Requirement Engineering:

An Overview

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

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 ambiguity is inevitable because requirements may serve a dual function.
  - They may be the basis for a **bid for a contract**—therefore, must be open to interpretation.
  - They may be the basis for the **contract itself**—therefore, must be defined in detail.
- Both these statements may be called requirements.

Requirements Abstraction (Davis)

"If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not predefined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization's needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system."

Agile Methods and Requirements

- Many agile methods argue that producing a requirements document is a waste of time as requirements change so quickly, and, therefore, the document is always out of date.
  - Methods such as XP use incremental planning (requirements engineering) and express requirements as "user stories" as discussed earlier.
  - This is practical for business systems but problematic for systems that require a lot of predelivery analysis (e.g., safety-critical systems) or systems developed by several teams.

A “Prescribing Medication’ Story

- In XP, a customer or user is part of the XP team and is responsible for making decisions on requirements.
- User requirements are expressed as scenarios or user stories.
- These are written on cards.

Prescribing medication

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select ‘current medication’, ‘new medication’ or ‘formulary’.

If she selects ‘current medication’, the system asks her to check the dose; If she wants to change the dose, she enters the new dose then confirms the prescription.

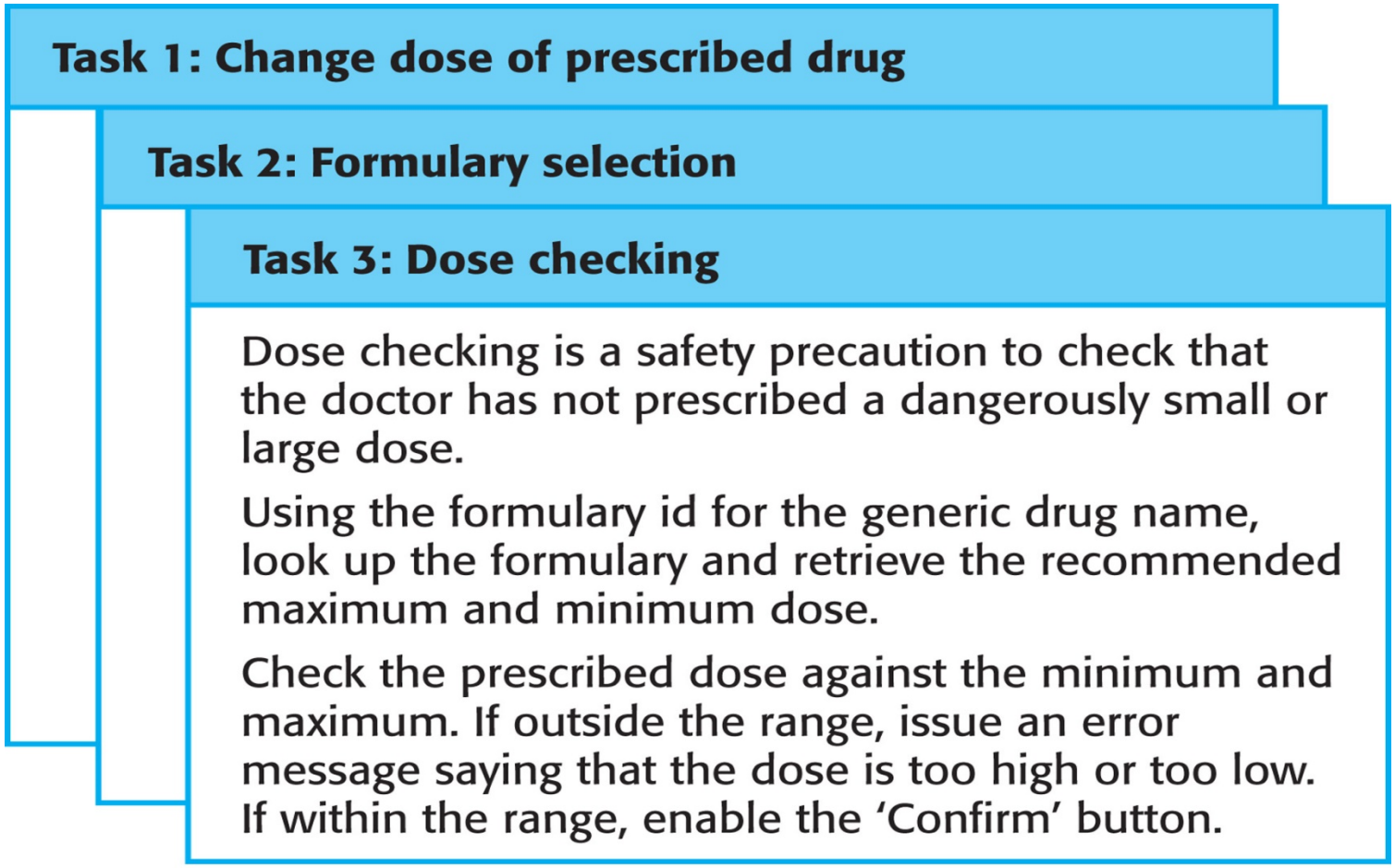
If she chooses ‘new medication’, the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose then confirms the prescription.

If she chooses ‘formulary’, the system displays a search box for the approved formulary. She can then search for the drug required. She selects a drug and is asked to check that the medication is correct. She enters the dose then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn’t, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks ‘OK’ or ‘Change’. If she clicks ‘OK’, the prescription is recorded on the audit database. If she clicks on ‘Change’, she reenters the ‘Prescribing medication’ process.

Examples of Task Cards



- The development team breaks them down into implementation tasks. These tasks are the basis of schedule and cost estimates.

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

User Requirements vs. System Requirements

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Types of Requirement

- **User requirements**
  - Statements in natural language (plus diagrams) of the services the system provides and its operational constraints; **written for customers**
- **System requirements**
  - A structured document setting out detailed descriptions of the system's services (functions) and operational constraints.
  - It defines what should be implemented so may be part of a **contract** between client and contractor.

User and System Requirements

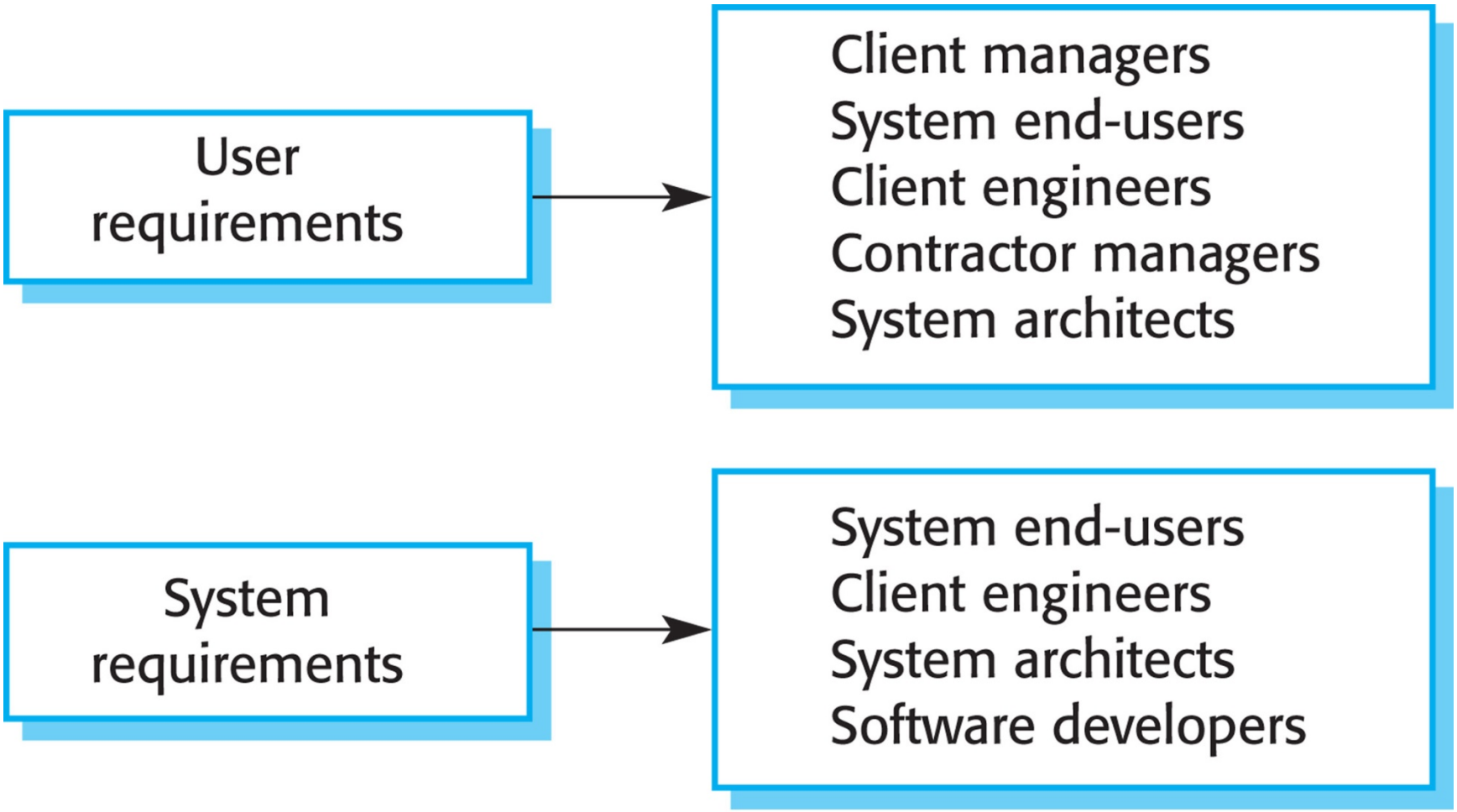
User requirements definition

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.

Readers of Different Types of Requirements Specification



System Stakeholders

- Any person or organization who is affected by the system in some way and who has a legitimate interest

- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Stakeholders in the Mentcare System

- **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 Mentcare System (cont.)

- 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

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Functional Requirements vs. Nonfunctional Requirements

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Functional/Nonfunctional Req.

- 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.

- Nonfunctional requirements
  - **Constraints** on the services offered by the system such as timing constraints, constraints on the development process, standards, and so on.
  - Often apply to the system as a whole rather than individual features or services.

Four Types of Requirements

- Functional user requirements
  - High-level statements about the **services** the system should provide
- Functional system requirements
  - Should describe the system **services** in detail
- Nonfunctional user requirements
  - High-level statements of the **constraints** on the services or functions offered by the system
- Nonfunctional system requirements
  - Should describe the **constraints** in detail.

Mentcare: 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 eight-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 (previous slide):

    "A user shall be able to search the appointments lists **for all clinics**."
    - 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.
- In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Nonfunctional Requirements

- These define system constraints.
  - System properties: reliability, response time and storage requirements, and so on.
  - Constraints: I/O device capability, data representations, and so on.
  - Process requirements may also be specified, mandating a particular IDE, programming language, or development method.
- Nonfunctional requirements may be more critical than functional requirements.
  - If these are not met, the system may be useless.

Importance of Nonfunctional Requirements

- Nonfunctional requirements may affect the overall architecture of a system rather than the individual components.
  - Example: To ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single nonfunctional 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.

Types of Nonfunctional Requirements

- Product requirements
  - Requirements that specify that the delivered product must behave in a particular way, for example, execution speed, reliability, and so on
- Organizational requirements
  - Requirements that are a consequence of organizational policies and procedures, such as process standards used, and implementation requirements
- External requirements
  - Requirements that arise from factors external to the system and its development process, such as interoperability requirements, legislative requirements, and so on

Examples in the Mentcare System

PRODUCT REQUIREMENT

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 08:30–17:30). Downtime within normal working hours shall not exceed 5 seconds in any one day.

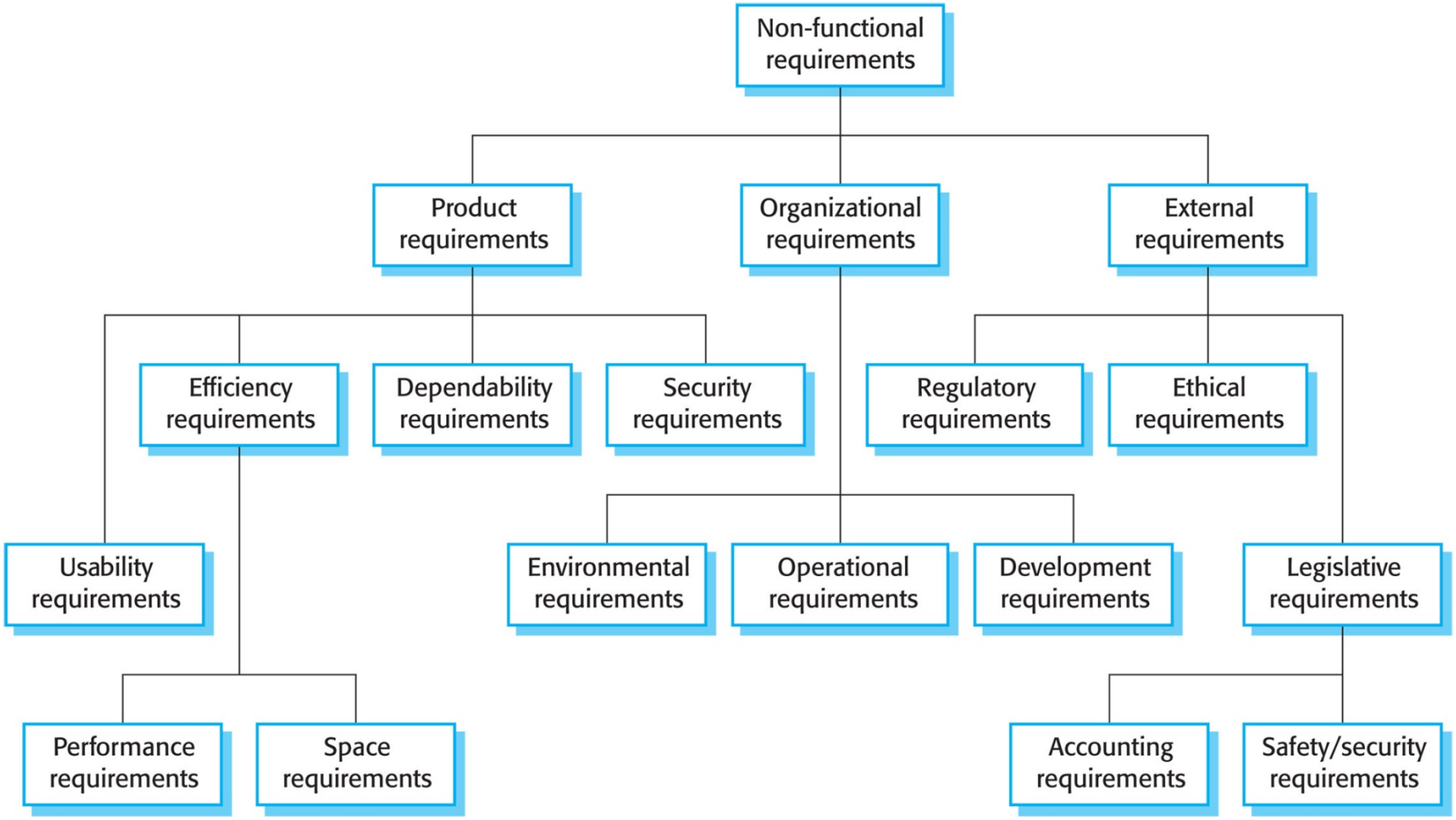
ORGANIZATIONAL REQUIREMENT

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

EXTERNAL REQUIREMENT

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

Types of Nonfunctional Requirement



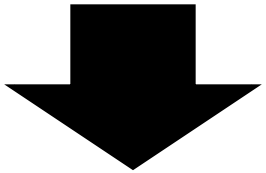
Goals and Requirements

- Nonfunctional requirements may be very difficult to state precisely, and imprecise requirements may be difficult to verify.
- Remedy: Start with goals, and convert them into verifiable nonfunctional requirements.
  - Goal
    - A general intention of the user, such as ease of use
  - Verifiable nonfunctional 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

- Goal:

  The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.



- Verifiable nonfunctional requirement

  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.

Metrics for Nonfunctional Req.

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

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Requirements Engineering Process: An Overview

Week 3: Requirements Engineering

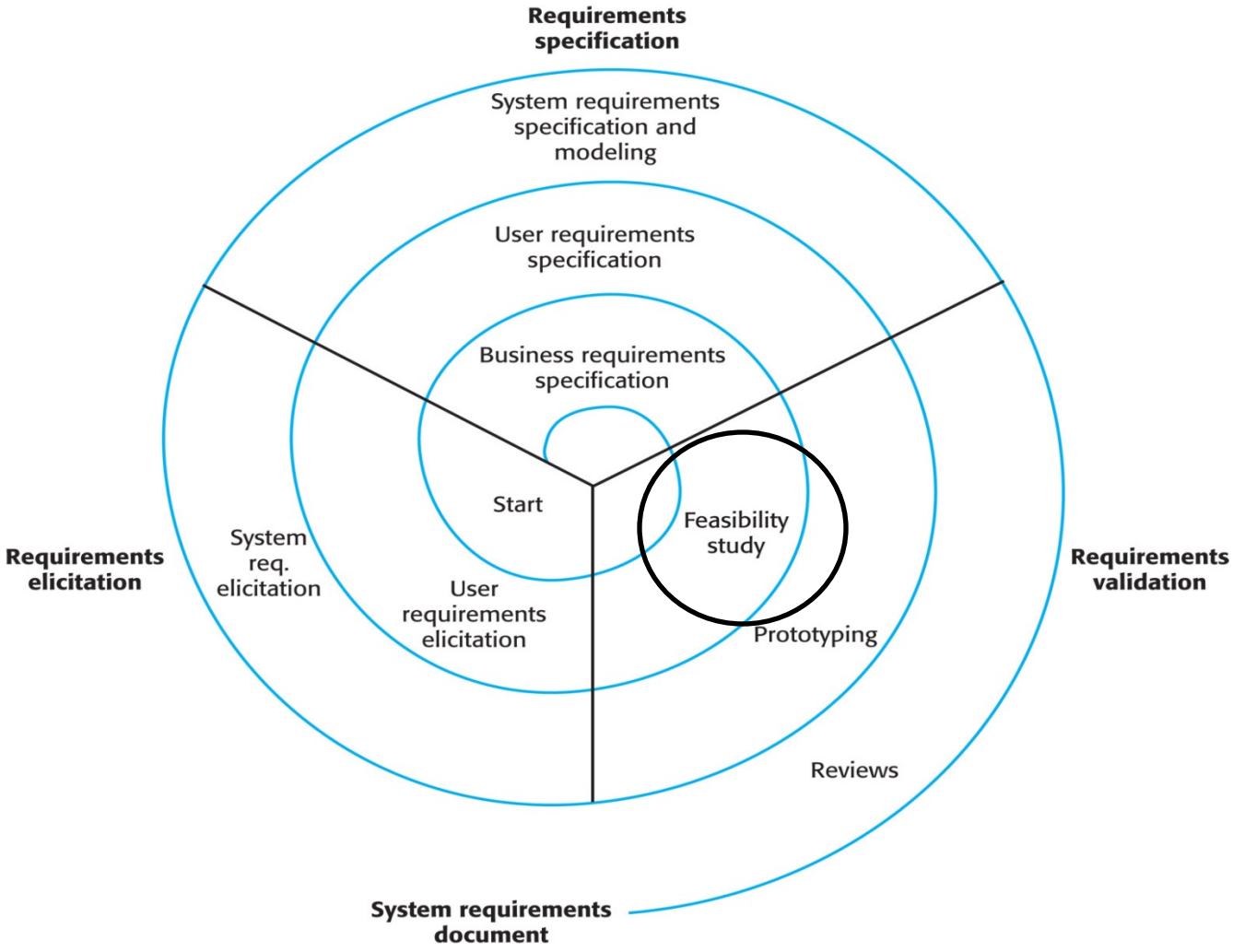
Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Requirements Engineering Processes

- 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.
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management
- In practice, RE is an iterative activity in which these processes are interleaved.

The Spiral RE View



Feasibility Study

- The aims of a feasibility study are to find out whether the system is worth implementing and if it can be implemented, given the existing budget and schedule.

- The input to the feasibility study is a set of preliminary business requirements, an outline description of the system, and a description of how the system is intended to support business processes.

- The results of the feasibility study should be a report (feasibility report) that recommends whether or not it is worth carrying on with the requirements engineering and system development process.

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

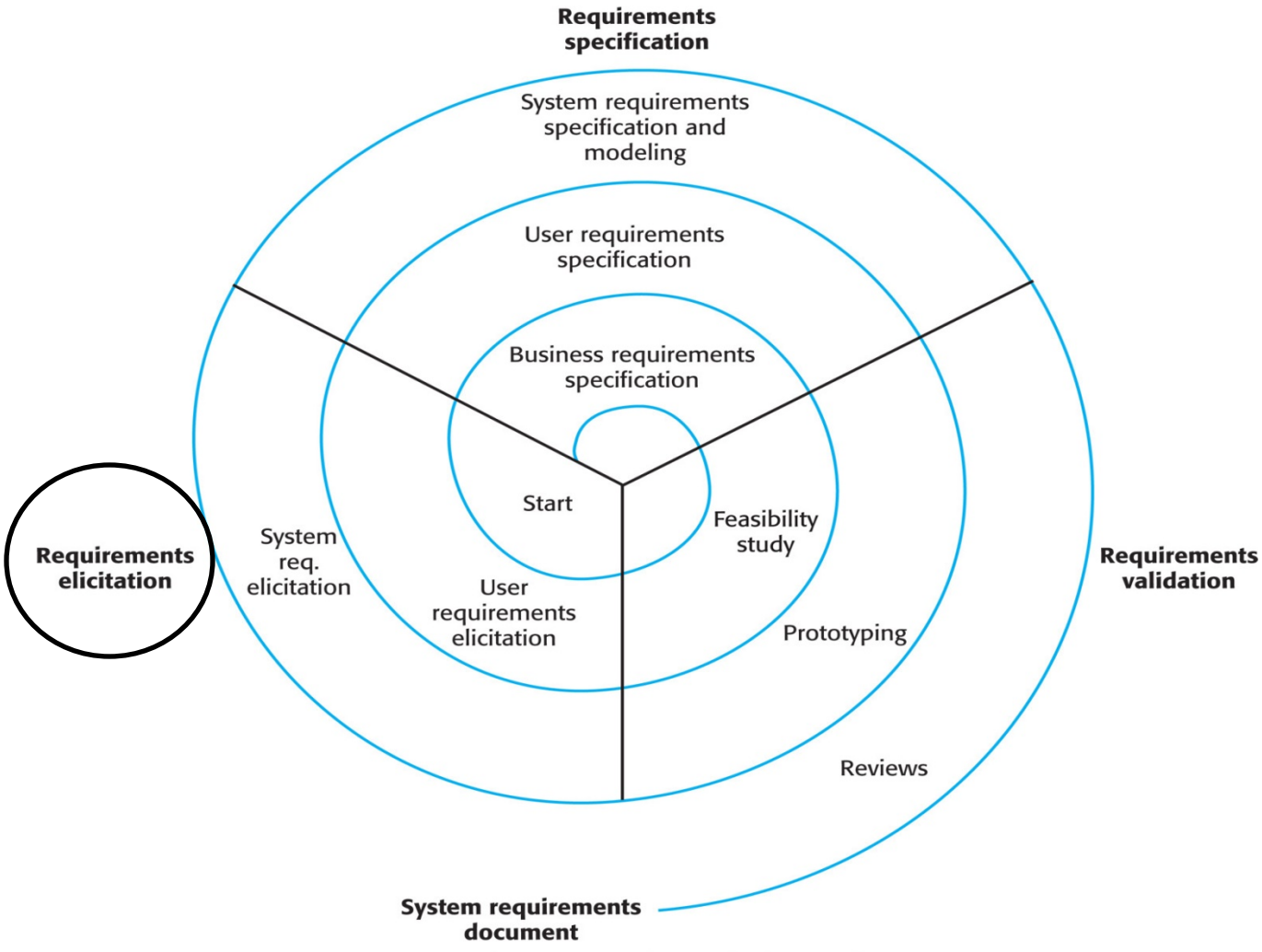
Requirements Elicitation/Analysis

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

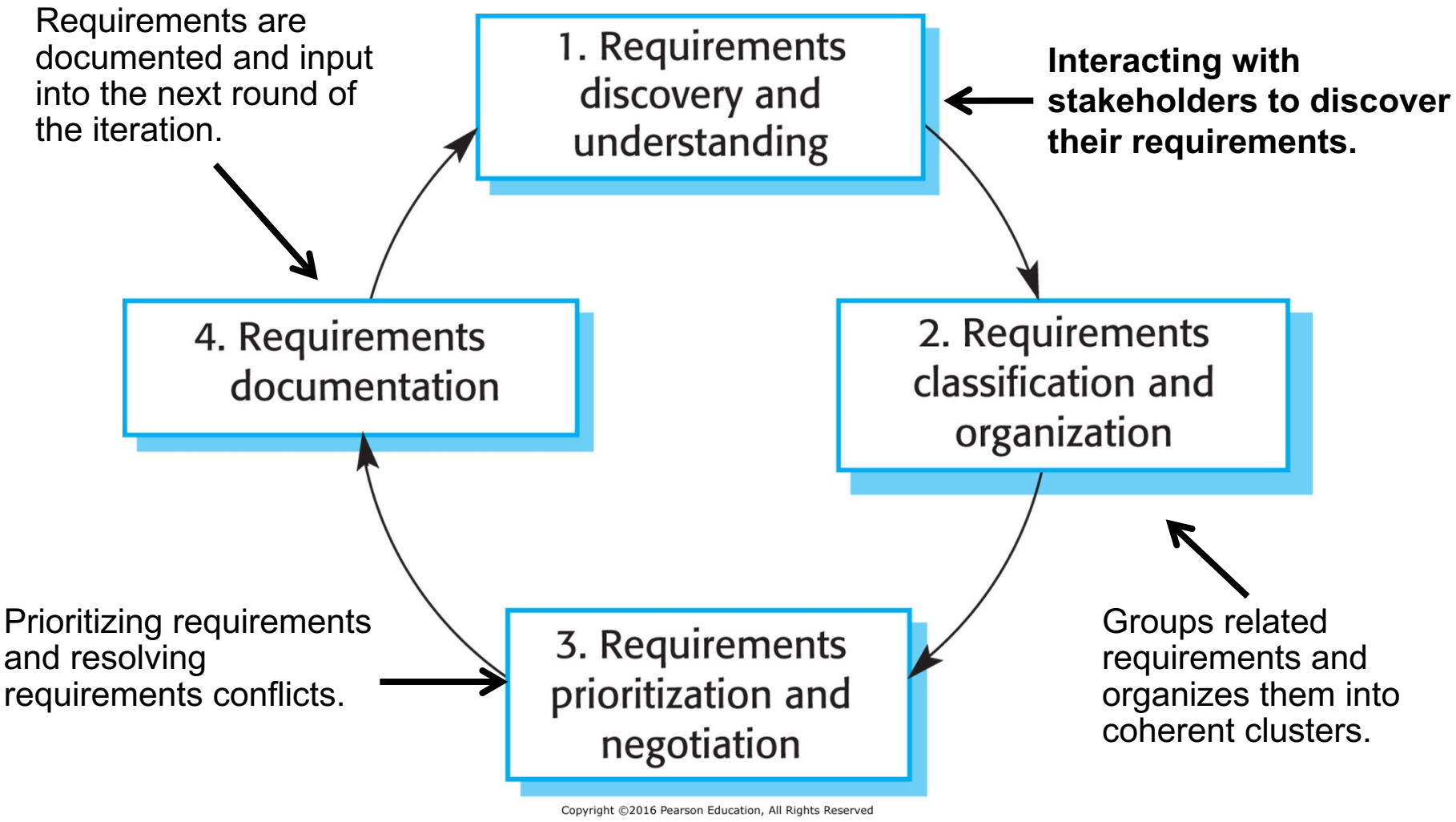
Requirements Elicitation/Analysis



Requirements Elicitation/Analysis

- Sometimes called requirements discovery (or, requirements gathering in the industry)
- 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, and so on; these are called stakeholders

Requirements Elicitation/Analysis



Requirements Discovery

- The process of gathering information about the required/existing systems and distilling the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation.
- You may also use scenarios and prototypes to help stakeholders understand what the system will be like.
- Systems normally have a range of stakeholders.

Requirements Discovery

- The process of gathering information about the required/existing systems and distilling the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation.
- You may also use scenarios and prototypes to help stakeholders understand what the system will be like.

Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview:
  - Closed interviews based on predetermined list of questions
  - Open interviews where various issues are explored with stakeholders
- Effective interviewing:
  - Be open-minded, avoid preconceived ideas about the requirements, and be willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype 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.

Requirements Discovery

- The process of gathering information about the required/existing systems and distilling the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation.
- You may also use scenarios and prototypes to help stakeholders understand what the system will be like.

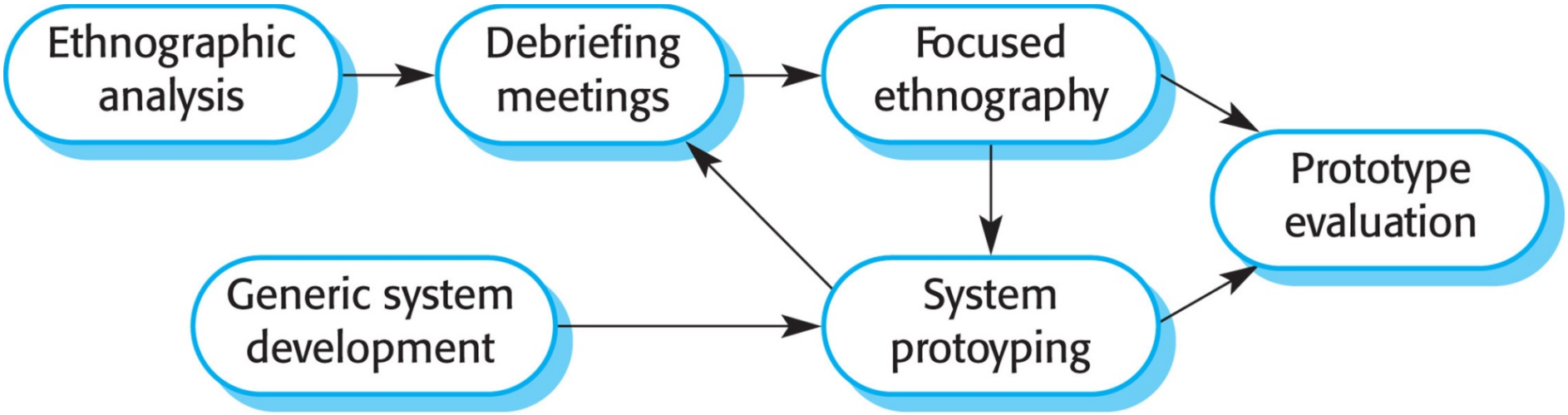
Ethnography

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

Scope of Ethnography

- Requirements are derived from the way that people actually work rather than the way process definitions suggest that they ought to work.
- Requirements are derived from cooperation and awareness of other people's activities.
  - Awareness of what other people are doing leads to changes in the ways in which we do things.
- Drawback:
  - Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Ethnography and Prototyping for Requirements Analysis



Requirements Discovery

- The process of gathering information about the required/existing systems and distilling the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation.
- You may also use scenarios and prototypes to help stakeholders understand what the system will be like.

Scenarios

- A structured form of user story
- Scenarios 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

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

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).

A Scenario for Mentcare

- 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

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.

Uploading Photos (ilearn)

Photo sharing in the classroom

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused on the fishing industry in the area, looking at the history, development, and economic impact of fishing. As part of this project, pupils are asked to gather and share reminiscences from relatives, use newspaper archives, and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo-sharing site because he wants pupils to take and comment on each other's photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers' group, which he is a member of, to see if anyone can recommend an appropriate system. Two teachers reply, and both suggest that he use KidsTakePics, a photo-sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

Uploading Photos (ilearn) (cont.)

Uploading photos to KidsTakePics

Initial assumption: A user or a group of users have one or more digital photographs to be uploaded to the picture-sharing site. These photos are saved on either a tablet or a laptop computer. They have successfully logged on to KidsTakePics.

Normal: The user chooses to upload photos and is prompted to select the photos to be uploaded on the computer and to select the project name under which the photos will be stored. Users should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.

On completion of the upload, the system automatically sends an email to the project moderator, asking them to check new content, and generates an on-screen message to the user that this checking has been done.

What can go wrong: No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator. Users should be informed of a possible delay in making their photos visible.

Photos with the same name have already been uploaded by the same user. The user should be asked if he or she wishes to re-upload the photos with the same name, rename the photos, or cancel the upload. If users choose to re-upload the photos, the originals are overwritten. If they choose to rename the photos, a new name is automatically generated by adding a number to the existing filename.

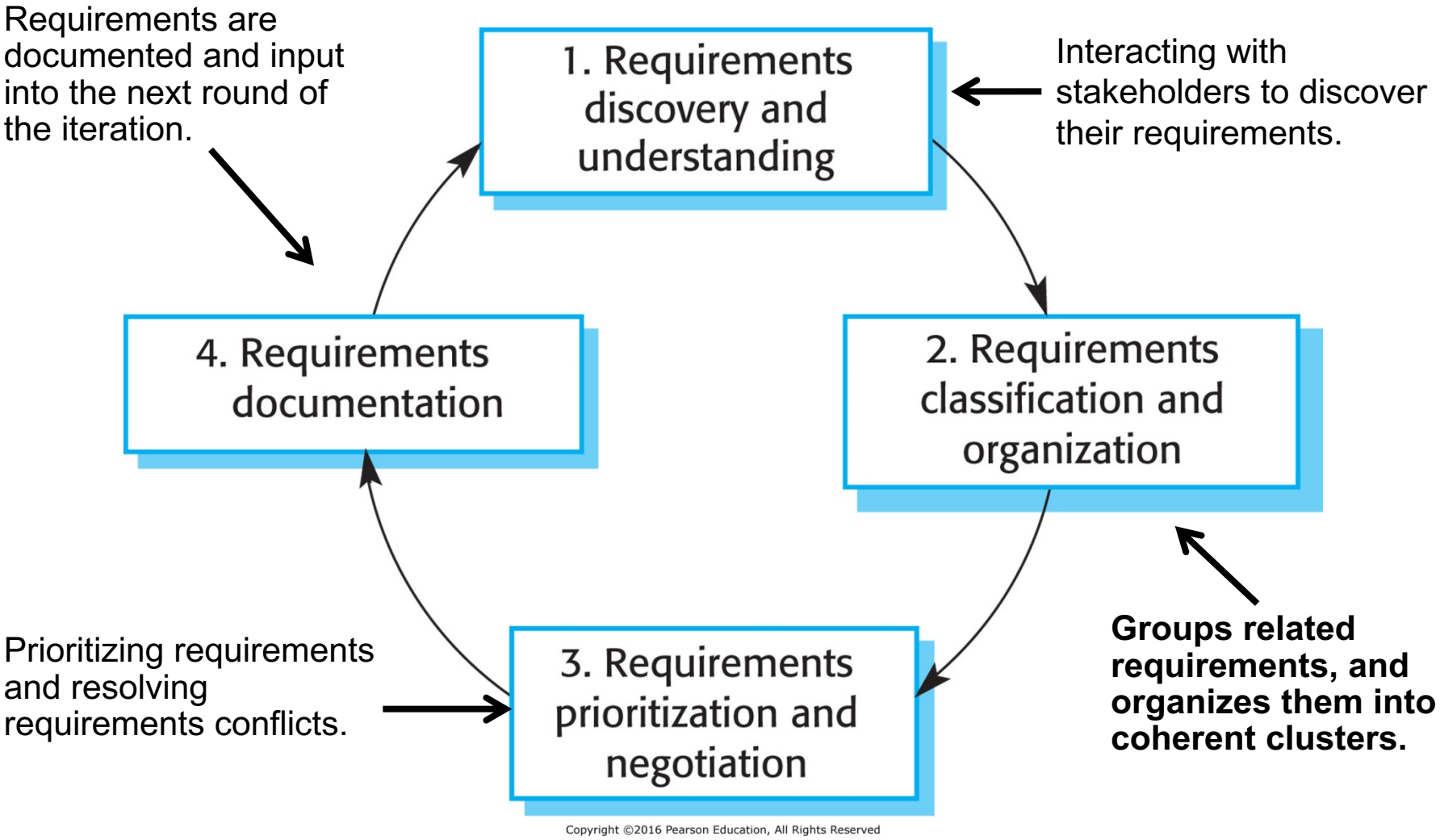
Other activities: The moderator may be logged on to the system and may approve photos as they are uploaded.

System state on completion: User is logged on. The selected photos have been uploaded and assigned a status "awaiting moderation." Photos are visible to the moderator and to the user who uploaded them.

Requirements Discovery (Revisited)

- The process of gathering information about the required/existing systems and distilling the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation.
- You may also use scenarios and prototypes to help stakeholders understand what the system will be like.

Requirements Elicitation/Analysis



Requirements Classification/Org.

- **Viewpoint-oriented** approaches to requirements engineering organize both the elicitation process and the requirements themselves using different viewpoints.
- A viewpoint is a way of organizing the requirements for a software system, based on some perspective such as an end user perspective or a manager's perspective.
- A key strength of viewpoint-oriented analysis is that:
  - It recognizes the existence of multiple perspectives.
  - It provides a framework for discovering conflicts in the requirements proposed by different stakeholders.

Requirements Classification/Org. (cont.)

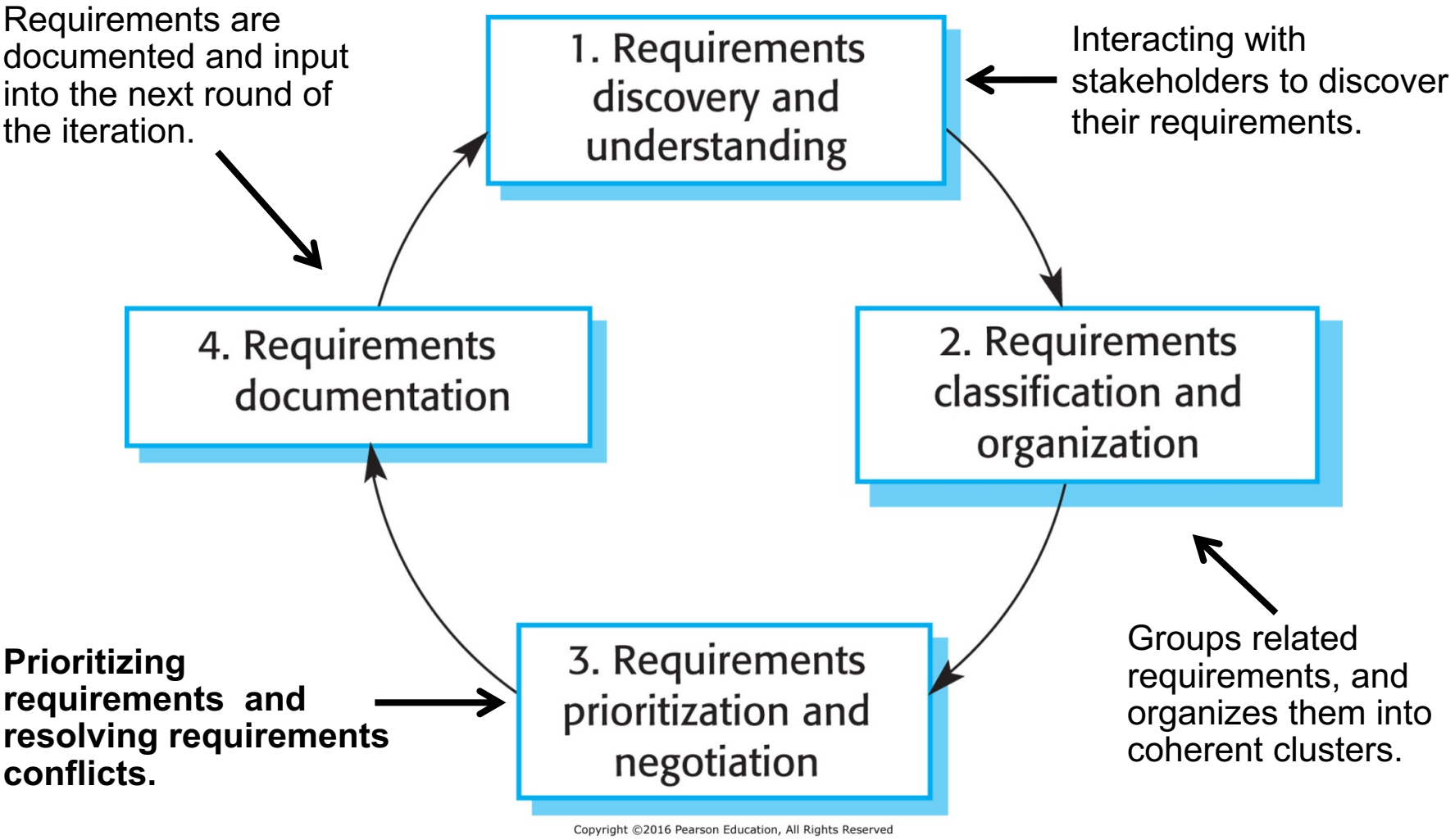
- There are three generic types of viewpoint:
  - **Interactor viewpoints** that represent people or other systems that interact directly with the system. In the bank ATM system, examples of interactor viewpoints are the bank's customers and the bank's account database.

    → Detailed system requirements covering the system features and interfaces
  - **Indirect viewpoints** that represent stakeholders who do not use the system themselves but who influence the requirements in some way. In the bank ATM system, examples of indirect viewpoints are the management of the bank and the bank security staff.

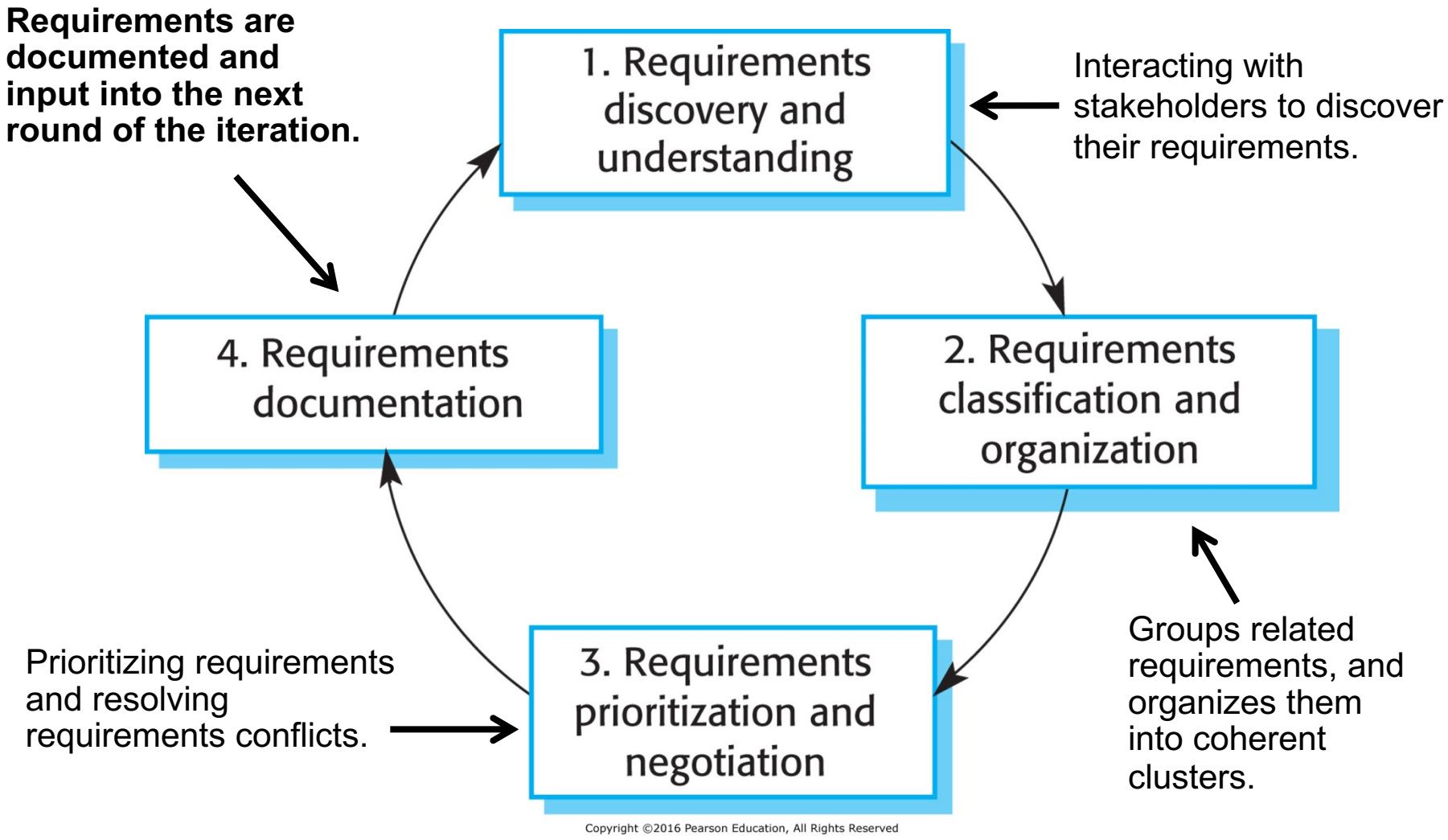
    → Higher-level organizational requirements and constraints
  - **Domain viewpoints** that represent domain characteristics and constraints that influence the system requirements. In the bank ATM system, an example of a domain viewpoint would be the standards that have been developed for interbank communications.

    → Domain constraints that apply to the system

Requirements Elicitation/Analysis



Requirements Elicitation/Analysis



SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

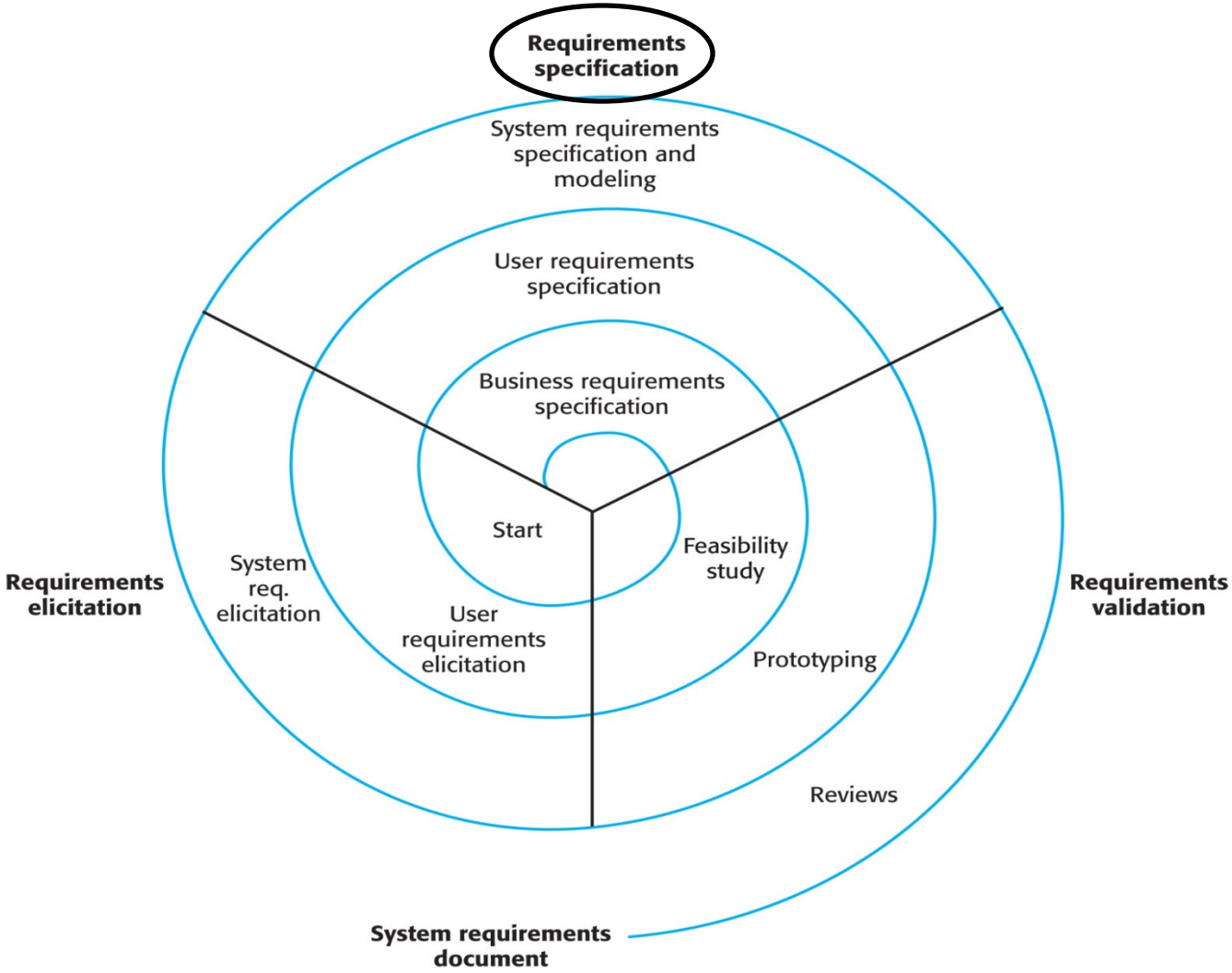
Requirements Specification

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

Requirements Specification



Requirements Specification (cont.)

- 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.

User and System Requirements (Revisited)

User requirements definition

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

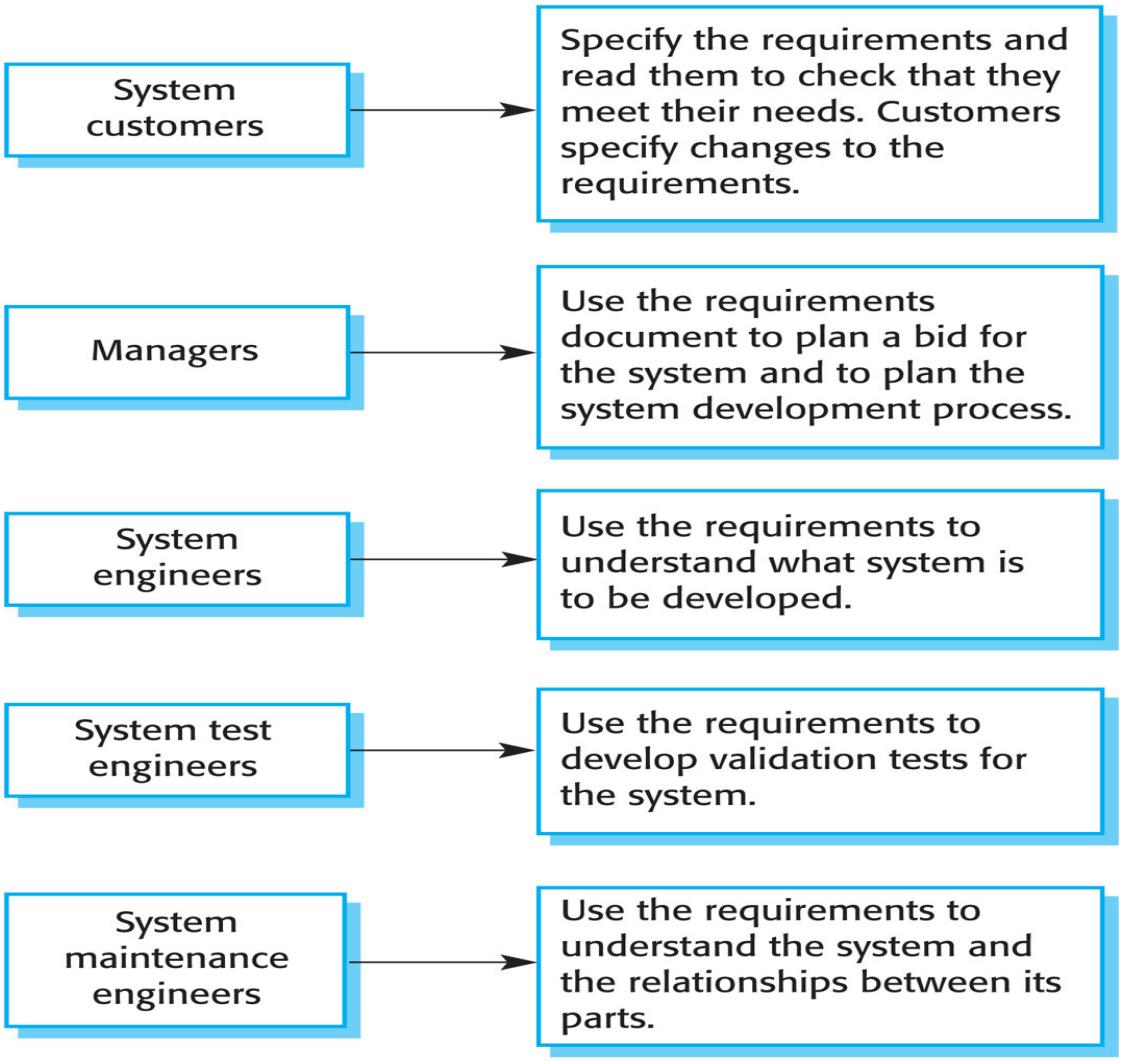
System requirements specification

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.

Software Requirements Document (aka SRS)

- The software requirements document is the official statement of what is required of the system developers.
  - Should include both user requirements and system requirements (further divided by functional vs. nonfunctional).
- It is NOT a design document.
  - It should specify WHAT the system should do rather than HOW the system 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.
  - Systems developed incrementally will, typically, have less detail in the requirements document.
  - Requirements documents standards have been designed, e.g., **IEEE standard**.
    - These are mostly applicable to the requirements for large systems engineering projects.
  - For this course, we will use a simplified template from the textbook.

Structure of a Requirements Doc.

Section	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 **nonfunctiona**l 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.

Structure of a Requirements Doc. (cont.)

Section	Description
System requirements specification	This should describe the **functional** and **nonfunctiona**l 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 because 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.

Ways of Writing a System Req. Spec.

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want, and they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

Natural Language Specification

- Requirements can be written as natural language (e.g., English) sentences supplemented by diagrams and tables.
  - We use a natural language to write/specify requirements because it is expressive, intuitive, and universal.
  - This means that the requirements can be understood by users and customers.

Guidelines for Writing Requirements

- Invent a standard format (e.g., one sentence/requirement), and use it for all requirements.
- Use language in a consistent way. Use shall for mandatory requirements, and should for desirable requirements.
- Use text **highlighting** to identify key parts of the requirement.
- Avoid the use of computer jargon. (Provide glossary.)
- Include an explanation (rationale) of why a requirement is necessary.

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.)*

Ways of Writing a System Req. Spec.

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want, and they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

Figure 4.11 Notations for writing system requirements

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 postconditions (if appropriate)
- The side effects (if any) of the function

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.

Form-Based Specifications (cont.)

- 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 postconditions (if appropriate)
- The side effects (if any) of the function

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.

Tabular Specification

- Used to supplement natural language.
- Particularly useful when you have to define a number of possible alternative courses of action.
  - The following example shows that the insulin pump system 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.

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 & ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Structured Specifications

- The freedom of the requirements writer is limited and requirements are written in a standard way (using standard templates).

- 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.

Ways of Writing a System Req. Spec.

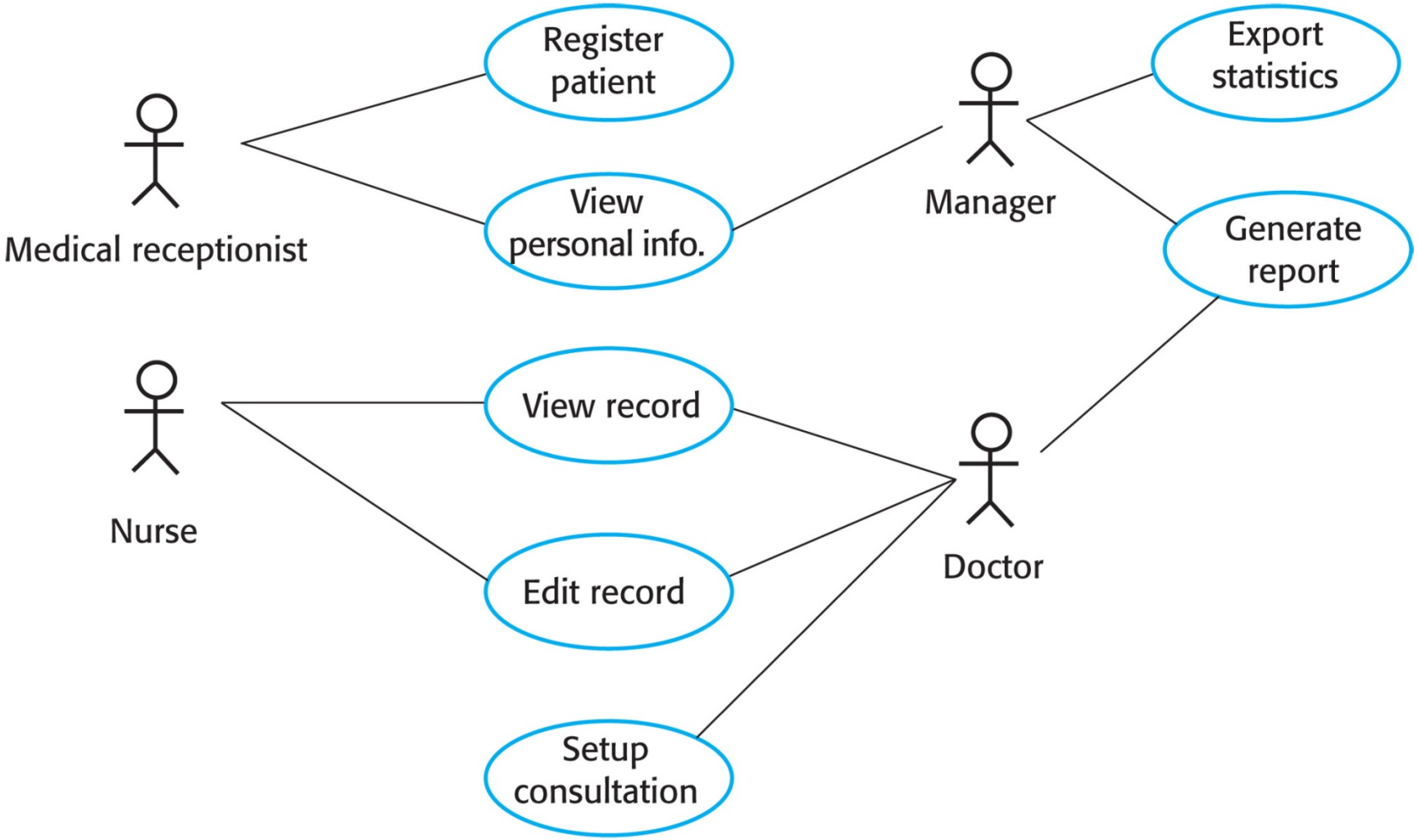
Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want, and they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

Figure 4.11 Notations for writing system requirements

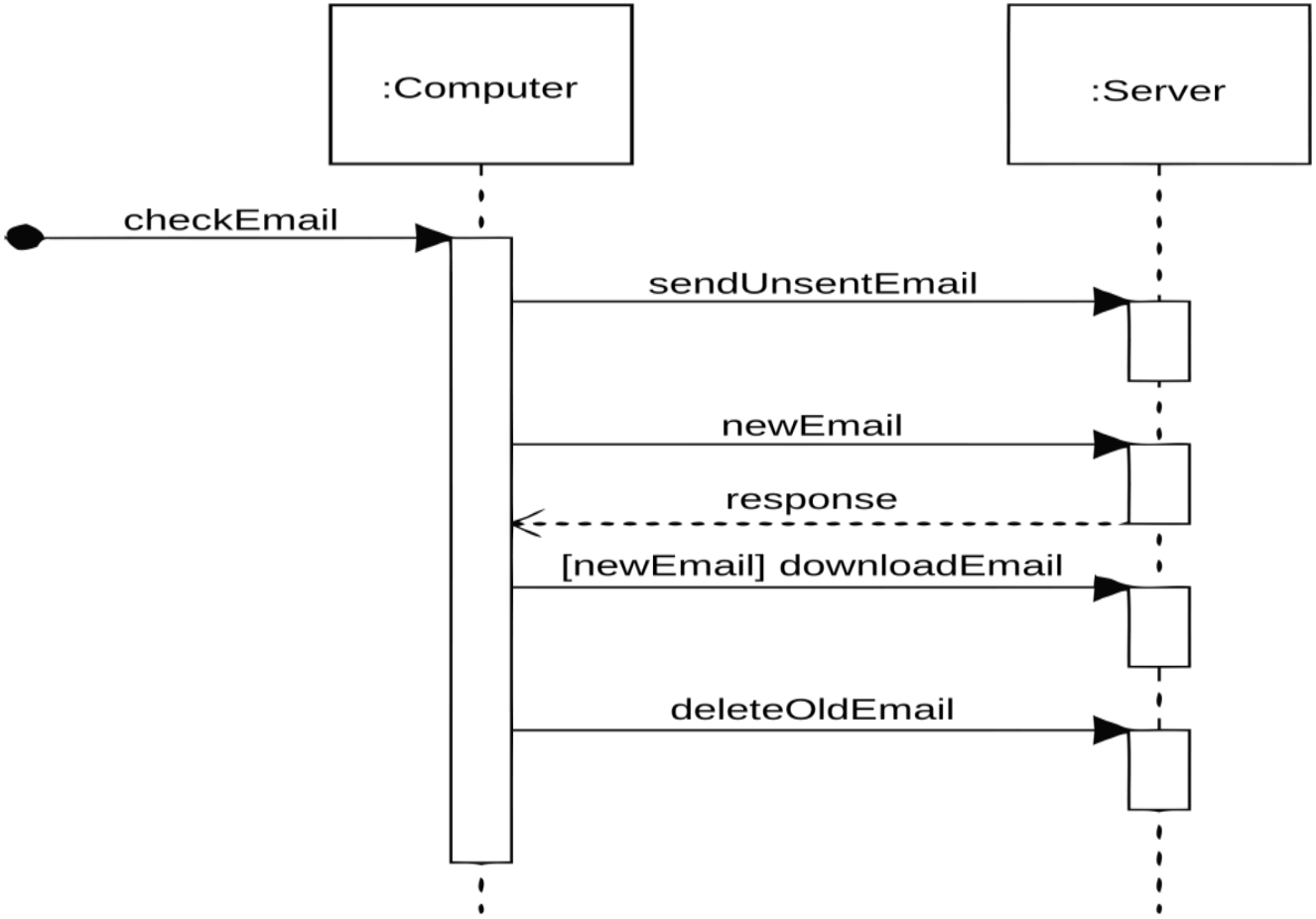
Graphical Notations (UML)

- Use cases are a kind of scenario that are included in the UML.
- Use cases identify the actors in an interaction and 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 (see chapter 5).
- UML sequence diagrams may be used to add detail to use cases by showing the sequence of event processing in the system.

UML: Use Case Diagrams



UML: Sequence Diagrams



Ways of Writing a System Req. Spec.

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want, and they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE

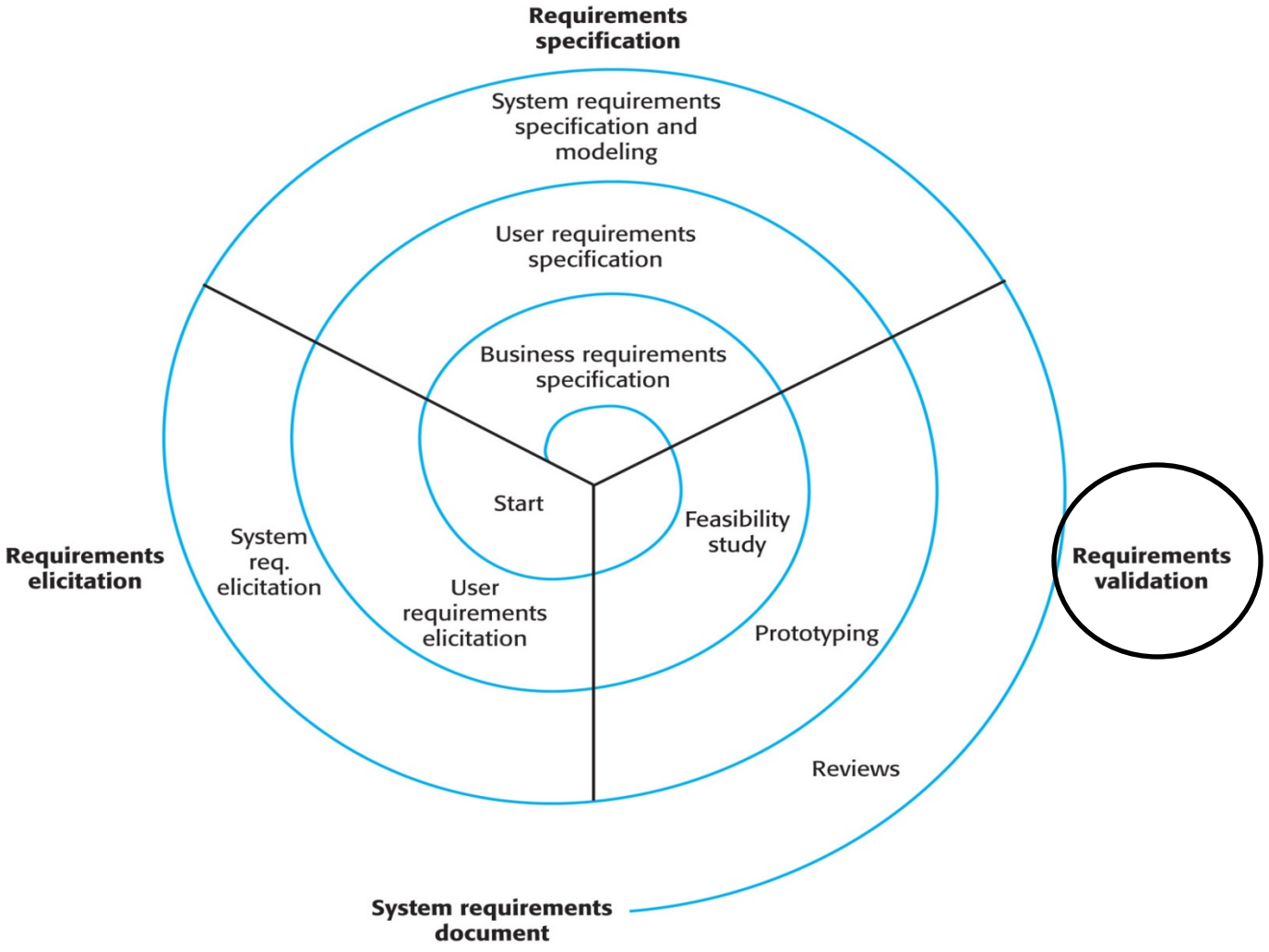
Requirements Validation

Week 3: Requirements Engineering

Edmund Yu, PhD
Associate Professor
esyu@syr.edu

SYRACUSE UNIVERSITY
ENGINEERING
& COMPUTER
SCIENCE

Requirements Engineering Processes



Requirements Validation

- Is the process of checking that requirements actually define the system that the customer really wants.
- It overlaps with analysis in that it is concerned with finding problems with the requirements.
- Requirements error costs are high so validation is very important.
  - The cost of fixing a requirements problem by making a system change is usually much greater (up to **100 times**) than repairing design or coding errors.
  - The reason: A change to the requirements usually means that the system design and implementation must also be changed, and the system must then be retested.

Types of Checks

- **Validity checks**

  A user may think that a system is needed to perform certain functions. Further analysis may identify different functions that are required.

- **Consistency checks**

  Requirements in the document should not conflict.

- **Completeness checks**

  The requirements document should include requirements that define all functions and the constraints intended by the system user.

- **Realism/reality checks**

  The requirements should be checked to ensure that they can actually be implemented. These checks should also consider the budget and schedule.

- **Verifiability**

  To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable.

Requirements Validation Techniques

- **Prototyping**
  - Using an executable model of the system to demonstrated to users/customers to see if it meets their real needs. Covered in chapter 2.
- **Test-case generation**
  - Requirements should be testable.
  - Tests should be developed to check testability.
  - Developing tests from the user requirements before any code is written is an integral part of extreme programming.
- **Requirements reviews**
  - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.

Requirements Validation Techniques

- **Prototyping**
  - Using an executable model of the system to demonstrated to users/customers to see if it meets their real needs. Covered in chapter 2.
- **Test-case generation**
  - Requirements should be testable.
  - Tests should be developed to check testability.
  - Developing tests from the user requirements before any code is written is an integral part of extreme programming.
- **Requirements reviews**
  - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.

Requirements Validation Techniques

- **Prototyping**
  - Using an executable model of the system to demonstrated to users/customers to see if it meets their real needs. Covered in chapter 2.
- **Test-case generation**
  - Requirements should be testable.
  - Tests should be developed to check testability.
  - Developing tests from the user requirements before any code is written is an integral part of extreme programming.
- **Requirements reviews**
  - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.

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.

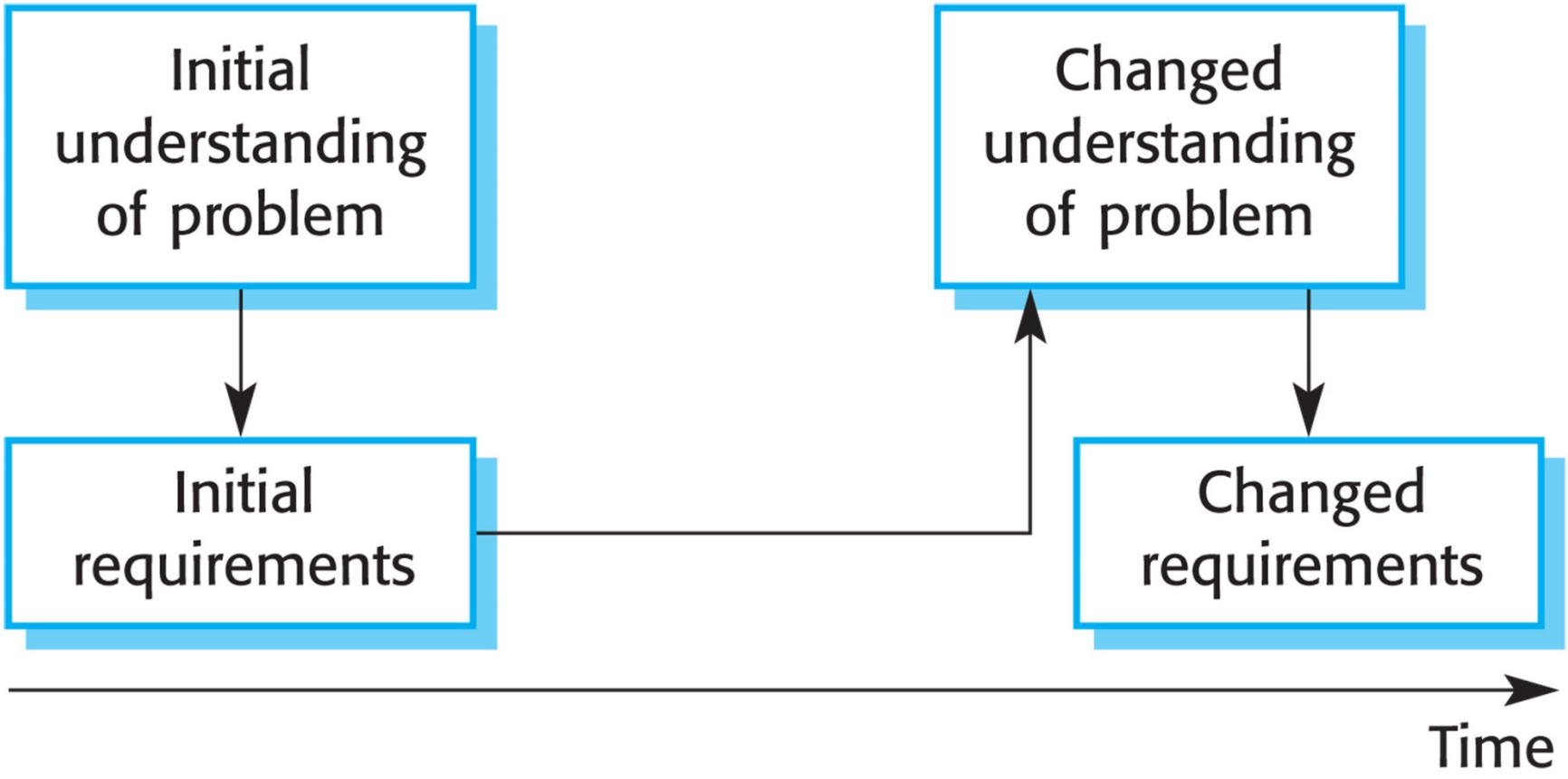
Requirements Management

- Requirements management is the process of managing **changing requirements** during the requirements engineering process and system development.

Changing Requirements

- The business and technical environment of the system always changes after installation.
- The people who pay for a system and the users of that system are rarely the same people.
  - Paying customers impose requirements because of organizational and budgetary constraints.
  - These may conflict with end user requirements
    - Hence, after delivery, new features may have to be added for user support.
- Large systems usually have a diverse user community, with conflicting requirements.
  - Hence, the final system requirements are inevitably a compromise.

Requirements Evolution



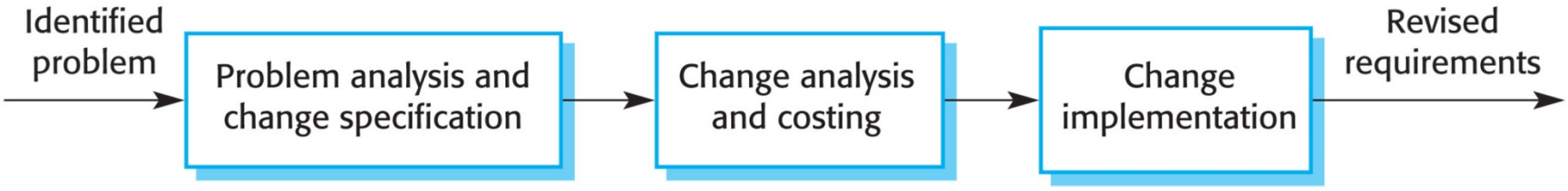
Requirements Change Management

- Deciding if a requirements change should be accepted

- 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.

Requirements Change Management

- Deciding if a requirements change should be accepted



- 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.

Requirements Change Management

- Deciding if a requirements change should be accepted

- 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.

SYRACUSE UNIVERSITY
ENGINEERING & COMPUTER SCIENCE