III. Software Requirements

- General concepts and definition of Software Engineering

- Qualities of interest developing software

- Software Lyfe Cycle

- Software Development Processes

  - Waterfall

  - Evolutionary

  - Component Based

  - Iterative

Objectives

- To introduce the concepts of **user** and **system** requirements

- To describe **functional** and **non-functional** requirements

- To explain how software requirements may be organised in a requirements document

- Today we will not specify **how to discover or analyse requirements**

Most of the following material has been provided by Ian Sommerville©

- Functional and non-functional requirements

- User requirements

- System requirements

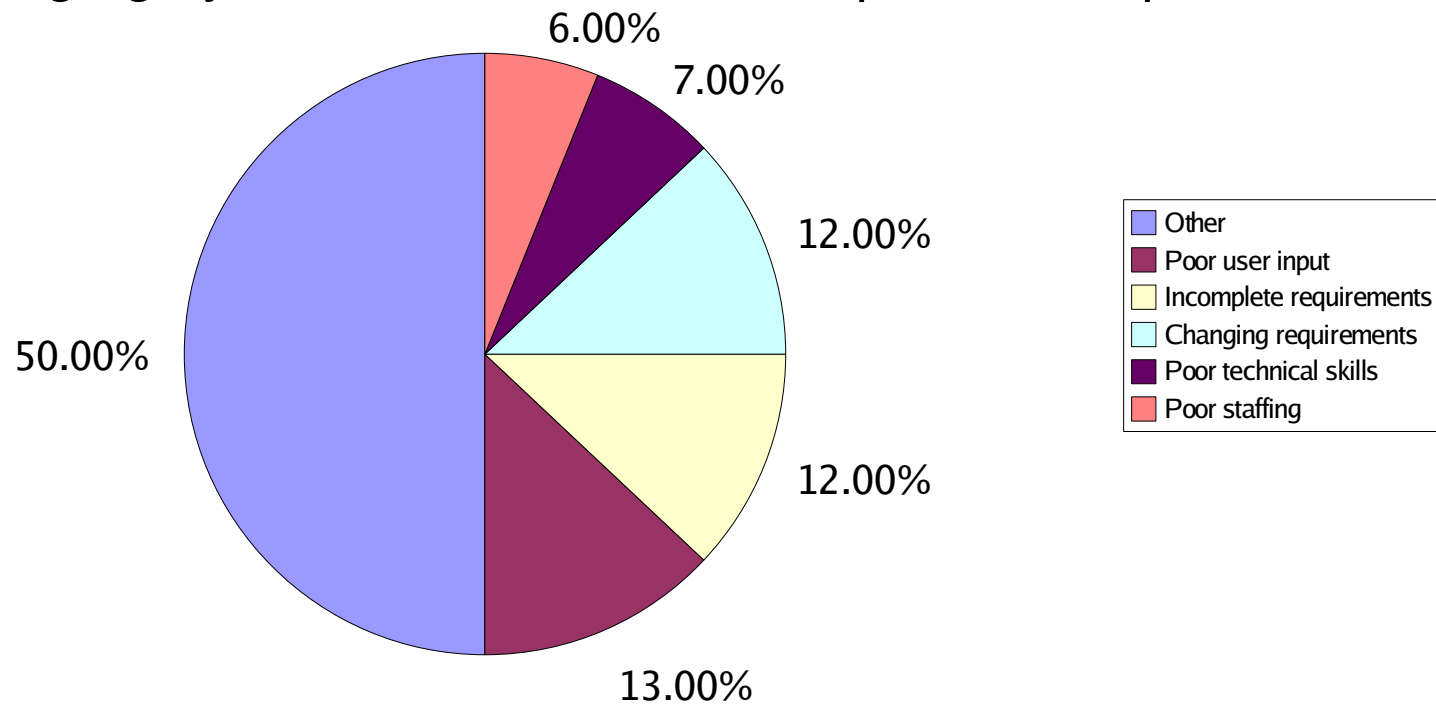- Interface specification

- The software requirements document

- 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.

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.

Relevance of Requirements

- Requirements are probably the activity that deserve the greatest care

- **Problems inserted in the system during the requirements phase are the most expensive to remove**

- Studies revealed that around **37%** of the problems, developing challenging systems, are related to requirements phases

Pie chart values: 6.00%, 7.00%, 12.00%, 12.00%, 13.00%, 50.00%

Legend:
- Other
- Poor user input
- Incomplete requirements
- Changing requirements
- Poor technical skills
- Poor staffing

"If a company wishes to let a contract for a large software development project, **it must define its needs in a sufficiently abstract way that a solution is not pre-defined**. The requirements must be written so that **several contractors can bid for the contract**, offering, perhaps, different ways of meeting the client organisation's needs. Once a contract has been awarded, **the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do.** Both of these documents may be called the *requirements document* for the system."

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 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.

Definition and Specifications

▼ User requirement definition

- The system must provide a mean of representing and accessing external files created by other tools

▼ System requirement specification

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

User requirements

- **Client Managers**
- **System End-users**
- **Client Engineers**
- **Contractor Managers**
- **System Architects**

Systems requirements

- **System End-users**
- **Client Engineers**
- **System Architects**
- **Software Developers**

Software Design Specification

- **Client Engineers (perhaps)**
- **System Architects**
- **Software Developers**

Functional and Non-functional Requirements

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

- 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.

Functional Requirements

- Describe **functionality** or system services.

- Depend on the **type of software**, **expected users** and the type of system where the software is used.

- Functional **user requirements** may be **high-level statements** of what the system should do but **functional system requirements should describe the system services in detail.**

- 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.

Ingegneria del Software I – A.A. 2006/2007

Andrea Polini

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

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

- 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.

- Writing requirements you do **not** have **to prove your language skills**. You should not engage finding synonyms

UNICAM
Università di Camerino

Requirements 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.

- 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.

Ingegneria del Software I – A.A. 2006/2007

Andrea Polini

Non-Functional Classifications

- **Product requirements**

  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
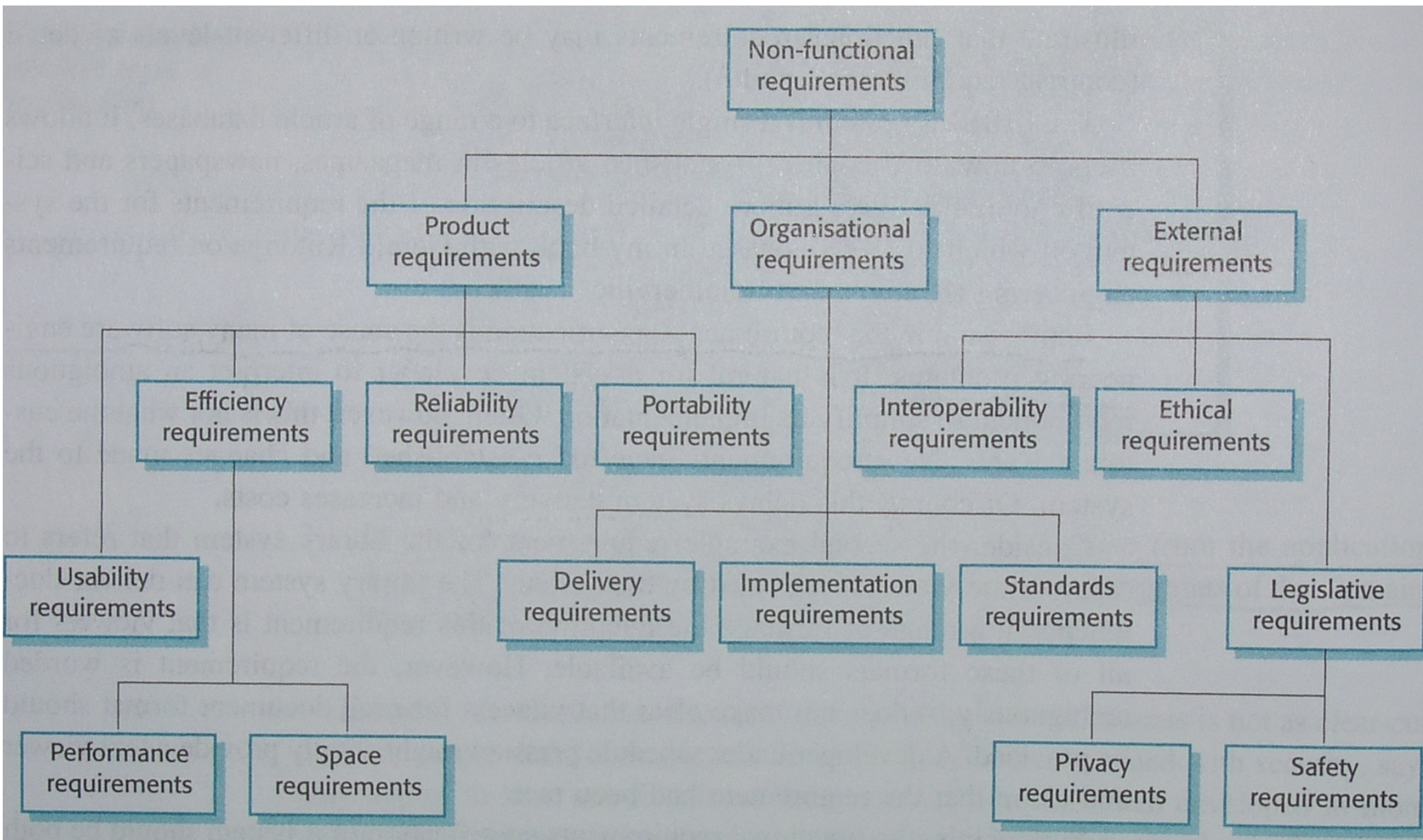
- **Organisational requirements**

  - 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.

Ingegneria del Software I – A.A. 2006/2007
Andrea Polini

Non-functional Requirements 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.

- **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.

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.

Ingegneria del Software I – A.A. 2006/2007

Andrea Polini

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

- **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?

- 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.

- 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.

- The deceleration of the train shall be computed as:

  - $D_{train} = D_{control} + D_{gradient}$

- 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 Requirements 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.

- 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.**

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.

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.

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.

- 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).

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

- **Invent 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**.

- **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** discussed in Chapter 8.

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.**

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.

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 Specifications

- 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.

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

- Used to supplement natural language.

- Particularly **useful when you have to define a number of possible alternative** courses of action

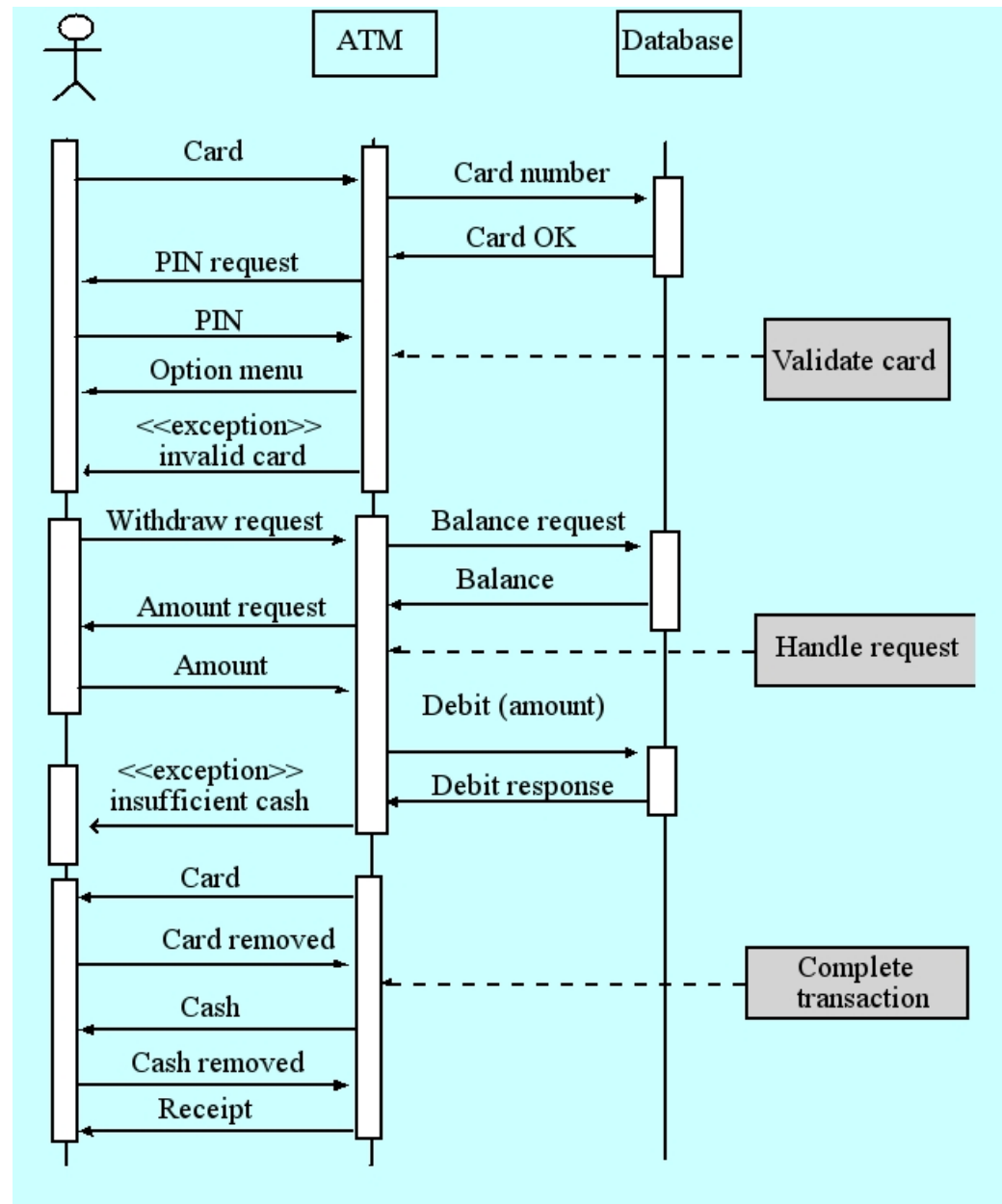| Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2-r1)<(r1-r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing. ((r2-r1) ≥ (r1-r0))	CompDose = round ((r2-r1)/4)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

- Different graphical models are explained in Chapter 8.

- 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.

- 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 (API);

  - Data structures that are exchanged;

  - Data representations.

- Formal notations are an effective technique for interface specification.

```
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
```

- 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 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 a Requirements Document

- **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

Defines a generic structure for a requirements document that must be instantiated for each specific system.

- Introduction.
- General description.
- Specific requirements.
- Appendices.
- Index.

Requirements Document Structure

- Preface

- Introduction

- Glossary

- User requirements definition

- System architecture

- System requirements specification

- System models

- System evolution

- Appendices

- Index

- 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.

- 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.

- Un distributore automatico di cibi e bevande permette la selezione e l'acquisto di cibo o bevande dietro il pagamento della rispondente somma di denaro. Per poter acquistare un prodotto il cliente dovrà depositare la somma necessaria e selezionare il prodotto prescelto. La conferma dell'aquisto avverrà pigiando il corrispondente pulsante oppure, per i clienti muniti di chiavetta, tramite l'inserimento di un codice personale. Se non sono stati inseriti soldi la pressione del pulsante permetterà di visualizzare il costo di un prodotto.

- **Come giudicate questo requisito? Vi sembra abbastanza chiaro?**

- Chiaramente il requisito è ambiguo e presenta molte omissioni:

  - Cosa accade se il cliente seleziona un prodotto più costoso rispetto alla somma inserita?

  - Selezione del prodotto ed inserimento dei soldi devono rispettare un ordine? Se seleziono un prodotto per visualizzare il prezzo dunque inserisco dei soldi e pigio il pulsante corrispondente di conferma il prodotto sarà erogato oppure no?

  - La selezione del prodotto e la conferma dovrà avvenire tramite lo stesso pulsante?

  - Cosa accade se inserisco soldi mentre una chiave è inserita?

  - Cosa accade se il codice personale è errato?

  - Come posso depositare soldi aumentando il credito su di una chiavetta?

  - E' possibile avere più prodotti alla volta?

  - **IL RESTO!!!**

Function: Acquisto di un prodotto in contanti

Description: processo che permette ad un cliente di acquistare un bene dietro pagamento del corrispettivo in denaro

Inputs: Monete, Selezione del prodotto, Conferma

Output: Prodotto, resto

Azioni: Il sistema riceve dal cliente un certo ammontare di denaro attraverso la fessura per il contante. Il sistema evidenzia i pulsanti di conferma per tutti i prodotti che hanno un prezzo inferiore alla somma inserita. Il sistema riceve la selezione di un prodotto tra quelli illuminati e attiva il lampeggio del relativo pulsante. Il sistema riceve la conferma dell'acquisto tramite la pressione del pulsante lameggiante e corrispondentemente fornisce il prodotto ed il resto

Pre-condizioni: il sistema è disponibile a ricevere un ordine, non vi sono altre transazioni in corso

Post-condizioni: La somma restituita è pari alla differenza tra quanto versato ed il prezzo del prodotto acquistato, il prodotto fornito corrisponde a quello selezionato dal cliente, la macchina è pronta per una nuova transazione

- Può verificarsi al più un errore ogni 1000 transazioni

- il sistema non deve mai fornire cibo e bevande se la somma corrispondente non è stata già versata

- Il sistema può essere indisponibile al più per un'ora la settimana

- Se il sistema va in errore deve essere possibile riportarlo in uno stato corretto in non più di 2 minuti

- Il tempo per illuminare un pulsante deve essere inferiore a 0.5 sec.

- Dopo aver concluso una transazione il sistema deve rendersi disponibile entro 3 sec.

- **Scrivete un documento dei requisiti la cui possibile implementazione corrisponda alla macchinetta con lo sportellino del piano terra !!**

Ingegneria del Software I – A.A. 2006/2007

Andrea Polini