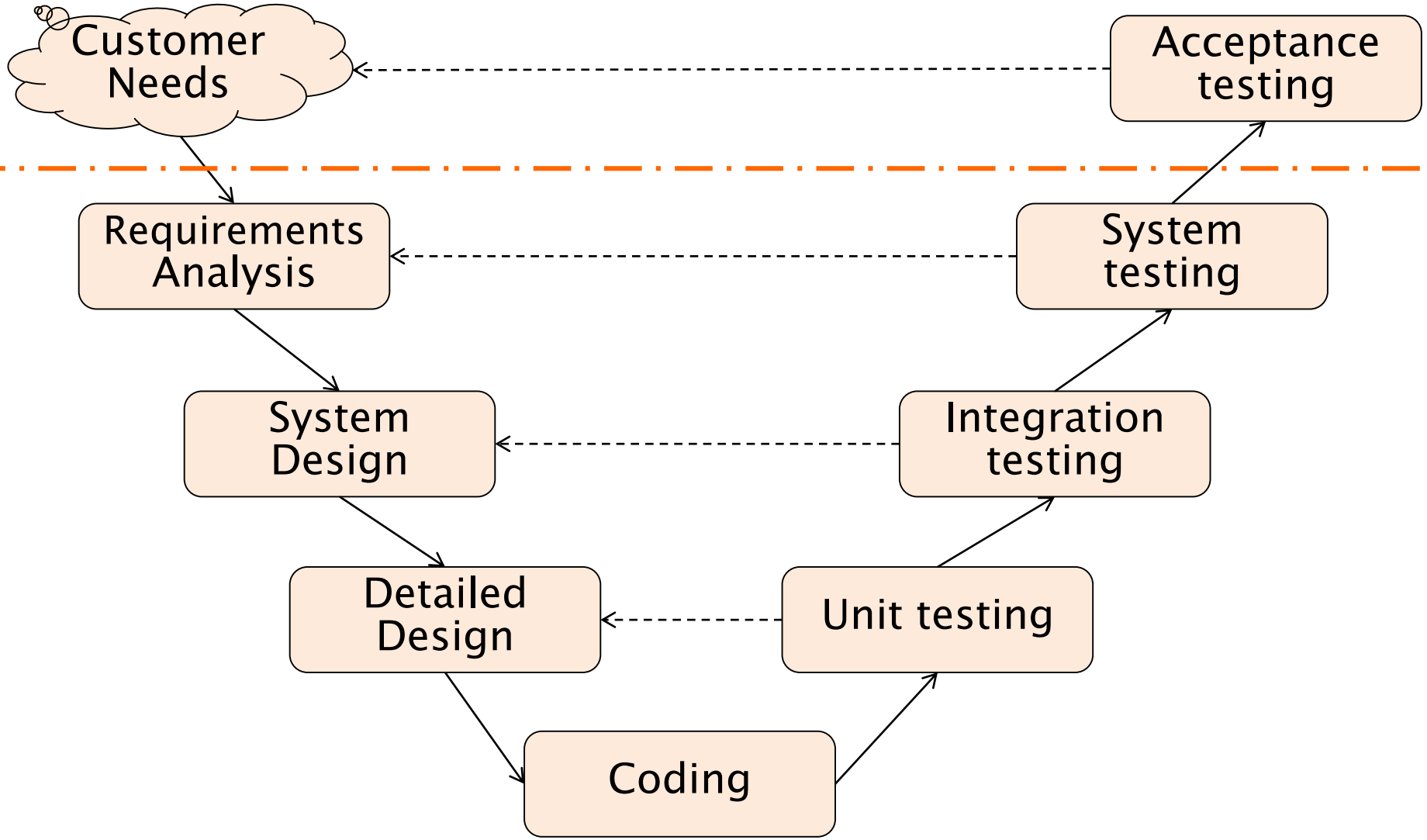
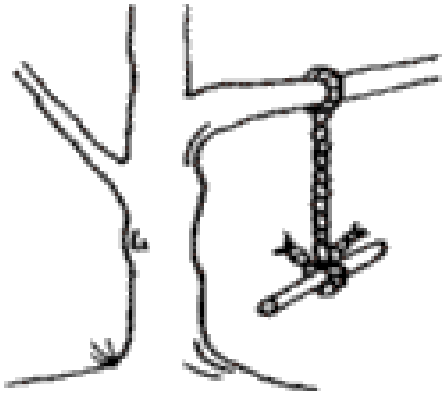
Requirements Engineering

SOftEng
http://softeng.polito.it

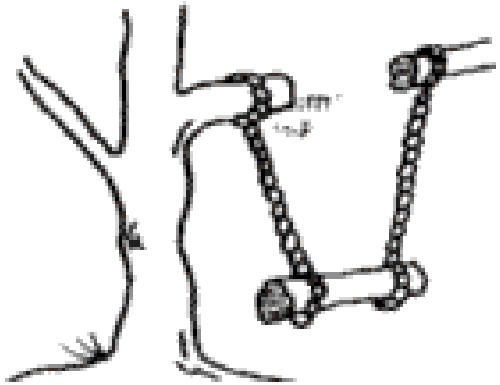
Version 1.11 – 26 October 2016

Software development

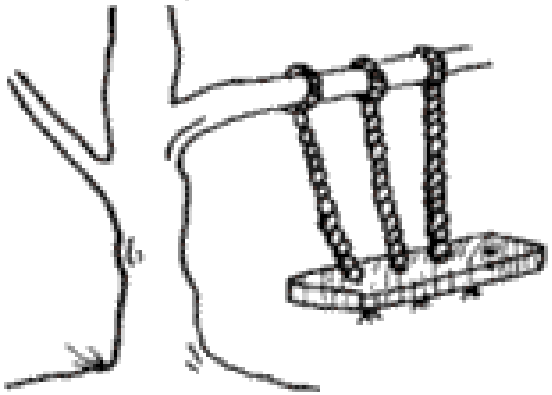




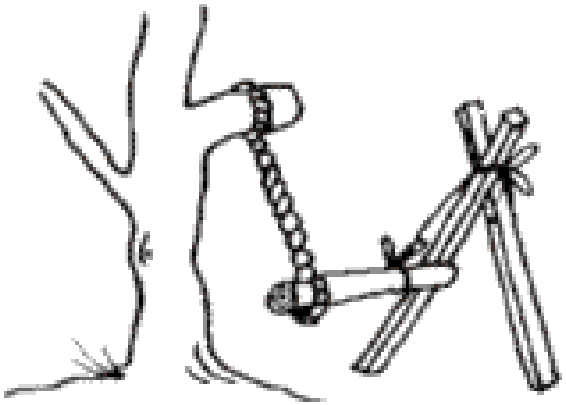
What the user asked for



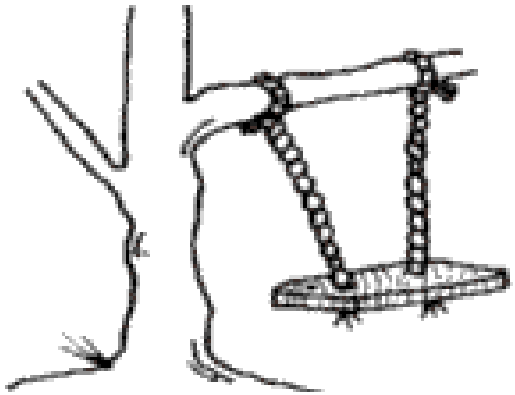
How the analyst saw it



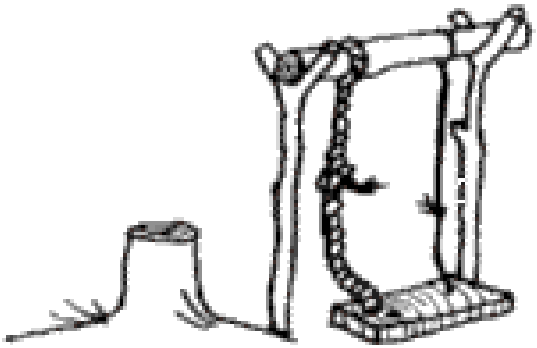
How the system was designed



As the programmer wrote it



What the user really wanted



How it actually works

Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

Activities in req. engineering

- Elicitation
- Analysis
- Formalization
- V&V (verification and validation)

Requirements vs. Design

- Requirements

  What the system should do

- Design

  How the system is structured

What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as 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.

Types of requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements (a.k.a. developer requirements)
  - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

Definitions and specifications

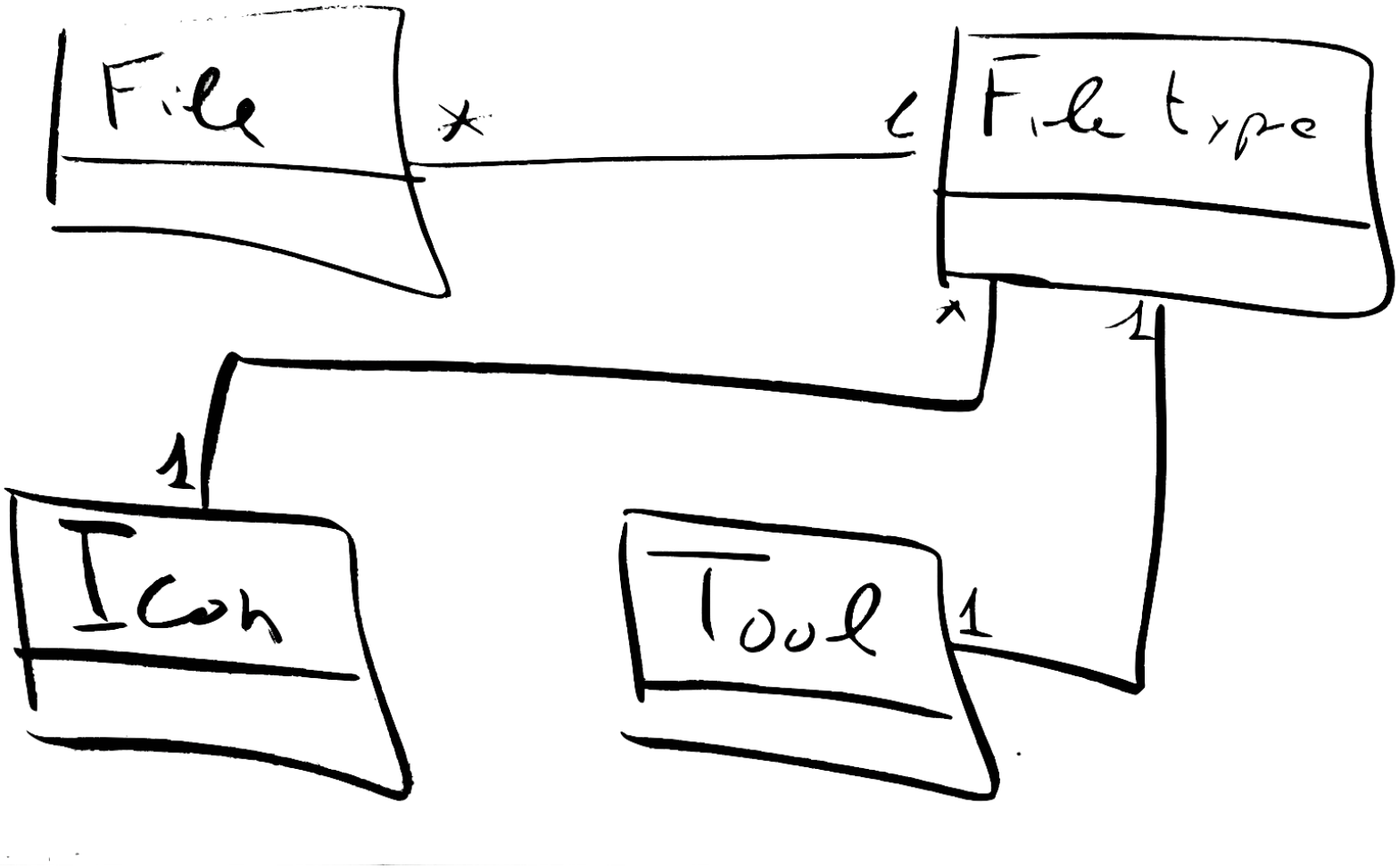
User requirement definition

The software must provide a means of representing and accessing external files edited by other tools

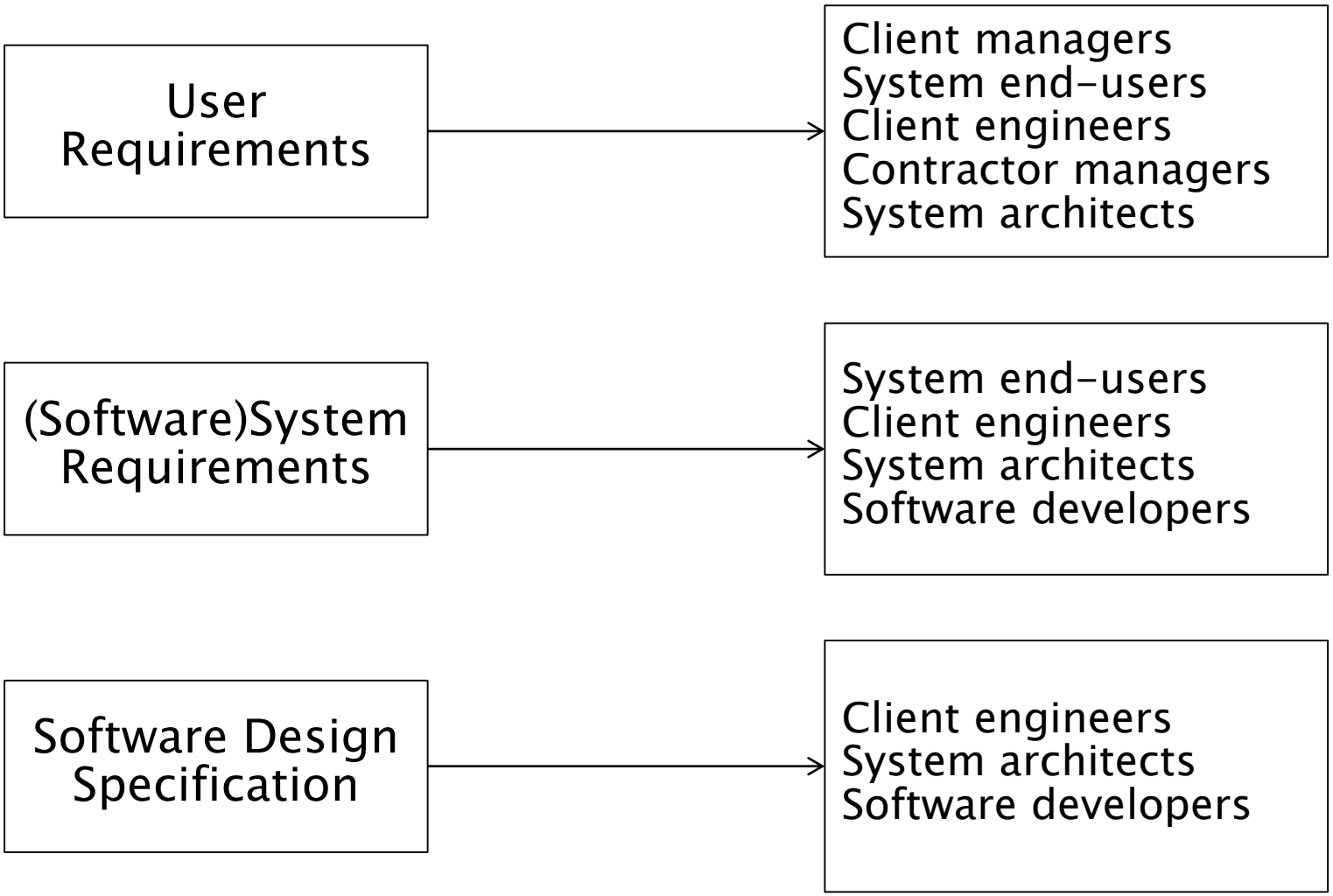
System requirements specification

1.1 The user should be provided with facilities to define the type of external files
1.2 Each external file type may have an associated tool which may be applied to the file
1.3 Each external file type may be represented as a specific icon on the user's display
1.4 Facilities should be provided for the icon representing an external file type to be defined by the user
1.5 When a user selects an icon representing an external file the effect of that selection is to apply the tool associated with the external file type to the file represented by the selected icon

Specification diagrams



Requirements readers



Functional vs. non-functional

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
- Non-functional requirements
  - Aka Quality requirements
  - constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
- Domain requirements
    - Requirements that come from the application domain of the system and that reflect characteristics of that domain.

Example: The LIBSYS system

- A library system that provides a single interface to a number of databases of articles in different libraries.
- Users can search for, download and print these articles for personal study.

Examples functional req.

FR1: The user shall be able to search either all of the initial set of databases or select a subset from it.

FR2: The system shall provide appropriate viewers for the user to read documents in the document store.

FR3: Every order shall be allocated a unique identifier (ORDER_ID) which the user shall be able to copy to the account's permanent storage area.

Requirements imprecision

- Problems arise when requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘appropriate viewers’
  - User intention – special purpose viewer for each different document type;
  - Developer interpretation – Provide a text viewer that shows the contents of the document.

Good Requirements

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

Correct

- Every requirement stated is one that the software shall meet
- Customer or users can determine if the requirement correctly reflects their actual needs
  - Traceability makes this easier

Unambiguous

- Every requirement has only one interpretation
- Each characteristic of the final product must be described using a single unique term
- Both to those who create it and to those who use it.

Complete

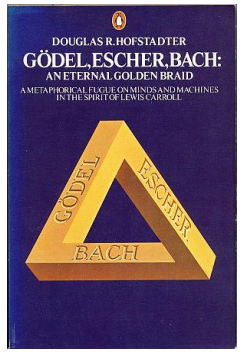
- Include all significant requirements
  - Address external requirements imposed by system specification
- Define response to all realizable inputs
  - Both correct or incorrect
- Define all terms and unit of measure

Internally Consistent

- No subset of requirements is in conflict
  - Characteristics of real-world objects (e.g. GUI
  - Logical or temporal
  - Different terms for the same object

Completeness vs. 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 document that is both complete and consistent
    - See: **Gödel's incompleteness theorems**



Ranked

- Stability in the future
- Necessity
  - Essential
  - Conditional
  - Optional

Verifiable

- There exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.
  - Ambiguous requirements are not verifiable

Modifiable

- structure and style such that any changes can be made easily, completely, and consistently while retaining the structure and style
  - Well structured
  - Non redundant
  - Separate requirements

Traceable

- Backward
  - explicitly referencing source in earlier documents
- Forward
  - unique name or reference number

Defects in requirements

- Omission/ incompleteness
- Inconsistency/contradiction
- Ambiguity
- Incorrect Fact
- Extraneous Information
  - Over-specification (design)
- Unstructured
- Redundancy

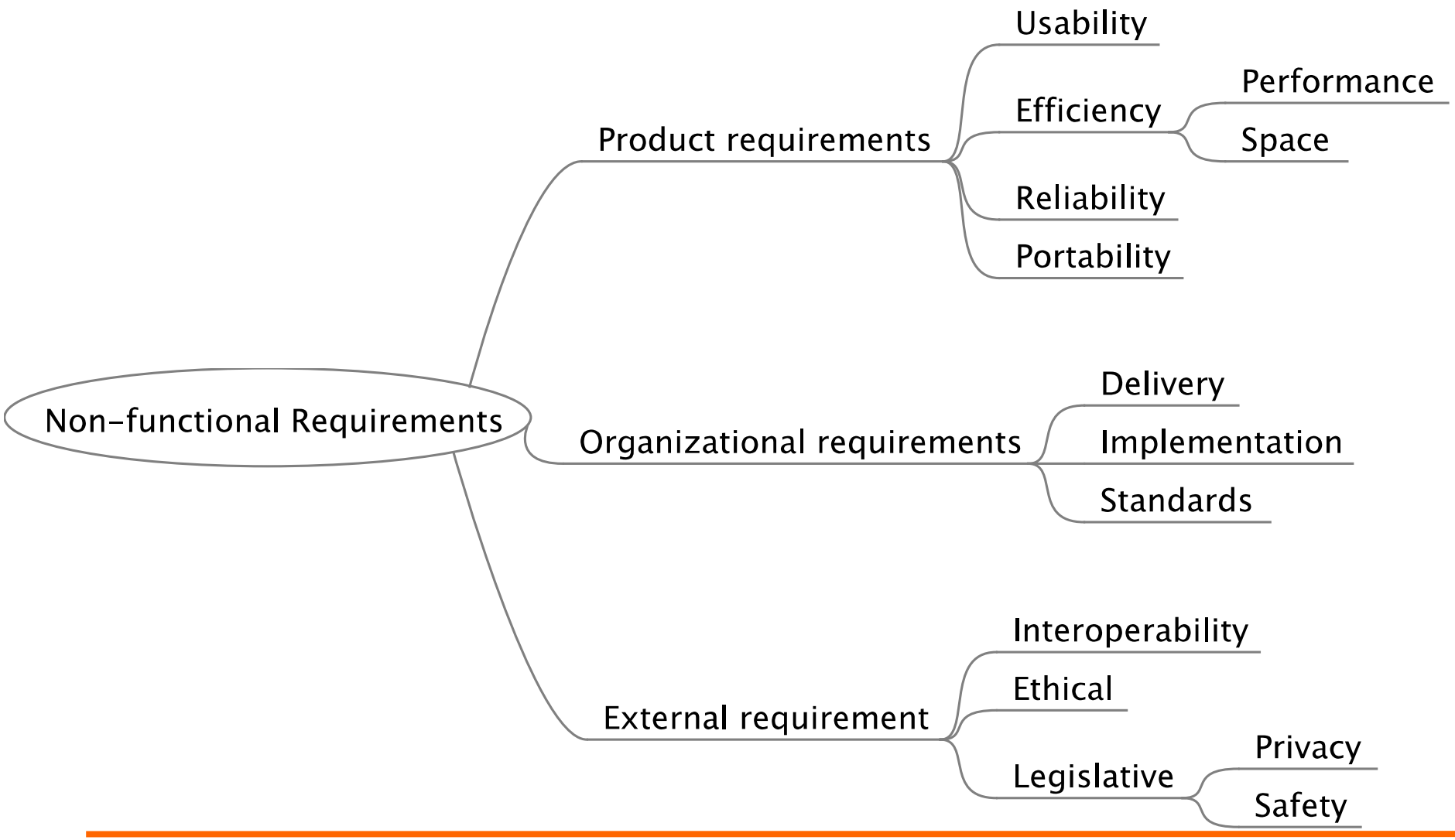
Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular CASE system, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system is useless.

Non-functional classifications

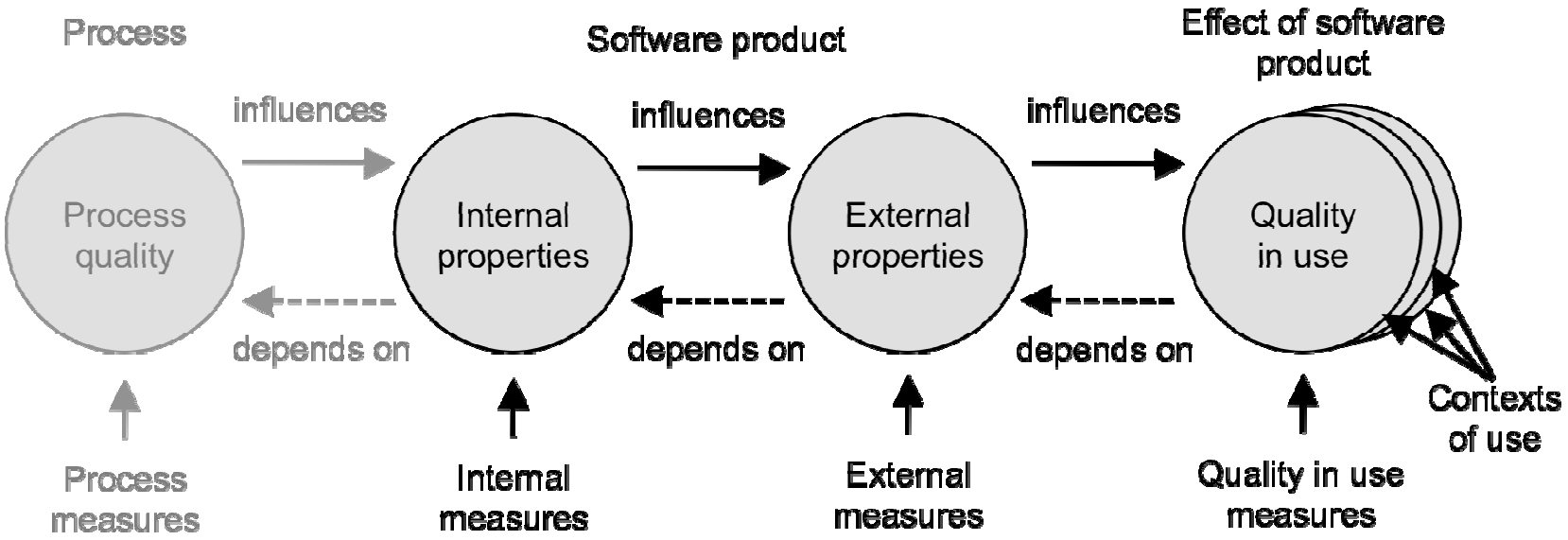
- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation 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 requirements

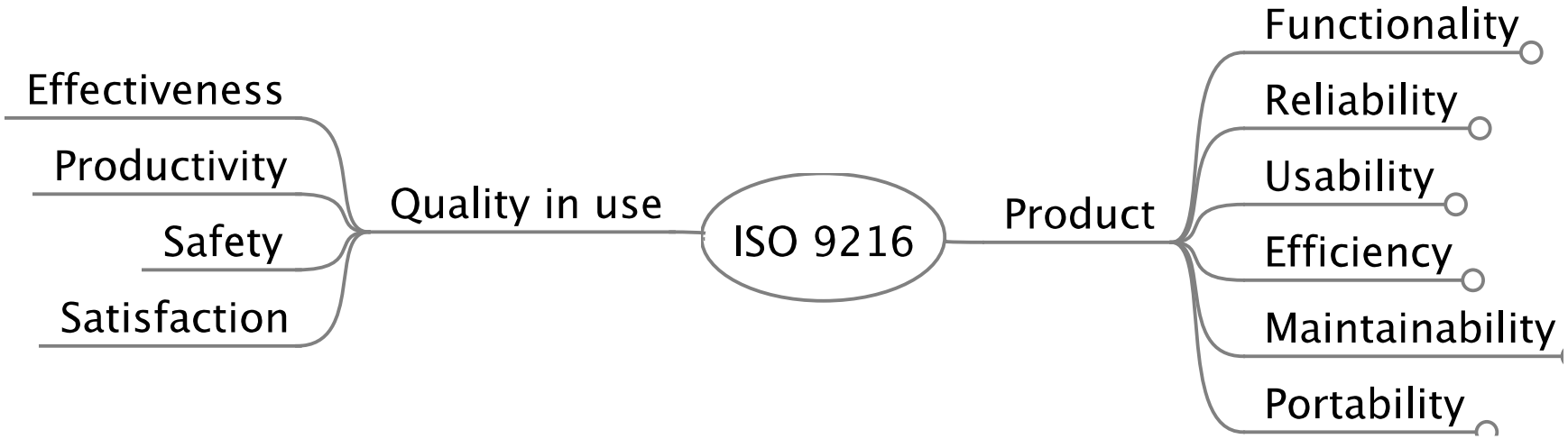


ISO 9126

- Software product quality
  - Issued 1991, revised 2001
  - Being replace by ISO/IEC 250xx
    - **SQuaRE** (Software product Quality Requirements and Evaluation)



ISO 9126 – Characteristics



Internal vs. External

- Internal features concern the static attributes of a software product
  - verified by review, inspection, simulation and/or automated tools
- External features concern the behavior of a system
  - verified and/or validated by executing the software product during testing and operation

ISO 9126 – External measure

- Breakdown avoidance

Purpose	How often can user avoid breakdown of system, even if critical failures occurred?
Method of application	Count the number of breakdowns occurrence with respect to number of failures. If it is under operation, analyze log of user operation history.
Definition	Breakdown avoidance ratio X= 1– (A / B) A= Number of breakdowns B= Number of failures NOTE: 1.The breakdown means executing of any user task is suspended until system is restarted up, or its control is lost until system is enforced to be shut down. 2. When none or a few failures observed, time between breakdown may be more suitable.
Interpretation	0<= X <= 1 The closer to 1.0 is the better.

ISO 9126 – Internal measure

- Test coverage

Purpose	How much of the required test cases are covered by the test plan?
Method of application	Count the number of test cases planned and compare it to the number of test cases required to obtain adequate test coverage.
Definition	X=A/B A=Number of test cases designed in test plan and confirmed in review B= Number of test cases required
Interpretation	0<= X Where X is the greater the better adequacy

ISO 9126

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

Non-functional req.: examples

- Product requirement
  - 8.1 The user interface for LIBSYS shall be implemented as simple HTML without frames or Java applets.
- Organisational requirement
  - 9.3.2 The system development process and deliverable documents shall conform to the process and deliverables defined in XYZCo-SP-STAN-95.
- External requirement
  - 7.6.5 The system shall not disclose any personal information about customers apart from their name and reference number to the operators of the system.

Goals and requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Examples

- **A system goal**
  - The system should be easy to use by experienced controllers and should be organised in such a way that user errors are minimised.
- **A verifiable non-functional requirement**
  - 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.

Requirements measures

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?

Development requirements

- Who are the project participants?
- What values will be reflected in the project (simple, soon, fast, or flexible)?
- What feedback or project visibility do the users and sponsors wish?
- What can we buy, what must we build, what is our competition to this system?
- What other process requirements are there (testing, installation, etc.)?
- What dependencies does the project operate under?

Domain requirements

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

LIBSYS domain requirements

- There shall be a standard user interface to all databases which shall be based on the Z39.50 standard.
- Because of copyright restrictions, some documents must be deleted immediately on arrival. Depending on the user's requirements, these documents will either be printed locally on the system server for manually forwarding to the user or routed to a network printer.

Train protection system

- The deceleration of the train shall be computed as:
  - $D_{train} = D_{control} + D_{gradient}$

where

$D_{gradient}$ is $9.81 ms^2$ * compensated gradient/alpha

and where the values of $9.81 ms^2$ /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.

User requirements

- Should describe functional and non-functional requirements in such a way that they are understandable by system users who don't have detailed technical knowledge.
- User requirements are defined using natural language, tables and diagrams as these can be understood by all users.

Problems, natural language

- Lack of clarity
  - Precision is difficult without making the document difficult to read.
- Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up.
- Requirements amalgamation
  - Several different requirements may be expressed together.

Problems, NL

- Ambiguity
  - The readers and writers of the requirement must interpret the same words in the same way. NL is naturally ambiguous so this is very difficult.
- Over-flexibility
  - The same thing may be said in a number of different ways in the specification.
- Lack of modularisation
  - NL structures are inadequate to structure system requirements.

LIBSYS requirement

4..5 LIBSYS shall provide a financial accounting system that maintains records of all payments made by users of the system. System managers may configure this system so that regular users may receive discounted rates.

Editor grid requirement

2.6 Grid facilities To assist in the positioning of entities on a diagram, the user may turn on a grid in either centimetres or inches, via an option on the control panel. Initially, the grid is off. The grid may be turned on and off at any time during an editing session and can be toggled between inches and centimetres at any time. A grid option will be provided on the reduce-to-fit view but the number of grid lines shown will be reduced to avoid filling the smaller diagram with grid lines.

Requirement problems

- Database requirements includes both conceptual and detailed information
  - Describes the concept of a financial accounting system that is to be included in LIBSYS;
  - However, it also includes the detail that managers can configure this system – this is unnecessary at this level.
- Grid requirement mixes three different kinds of requirement
  - Conceptual functional requirement (the need for a grid);
  - Non-functional requirement (grid units);
  - Non-functional UI requirement (grid switching).

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.

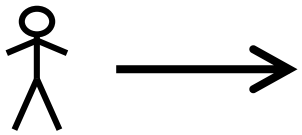
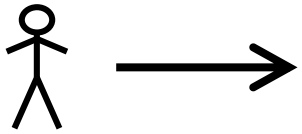
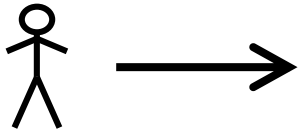
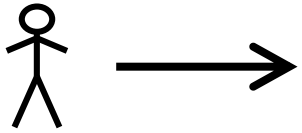
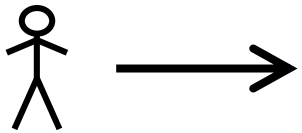
Rationale: A grid helps the user to create a tidy diagram with well-spaced entities. Although an active grid, where entities 'snap-to' grid lines can be useful, the positioning is imprecise. The user is the best person to decide where entities should be positioned.

Specification: ECLIPSE/WS/Tools/DE/FS Section 5.6
Source: Ray Wilson, Glasgow Office

The requirements document

- The requirements document is the official statement of what is required of the system developers.
- Should include both a definition of user requirements and a specification of the system/developer requirements.
- It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it

Users of requirements

System customers		Specify the requirements and read them to check that they meet their needs. They specify changes to the requirements
Managers		Use the requirements document to plan a bid for the system and to plan the system development process
System engineers		Use the requirements to understand what system is to be developed
System test engineers		Use the requirements to develop validation tests for the system
System maintenance engineers		Use the requirements to help understand the system and the relationship between its parts

IEEE requirements standard

- IEEE Std 830:1998
  - Superseded by **ISO/IEC/IEEE 29148:2011**
- Defines a generic structure for a requirements document that must be instantiated for each specific system.
  - Introduction.
  - Overall description.
  - Specific requirements.
  - Appendixes.
  - Index.

Req. document structure

- Preface
- Introduction
- Glossary
- User requirements definition
- System architecture
- System requirements specification
- System models
- System evolution
- Appendices
- Index

Organizing requirements

- Mode
- User class
- Object
- Feature
- Stimulus
- Functional hierarchy

Requirements Document simplified

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used
5. Other various requirements
6. Human backup, legal, political, organizational issues

Requirements Document

1. Purpose and scope

- What is the overall scope and goal?
- Stakeholders (who cares?)
- What is in scope, what is out of scope

organizational issues

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used
5. Other various requirements
6. Human backup, legal, political, organizational issues

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases

- The primary actors and their general goals
- The business use cases (operations concepts)
- The system use cases

Requirements Document

1. Purpose and scope
2. The terms used / Glossary
3. The use cases
4. The technology to be used

- What technology requirements are there for this system?
- What systems will this system interface with, with what requirements?

R

- Development process
- Business rules
- Performance
- Operations, security, documentation
- Use and usability
- Maintenance and portability
- Unresolved or deferred

5. Other various requirements
6. Human backup, legal, political, organizational issues

Requirements Document

- What is the human backup to system operation?
- What legal, what political requirements are there?
- What are the human consequences of completing this system?
- What are the training requirements?
- What assumptions, dependencies are there on the human environment?

6. Human backup, legal, political, organizational issues

Guidelines for requirements

- Define a standard format and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.

System/developer requirements

- More detailed specifications of system functions, services and constraints than user requirements.
- They are intended to be a basis for designing the system.
- They may be incorporated into the system contract.
- System requirements may be defined or illustrated using system models (UML)

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific design may be a domain requirement.

Alternatives to NL specification

Notation	Description
Structured natural language	This approach depends on defining standard forms or templates to express the requirements specification.
Design description languages	This approach uses a language like a programming language but with more abstract features to specify the requirements by defining an operational model of the system. This approach is not now widely used although it can be useful for interface specifications.
Graphical notations	A graphical language, supplemented by text annotations is used to define the functional requirements for the system. An early example of such a graphical language was SADT. Now, use-case descriptions and sequence diagrams are commonly used .
Mathematical specifications	These are notations based on mathematical concepts such as finite-state machines or sets. These unambiguous specifications reduce the arguments between customer and contractor about system functionality. However, most customers don't understand formal specifications and are reluctant to accept it as a system contract.

Structured language

- The freedom of the requirements writer is limited by a predefined template for requirements.
- All requirements are written in a standard way.
- The terminology used in the description may be limited.
- The advantage is that the most of the expressiveness of natural language is maintained but a degree of uniformity is imposed on the specification.

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 (r2), the previous two readings (r0 and r1)

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

Outputs CompDose – 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 r0 is replaced by r1 then r1 is replaced by r2

Side-effects None

Tabular specification

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

Tabular specification

Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2-r1)<(r1-r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ((r2-r1) ≥ (r1-r0))	CompDose = round ((r2-r1)/4) If rounded result = 0 then CompDose = MinimumDose

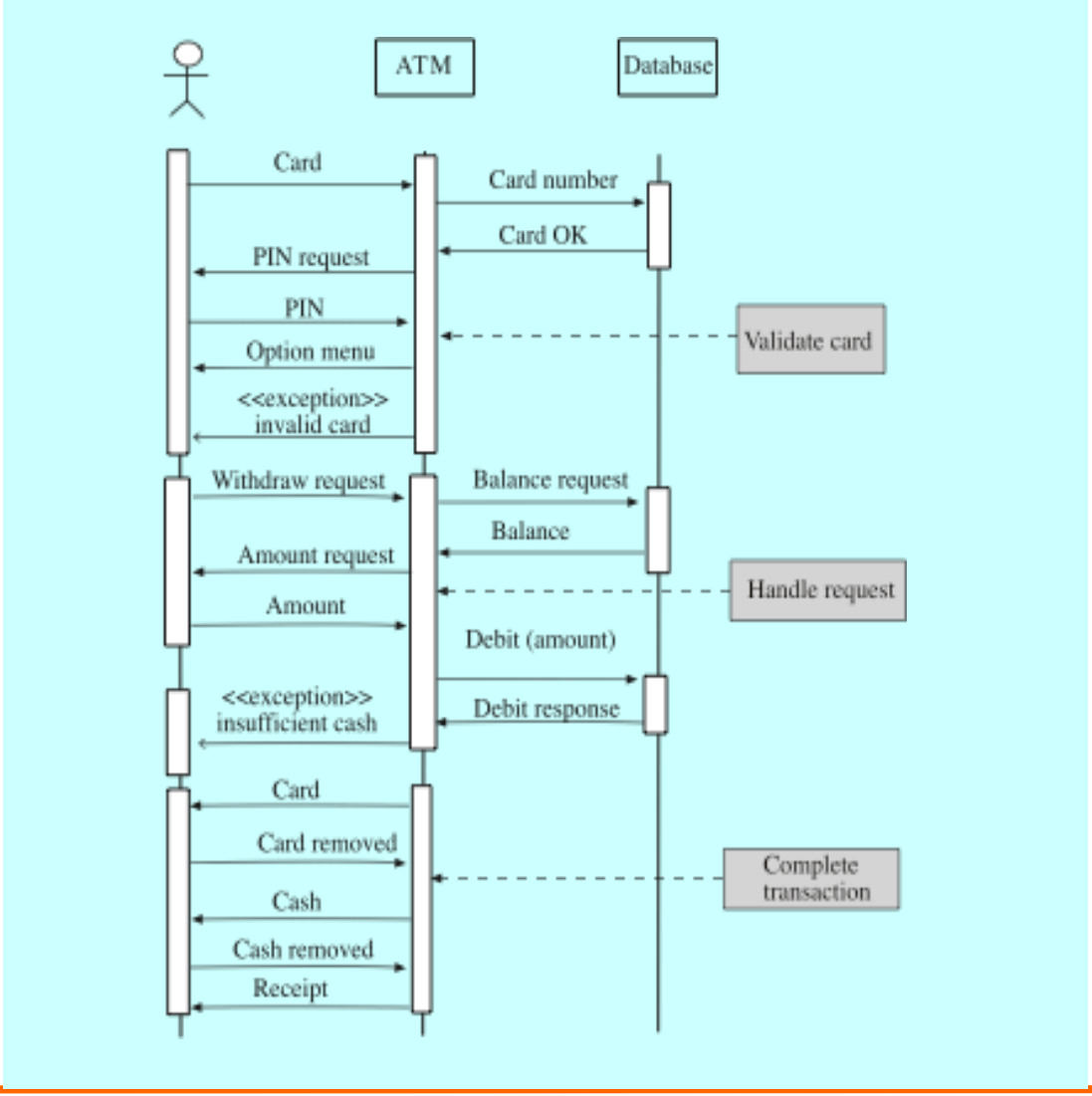
Graphical models

- Graphical models are most useful when you need to show how state changes or where you need to describe a sequence of actions.
- See UML, sequence diagrams.

Sequence diagrams

- These show the sequence of events that take place during some user interaction with a system.
- You read them from top to bottom to see the order of the actions that take place.
- Cash withdrawal from an ATM
  - Validate card;
  - Handle request;
  - Complete transaction.

Example seq of ATM withdrawal



Interface specification

- Most systems must operate with other systems and the operating interfaces must be specified as part of the requirements.
- Three types of interface may have to be defined
  - Procedural interfaces;
  - Data structures that are exchanged;
  - Data representations.
- 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
```

User interfaces

- Mostly non-functional
  - Elements of functional reqs (input)
- Prototypes are essential
  - Provide clear information to developers
  - Get feedback from users
  - Get commitment

V&V of requirements

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

Tools

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

References

- IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998, Revision of IEEE Std 830-1993)
- ISO/IEC 250xx:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE)

Key points

- Requirements set out what the system should do and define constraints on its operation and implementation.
- Functional requirements set out services the system should provide.
- Non-functional requirements constrain the system being developed or the development process.
- User requirements are high-level statements of what the system should do. User requirements should be written using natural language, tables and diagrams.

Key points

- System requirements are intended to communicate the functions that the system should provide.
- A software requirements document is an agreed statement of the system requirements.
- The IEEE standard is a useful starting point for defining more detailed specific requirements standards.