Course Lecture Schedule

Maria
Susan
Sami
Hyrynsalmi

Date	Topic	Book Chapter(s)
Wed 8.9.	Course introduction	
Tue 14.9.	Introduction to Software Engineering	Chapter 1
Tue 21.9.	Software Processes	Chapter 2
Mon 27.9	Agile Software Engineering	Chapter 3
Tue 5.10.	**Requirements Engineering**	**Chapter 4**
Mon 11.10.	Architectural Design	Chapter 6
Wed 20.10.	Modeling and implementation	Chapters 5 & 7
Mon 1.11.	Testing & Quality	Chapters 8 & 24
Mon 8.11.	Software Evolution & Configuration Management	Chapters 9 & 25
Mon 15.11.	Software Project Management	Chapter 22
Mon 22.11.	Software Project Planning	Chapter 23
Mon 29.11.	Global Software Engineering	
Wed 8.12.	Software Business	
Mon 13.12.	Last topics	

Lecture5
Requirements Engineering

Topics covered

- ✧ What is requirement engineering（RE）
- ✧ Types of requirement
  - User requirements and system requirements
  - Functional and non-functional requirements
- ✧ Requirements engineering processes
  - Requirements elicitation
  - Requirements specification
  - Requirements validation
  - Requirements change

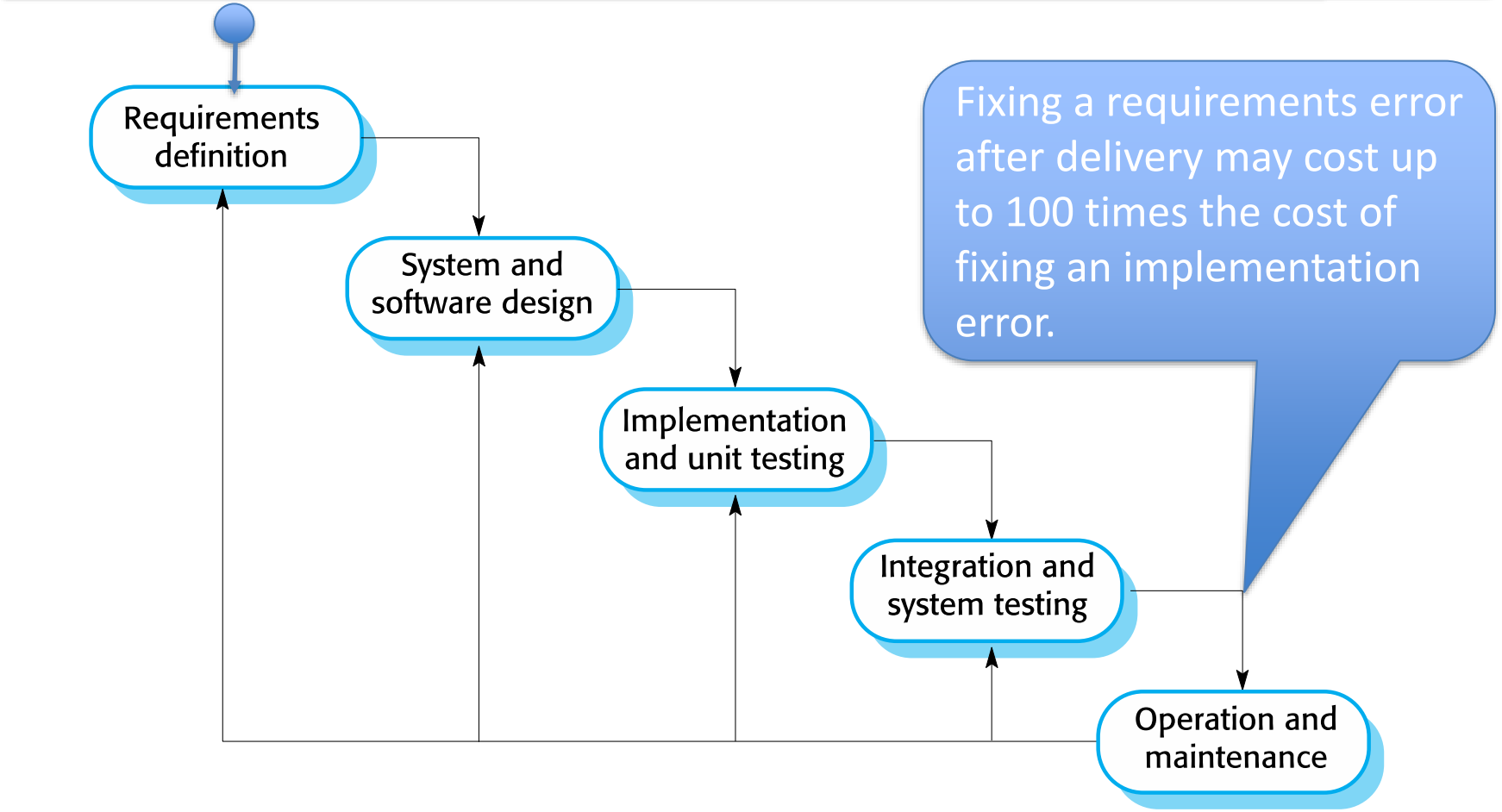
What is requirement engineering（RE）

What is requirements

"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 pre-defined. 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." (-Davis 1993)

- ✧ Abstract definition of the client's needs
- ✧ Detail definition of the system(software)

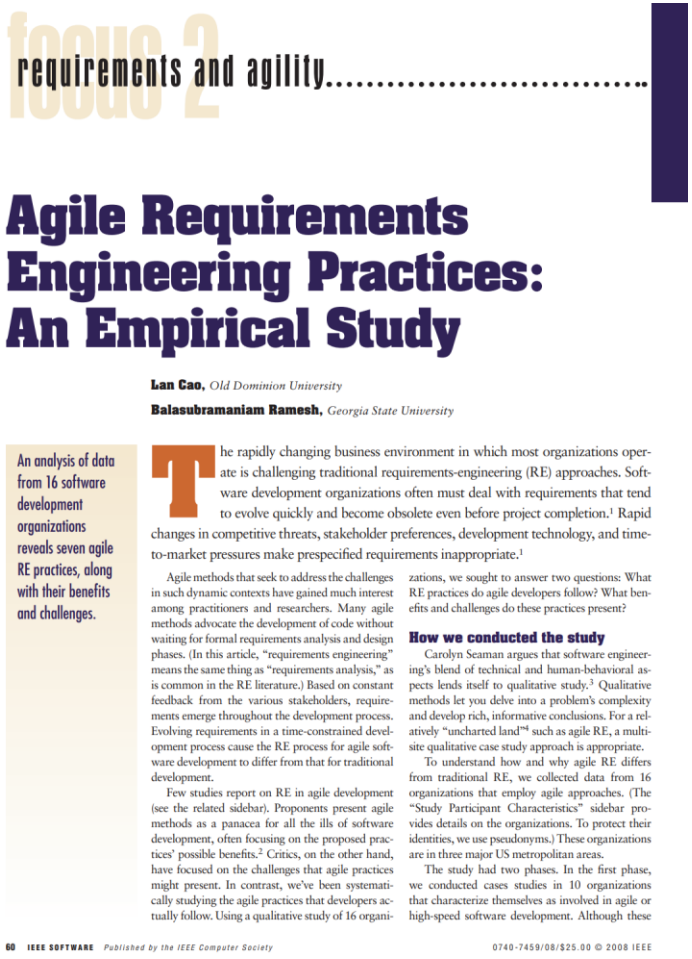
Requirements definition in waterfall model



Requirements definition in agile

Agile development occurs in an environment where requirements are ambiguous and incomplete. Users take part in development assisting in requirements definition.

- Advantage: improved understanding of customer needs and the ability to adapt to the evolving needs of dynamic environment.
- Disadvantage: Agile pose several distinct challenges on requirments.



focus 2

requirements and agility

Agile Requirements Engineering Practices: An Empirical Study

Lan Cao, *Old Dominion University*

Balasubramaniam Ramesh, *Georgia State University*

An analysis of data from 16 software development organizations reveals seven agile RE practices, along with their benefits and challenges.

The rapidly changing business environment in which most organizations operate is challenging traditional requirements-engineering (RE) approaches. Software development organizations often must deal with requirements that tend to evolve quickly and become obsolete even before project completion.[1] Rapid changes in competitive threats, stakeholder preferences, development technology, and time-to-market pressures make prespecified requirements inappropriate.[1]

Agile methods that seek to address the challenges in such dynamic contexts have gained much interest among practitioners and researchers. Many agile methods advocate the development of code without waiting for formal requirements analysis and design phases. (In this article, "requirements engineering" means the same thing as "requirements analysis," as is common in the RE literature.) Based on constant feedback from the various stakeholders, requirements emerge throughout the development process. Evolving requirements in a time-constrained development process cause the RE process for agile software development to differ from that for traditional development.

Few studies report on RE in agile development (see the related sidebar). Proponents present agile methods as a panacea for all the ills of software development, often focusing on the proposed practices' possible benefits.[2] Critics, on the other hand, have focused on the challenges that agile practices might present. In contrast, we've been systematically studying the agile practices that developers actually follow. Using a qualitative study of 16 organizations, we sought to answer two questions: What RE practices do agile developers follow? What benefits and challenges do these practices present?

How we conducted the study

Carolyn Seaman argues that software engineering's blend of technical and human-behavioral aspects lends itself to qualitative study.[3] Qualitative methods let you delve into a problem's complexity and develop rich, informative conclusions. For a relatively "uncharted land"[4] such as agile RE, a multisite qualitative case study approach is appropriate.

To understand how and why agile RE differs from traditional RE, we collected data from 16 organizations that employ agile approaches. (The "Study Participant Characteristics" sidebar provides details on the organizations. To protect their identities, we use pseudonyms.) These organizations are in three major US metropolitan areas.

The study had two phases. In the first phase, we conducted cases studies in 10 organizations that characterize themselves as involved in agile or high-speed software development. Although these

0740-7459/08/$25.00 © 2008 IEEE

Do you think it will be difficult to do requirements definition? Why?

- Blind men touch elephant—radish, fan, wall, pillar, …?
- Different system stakeholders have different viewpoints&concerns;
- Requirements definition should be a comprehensive description of the whole system.

System stakeholders

https://www.youtube.com/watch?v=P5X-ridjaOY&t=7s（8 min，Stakeholders）

- Any person or organization who is affected by the system in some way and so who has a legitimate interest.
- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders
- Different system stakeholders have different requirements on the same system.
- System requirements should be both complete and consistent.

What is requirement engineering

- In practice, because of system and environmental complexity, it is almost impossible to produce a complete and consistent requirements document.
- The requirements definition is very difficult and very complex .
- The process of finding out, analyzing, documenting and checking services and constraints is called requirement engineering(RE).

Types of requirements

-User requirements and system requirements
-Functional and non-functional requirements

User requirements and system requirements

Statements in **natural language plus diagrams** of the services the system provides and its operational constraints.

User requirements → Client managers, System end-users, Client engineers, Contractor managers, System architects

System requirements → System end-users, Client engineers, System architects, Software developers

A **structured document** setting out detailed descriptions of the system's functions, services and operational constraints.
May be part of a contract between client and contractor.

Example: 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.
- 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.

Example: User requirements and system requirements in Mentcare system

User requirements definition

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

Written in natural language, for customers

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.

Detailed description, technical contract

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.
- Non-functional requirements
  - Constraints on the, such as timing constraints, constraints on the development process, standards, etc.

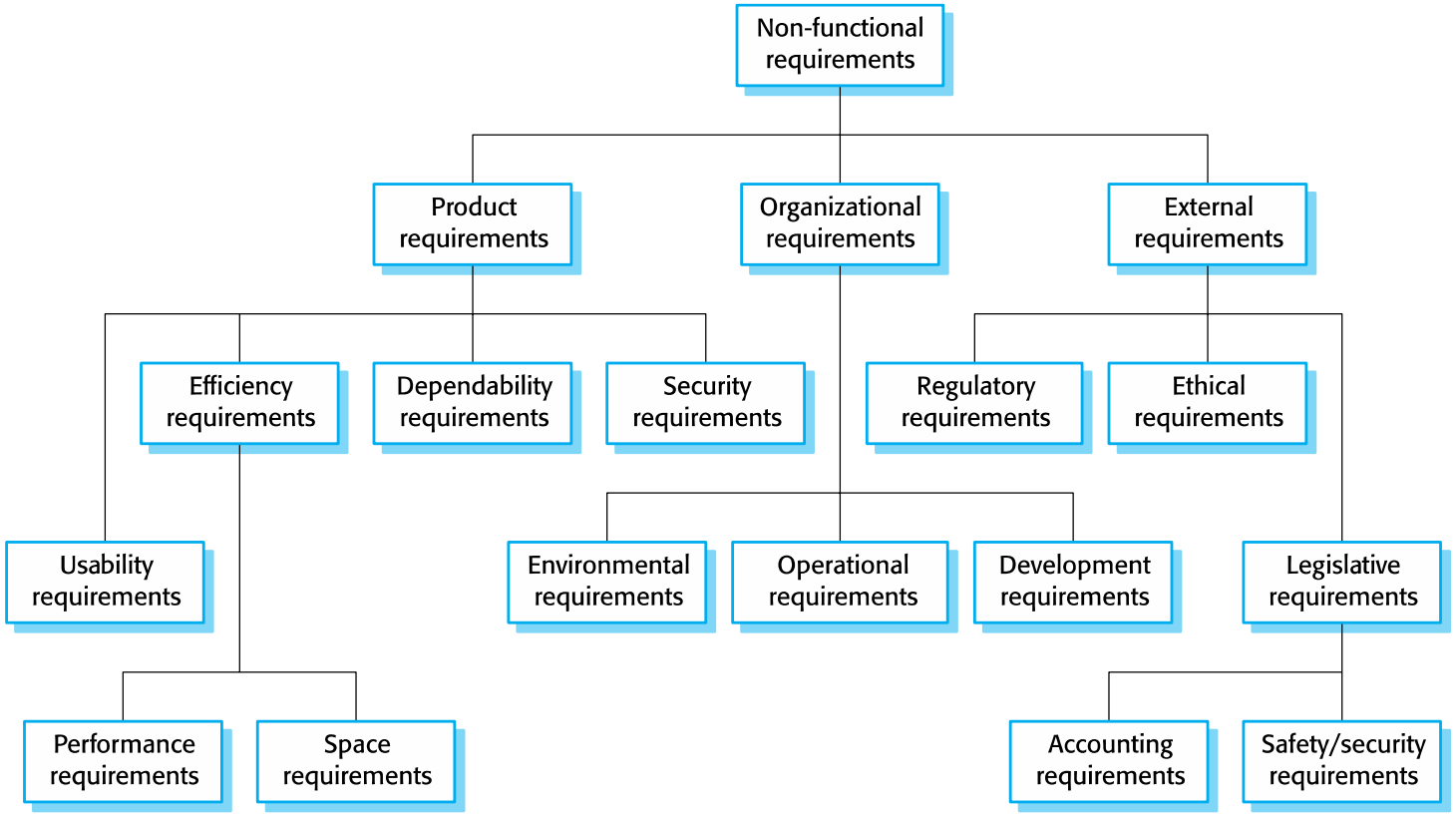
Functional requirements

- 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.

Non-functional requirements

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

Types of nonfunctional requirement

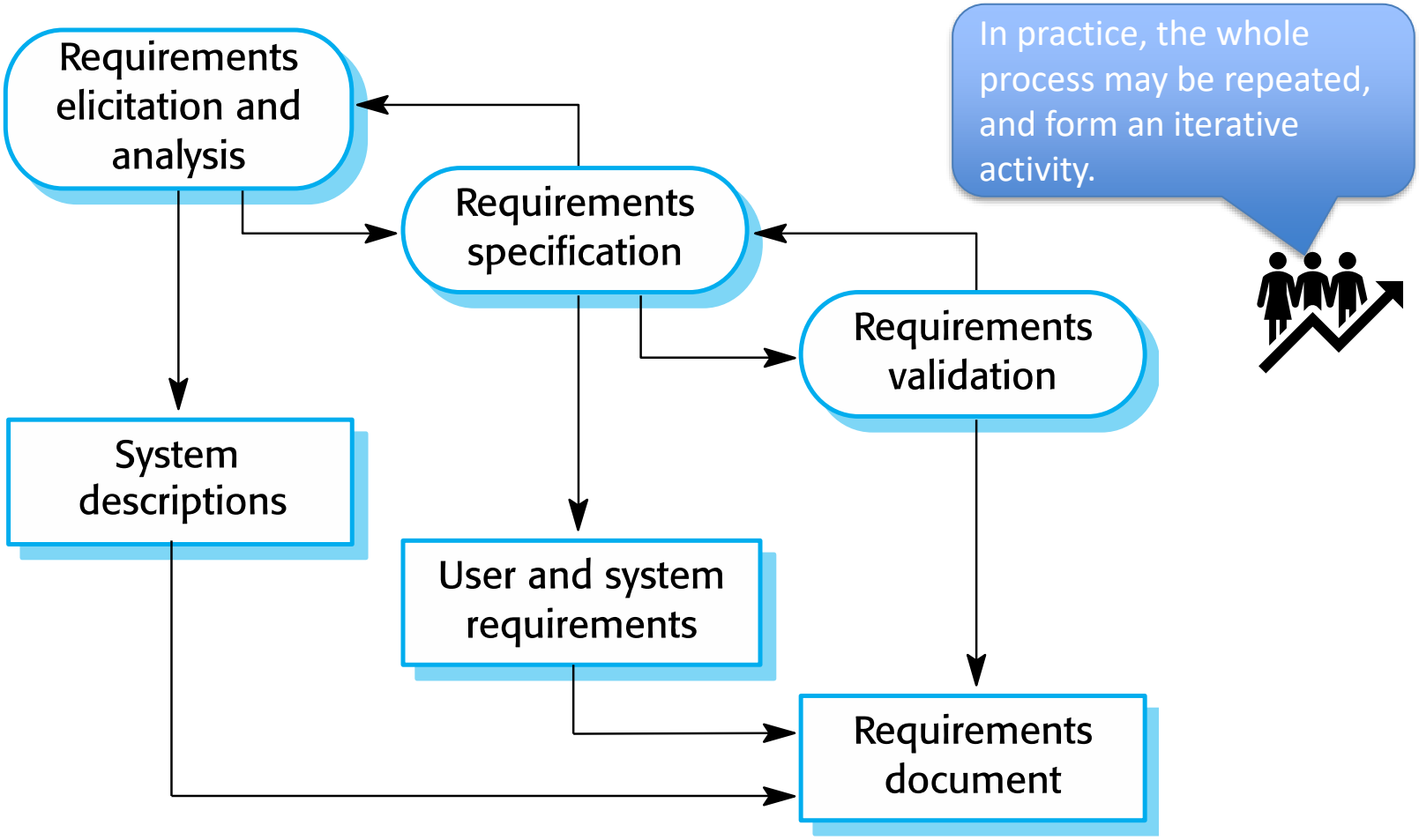


Requirements engineering processes

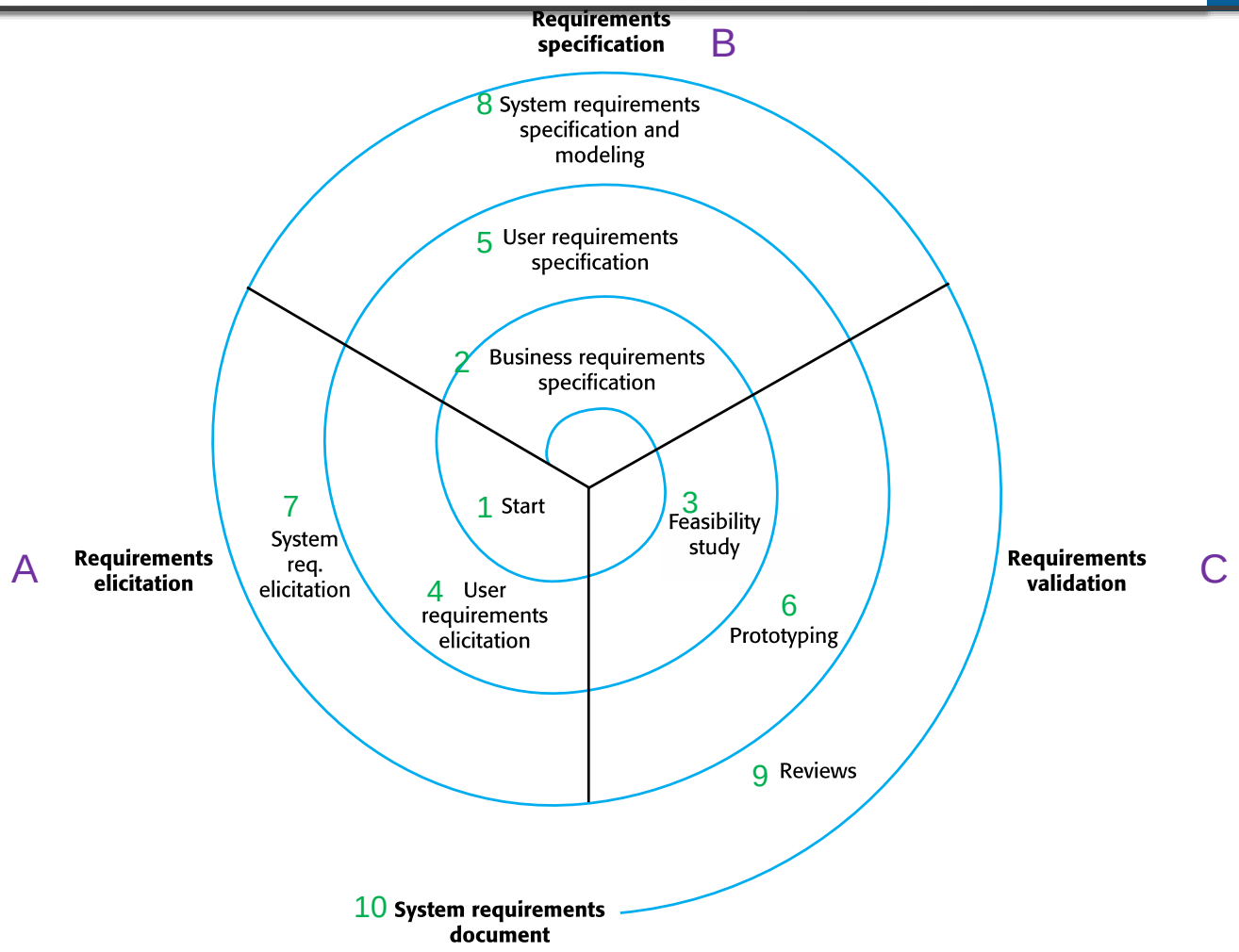
Requirements engineering processes

- 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 specification
  - Requirements validation;
  - Requirements management(evolution).

The requirements engineering process



A spiral view of the requirements engineering process



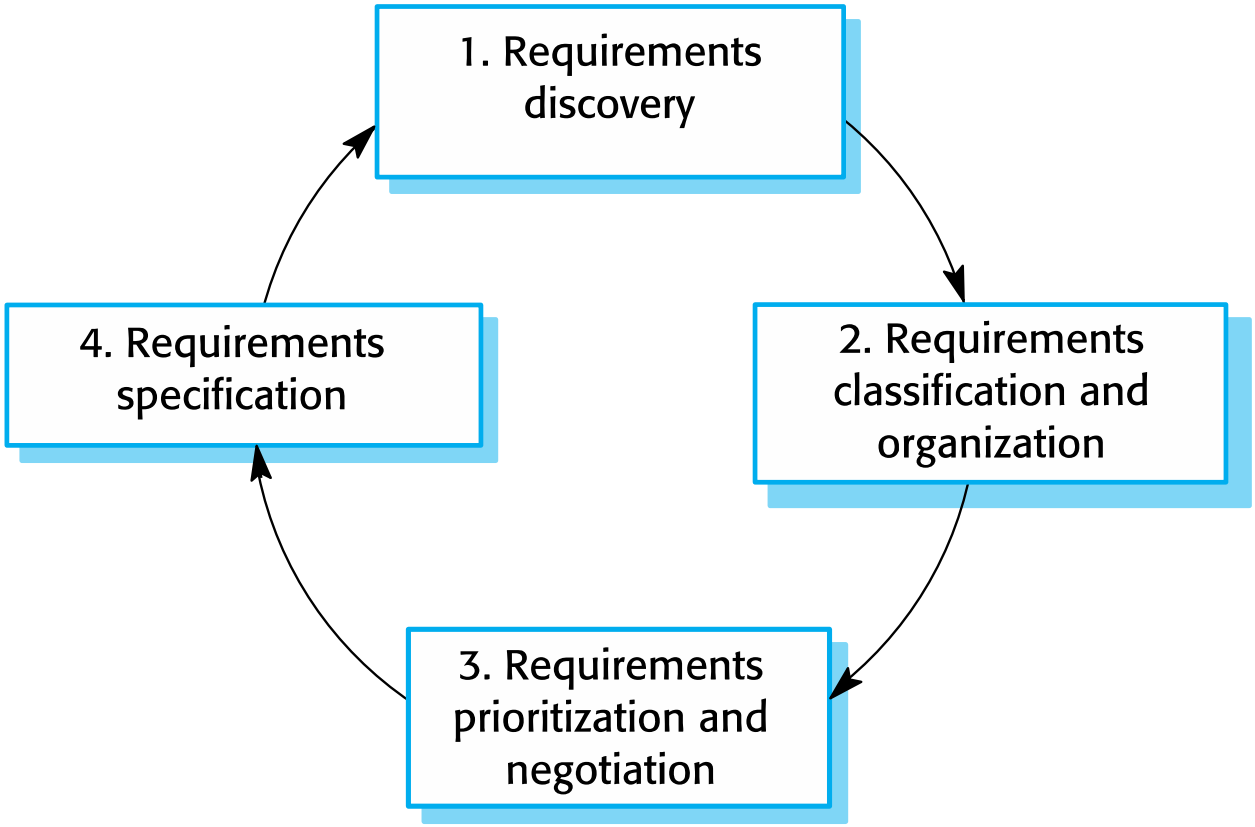
Requirements elicitation and analysis

- Sometimes called requirements discovery.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.
- 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.

Difficulties of requirements elicitation

- 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, because new stakeholders may emerge and the business environment may change.

The requirements elicitation and analysis process



Requirement elicitation techniques

- The process involves meeting(interview) with different system stakeholders to discover information about the system.
- Supplement with information, documents, knowledge of existing systems.
- Understand how people work, how they may need to change to accommodate a new system(Ethnography, Stories and scenarios).

Interviewing

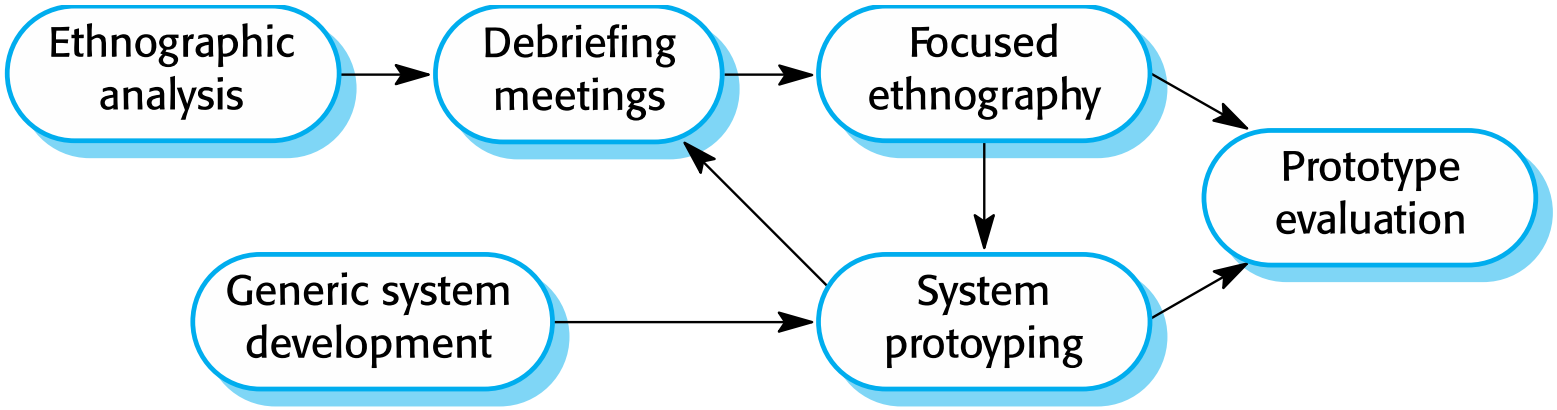
- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - Closed interviews based on pre-determined list of questions.
  - Open interviews where various issues are explored with stakeholders.
- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

Ethnography

- Analysts can spend a considerable time observing and analysing how people actually work.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Ethnography and prototyping for requirements analysis

- Combines ethnography with prototyping
- Prototype development results in unanswered questions which focus the ethnographic analysis.



Stories and scenarios

- Stories and user scenarios are real-life examples of how a system can be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Example：story of Photo sharing in the classroom (iLearn)

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, 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 as he wants pupils to take and comment on each others' 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 uses 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.

Scenarios

- A structured form of user story, should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

Example：scenario of Uploading photos（iLearn)-A

- **Initial assumption**: A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.
- **Normal**: The user chooses upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They 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 has been done.

Example：scenario of Uploading photos （iLearn)-B

- **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 that there could be a 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 they wish to re-upload the photos with the same name, rename the photos or cancel the upload. If they chose to re-upload the photos, the originals are overwritten. If they chose to rename the photos, a new name is automatically generated by adding a number to the existing file name.
- **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 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 software requirements document

- The software requirements document is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should be done.
- The software requirements document is the official statement of what is required of the system developers.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Requirements document variability

- Requirements documents standards have been designed e.g. IEEE standard.
- Information in requirements document is variable, depends on type of system and the approach to development used.

The IEEE standard for requirements documents

The most widely known requirements document standard is IEEE/ANSI 830-1998 (IEEE, 1998). This IEEE standard suggests the following structure for requirements documents:

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

3.**Specific requirements**, covering functional, non-functional and interface requirements. This is obviously the most substantial part of the document but because of the wide variability in organisational practice, it is not appropriate to define a standard structure for this section. The requirements may document external interfaces, describe system functionality and performance, and specify logical database requirements, design constraints, emergent system properties and quality characteristics.

4.**Appendices**

5.**Index**

Although the IEEE standard is not ideal, it contains a great deal of good advice on how to write requirements and how to avoid problems. It is too general to be an organisational standard in its own right. It is a general framework that can be tailored and adapted to define a standard geared to the needs of a particular organisation.

Notations for writing system requirements

Notation	Description
Natural language	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.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; UML 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 are reluctant to accept it as a system contract

Natural language specification

- Requirements are written as natural language sentences supplemented by diagrams and tables.
- Problems with natural language
  - Precision is difficult.
  - Several different requirements may be expressed together.
  - Functional and non-functional requirements tend to be mixed-up.

Structured specifications

- An approach where requirements are written in a standard way.
- This works well for some types of requirements e.g. requirements for embedded control system
- But it is sometimes too rigid for writing business system requirements.

Example：a structured specification of a requirement for an insulin pump（A）

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.

Example：a structured specification of a requirement for an insulin pump（B）

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.
- For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

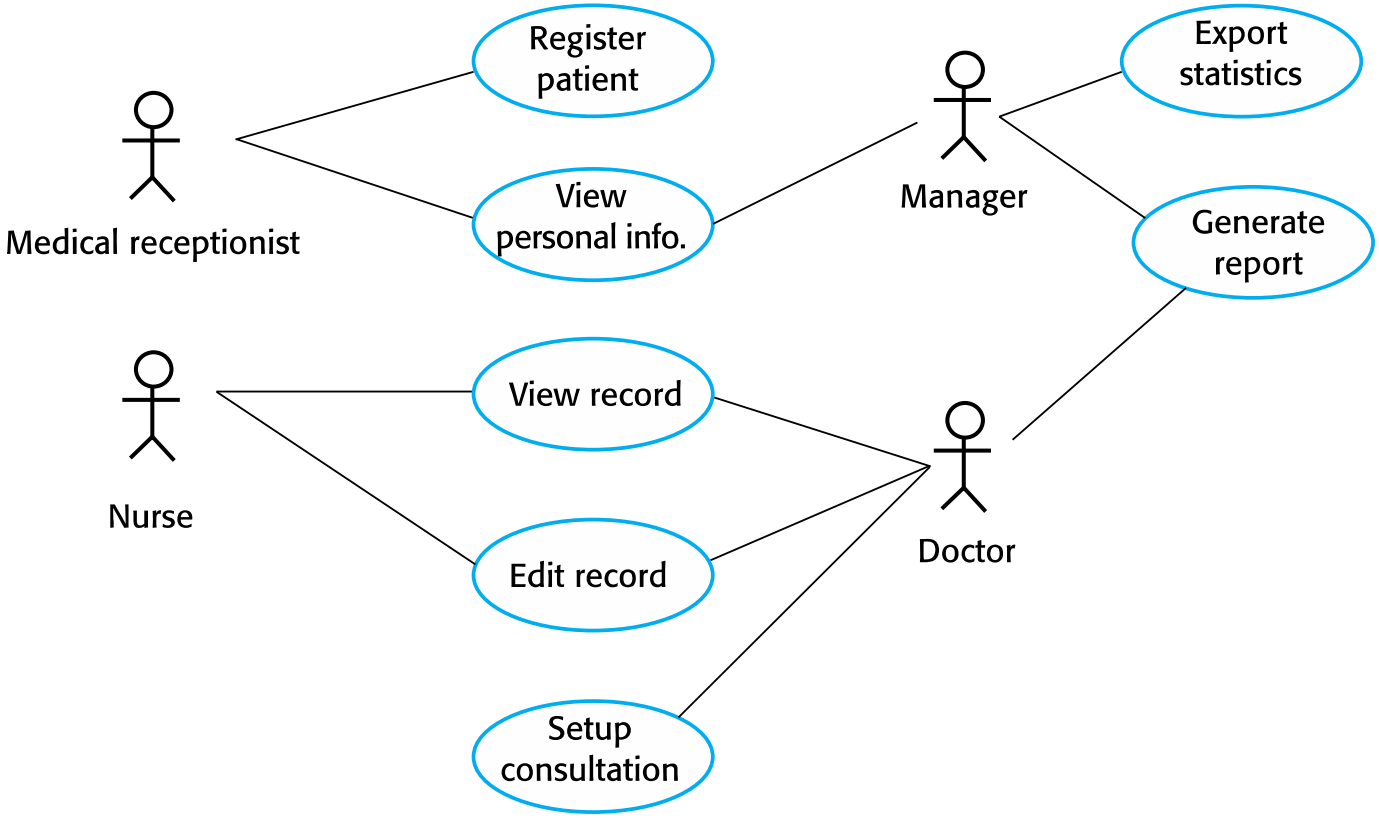
Example: Tabular specification of computation for an insulin pump

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

Use cases diagram

- Use-cases are a kind of scenario that are included in the UML.
- Use cases 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.
- UML sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Example: Use cases for the Mentcare system



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

Example: nonfunctional requirements in the Mentcare system

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

Product requirement
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

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

Requirements checking

Requirements error costs are high, so validation is very important.

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

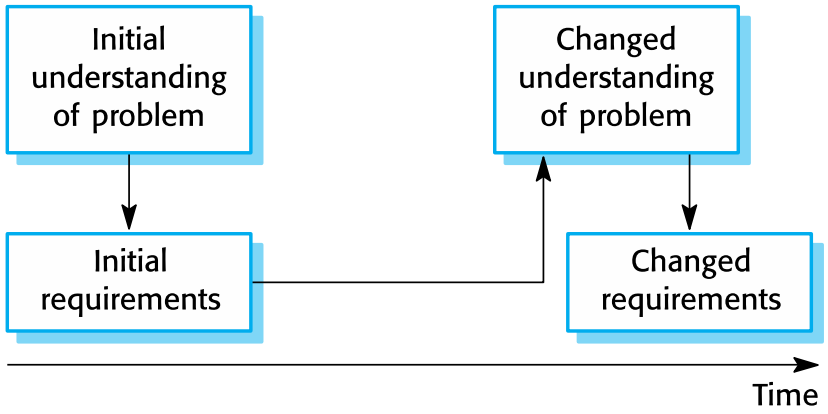
Requirements validation techniques

Checking techniques:

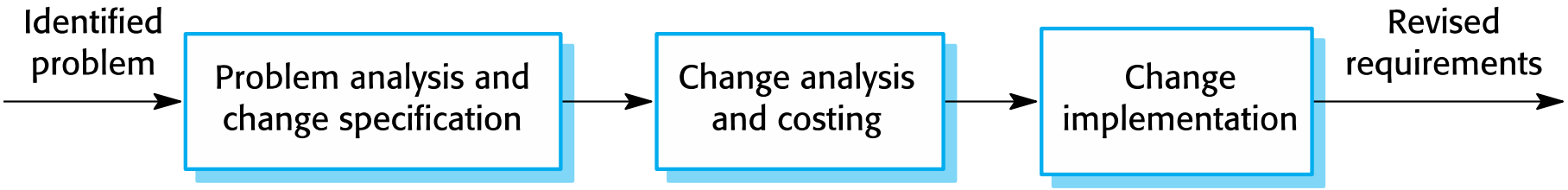
- ✧ Requirements reviews
  - Systematic manual analysis of the requirements.
- ✧ Prototyping
  - Using an executable model of the system to check requirements (Covered in Chapter 2).
- ✧ Test-case generation
  - Developing tests for requirements to check testability.

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.
- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.



Requirements change management



Essay4：Requirements engineering in traditional and agile contexts

- ✧ Research on traditional requirements and in requirements agile context.
- ✧ Write a 750-word essay contrasting requirements engineering in traditional (waterfall) and agile development contexts.

- Use the reading materials (textbook, articles, slides). You may use additional materials (that you have found by yourself), as well.
- Use referencing according to the guidelines, and try to aim for the 750-word limit. Too short or overly long essays will receive a lower score. For the highest score, your essay should be within the 560-940 word range (within 25%).
- Reference list is NOT included in the word limit
- Remember the importance of referencing, and of avoiding plagiarism.

Essay will be graded on:

- the correctness and comprehensiveness of traditional requirements engineering and agile requirements analysis.
- the reasonableness and appropriateness of your viewpoint of the combination method.
- the length, as described above.
- the correct use of referencing. Note that absence of references fails the essay completely, as you are required to reference).
- an overall assessment of the essay quality.