

U

O

W

Software Requirements, Specifications and Formal Methods

Lecture 01 – Software Engineering Introduction

Slides mainly taken from **Dr. Fenghui Ren** of UOW



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Chapter 1- Introduction



Software engineering

- The economies of ALL developed nations are dependent on software.
- More and more systems are software controlled.
- Expenditure on software represents a significant fraction of GDP in all developed countries.
- Software engineering is concerned with **theories**, **methods** and **tools** for professional software development.



Software project failure

- *Increasing system complexity*
 - As new software engineering techniques help us to build **larger**, **more complex** systems, the demands change. Systems have to be built and delivered **more quickly**; systems have to have new capabilities that were previously thought to be impossible.
- *Failure to use software engineering methods*
 - It is fairly easy to write computer programs without using software engineering methods and techniques. Many companies have drifted into software development as their products and services have evolved. They do not use software engineering methods in their everyday work. Consequently, their software is often **more expensive** and **less reliable** than it should be.



Software engineering

- Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use.
- Engineering discipline
 - Using appropriate theories and methods to solve problems bearing in mind organizational and financial constraints.
- All aspects of software production
 - Not just technical process of development. Also project management and the development of tools, methods etc. to support software production.



Importance of software engineering

- More and more, individuals and society rely on advanced software systems. We need to be able to produce **reliable** and **trustworthy** systems **economically** and **quickly**.
- It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.



Software process activities

- **Software specification**, where customers and engineers **define** the software that is to be produced and the constraints on its operation.
- **Software development**, where the software is designed and programmed.
- **Software validation**, where the software is checked to ensure that it is what the customer requires.
- **Software evolution**, where the software is modified to reflect changing customer and market requirements.



Frequently asked questions about software engineering

Question	Answer
What is software?	Computer programs and associated documentation. Software products may be developed for a particular customer or may be developed for a general market.
What are the attributes of good software?	Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.
What is software engineering?	Software engineering is an engineering discipline that is concerned with all aspects of software production.
What are the fundamental software engineering activities?	Software specification, software development, software validation and software evolution.
What is the difference between software engineering and computer science?	Computer science focuses on theory and fundamentals; software engineering is concerned with the practicalities of developing and delivering useful software.
What is the difference between software engineering and system engineering?	System engineering is concerned with all aspects of computer-based systems development including hardware, software and process engineering. Software engineering is part of this more general process.

Frequently asked questions about software engineering

Question	Answer
What are the key challenges facing software engineering?	Coping with increasing diversity, demands for reduced delivery times and developing trustworthy software.
What are the costs of software engineering?	Roughly 60% of software costs are development costs, 40% are testing costs. For custom software, evolution costs often exceed development costs.
What are the best software engineering techniques and methods?	While all software projects have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. You can't, therefore, say that one method is better than another.
What differences has the web made to software engineering?	The web has led to the availability of software services and the possibility of developing highly distributed service-based systems. Web-based systems development has led to important advances in programming languages and software reuse.

Essential attributes of good software

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

The software process

- A structured set of activities required to develop a software system.
- Many different software processes but all involve:
 - Specification – defining what the system should do;
 - Design and implementation – defining the organization of the system and implementing the system;
 - Validation – checking that it does what the customer wants;
 - Evolution – changing the system in response to changing customer needs.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.



System stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholder types
 - End users
 - System managers
 - System owners
 - External stakeholders

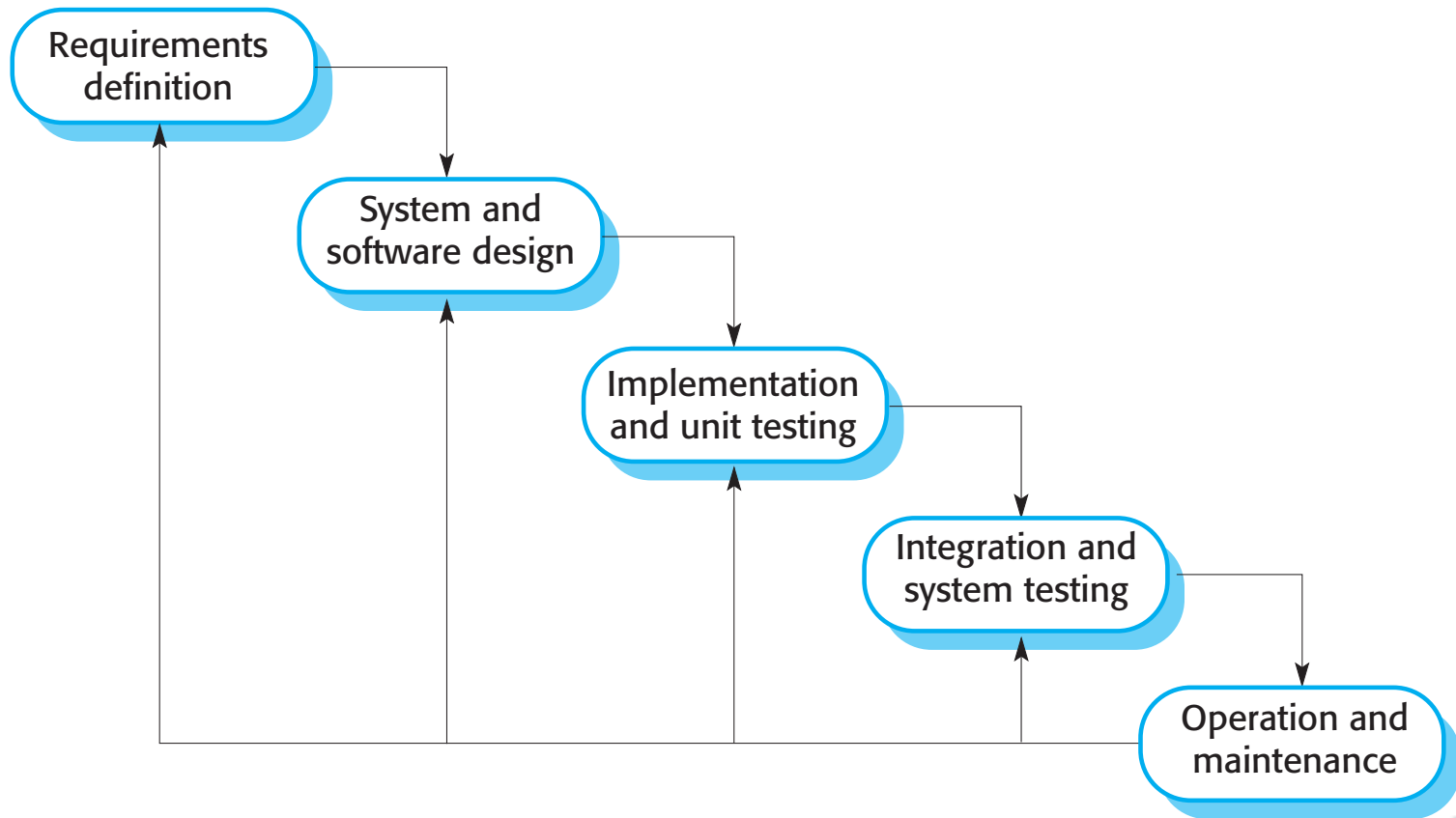


Software process models

- The waterfall model
 - Plan-driven model. Separate and distinct phases of specification and development.
- Incremental development
 - Specification, development and validation are interleaved. May be plan-driven or agile.
- Integration and configuration
 - The system is assembled from existing configurable components. May be plan-driven or agile.
- In practice, most large systems are developed using a process that incorporates elements from all of these models.



The waterfall model



Waterfall model phases

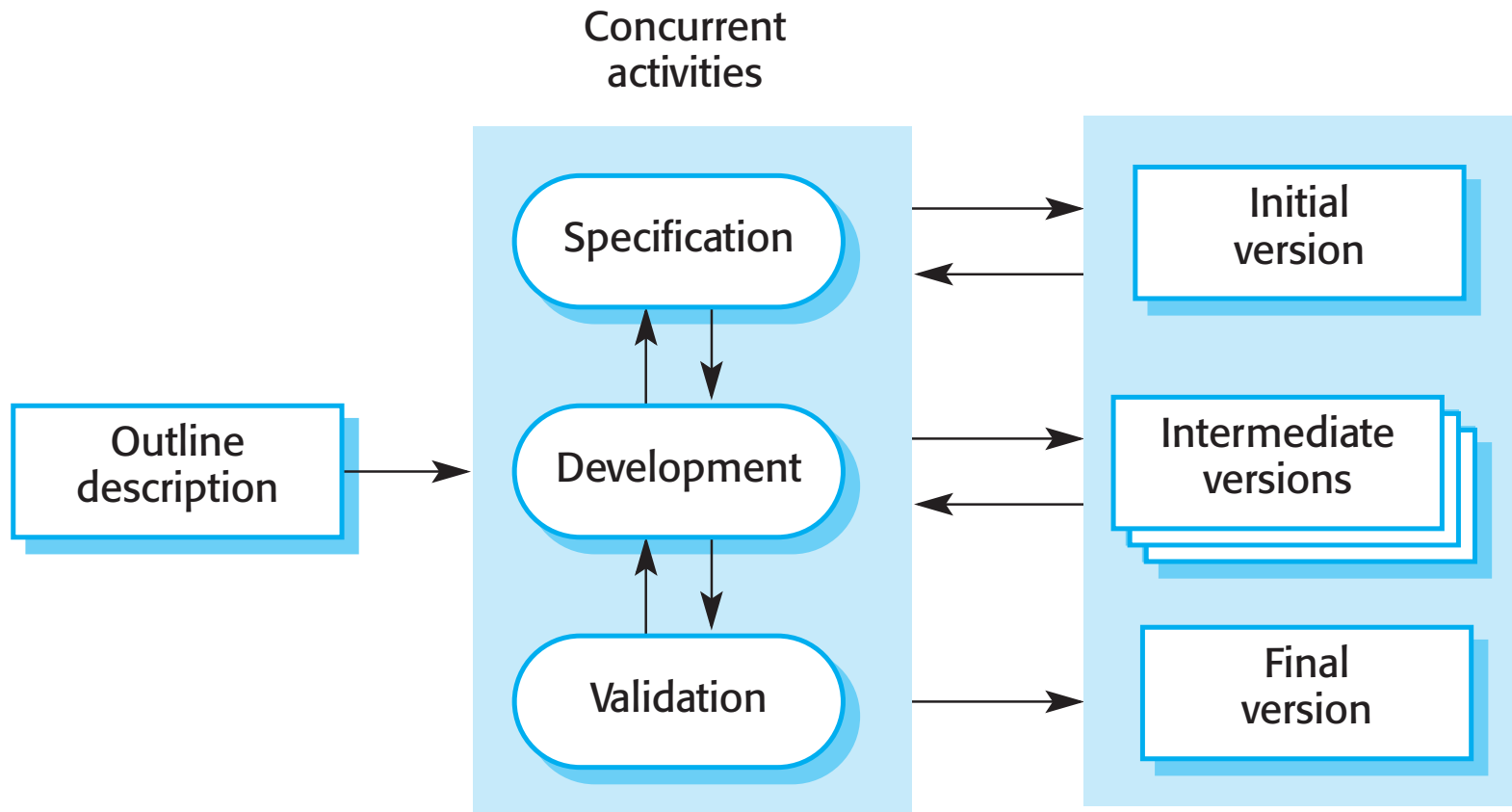
- There are separate identified phases in the waterfall model:
 - Requirements analysis and definition
 - System and software design
 - Implementation and unit testing
 - Integration and system testing
 - Operation and maintenance
- The main drawback of the waterfall model is the **difficulty of accommodating change** after the process is underway. In principle, a phase has to be complete before moving onto the next phase.

Waterfall model problems

- Inflexible partitioning of the project into distinct stages makes it difficult to respond to changing customer requirements.
 - Therefore, this model is only appropriate when the requirements are well-understood and changes will be fairly limited during the design process.
 - Few business systems have stable requirements.
- The waterfall model is mostly used for large systems engineering projects where a system is developed at several sites.
 - In those circumstances, the plan-driven nature of the waterfall model helps coordinate the work.



Incremental development



Incremental development benefits

- The cost of accommodating changing customer requirements is reduced.
 - The amount of analysis and documentation that has to be redone is much less than is required with the waterfall model.
- It is easier to get customer feedback on the development work that has been done.
 - Customers can comment on demonstrations of the software and see how much has been implemented.
- More rapid delivery and deployment of useful software to the customer is possible.
 - Customers are able to use and gain value from the software earlier than is possible with a waterfall process.



Incremental development problems

- The process is not visible.
 - Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
- System structure tends to degrade as new increments are added.
 - Unless time and money is spent on refactoring to improve the software, regular change tends to corrupt its structure. Incorporating further software changes becomes increasingly difficult and costly.



Integration and configuration

- Based on software **reuse** where systems are integrated from **existing** components or application systems (sometimes called COTS -Commercial-off-the-shelf systems).
- Reused elements may be configured to **adapt** their behaviour and functionality to a user's requirements
- Reuse is now the standard approach for building many types of business system

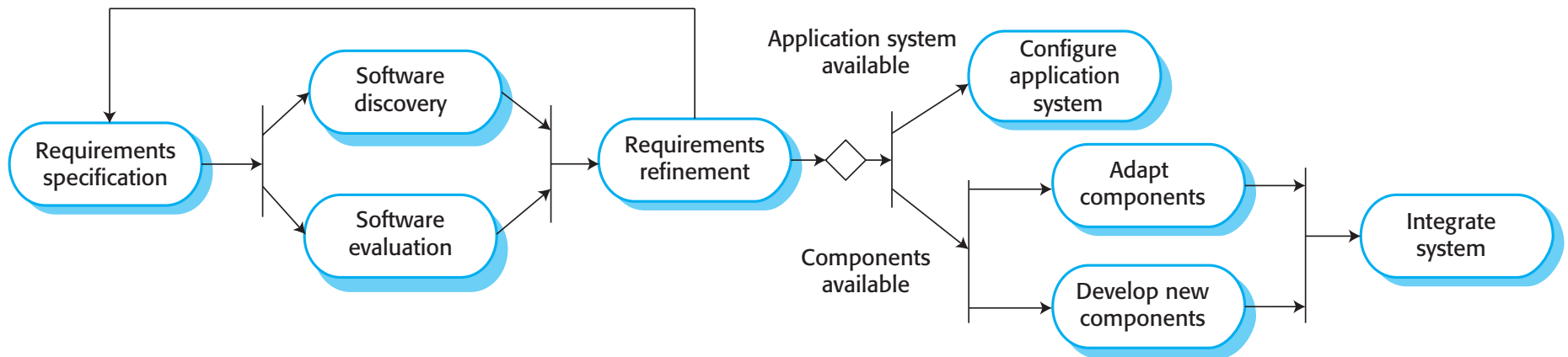


Types of reusable software

- Stand-alone application systems that are configured for use in a particular environment.
- Collections of objects that are developed as a package to be integrated with a component framework such as .NET or J2EE.
- Web services that are developed according to service standards and which are available for remote invocation.



Reuse-oriented software engineering



Key process stages of integration and configuration

- Requirements specification
- Software discovery and evaluation
- Requirements refinement
- Application system configuration
- Component adaptation and integration



Advantages and disadvantages

- Reduced costs and risks as less software is developed from scratch
- Faster delivery and deployment of system
- But requirements compromises are inevitable so system may not meet real needs of users
- Loss of control over evolution of reused system elements



Requirements engineering

- The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.
- The system requirements 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.



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.



Mentcare: A patient information system for mental health care

- A patient information system to support mental health care is a medical information system that maintains information about patients suffering from mental health problems and the treatments that they have received.
 - Most mental health patients do not require dedicated hospital treatment but need to attend specialist clinics regularly where they can meet a doctor who has detailed knowledge of their problems.
 - To make it easier for patients to attend, these clinics are not just run in hospitals. They may also be held in local medical practices or community centres.
-

Mentcare

- Mentcare is an information system that is intended for use in clinics.
- It makes use of a centralized database of patient information but has also been designed to run on a PC, so that it may be accessed and used from sites that do not have secure network connectivity.
- When the local systems have secure network access, they use patient information in the database but they can download and use local copies of patient records when they are disconnected.

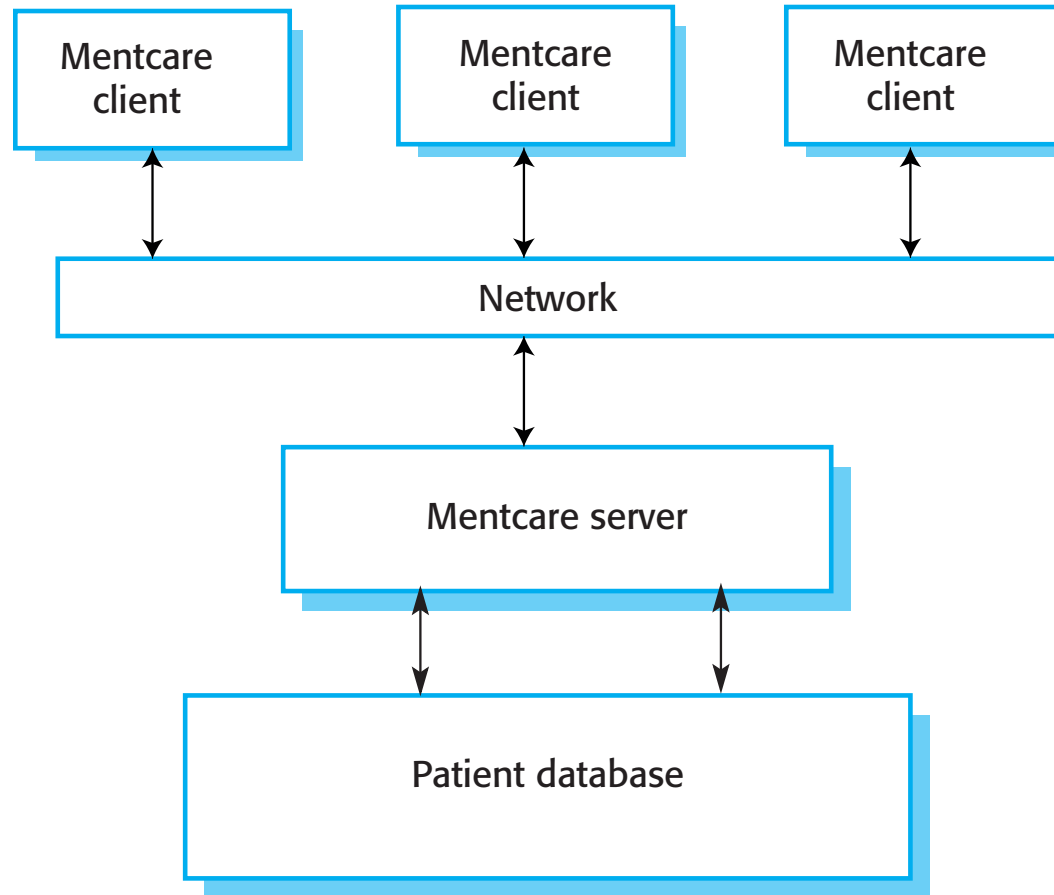


Mentcare goals

- To generate management information that allows health service managers to assess performance against local and government targets.
- To provide medical staff with timely information to support the treatment of patients.



The organization of the Mentcare system



Key features of the Mentcare system

- Individual care management
 - Clinicians can create records for patients, edit the information in the system, view patient history, etc. The system supports data summaries so that doctors can quickly learn about the key problems and treatments that have been prescribed.
- Patient monitoring
 - The system monitors the records of patients that are involved in treatment and issues warnings if possible problems are detected.
- Administrative reporting
 - The system generates monthly management reports showing the number of patients treated at each clinic, the number of patients who have entered and left the care system, number of patients sectioned, the drugs prescribed and their costs, etc.

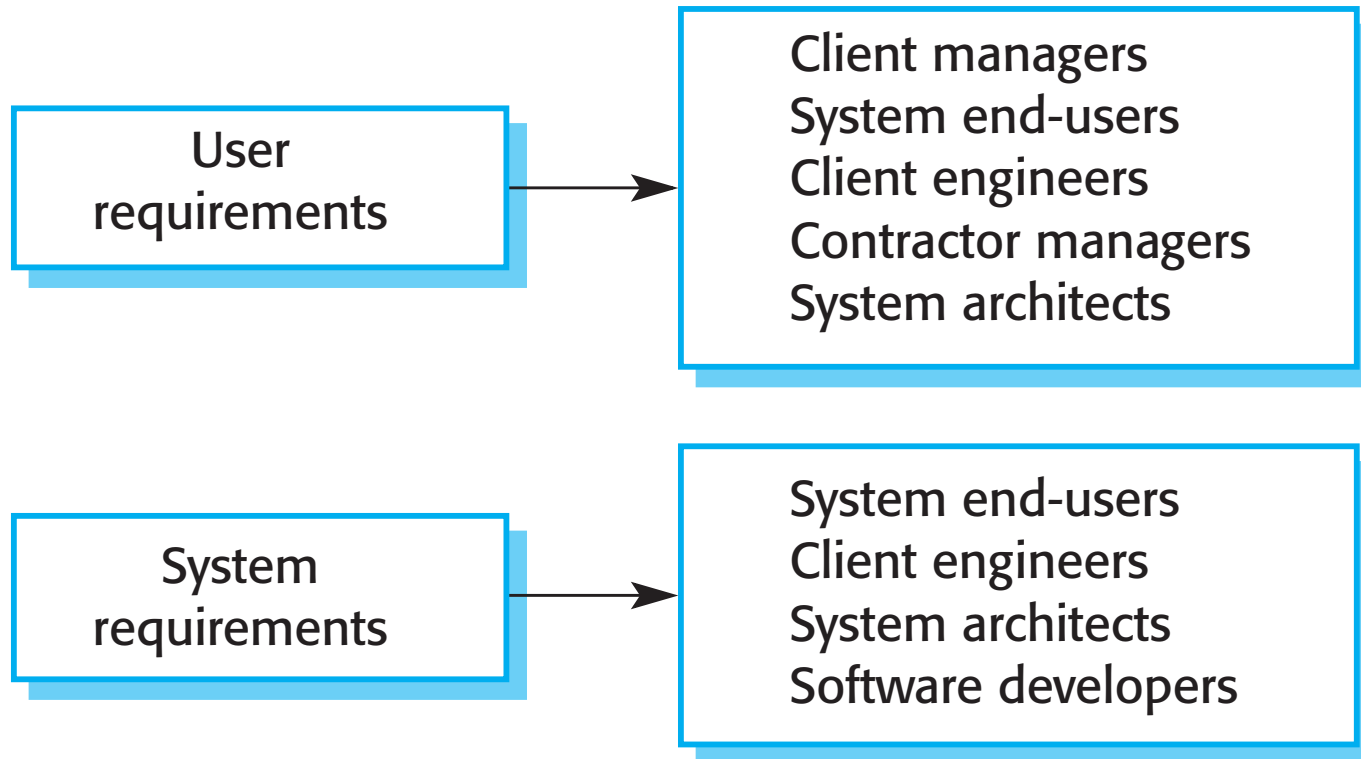


Mentcare system concerns

- Privacy
 - It is essential that patient information is confidential and is never disclosed to anyone apart from authorised medical staff and the patient themselves.
- Safety
 - Some mental illnesses cause patients to become suicidal or a danger to other people. Wherever possible, the system should warn medical staff about potentially suicidal or dangerous patients.
 - The system must be available when needed otherwise safety may be compromised and it may be impossible to prescribe the correct medication to patients.



Readers of different types of requirements specification



User and system requirements

User requirements definition

- 1.** The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

- 1.1** On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
- 1.2** The system shall generate the report for printing after 17.30 on the last working day of the month.
- 1.3** A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
- 1.4** If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
- 1.5** Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.



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.
 - May state what the system should not do.
- Non-functional requirements
 - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
 - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
 - Constraints on the system from the domain of operation

Constraints

Constraints are global requirements. They apply to the entire product, and need to be defined before beginning work on gathering the requirements.

Example: the purpose of the product is a constraint – individual requirements must contribute to this purpose.



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.
- Functional system requirements should describe the system services in detail.



Mentcare system: functional requirements

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.



Requirements imprecision

- Problems arise when functional requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘search’ in requirement 1
 - User intention – search for a patient name across all appointments in all clinics;
 - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.



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, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.



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 IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.



Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
 - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
 - It may also generate requirements that restrict existing requirements.

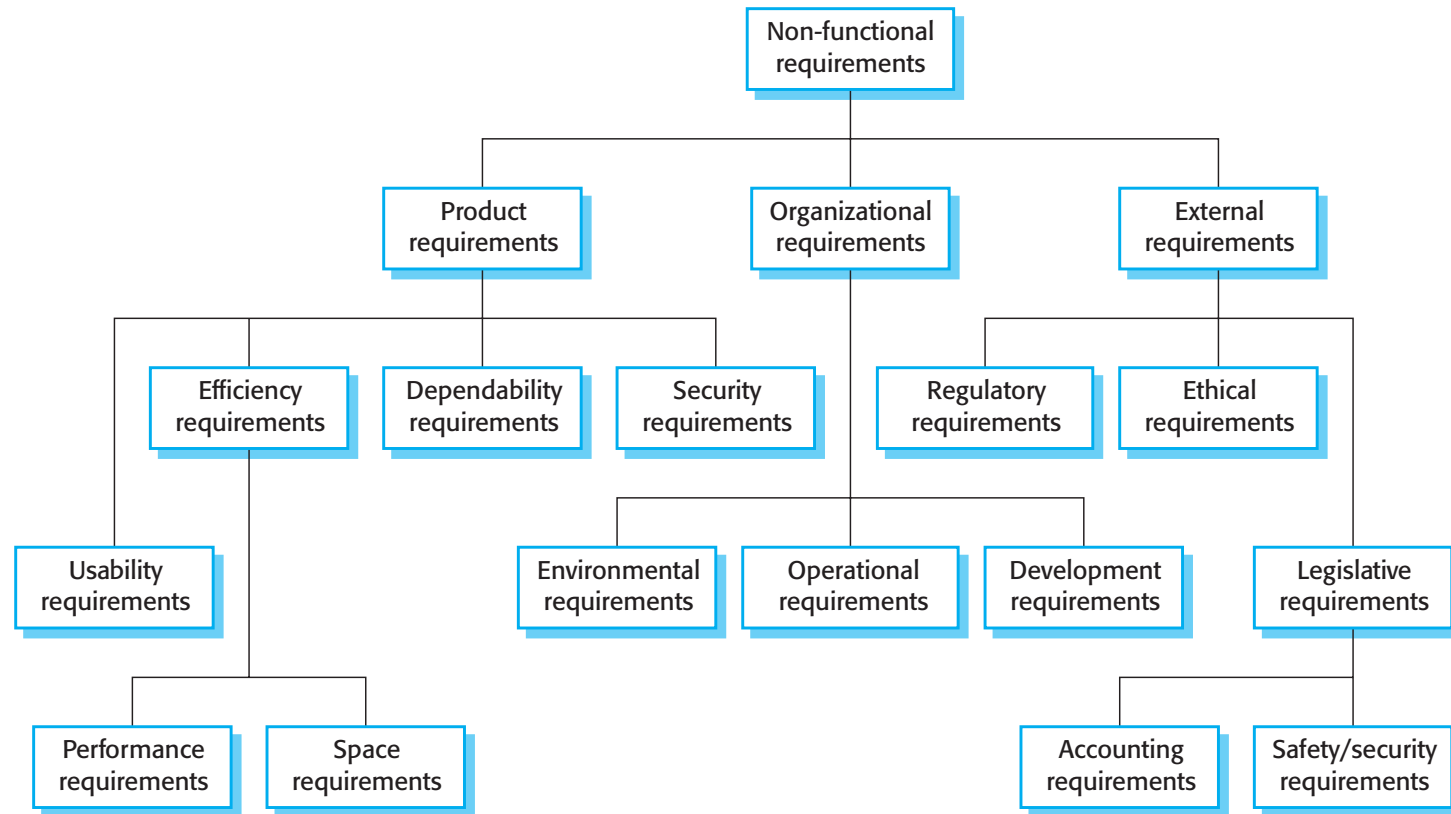


Non-functional classifications

- 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.



Types of nonfunctional requirement



Examples of nonfunctional requirements in the Mentcare system

Product requirement

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement

Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement

The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.



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.



Usability requirements

- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)
- Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Testable non-functional requirement)



Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes 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

Requirement Documents

Requirement document describes the following

- The services and functions which the system should provide.
- The constraints under which the system must operate.
- Overall properties of the system, i.e. constraints on the system's emergent properties.
- Definitions of other systems which the system must integrate with
- Information about the application domain of the system, e.g. how to carry out particular types of computation.
- Constraints on the process used to develop the system.



The Template

All of requirements and constraints are written in a requirements specification. This is a complete description of the product's capabilities. Specification can take many forms such as informal specifications (diagrams) or formal specifications with well established mathematical models.

The IEEE Requirements Specification Template is a compartmentalized container for a requirements description. It gives you a framework for writing a specification.

The IEEE Template of Requirement Documents

Product Constraints – restrictions and limitations that apply to the project

1. The purpose of the product
2. The client, customer and other stakeholder
3. Users of the product
4. Requirements constraints
5. Naming conventions and definitions
6. Relevant facts
7. Assumptions



The Template (continue)

Functional requirements

- 8. The scope of the product
- 9. Functional and data requirements

Non-functional requirements

- 10. Look and feel requirements
- 11. Usability requirements
- 12. Performance requirements
- 13. Operational requirements
- 14. Maintainability and portability requirements
- 15. Security requirements
- 16. Cultural and political requirements

Legal requirements



The Template (continue)

Project issues

- 18. Open issues
- 19. Off-the-shelf solutions
- 20. New problems
- 21. Tasks
- 22. Cutover
- 23. Risk
- 24. Costs
- 25. User documentation
- 26. Waiting room



Guideline for Writing Requirements

1. Define standard templates for describing requirements.
2. Use language simply, consistently, and concisely.
3. Use diagrams appropriately.
4. Supplement natural language with other descriptions of requirements.
5. Specify requirements quantitatively.

We will learn formal specification techniques in this subject.