

Requirement Engineering and Scope Management

Buyang Cao

School of Software Engineering

Tongji University

Cell: 15000215319

E-mail: caobuyang@tongji.edu.cn

Contents

- Functional and non-functional requirements
- The software requirements document
- Requirements specification
- Requirements engineering processes
- Requirements elicitation and analysis
- Requirements validation
- Requirements and scope management

Part I

Requirements engineering

- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.

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 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 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.

User and system requirements examples

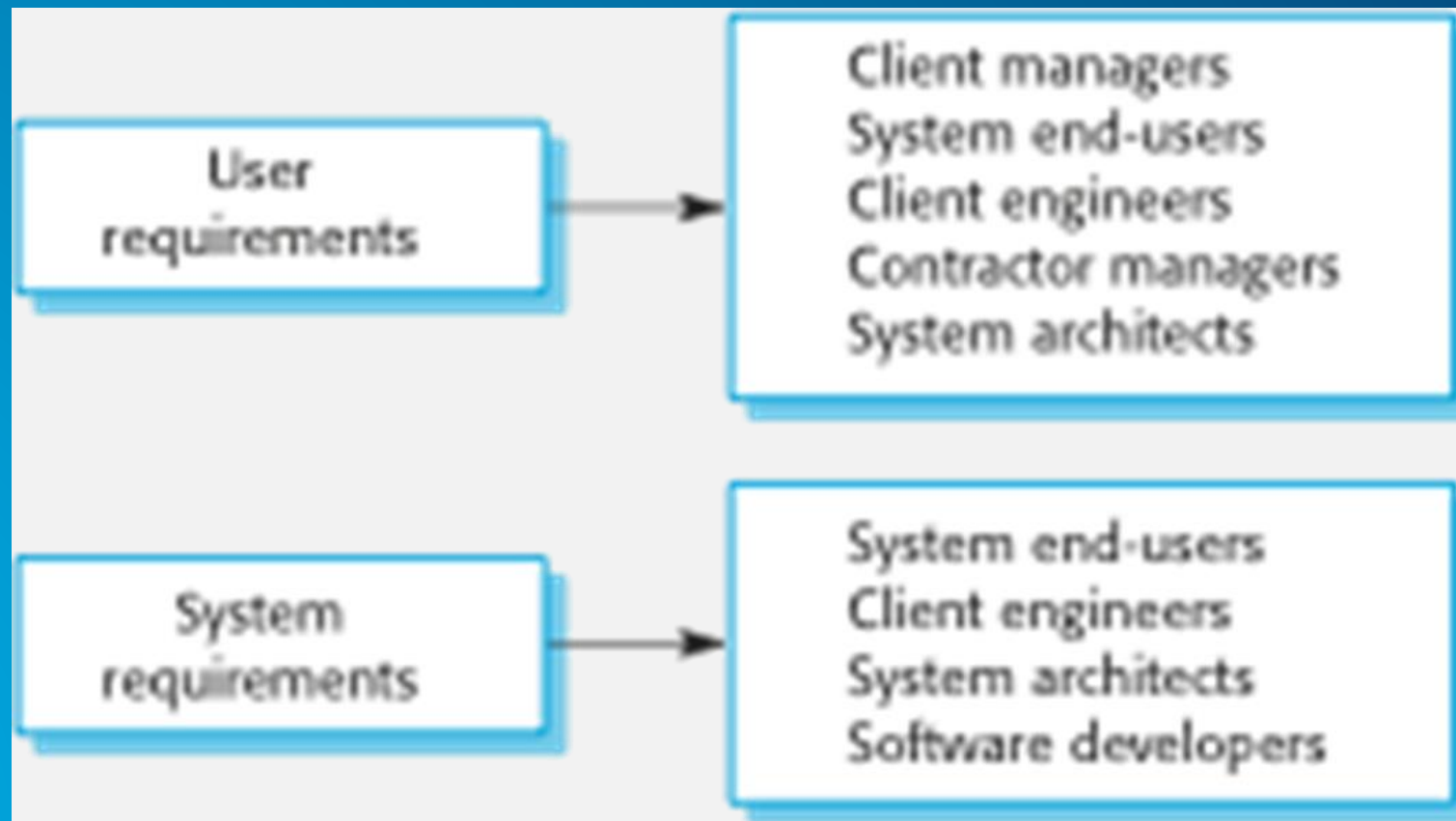
User requirement definition

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 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

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 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.
- Consider the term 'search' in requirement 1 for **MHC-PMS**
 - 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.

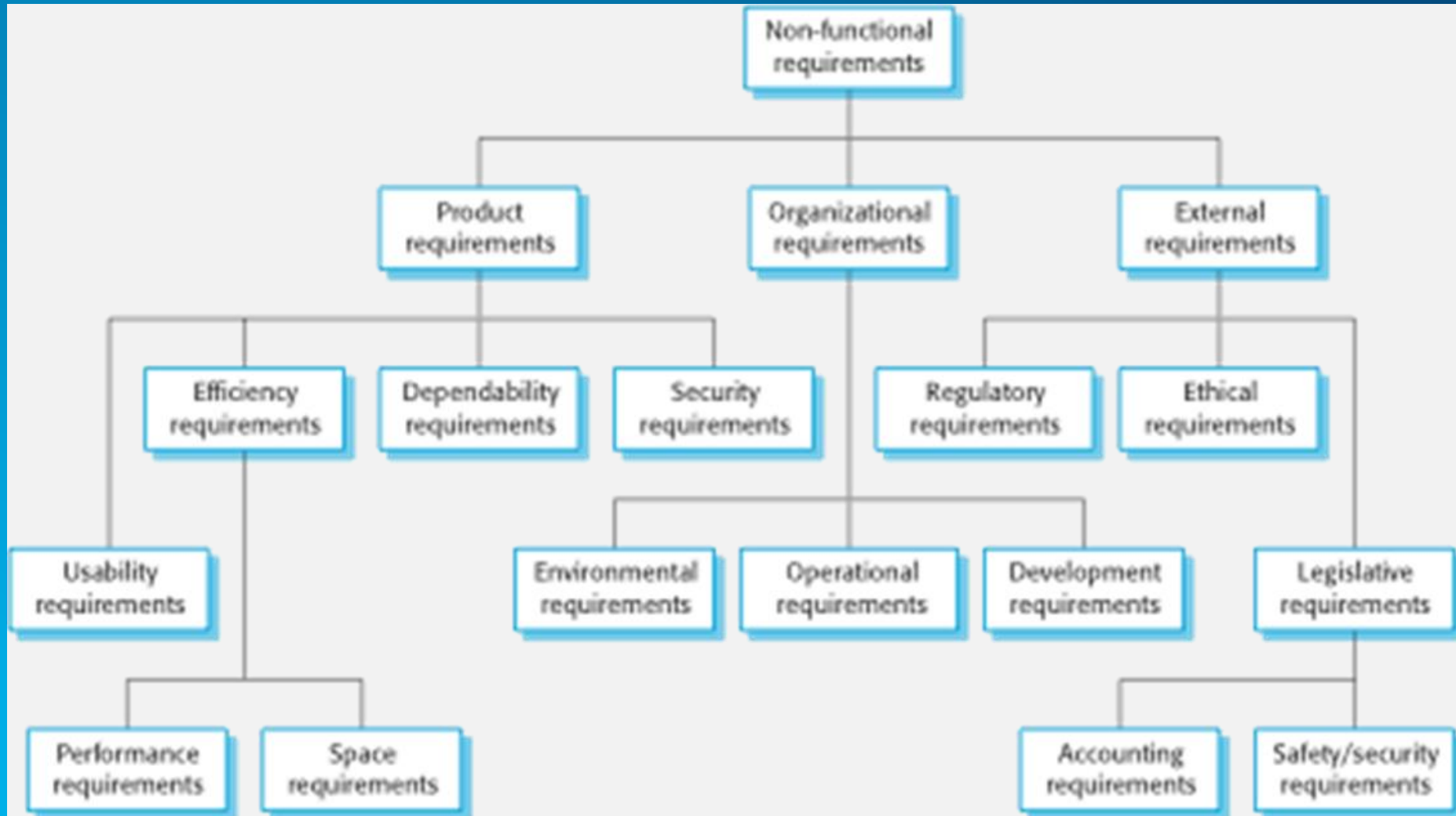
Requirements completeness and consistency

- 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.
- It is usually impossible to produce a complete and consistent requirements document though we should try our best.

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.

Types of nonfunctional requirement



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.

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.

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.

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

Domain requirements

- The system's operational domain imposes requirements on the system.
 - For example, a train control system has to take into account the braking characteristics in different weather conditions.
- Domain requirements be new functional requirements, constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable.

Domain requirements problems

- Understandability
 - Requirements are expressed in the language of the application domain;
 - This is often not understood by software engineers developing the system.
- Implicitness
 - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Key points

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- They often relate to the properties of the system and therefore apply to the system as a whole.

Part II

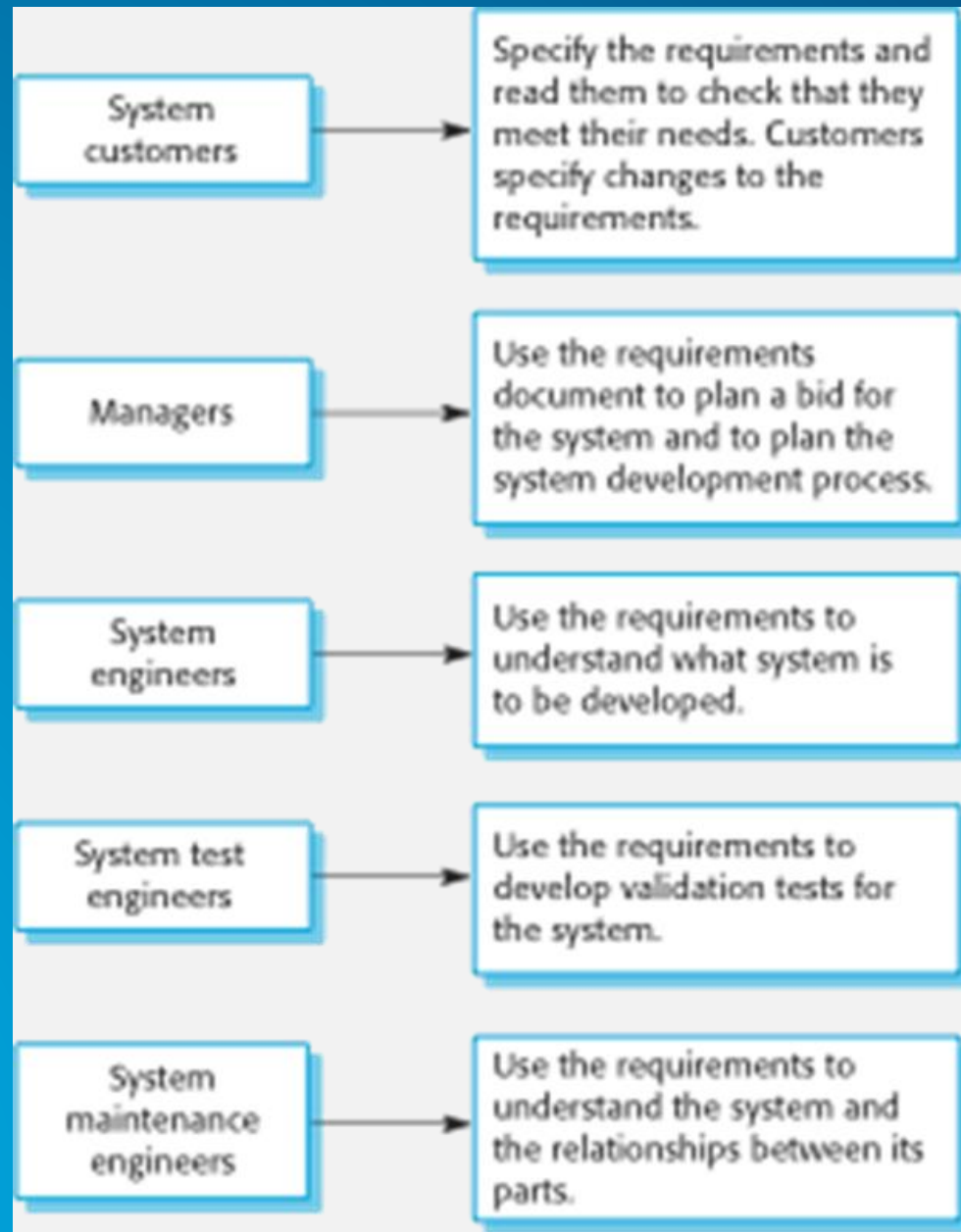
The software requirements document

- 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.

Agile methods and requirements

- Many agile methods argue that producing a requirements document is a waste of time as requirements change so quickly.
- The document is therefore always out of date.
- Methods such as extreme programming use incremental requirements engineering and express requirements as 'user stories'.
- This is practical for business systems but problematic for systems that require a lot of pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Users of a requirements document



Requirements document variability

- Information in requirements document depends on type of system and the approach to development used.
- Systems developed incrementally will, typically, have less detail in the requirements document at early stages.
- Requirements documents standards have been designed e.g. IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

The structure of a requirements document

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

The structure of a requirements document

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the 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 the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the 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.

Requirements specification

- The process of writing down 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.

Requirements and design

- In principle, requirements should state what the system should do and the design should describe how it does this.
- In practice, requirements and design are inseparable
 - A system architecture may be designed to structure the requirements;
 - The system may inter-operate with other systems that generate design requirements;
 - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
 - This may be the consequence of a regulatory requirement.

Guidelines for writing requirements

- Invent a standard format and use it for all requirements.
- Use language in a consistent way, e.g., 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.

Requirements engineering processes

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

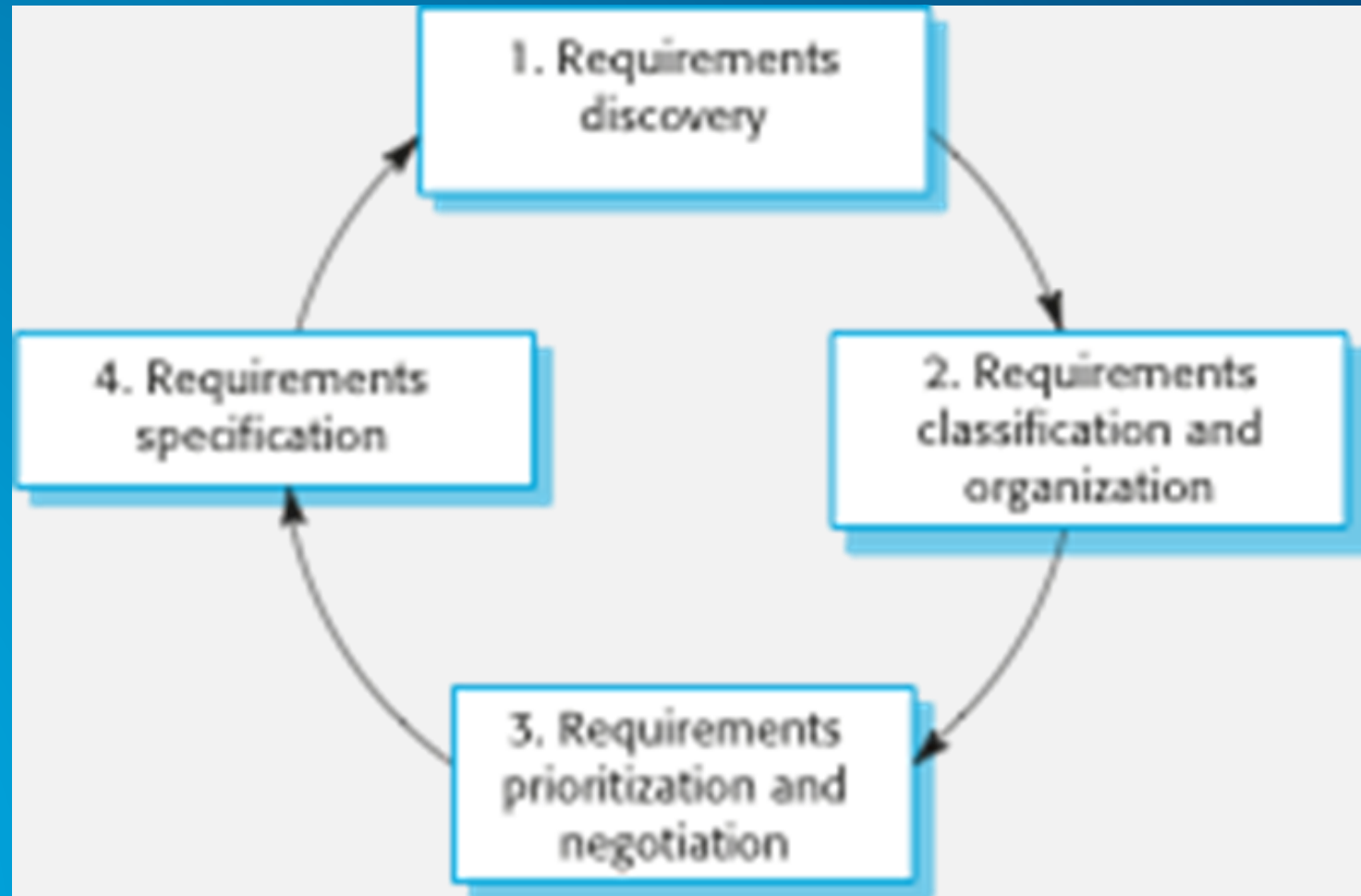
Requirements elicitation and analysis

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

Requirements elicitation and analysis

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
 - Requirements discovery,
 - Requirements classification and organization,
 - Requirements prioritization and negotiation,
 - Requirements specification

The requirements elicitation and analysis process



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.
- Prioritization and negotiation
 - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
 - Requirements are documented and input into the next iteration.

Problems of requirements analysis

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

Key points

- The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- The 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.

Part III

Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

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 the system.

Stakeholders in the MHC-PMS

- 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 the 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.

Scenarios

- Scenarios are real-life examples of how a system can be used.
- They should include
 - A description of the starting situation;
 - A description of the normal flow of events;
 - A description of what can go wrong;
 - Information about other concurrent activities;
 - A description of the state when the scenario finishes.
- In UML-based analysis, it is called ***Use Case***.

Scenario for collecting medical history in MHC-PMS

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

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

The nurse chooses the menu option to add medical history.

The 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: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.

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

Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability/unwillingness to provide information. The system should print the 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 the patient.

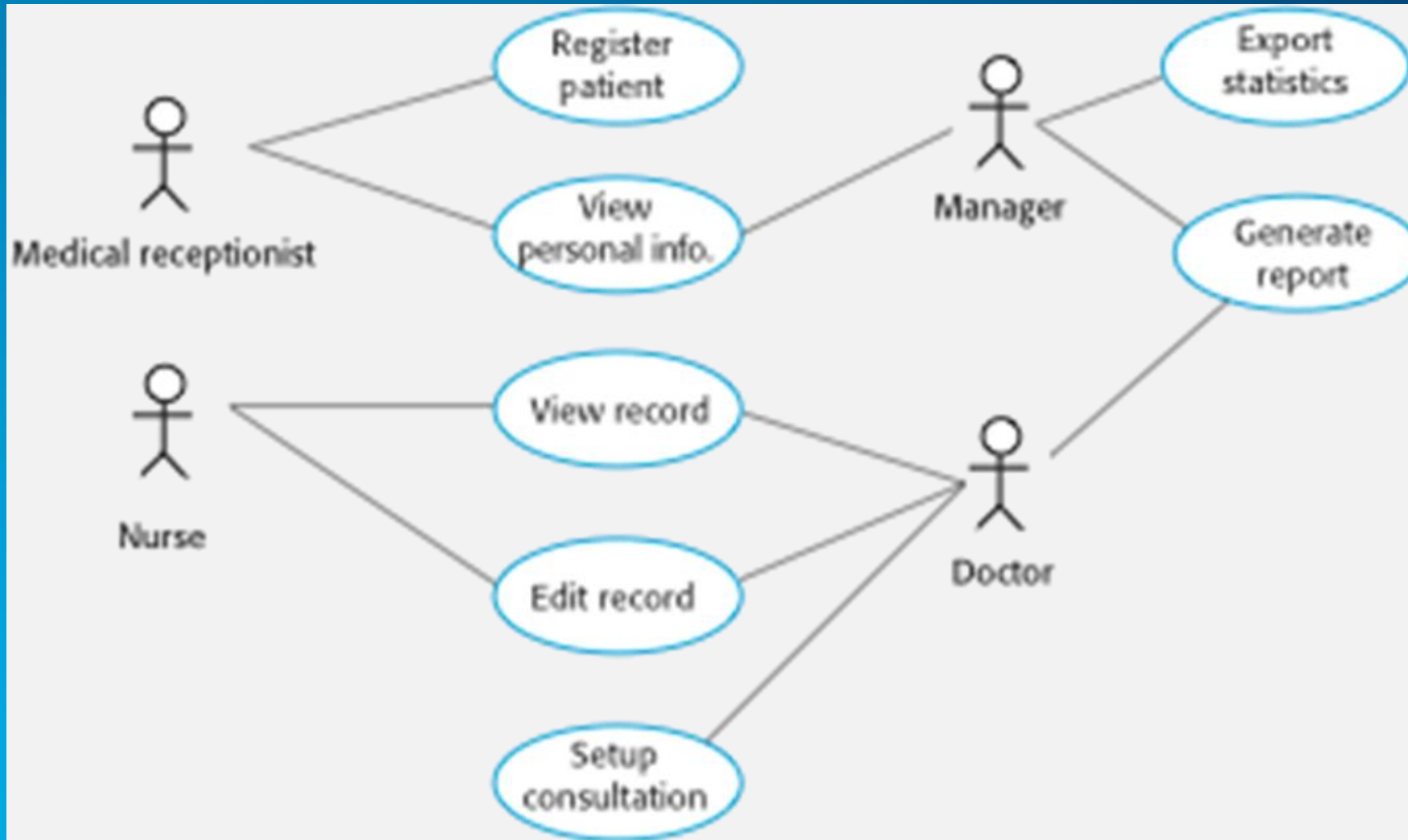
Other activities: Record may be consulted but not edited by other staff while information is being entered.

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

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- High-level graphical model supplemented by more detailed tabular description (we had learned this last semester).
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use cases for the MHC-PMS



Requirements validation

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

Requirements checking

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

Requirements validation techniques

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

Requirements reviews

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

Key points

- Some techniques for requirements elicitation including interviews, scenarios, use-cases.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.

Part IV

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.
- Individual requirements should be tracked and links between dependent requirements are maintained so that the impact of requirements changes can be assessed.
- A formal process for making change proposals and linking these to system requirements needs to be established.

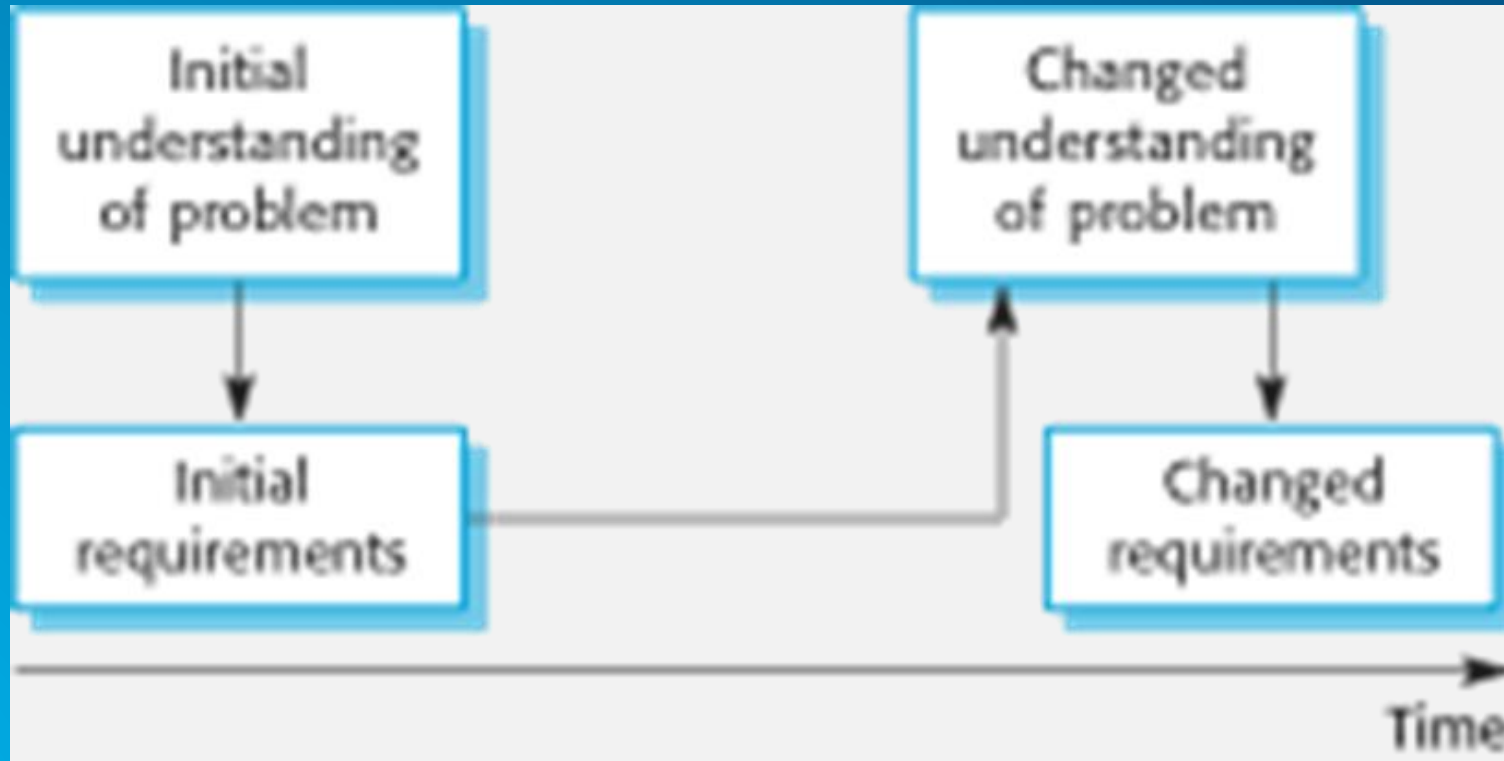
Changing requirements

- 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 (with consequent changes in the system support required), 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 because of 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 evolution



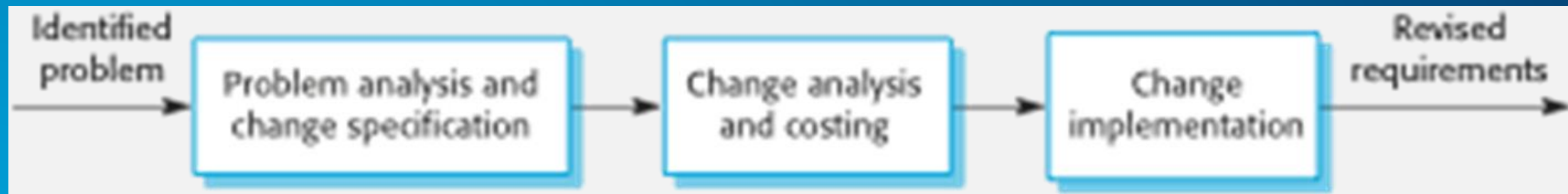
Requirements management planning

- Establishes the level of required requirements management detail.
- 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 the impact and cost of changes. I discuss this process in more detail in the following section.
 - *Traceability policies* These policies define the relationships between each requirement and between the requirements and the 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.

Requirements change management

- Deciding if a requirements change should be accepted
 - *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 the 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*
 - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Requirements change management



Scope Management

- It includes the processes required to ensure that the project includes all the work required, and only the work required, to complete the project successfully.
- Managing the project scope is primarily concerned with defining and controlling what is and is not included in the project.
- It has very close relationship with requirement engineering discussed previously.

Scope Management

- Scope planning: collecting requirements and defining the project scope, deciding how the work breakdown structure (WBS) to be created
- Scope definition: developing a detailed project scope statement as the basis
- Create WBS: dividing the major project deliverables and project work into smaller and more manageable components
- Scope verification: formalizing acceptance of the completed project deliverables
- Scope control: controlling changes into project scope

Define Scope

- The process of developing a detailed description of the project and/or product.
- It is critical to project success and builds upon the major deliverables, assumptions, and constraints that are documented during project initiation.
- During planning, the project scope will be refined and described with greater specificity as more information about the project is known.
- Existing risks, assumptions, and constraints are analyzed for completeness; additional risks, assumptions, and constraints are added as necessary (we will discuss risk management late on).

Define Scope (inputs, tools, outputs)



Output: Project Scope Statement

- It describes in detail the project's deliverables and the work required to create these deliverables.
- It provides the common understanding of the project scope among all project stakeholders.
- It enables the project team to perform more detailed planning, guides the team work during the execution, etc.

Project Scope Statement Contents

- **Project objectives:** attributes such as success criteria, costs, etc.
- **Scope description:** the characteristics of the product/service, or result of the project
- **Project requirements:** conditions or capabilities must be met
- **Project boundaries:** what is included in the project
- **Project deliverables:** product/service and associated support materials
- **Project constraints:** that limit the team's options (e.g., technology, or organization, etc.)
- **Project assumptions:** those associated with the project scope

Project Scope Statement Contents (cont.)

- Initial project organization
- Initial defined risks
- Fund limitation: any limitation placed on the funding for the project
- Cost estimate: rough estimate
- Configuration management: how change (source) control to be implemented
- Project specifications
- Approval requirements: Requirements that can be applied to items such as objectives, deliverables, document, and work.

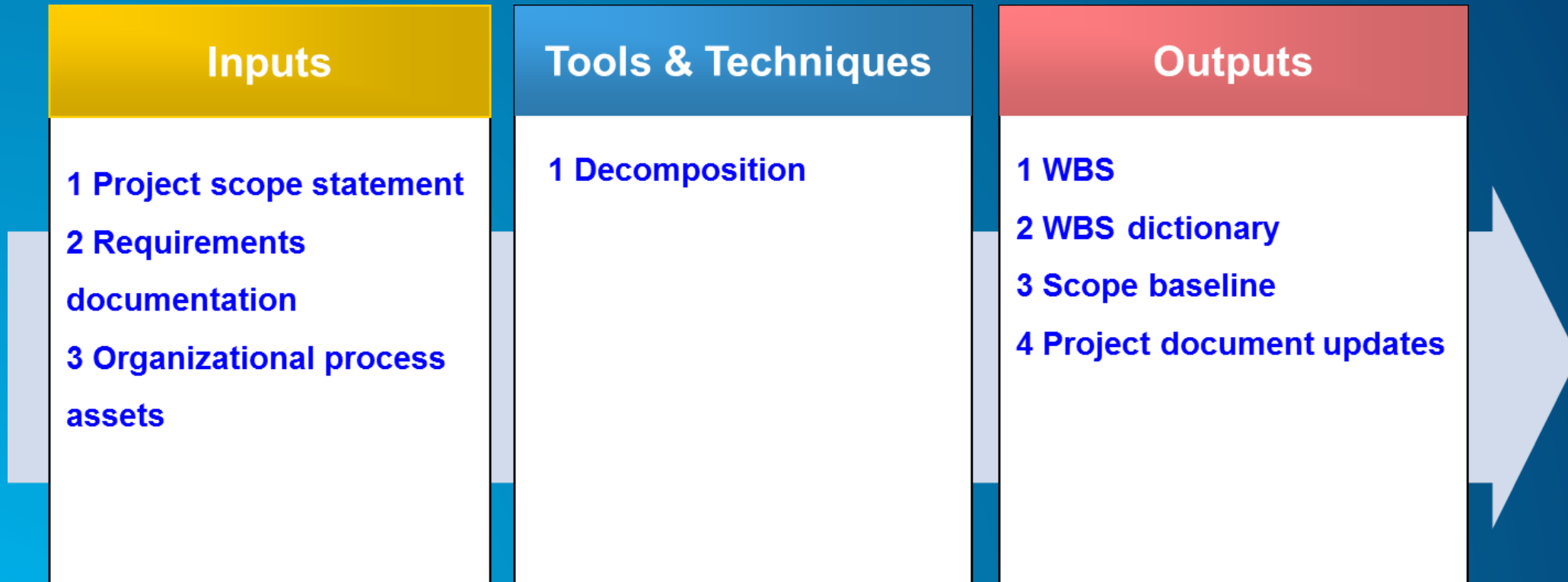
Create WBS

- The process of subdividing project deliverables and project work into smaller, more manageable components.
- The work breakdown structure (WBS) is a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables
- Each descending level of the WBS representing an increasingly detailed definition of the project work.
- The WBS organizes and defines the total scope of the project, and represents the work specified in the current approved project scope statement.

Work Breakdown Structure

- The planned work is contained within the lowest level WBS components, which are called *work packages*.
- A work package can be scheduled, cost estimated, monitored, and controlled. In the context of the WBS, work refers to working on products or deliverables that are the result of effort.

Workflow of Creating WBS



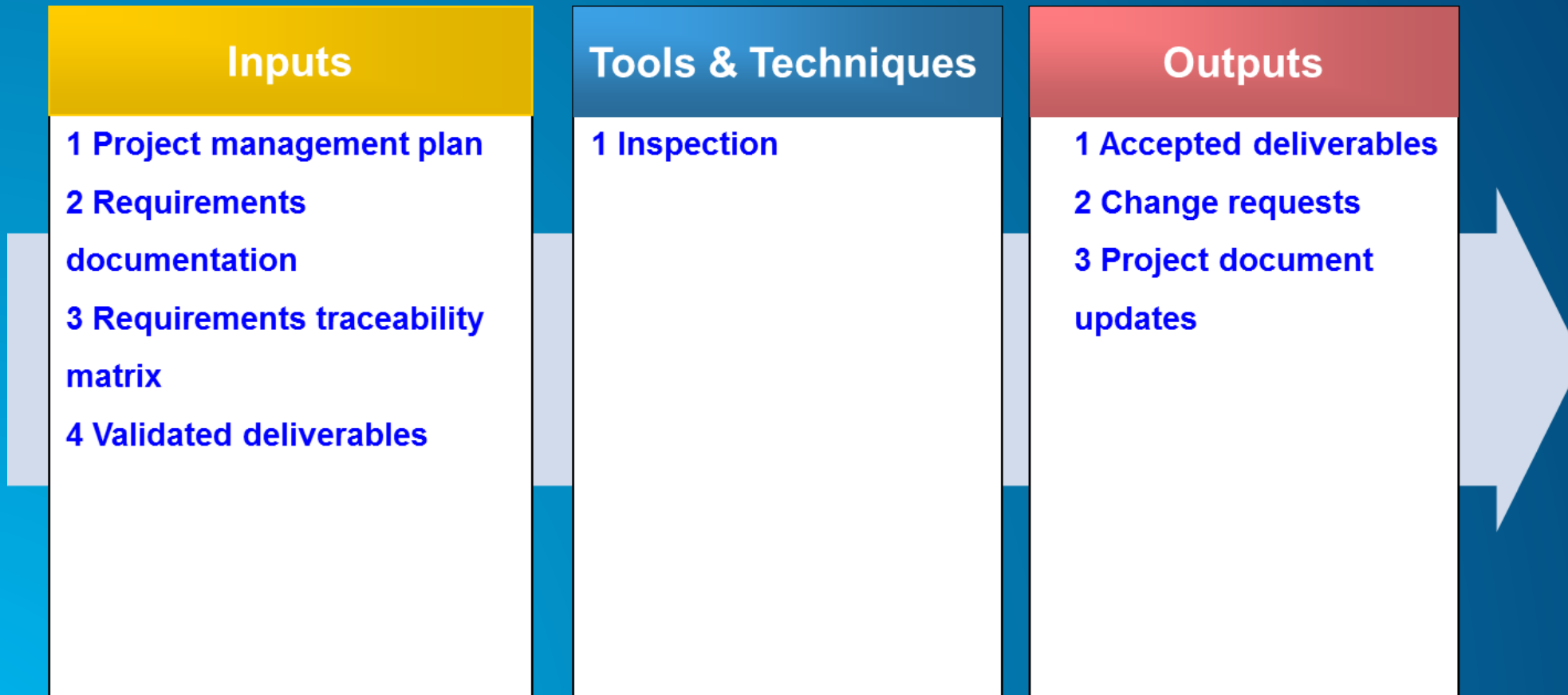
Outputs of WBS creation

- Updated project scope statement: something might have been changed during WBS creation
- Work breakdown structure
- WBS dictionary: the content of the components contained in WBS such as work packages and control accounts, identifiers, milestone of each WBS, etc.
- Requested changes: requested changes to scope statement and components

Verify Scope

- The process of formalizing acceptance of the completed project deliverables.
- Verifying scope includes reviewing deliverables with the customer or sponsor to ensure that they are completed satisfactorily and obtaining formal acceptance of deliverables by the customer or sponsor.
- Scope verification is primarily concerned with acceptance of the deliverables (what a project team must deliver)

Verify Scope



Outputs of Scope Verification

- Accepted deliverables: the document contains deliverables that have been accepted (approved).
- Requested change: certain changes generated from the scope verification
- Recommended corrective actions (what needs to be corrected and how)

Control Scope

- Project scope control is used to manage the actual changes when they occur and is integrated with the other control processes.
- Uncontrolled changes are often referred to as project scope creep.
- Change is inevitable, thereby mandating some type of change control process.

Control Scope

Inputs

- 1 Project management plan
- 2 Work performance information
- 3 Requirements documentation
- 4 Requirements traceability matrix
- 5 Organizational process assets

Tools & Techniques

- 1 Variance analysis

Outputs

- 1 Work performance measurements
- 2 Organizational process assets updates
- 3 Change requests
- 4 Project management plan updates
- 5 Project document updates

Control Scope: Technique

- Variance analysis: to assess the magnitude of variation.
Important aspects of project scope control include determining the cause of variance relative to the scope baseline.