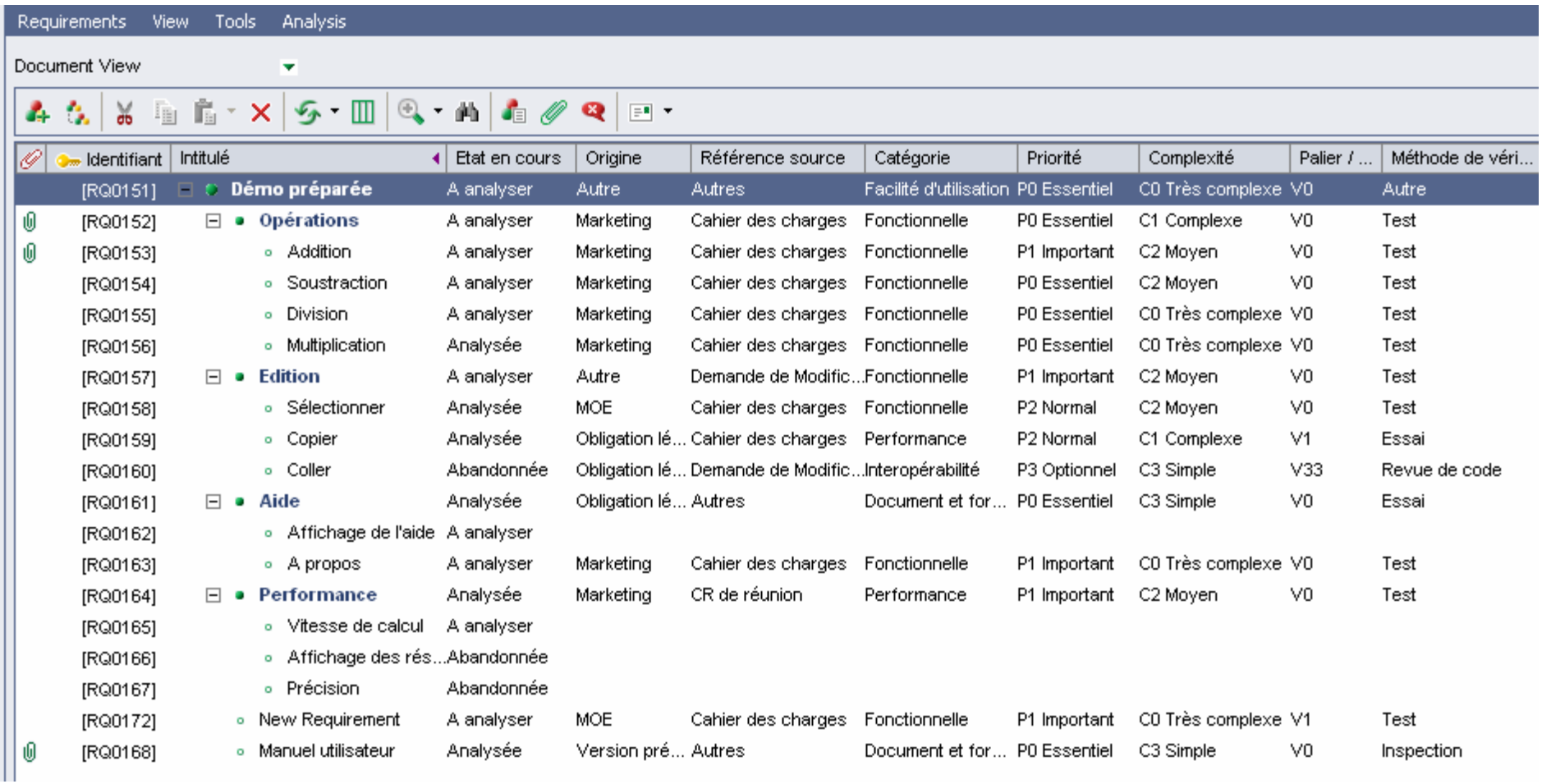
Importance of traceability

- Traceability allows
  - To go back to primary needs
  - To assess the cost of an evolution
    - Down to the code
  - To check the coverage
    - Relevance of requirements
    - Progress of the project
- Tooling is necessary

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)

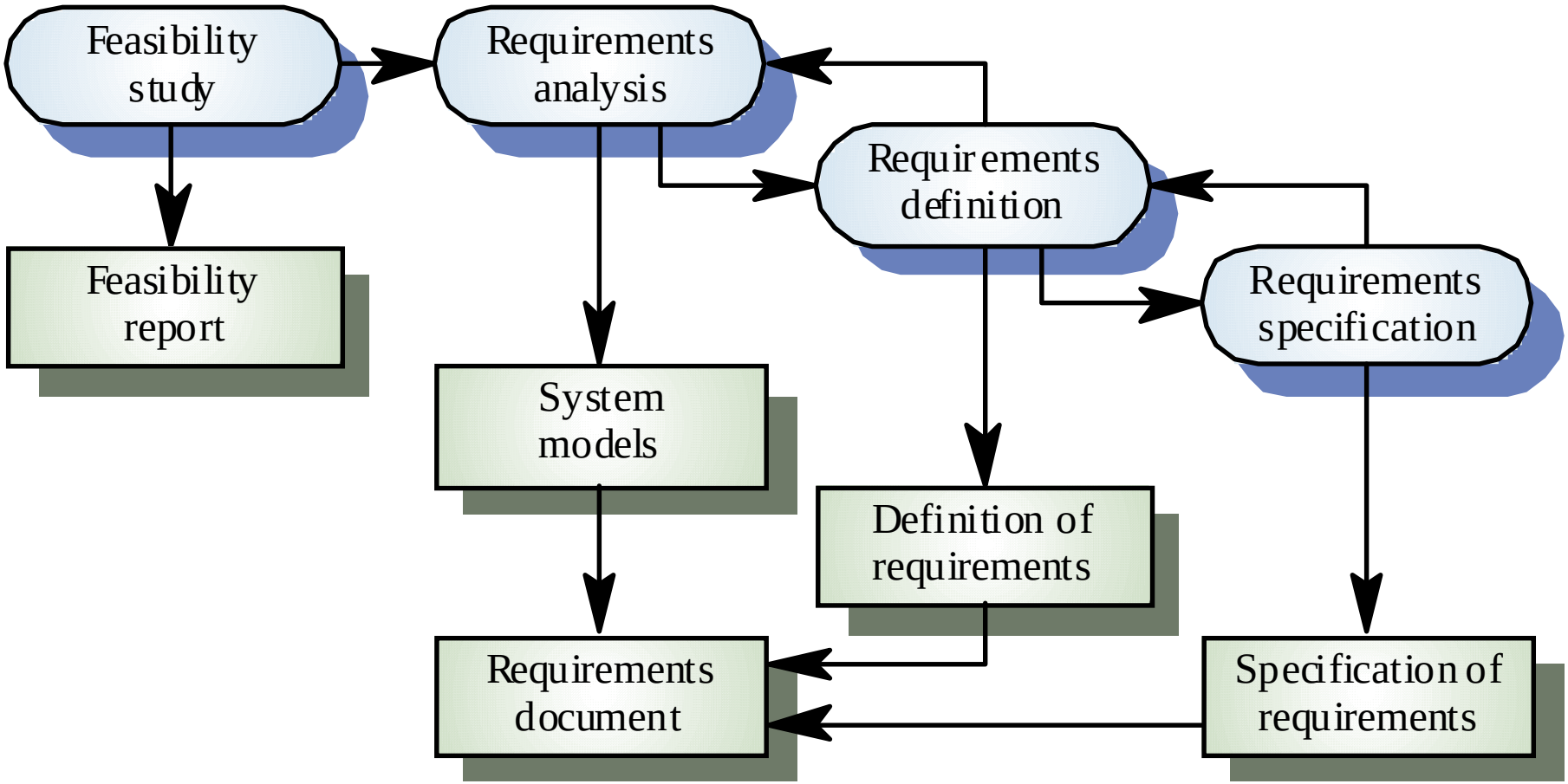
Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

Intertwined processes

- Feasibility
- Analysis
- Definition
- Specification
- Verification, validation,
- Evolution management

An iterative process – from Sommerville



Feasibility study

- Short and very focused study
- The goal is to answer the following questions
  - Does the system answer business goals ?
  - Can the system be realized with current techno ?
  - Can the system be integrated with existing systems?
  - Can available tools be used ?
- Documents
  - In: call for tenders
  - Out: report (go/no-go) with recommendations

Feasibility study: example

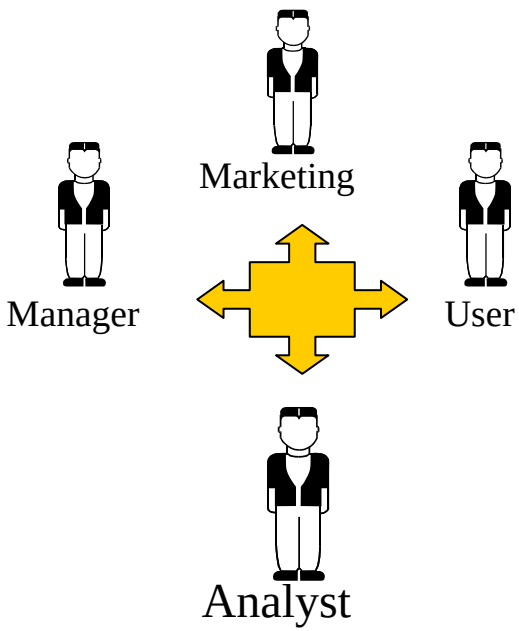
- CISCO considers the smart gateway market
- Marketing side
  - Who are the main actors ?
  - Who are the new actors ?
  - What are the profits (margins) ?
  - Evaluation of gains and risks ?
- Technical side
  - Can existing hardware be reused ?
  - Can dev processes be reused ?
  - Is a new embedded database needed ?
  - Evaluation of costs and potential issues ?

Analysis

- The purpose is to gather information about the software
  - Understand the domain and computing environment
  - Identify goals 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

Analysis: example

- After the feasibility study, the CISCO project is launched. The purpose of the analyst is to write down requirements
- Problem understanding
  - Discussion with marketing
  - Meetings with FT
  - Meetings with users
  - Meetings with dev parties



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

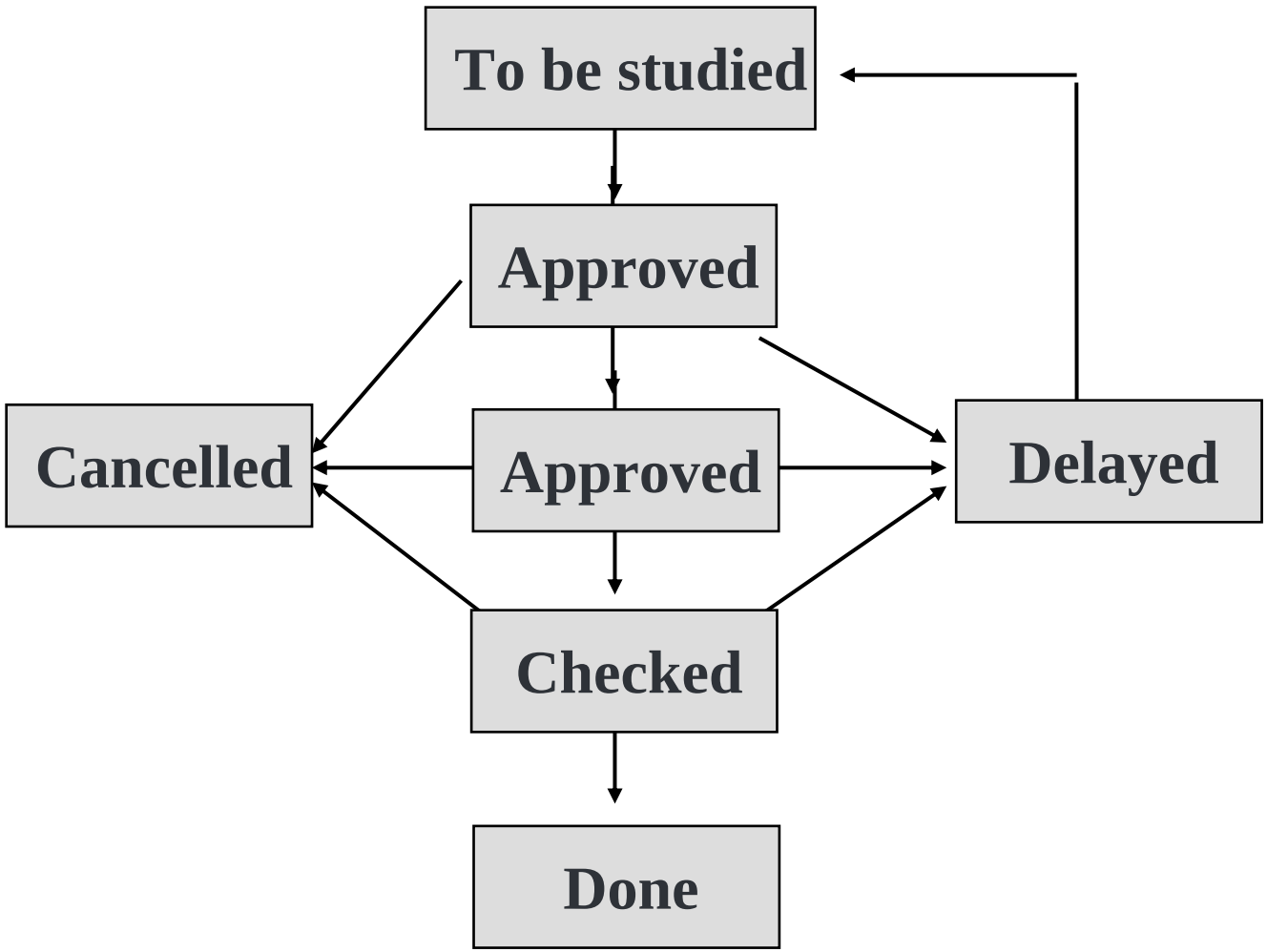
Requirements specification

- The purpose is to clearly describe the software to produce
  - Documentation in form understandable by all parties -> contractual basis
- 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

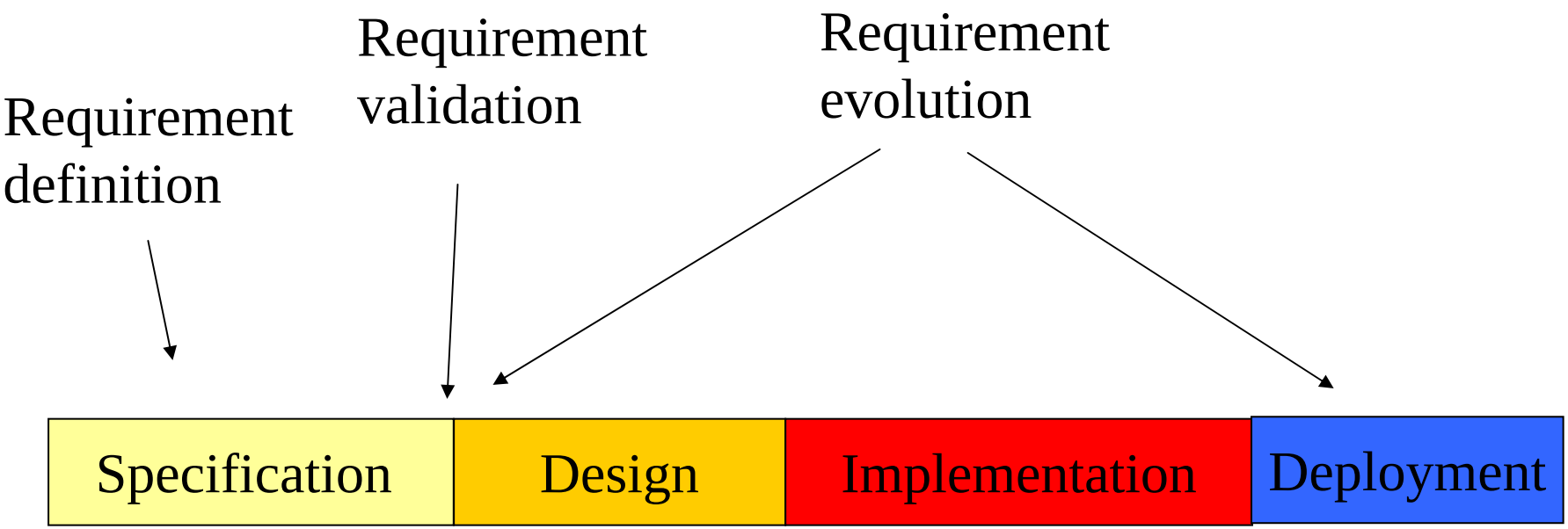
Requirement states



Flaws in spec. document – from Van Lamsweerde

- Inadequacy, omission, contradiction, ambiguity
- Noise: yield no information
- Forward reference: use of not yet defined feature
- Remorse: incident statement of a requirements (in parenthesis)
- Over specification: solution space
- Wishful thinking: item stating some feature that cannot be assessed, tested or verified

Requirements and lifecycle



Requirement engineering

Requirements evolution

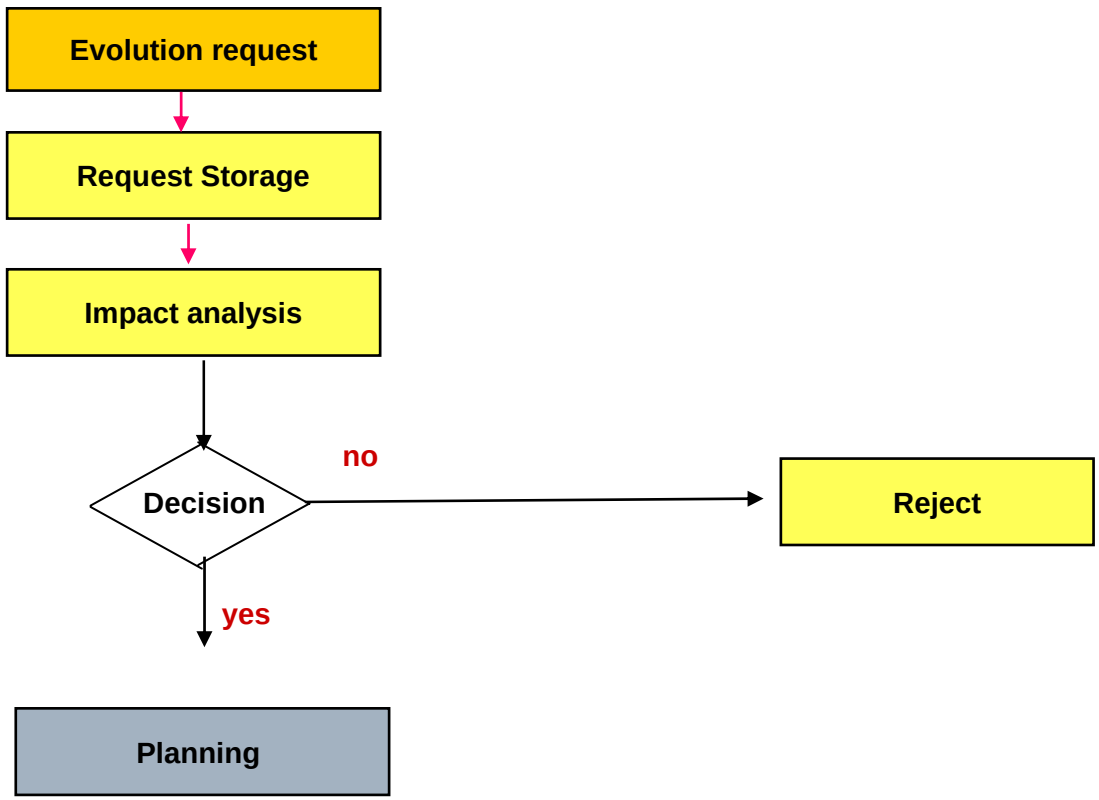
- Requirements evolution is unavoidable
- Doable if the initial work of collect, analysis and validation has been performed rigorously
- Impacts have to be assessed
  - Traceability mechanisms are necessary
- A tool is necessary
  - At least to automate traceability links management

Requirements evolution issues

- Requirements evolution without impact analysis
- Requirements modification by the hierarchy
- New stakeholders (very frequent)
- No formalism, no process for process evolution
- Weak traceability

- A good initial process is not enough: an evolution management process is necessary

Requirements evolution process



Outline

- Introduction
- Requirements
- Functional requirements
- Non functional requirements
- Requirements traceability
- Requirement engineering
- Conclusion

Requirements errors – from Van Lamsweerde

- The most numerous
  - 33% of software errors
- The most persistent
  - Often found after delivery
- The most expensive
  - detection/fix costs 5x more during design, 10x more during implementation, 20x more during testing, 200x more after delivery
- The most dangerous

Some figures

- Major sources of failure
  - lack of user involvement 13%
  - incomplete requirements 13%
  - changing requirements 9%
  - unrealistic expectations 10%
  - unclear goals 5%

  - See www.standishgroup.com

Synthesis – from Van Lamsweerde

- A few confusions
  - requirements are *not* domain properties
  - requirements are *not* software specifications
  - requirements are problem formulations, *not* solution formulations (i.e. design specs)
  - RE is *not* translation of pre-existing problem formulations
  - composition is *not* necessarily conjunction
  - "precise" does *not* necessarily mean "formal"
  - a set of notations is *not* sufficient for a "method"

RE is not popular !

- 10 Top reasons (http://www.volere.co.uk/tentopreasons.htm)
  - 10. We don't need requirements, we're using objects ...
  - 9. The users don't know what they want
  - 8. We already know what the users want
  - 7. Who cares what the users want?
  - 6. We don't have time to do requirements
  - 5. It's too hard to do requirements
  - 4. My boss frowns when I write requirements
  - 3. The problem is too complex to write requirements
  - 2. It's easier the change the system later …
  - 1. We have already started writing code: we don't want to spoil it

Conclusion

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

© Gary Larson