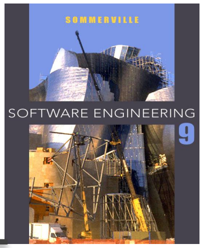
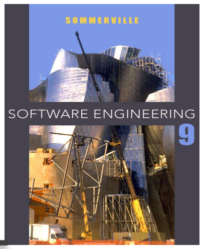


Chapter 4 – Requirements Engineering

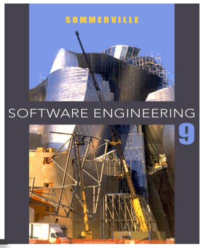
Topics covered



1. What is requirements
2. Types of requirements
3. Requirements engineering processes

   The overall requirements engineering process includes four high-level requirements engineering sub-process

   - Requirements elicitation
   - Requirements analysis
   - Requirements validation
   - Requirements management



User Requirement Definition

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

System Requirements Specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

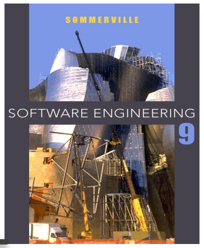
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed, and the total cost of the prescribed drugs.

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

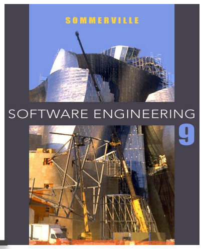
1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

What is a requirements?



- It may range from a high-level abstract **statement of a service or of a system constraint** to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract – therefore must be open to interpretation;
  - May be the basis for the contract itself – therefore must be defined in detail;
  - Both these statements may be called requirements.

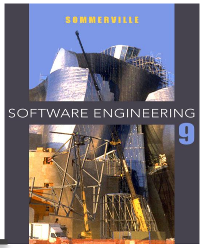
Functional and non-functional requirements



- Functional requirements
  - Statements of **services the system should provide**, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.

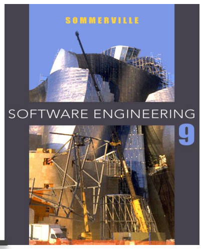
- Non-functional requirements
  - **Constraints on the services or functions** offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.

Functional requirements



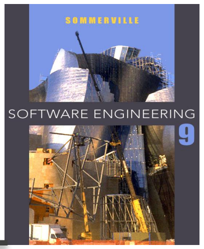
- Describe **functionality or system services**.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services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

Non-functional requirements

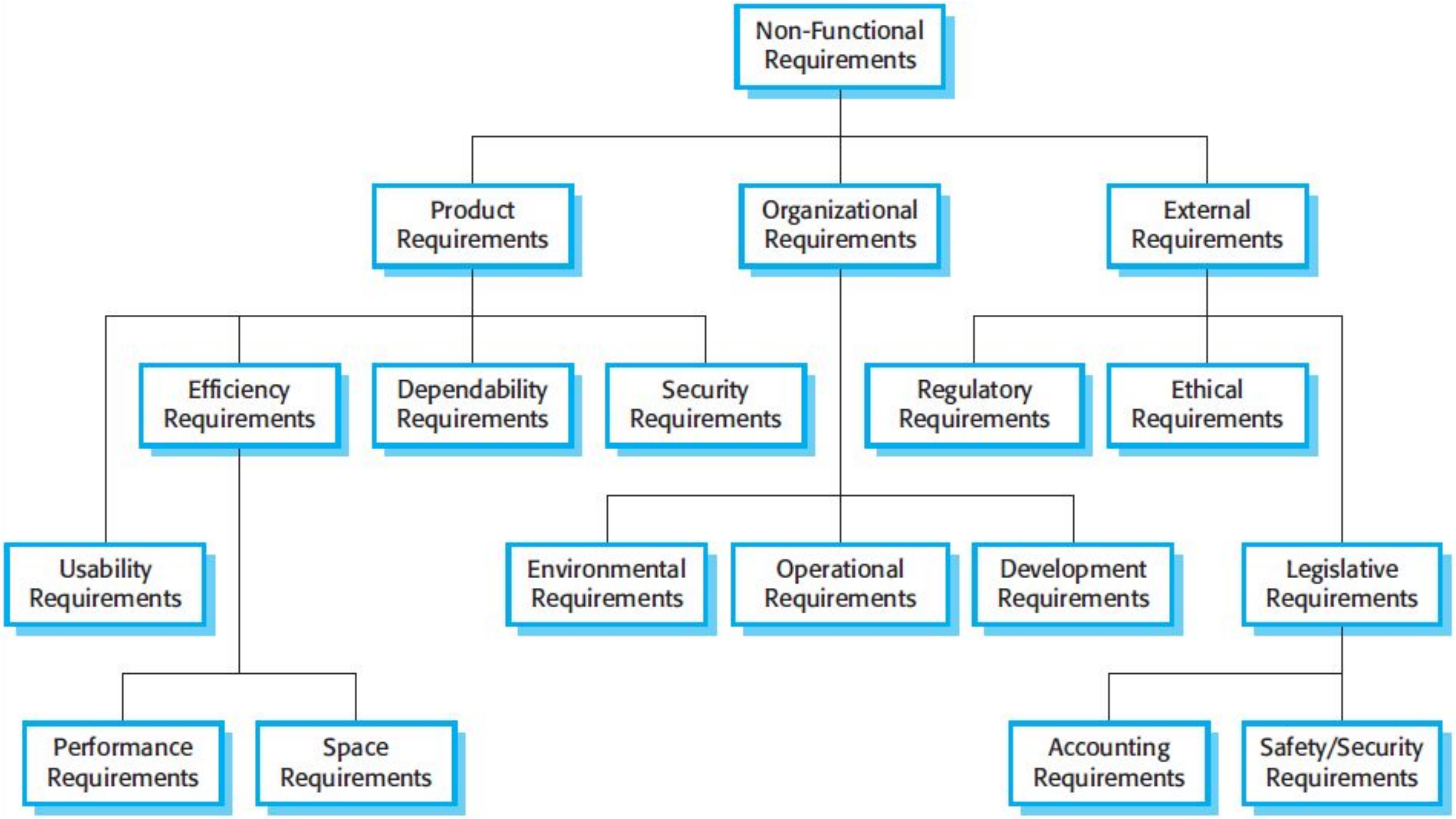


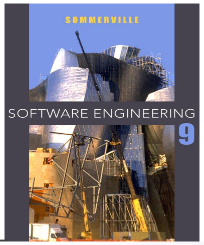
- These define **system properties and constraints** e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional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

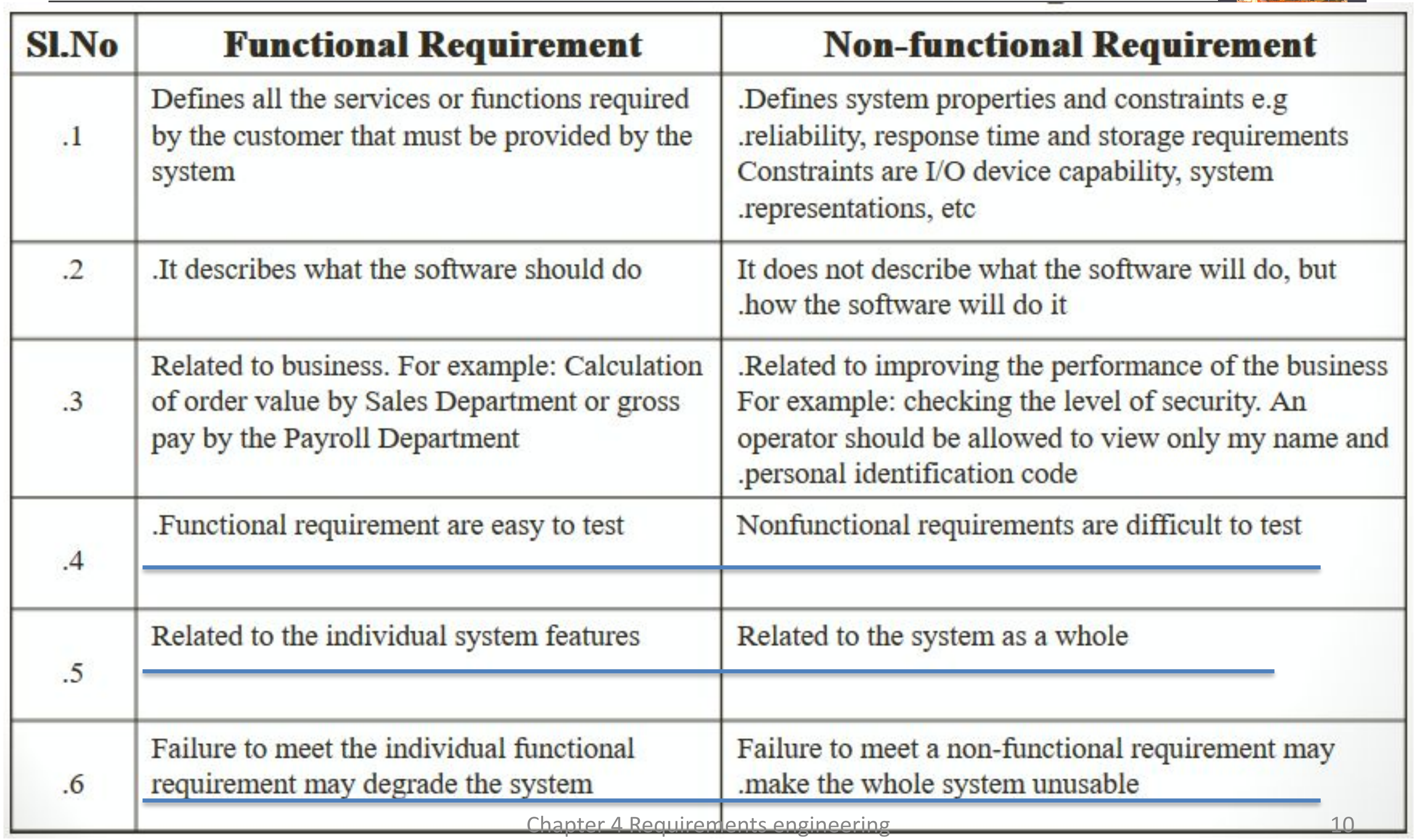
Non-functional classifications



- Product requirements
  - Requirements which specify that the delivered **product must behave in a particular way** e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of **organisational policies and procedures** e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are **external to the system** and its development process e.g. interoperability requirements, legislative requirements, etc.

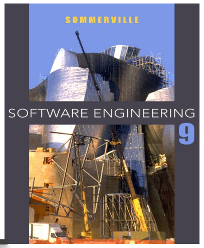

Non-Functional Requirements
Product Requirements
Organizational Requirements
External Requirements
Efficiency Requirements
Dependability Requirements
Security Requirements
Regulatory Requirements
Ethical Requirements
Usability Requirements
Environmental Requirements
Operational Requirements
Development Requirements
Legislative Requirements
Performance Requirements
Space Requirements
Accounting Requirements
Safety/Security Requirements





Sl.No	Functional Requirement	Non-functional Requirement
.1	Defines all the services or functions required by the customer that must be provided by the system	.Defines system properties and constraints e.g .reliability, response time and storage requirements Constraints are I/O device capability, system .representations, etc
.2	.It describes what the software should do	It does not describe what the software will do, but .how the software will do it
.3	Related to business. For example: Calculation of order value by Sales Department or gross pay by the Payroll Department	.Related to improving the performance of the business For example: checking the level of security. An operator should be allowed to view only my name and .personal identification code
.4	.Functional requirement are easy to test	Nonfunctional requirements are difficult to test
.5	Related to the individual system features	Related to the system as a whole
.6	Failure to meet the individual functional requirement may degrade the system	Failure to meet a non-functional requirement may .make the whole system unusable

Examples of nonfunctional requirements in the MHC-PMS



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

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

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

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

3. Functional requirements

ARENA supports five types of users:

- The *operator* should be able to define new games, define new tournament styles (e.g., knock-out tournaments, championships, best of series), define new expert rating formulas, and manage users.
- *League owners* should be able to define a new league, organize and announce new tournaments within a league, conduct a tournament, and declare a winner.
- *Players* should be able to register in an arena, apply for a league, play the matches that are assigned to the player, or drop out of the tournament.
- *Spectators* should be able to monitor any match in progress and check scores and statistics of past matches and players. Spectators do not need to register in an arena.
- The *advertiser* should be able to upload new advertisements, select an advertisement scheme (e.g., tournament sponsor, league sponsor), check balance due, and cancel advertisements.

4. Nonfunctional requirements

- *Low operating cost.* The operator must be able to install and administer an arena without purchasing additional software components and without the help of a full-time system administrator.
- *Extensibility.* The operator must be able to add new games, new tournament styles, and new expert rating formulas. Such additions may require the system to be temporarily shut down and new modules (e.g., Java classes) to be added to the system. However, no modifications of the existing system should be required.
- *Scalability.* The system must support the kick-off of many parallel tournaments (e.g., 10), each involving up to 64 players and several hundreds of simultaneous spectators.
- *Low-bandwidth network.* Players should be able to play matches via a 56K analog modem or faster.

VOIP (conference call Project)

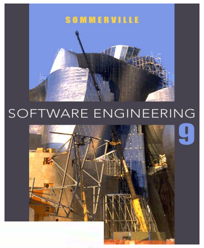
Functional Requirements

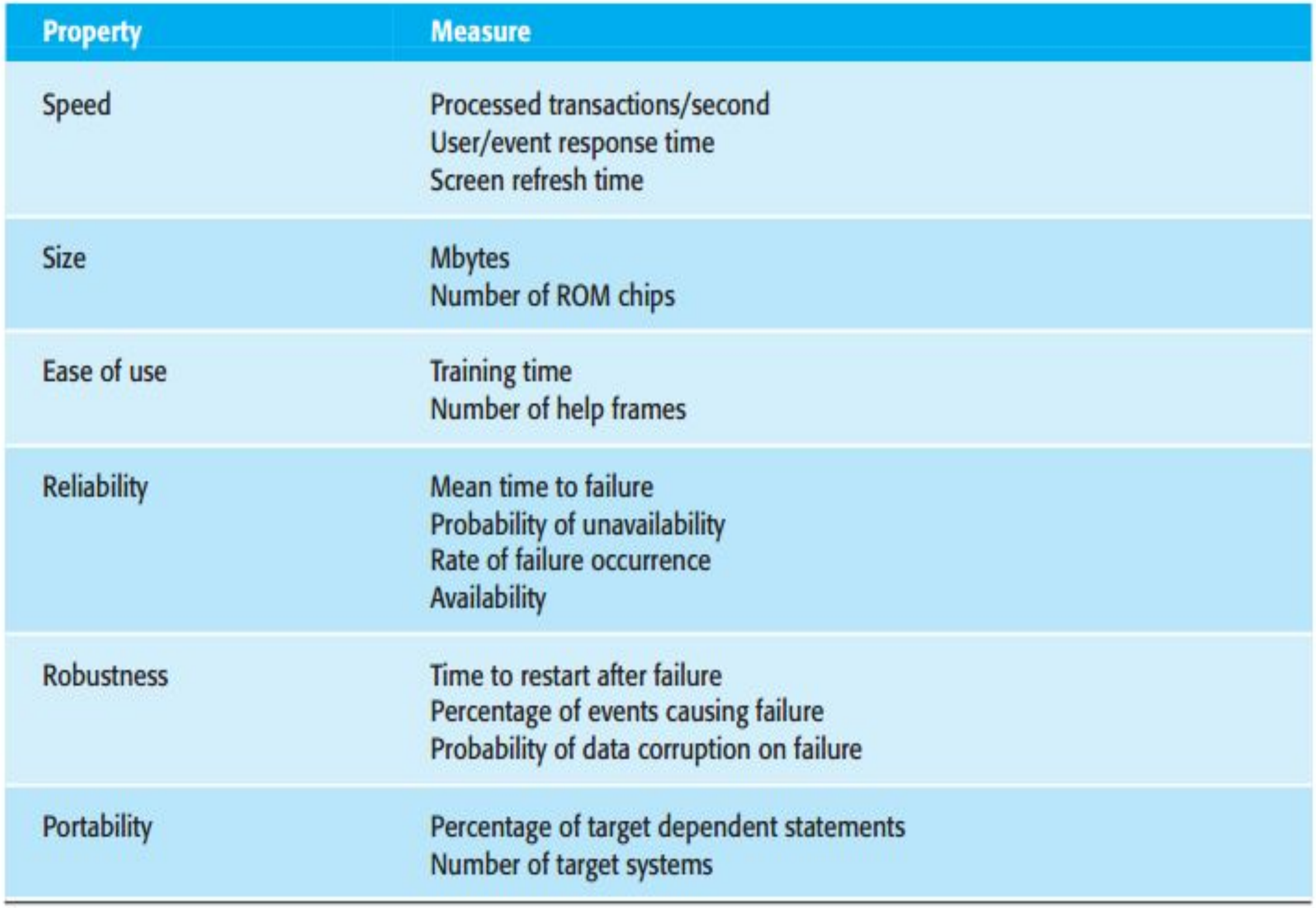
- Add Participants
- Drop Participants
- Count Participants
- Mute Microphone
- Invite/prompt Operator

Non-Functional Requirements

- Voice/ Video quality
- Reliability
- Availability
- Ease of use
- Cost
- Localization

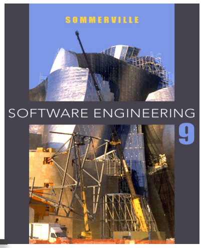
Metrics for specifying non-functional requirements





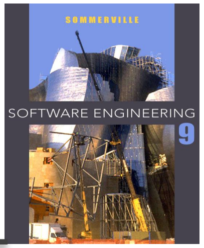
Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart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

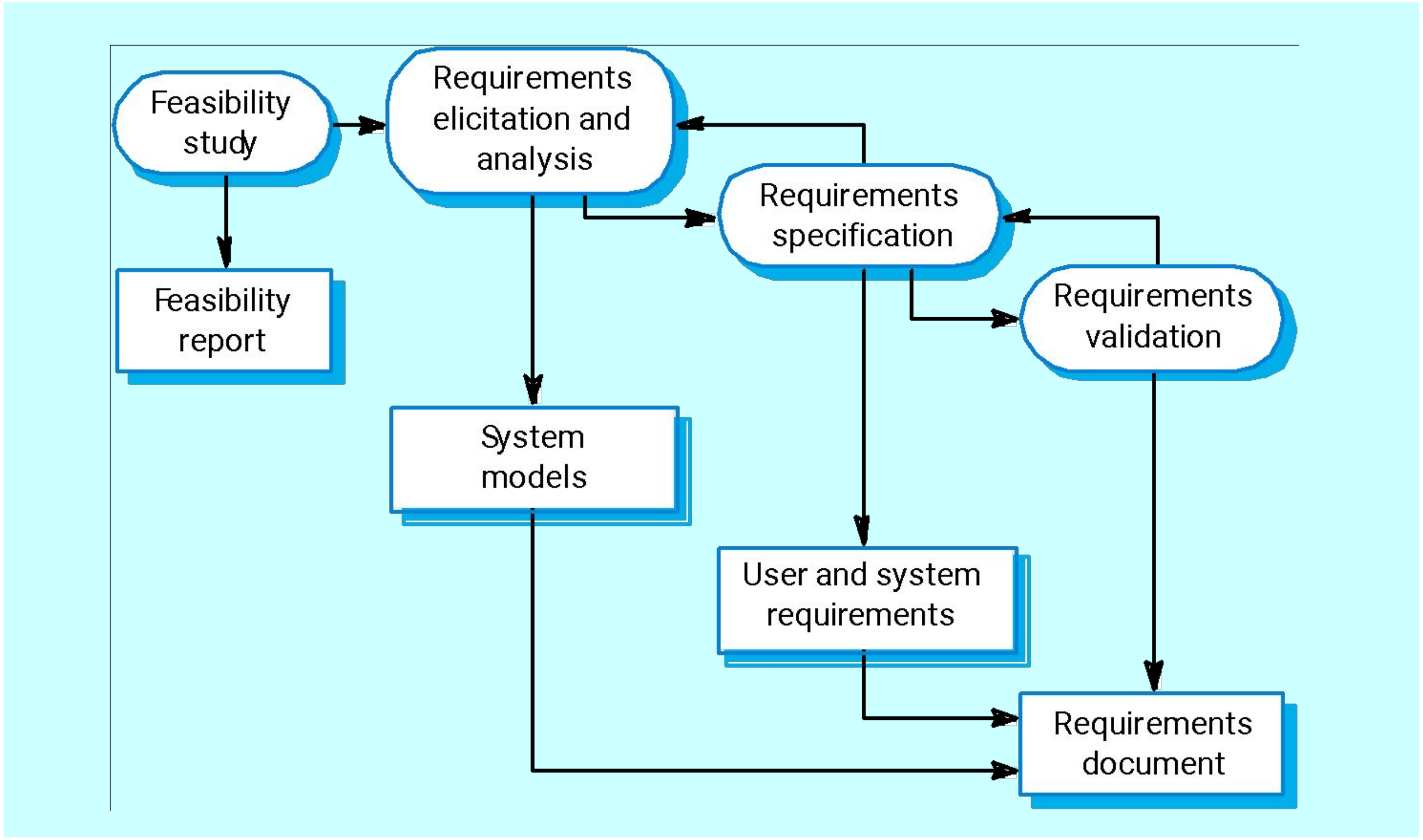
Requirements engineering processes



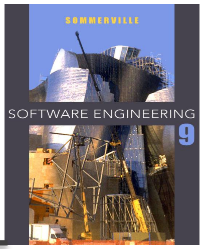
- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- However, there are a number of generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements management.

The requirements engineering process



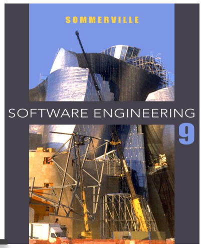


Feasibility studies



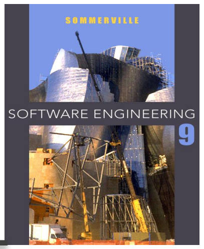
- A feasibility study decides whether or not the proposed system is worthwhile.
- A short focused study that checks
  - If the system contributes to organisational objectives;
  - If the system can be engineered using current technology and within budget;
  - If the system can be integrated with other systems that are already used.

Feasibility study implementation



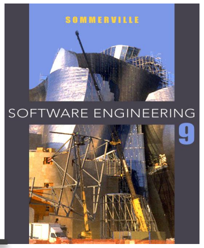
- Based on information assessment (what is required), information collection and report writing.
- Questions for people in the organisation
  - What if the system wasn’t implemented?
  - What are current process problems?
  - How will the proposed system help?
  - What will be the integration problems?
  - Is new technology needed? What skills?
  - What facilities must be supported by the proposed system?

Elicitation and analysis



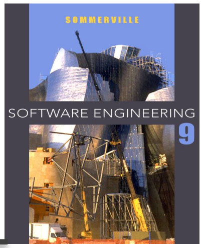
- Sometimes called requirements elicitation or requirements discovery.
- Involves ***technical staff working with customers*** to find out about the ***application domain***, the ***services*** that the system should provide and the system's ***operational constraints***.
- May involve end-users, managers,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

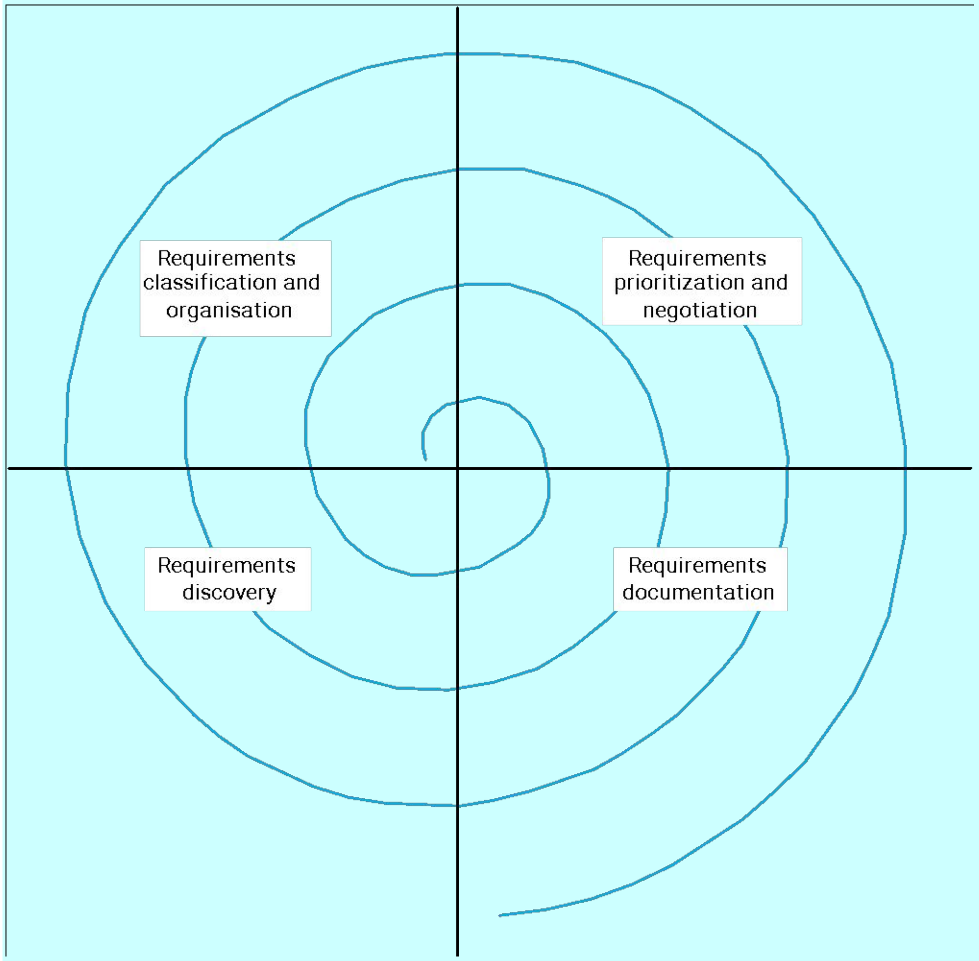
Problems of requirements analysis



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

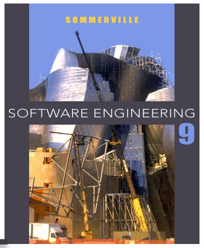
The requirements spiral for elicitation and analysis process




Requirements classification and organisation
Requirements prioritization and negotiation
Requirements discovery
Requirements documentation

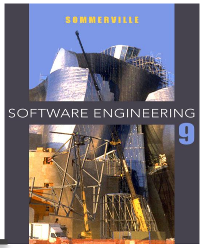
This iterative process starts from requirements discovery and ends with requirements documentation.

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.
    - *For example, it is possible to use model of the system architecture to identify subsystem and grouping related requirements.*
- ***Prioritisation and negotiation***
  - Prioritising requirements and resolving requirements conflicts.
    - *For example it is possible to organize stakeholders negotiations so that compromises can be reached.*
- ***Requirements documentation***
  - Requirements are documented and input into the next round of the spiral. The documentation can be formal or informal.
    - *For example, extreme programming uses formal documents and cards and it is very easy for stakeholders to handle, change and organise requirements.*

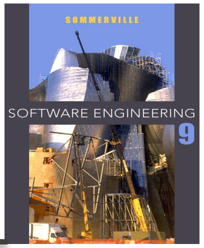
Requirements discovery



- The process of gathering information about the proposed and existing systems and distilling the user and system requirements from this information.
- Sources of information include documentation, system stakeholders and the specifications of similar systems.

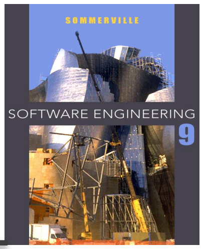
http://www.navodayaengg.in/wp-content/uploads/2015/10/Lect6_Requirements-Discovery.pdf

Requirements discovery and viewpoints



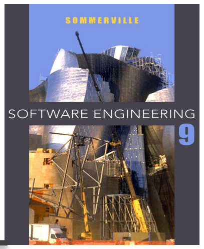
- Viewpoints are a way of structuring the requirements ***to represent the perspectives of different stakeholders***. Stakeholders and other sources may be classified under different viewpoints.
- This multi-perspective analysis is important as there is no single correct way to analyse system requirements.
- Types of viewpoint
  - Interactor viewpoints (customers and account database)
  - Indirect viewpoints (management and security staff.)
  - Domain viewpoints (standards that have been developed for interbank communications)

Requirements discovery and interviewing



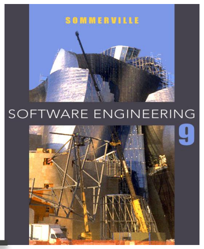
- In formal or informal interviewing, the RE team puts questions to stakeholders about the system that they use and the system to be developed.
- There are two types of interview
  - Closed interviews where a pre-defined set of questions are answered.
  - Open interviews where there is no pre-defined agenda and a range of issues are explored with stakeholders.

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.
- Interviews are ***not*** good for understanding domain requirements for two reasons:
  - 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.

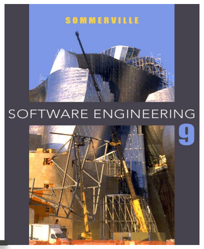
Requirements discovery and scenarios (1/2)



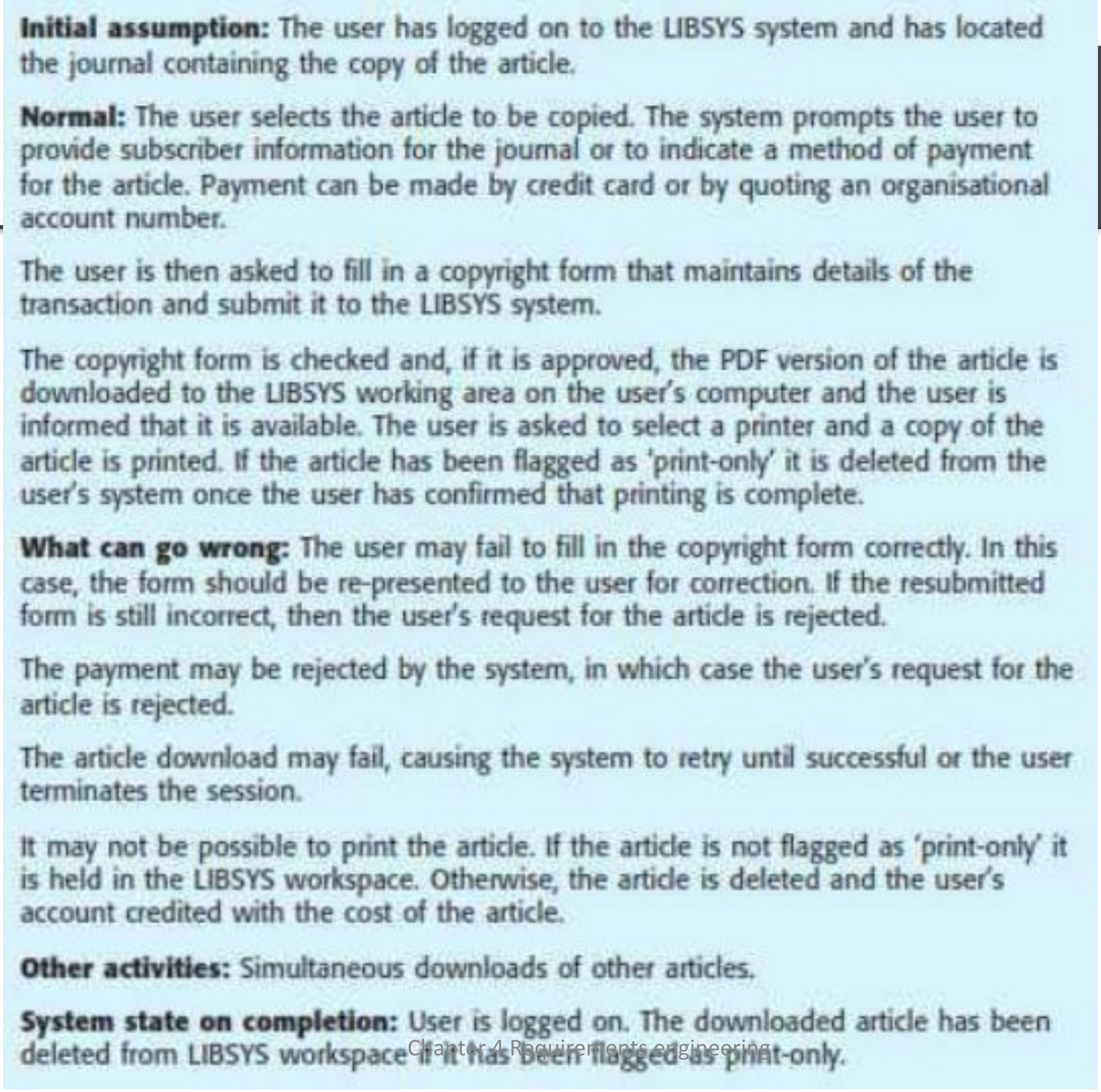
- People usually find it easier to relate to real-life examples than to abstract description.
- They can understand and critique a scenario of how they can interact with the system.
- That is, scenario can be particularly useful for adding detail to an outline requirements description:
  - they are description of example interaction sessions;
  - each scenario covers one or more possible interaction;

Several forms of scenarios have been developed, each of which provides different types of information at different level of detail about the system.

Requirements discovery and scenarios (2/2)



- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.
- Scenarios described as text might be supplemented by some kind of diagrams, screen-shot and so on.
- Alternatively, a more structured approach such as event scenarios and/or use-case may be adopted.



Initial assumption: The user has logged on to the LIBSYS system and has located the journal containing the copy of the article.

Normal: The user selects the article to be copied. The system prompts the user to provide subscriber information for the journal or to indicate a method of payment for the article. Payment can be made by credit card or by quoting an organisational account number.

The user is then asked to fill in a copyright form that maintains details of the transaction and submit it to the LIBSYS system.

The copyright form is checked and, if it is approved, the PDF version of the article is downloaded to the LIBSYS working area on the user's computer and the user is informed that it is available. The user is asked to select a printer and a copy of the article is printed. If the article has been flagged as 'print-only' it is deleted from the user's system once the user has confirmed that printing is complete.

What can go wrong: The user may fail to fill in the copyright form correctly. In this case, the form should be re-presented to the user for correction. If the resubmitted form is still incorrect, then the user's request for the article is rejected.

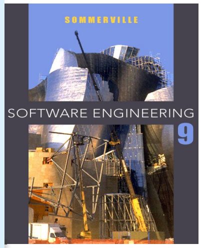
The payment may be rejected by the system, in which case the user's request for the article is rejected.

The article download may fail, causing the system to retry until successful or the user terminates the session.

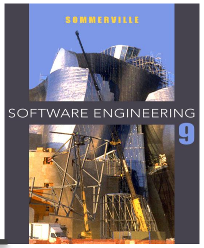
It may not be possible to print the article. If the article is not flagged as 'print-only' it is held in the LIBSYS workspace. Otherwise, the article is deleted and the user's account credited with the cost of the article.

Other activities: Simultaneous downloads of other articles.

System state on completion: User is logged on. The downloaded article has been deleted from LIBSYS workspace if it has been flagged as print-only.

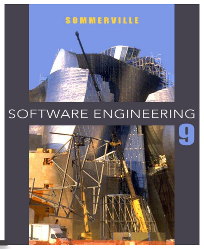


Use-cases and scenarios (1/2)



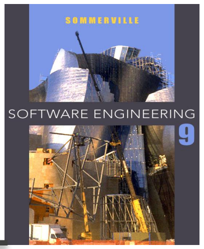
- Scenarios and use-cases are ***effective technique for eliciting*** requirements for ***interactor*** viewpoints. In fact each interaction can be represented as a use-case supplemented by scenarios.
- They may also be used for ***indirect viewpoint*** when these viewpoint receive some result. For example consider the Library Manager of our example.

Use-cases and scenarios (2/2)



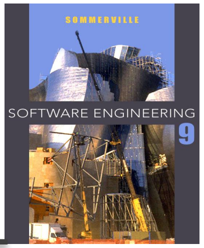
- Due to their *"low-level nature"* in this context (*requirements engineering process*), scenarios and use-cases ***are not so effective***
  - for discovering constraints and/or ***<u>high-level non-functional</u>*** requirements;
  - for discovering ***<u>domain</u>*** requirements.

Requirements validation



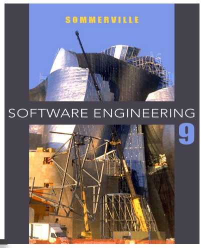
- Concerned with demonstrating that the requirements ***define*** the system that the customer really wants.
- Requirements ***validation covers a part of analysis*** in that it is concerned with ***finding problems*** with requirements.
- ***Requirements error costs are high*** so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.
  - In fact, a change to the requirements usually means that the ***system design*** and the ***implementation*** must also be changed and the ***testing*** has to be performed again.

Checks required during the requirements validation process



- Validity checks. Does the system provide the functions which best support the customer's needs? (Other functions maybe identified by a further analysis)
- Consistency checks. Are there any requirements conflicts?
- Completeness checks. Are all the requirements needed to define all functions required by the customer sufficiently specified?
- Realism checks. Can the requirements be implemented given available budget, technology and schedule?
- Verifiability. Can the requirements be checked?

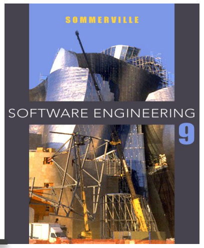
Requirements validation techniques



The following techniques can be used *individually* or in *conjunction*.

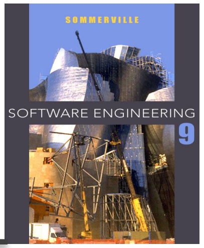
- ***Requirements reviews***
  - Systematic manual analysis of the requirements performed by a team of reviewers.
- ***Prototyping***
  - Using an executable model of the system to check requirements.
- ***Test-case generation***
  - Developing tests for requirements to check testability.
  - If the test is difficult to design, usually the related requirements are difficult to implement.

Requirements management



- The requirements for large systems are frequently changing.
- In fact, during the software process, the ***stakeholders' understanding of the problem*** is constantly changing.
- Requirements management is the process of managing ***changing requirements*** during the ***requirements engineering process*** and ***system development.***
- Requirements are inevitably incomplete and inconsistent
  - New requirements emerge during the process as business needs change and a better understanding of the system is developed;
  - Different viewpoints have different requirements and these are often contradictory.

Requirements management

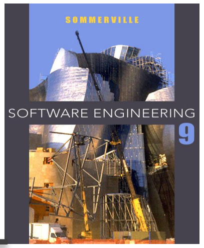


- ***It is hard*** for the users and customers ***to anticipate*** what effects the new system will have on the organization.

- Often, ***only when the system has been deployed***, new requirements inevitably emerge.

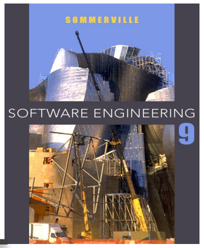
- This is mainly due to the fact that, ***when the end-users have experience of the new system***, they discover new needs and priority.

Requirements change



1. The priority of requirements from different viewpoints changes during the development process. ***Conflicts*** have to inevitably ***converge*** in a ***compromise***.
2. System customers may specify requirements from a ***business perspective that conflict with end-user*** requirements.
3. The ***business and technical*** environment of the system ***changes*** during its development.
4. New hardware, new interface, business priority, new regulations, etc.

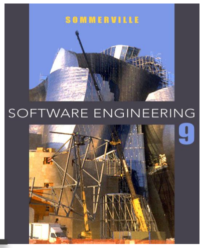
Requirement changes and the requirements management

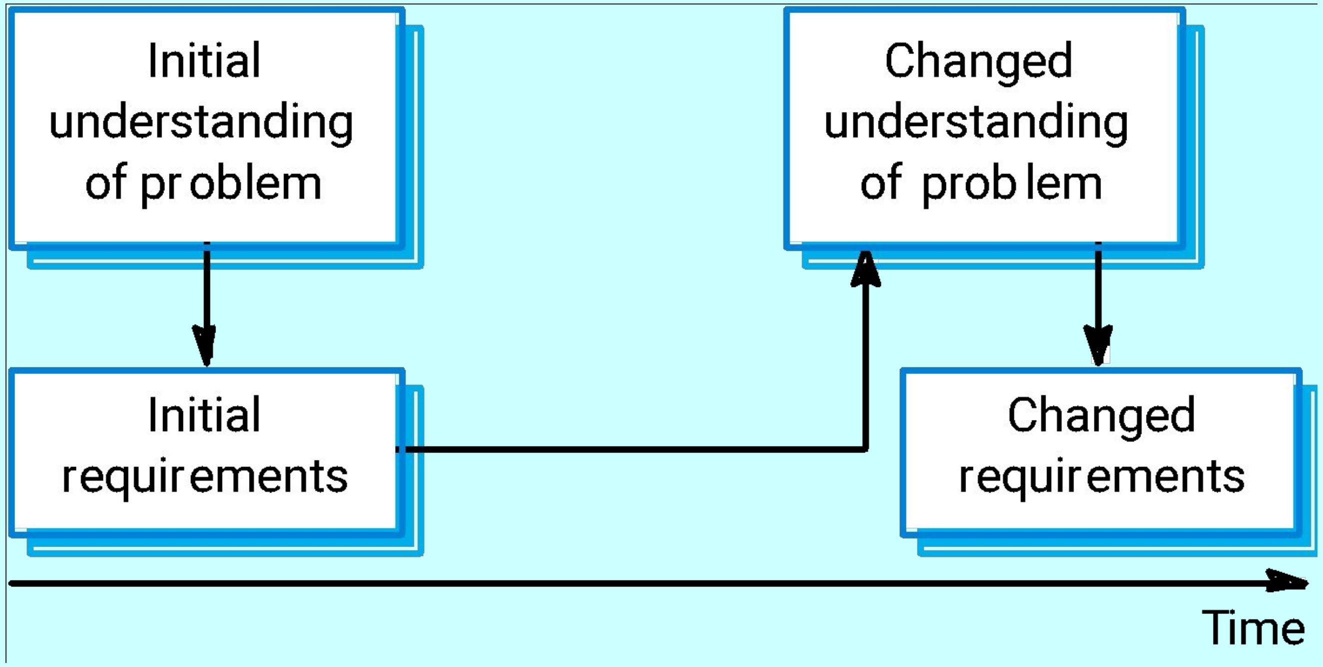


The ***requirements management is the process*** of identifying, understanding and controlling changes to system requirements.

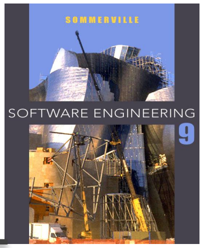
- It might be useful ***to keep track of individual requirements and maintain links*** between dependent requirements so that you can asset the impact of requirements changes.
- The process of requirements management ***should start as soon as*** a draft a version of the requirement document is available.

Requirements evolution



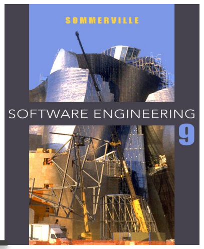

Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

Enduring and volatile requirements



- From an ***evolution perspective***, requirements fall into two classes:
- Enduring requirements. Stable requirements derived from the ***core activity*** of the customer organisation and ***relate directly to the domain*** of the system.
  - E.g., In a hospital, requirements will always relate to doctors, nurses, etc.
  - These requirements may be derived from a ***domain conceptual models*** that show ***entities and relations*** between them.
- Volatile requirements. Requirements which ***change during development or when the system is in use***.
  - E.g., In a hospital, requirements derived from healthcare policy;

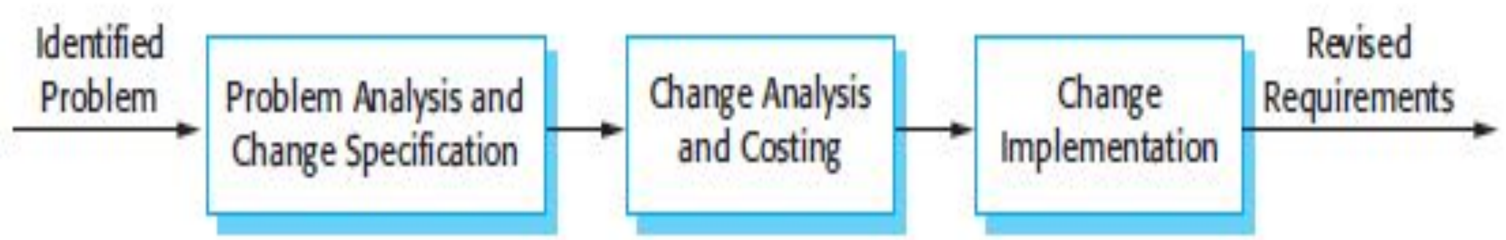
Requirements management planning



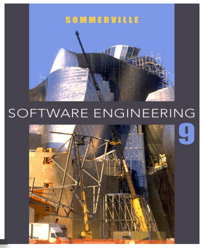
- Since the RE process is very expensive, it might be useful to establish a planning.
- In fact, during the requirements engineering process, you have to plan:
  - ***Requirements identification***
    - How requirements are individually identified; they should be uniquely identified in order to keep a better traceability.
  - ***A change management process***
    - The process followed when requirements change: the ***set of activities*** that estimate the ***impact and cost of changes***.
  - ***Traceability policies***
    - *The policy for managing* the amount of information about relationships between requirements and between system design and requirements that should be maintained (e.g., in a Data Base)
  - ***CASE tool support***
    - The tool support required to help manage requirements change; tolls can range from specialist requirements management systems to simple data base systems.

Change Management Phases

- *Problem analysis and change specification*
- *Change analysis and costing*
- *Change implementation*


Identified Problem
Problem Analysis and Change Specification
Change Analysis and Costing
Change Implementation
Revised Requirements

Key points



- The requirements engineering process includes a feasibility study, requirements elicitation and analysis, requirements specification and requirements management.
- Requirements elicitation and analysis is iterative involving domain understanding, requirements collection, classification, structuring, prioritisation and validation.
- Systems have multiple stakeholders with different requirements.
- Business changes inevitably lead to changing requirements.
- Requirements management includes planning and change management.
- Requirements validation is concerned with checks for validity, consistency, completeness, realism and verifiability.