

BBM-382 – SOFTWARE ENGINEERING

SPRING 2021

Lecture 3

Assoc. Prof. Dr. Ayça TARHAN

Dr. Tuğba ERDOĞAN

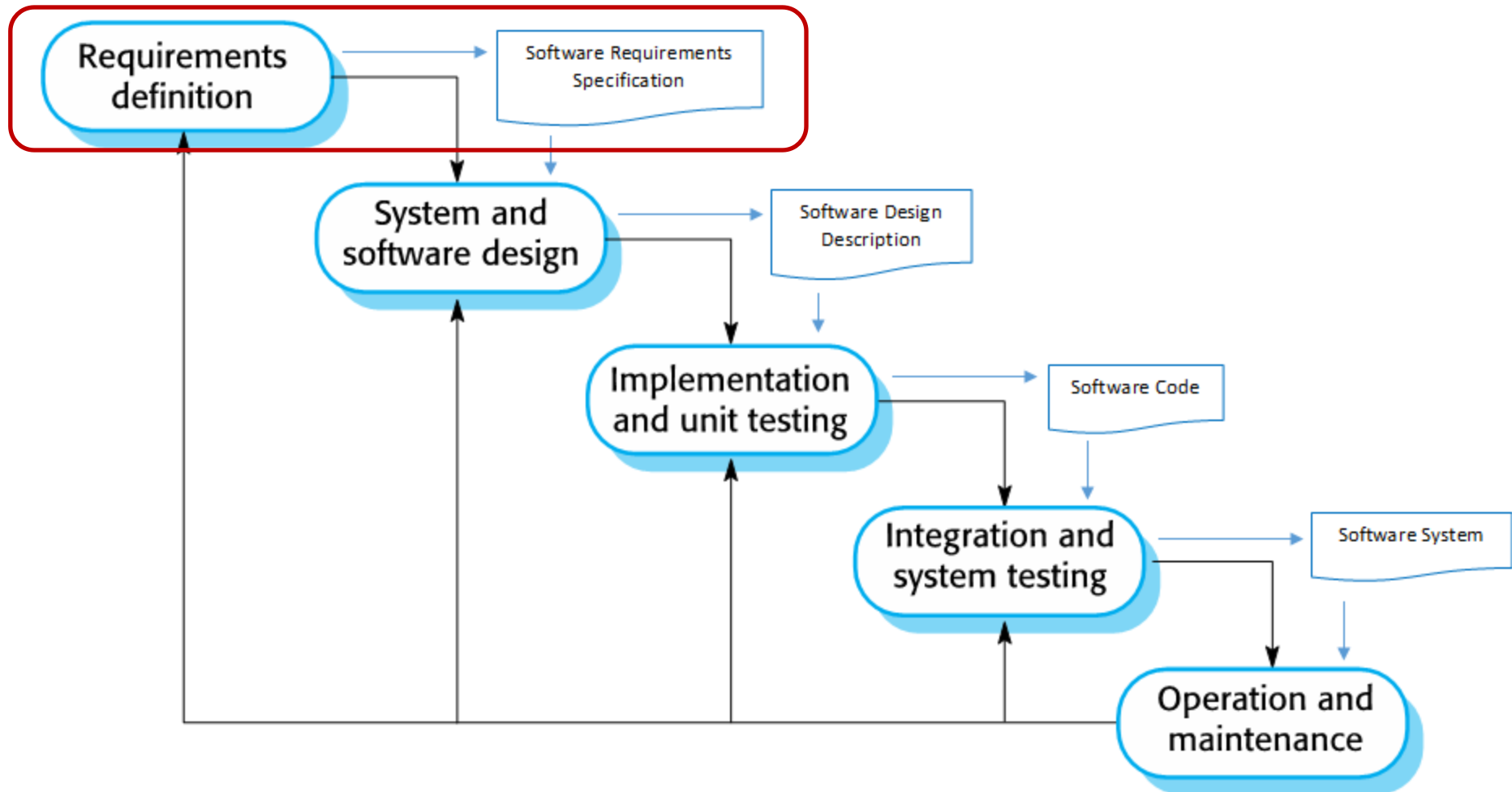




CHAPTER 4 – REQUIREMENTS ENGINEERING

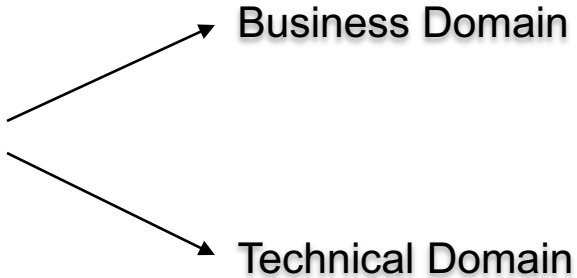


Waterfall Development Model - Revisited





Challenges - Revisited

- Characteristics of software:
 - Intangible (abstract)
 - Changeable
- Characteristics of software development:
 - Human-intensive
 - Multi-disciplinary

```
graph LR; A[Multi-disciplinary] --> B[Business Domain]; A --> C[Technical Domain]
```



Business Domains and Requirements



Education



Tourism



Development Team:
WHAT TO DEVELOP?



Shopping

- 1) What are customer needs?
2) **How will software behave**
to satisfy these needs?



Requirements

- Requirements are the most critical aspect of the software development cycle.
 - Studies point to a **more than 60% failure rate** for software projects in the U.S., with poor requirements as one of the top five reasons.
 - Studies also show a high percentage of project schedule overruns, with **80% due to creeping requirements**.
- Managing requirements must be an integral part of an organization's overall development process, and is vital to mitigating risk of cost overruns and delays on large development projects.





Requirements engineering

- *Requirements Engineering* is:
 - the process of establishing the services that the customer requires from a system, and
 - the constraints under which it operates and is developed.

- The *requirements* themselves are:
 - the descriptions of the system services and constraints that are generated during the requirements engineering process.

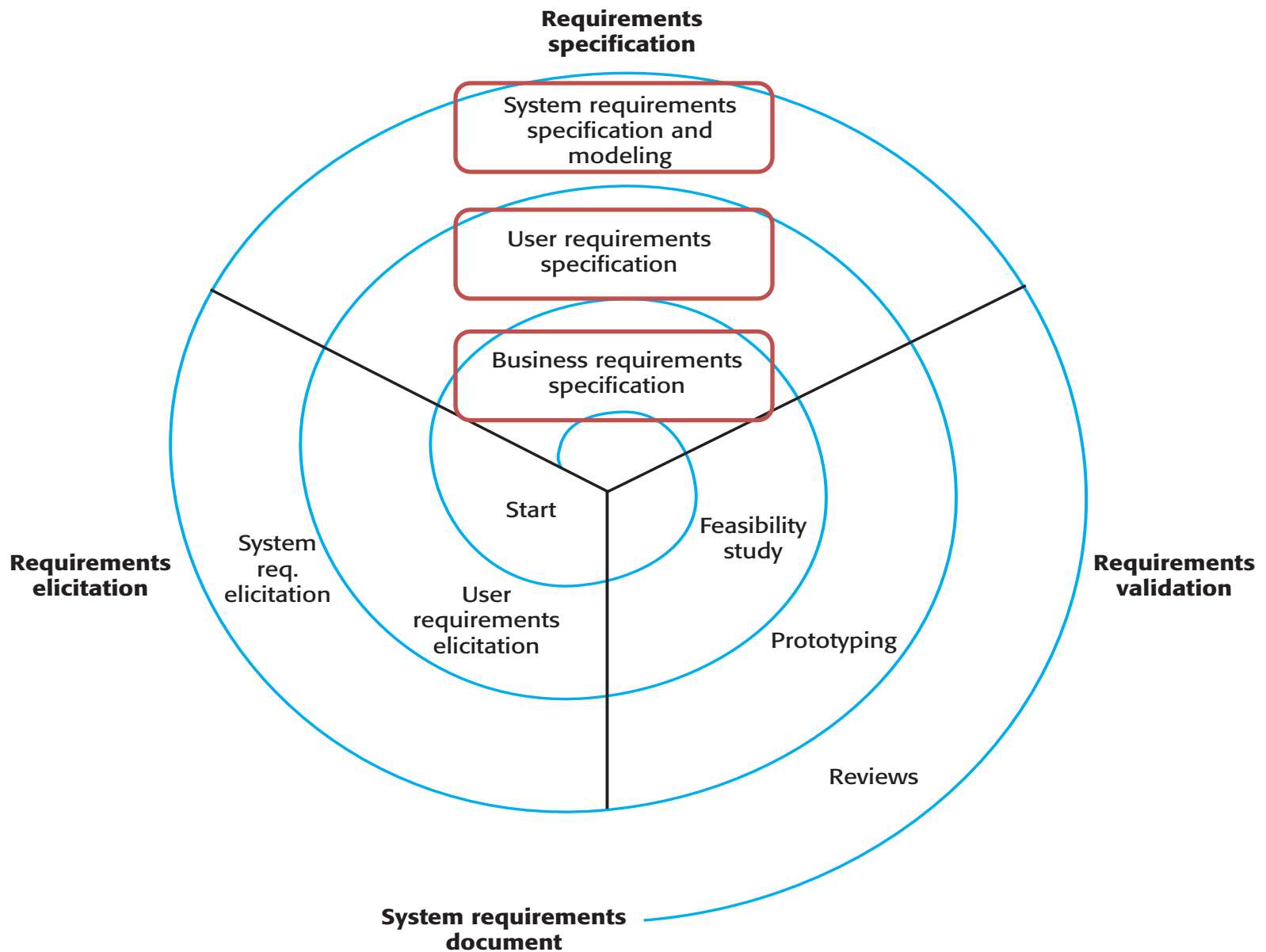


What is a requirement?

- It may range from:
 - **a high-level abstract statement** of a service or of a system constraint to:
 - **a detailed specification**, e.g., of a mathematical function
- This is inevitable as requirements may serve a *dual function*
 - the basis for a bid for a contract - therefore must be open to interpretation;
 - contract itself - therefore must be defined in detail;

Both statements may be called requirements.

A spiral view of the requirements engineering process





Stages (Levels) of requirements

■ 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 and 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.
- Written for systems and software developers.

- Typically, a user requirement is detailed into more than one system requirements.



Example: Flight Booking System (FBS)

- Scope of Development:
 - A Flight Booking System (FBS) that serves different functionalities for reservation agents and reservation managers shall be designed and developed.

The screenshot shows a window titled "THY Ticketing System" with a close button (X) in the top right corner. The window contains the Turkish Airlines logo (a red stylized bird) and the text "TURKISH AIRLINES" and "TÜRK HAVA YOLLARI". Below the logo, there are two input fields: "Username:" with the text "vgarousi" and "Password:". To the right of the "Username:" field is a "Login" button. To the right of the "Password:" field is a "Forgot Password" button. Below the "Username:" and "Password:" fields is a "Search for Flights" button. At the bottom of the window is an "Exit App" button.



Example: FBS – User Requirements

■ User requirement

- (1) A reservation manager shall be able to manipulate flights (enter new ones, update existing flight information, delete flights, etc.), and generate inventory reports of flights.

■ System requirements

- (1.1) A reservation manager should be able to manipulate flights (enter new ones, update existing flight information, delete flights, etc.), and generate inventory reports of flights. The inventory report should contain a summary of all flights in the system that still have unsold seats (either economy or economy), and the total number of unsold seats of business class and economy class in all flights.
- (1.2) Flight information (to be manipulated by reservation managers) should include the following: airline code (e.g., AC), flight number (e.g., 321), departure airport, departure terminal number, departure date and time, arrival airport, arrival terminal number, arrival date and time, cost of business class and cost of economy class ticket.
- (1.3) The flight number is a 3-digit number, prefixed with 0's if less the actual number is less than 100.
- (1.4) The airline code must be a two-letter code defined in http://en.wikipedia.org/wiki/IATA_airline_designator .
- (1.5) The airport location must be one of airport with a three-letter code defined in http://en.wikipedia.org/wiki/List_of_airports_by_IATA_code.



Types of Requirements

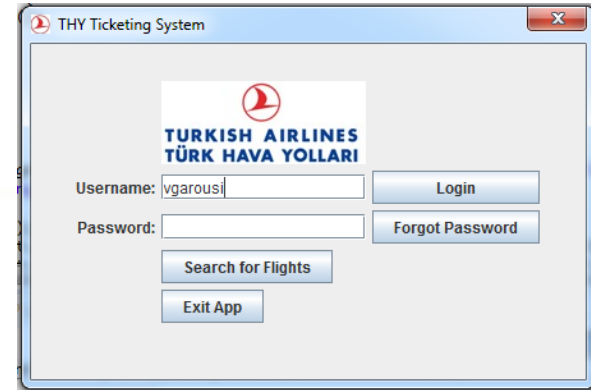
■ Functional requirements

- 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 also 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.

Example: FBS – Functional and Non-functional requirements



■ Functional requirement:

- A reservation manager shall be able to log into the system with his/her username and password.

Software
Behavior

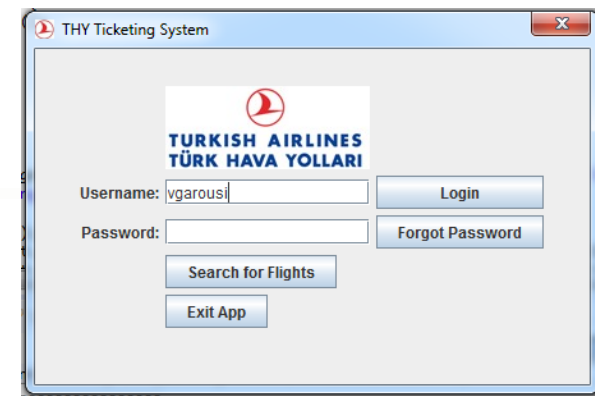
■ Non-functional requirements:

- The system shall enable log in of its users within 5 sec
 - *Performance requirement*
- The system shall enable secure log in of its users.
 - *Security requirement*

Constraints
on
Software
Behavior

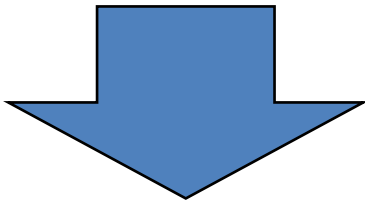


Example: FBS – Usability (non-functional) requirement



■ User need:

- The system shall be easy to use by flight booking staff and shall be organized in such a way that user errors are minimized.



■ Verifiable non-functional requirement:

- Flight booking staff shall be able to use all the system functions after six hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use.



In-class exercise



- Write performance non-functional requirements for the FBS under these items:
 - User need
 - Verifiable non-functional requirement

- Time: 5 minutes

The screenshot shows a window titled "THY Ticketing System" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, the Turkish Airlines logo is displayed at the top, followed by the text "TURKISH AIRLINES" and "TÜRK HAVA YOLLARI". Below the logo, there are two input fields: "Username:" with the text "vgarousi" entered, and "Password:". To the right of the "Username:" field is a "Login" button. To the right of the "Password:" field is a "Forgot Password" button. Below these fields, there is a "Search for Flights" button and an "Exit App" button.



Types of Requirements (cont'd)

■ Domain requirements

- Constraints on the system from its operational domain
- If not satisfied, the system may be unworkable
 - Requirements are expressed in the language of the application domain, and this is often not understood by engineers developing the system.
 - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

■ Process requirements

- Requirements that are specified mandating a particular IDE, programming language, development method or standard.



Example: FBS – Domain and Process requirements

- Domain requirement:
 - When there is a natural disaster (e.g. strong storm or tornado) on a specific location and date as reported by Weather Monitoring System, all flights that depart from/arrive to that location on that date will be cancelled.

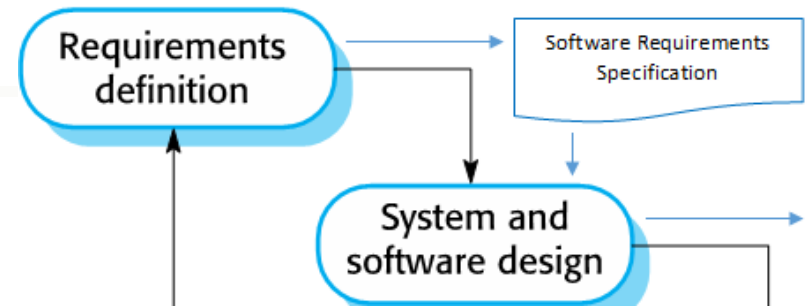
- Process requirements:
 - The system shall be developed by following Waterfall Development Model.
 - The system shall be a simple client-server, web-based application.
 - The system shall be developed with Java.



Specification of Requirements

- Requirements should be :
 - Unambiguous:
 - They should not be interpreted in different ways by developers and users.
 - Complete:
 - They should include descriptions of all facilities required.
 - Consistent:
 - There should be no conflicts or contradictions in the descriptions of the system facilities.

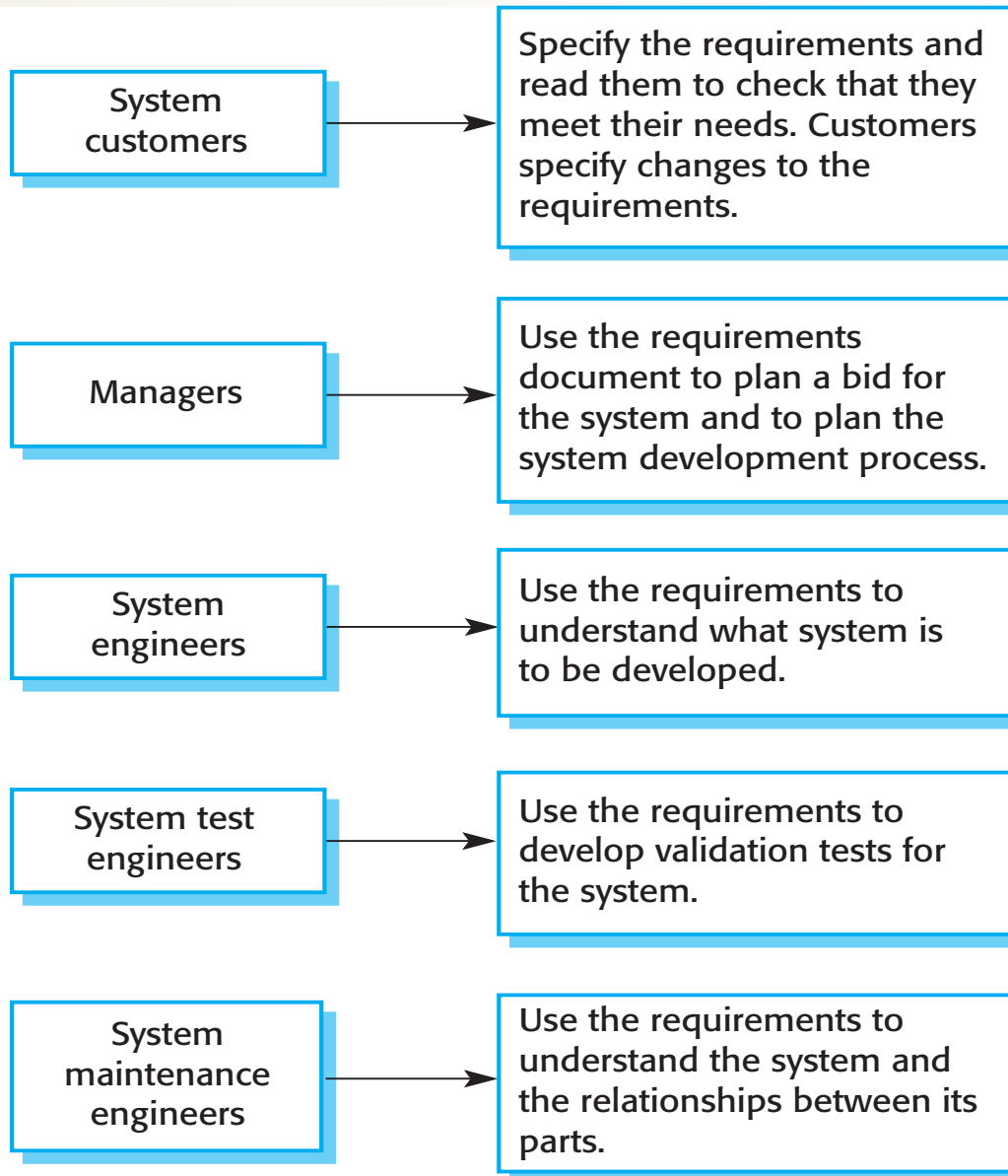
The software requirements document



- The *software 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





Requirements document variability

- Information in requirements document depends on type of system and the approach to development used, e.g.
 - A systems developed by Waterfall Model are required to have all details in its requirements document.
 - A system developed incrementally (e.g. by iterations) will typically have less detail in its requirements document.
- Requirements documents standards have been designed, e.g. **IEEE standard 830**.
 - These are mostly applicable to the requirements for large systems engineering projects.



Requirements engineering (RE) process

- The processes used for RE vary widely depending on the application domain, the people involved and the organization developing the requirements.
- However, there are a number of generic activities common to all processes:
 1. **Requirements elicitation and analysis;**
 2. **Requirements specification;**
 3. **Requirements validation;**
 4. **Requirements management.**
- In practice, RE is an iterative activity in which these processes are interleaved.




In-class exercise



- Discuss with your friend and write down how you will conduct these for the FBS:
 - Requirements elicitation and analysis
 - Requirements specification
 - Requirements validation
 - Requirements management

- Time: 5 minutes

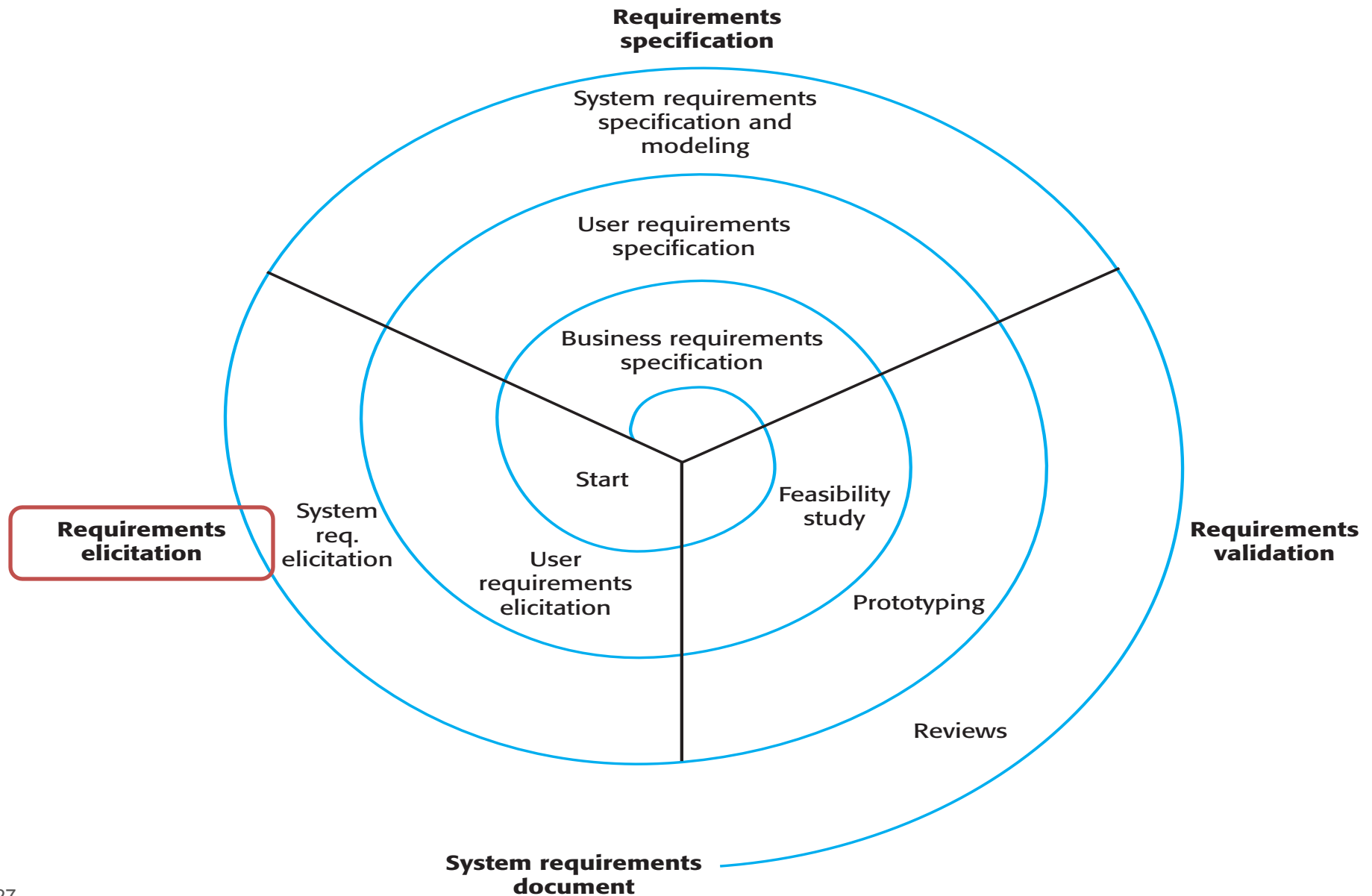
THY Ticketing System


TURKISH AIRLINES
TÜRK HAVA YOLLARI

Username:

Password:

A spiral view of the requirements engineering process - revisited





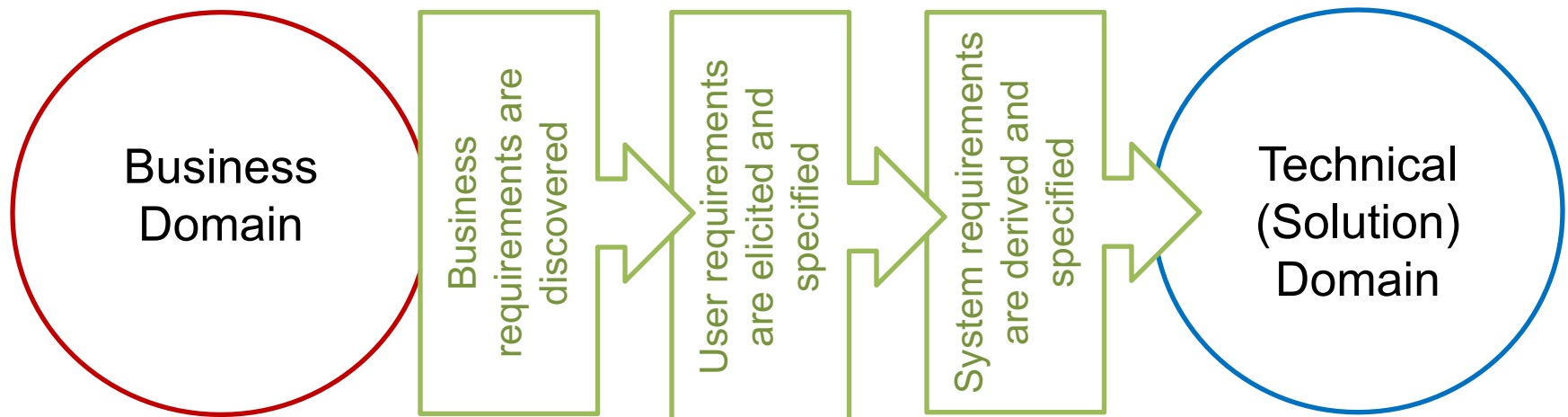
1) Requirements elicitation and analysis

- Technical staff works 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, domain experts, engineers, trade unions, etc.
 - These are called *stakeholders*.
- Starts with *requirement discovery*



Requirements discovery

- It is the process of:
 - Gathering information about the required and existing systems, and
 - Distilling the user and system requirements from this information.
- Requires interaction with system stakeholders from managers to external regulators.





Problems of requirements elicitation and 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 may change.



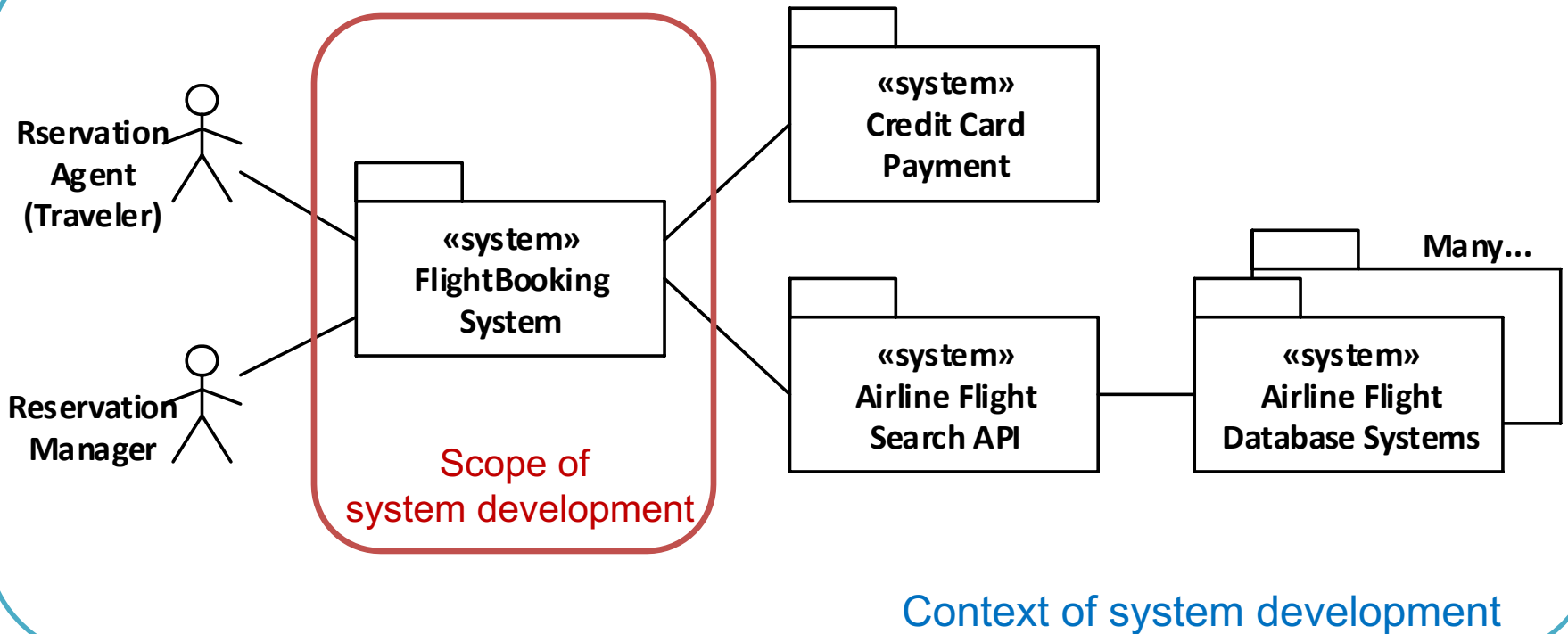
System context

- From an early stage of system specification, we should decide on the boundaries of the system under development, e.g.:
 - What functionality will be included in the system,
 - What is provided (e.g. inputs) by the system's environment,
 - Manual versus automated functionality of business processes.
- We can use a *context model* to represent this information (see Chapter 5).

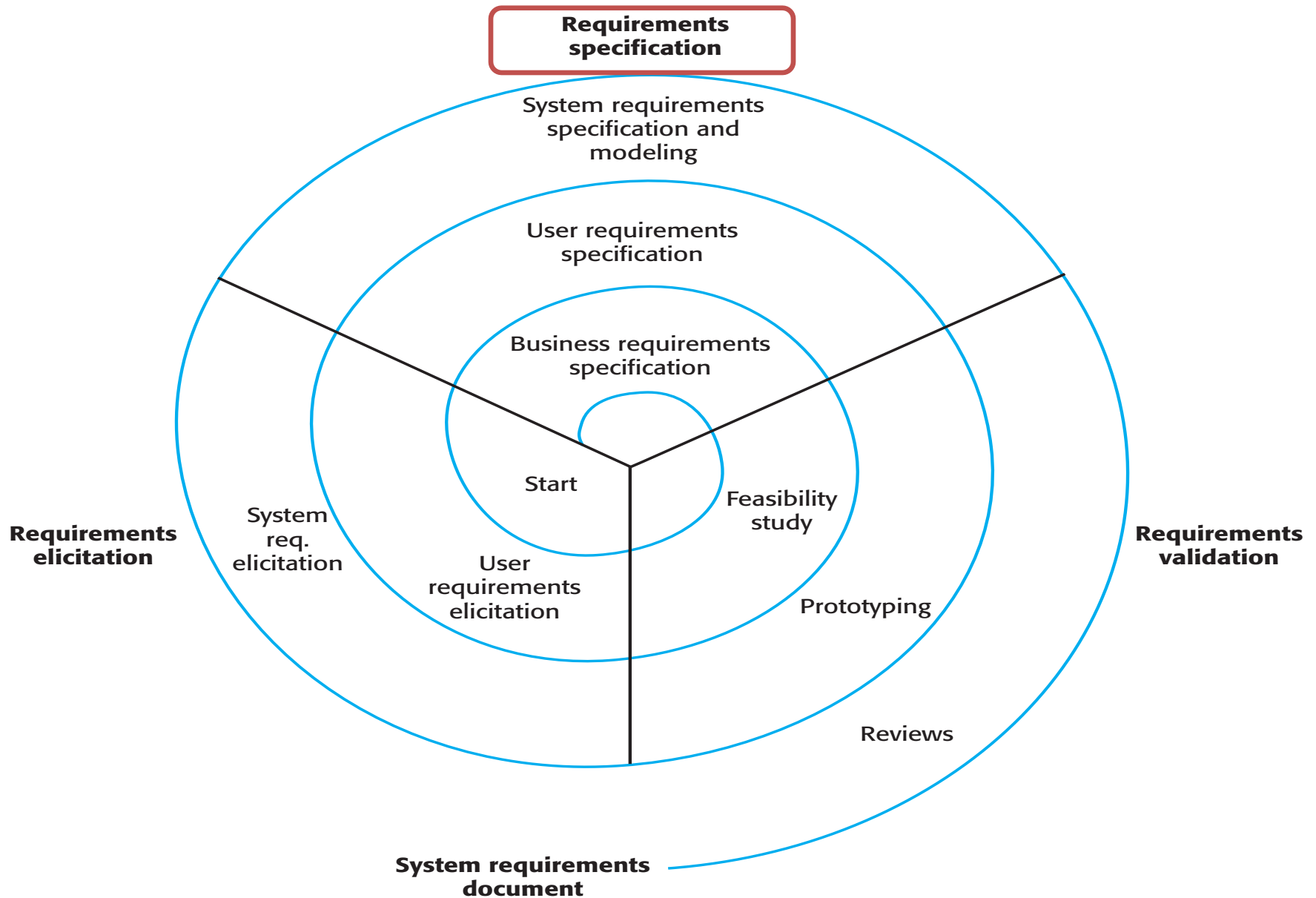




Example: FBS – Context model



A spiral view of the requirements engineering process





2) Requirements specification

- *Requirements specification* is the process of writing the user and system requirements in a *requirements document*.
 - User requirements have to be understandable by end-users and customers who do not have a technical background.
 - System requirements are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
 - It is therefore important that these are as complete as possible.



Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. <u>Each sentence should express one requirement.</u>
Structured natural language	The requirements are written in natural language on a standard form or template. <u>Each field provides information about an aspect of requirement.</u>
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify requirements by defining an operational model of system. This approach is now <u>rarely used</u> although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define functional requirements for system; <u>UML use case and sequence diagrams</u> are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these <u>unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification.</u> They cannot check that it represents what they want and are reluctant to accept it as a system contract



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, an **may** for optional requirements.
- Use text highlighting to identify key parts of the requirement.
- Avoid the use of computer jargon.
- Include an explanation (rationale) of why a requirement is necessary.



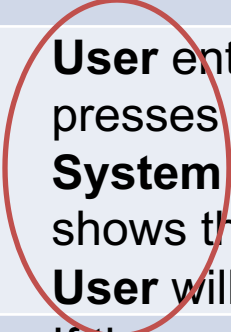
Example: FBS requirements – Natural Language

1. A reservation manager *shall* be able to use the system to create accounts for reservation agents.
2. The system *shall* enable a reservation manager to log in with his/her username and password.
3. The reservation manager *shall* be able to manipulate flights (enter new ones, update existing flight information, delete flights, etc.), and generate inventory reports of flights.
4. When a flight is canceled by the reservation manager, all itineraries that are reserved or booked and include that flight *shall* be canceled as well.
5. The reservation manager *shall* be able to view information of a reservation agent and confirm a ticket for him/her.
6. The reservation manager *shall* be able to log out at any stage during his/her session.



Example: FBS requirements – Structured Natural Language (e.g. Tabular structure)

Use case: Login	
Actors	Traveler
Precondition	Traveler is not logged in
Post-condition	Traveler is logged in
Main (happy) path:	<ol style="list-style-type: none">1. User enters her/his username and password and presses on the Login button2. System checks the username and password and shows the app's main window (page)3. User will be able to use the feature afterwards
Alternative path:	<ol style="list-style-type: none">1. If the username and password combination is invalid, the system will show a message indicating it2. User will be able to enter another username and password

- 
- *Interaction between user(s) and the software system to be developed!*
 - *Only outside behavior of the systems as observed by the user(s)!*




In-class exercise



- Write the structured tabular requirements for the feature “Search for Flights”
- Time: 5 minutes

THY Ticketing System


TURKISH AIRLINES
TÜRK HAVA YOLLARI

Username: vgarousi

Password:

Use case: Login

Actors	Traveler
Precondition	Traveler is not logged in
Post-condition	Traveler is logged in
Main (happy) path:	<ol style="list-style-type: none">1. User enters her/his username and password and presses on the Login button2. System checks the username and password and shows the app's main window (page)3. User will be able to use the feature afterwards
Alternative path:	<ol style="list-style-type: none">1. If the username and password combination is invalid, the system will show a message indicating it2. User will be able to enter another username and password



Scenarios (or user stories)

- *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.



Example: FBS Scenarios (in Natural Language)

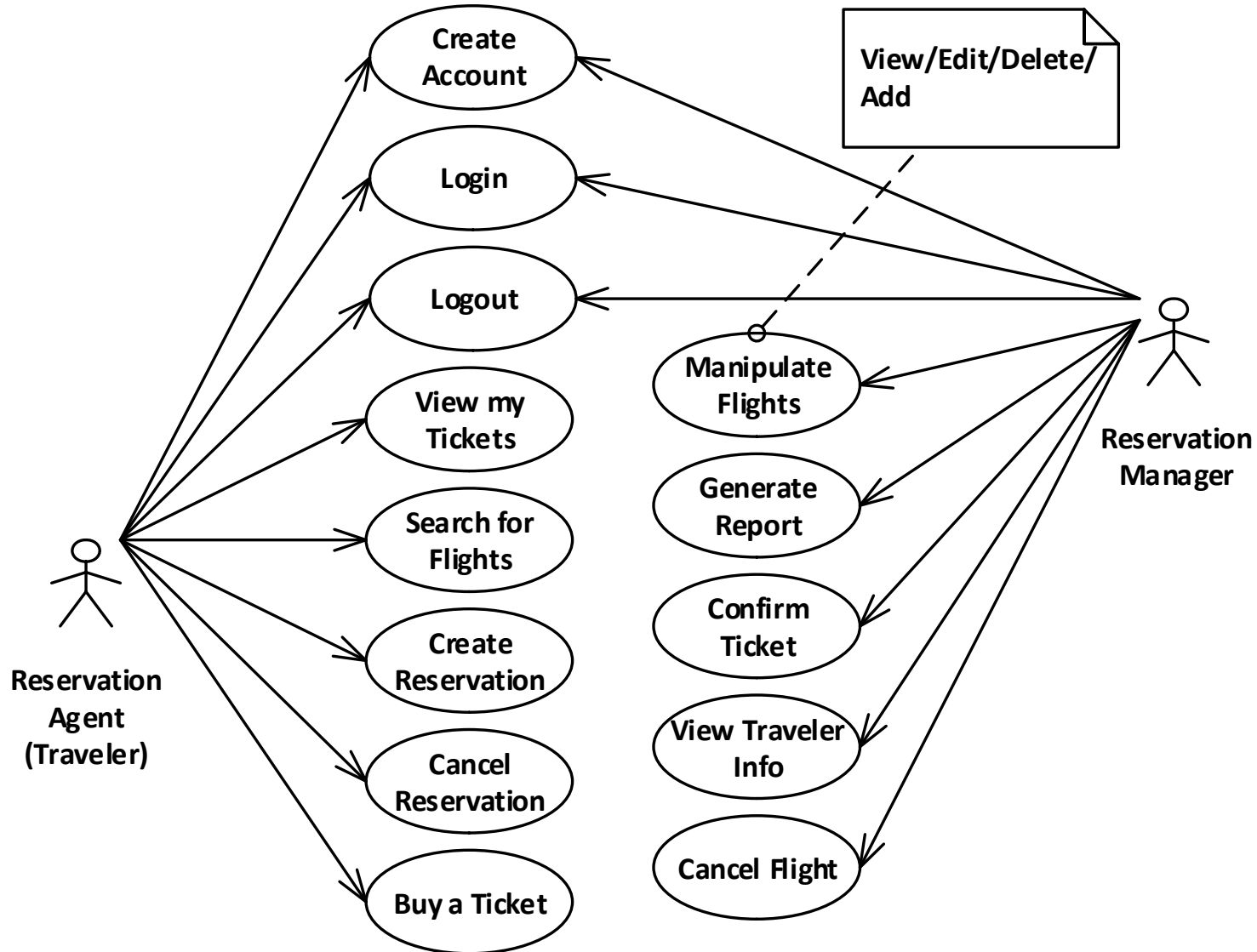
- Scenario: Search for flights & create reservation
 - When a reservation agent wants to create and book a travel itinerary, he/she first shall search for flight information. A list of available flight options shall be shown with departure/arrival and cost information. Once the agent selected an itinerary from the list, he/she shall have the option to reserve it.
- Scenario: Buy a ticket
 - Once the reservation agent reserved an itinerary, he/she shall have the option to book it by providing payment information via credit card. If the credit card information is not on-file for the agent, he/she shall be prompted to enter the credit card information. Once the credit card is validated, the reservation agent shall be shown with the actual ticket information.



Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. <u>Each sentence should express one requirement.</u>
Structured natural language	The requirements are written in natural language on a standard form or template. <u>Each field provides information about an aspect of requirement.</u>
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify requirements by defining an operational model of system. This approach is now <u>rarely used</u> although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define functional requirements for system; <u>UML use case and sequence diagrams</u> are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these <u>unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification.</u> They cannot check that it represents what they want and are reluctant to accept it as a system contract

Scope of FBS Requirements by Use-Case Diagram





Use cases

- *Use-cases* are a scenario based technique in UML (Unified Modeling Language), which identify actors in an interaction and describe externally observable interaction between the system and its actors.
- A set of use cases should describe all possible interactions with the system (e.g. **Use-case diagram**).
 - High-level graphical model supplemented by more detailed tabular description (see Chapter 5).
 - **Activity diagrams** or sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system (see Chapter 5).





Exercise for HOME



- System: student registration management system (like BILSIS2)
- Draw the use-case diagram

Not secure | www.oid.hacettepe.edu.tr/bilsis2/

**HACETTEPE ÜNİVERSİTESİ**
Öğrenci Bilgi Sistemi



Sisteme Giriş

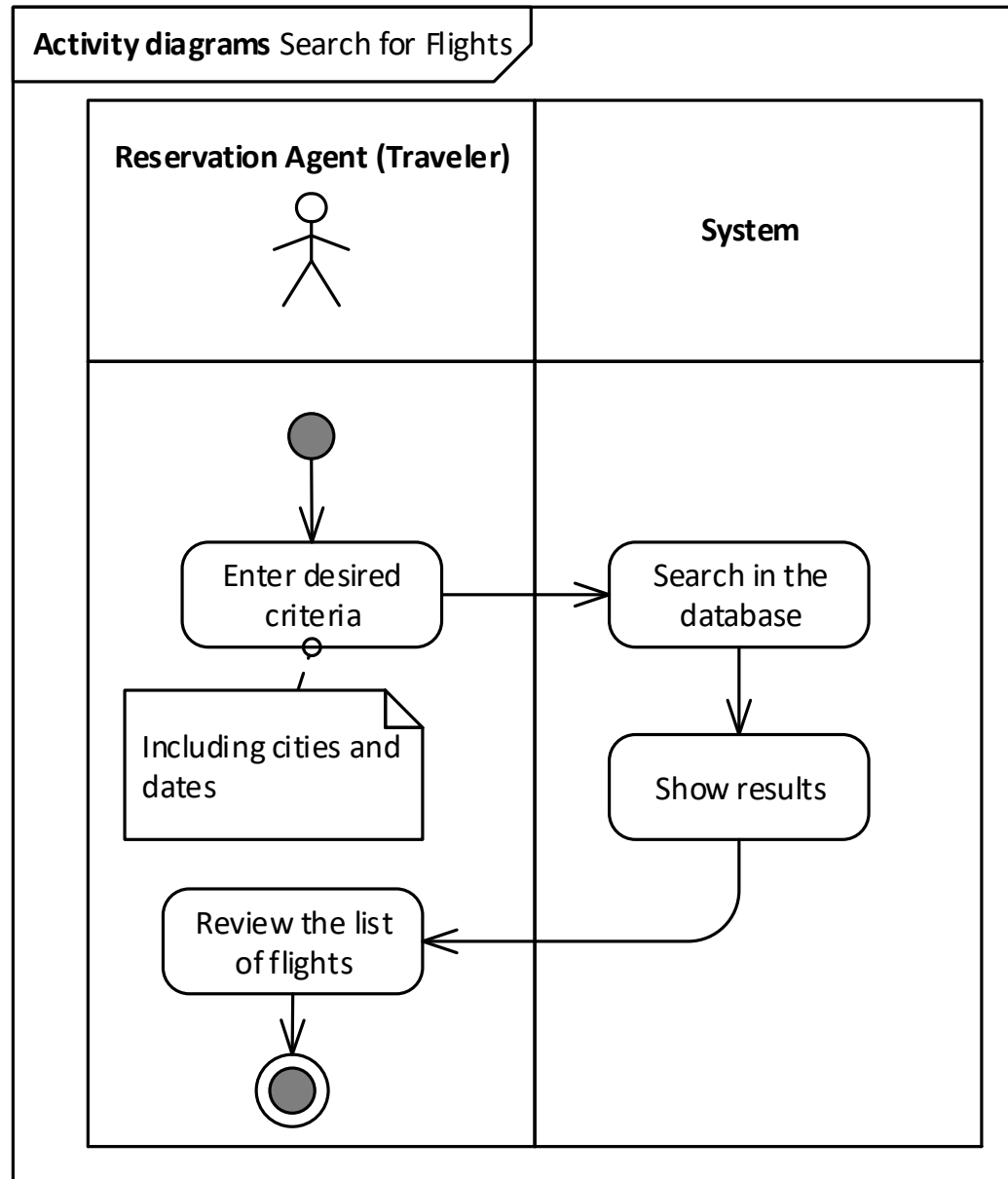
[Şifremi unuttum !!!](#)

Tamam

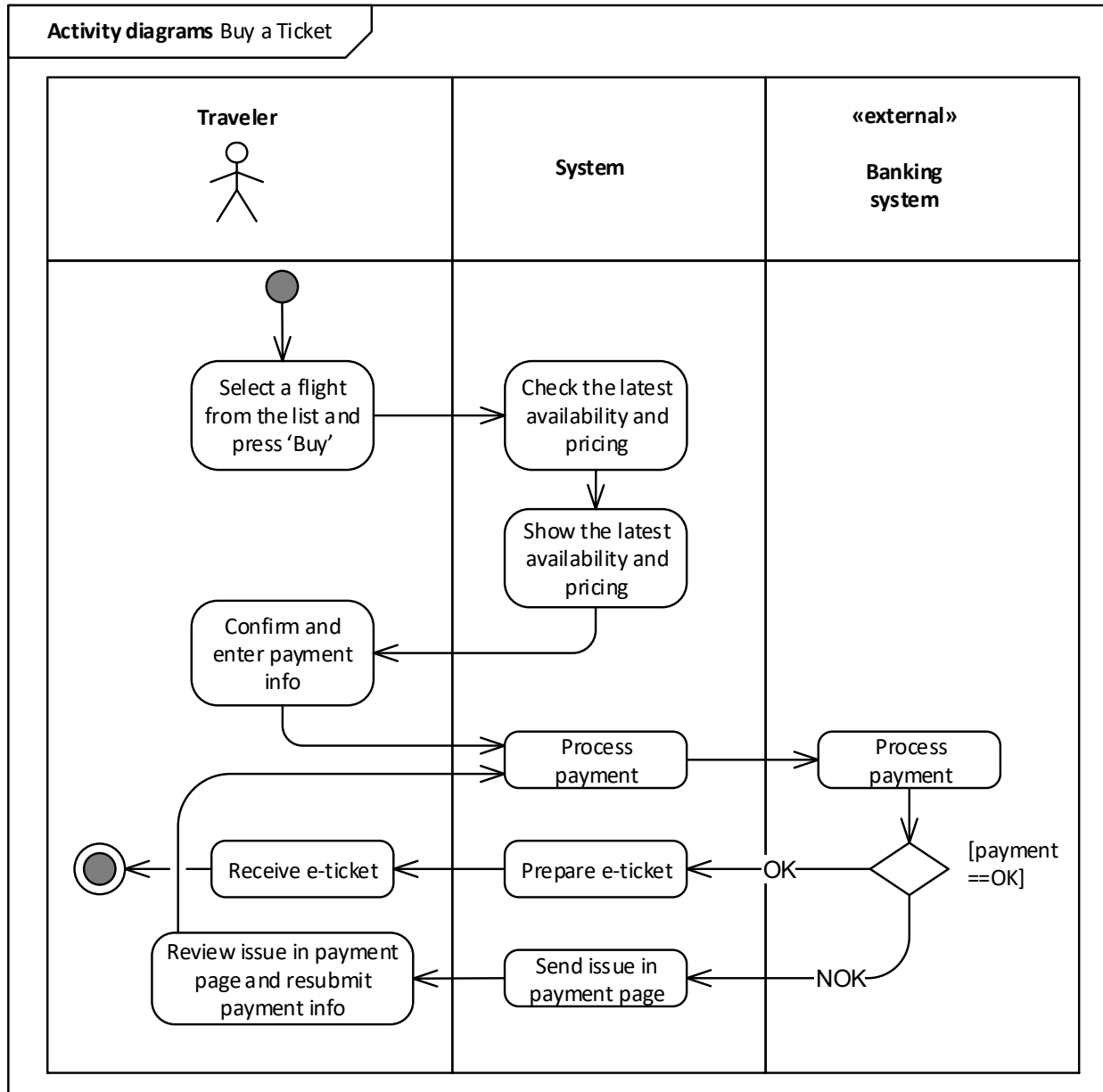
BilSis v.2.0

Hacettepe Üniversitesi
Öğrenci İşleri Daire Başkanlığı
Bilgi Sistemleri Müdürlüğü

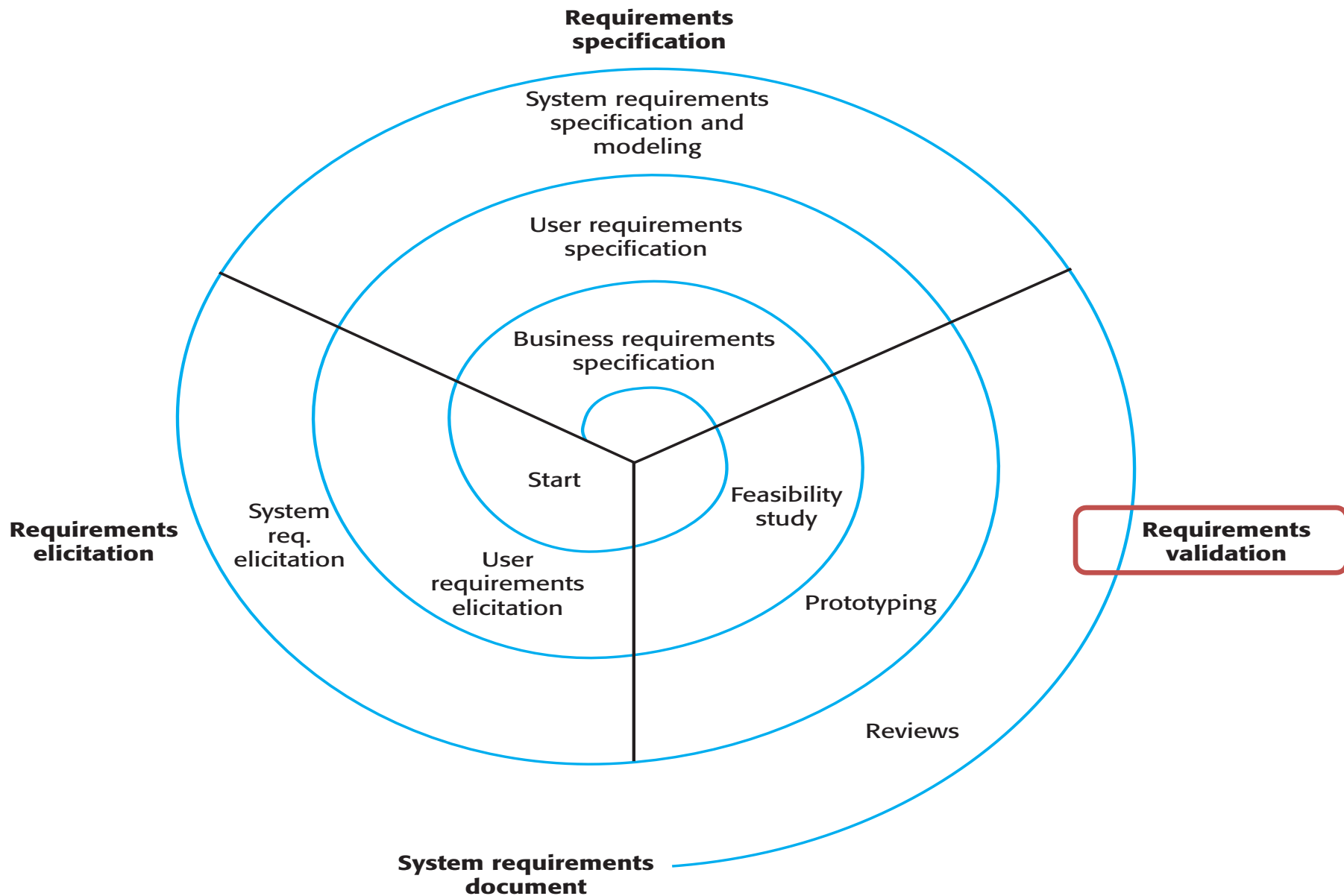
Example: FBS – Details of "Search for Flights" use-case description by Activity Diagram



Example: FBS – Details of "Buy a Ticket" use-case description by Activity Diagram



A spiral view of the requirements engineering process

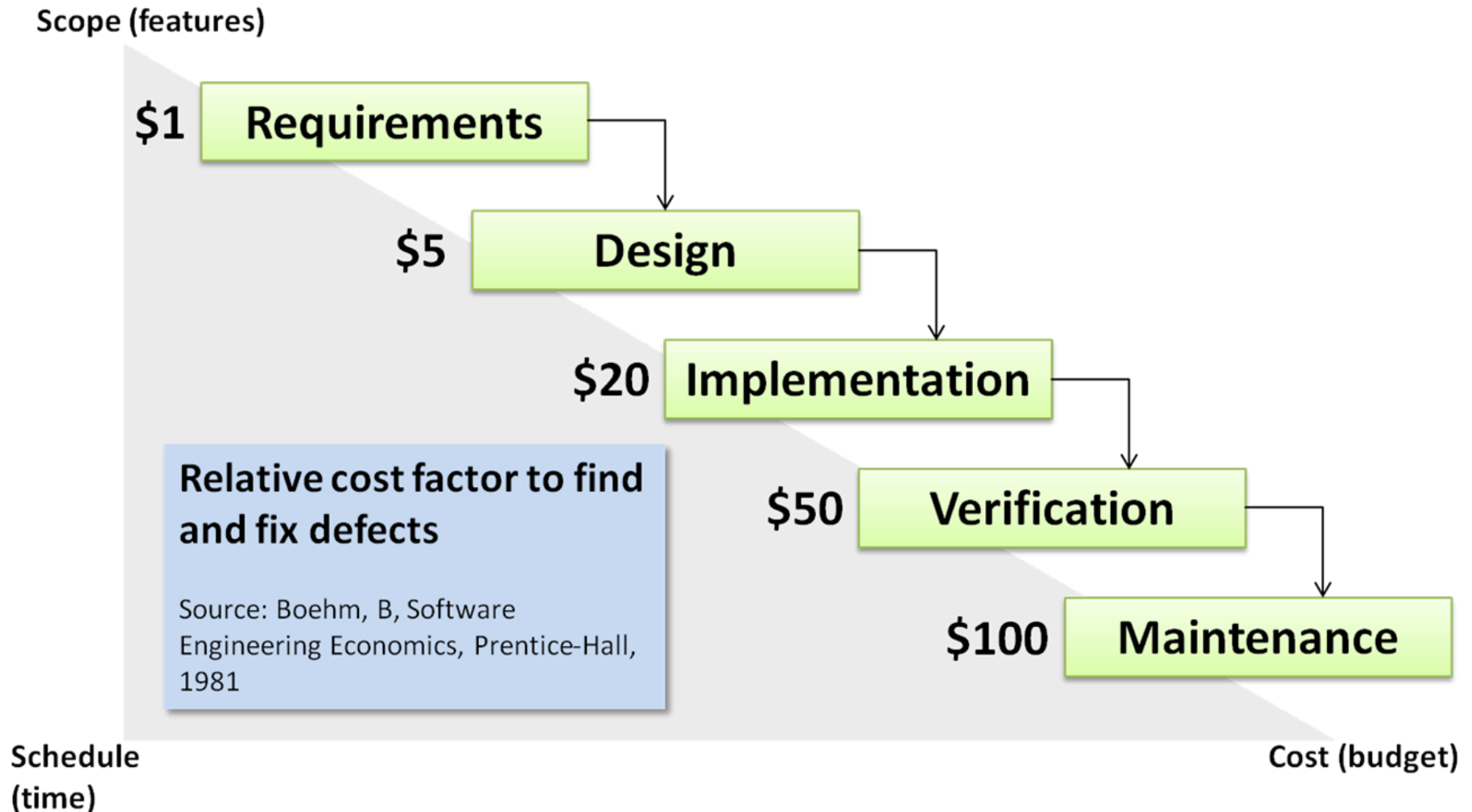




3) Requirements validation

- 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.

Relative cost factor to find and fix defects





Checklist for requirements validation

- **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 verified (e.g. tested)?



Requirements validation techniques

■ Requirements review

- Systematic manual analysis of the requirements.

■ Prototyping

- Using an executable model of the system to check requirements (see Chapter 2).

■ Test-case generation

- Developing tests for requirements to check testability.



Requirements review

- 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.



Checklist for requirements review

■ 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?

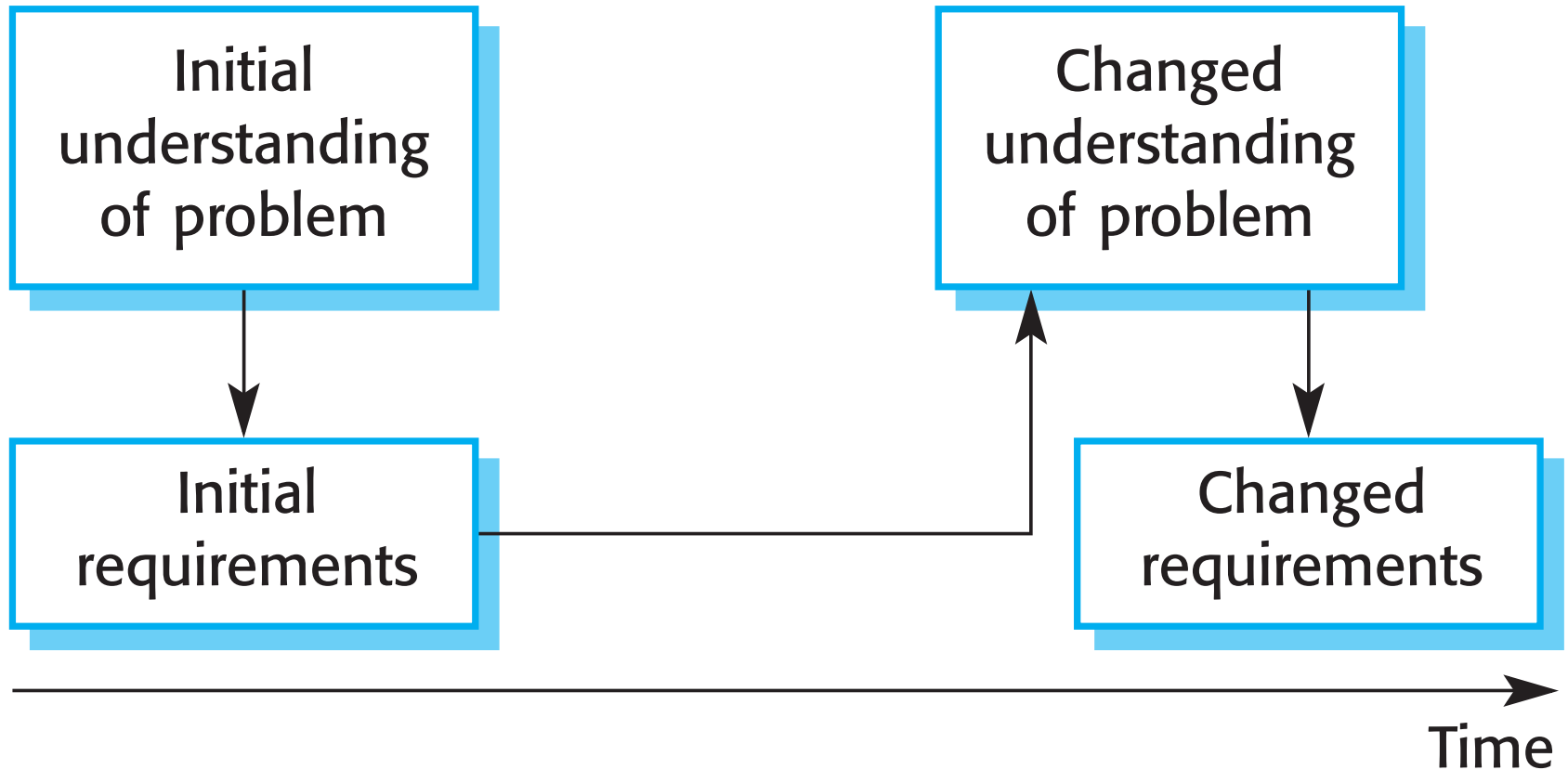


4) Requirements management

- Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- New requirements emerge as a system is being developed and after it has gone into use.
 - We keep track of individual requirements and maintain *links between dependent requirements* so that you can assess the impact of requirements changes.
 - We establish a *formal process* for making change proposals and linking these to system requirements.



Requirements evolution





Changing requirements

- **The business and technical environment of the system always changes after installation.**
 - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change and new legislation and regulations may be introduced that the system must necessarily abide by.
- **The people who pay for a system and the users of that system are rarely the same people.**
 - System customers impose requirements due to organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

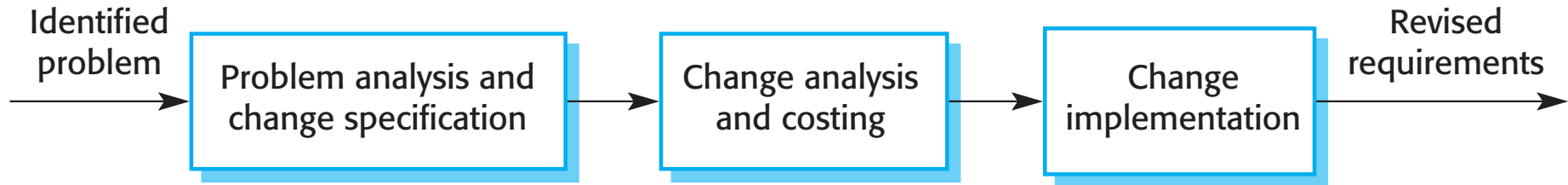


Changing requirements

- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
 - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.



Requirements change management





Requirements change management

■ *Problem analysis and change specification*

- During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to change requestor who may respond with a more specific requirements change proposal, or decide to withdraw the request.

■ *Change analysis and costing*

- The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.

■ *Change implementation*

- Requirements document and, where necessary, system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.



Traceability of Requirements to Requirements

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	

D: "dependent"
R: "relational"



Traceability of Requirements to Test Cases

Requirement Traceability Matrix												
	Test Case ID →	TC_1	TC_2	TC_3	TC_4	TC_5	TC_6	TC_7	TC_8	TC_9	TC_10	# Test Cases for respective Requirement ↓
Req. ID ↓												
Req_1		×		×			×					3
Req_2			×			×						2
Req_3				×								1
Req_4					×		×					2
Req_5						×		×				2
Req_6							×					1
Req_7						×		×				2
Req_8									×			1
Req_9										×		1
Req_10											×	1

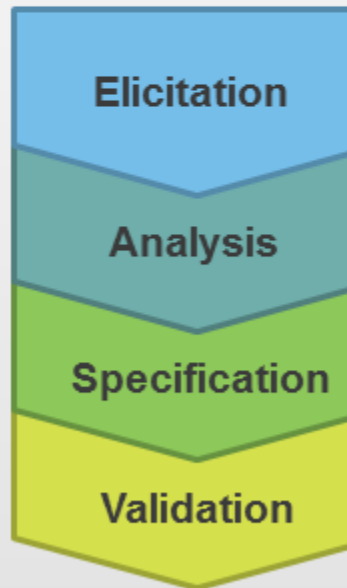
© TestingClub.blogspot.com



Requirements Engineering – Wrap up

Requirements Engineering

REQUIREMENTS DEVELOPMENT



Stakeholder consultation, review and context normalization



Understanding of desired system



Structured documentation of desired system



Identification of omitted, redundant & inconsistent req'ts

REQUIREMENTS MANAGEMENT

Traceability



Tracking where requirements are met

Change Mgmt



Req't maintenance & propagation

Fulfillment



Results & confirmation of successful fulfillment



FOR STUDENT READING...



Key points

- *Requirements engineering* process is an iterative process including requirements elicitation, specification and validation.
- Requirements elicitation and analysis is an iterative process that can be represented as a spiral of activities – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.
- You can use a range of techniques for requirements elicitation including interviews, scenarios, use-cases and ethnography.



Key points

- Requirements specification is the process of writing the user and system requirements in a requirements document.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Business, organizational and technical changes inevitably lead to changes to the requirements for a software system. Requirements management is the process of managing and controlling these changes.



Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
 - *Closed interviews* based on pre-determined list of questions
 - *Open interviews* where various issues are explored with stakeholders.
- Effective interviewing
 - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to 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.



User and system requirements

User requirement definition

1. The MHC-PMS 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.

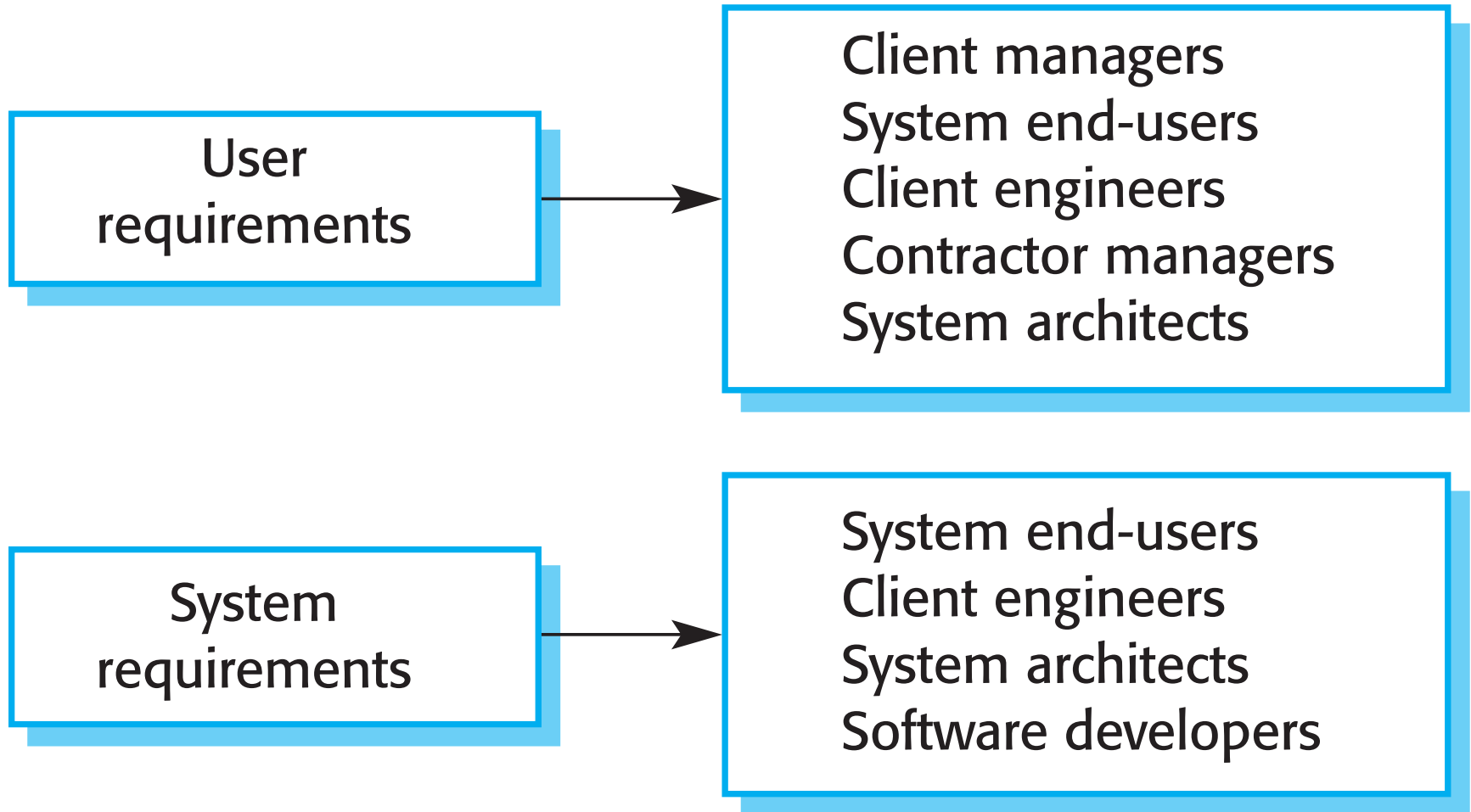
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

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, 20 mg, etc.) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Readers of different types of requirements specification





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.



Functional requirements for the MHC-PMS

- 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 requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Ex: Consider the term 'search' in the first requirement
 - 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, it is very difficult to produce a complete and consistent requirements document.

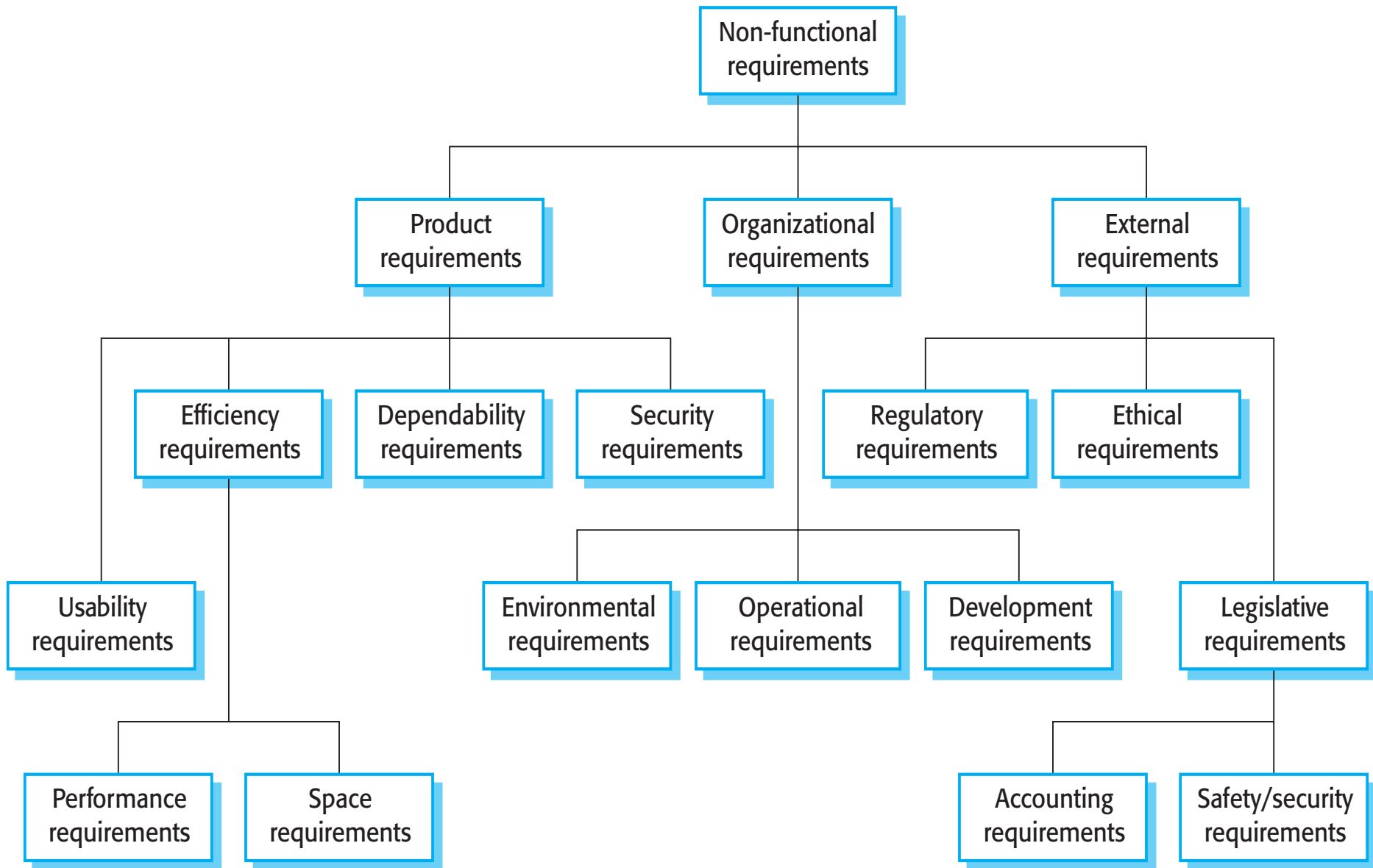


Non-functional requirements

- These define system properties and constraints, e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.



Types of nonfunctional requirements





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.

Examples of nonfunctional requirements in the MHC-PMS

Product requirement

The MHC-PMS 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.



Non-functional classifications

- Organisational requirements
 - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
 - Requirements that arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

Examples of nonfunctional requirements in the MHC-PMS

Organizational requirement

Users of the MHC-PMS 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.

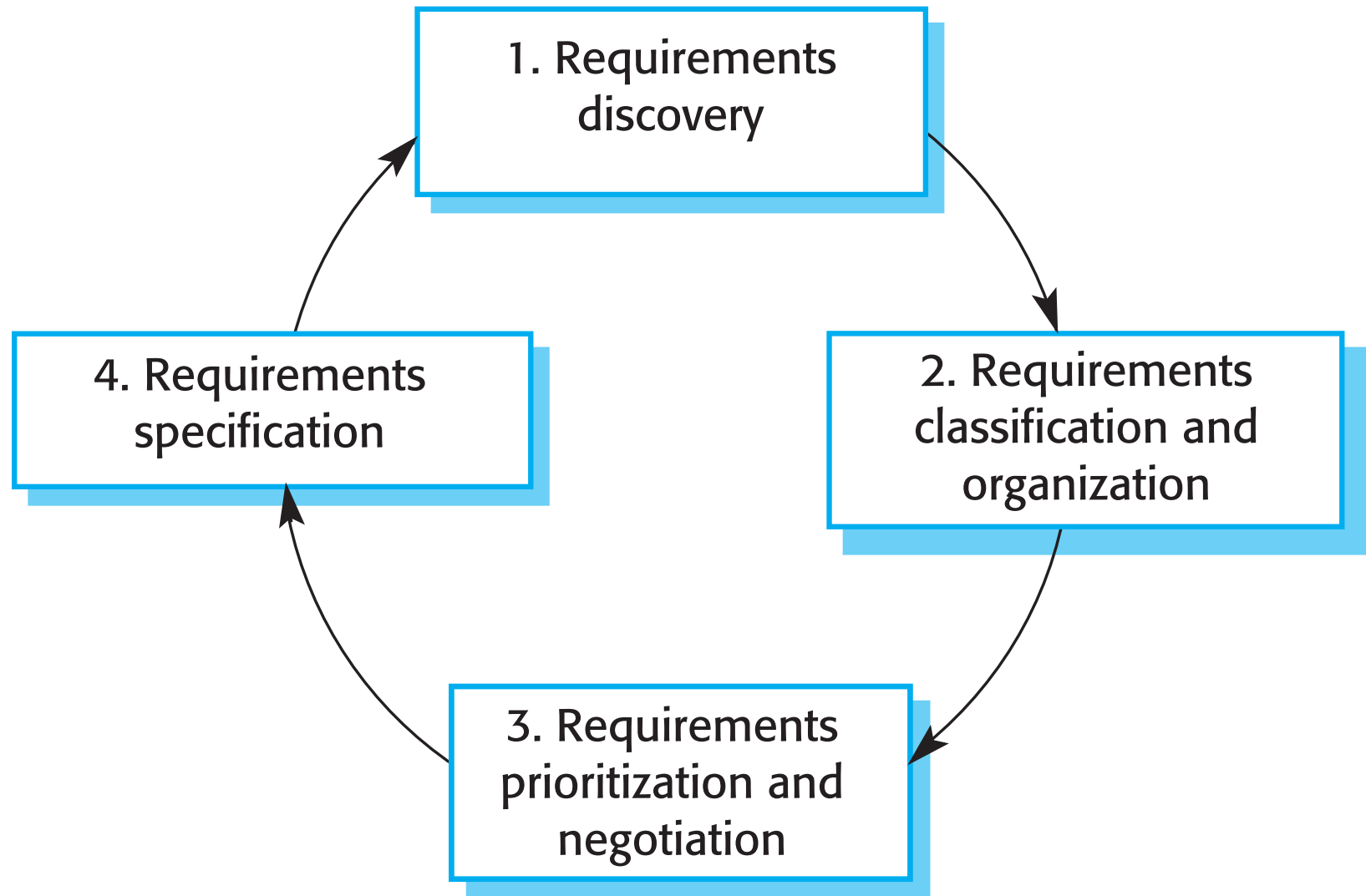


Metrics for specifying nonfunctional requirements

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



Requirements elicitation and analysis process





Requirements elicitation and analysis:

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 specification
 - Requirements are documented and input into the next round of the spiral.



Stakeholders in the MHC-PMS

- Patients whose information is recorded in the system.
- Doctors who are responsible for assessing and treating patients.
- Nurses who coordinate the consultations with doctors and administer some treatments.
- Medical receptionists who manage patients' appointments.
- IT staff who are responsible for installing and maintaining system.
- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- Health care managers who obtain management information from system.
- Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.



The structure of a requirements document

Chapter	Description
System requirements specification	This should describe <u>functional and nonfunctional</u> requirements in more detail. <u>Interfaces to other systems</u> may be defined.
System models	This might include <u>graphical system models</u> (object models, data-flow models, or semantic data models) showing relationships between system components and system and its environment.
System evolution	This should describe fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. (This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.)
Appendices	These should provide detailed, specific information that is related to application being developed; for example, hardware and database descriptions. <u>Hardware requirements</u> define minimal and optimal configurations for system. <u>Database requirements</u> define logical organization of data used by system and relationships between data.
Index	Several indexes to document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.



Ways of writing a system requirements specification

Notation	Description
Natural language	The requirements are written using numbered sentences in natural language. <u>Each sentence should express one requirement.</u>
Structured natural language	The requirements are written in natural language on a standard form or template. <u>Each field provides information about an aspect of requirement.</u>
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify requirements by defining an operational model of system. This approach is now <u>rarely used</u> although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define functional requirements for system; <u>UML use case and sequence diagrams</u> are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these <u>unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification.</u> They cannot check that it represents what they want and are reluctant to accept it as a system contract



A) Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
 - It is expressive, intuitive and universal.
 - Requirements can be understood by users and customers.



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.



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.
- Include an explanation (rationale) of why a requirement is necessary.



Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*



B) Structured specifications

- An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.



B-1) 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.
- Information about the information needed for the computation and other entities used.
- Description of the action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.



A structured specification of a requirement for an insulin pump

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.
Requirements	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.



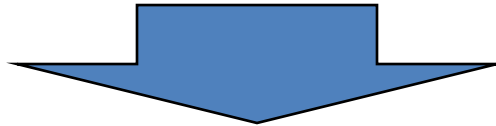
B-2) Tabular specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

Tabular specification of computation for an insulin pump

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.



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



Scenario for collecting medical history in MHC-PMS

Initial assumption: The *patient* has seen a *medical receptionist* who has created a record in system and collected patient's personal information (name, address, age, etc.). A *nurse* is logged on to system and is collecting medical history.

Normal: *Nurse* searches for the patient by family name. If there is more than one patient with same surname, given name (first name in English) and date of birth are used to identify patient.

Nurse chooses the menu option to add medical history.

Nurse then follows a series of prompts from the system to enter information about consultations elsewhere

- on mental health problems (free text input),
- existing medical conditions (nurse selects conditions from menu),
- medication currently taken (selected from menu),
- allergies (free text),
- and home life (form).



Scenario for collecting medical history in MHC-PMS

What can go wrong: Patient's record does not exist or cannot be found. *Nurse* should create a new record and record personal information.

Patient conditions or medication are not entered in menu. *Nurse* should choose the 'other' option and enter free text describing the condition/medication.

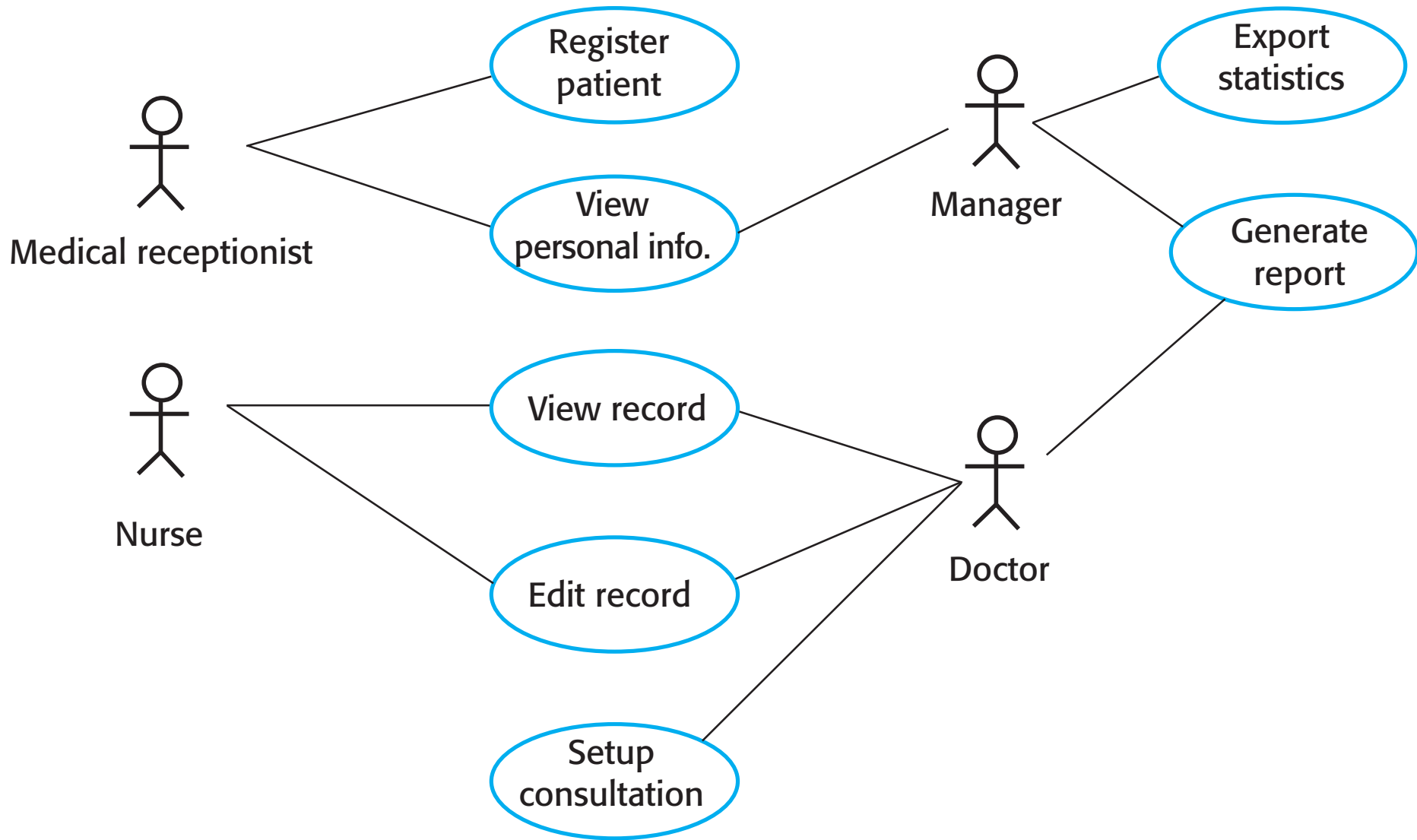
Patient cannot/will not provide information on medical history. *Nurse* should enter free text recording the patient's inability/unwillingness to provide information. System should print standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to patient.

Other activities: Record may be consulted but not edited by other staff while information is being entered.

System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.



Use cases for the MHC-PMS





The structure of a requirements document

Chapter	Description
Preface	This should define expected readership of document and describe its <u>version history</u> , including a <u>rationale</u> for creation of a new version and a <u>summary of changes</u> made in each version.
Introduction	This should describe <u>need for the system</u> . It should briefly describe system's functions and explain how it will work with other systems. It should also describe <u>how system fits into the overall business or strategic objectives</u> of organization commissioning software.
Glossary	This should <u>define technical terms</u> used in document. You should not make assumptions about experience or expertise of reader.
User requirements definition	Here, you <u>describe services provided for user</u> . <u>Nonfunctional system requirements should also be described</u> in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. <u>Product and process standards that must be followed should be specified</u> .
System architecture	This chapter should present <u>a high-level overview of anticipated system architecture, showing distribution of functions across system modules</u> . Architectural components that are reused should be highlighted.



Requirements management planning

- Establishes the level of requirements management detail that is required.
- Requirements management decisions:
 - *Requirements identification* Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
 - *A change management process* This is the set of activities that assess impact and cost of changes.
 - *Traceability policies* These policies define relationships between each requirement and between requirements and system design that should be recorded.
 - *Tool support* Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.