U O W

Software Requirements, Specifications and Formal Methods

Dr. Shixun Huang

UNIVERSITY OF WOLLONGONG AUSTRALIA

Lecture 1- Introduction

Software engineering

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

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; larger, even more complex systems are required;
  - 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.
- For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

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 efficient, maintainable and user-friendly.
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?	Requirement 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 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.

System stakeholders

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

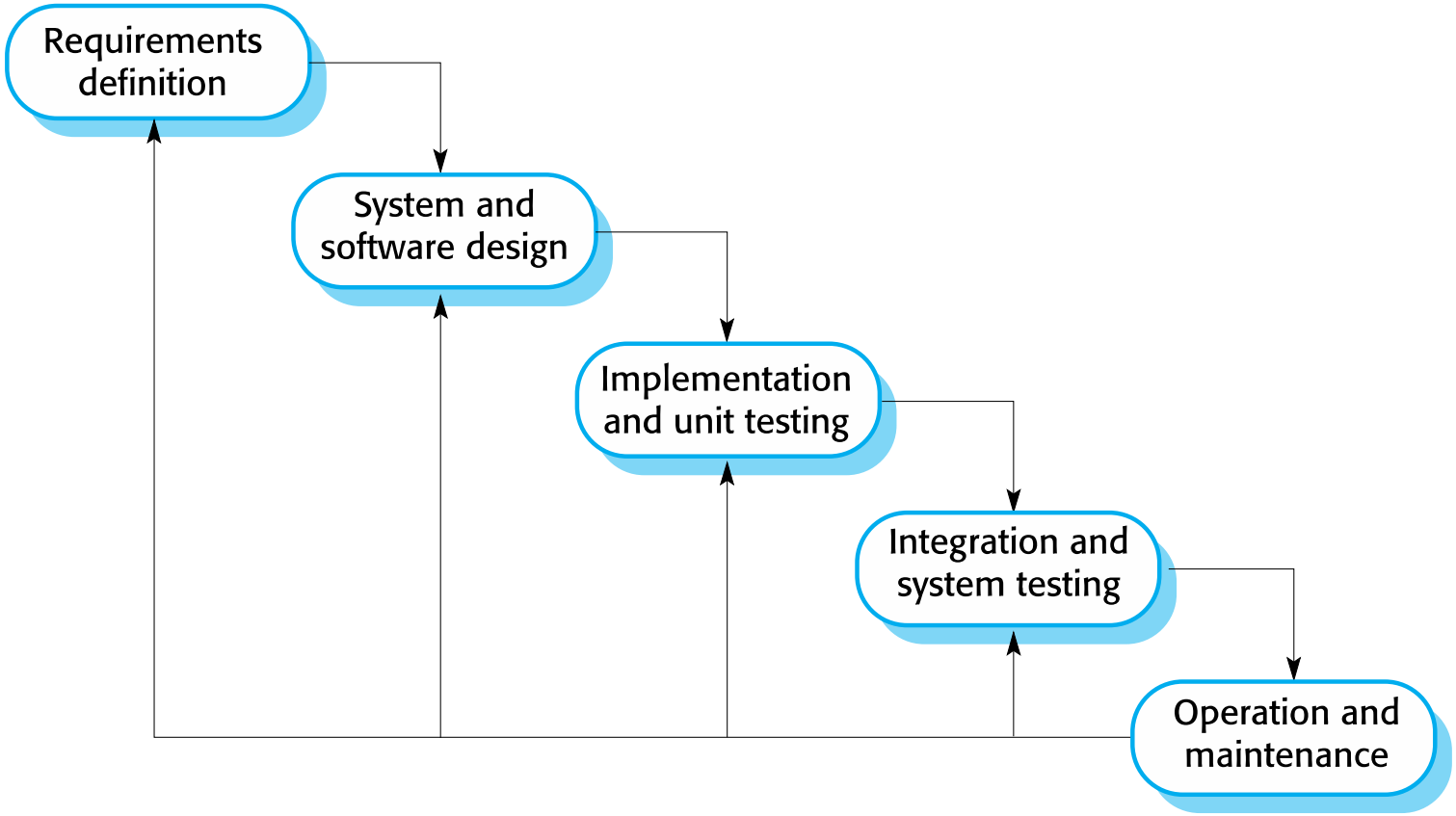
The software process

- 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.

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



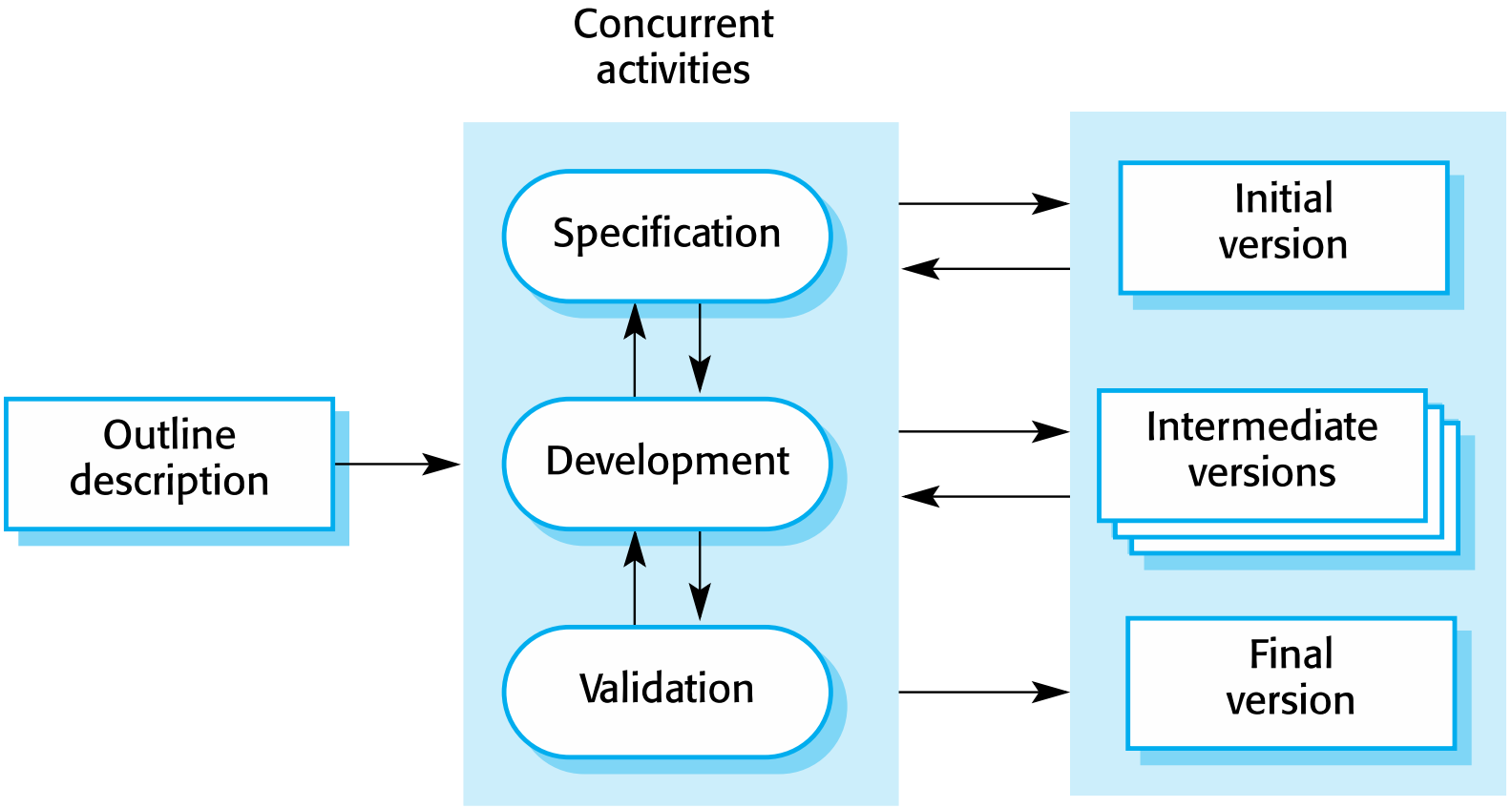
Waterfall model phases

- 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.

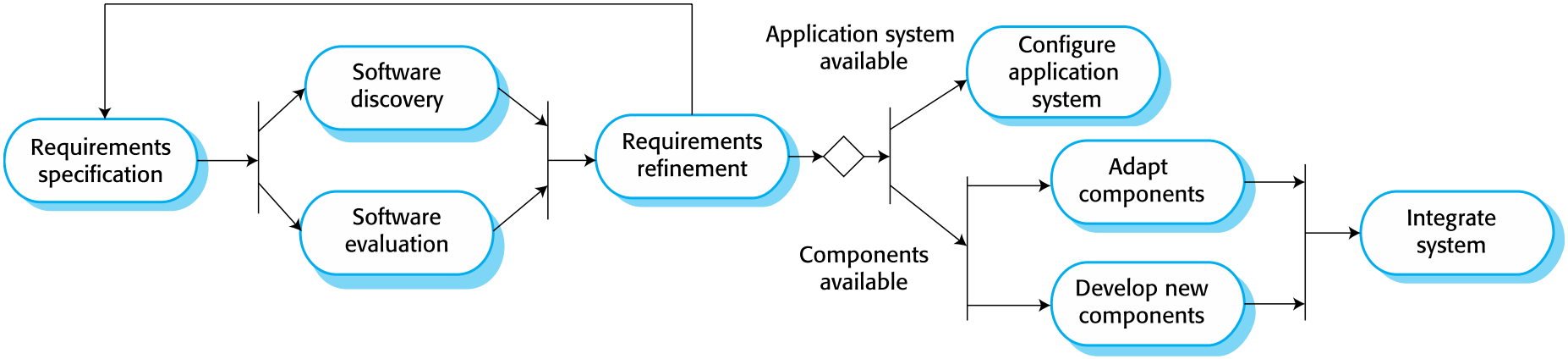
Integration and configuration

- 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 (sometimes called COTS) 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



Requirements engineering

Definition

- Requirements engineering is the branch of software engineering concerned with the real-world goals for, functions of, and constraints on software systems. It is also concerned with the relationship of these factors to precise specifications of software behavior, and to their evolution over time and across software families. (Zave 1997)

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.

- These varying representation forms occur because
  - Stakeholders have needs at different levels, hence, depend on different abstraction representations.
  - Stakeholders also have varying abilities to make and read these representations (e.g., a business customer vs. a design engineer), leading to diverse quality in the requirements.

Requirements vs. Goals

Some engineers and customers often confuse requirements and goals

- Goals are high-level objectives of a business, organization, or system,
  - A goal: to build the safest bridge in the world
- Goals are difficult to prove and evolve as stakeholders change their minds

- Requirements specify how a goal should be accomplished by a proposed system.
  - Requirements: building techniques, bridge materials, qualifications of the contractor and engineers,

Requirements Level Classification

Sommerville (2005) suggests organizing them into three levels of abstraction:

- User requirements
  - discovered first, are used as the basis for final acceptance testing.
- System requirements
  - Developed after the user requirements, are used as the basis for the integration testing that precedes acceptance testing.
- Design specifications
  - Are derived from the system requirements are used for unit testing as each code unit is implemented.

User Requirements

- User requirements are abstract statements written in natural language with accompanying informal diagrams.
- They specify what services (user functionality) the system is expected to provide and any constraints.
- Collected user requirements often appear as a “concept of operations” (Conops) document.
- In many situations user stories can play the role of user requirements.

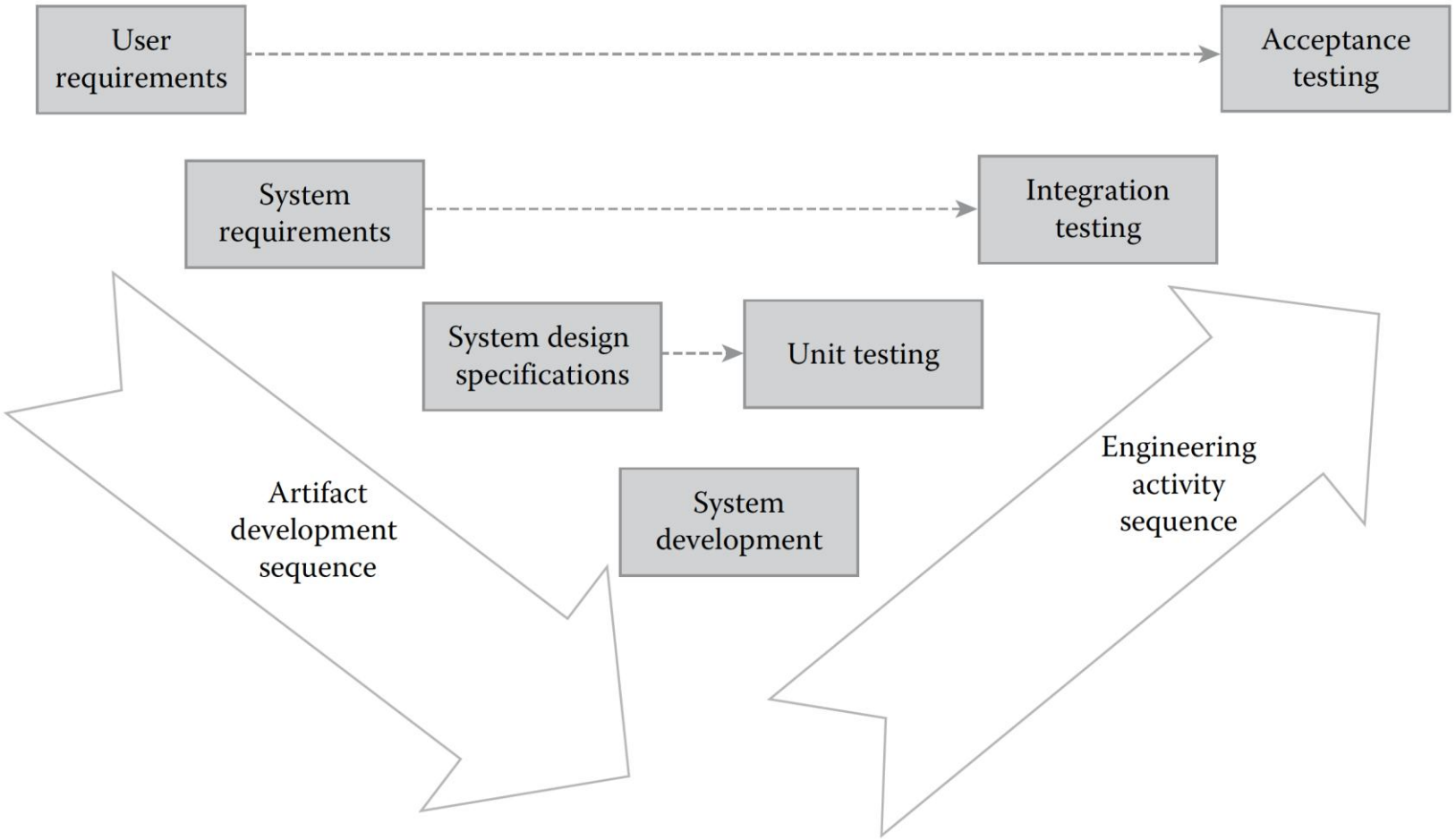
System Requirements

- System requirements are detailed descriptions of the services and constraints.
- System requirements are sometimes referred to as functional specification or technical annex (a term that is rarely used).
- These requirements are derived from analysis of the user requirements and they should be structured and precise.
- The requirements are collected in a systems requirements specification (SRS) document.
- Use cases can play the role of system requirement in many situations.

Design Requirements

- Finally, design specifications emerge from the analysis and design documentation used as the basis for implementation by developers.
- The system design specification is essentially derived directly from analysis of the system requirements specification.

Requirements Level Classification



Requirements Level Classification

For an airline baggage handling system

- A user requirement
  - The system shall track baggage from check-in to arrival.
- Some related system requirements
  - Each bag processed shall trigger a baggage event.
- Some design specifications
  - To handle each triggered baggage event, the system shall use an event-driven architecture with a message queue to ensure real-time processing and tracking of each bag

Requirements Level Classification

For a pet store POS system

- A user requirement
  - The system shall accurately compute sale totals, including discounts, taxes, refunds, and rebates; print an accurate receipt; and update inventory counts accordingly
- Some related system requirements
  - The system shall integrate with a payment processing module to compute the correct sale total, including taxes, discounts, refunds, and rebates.
  - The system shall interface with an inventory management module to update inventory counts after each sale.

Requirements Level Classification

For a pet store POS system

- Some design specifications
  - The system shall use a relational database with tables for customers, products, inventory, and transactions to ensure efficient storage and retrieval of data.
  - The system shall provide a graphical user interface (GUI) with intuitive workflows for cashier operations, including product scanning, manual input for discounts, and receipt printing.
- The systems specification in the appendices also contains numerous specifications organized by level for your exploration.

Requirements Specifications Types

Another taxonomy for requirements specifications focuses on the type of requirement

- Functional requirements (FRs)
- Non-functional requirements (NFRs)
- Domain requirements

Functional Requirements

Functional requirements (FRs)

- describe the services the system should provide and how the system will react to its inputs.
- need to explicitly state certain behaviors that the system should not do
- can be high level and general or they can be detailed, expressing inputs, outputs, exceptions
- natural language, visual models, formal methods

Functional Requirements

Functional requirements (FRs) in the POS system

- 4.1 When the operator presses the “total” button, the current sale enters the closed-out state.
  - 4.1.1 When a sale enters the closed-out state, a total for each nonsale item is computed as number of items times the list price of the item.
  - 4.1.2 When a sale enters the closed-out state, a total for each sale item is computed.

Non-functional Requirements

Non-functional requirements

- describe how the system behaves with respect to some observable attributes such as reliability, reusability, maintainability, etc.
- are more important in distinguishing between the competing products
- are subjective, relative, and they tend to become scattered among multiple modules when they are mapped from the requirements domain to the solution space.
- can often interact

Non-functional Requirements

Despite the challenging nature of NFRs, reports consistently indicate that neglecting them can lead to catastrophic project failures, or at the very least, to considerable delays and consequently to significant increases in the final cost

- In 1992, The London Ambulance Service (LAS) introduced a new computer aided dispatch system which was intended to automate the system that dispatched ambulances in response to calls from the public and the emergency services. This new system was extremely inefficient and ambulance response times increased markedly. The failure of the system was mainly due to a failure to consider “human and organizational factors” in the design of the system (Finkelstein and Dowell 1996).

Non-functional Requirements

Neto et al. (2000) enumerate some of the well-known problems of the software development due to the NFRs omission:

- cost and schedule overruns,
- software systems discontinuation, and
- dissatisfaction of software systems users.

For all that, it is important to affirm that NFRs should affect all levels of the software/ systems life cycle and should be identified as soon as possible, and their elicitation must be accurate and complete.

Non-functional Requirements

To deal with the NFRs, five common NFR categories are identified

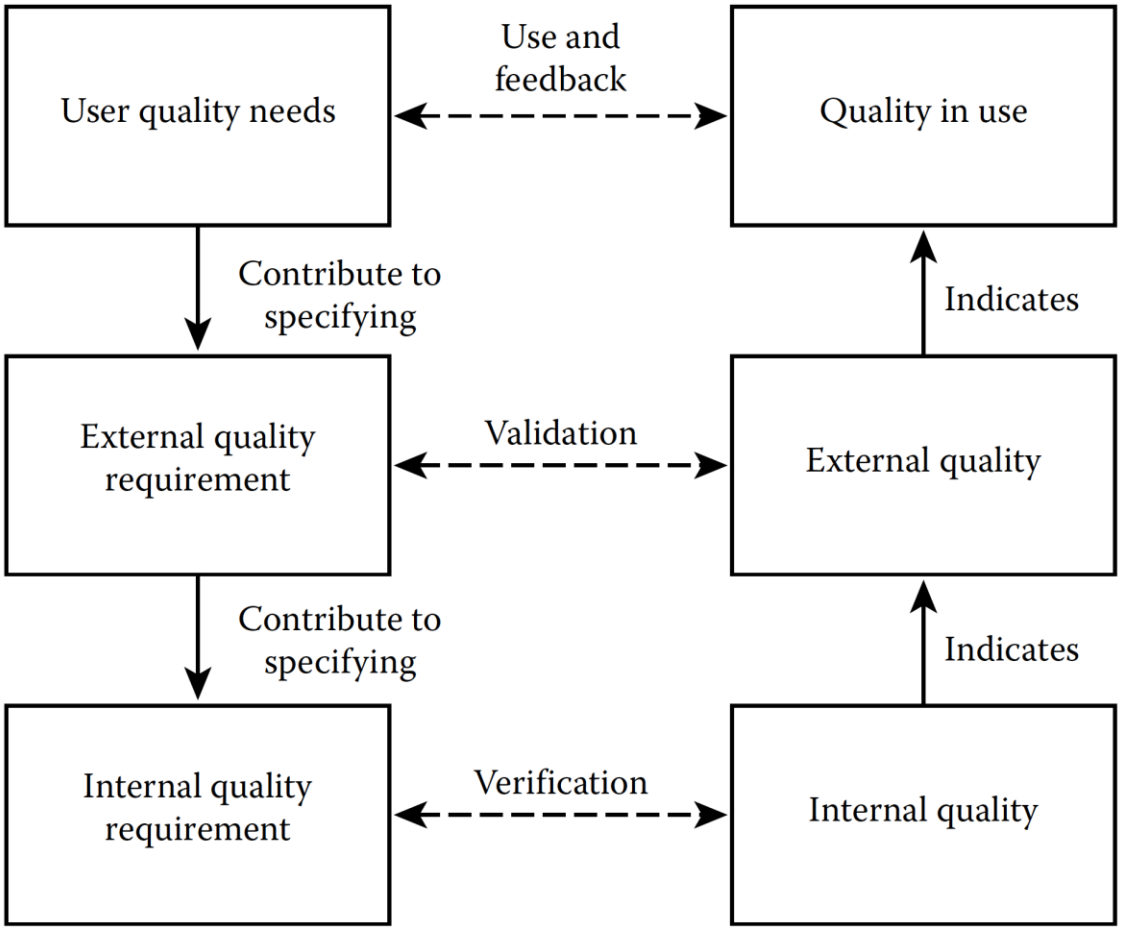
- quality,
- design,
- economic,
- operating, and
- political/cultural.

NFR Categories: Quality

Quality requirements are

- the most important category in the NFR world.
- the totality of characteristics of an entity that bear on its ability to satisfy stated and implied needs (ISO12601).
- may include safety, privacy, reliability, usability, and maintainability requirements.
- can be evaluated by measuring internal attributes, by measuring external attributes, or by measuring quality in use attributes.

Key Process Stages



NFR Categories: Design

Design/Implementation Constraints:

- Constraints are not usually subject to negotiation and, once agreed upon, are off-limits during design trade-offs.
- Constraints are defined as restrictions on the design of the system, or the process by which a system is developed, that do not affect the external behavior of the system but that must be fulfilled to meet technical, business, or contractual obligations (Leffingwell and Widrig 2003).
- A key property of a constraint is that a penalty or loss of some kind applies if the constraint is not respected.
- An example of design/implementation constraints includes the restrictions on using certain architectural patterns or specific programming languages.

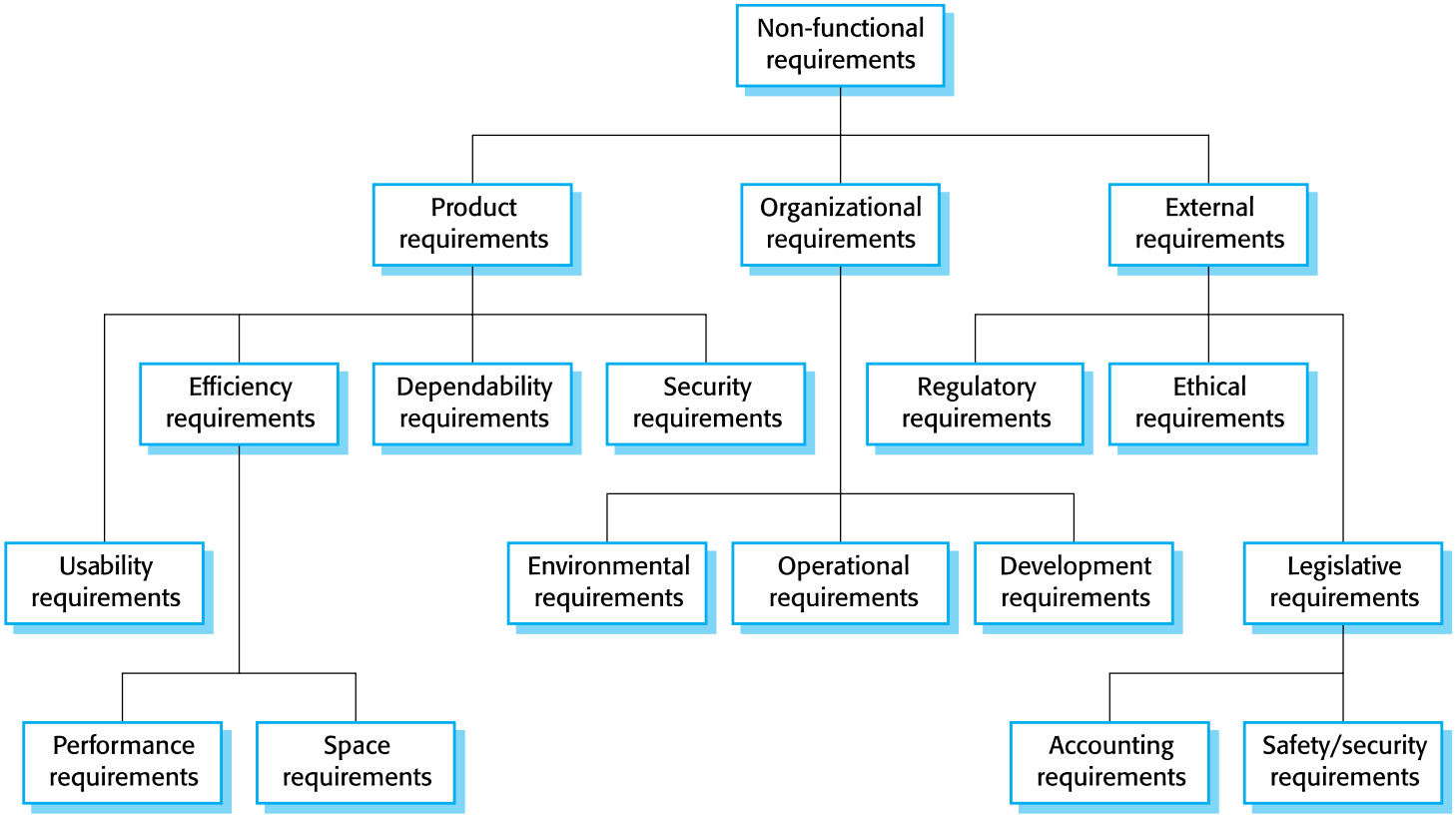
NFR Categories: others

- Economic Constraints: These are constraints which include the immediate and/or long-term development cost.

- Operating Constraints: These are constraints which include physical constraints, personnel availability, skill-level considerations, system accessibility for maintenance, etc.

- Political/Cultural Constraints: These are constraints which include policy and legal issues (e.g., what laws and standards apply to the product).

Types of Nonfunctional Requirement



Domain Requirements

- Domain requirements are derived from the application domain.
- These types of requirements may consist of new functional requirements or constraints on existing functional requirements, or they may specify how particular computations must be performed.

Domain Requirements

- In the baggage handling system, for example, various domain realities create requirements.
- There are industry standards.
- There are constraints imposed by existing hardware available.
- And there may be constraints on performance mandated by collective bargaining agreements with the baggage handlers union.

Domain Requirements

For the pet store POS system, domain requirements are imposed by the conventional store practices. For example:

- Handling of cash, credit cards, and coupons
- Conventions in the pet store industry (e.g., frequent-buyer incentives, buy one get one free)
- Sale of items

Domain Vocabulary Understanding

- The requirements engineer must be sure to fully understand the application domain vocabulary as there can be subtle and profound differences in the use of terms in different domains.

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.

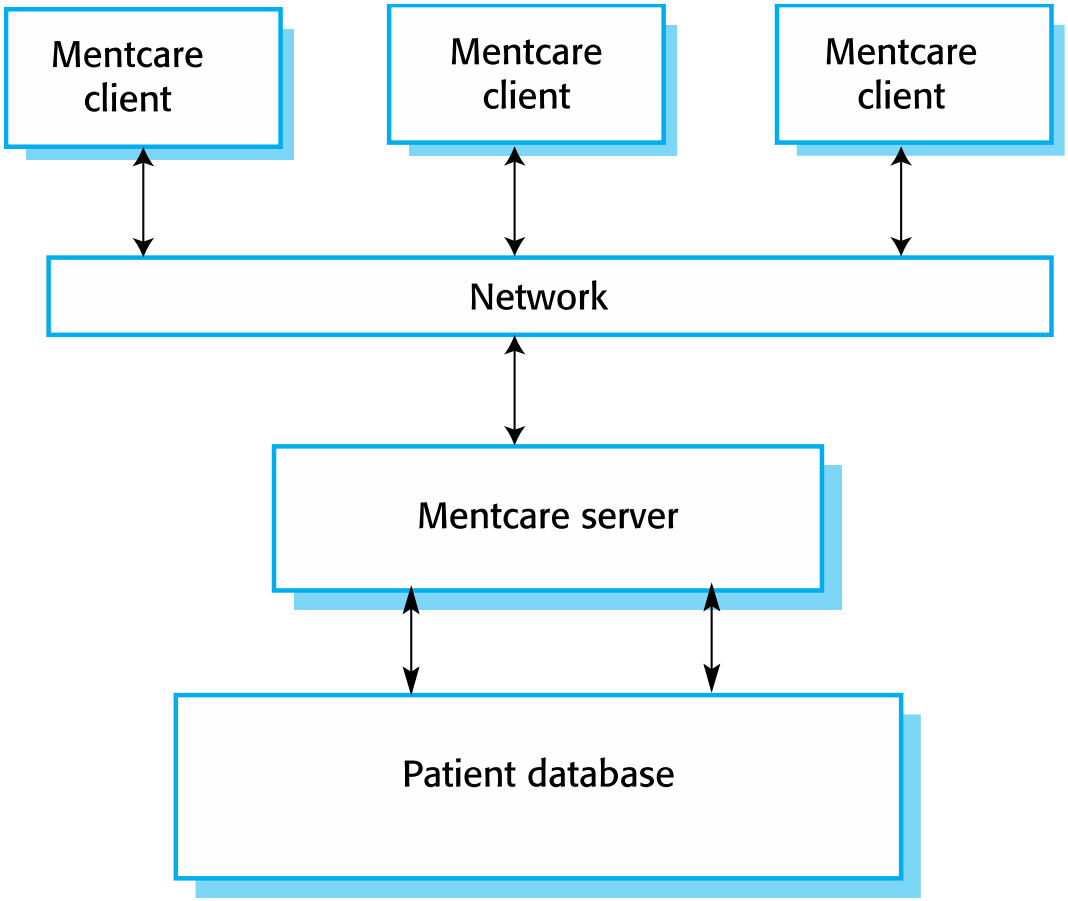
Mentcare

- 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 information to support the treatment of patients.

The organization of the Mentcare system



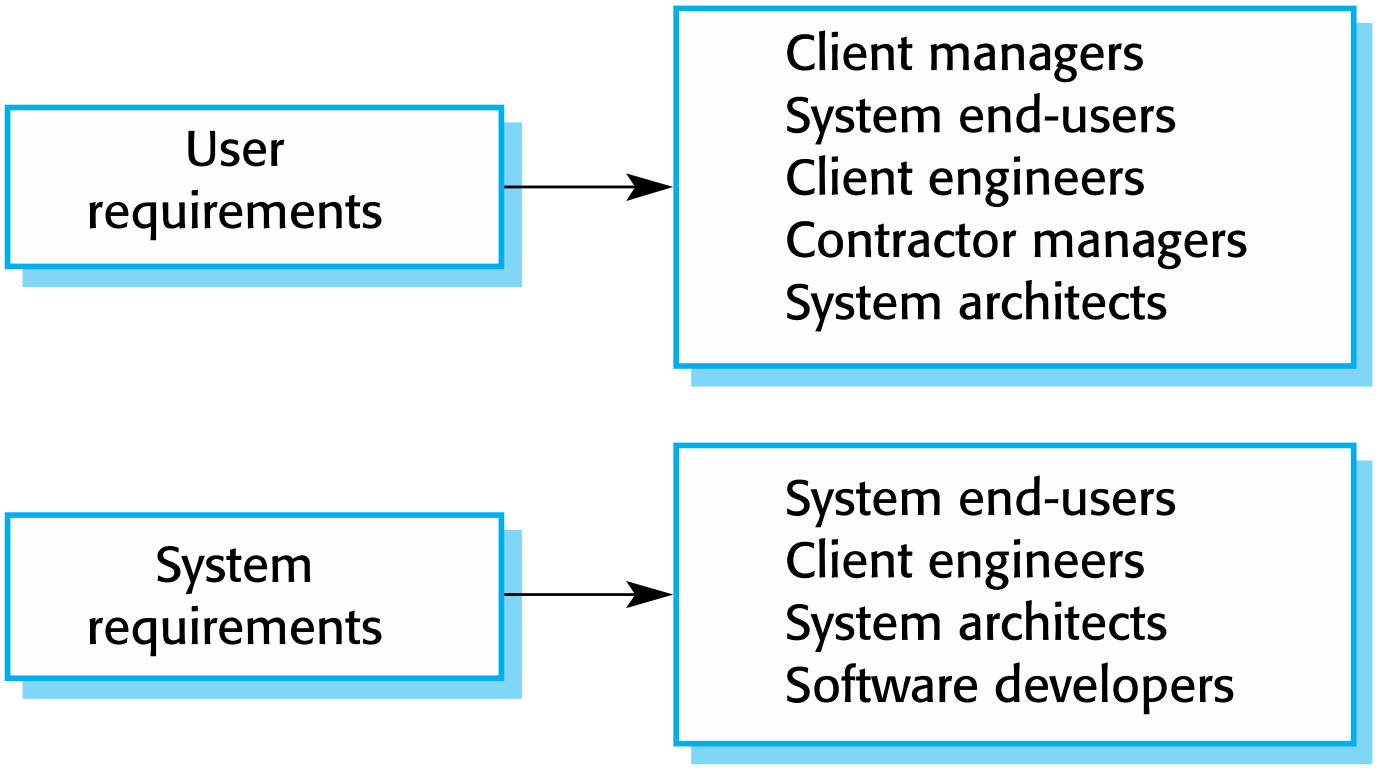
Key features of the Mentcare system

- 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.

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.

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.

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. Use 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.