

Requirement Engineering

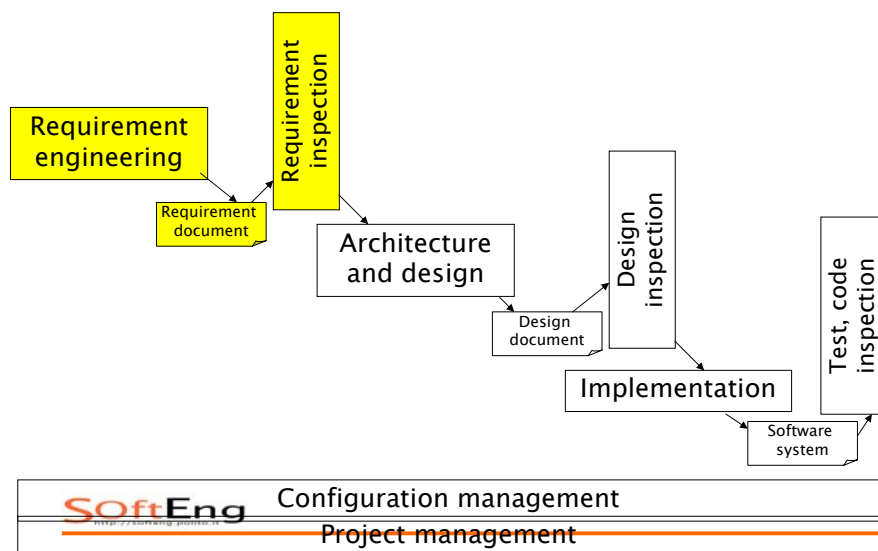


Outline

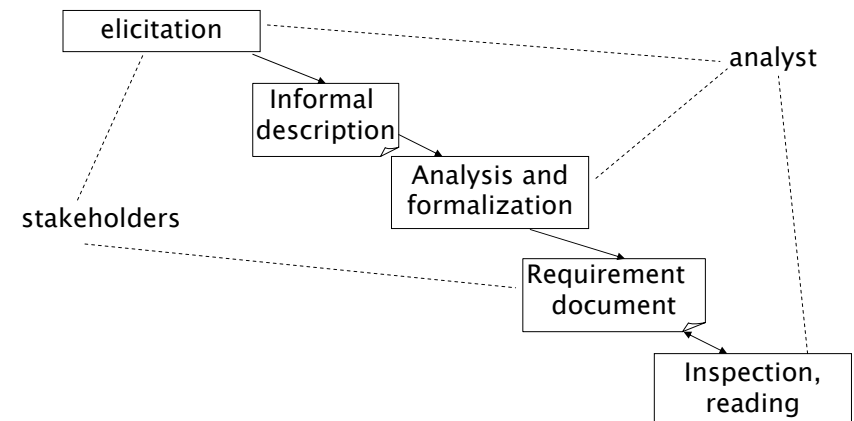
- Stakeholders
- Types of requirements
- Context diagram and interfaces
- Numbering requirements
- Scenarios
- Use cases
- Glossary



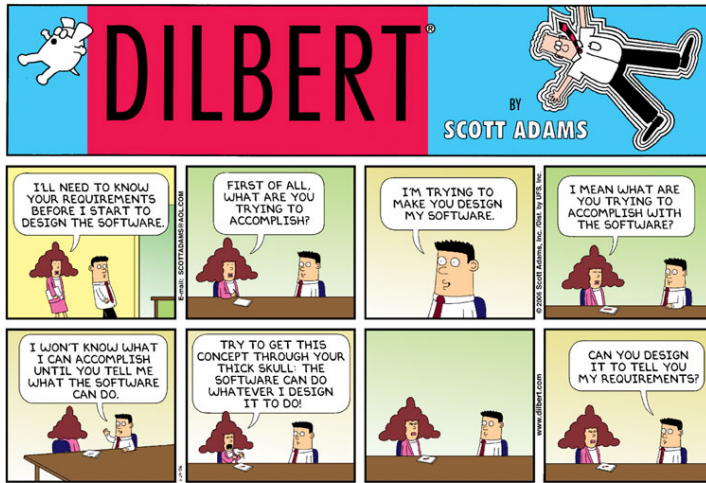
The process



Requirement engineering



RE



© Scott Adams, Inc./Dist. by UFS, Inc.

SoftEng
<http://softeng.unibz.it>

Stakeholder

- Role or person with an interest (stake) in the system to be built
- ◆ User
 - Uses the system
 - Can include different user profiles
- ◆ Buyer
 - Pays for the system
- ◆ Administrator
- ◆ Analyst
 - Expert in requirement engineering, and or in domain
- ◆ Developer

SoftEng
<http://softeng.unibz.it>

Stakeholders – example

- Point Of Sale in a supermarket
- User
 - ◆ Clerk at POS (profile 1)
 - ◆ Supervisor, inspector (profile 2)
 - ◆ Customer at POS (indirectly through clerk)
- Administrator
 - ◆ POS application administrator (profile 3)
 - ◆ IT administrator (profile 4)
 - Manages all applications in the supermarket
 - ◆ Security manager (profile 5)
 - Responsible for security issues
 - ◆ DB administrator (profile 6)
 - Manages DBMSs on which applications are based
- Buyer
 - ◆ CEO and/or CTO of supermarket

SoftEng
<http://softeng.unibz.it>

Requirement

- Description of a system / service and of constraints on it
 - ◆ Different levels of abstraction
 - ◆ Different levels of formality

SoftEng
<http://softeng.unibz.it>

Requirements – informal

- A POS (Point-Of-Sale) system is a computer system typically used to manage the sales in retail stores. It includes hardware components such as a computer, a bar code scanner, a printer and also software to manage the operation of the store.
- The most basic function of a POS system is to handle sales.

Requirements imprecision

- ♦ Problems arise when requirements are not precisely stated.
- ♦ Ambiguous requirements may be interpreted in different ways by developers and users.
- ♦ ‘manage sales’ ‘handle sales’?

Completeness and consistency

- In principle, requirements should be both complete and consistent.
 - ♦ Complete
 - They should include descriptions of all facilities required.
 - ♦ Consistent
 - There should be no conflicts or contradictions in the descriptions of the system facilities.
 - ♦ In practice, it is impossible to produce a complete and consistent requirements document.

Requirements – defects

- Omission/ incompleteness
- Incorrect Fact
- Inconsistency/contradiction
- Ambiguity
- Extraneous Information
 - ♦ Overspecification (design)
- Redundancy

Requirement techniques

- Context diagram
- Requirement numbering and type
- Glossary
- Use case diagram (UML)
- Scenario, sequence diagram (UML)
- System diagram (UML)
- Class diagram (UML)

Requirement doc structure

- Overall description
- Stakeholders
- Context diagram and interfaces
- Requirements
 - ◆ Functional
 - ◆ Non functional
 - ◆ Domain
- Use case diagram
- Scenarios
- Class diagram
- Glossary

IEEE requirements standard

- IEEE Std 830 1984
- Defines a generic structure for a requirements document that must be instantiated for each specific system.
 - ◆ Introduction.
 - ◆ General description.
 - ◆ Specific requirements.
 - ◆ Appendices.
 - ◆ Index.

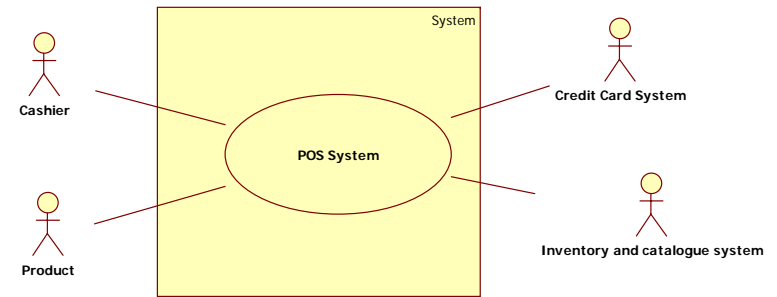
Other requirement templates

- <http://readysset.tigris.org>

Context diagram

- Defines what is **inside** the system to be developed, what is **outside**
 - Other systems/subsystems/applications
 - Human users
- Defines **interfaces** between inside and outside

Context diagram



Interfaces

- With cashier
 - ♦ Physical level: Screen, keyboard
 - ♦ Logical: GUI (to be described)
- With product
 - ♦ Physical level: laser beam (bar code reader)
 - ♦ Logical level: bar code
- With credit card system
 - ♦ Physical level: internet connection
 - ♦ Logical: web services (functions to be described, data exchanged, SOAP + XML)

Interface specification

- ♦ Three types of interface may have to be defined
 - User interfaces, GUIs
 - Procedural interfaces;
 - Data exchanged;
- ♦ Formal notations are an effective technique for interface specification.

PDL interface description

```
interface PrintServer {  
  
    // defines an abstract printer server  
    // requires: interface Printer, interface PrintDoc  
    // provides: initialize, print, displayPrintQueue, cancelPrintJob, switchPrinter  
  
    void initialize ( Printer p );  
    void print ( Printer p, PrintDoc d );  
    void displayPrintQueue ( Printer p );  
    void cancelPrintJob (Printer p, PrintDoc d );  
    void switchPrinter (Printer p1, Printer p2, PrintDoc d );  
} //PrintServer
```

Data interface (XML)

```
<food>  
<name>Belgian Waffles</name>  
<price>$5.95</price>  
<description>  
two of our famous Belgian Waffles with plenty of real maple syrup  
</description>  
<calories>650</calories>  
</food>
```

```
<food>  
<name>Strawberry Belgian Waffles</name>  
<price>$7.95</price>  
<description>  
light Belgian waffles covered with strawberries and whipped cream  
</description>  
<calories>900</calories>  
</food>
```

GUI interface

- Sketch of interface, typically built with GUI builder



Requirement types

- Functional
 - ♦ Description of services / behaviors provided by the system
 - ♦ Application vs. domain
- Non functional
 - ♦ Constraints on the services
 - ♦ Application vs. domain
- ♦ (Domain
 - From the domain (= set of related applications, ex banking, telecom))

Functional

Requirement ID	Description
F1	Handle sale transaction
F2	Start sale transaction
F3	End sale transaction
F4	Log in
F5	Log out
F6	Read bar code

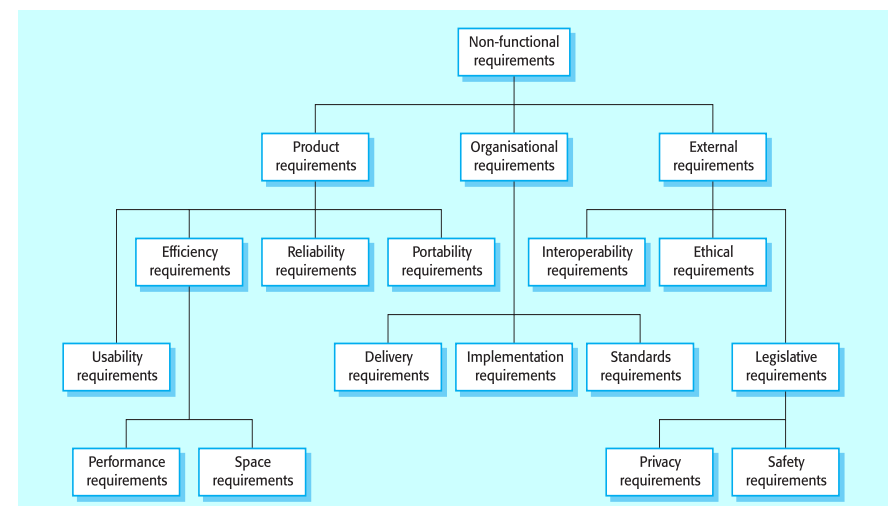
Non Functional (ISO 9126)

- Efficiency
- Usability
- Reliability
- ...

ISO 9126

- ♦ Defines 6 properties of software systems
 - 5 non functional
 - Functionality
 - Reliability
 - Usability
 - Efficiency
 - Maintainability
 - Portability

Non-functional requirements



Non-functional reqs

- **Product requirements**
 - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- **Organisational requirements**
 - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- **External requirements**
 - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Non Functional

Requirement ID	Description
NF1(efficiency)	Function F1.1 less than 1msec
NF2 (efficiency)	Each function less than ½ sec
Domain1	Currency is Euro – VAT is computed as ..

Non-functional requirements

- ♦ Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

NF Req should be measurable

- **Not measurable**
 - ♦ The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- **Measurable**
 - ♦ Experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

Measures for NF reqs

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	M Bytes Number of ROM chips
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

Requirements interaction

- Conflicts between different non-functional requirements are common in complex systems.
- Spacecraft system
 - To minimise weight, the number of separate chips in the system should be minimised.
 - To minimise power consumption, lower power chips should be used.
 - However, using low power chips may mean that more chips have to be used. Which is the most critical requirement?

Domain requirements

- ♦ Derived from the application domain and describe system characteristics and features that reflect the domain.
- ♦ Domain requirements can be new functional requirements, constraints on existing requirements or define specific computations.
- ♦ If domain requirements are not satisfied, the system may be unworkable.

Train protection system

- The deceleration of the train shall be computed as:
 - ♦ $D_{\text{train}} = D_{\text{control}} + D_{\text{gradient}}$
where D_{gradient} is 9.81ms^2 * compensated gradient/alpha and where the values of 9.81ms^2 /alpha are known for different types of train.

Domain req. problems

- Understandability
 - ♦ Requirements are expressed in the language of the application domain;
 - ♦ This is often not understood by software engineers developing the system.
- Implicitness
 - ♦ Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Glossary

- Sale = commercial transaction between customer and retailer, where customer buys a number of products from the retailer
- Product = ..

Scenario

- Sequence of steps that describe a typical interaction with the system

Step	Description
1	Start sales transaction
2	Read bar code
3	Retrieve name and price given barcode
	Repeat 2 and 3 for all products
4	Compute total
5	Manage payment cash
6	Deduce stock amount of product
7	Print receipt
8	Close transaction

Use case

- Set of scenarios with common user goal

Styles in RD

- Operational, descriptive
- Formal, semiformal, informal
- System and software (non qui)

V&V of requirements

- Natural language, UML
 - ♦ Inspection, reading
 - By user, by developer
- UML
 - ♦ Some syntactic check by tools
- Prototyping
- Formal language
 - ♦ Model checking

- ♦ (see V&V chapter)

Tools

- RequisitePro, Doors, Serena RM
- Word, Excel
- UML tools
 - ♦ Powerpoint, Visio, specialized tools (StarUML)

Summary

- Goal of requirement engineering is describing *what* the system should do in a requirement document
- Many stakeholders are involved in the process
- Techniques to make the document more precise are
 - ♦ Context diagram and interfaces
 - ♦ Identifying requirements and classifying them (functional, non functional, domain)
 - ♦ Scenarios
 - ♦ Use cases

Summary

- Requirements engineering is a key phase
 - ♦ Most defects come from this phase, and they are the most disruptive and most costly to fix
- Verification and validation is essential
 - ♦ Inspection
 - ♦ prototype

Appendix

Domain

- Collection of related functionality
or
- collection of applications with similar functionality)
 - ♦ Ex. banking, that includes subdomains account management, portfolio management, etc
 - ♦ Ex. telecommunication, that includes subdomains switching, protocols, telephony, switching

Application

- Or system
- Software system supporting a specific set of functions. Belongs to one or more domains

Other techniques for RE

Structured presentation

2.6.1 Grid facilities

The editor shall provide a grid facility where a matrix of horizontal and vertical lines provide a background to the editor window. This grid shall be a passive grid where the alignment of entities is the user's responsibility.

Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities 'snap-to' grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6

Source: Ray Wilson, Glasgow Office

Form-based specifications

- ◆ Definition of the function or entity.
- ◆ Description of inputs and where they come from.
- ◆ Description of outputs and where they go to.
- ◆ Indication of other entities required.
- ◆ Pre and post conditions (if appropriate).
- ◆ The side effects (if any) of the function.

Form-based

Insulin Pump Control Software/SRS/3.3.2

Function Compute insulin dose: Safe sugar level

Description Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r_2), the previous two readings (r_0 and r_1)

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose Š the dose in insulin to be delivered

Destination Main control loop

Action: CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requires Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r_0 is replaced by r_1 then r_1 is replaced by r_2

Side-effects None

<http://softeng.pml.co.uk>

Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.

SoftEng

<http://softeng.pml.co.uk>

Tabular specification

Condition	Action
Sugar level falling ($r_2 < r_1$)	CompDose = 0
Sugar level stable ($r_2 = r_1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r_2 - r_1) < (r_1 - r_0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ($(r_2 - r_1) \geq (r_1 - r_0)$)	CompDose = round $((r_2 - r_1) / 4)$ If rounded result = 0 then CompDose = MinimumDose

<http://softeng.pml.co.uk>