322 371 Software Engineering

Chapter 05: Requirements Engineering

By

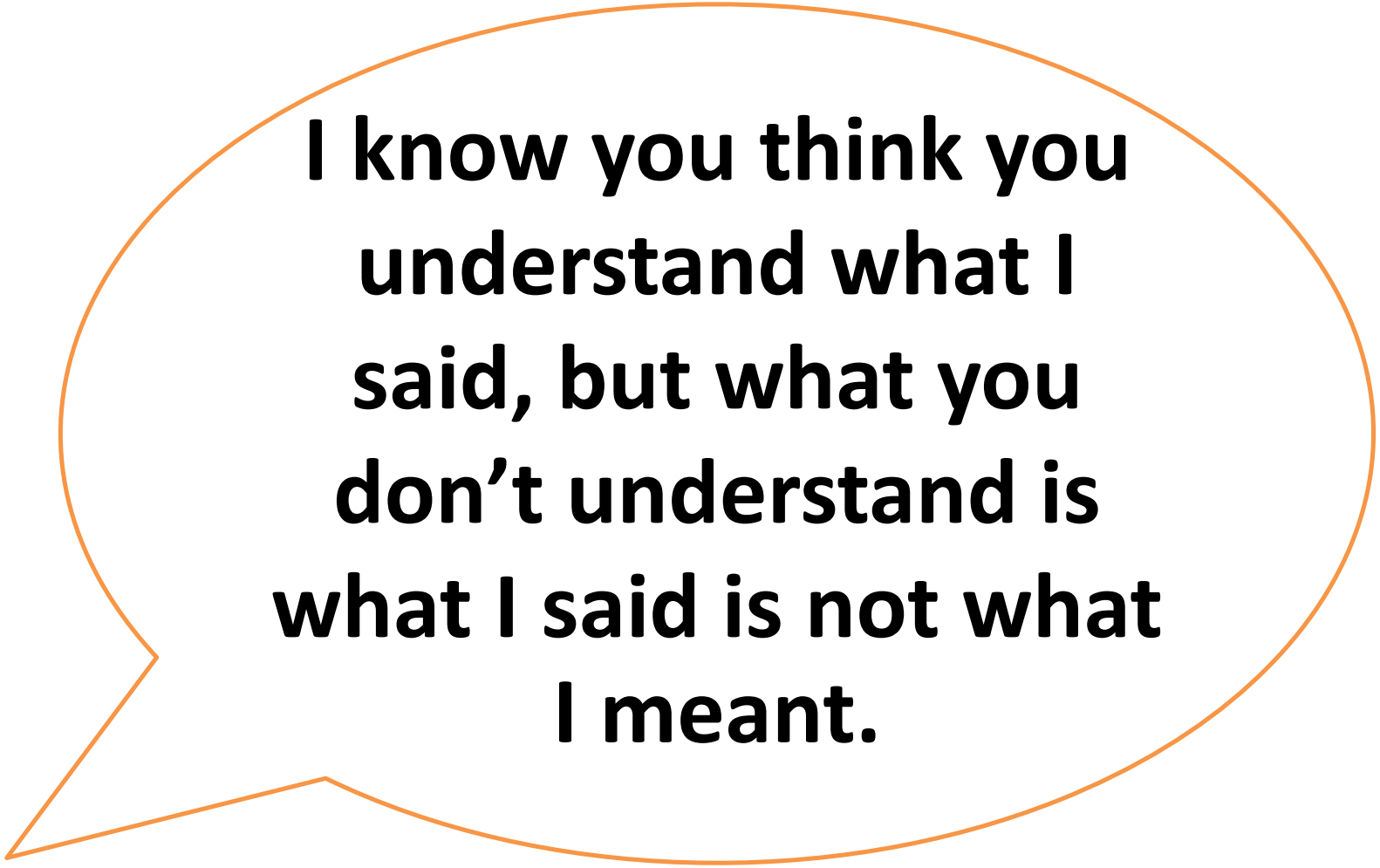
Chitsutha Soomlek, Ph.D.

Department of Computer Science

Faculty of Science, KKU

2/2560

February 5-6, 2018


I know you think you understand what I said, but what you don’t understand is what I said is not what I meant.

คำอธิบายทุกสิ่งอย่างที่เกี่ยวกับตัวระบบ ระบบทำอะไร ระบบทำงานแบบไหน รวมไปถึงวิธีการใช้งาน
ไม่ได้พูดถึงแค่ว่าระบบมีฟังก์ชันอะไร

What are requirements?

วัตถุประสงค์ของผู้ใช้

- The descriptions of what the system should do, the services that it provides, and the constraints on its operation
- The needs of customers for a system that serves a certain purpose
- Detailed and formal definition of a system function
- The intended functionality of a system

Sources of Requirements

ได้จากลูกค้า แต่บางครั้งไม่ได้ทำใช้เอง แต่ทำไปขายคนอื่น ลูกค้าจึงอาจไม่ใช่คนที่รู้ที่สุดก็ได้

- **Stakeholders** - All of the people with an interest in an application's outcome
  - Customers สมมติทำแอพเด็กตาบอด คุณครูที่สอนคือ Stakeholders
  - People paying for an application
  - People who will be using the application
  - People designated to work out the requirements
- **Written materials** - documents and books
- **Others** - market research, conversations

จากลูกค้า จากเอกสารแล้วยังไม่พอ อาจจำเป็นต้องทำ Market reserarch , Survey.

High-level vs. Detailed Requirements

เจาะ ละเอียด เหมาะกับทีม Dev

High-level requirements

- Business requirements
- An overview of the system
- An idea of what the application is all about
- Well suited to customers to read

Detailed requirements

- The complete particulars
- Useful for developers, who need to know precisely what they have to build
- Understandable to the customer

High level ให้ภาพรวมเกี่ยวกับระบบ บอกถึงระบบกว้างๆ ถ้าพูดถึงแล้วคุณจะรู้ว่าระบบทำอะไรได้บ้าง คนทั่วไปอ่านเข้าใจ

Examples of high-level requirements

- The Video Store application shall enable clerks to check DVDs in and out.
  ร้านเช่าดีวีดี - พนักงานสามารถเช็คอิน เช็คเอาต์ดีวีดีได้ //ให้ยืม รับคืนได้
- The following shows a sketch of the main user interface: . . . มีภาพประกอบให้ดู แต่ไม่ได้บอกว่าทำงานยังไง

Examples of detailed requirements

ถ้ายืมแล้วมาคืนเลท วิธีคิดค่าปรับคือ..... และคิดค่าปรับไม่เกิน DVD ที่อยู่ในหมวด ..... และเพิ่มไปอีก 5$

- The daily late charge on a DVD shall be computed at half the regular two - day rental rate, up to the value of the DVD listed in the "Intergalactic Video Catalog." When the amount owed reaches this value, the total late charge is computed as this amount plus $5.

- When the "commit" button is pressed on GUI 37, the GUI shall disappear and GUI 15 shall appear with a superimposed green border (RGB = 6, 32 , 8) and the name and address fields filled with the data for the customer.

เมื่อกด commit บน GUI37 จะต้องหายไป GUI15 จะต้องขึ้นมาแทน และขึ้นสีเขียว ละชื่อที่อยู่ต้องมีด้วย

User requirements vs. System requirements

คือรีไควเม้นที่เกี่ยวข้องกับผู้ใช้ อธิบายง่ายๆ ลูกค้าสามารถเข้าใจได้ อาจจะภาพ

User requirements

- The high-level abstract requirements
- Statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate

(Functional Specifinactiion)

System requirements

- The detailed description of what the system should do
- More detailed descriptions of the software system's functions, services, and operational constraints
- The system requirements document (so called *functional specification*) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers

คือสิ่งีที่นอกจากระบบจะทำอะไรได้บ้าง แต่ต้องบอกว่าระบบมีฟังก์ชัน มีเซอวิส มีเงื่อนไข ในการทำงานยังไง ในเชิงเทคนิค ดาต้าเบสเป็นอะไร เก็บยังไง

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

แต่ละสิ้นเดือนต้องมีการสรุปการจ่ายยา

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.

ต้องเจนรีพอร์ตออกมาอัตโนมัติหลัง 17.30 วันสิ้นเดือน

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.

Text จำนวนยาที่จ่ายออกไปคิดเป็นมูลค่าเท่าไห

1.4 If drugs are available in different dose units (e.g., 10 mg, 20 mg) separate reports shall be created for each dose unit. ถ้ายาเดียวกันปริมาณไม่เท่ากันต้องเขียนแยก

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

Figure from I. Sommerville, Software Engineering, 9th ed.

Readers

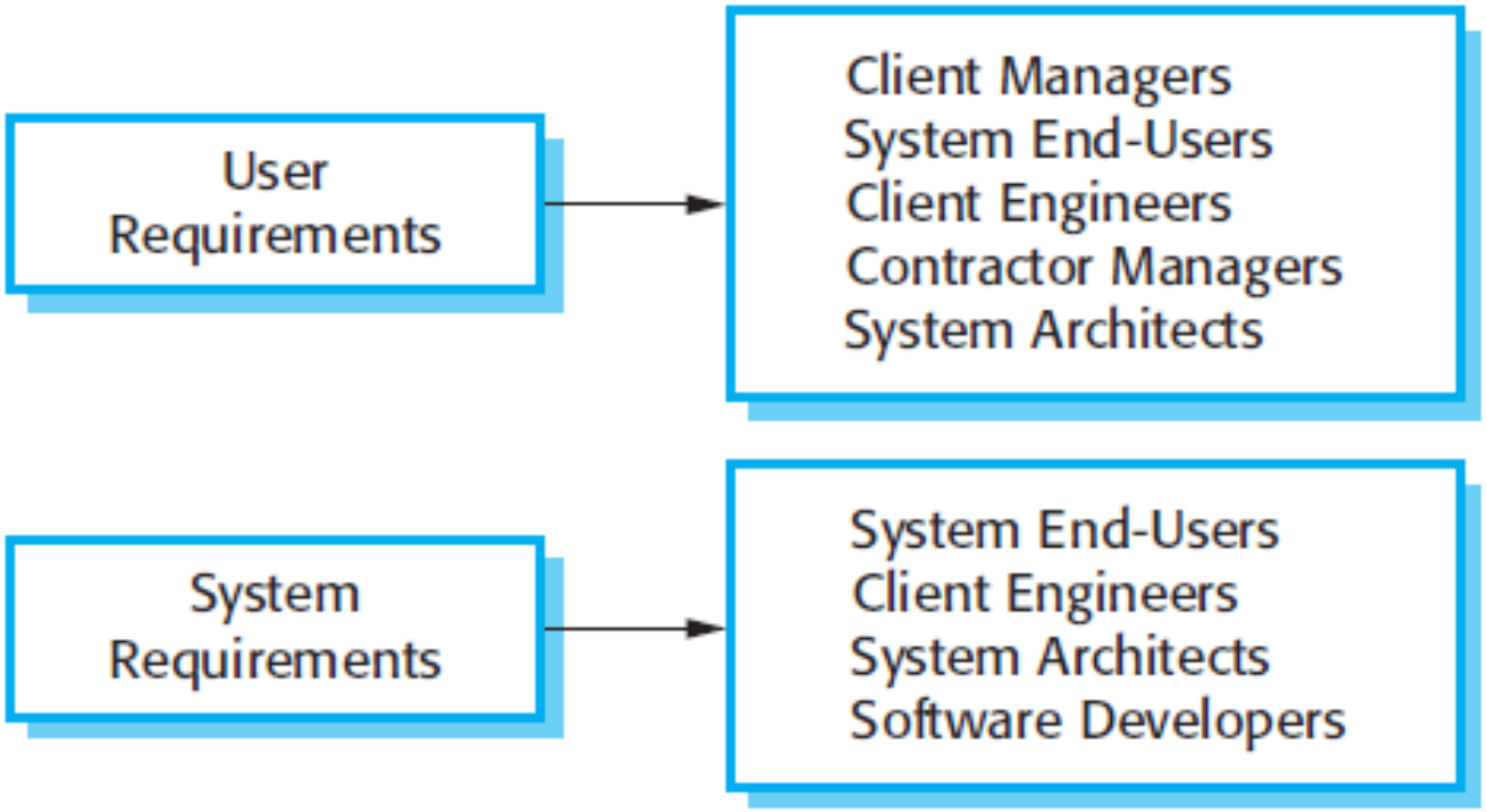


Figure from I. Sommerville, Software Engineering, 9th ed.

Functional vs. Non-functional requirements

ไม่ได้เป็นฟังชั่นการทำงาน แต่มีผลกับระบบ
สำคัญกว่า functional เพราะทำได้ยากและส่งผลต่อตัวระบบโดยตรง

Functional requirements

- *Behavioral requirements*
- Describe functionalities or system services.
- 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.

ระบบทำอะไรได้บ้าง
ทำอะไรไม่ได้บ้าง
ในส่วนของการทำงาน
ควรทำอะไรได้
ควรทำอะไรไม่ได้

Non-functional requirements

- Requirements that are not directly concerned with the specific services delivered by the system to its users.
- 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.

Functional vs. Non-functional requirements

- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.
- Non-functional requirements may affect the overall architecture of a system rather than the individual 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.

Examples

- Examples of functional requirements for the MHC-PMS system, used to maintain information about patients receiving treatment for mental health problems:

  1. A user shall be able to search the appointments lists for all clinics. ไม่ชัดเจน /ใครเสิจ/ใช้อะไรเสิจ

  2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day. เจนว่าคนไข้คนไหนที่จะมาบ้างวันนี้

  3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number. ถ้าต้องการจะใช้ระบบจะต้องมีการ indentify ตัวเองโดยใช้เลขพนักงานแปดหลัก

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
  - **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, the functional requirements specification of a system should be both complete and consistent.
- **Completeness** - all services required by the user should be defined. เก็บรีไควเม้นได้ครบถ้วน
- **Consistency** - There should be no conflicts or contradictions in the descriptions of the system facilities. เกิดการขัดแย้งกัน
- In practice, it is impossible to produce a complete and consistent requirements document.

Types of Non-functional Requirements

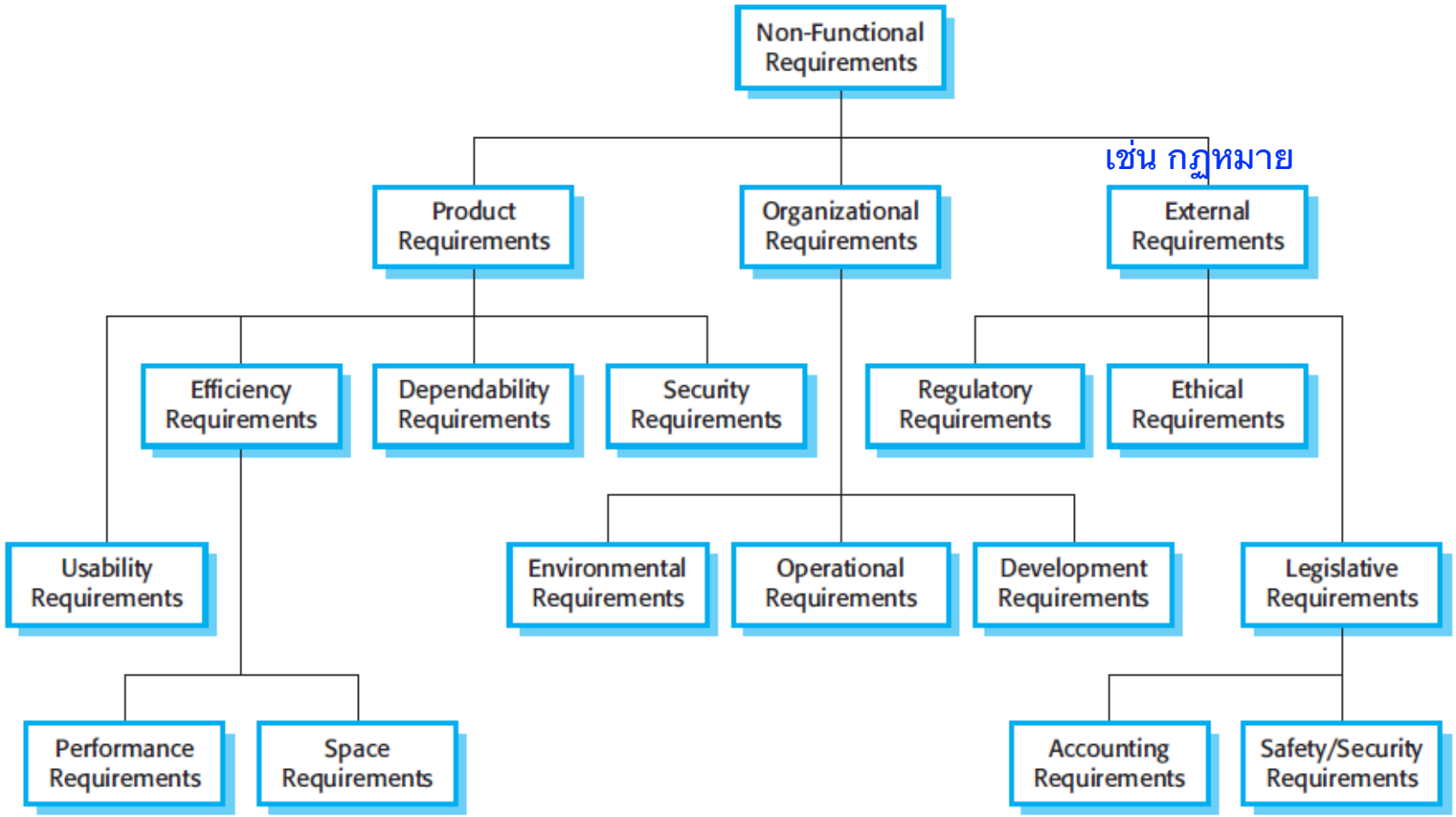


Figure from I. Sommerville, Software Engineering, 9th ed.

Examples

Product requirement
The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 08.30–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement รีไควเม้นที่เกิดจากองค์กร
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.

Figure from I. Sommerville, Software Engineering, 9th ed.

Goals and Requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Example:

Text
ระบบควรจะใช้งานง่าย และ ควรจะจัดการให้ลด user error ให้เหลือน้อยที่สุด

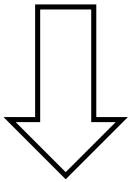
The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.

Goals and Requirements

- **Goals**
  - Goals are helpful to developers as they convey the intentions of the system users.
  - From the example -> A general intention of the user such as ease of use. Text
- **Verifiable non-functional requirement**
  - A statement using some measure that can be objectively tested.
  - Whenever possible, you should write non-functional requirements quantitatively so that they can be objectively tested.

Goals and 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)



ปรับเปลี่ยนให้เข้าใจง่ายขึ้น โดยเขียนวิธีการให้ชัดเจน

"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)

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 ความทนทาน	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability หลาย Platform	Percentage of target dependent statements Number of target systems

โอกาสที่จะเฟลเท่าไหร่
data corruption เป็นเท่าไหร่

ระบุให้ชัดเจนว่ารันบนอะไรได้บ้าง

Figure from I. Sommerville, Software Engineering, 9th ed.

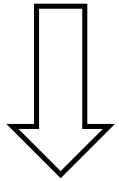
Examples

Functional requirements

- The application allows clerks to check out DVDs.
- The application allows clerks to display the customer's account status.

Non-functional requirements

- The system shall retrieve user information quickly.



- Once the OK button is pressed on the "Retrieve Account Information" screen, the user's account information shall be displayed in less than 3 seconds.

What is Requirements Engineering?

- The tasks lead to an understanding of
  - what the business impact of the software will be
  - what the customer wants
  - how end users will interact with the software
- The process of finding out, analyzing, documenting, and checking these services and constraints
- The process of providing all parties with a written understanding of the problem
- What are the priorities, what is essential, when it is required?

Products of Requirements Engineering

- Usage scenarios
- List of functions
- List of features
- Requirement models
- Specification
- User stories
- Etc.

What are the steps?

กำหนดขอบเขตงานที่จะทำ

- **Requirements Elicitation** – defines the scope and nature of the problem to be solved and helps stakeholders define what is required.
- **Requirements Analysis** – basic requirements are refined and modified.
- **Requirements Validation** – specified problems are reviewed and validated. ตรวจสอบกับลูกค้าว่าตรงตามความต้องการของลูกค้าหรือไม่
- **Requirements Management** – a process of managing changing requirements during the requirements engineering process and system development เก็บรักษารีไควเม้นให้เป็นเวอร์ชั่น เพราะซอฟแวต้องทำไปเรื่อยๆจนกว่าจะประกาศว่าเลิกใช้แล้ว จะมีการเปลี่ยนแปลงเข้ามาเรื่อยๆ

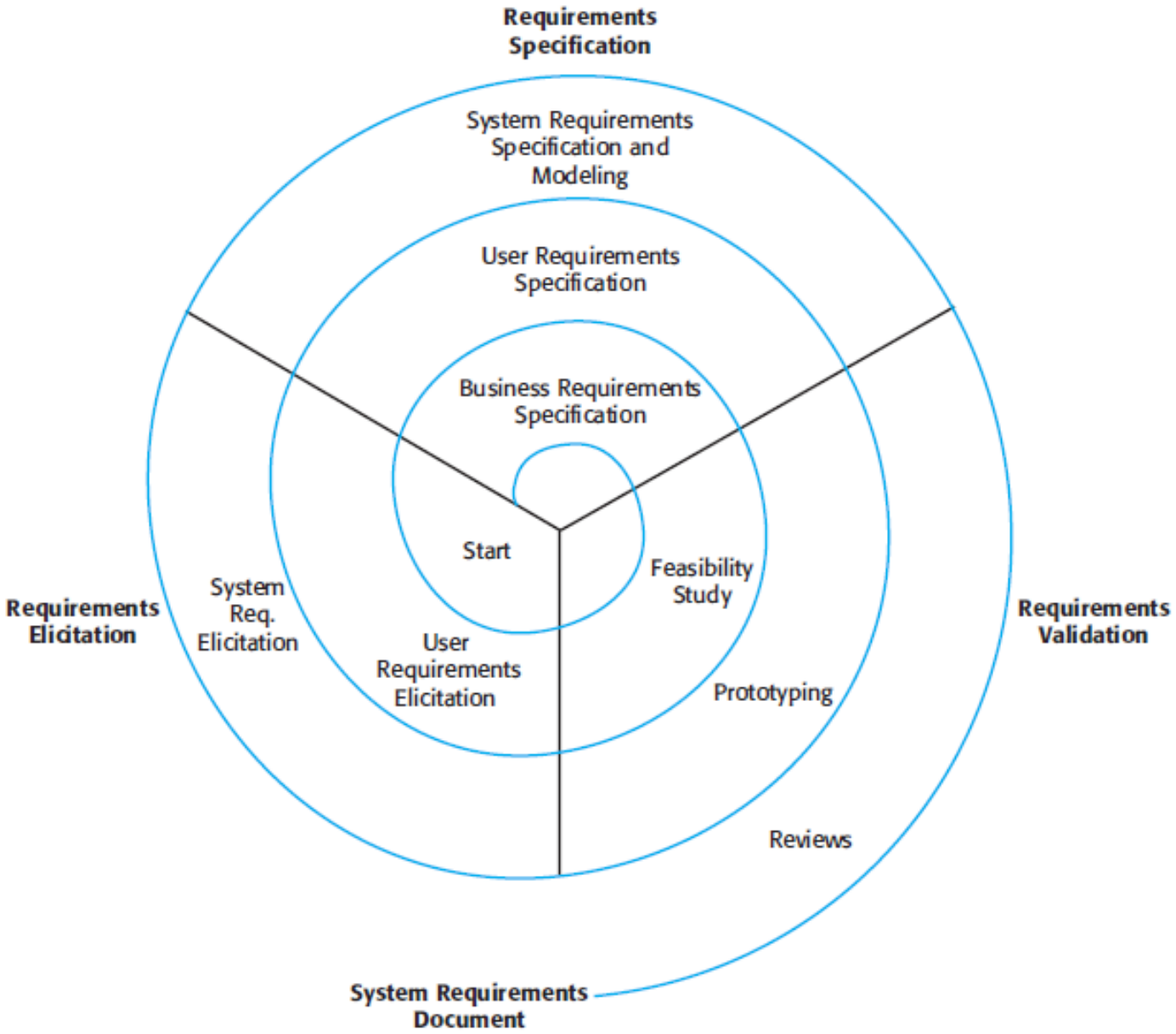


Figure from I. Sommerville, Software Engineering, 9th ed.

ควรจะทำก่อนจะเริ่มโปรเจคใดๆก็ตาม

Feasibility Studies

- A feasibility study should take place early in the RE process.
- It should answer three key questions: ต้องตอบสามข้อนี้ได้
  a) does the system contribute to the overall objectives of the organization?
  b) can the system be implemented within schedule and budget using current technology? and
  c) can the system be integrated with other systems that are used? สิ่งที่จะทำ เอาไปใช้กับระบบอื่นได้หรือไม่
- If the answer to any of these questions is no, you should probably not go ahead with the project.

Requirements Elicitation and Analysis Process

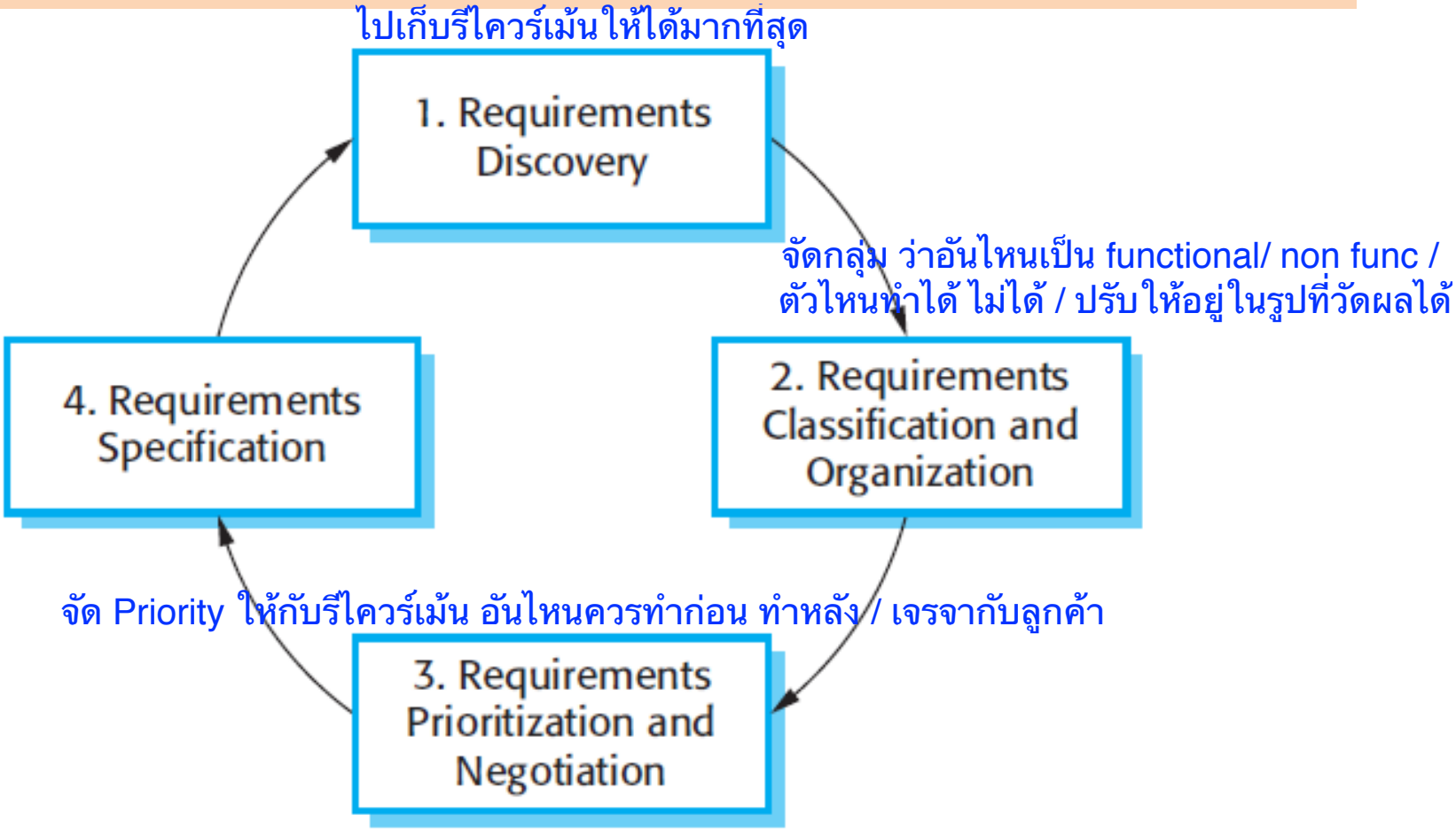


Figure from I. Sommerville, Software Engineering, 9th ed.

Requirements “Analysis"

- The process of gaining the necessary understanding of the requirements.
- The process of understanding what's wanted and needed in an application.
- We express requirements in writing to complete our understanding and to create a contract between developer and customer.
- Output = Software requirements specification (SRS)

Techniques Used in Requirement Discovery

- Interview สัมภาษณ์
- Document review รีวิวเอกสาร
- Ethnography/Observation การสังเกตการใช้งานในการทำงานจริงของยูซเซอ
- Survey and questionnaire
- Sampling
- Research
- etc.

Note: you may 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 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.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Ethnography/Observation

- Observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.
- Ethnography can be combine with prototyping.

Scope of Ethnography

- Requirements that are derived from the way that people actually work rather than the way which process definitions suggest that they ought to work.
- Requirements that 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.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

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 and how this is handled;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.
- The stories used in extreme programming are a type of requirements scenario.

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).

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.

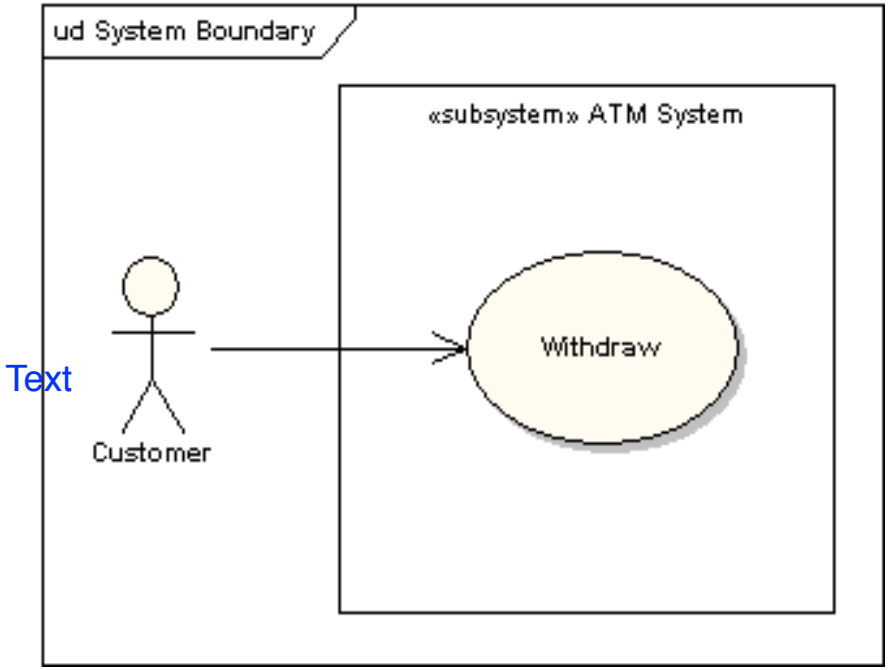
SYSTEM STATE ON COMPLETION:
User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Use Case Diagram

- Use cases are a requirements discovery technique that were first introduced in the Objectory method.
- 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.
- Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use Case Diagram

- Actor
- Use case
- Relationships
  - Association
  - Use case inclusion
  - Actor generalization
  - Use case extension
- System boundary



Use Case Description

- General comments and notes
- Requirements
- Constraints
  - Pre-conditions
  - Post-conditions
  - Invariants
- Scenarios
- Scenario diagrams
- Additional attributes

How to draw a use case diagram

Step1: Discover the actors

Step2: Discover the use cases

Step3: Discover the relationships

Step4: Draw the use case diagram

User Stories

- Combined with conversations and acceptance criteria
- Focus on business and customer value (the things real people are going to need the software to do for them)
- Often written by hand on index cards or sticky notes
- Annotate with notes, pictures, acceptance criteria, dependencies, and exceptions
- Details come from conversations with customers, stakeholders, end user, or product owner

Template: Focus on the user

As a <type of user>

I want <to do something>

So that <some value is created>

Example

As a member of the One Direction Fan Club,

I want to order concert tickets by phone before they go on sale to the general public

So that I can get great seats and feel special.

Template: Focus on the goal

In order to <achieve some goal>

As a <type of user>

I want <to do something>

Template: Focus on the value

In order to <create some value>

As a <type of user>

I want <to do something>

เราจะรู้ได้ยังไงว่าตรงตามโจทย์ เราต้องร่าง Accceptance criteria
สร้าง list ที่บอกว่าส่งงานแล้วผ่าน หรือไม่ผ่านเป็นยังไง

Acceptance Criteria

- Clarification of the story
- A list of pass/fail tests, written in plain English, such that if they all pass, then everyone involved in the conversation would agree that the story is implemented as intended.
- “How will we know when it does what it should do?”
- Guide for (automated) acceptance tests
- Documentation

Examples

Cancel Reservation
As a traveler, I want to cancel all of my bookings easily, without having to cancel one at a time

Acceptance Criteria
- Cancel > 24 hrs before the booking date
- Charge 10% fee
- Ask “Are you sure that you want to cancel?”
- Display confirmation of what has been canceled
- Process cancellation within 4 hrs
- Email the confirmation

ATDD

- Acceptance Test-Driven Development
- Start from a user story
- Write acceptance criteria for the story
- Write automated test based on the acceptance criteria
- Implement the code

Case Study

- https://youtu.be/szr0ezLyQHY

Software Requirements Document

- Software requirements document = Software requirements specification (SRS)
- It is an official statement of what the system developers should implement.
- It should include both the user requirements for a system and a detailed 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

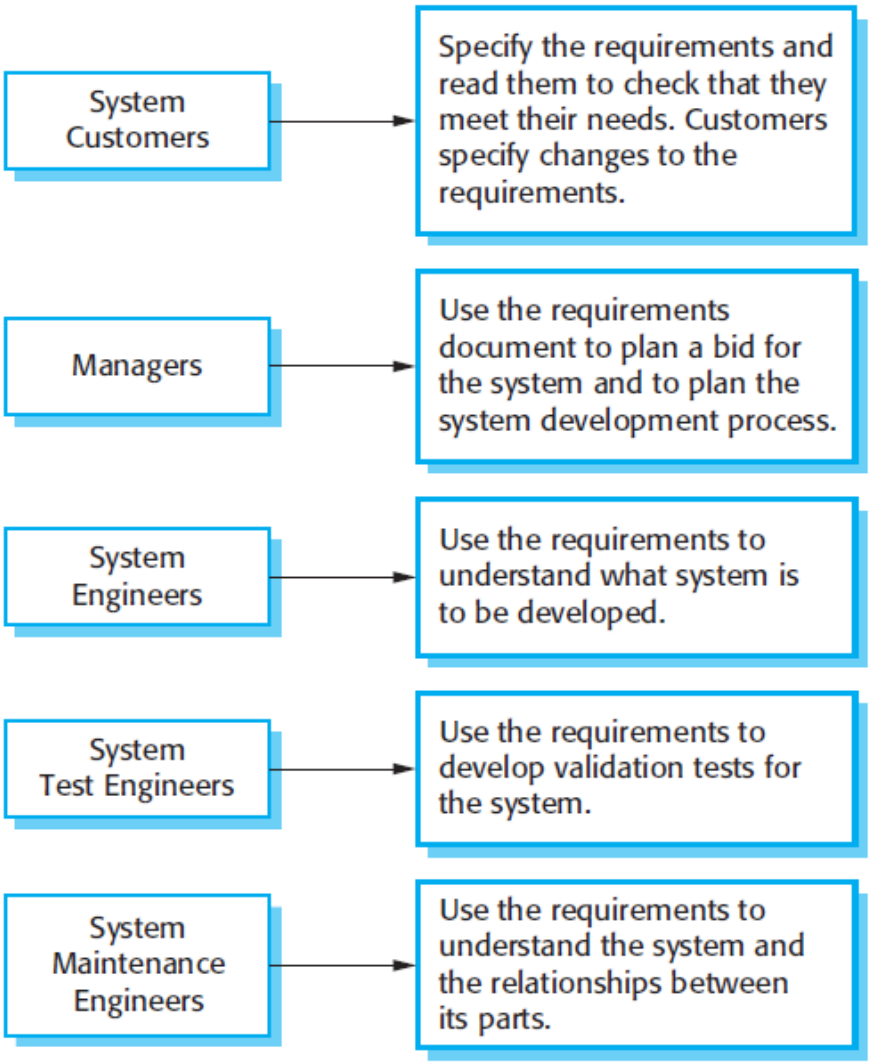


Figure from I. Sommerville, Software Engineering, 9th ed.

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.

Figure from I. Sommerville, Software Engineering, 9th ed.

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.

จะมีการขยายระบบในลักษณะยังไง

ข้อมูลอื่นๆที่ต้องการให้เพิ่มเติม

Figure from I. Sommerville, Software Engineering, 9th ed.

Example of Requirements Specification

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.*)

Figure from I. Sommerville, Software Engineering, 9th ed.

Example of a structured specification of a requirement

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.

Figure from I. Sommerville, Software Engineering, 9th ed.

Example of a Tabular Specification

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

Figure from I. Sommerville, Software Engineering, 9th ed.

The IEEE standard for requirements documents

- IEEE/ANSI 830-1998 (IEEE, 1998)

1. Introduction
- 1.1 Purpose of the requirements document
- 1.2 Scope of the product
- 1.3 Definitions, acronyms and abbreviations
- 1.4 References
- 1.5 Overview of the remainder of the document

2. General description
- 2.1 Product perspective
- 2.2 Product functions
- 2.3 User characteristics
- 2.4 General constraints
- 2.5 Assumptions and dependencies
- 2.6 Apportioning of requirements

3.Specific requirements

4.Appendices

5.Index

Figure from I. Sommerville, Software Engineering, 9th ed.

Requirements Evolution

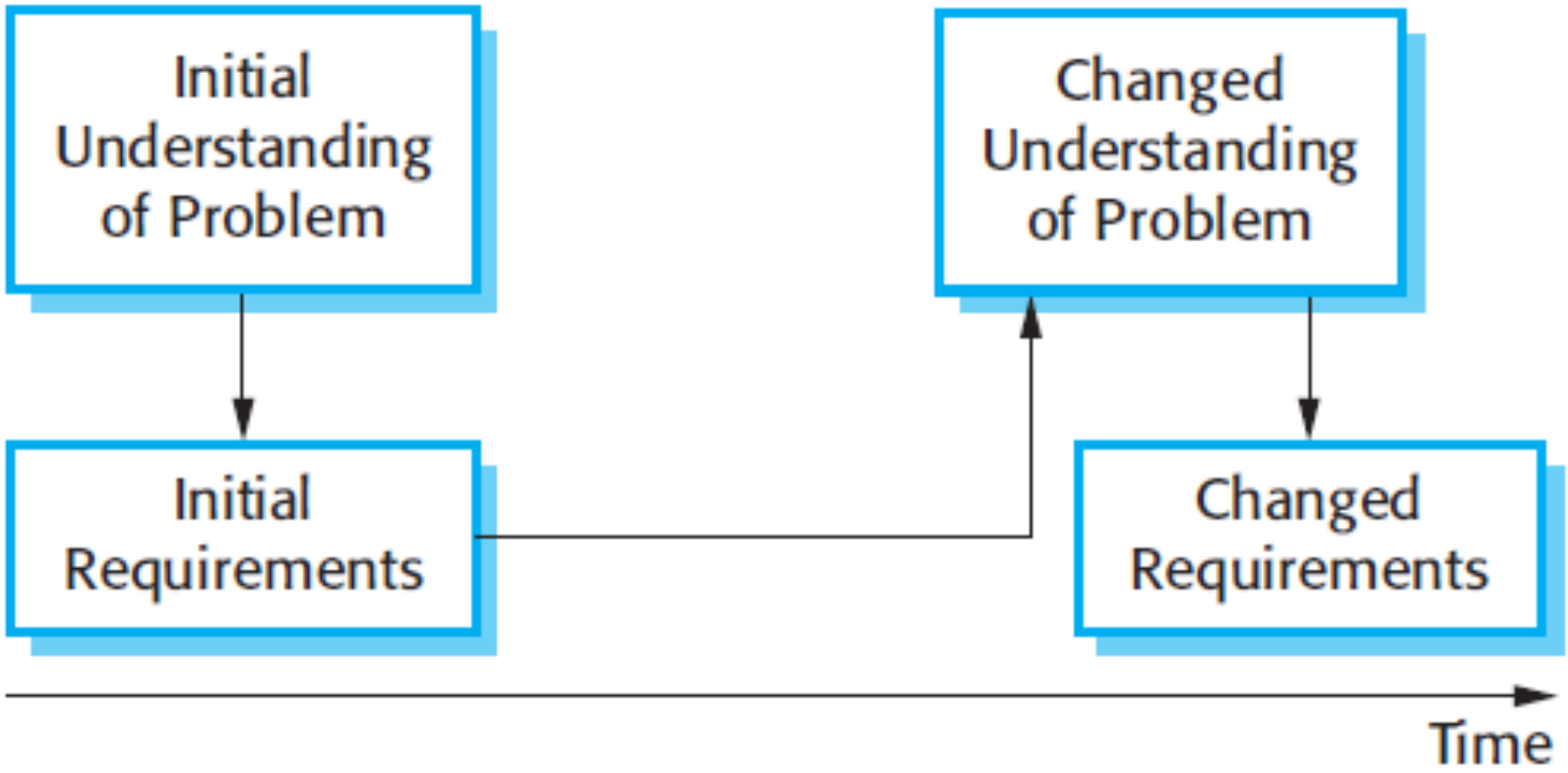


Figure from I. Sommerville, Software Engineering, 9th ed.

Requirements Management

- A process of managing changing requirements during the requirements engineering process and system development
- You need to keep track of individual requirement and maintain links between dependent requirements so that you can assess the impact of requirements changes.
- You need to establish a formal process for making change proposals and linking these to system requirements.

Requirements Management Planning

- 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.
  - *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

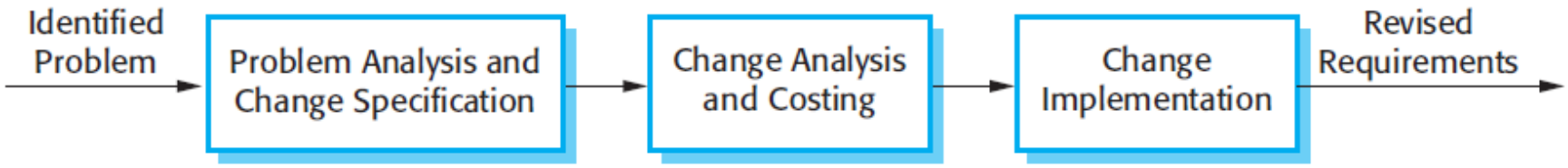


Figure from I. Sommerville, Software Engineering, 9th ed.