

CSE344

Software Engineering

(SWE)

Week – 3

Requirements & Req. Engineering

Requirement (İster) ...

- *...is a way of expressing a client's need* (a service or a constraint on the system) ranging at one end from
 - *a high-level abstract statement* at the other end to
 - *a detailed mathematical functional specification.*
- *...may serve a dual function*
 - *Basis for a bid (öneri) for a contract \Leftrightarrow open to interpretation*
 - *Basis for the contract itself \Leftrightarrow defined in detail;*

Requirements abstraction (Davis)

“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.”

A Classification of req.s by their reader

- **User requirements** (written *for customers*)
 - *Statements in natural language plus diagrams of the services of the system and its operational constraints.*
- **System requirements**
 - a *structured* document
 - detailed descriptions of the system's functions, and operational constraints (functionality of the system).
 - *may be part of a contract between client and contractor.*

Definitions and specifications

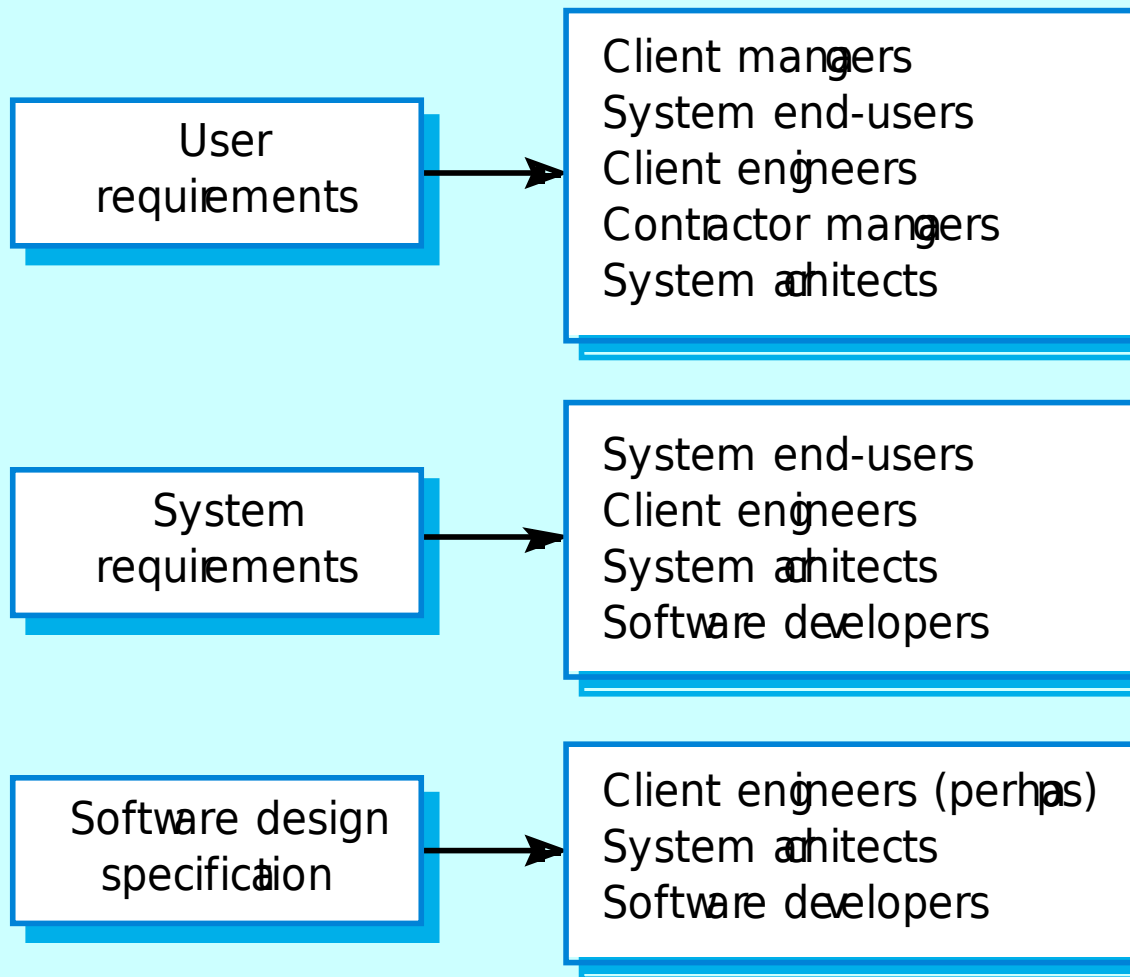
User requirement definition

1. The software must provide a means of representing and accessing external files created 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 must have an associated tool which may be applied to the file
- 1.3 Each external file type must 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 type of the external file to the file represented by the selected icon.

Requirements readers



Another classification regarding what they state...

- ***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.

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 of functional requirements

- 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.

Requirements completeness and consistency

- In principle, set of requirements should be both complete and consistent.
- **Completeness**
 - ...includes descriptions of all facilities required.
- **Consistency**
 - ... no conflicts or contradictions.
- In practice, it is *impossible* to produce a complete and consistent requirements document.

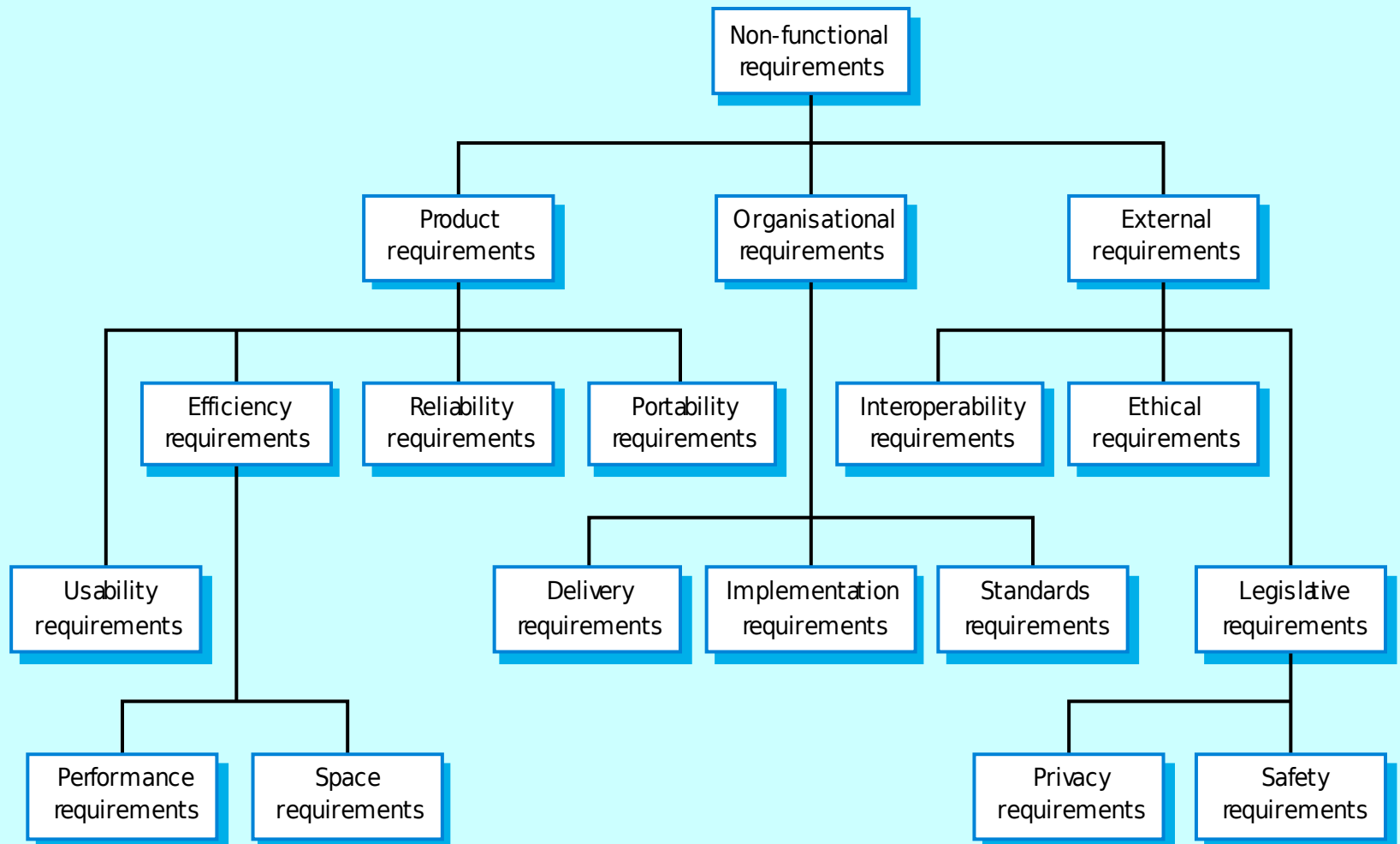
Non-functional (NF) Requirements

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

Classifications of NF Requirements

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

Types of NF Requirements



Examples of NF Requirements

- ***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 (legislative here)***

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.

Verifiability of NF Requirements

- NF requirements ...
 - ...are usually hard to state precisely and hence to verify.
 - ...originate from *user's intention (i.e., a goal)* to state a property about the system
 - ...are *system properties* related to the entirety of the system.
- A *verifiable* NF requirement ...
 - ...is a statement using some measure that can be objectively tested.

Example...

- ***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.

NF Requirements measures

Property	Measure
Speed	Processed transactions/second User/Event response time Screen refresh time
Size	Mega 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 NF requirements* are possible in complex systems.
- Example: *Spacecraft system*
 - To minimise weight: use a minimum number of separate chips in the system.
 - To minimise power consumption: use lower-power chips meaning more chips compared with when using higher-power chips.
 - **Conflict(!!!)**
 - **Question:** *Which is the most critical requirement?*

Domain requirements

- ...originate from the *application domain* and
- ...describe *system characteristics* that reflect the domain.
- ...may be
 - ...new functional requirements, or
 - ...constraints on existing requirements or
 - ...define specific computations.

Library system domain requirements (Examples 1&2)

- 1) There shall be a standard user interface to all databases which shall be based on the *Z39.50 standard*.
- 2) 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 (Example 3)

1) The deceleration of the train shall be computed as:

$$D_{\text{train}} = D_{\text{control}} + D_{\text{gradient}}$$

where D_{gradient} is $9.81\text{ms}^2 * \text{compensated gradient}/\alpha$ and where the values of $9.81\text{ms}^2 / \alpha$ 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.

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 with 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.

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

Guidelines for writing requirements

- 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.

System 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 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.

Problems with NL specification

- 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.

Alternatives to NL specification

- Structured NL
 - Form-based or tabular specifications
- Design description languages (DDLs)
 - ...use a language like a prog. lang. But with more abstract features to specify req.s by defining an operational model of the system.
 - Not very popular
- Graphical Notations
 - UML diagrams such as use-case or sequence diagrams
 - Very popular
- Mathematical Specifications
 - State machines or sets
 - Unambiguous, but formal and hard to understand by most customers

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 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.

Form-based node specification

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 ($r_2 < r_1$)	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

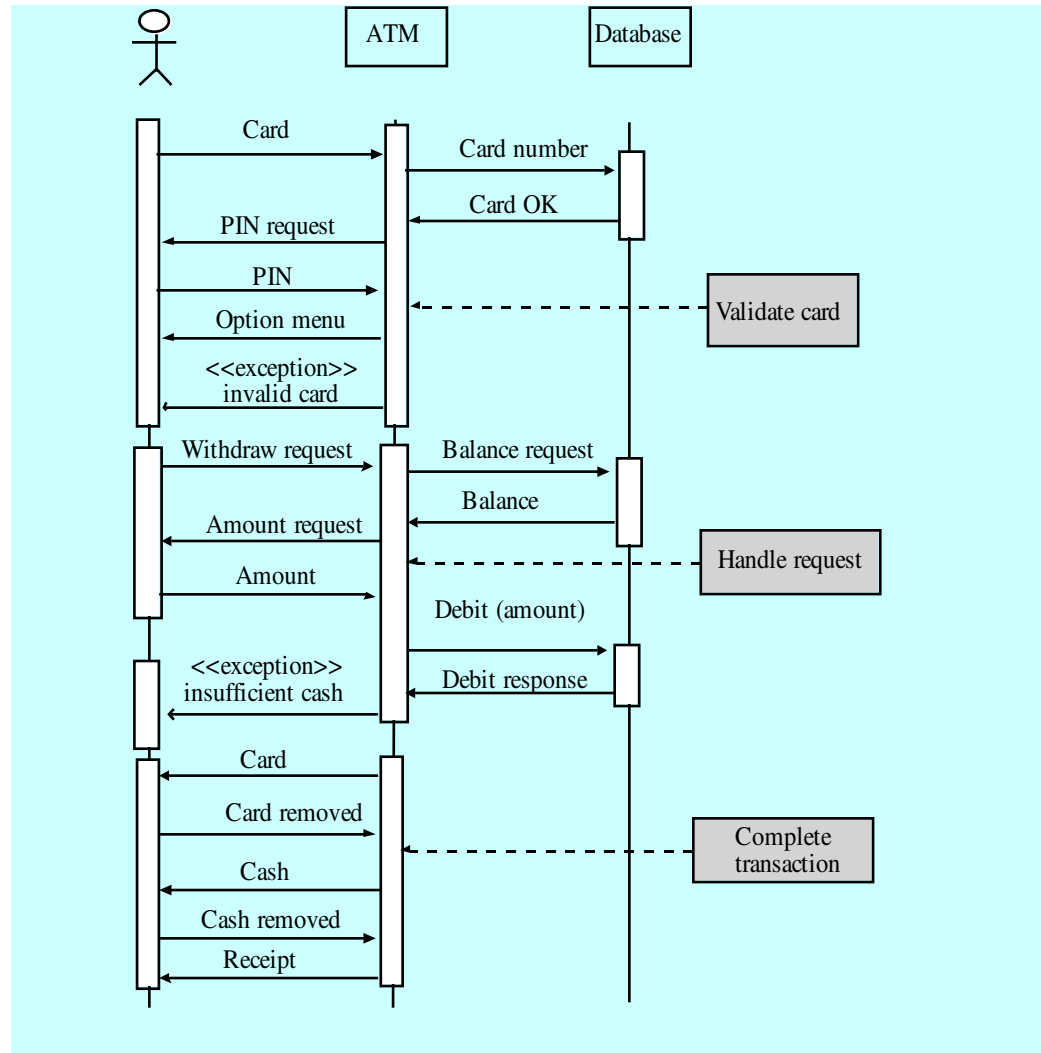
Graphical models

- 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.

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.

Sequence diagram 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.

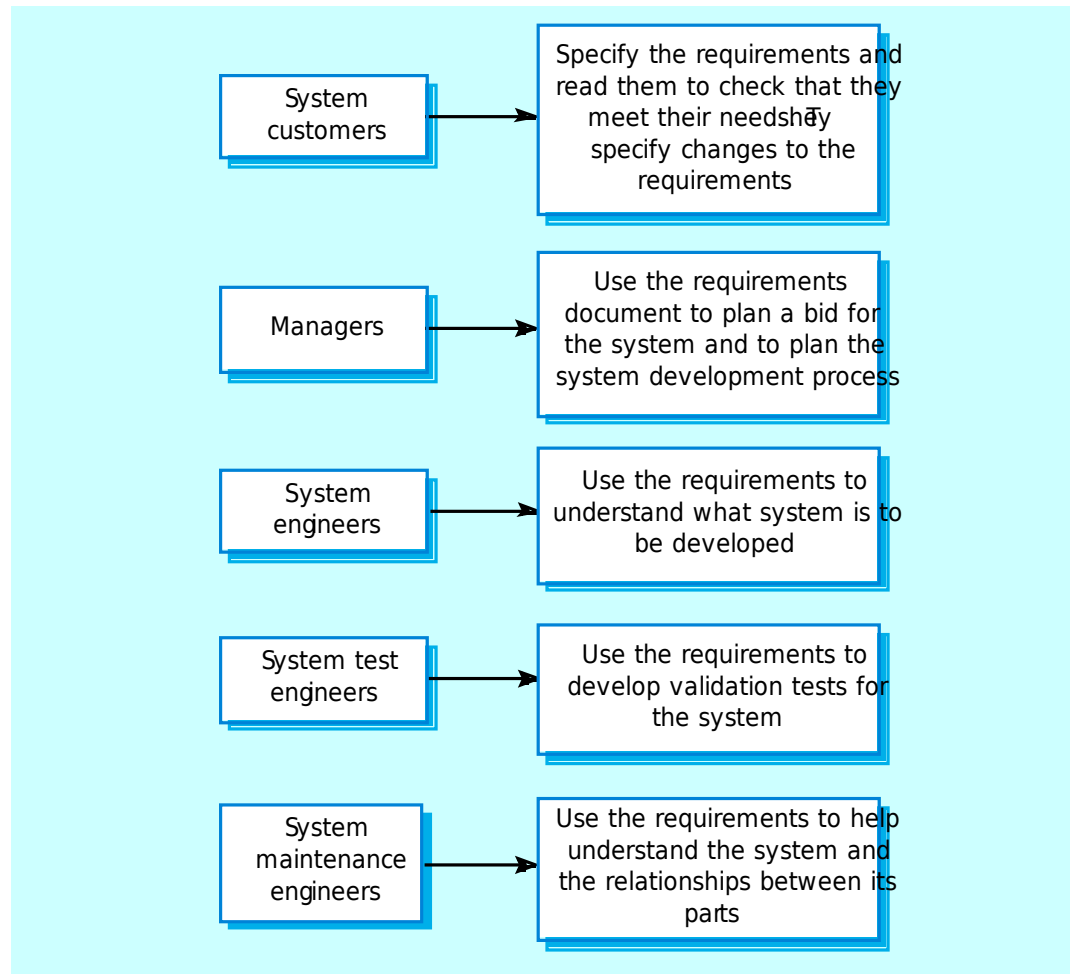
PDL interface description

```
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
```

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 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



IEEE requirements standard

- 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

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.

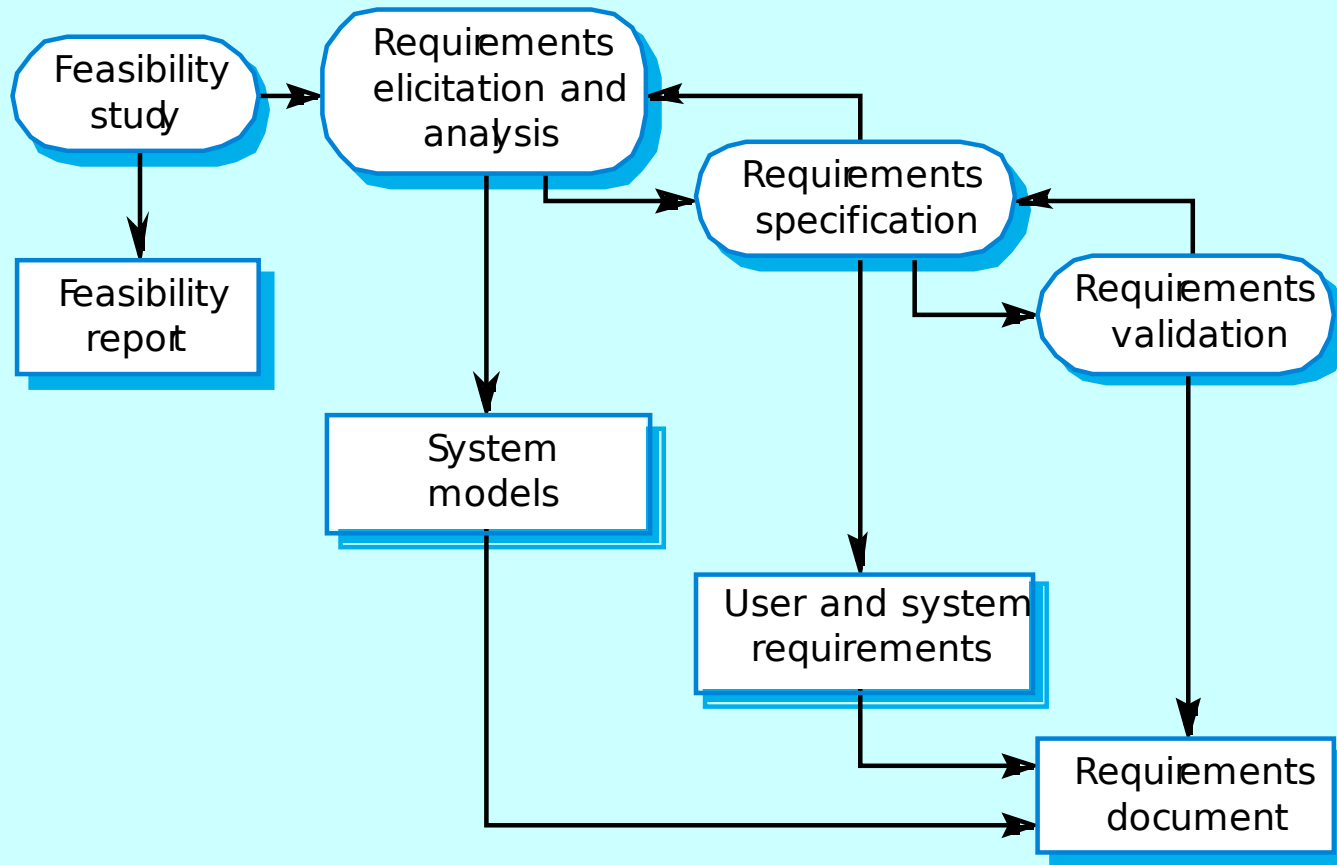
Requirements engineering

- 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.

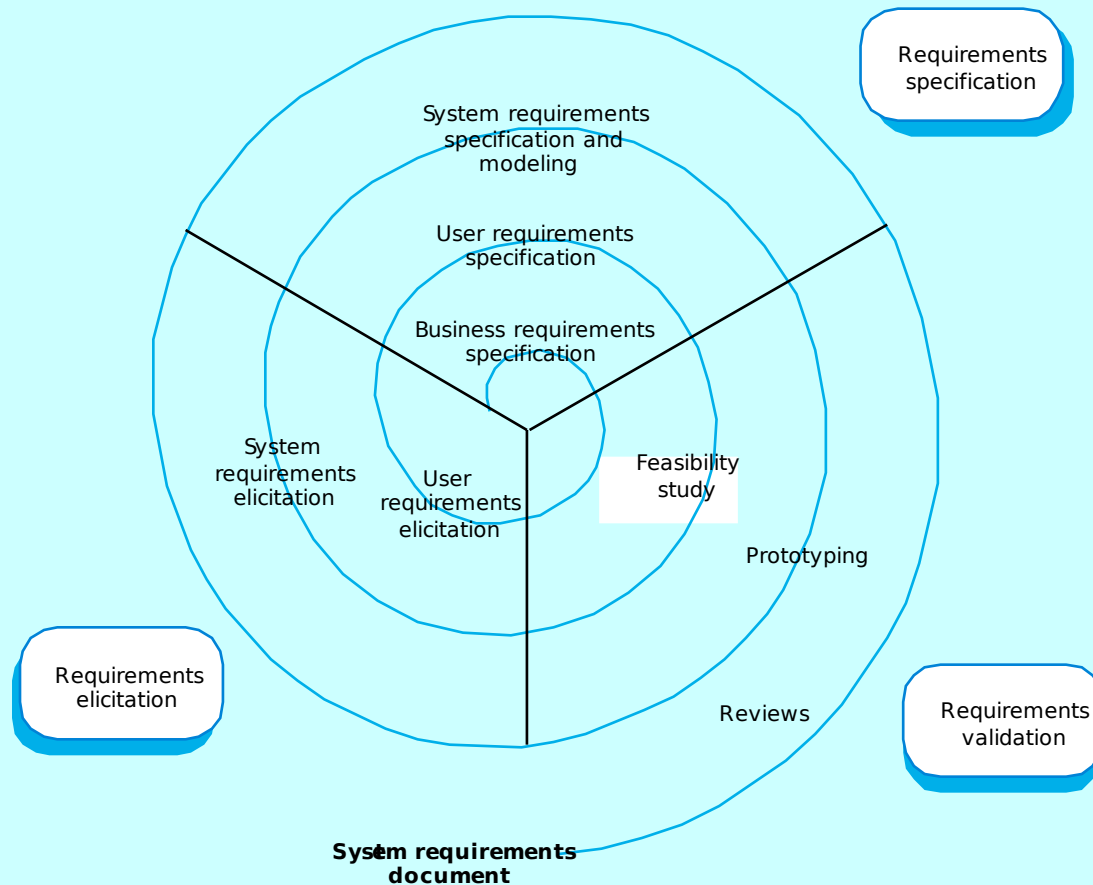
Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- However, there are a number of generic activities common to all processes
 - Requirements elicitation;
 - Requirements analysis;
 - Requirements validation;
 - Requirements management.

The requirements engineering process



Requirements engineering



Feasibility studies

- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
 - If the system contributes to organisational objectives;
 - If the system can be engineered using current technology and within budget;
 - If the system can be integrated with other systems that are used.

Feasibility study implementation

- Based on information assessment (what is required), information collection and report writing.
- Questions for people in the organisation
 - What if the system wasn't implemented?
 - What are current process problems?
 - How will the proposed system help?
 - What will be the integration problems?
 - Is new technology needed? What skills?
 - What facilities must be supported by the proposed system?

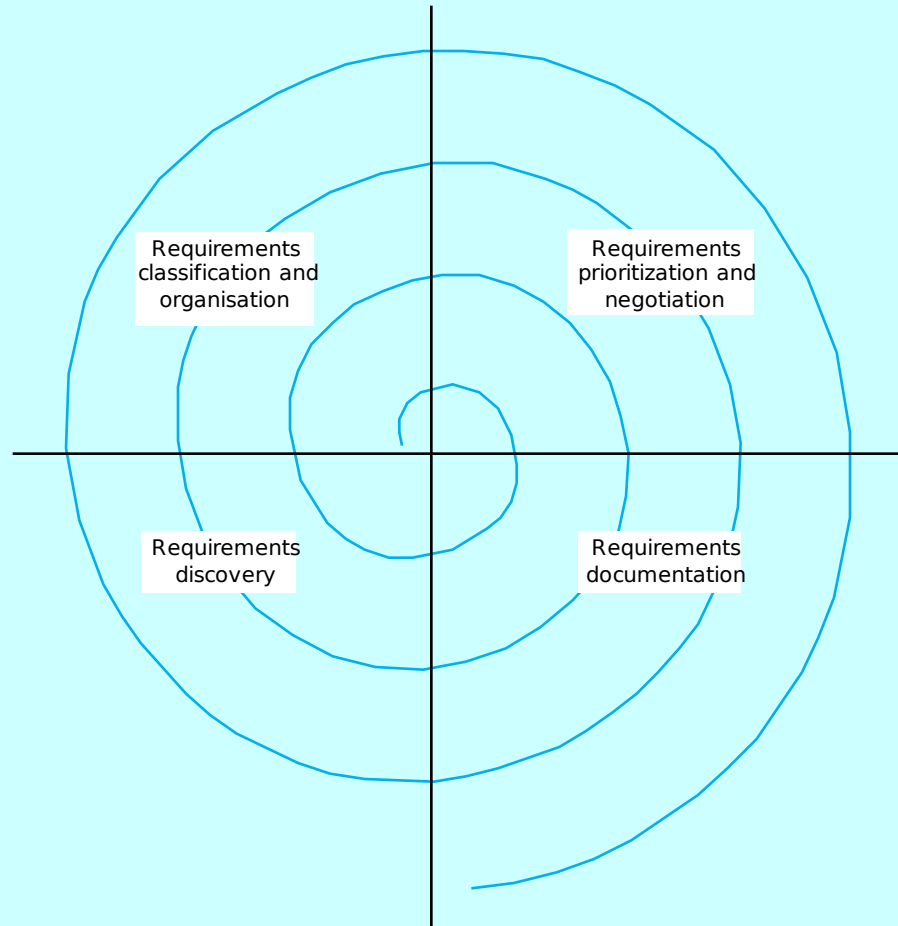
Elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*.

Problems of requirements analysis

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment change.

The requirements spiral



Process activities

- Requirements discovery
 - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
 - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
 - Prioritising requirements and resolving requirements conflicts.
- Requirements documentation
 - Requirements are documented and input into the next round of the spiral.

Requirements discovery

- The process of gathering information about the proposed and existing systems and distilling the user and system requirements from this information.
- Sources of information include documentation, system stakeholders and the specifications of similar systems.

ATM stakeholders

- Bank customers
- Representatives of other banks
- Bank managers
- Counter staff
- Database administrators
- Security managers
- Marketing department
- Hardware and software maintenance engineers
- Banking regulators

Viewpoints

- Viewpoints are a way of structuring the requirements to represent the perspectives of different stakeholders. Stakeholders may be classified under different viewpoints.
- This multi-perspective analysis is important as there is no single correct way to analyse system requirements.

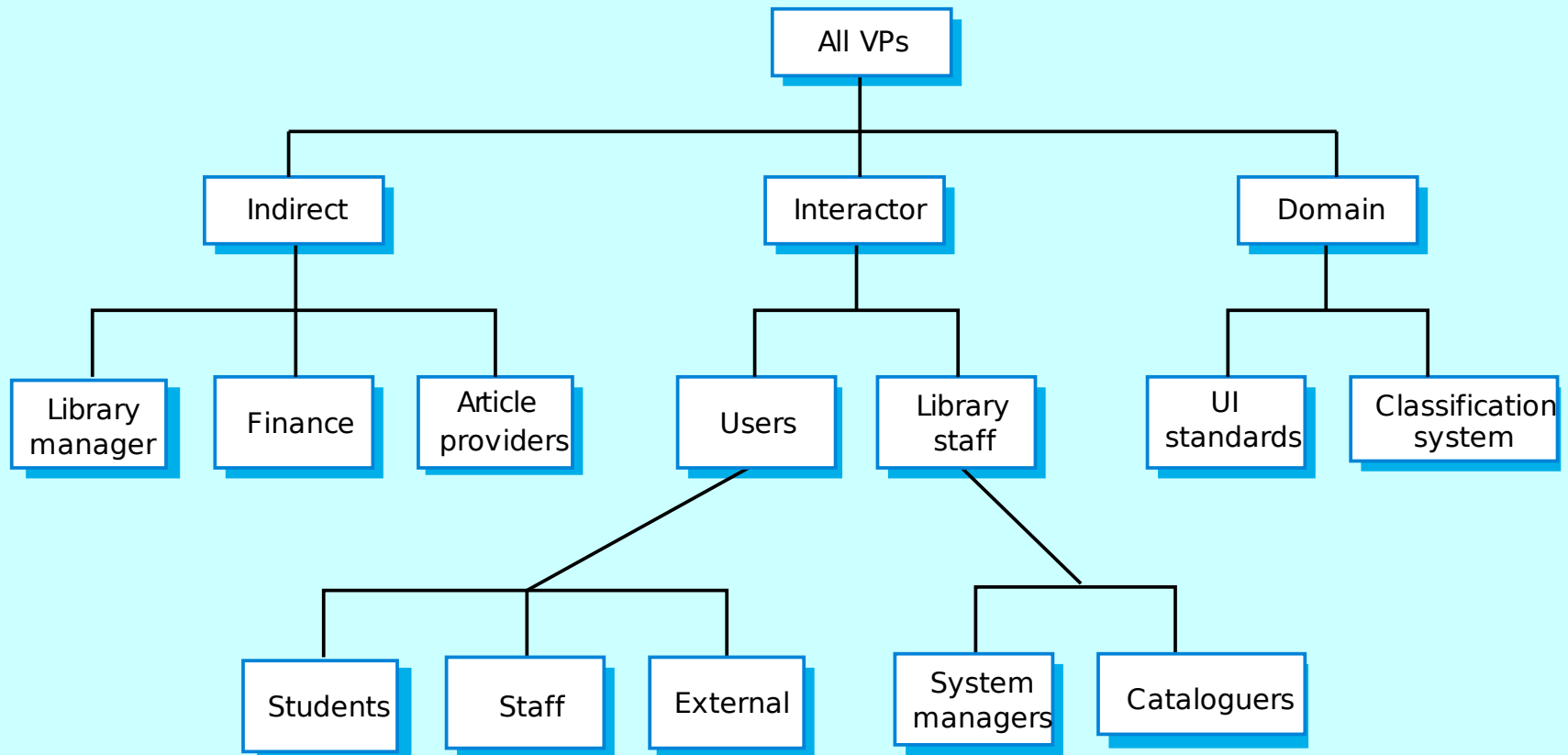
Types of viewpoint

- Interactor viewpoints
 - People or other systems that interact directly with the system. In an ATM, the customer's and the account database are interactor VPs.
- Indirect viewpoints
 - Stakeholders who do not use the system themselves but who influence the requirements. In an ATM, management and security staff are indirect viewpoints.
- Domain viewpoints
 - Domain characteristics and constraints that influence the requirements. In an ATM, an example would be standards for inter-bank communications.

Viewpoint identification

- Identify viewpoints using
 - Providers and receivers of system services;
 - Systems that interact directly with the system being specified;
 - Regulations and standards;
 - Sources of business and non-functional requirements.
 - Engineers who have to develop and maintain the system;
 - Marketing and other business viewpoints.

LIBSYS viewpoint hierarchy



Interviewing

- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
 - Closed interviews where a pre-defined set of questions are answered.
 - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

Interviews in practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviews are not good for understanding domain requirements
 - Requirements engineers cannot understand specific domain terminology;
 - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Effective interviewers

- Interviewers should be open-minded, willing to listen to stakeholders and should not have pre-conceived ideas about the requirements.
- They should prompt the interviewee with a question or a proposal and should not simply expect them to respond to a question such as ‘what do you want’.

Scenarios

- Scenarios are real-life examples of how a system can be used.
- They should include
 - A description of the starting situation;
 - A description of the normal flow of events;
 - A description of what can go wrong;
 - Information about other concurrent activities;
 - A description of the state when the scenario finishes.

LIBSYS scenario (1)

Initial assumption: The user has logged on to the LIBSYS system and has located the journal containing the copy of the article.

Normal: The user selects the article to be copied. He or she is then prompted by the system to either provide subscriber information for the journal or to indicate how they will pay for the article. Alternative payment methods are by credit card or by quoting an organisational account number.

The user is then asked to fill in a copyright form that maintains details of the transaction and they then submit this to the LIBSYS system.

The copyright form is checked and, if OK, the PDF version of the article is downloaded to the LIBSYS working area on the user's computer and the user is informed that it is available. The user is asked to select a printer and a copy of the article is printed. If the article has been flagged as 'print-only' it is deleted from the user's system once the user has confirmed that printing is complete.

LIBSYS scenario (2)

What can go wrong: The user may fail to fill in the copyright form correctly. In this case, the form should be re-presented to the user for correction. If the resubmitted form is still incorrect then the user's request for the article is rejected.

The payment may be rejected by the system. The user's request for the article is rejected.

The article download may fail. Retry until successful or the user terminates the session.

It may not be possible to print the article. If the article is not flagged as 'print-only' then it is held in the LIBSYS workspace. Otherwise, the article is deleted and the user's account credited with the cost of the article.

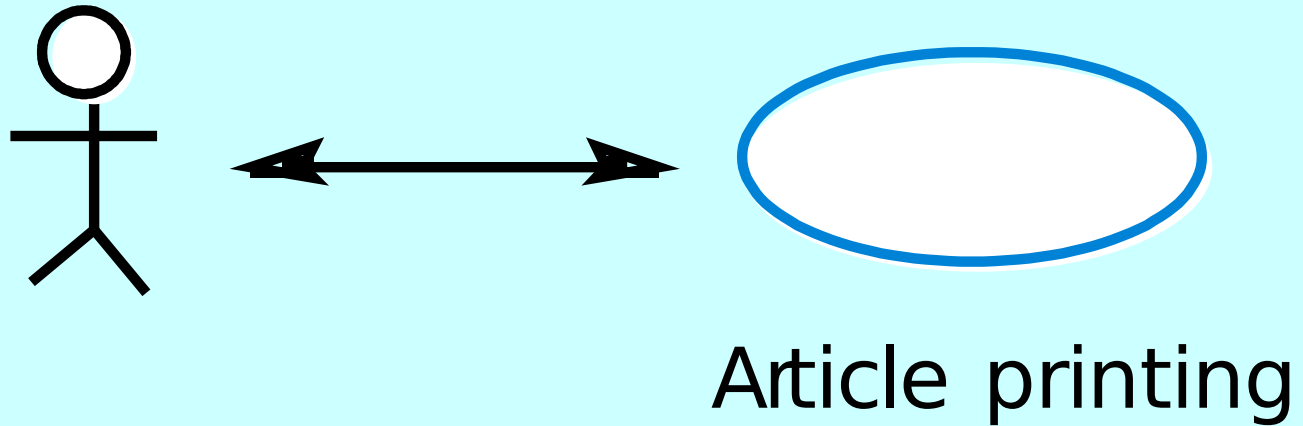
Other activities: Simultaneous downloads of other articles.

System state on completion: User is logged on. The downloaded article has been deleted from LIBSYS workspace if it has been flagged as print-only.

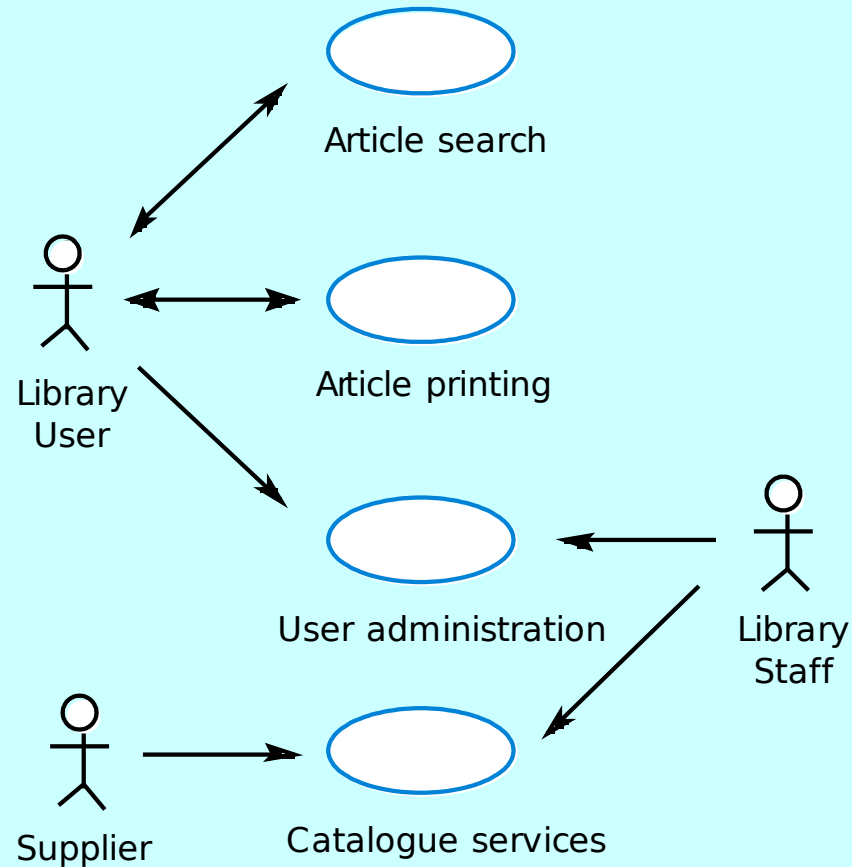
Use cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

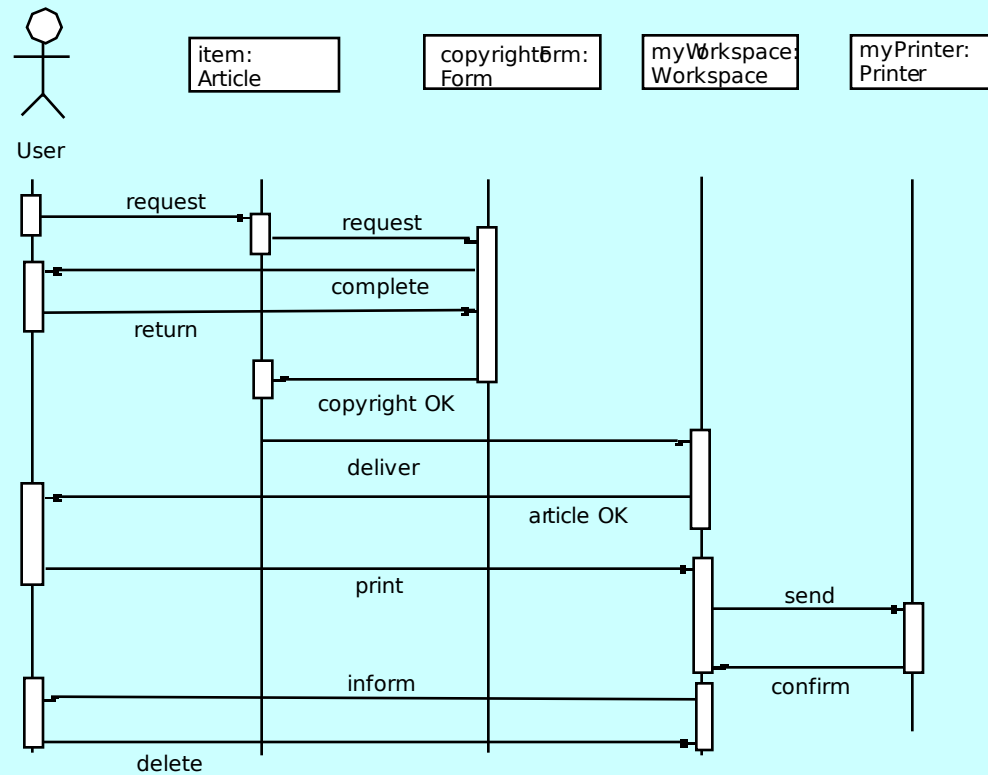
Article printing use-case



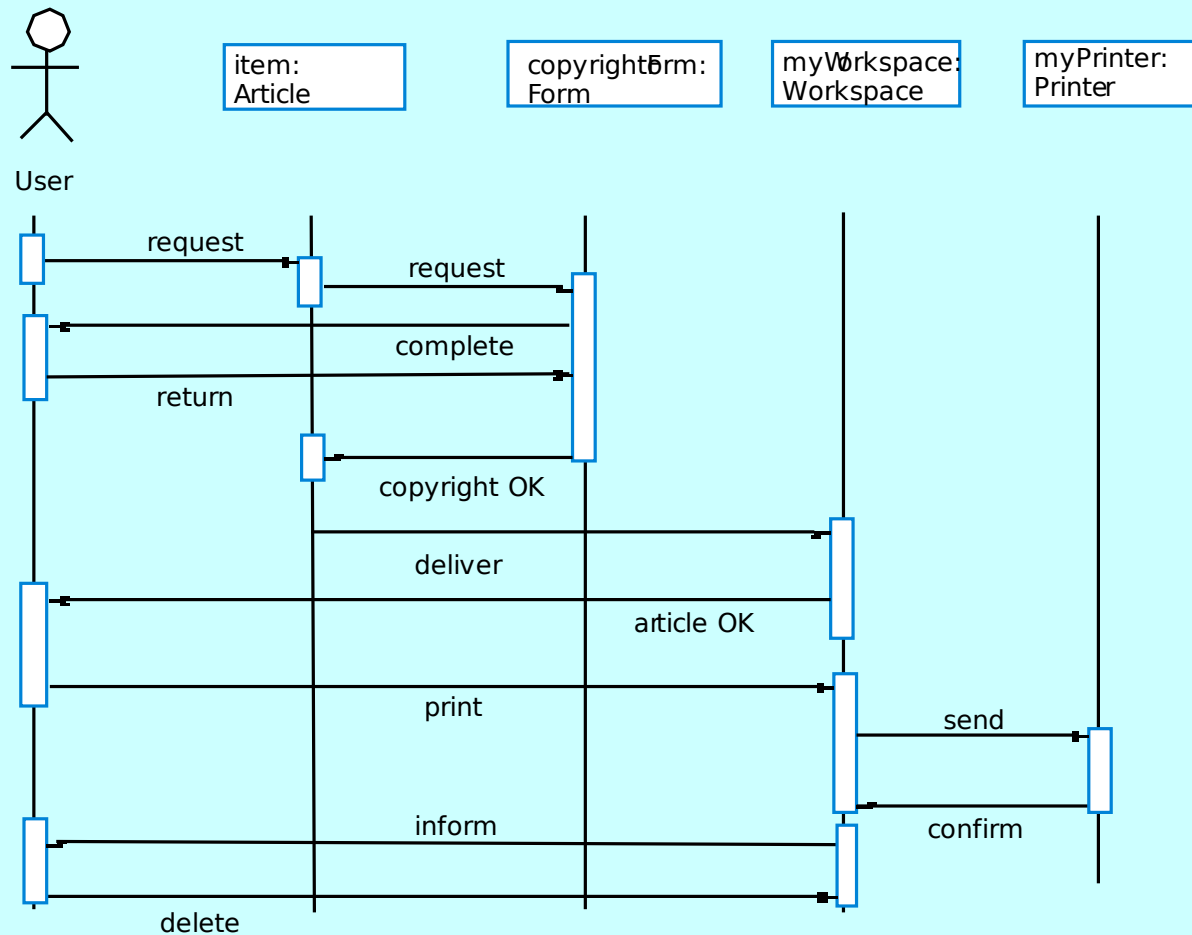
LIBSYS use cases



Article printing



Print article sequence



Social and organisational factors

- Software systems are used in a social and organisational context. This can influence or even dominate the system requirements.
- Social and organisational factors are not a single viewpoint but are influences on all viewpoints.
- Good analysts must be sensitive to these factors but currently no systematic way to tackle their analysis.

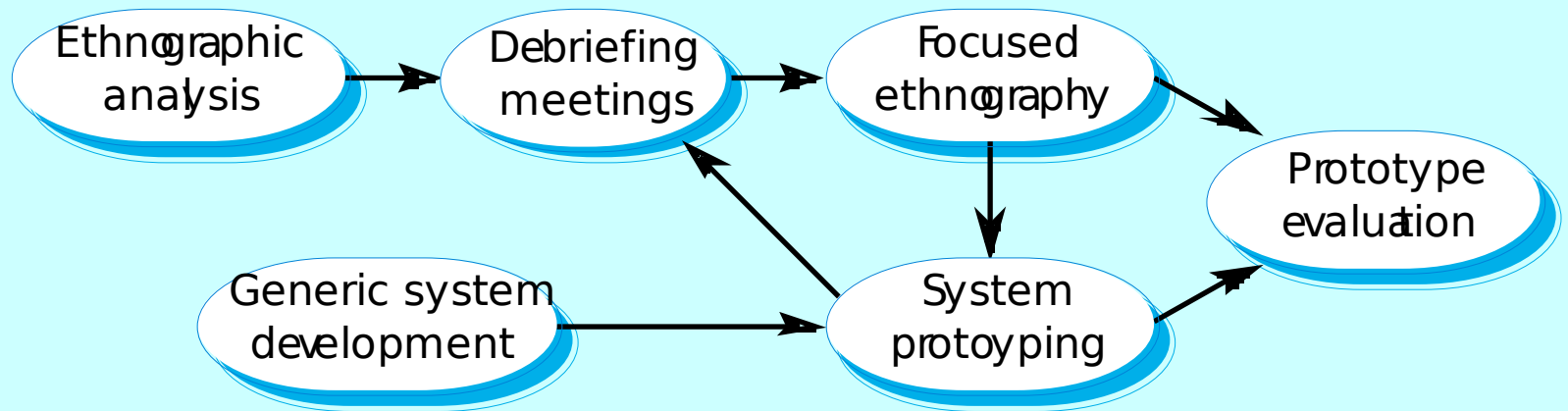
Ethnography

- A social scientists spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Focused ethnography

- Developed in a project studying the air traffic control process
- Combines ethnography with prototyping
- Prototype development results in unanswered questions which focus the ethnographic analysis.
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

Ethnography and prototyping



Scope of ethnography

- Requirements that are derived from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.

Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
 - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking

- **Validity**. Does the system provide the functions which best support the customer's needs?
- **Consistency**. Are there any requirements conflicts?
- **Completeness**. Are all functions required by the customer included?
- **Realism**. Can the requirements be implemented given available budget and technology
- **Verifiability**. Can the requirements be checked?

Requirements validation techniques

- Requirements reviews
 - Systematic manual analysis of the requirements.
- Prototyping
 - Using an executable model of the system to check requirements. Covered in Chapter 17.
- Test-case generation
 - Developing tests for requirements to check testability.

Requirements reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

Review checks

- **Verifiability**. Is the requirement realistically testable?
- **Comprehensibility**. Is the requirement properly understood?
- **Traceability**. Is the origin of the requirement clearly stated?
- **Adaptability**. Can the requirement be changed without a large impact on other requirements?

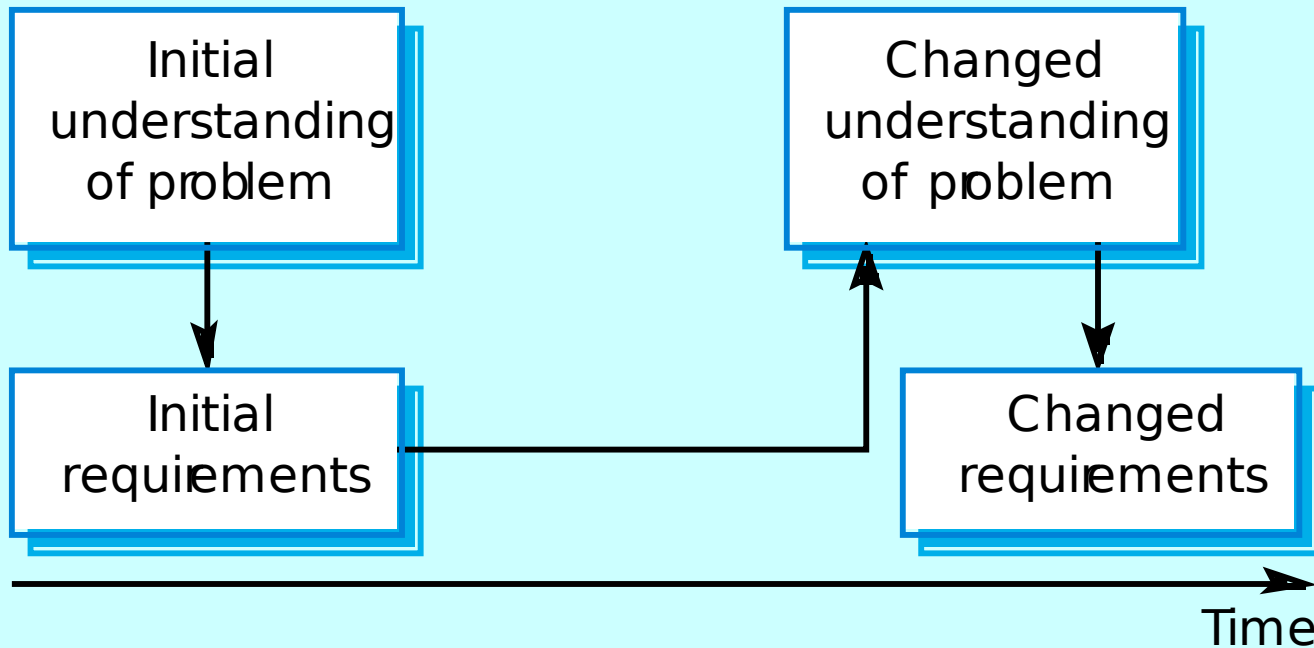
Requirements management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- Requirements are inevitably incomplete and inconsistent
 - New requirements emerge during the process as business needs change and a better understanding of the system is developed;
 - Different viewpoints have different requirements and these are often contradictory.

Requirements change

- The priority of requirements from different viewpoints changes during the development process.
- System customers may specify requirements from a business perspective that conflict with end-user requirements.
- The business and technical environment of the system changes during its development.

Requirements evolution



Enduring and volatile requirements

- **Enduring requirements.** Stable requirements derived from the core activity of the customer organisation. E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- **Volatile requirements.** Requirements which change during development or when the system is in use. In a hospital, requirements derived from health-care policy

Requirements classification

Requirement Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.
Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

Requirements management planning

- During the requirements engineering process, you have to plan:
 - Requirements identification
 - How requirements are individually identified;
 - A change management process
 - The process followed when analysing a requirements change;
 - Traceability policies
 - The amount of information about requirements relationships that is maintained;
 - CASE tool support
 - The tool support required to help manage requirements change;

Traceability

- Traceability is concerned with the relationships between requirements, their sources and the system design
- Source traceability
 - Links from requirements to stakeholders who proposed these requirements;
- Requirements traceability
 - Links between dependent requirements;
- Design traceability
 - Links from the requirements to the design;

A traceability matrix

Req. id	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		D	R					
1.2			D			D		D
1.3	R			R				
2.1			R		D			D
2.2								D
2.3		R		D				
3.1								R
3.2							R	

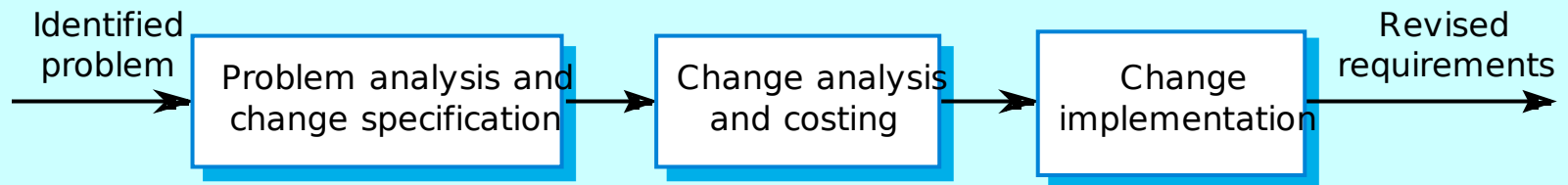
CASE tool support

- Requirements storage
 - Requirements should be managed in a secure, managed data store.
- Change management
 - The process of change management is a workflow process whose stages can be defined and information flow between these stages partially automated.
- Traceability management
 - Automated retrieval of the links between requirements.

Requirements change management

- Should apply to all proposed changes to the requirements.
- Principal stages
 - Problem analysis. Discuss requirements problem and propose change;
 - Change analysis and costing. Assess effects of change on other requirements;
 - Change implementation. Modify requirements document and other documents to reflect change.

Change management



Key points

- The requirements engineering process includes a feasibility study, requirements elicitation and analysis, requirements specification and requirements management.
- Requirements elicitation and analysis is iterative involving domain understanding, requirements collection, classification, structuring, prioritisation and validation.
- Systems have multiple stakeholders with different requirements.

Key points

- Social and organisation factors influence system requirements.
- Requirements validation is concerned with checks for validity, consistency, completeness, realism and verifiability.
- Business changes inevitably lead to changing requirements.
- Requirements management includes planning and change management.