Software Engineering I

Dr. Elham Mahmoudzadeh

Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2022

Requirements Engineering

Software Development Life Cycle

- Planning
- Analysis
- Design
- Implementation

Introduction

- The systems development process aids an organization in moving from the current system (often called the *as-is system*) to the new system (often called the *to-be system*).

- The output of planning, is the system request, which provides general ideas for the to-be system, defines the project's scope, and provides the initial work-plan.

- Analysis takes the general ideas in the system request and refines them into a detailed requirements definition.

Let’s Start

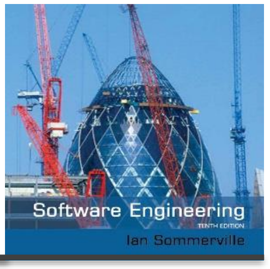
Steps(I)

1. Preparing proposal
2. Requirements determination
   - ➢ User story
3. Abstract Business Process Modelling
4. Analysis
   - ➢ Functional Modelling
   - ➢ Structural Modelling
   - ➢ Behavioral Modelling

Steps(II)

5. Design
   - ➢ Optimization
   - ➢ Database Management
   - ➢ User Interface
   - ➢ Physical Architecture

Requirements engineering

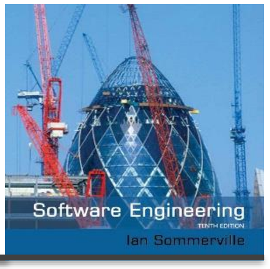


- ✧ The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.

What is requirement?

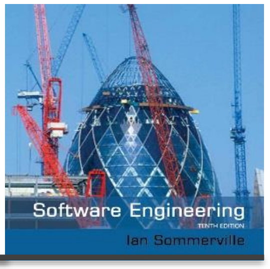
- Requirement is : new system’s capabilities.
- A *requirement* is simply a statement of what the system must do or what characteristic it must have.
- During analysis, requirements are written from the perspective of the businessperson, and they focus on the “what” of the system.
  - Focus on the needs of the business user, *business requirements* (user requirements).

System stakeholders

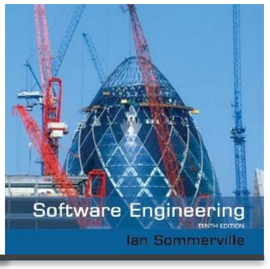


- ✧ Any person or organization who is affected by the system in some way and so who has a legitimate interest
- ✧ Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

Agile methods and requirements



- ✧ Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- ✧ The requirements document is therefore always out of date.
- ✧ Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’.
- ✧ This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.



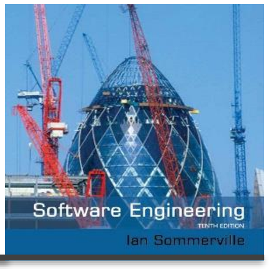
Functional and non-functional requirements

Functional Requirements

- Relates directly to a process that a system has to perform or information it needs to contain.
  - For example, requirements stating that a system must have the ability to search for a product.

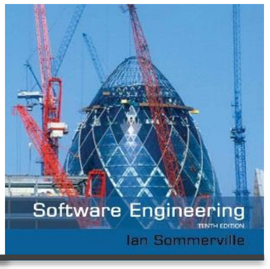
- Flow directly into the creation of functional, structural, and behavioral models that represent the functionality of the evolving system.

Functional requirements



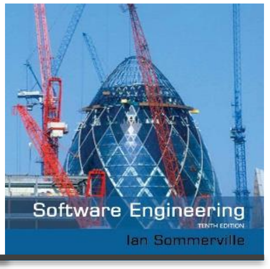
- ✧ Describe functionality or system services.
- ✧ Depend on the type of software, expected users and the type of system where the software is used.
- ✧ Functional user requirements may be high-level statements of what the system should do.
- ✧ Functional system requirements should describe the system services in detail.

Mentcare system: functional requirements



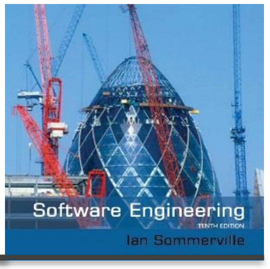
- ✧ A user shall be able to search the appointments lists for all clinics.
- ✧ The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- ✧ Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements imprecision



- ✧ Problems arise when functional 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

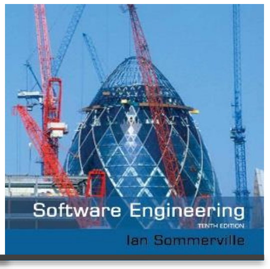


- ✧ In principle, requirements should be both complete and consistent.
- ✧ Complete
  - They should include descriptions of all facilities required.
- ✧ Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- ✧ In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

Non-functional Requirements

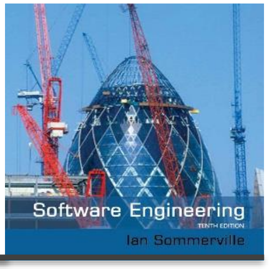
- Refer to behavioral properties that the system must have, such as performance and usability.
  - The ability to access the system using a Web browser is considered a nonfunctional requirement.
- Can influence the rest of analysis (functional, structural, and behavioral models) but often do so only indirectly;
- Are used primarily in design when decisions are made about the database, the user interface, the hardware and software, and the system's underlying physical architecture.

Non-functional requirements



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

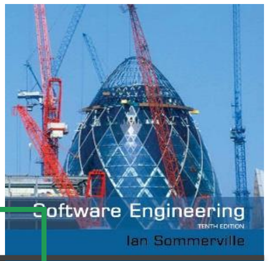
Non-functional requirements implementation



✧ Non-functional requirements may affect the overall architecture of a system rather than the individual components.

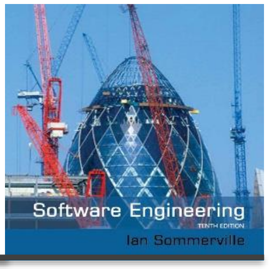
- For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.

Metrics for specifying nonfunctional requirements



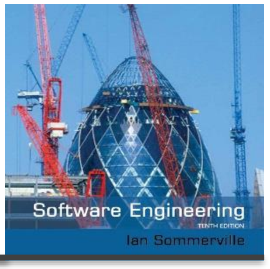
Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

معیارهایی برای مشخص کردن نیازمندیهای غیرعملکردی
ویژگی:
سرعت
اندازه گرفتن:
تراکنش های پردازش شده/ثانیه - زمان پاسخ کاربر/رویداد - زمان به روز رسانی صفحه
ویژگی:
اندازه
اندازه گرفتن:
مگابایت - تعداد تراشه های ROM\
ویژگی:
راحتی در استفاده
اندازه گرفتن:
زمان آموزش - تعداد فریم های راهنما
ویژگی:
قابلیت اطمینان
اندازه گرفتن:
میانگین زمان تا شکست - احتمال در دسترس نبودن - میزان وقوع شکست
دسترسی
ویژگی:
نیرومندی
اندازه گرفتن:
زمان راه اندازی مجدد پس از شکست - درصد رویدادهایی که باعث خرابی می شوند - احتمال خراب شدن داده ها در صورت شکست
ویژگی:
قابل حمل بودن
اندازه گرفتن:
درصد عبارات وابسته به هدف - تعداد سیستم های هدف



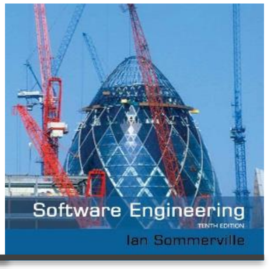
Requirements engineering processes

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

Requirements elicitation and analysis

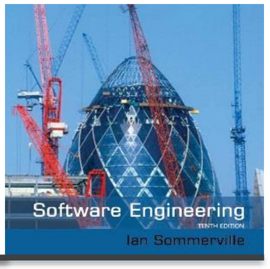


✧ Sometimes called requirements elicitation or requirements discovery.

✧ Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system’s operational constraints.

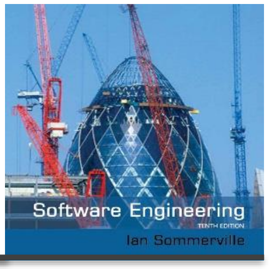
✧ May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*

Requirements elicitation



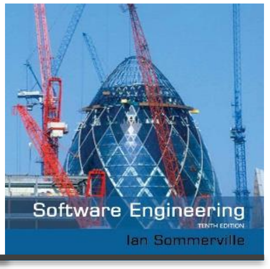
- ✧ Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- ✧ Stages include:
  - Requirements discovery,
  - Requirements classification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

Problems of requirements elicitation



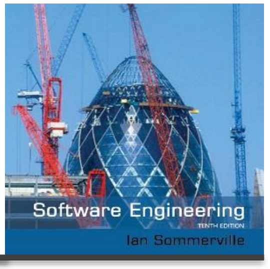
- ✧ Stakeholders don’t know what they really want.
- ✧ Stakeholders express requirements in their own terms.
- ✧ Different stakeholders may have conflicting requirements.
- ✧ Organisational and political factors may influence the system requirements.
- ✧ The requirements change during the analysis process.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

Process activities



- ✧ Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- ✧ Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- ✧ Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- ✧ Requirements specification
  - Requirements are documented and input into the next round of the spiral.

Requirements discovery



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

Requirements Determination(I)

- Usually, users don’t know exactly what they want, and analysts need to help them discover their needs.
- Analysts guide the users in explaining what is wanted from a system.
- Analysts help users critically examine the current state of systems and processes (the *as-is system*), identify exactly what needs to change, and develop a concept for a new system (the *to-be syste*m).

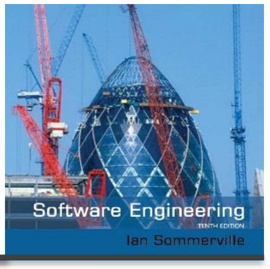
Requirements Determination(II)

- Creating a requirements definition is an iterative and ongoing process whereby the analyst collects information with requirements-gathering techniques.
- Then analyst analyzes the information to identify appropriate business requirements for the system.
- The requirements definition is kept up to date so that the project team and business users can refer to it and get a clear understanding of the new system.

Requirements gathering

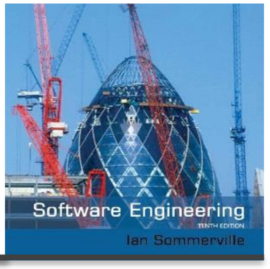
- Using a variety of techniques and make sure that the current business processes and the needs for the new system are well understood before moving into design.
- Not to discover later that they have wrong key requirements.
- All the key stakeholders must be included in the requirements-gathering process.

Interviewing



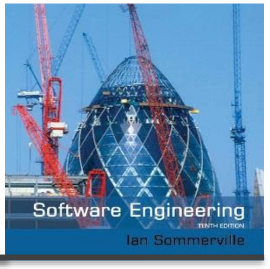
- ✧ Formal or informal interviews with stakeholders are part of most RE processes.
- ✧ Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- ✧ Effective interviewing
  - Be open-minded, avoid pre-conceived 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.

Interviews in practice



- ✧ Normally a mix of closed and open-ended interviewing.
- ✧ Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- ✧ Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- ✧ You need to prompt the use to talk about the system by suggesting requirements rather than simply asking them what they want.

Problems with interviews



- ✧ Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- ✧ 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.

Requirements Gathering Techniques(I)

- **Interview**: is the most commonly used requirements-gathering technique. After all, it is natural—if you need to know something, you usually ask someone.

- **Joint Application Development (JAD)**: is a technique that allows the project team, users, and management to work together to identify requirements for the system.

- **Questionnaires**: is a set of written questions used to obtain information from individuals.
  - Are often used when there is a large number of people from whom information and opinions are needed.

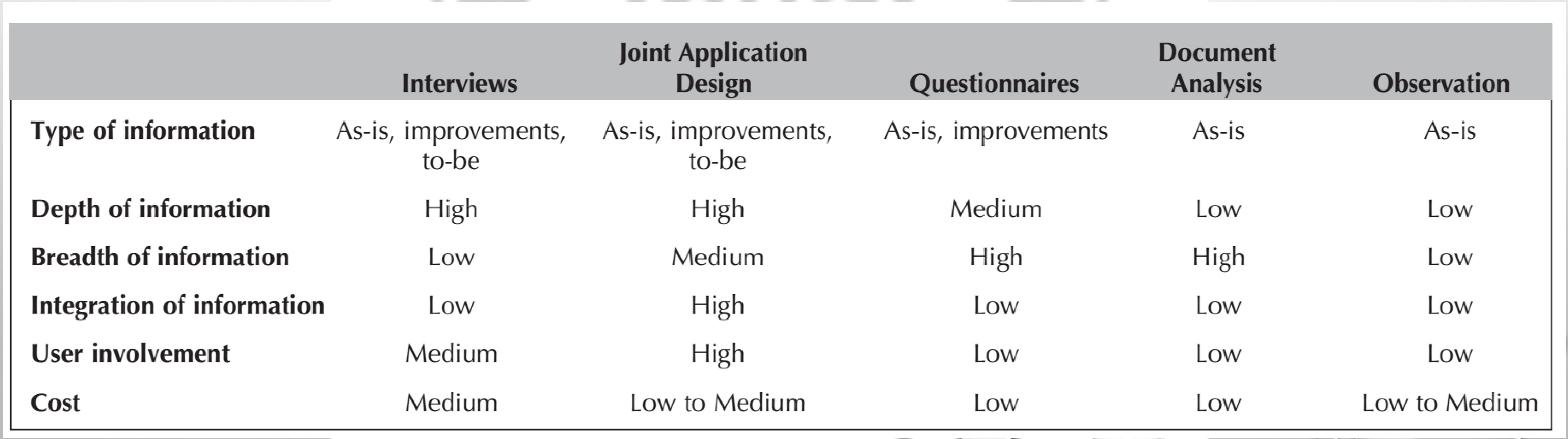
Requirements Gathering Techniques(II)

- **Document Analysis**: Project teams often use document analysis to understand the as-is system.

- **Observation**: the act of watching processes being performed, is a powerful tool for gathering information about the as-is system because it enables the analyst to see the reality of a situation, rather than listening to others describe it in interviews or JAD sessions. <u>Observation is a good way to check the validity of information gathered from indirect sources such as interviews and questionnaires.</u>

Which one is appropriate?

- No one technique is always better than the others.
- In practice most projects use a combination of techniques.
- It is important to understand the strengths and weaknesses of each technique and when to use.



	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low to Medium	Low	Low	Low to Medium

Type of Information

- Different stages of the analysis process: *understanding the as-is system*, *identifying improvements*, and *developing the to-be system*.
- **Interviews** and **JAD** are commonly used in all three stages.
- In contrast, **document analysis** and **observation** usually are most helpful for understanding the as-is, although occasionally they provide information about current problems that need to be improved.
- **Questionnaires** are often used to gather information about the as-is system as well as general information about improvements.

Depth of Information

- Refers to how rich and detailed the information is that the technique usually produces and the extent to which the technique is useful for obtaining not only facts and opinions but also an understanding of *why* those facts and opinions exist.
- **Interviews** and **JAD** sessions are very useful for providing a good depth of rich and detailed information and helping the analyst to understand the reasons behind them.
- **Document** analysis and **observation** are useful for obtaining facts, but little beyond that.
- **Questionnaires** can provide a medium depth of information, soliciting both facts and opinions with little understanding of why they exist.

Breadth of Information

- Refers to the range of information and information sources that can be easily collected using the chosen technique.
- **Questionnaires** and **document analysis** are both easily capable of soliciting a wide range of information from a large number of information sources.
- **Interviews** and **observation** require the analyst to visit each information source individually and, therefore, take more time.
- **JAD** sessions are in the middle because many information sources are brought together at the same time.

Integration of Information

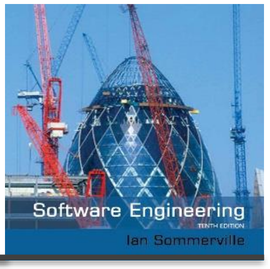
- One of the most challenging aspects of requirements gathering is integrating the information from different sources.
- Simply put, different people can provide conflicting information. Combining this information and attempting to resolve differences in opinions or facts is usually very time consuming because it means contacting each information source in turn, explaining the discrepancy, and attempting to refine the information.
- All techniques suffer integration problems to some degree, but **JAD** sessions are designed to improve integration because all information is integrated when it is collected, not afterward. The immediate integration of information is the single most important benefit of JAD that distinguishes it from other techniques.

User Involvement

- Refers to the amount of time and energy the intended users of the new system must devote to the analysis process.
- It is generally agreed that as users become more involved, the chance of success increases.
- User involvement can have a significant cost, and not all users are willing to contribute valuable time and energy.
- **Questionnaires**, **document analysis**, and **observation** place the least burden on users.
- **JAD** sessions require the greatest effort.

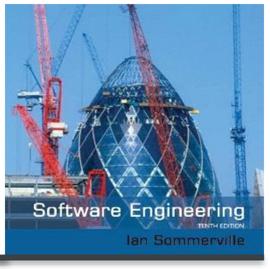
Cost

- Is always an important consideration.
- **Questionnaires**, **document analysis**, and **observation** are low-cost techniques (although observation can be quite time consuming).
- **Interviews** and **JAD** sessions generally have moderate costs. In general, JAD sessions are much more expensive initially, because they require many users to be absent from their offices for significant periods of time, and they often involve highly paid consultants. However, JAD sessions significantly reduce the time spent in information integration and thus can cost less in the long term.



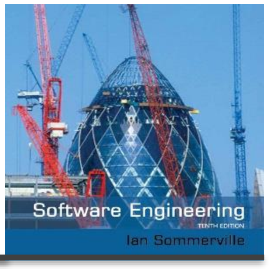
Requirements specification

Requirements specification



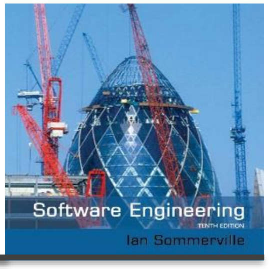
- ✧ The process of writing down the requirements in a requirements document.
- ✧ The requirements may be part of a contract for the system development
  - ▪ It is therefore important that these are as complete as possible.

Guidelines for writing requirements



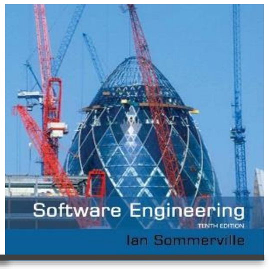
- ✧ Invent a standard format and use it for all requirements.
- ✧ Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- ✧ Use text highlighting to identify key parts of the requirement.
- ✧ Avoid the use of computer jargon.
- ✧ Include an explanation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

Problems with natural language



- ✧ Lack of clarity
  - Precision is difficult without making the document to read.
- ✧ Requirements confusion
  - Functional and non-functional requirements tend to be mixed-up.
- ✧ Requirements amalgamation
  - Several different requirements may be expressed together.

Structured specifications

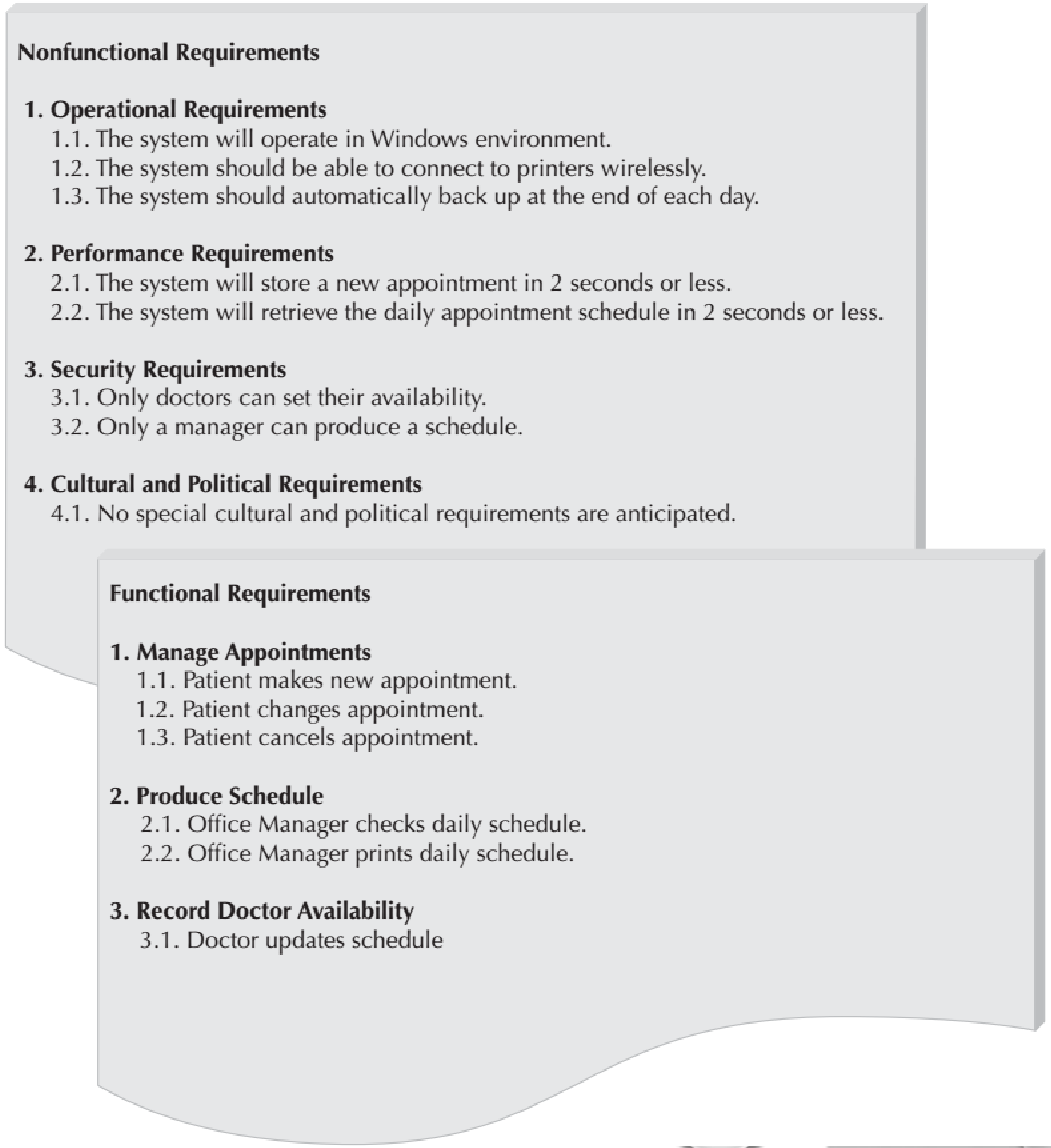


- ✧ An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- ✧ This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Requirements document(I)

- Is a straightforward text report that simply lists the functional and nonfunctional requirements.
- Sometimes business requirements are prioritized on the requirements definition. They can be ranked as having high, medium, or low importance in the new system, or they can be labeled with the version of the system that will address the requirement (e.g., release 1). This practice is particularly important when using object-oriented methodologies since they deliver systems in an incremental manner.
- The most important purpose of the requirements definition, is to define the scope of the system.
- When discrepancies arise, the document serves as the place to clarify.

An Example



Nonfunctional Requirements

1. Operational Requirements
1.1. The system will operate in Windows environment.
1.2. The system should be able to connect to printers wirelessly.
1.3. The system should automatically back up at the end of each day.

2. Performance Requirements
2.1. The system will store a new appointment in 2 seconds or less.
2.2. The system will retrieve the daily appointment schedule in 2 seconds or less.

3. Security Requirements
3.1. Only doctors can set their availability.
3.2. Only a manager can produce a schedule.

4. Cultural and Political Requirements
4.1. No special cultural and political requirements are anticipated.

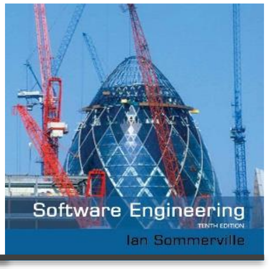
Functional Requirements

1. Manage Appointments
1.1. Patient makes new appointment.
1.2. Patient changes appointment.
1.3. Patient cancels appointment.

2. Produce Schedule
2.1. Office Manager checks daily schedule.
2.2. Office Manager prints daily schedule.

3. Record Doctor Availability
3.1. Doctor updates schedule

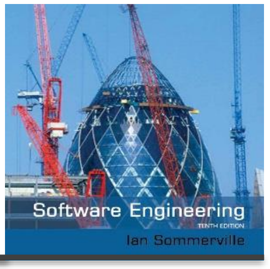
The software requirements document



- ✧ The software requirements document is the official statement of what is required of the system developers.
- ✧ Should include both a definition of user requirements and a specification of the system requirements.
- ✧ It is NOT a design document. As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

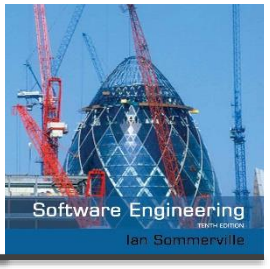
Requirements validation

Requirements validation



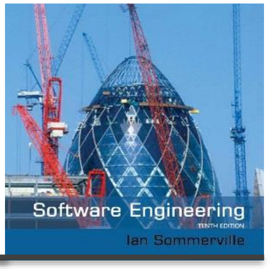
- ✧ Concerned with demonstrating that the requirements define the system that the customer really wants.
- ✧ Requirements error costs are high so validation is very important
  - ▪ Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

Requirements checking



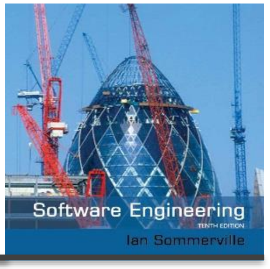
- ✧ Validity. Does the system provide the functions which best support the customer’s needs?
- ✧ Consistency. Are there any requirements conflicts?
- ✧ Completeness. Are all functions required by the customer included?
- ✧ Realism. Can the requirements be implemented given available budget and technology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

Requirements validation techniques

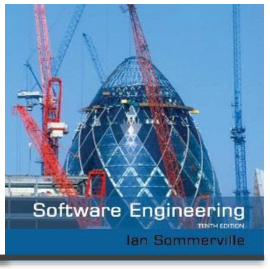


- ✧ Requirements reviews
  - Systematic manual analysis of the requirements.
- ✧ Prototyping
  - Using an executable model of the system to check requirements.
- ✧ Test-case generation
  - Developing tests for requirements to check testability.

Requirements reviews

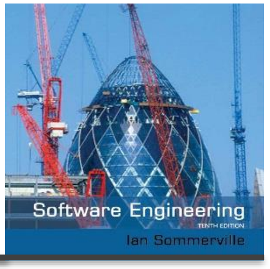


- ✧ Regular reviews should be held while the requirements definition is being formulated.
- ✧ Both client and contractor staff should be involved in reviews.
- ✧ Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems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



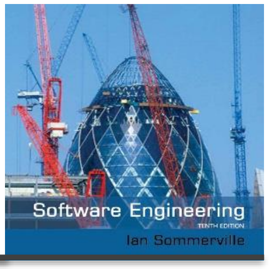
Requirements change

Changing requirements



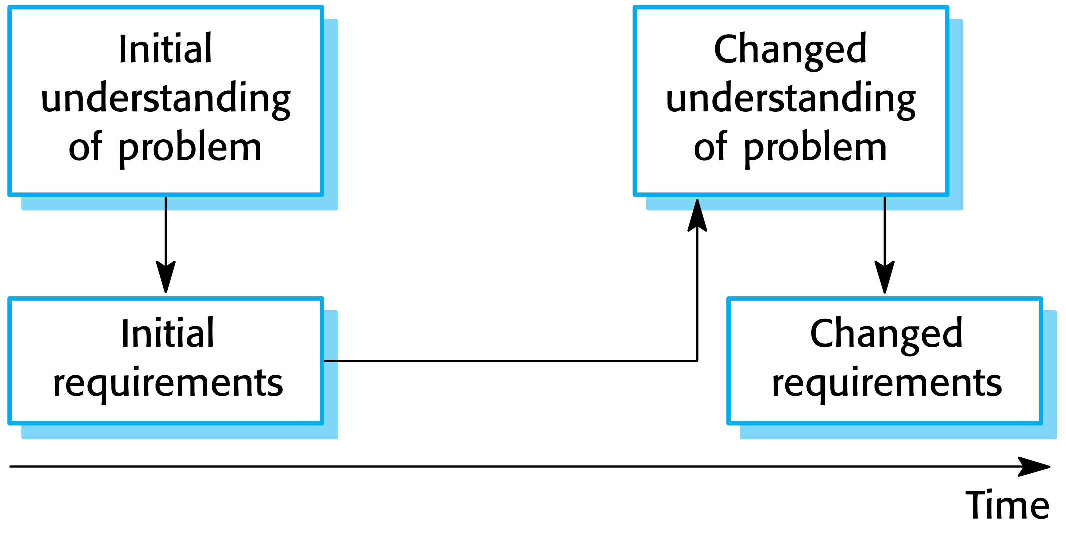
- ✧ The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation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

Changing requirements

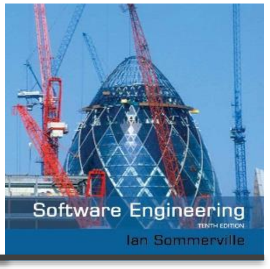


- ✧ Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
    - ▪ The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance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

Requirements evolution


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Requirements management



- ✧ Requirements management is the process of managing changing requirements during the requirements engineering process and system development.
- ✧ New requirements emerge as a system is being developed and after it has gone into use.
- ✧ You need to keep track of individual requirements 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.

References

- Dennis, Wixon, Tegarden, “System Analysis and Design, An Object Oriented Approach with UML”, 5th Edition, 2015.
- Summerville, “Software Emgineering”, 10th Edition, 2014.