chapter 1-7 ti kan yayehudar qaba inni dura kun itti dabalameeti malee

Advanced Requirement Engineering SEng-6111

Seffi G. (Assis. Prof)

gseffi2010@gmai.com

Requirement Engineering process Framework

Requirements engineering

- is the branch of software Engineering concerned with the real world goals for, Functions of and constraints on software systems.
- concerned with the relationship of these factors to precise specifications of software behaviour, and to their evolution over time and across software families
- three fundamental concerns of Requirements Engineering
  - the concern of understanding a problem ('what the problem is')
  - the concern of formally describing a problem
  - the concern of attaining an agreement on the nature of the problem.

Requirement Engineering and Requirements

- Requirements discovered at early stages of the software development life cycle.
- requirements describe how the system will behave, system domain, user level facilities and the constraints on which the system should be operated
- Types
  - User requirements
    - describe the expected services from the system, constraints on achieving them and the way the system provides the requirements
    - defined using the external activities and behaviour of the system and is never defined based on system design or implementation

..contd

- System Requirements
  - provide the in-depth knowledge of the user requirements
  - basic principles that should be followed to design the system architecture
  - software engineer should analyse these requirements to know about what exactly has to be implemented and provided in the proposed system
- Types
  - Functional requirements
    - What the system is expected to provide
    - How the system should respond or react to particular input or situation
    - What the system should not do
    - Constraints on implementing the above said requirements

….contd

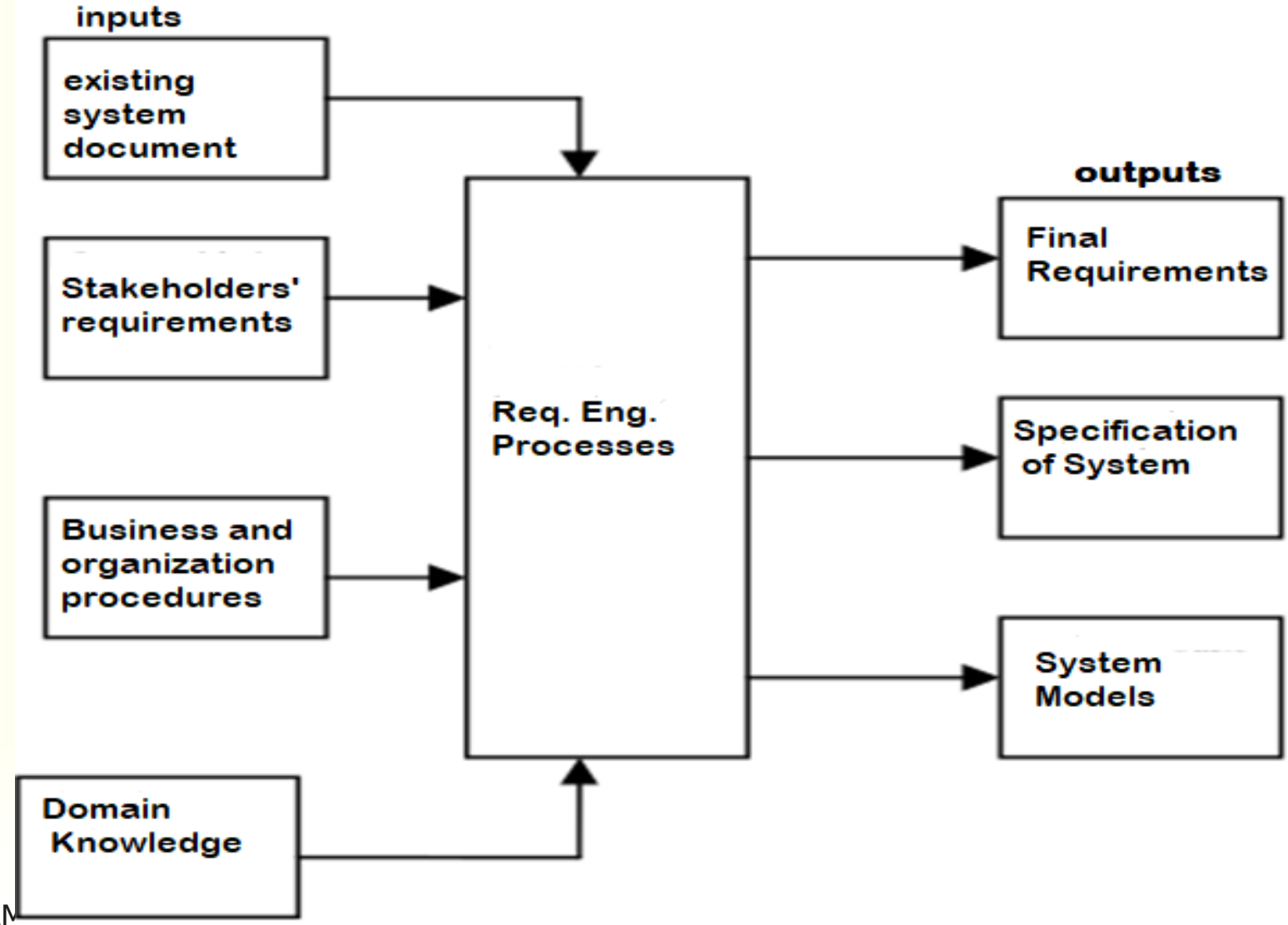
- These requirements mainly depend on the users of the system and the type of software that is developed
- Non functional requirements
  - define the effectiveness of the functions provided by the system
  - not provided by the system, but they affect the functions provided by the system
  - form the basis of the quality of the system
- requirements engineering involves acquisition', 'elicitation', 'analysis', 'specification', 'validation' etc
- The cost of the requirements engineering depends on the size and the type of the system that is being developed. For large systems it wills costs 15% of the total budget only for formal requirement specification, for smaller systems it varies from 8 to 10%

using wrong requirements can have significant negative impacts on the development process and the resulting software system. It is essential to ensure that the requirements are accurate, feasible, verifiable, and consistent to avoid these problems.d text here

problems of using wrong requirements

- Delayed and over budget projects.
- The product does not reach the intended purpose. The customers, who are actually paying for the system, are not satisfied.
- The errors encountered in the development of the system, is the reason for the problems in using the system.
- The continuous use of such system makes it error prone, and thus increases the cost of the maintenance.
- The cost of correcting the requirements errors is 100 times more than the cost of the simple errors occurred in the later stages of the project (Kotonya and Ian Somerville)

Requirement Engineering process Framework



Requirement Engineering process Framework

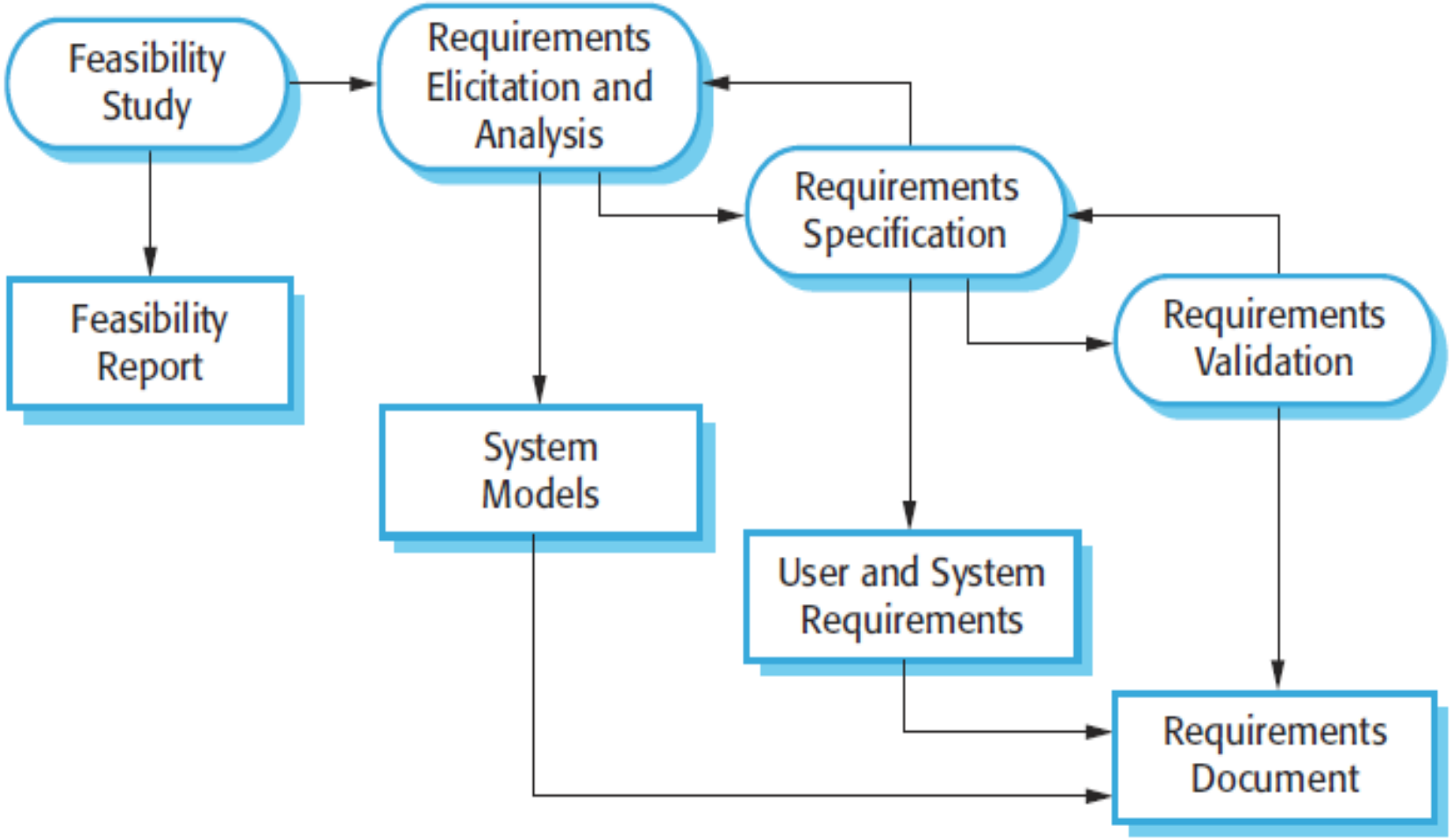
- inputs of the requirements engineering process
  - Existing system document
  - user and stake holders requirements
  - organization and business procedures
  - Domain Knowledge
- outputs for the Requirements engineering process are
  - Final requirements
  - Specification of system
  - System models

- For Library information System(LIS) the following are example requirements for inputs
  - Existing System information
    - The LIS shall poll the bar code reader system and process all of the transaction requests every two seconds
  - Stakeholder needs
    - The system should provide a help facility which will explain the facilities of the system to new user. This help facility should be accessible from all user interface screens
  - Organizational standards
    - The system shall run on a Sun server running the Solaris 2.0 operating system

….contd…

- Regulations
  - The system shall include a facility to a print all of the personnel information which is maintained for a library user
- Domain information
  - Books are uniquely identified by an international Standard Book Number which is a 10 digit identifier

Requirement Engineering Processes



Requirement Engineering process Framework

- Feasibility study
  - necessity of the proposed system in the organization and the domain information of the system
    - Does the overall objectives of the organisation is satisfied by the system?
    - Can the system be developed within the proposed budget and timeline?
    - Can the proposed system made to co-exist with the existing systems?
- This feasibility study involves set of activities such as information gathering, assessment and the report of the information

Requirement Engineering process Framework

- Feasibility study
  - The reaction of the organisation if the proposed system is not implemented
  - Can the existing system integrated with the proposed system to solve the problems.
  - The level of problems that has to be solved by the proposed system.
  - Can the proposed system be developed and used with the existing technology or newer technology is needed?
  - What exactly the proposed systems should and should not support?
- Participants: the total management department, technical professional, expert's judgment and experienced users.

Feasibility study

- What if the system wasn't implemented?
- What are current process problems?
- Do technical resources exist?
- What is the risk associated with the technology?
- Is new technology needed? What skills?
- How will the proposed project help?
- How does the proposed project contribute to the overall objectives of the organization?
- Have the benefits identified with the system being identified clearly?

Feasibility study

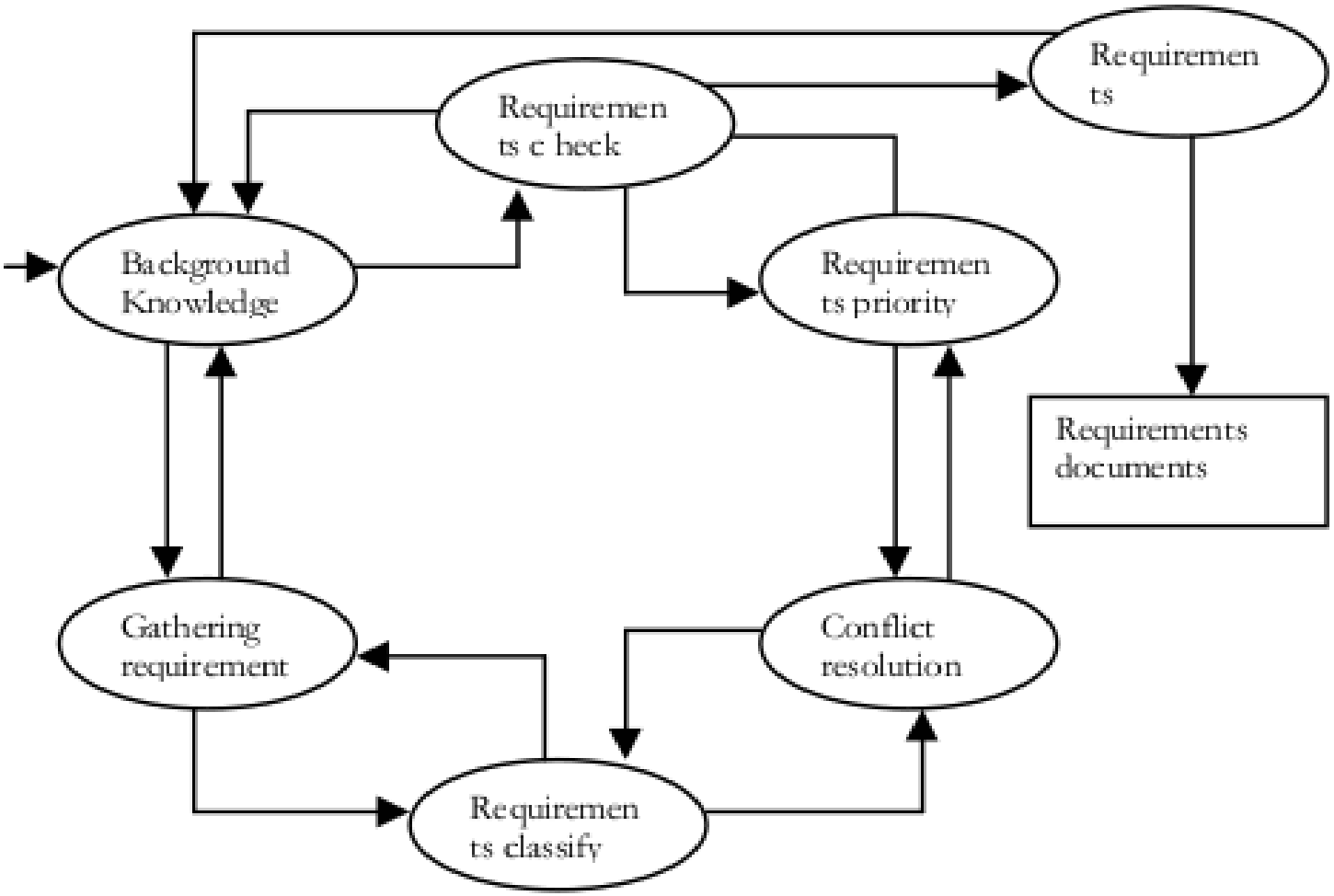
- What will be the integration problems?
- What facilities must be supported by the system?
- What is the risk associated with cost and schedule?
- What are the potential disadvantages/advantages?
- Are there legal issues?
- Are there issues linked with the fact that this is an offshore project?

Feasibility report

Requirements Elicitation and Analysis

- Requirements elicitation is the process of gathering the requirements
- technical professional in the organization, like software developers and the system engineers, work together with the users of the system and the customers
- What the proposed system should provide?
- What are the expected services from the system?
- What are the required characteristics of the system?
- What are the required hardware and software constraints of the system?

Requirements Elicitation and Analysis



Requirements Elicitation and Analysis challenges

- Lack of technical knowledge and unawareness of the technical aspects of the system from the stake holder's side.
- Some times they may demand unrealistic things and they do not know what they exactly expecting form the system.
- Stakeholders express their requirements in the most general terms; it is difficult to find the technical aspects of the system form the general terms.
- Different people expect different services from the proposed system, requirements engineers has to discover the good sources of requirements and commonalities.
- Business procedures, political influences and the budget matters where the clear analysis is required.

Requirements Elicitation Techniques

Various techniques have emerged for determining a customer's needs

- Traditional: Questionnaires, Interviews, Process Analysis
- Group-Oriented: Focus Groups, Brainstorming, Workshops
- Early Feedback: Prototyping (High- and Low-Fidelity)
- Model-Driven: Goal-Based, Scenario-Based
- Cognitive: Protocol Analysis, Laddering, Card Sorting
- Contextual: Ethnography, Conversation Analysis
  - ***Reading assignment- Acquire a clear understanding of the aforementioned elicitation techniques***

hikaaa prototype fi akkaku prototype haala salpa ta'een ibsa

Prototyping is a technique used in software development to create a preliminary version of a software system. Prototypes can be used to explore design options, test functionality, and gather feedback from stakeholders. There are two main types of prototypes: high-fidelity and low-fidelity.

High-fidelity prototypes are detailed and realistic representations of the final software system. They are often fully functional and can be used to test the user interface, functionality, and performance of the system. High-fidelity prototypes are typically more time-consuming and expensive to create than low-fidelity prototypes, but they provide a more accurate representation of the final product.

Low-fidelity prototypes , on the other hand, are simple and abstract representations of the final software system. They are often created using paper sketches, wireframes, or basic mockups, and are used to explore design options and gather feedback from stakeholders. Low-fidelity prototypes are typically faster and less expensive to create than high-fidelity prototypes, but they provide a less accurate representation of the final product.

Both high- and low-fidelity prototypes can be useful in the software development process. Low-fidelity prototypes can be used early in the process to explore design options and gather feedback from stakeholders, while high-fidelity prototypes can be used later in the process to test functionality and refine the design.

Analysis of Existing Systems

Analysis of Existing Systems

Useful when building a new improved version of an existing system

Important to know:

- What is used, not used, or missing
- What works well, what does not work
- How the system is used (with frequency and importance) and it was supposed to be used, and
- how we would like to use it

Analysis of Existing Systems

Why analyze an existing system?

- Users may become disillusioned with new system or do not like the new system if it is too different or does not do what they want (risk of nostalgia for old system)
- To appropriately take into account real usage patterns, human issues, common activities, relative importance of tasks/features
- To catch obvious possible improvements (features that are missing or do not currently work well)
- To find out which "legacy" features can/cannot be left out

Review Available Documentation

- Start with reading available documentation
  - User documents (manual, guides…)
  - Development documents
  - Requirements documents
  - Internal memos
  - Change histories
  - …

Of course, often these are out of date, poorly written, wrong, etc., but it's a good starting point

Discourse analysis

Use of words and phrases is examined in written or spoken language

Social analysis

- Observation is the method of collecting requirements by observing the people doing their normal work
- when the user is unable to explain their expected requirements from the new product and problems with the existing product
- Types
  - Passive observations
  - Active observation
  - Explanatory Observations
  - Ethnography
    - attempts to discover social, human, and political factors, which may also impact requirements

Social analysis

Can be supplemented later with questionnaires

- Based on what you know now – the results of observation
- To answer questions that need comparison or corroboration (confirmation)
- To obtain some statistics from a large number of users (look for statistical significance!), e.g.:
  - *How often do you use feature X?*
  - *What are the three features you would most like to see?*

Can be supplemented later with interviews

- After getting a better idea of what is to be done, probably some questions require more detailed answers
- You will not be wasting other people's time or your own
- This is very labour intensive!

Interviews

Interviews

- Requires preparation and good communication management
  - Achieve interview objectives without preventing the exploration of promising leads
- Interview as many stakeholders as possible
  - Not just clients and users
- Ask problem-oriented questions
  - Ask about specific details, but…
  - Detailed and solution-specific questions may miss the stakeholder's real requirements. Example:
    - Would you like Word 2007, Excel 2007 or both?

      vs.
    - Would you like to do word processing, computations, or both?

Interviews

Three main objectives:

- Record information to be used as input to requirements analysis and modeling
- Discover information from interviewee accurately and efficiently
- Reassure interviewee that his/her understanding of the topic has been explored, listened to, and valued

Process consists of four important steps:

- Planning and preparation
- Interview session
- Consolidation of information
- Follow-up

Many strategies for questioning

Interviews

Important to plan and prepare interviews

Set goals and objectives for the interview

Acquire background knowledge of the subject matter to conduct an effective interview

- About the domain (vocabulary, problems...) but also about the interviewee (work tasks, attitude...)

Prepare questions in advance, by subject

Organize the environment for conducting an effective interview

- Determine how the elicitation notes will be taken (manually, audio, video, by whom…)

Interviews

Make the interviewee comfortable and confident

Be polite and respectful!

Adjust to the interviewee

You have your goals – be persistent but flexible

Interview several people at once to create synergy

Try to detect political aspects as they may influence the said and the unsaid

Interviews

Revise and complete the elicitation notes after the interview

- Needs to be done soon after because one forgets the details (and everything else)

Identify inconsistencies and address them in a follow-up interview or by email

Keep all diagrams, charts, models created during the discussions

You are learning, so be precise(show example from SUST)

- Pay attention to terminology
- Use the interviewee's terminology
- Identify synonyms
- Create a glossary if necessary

Common Interviewing Mistakes

Not interviewing all of the right people

- Different points of view of stakeholders

Asking direct questions too early

- e.g., design of a transportation system:
  - How much horsepower do you need? (direct)
  - How many passengers? How far? How fast? (indirect)
- e.g., camera design for novice photographer:
  - How important is control over shutter speed and aperture? (direct)
  - Will you be taking action shots, still shots, or both? (indirect)

Interviews – Start Up Questions

Context-free questions to narrow the scope a bit (Weinberg)

Identify customers, goals, and benefits

- Who is (really) behind the request for the system?
- Who will use the system? Willingly?
- Are there several types of users?
- What is the potential economic benefit from a successful solution?
- Is a (pre-existing) solution available from another source?

Interviews – Start Up Questions

When do you need it by?

Can you prioritize your needs?

What are your constraints?

- Time
- Budget
- Resources (human or otherwise)

Expected milestones (deliverables and dates)?

Interviews – Start Up Questions

Try to characterize the problem and its solution

- What would be a "good" solution to the problem?
- What problems is the system trying to address?
- In what environment will the system be used?
- Any special performance issues?
- Other special constraints?
- What is (un)likely to change?
- Future evolution?
- What needs to be flexible (vs. quick & dirty)?

Interviews – Start Up Questions

Calibration and tracking questions

- Are you the right person to answer these questions?
- Are your answers "official"? If not, whose are?
- Are these questions relevant to the problem as you see it?
- Are there too many questions? Is this the correct level of detail?
- Is there anyone else I should talk to?
- Is there anything else I should be asking you? Have you told me everything you know about the problem?
- Do you have any questions?

Interviews – Start Up Questions

Questions that cannot be asked directly (ask indirectly)

Are you opposed to the system?

Are you trying to obstruct/delay the system?

Are you trying to create a more important role for yourself?

Do you feel threatened by the proposed system?

Are you trying to protect your job?

Is your job threatened by the new system?

Is anyone else's?

Interviews – Specific Questions

Functional requirements

- What will the system do?
- When will the system do it?
- Are there several modes of operations?
- What kinds of computations or data transformations must be performed?
- What are the appropriate reactions to possible stimuli?
- For both input and output, what should be the format of the data?
- Must any data be retained for any period of time?

Interviews – Specific Questions

Design Constraints

Physical environment

- Where is the equipment to be located?
- Is there one location or several?
- Are there any environmental restrictions, such as temperature, humidity, or magnetic interference?
- Are there any constraints on the size of the system?
- Are there any constraints on power, heating, or air conditioning?
- Are there constraints on the programming language because of existing software components?

Interviews – Specific Questions

Design Constraints

Interfaces

- Is input coming from one or more other systems?
- Is output going to one or more other systems?
- Is there a prescribed way in which input/output need to be formatted?
- Is there a prescribed way for storing data?
- Is there a prescribed medium that the data must use?

Standards

- Are there any standards relevant to the system?

Interviews – Specific Questions

Performance

Are there constraints on execution speed, response time, or throughput?

What efficiency measure will apply to resource usage and response time?

How much data will flow through the system?

How often will data be received or sent?

Interviews – Specific Questions

Usability and Human Factors

What kind of training will be required for each type of user?

How easy should it be for a user to understand and use the system?

How difficult should it be for a user to misuse the system?

Interviews – Specific Questions

Security

Must access to the system or information be controlled?

Should each user's data be isolated from data of other users?

Should user programs be isolated from other programs and from the operating system?

Should precautions be taken against theft or vandalism?

Interviews – Specific Questions

Reliability and Availability

Must the system detect and isolate faults?

What is the prescribed mean time between failures?

Is there a maximum time allowed for restarting the system after failure?

How often will the system be backed up?

Must backup copies be stored at a different location?

Should precautions be taken against fire or water damage?

Interviews – Specific Questions

Maintainability

Will maintenance merely correct errors, or will it also include improving the system?

When and in what ways might the system be changed in the future?

How easy should it be to add features to the system?

How easy should it be to port the system from one platform (computer, operating system) to another?

Interviews – Specific Questions

Precision and Accuracy

How accurate must data calculations be?

To what degree of precision must calculations be made?

Types

- Closed Interview: In this interview the requirements engineer prepare some predefined questions and he tries to get the answers for these questions from the stake holders.
- Open interview: In this interview the requirements engineer does not prepare any predefined questions, and he tries to get the information from the stakeholders in open discussions.

Questionnaire

- Questionnaires are one of the methods of gathering requirements in less cost .
- Questionnaires reach a large number of people, not only in less time but also in a lesser cost.

result from the questionnaires mainly depends on the two factors

- Effectiveness and the design of the questionnaire
- Honesty of the respondent.

Questionnaire

The steps involved in designing and administering a questionnaire are

- The purpose of the survey should be defined
- The sampling group (respondents to the survey) should be decided
- Preparing and developing the Questionnaire
- Conducting the Questionnaire process
- Gathering and analysing the results

Questionnaire

Steps in arranging a questionnaire

- The questions should be arranged well, so that general questions are followed by articular questions.
- Arrange the questions such that, easy questions comes first.
- Arrange the questions in a order of known to unknown
- Try to use closed format questions in the beginning.
- The questions relevant to the main subject should be given high priority and stated at the start of the questionnaire.
- Avoid personal and intimate questions at the beginning

Questionnaire

factors which affect the usage of the questionnaire

- The available resources to gather the requirements
- Type of Requirements that has to be gathering
- Anonymity provided to the respondent

Prototyping

- Prototype is the representations or visualizations of the actual system parts
- designed in the early stages of the implementation of the project.
- provides the General idea of the actual system functions and the work flow
- A prototype represents the actual product in both functional and graphical sense

Prototyping

A software requirements prototype is a mock-up or partial implementation of a software system

- Helps developers, users, and customers better understand system requirements
- Helps clarify and complete requirements
- Provides early response to “I'll know it when I’ll see (or won’t see) it” attitude
- Effective in addressing the “Yes, But” and the “Undiscovered Ruins” syndromes
- Helps find new functionalities, discuss usability, and establish priorities
- Focus prototype development on these uncertain parts
- Encourages user participation and mutual understanding

Prototyping

- When the users are unable to express their requirements.
- If it is a new product and the users have no experience with this product.
- When ever the requirements analysis and feasibility studies is difficult.
- Types
  - Throw-away prototypes: This type of prototype is not reusable and hence is discarded when ever the requirements elicitation process is complete.
  - Evolutionary prototypes: This type of prototypes is reusable. They are evolved or improved according to the feedback and is given as the original product.

Prototyping

Horizontal: focus on one layer – e.g., user interface

Vertical: a slice of the real system

Evolutive: turned into a product incrementally, gives users a working system more quickly (begins with requirements that are more understood)

Fidelity is the extent to which the prototype is real and (especially) reactive

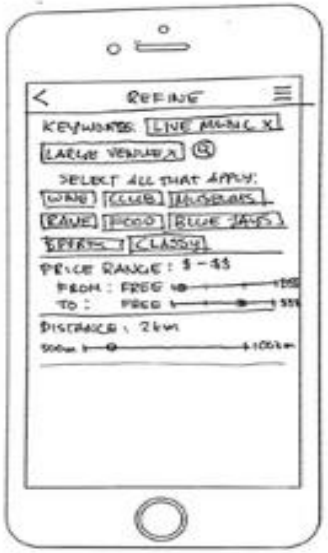
Fidelity may vary for throw-away prototypes

High-fidelity

Low-fidelity

Prototype Stage

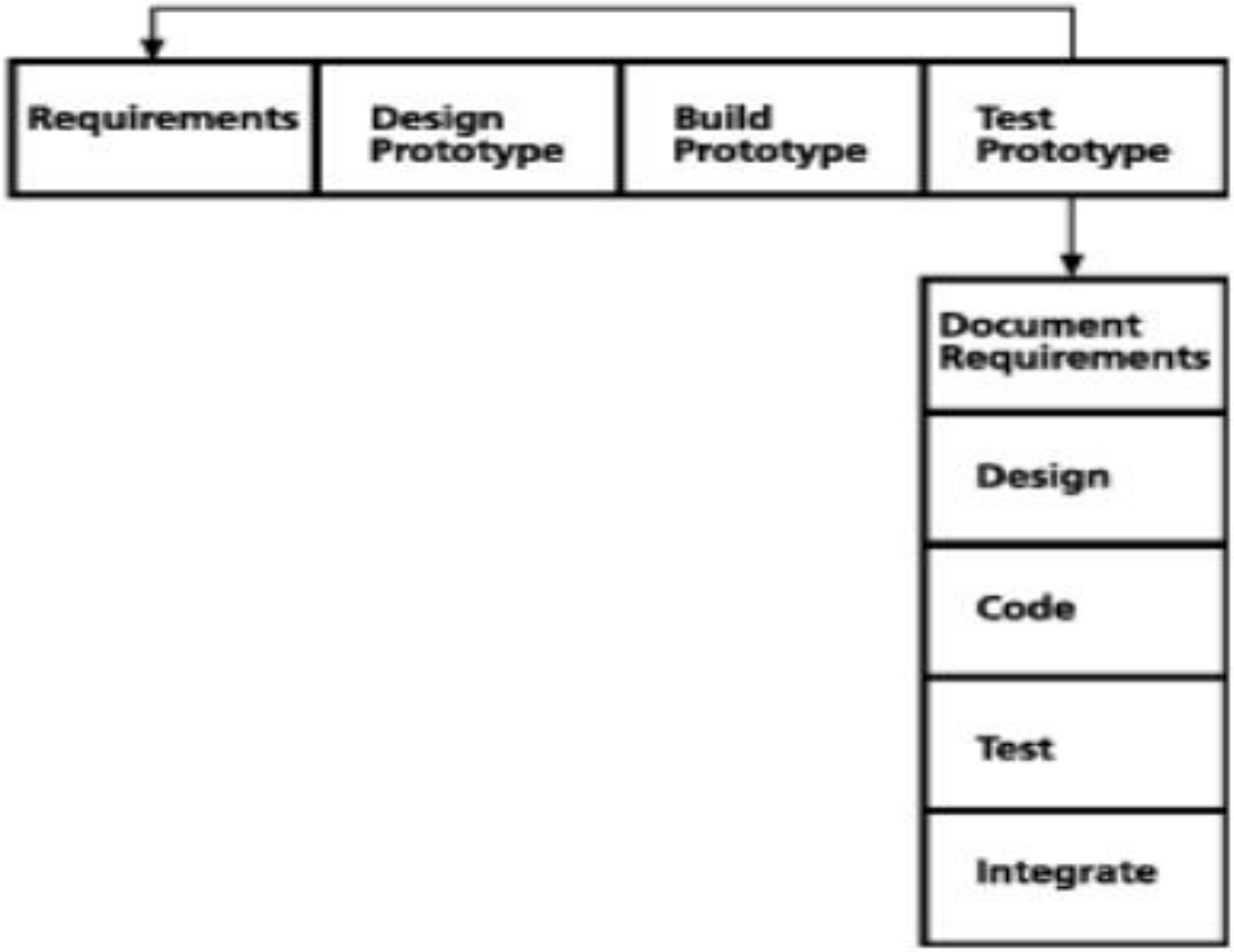
Sketch



Low Fidelity

High Fidelity

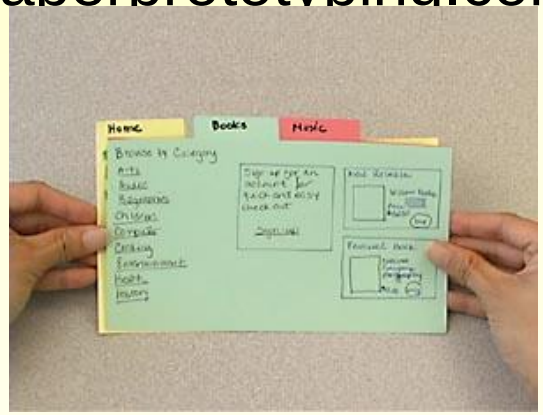
Prototyping



Prototyping – Realizations

Prototypes can take many forms:

- Paper prototypes (see http://www.paperprototyping.com/)
  - Prototype on index card
  - Storyboard
- Screen mock-ups
- Interactive prototypes
  - Using high-level languages (e.g., Visual Basic, Delphi, Prolog)
  - Using scripting languages (e.g., Perl, Python)
  - Using animation tools (e.g., Flash/Shockwave)
- Models (executables)
- Pilot systems



…

Prototyping

Advantages of Prototyping

- Reduces time of development.
- Reduces cost of development.
- The users provided with a visual representation, thus facilitating system implementation.
- Provides high level of user satisfaction.
- The ways in which the system can be enhanced in future is known.

Prototyping

Disadvantages of Prototyping

- The users may expect the finished product to be the same as the prototype
- Developers may be tempted to stop with the prototype.
- Can lead to unfinished system implementation.

Prototyping – Risks

Prototypes that focus on user-interface tends to lose the focus of demonstrating/exploring functionality

Prototypes can bring customers' expectations about the degree of completion unrealistically up

Do not end-up considering a throwaway prototype as part of the production system

- Always clearly state the purpose of each prototype before building it

Brainstorming

- Brainstorming is a techniques used to generate new ideas and find the solution to a specific issue
- This is conducted as a conference with six to ten members.
- The members are from the different departments and domain experts are also included.
- This conference is headed by the organizer, who states the issue to be discussed.

Brainstorming

Brainstorming can be used to answer the following questions

What exactly the system should provide

What are the business and organization rules required to follow

What type of questions should be there in the interviews and questionnaires?

What are the risk factors effect the proposed system development and what to do avoid that?

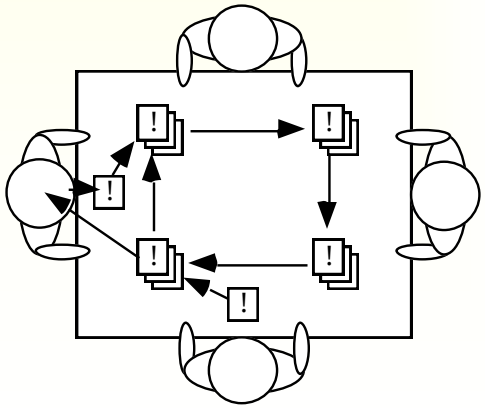
Brainstorming

To invent new way of doing things or when much is unknown

- When there are few or too many ideas
- Early on in a project particularly when:
  - There is little expertise for the type of applications
  - Innovation is important (e.g., novel system)

Two main activities:

- The Storm: Generating as many ideas as possible (quantity, not quality) – wild is good!
- The Calm: Filtering out of ideas (combine, clarify, prioritize, improve…) to keep the best one(s) – may require some voting strategy



Roles: scribe, moderator (may also provoke), participants

Brainstorming – Objectives

Hear ideas from everyone, especially unconventional ideas

- Keep the tone informal and non-judgemental
- Keep the number of participants “reasonable“ – if too many, consider a “playoff “-type filtering and invite back the most creative to multiple sessions

Encourage creativity

- Choose good, provocative project name.
- Choose good, provocative problem statement
- Get a room without distractions, but with good acoustics, whiteboards, coloured pens, provide coffee/donuts/pizza/beer
- Provide appropriate props/mock-ups

Brainstorming – Roles

Scribe

- Write down all ideas (may also contribute)
- May ask clarifying questions during first phase but without criticizing

Moderator/Leader

- Cannot be the scribe
- Two schools of thought: traffic cop or agent provocateur
- Traffic cop – enforces "rules of order", but does not throw his/her weight around otherwise
- Agent provocateur – traffic cop plus more of a leadership role, comes prepared with wild ideas and throws them out as discussion wanes

Brainstorming – Voting on Ideas

Voting with threshold

- Each person is allowed to vote up to n times
- Keep those ideas with more than m votes
- Have multiple rounds with smaller n and m

Voting with campaign speeches

- Each person is allowed to vote up to j < n times
- Keep those ideas with at least one vote
- Have someone who did not vote for an idea defend it for the next round
- Have multiple rounds with smaller j

Joint Application Design (JAD)

A more structured and intensive brainstorming approach

Developed at IBM in the 1970s

- Lots of success stories

"Structured brainstorming", IBM-style

- Full of structure, defined roles, forms to be filled out...

Several activities and six (human) roles to be played

JAD session may last few days

The whole is more than the sum of its parts. The part is more than a fraction of the whole.[1]

Joint Application Development

- This is conducted in the same manner as brainstorming, except that the stakeholders and the users are also allowed to participate and discuss on the design of the proposed system

The success of the JAD depends on [47,58]

- leader of the JAD session
- Developers, end-users and the stakeholders of the product.
- Group involvement.

Joint Application Design – Four Main Tenets

Effective use of group dynamics

- Facilitated and directed group sessions to get common understanding and universal buy-in

Use of visual aids

- To enhance understanding, e.g., props, prepared diagrams

Defined process

- I.e., not a random hodgepodge

Standardized forms for documenting results

Joint Application Design – Activities

Preparation

- Pre-session Planning
- Pre-work

Working Session

Summary

- Follow-up
- Wrap-up

The 6 “P”s

1. Purpose - Why do we do things? (Goals, needs, motivation)
2. Participants - Who is involved? (People, roles, responsibilities)
3. Principles - How do we function? (Guidelines, working agreements, ground rules)
4. Products - What do we create? (Deliverables, decisions, plans, next steps)
5. Place - Where is it located? (Venue, logistics)
6. Process - When do we do what? (Activities, sequence)

Joint Application Design – Roles

Session leader

- Organizer, facilitator, JAD expert
- Good with people skills, enthusiastic, sets tone of meeting

Analyst

- Scribe++
- Produces official JAD documents, experienced developer who understands the big picture, good philosopher/writer/organizer

Executive sponsor

- Manager who has ultimate responsibility for product being built
- Provides strategic insights into company's high-level goals/practices, makes executive decisions later on as required

Joint Application Design – Roles

User representatives

- Selection of knowledgeable end-users and managers
- Come well-prepared with suggestions and ideas of needs, will brainstorm for new or refined ideas, eventually review completed JAD documents

Information system representatives

- Technical expert on ISs
- Helps users think big, know what is easy/hard/cheap/expensive, mostly there to provide information rather than make decisions

Specialists

- Technical expert on particular narrow topics, e.g., security, application domain, law, UI issues…

User Centered Design

- the user acts as the part of the development team. The user takes active part in the designing and development of the system

User Centered Design

Advantages of User Centred Design

- The user is closely involved with the development
- The user feedback can be obtained immediately
- Reduces time and cost spent on gathering user feedback.

Goal-Based Modeling

By tying each software requirement to a specific objective that the software needs to meet, developers avoid over-specifying or missing actual requirements.

Shortcoming: Articulating the goals of the software might be difficult and time-consuming.

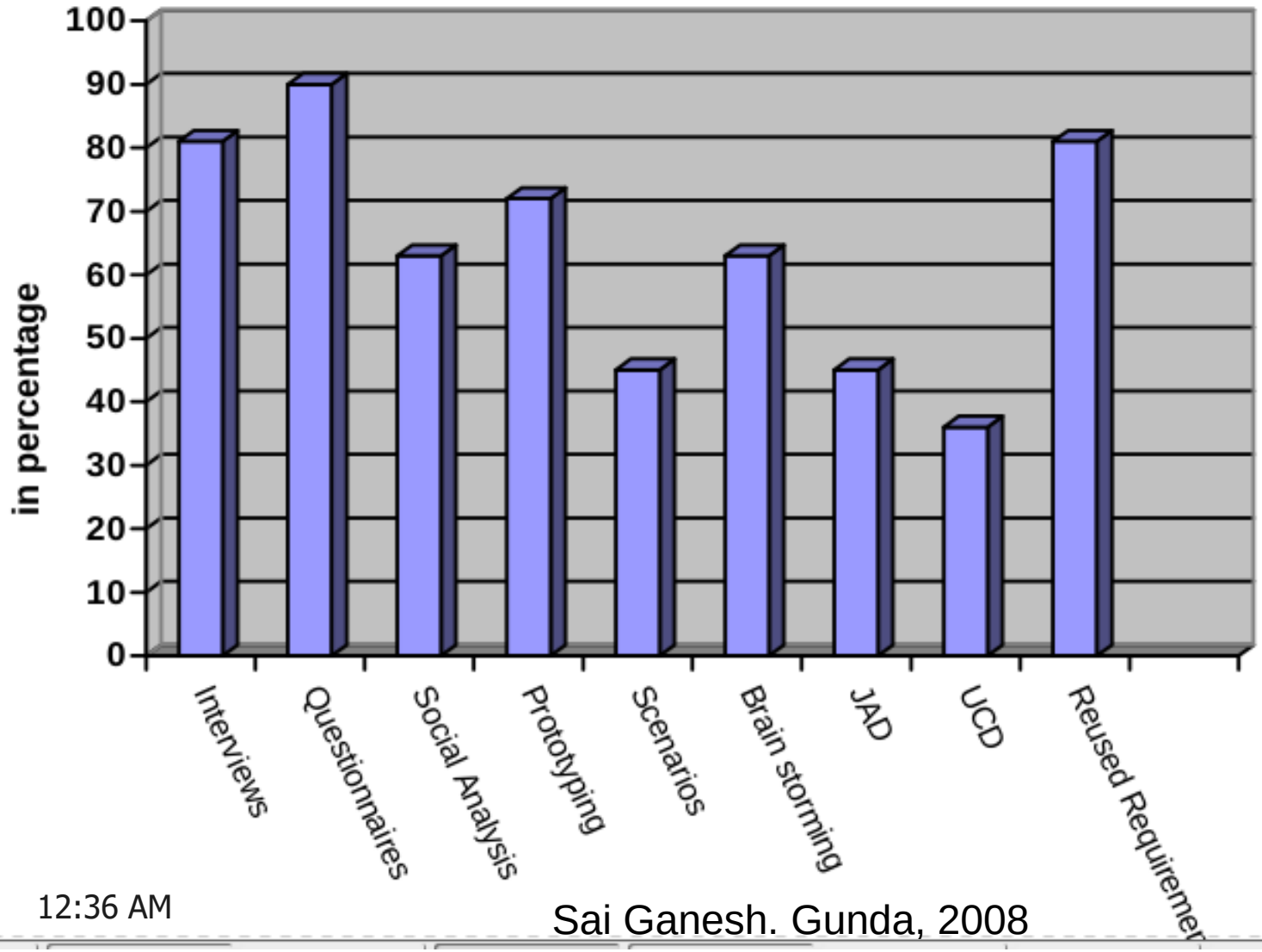
Scenario-Based Modeling

By expanding use cases into full narratives about how particular users will utilize the software system, the requirements of the system become clearer.

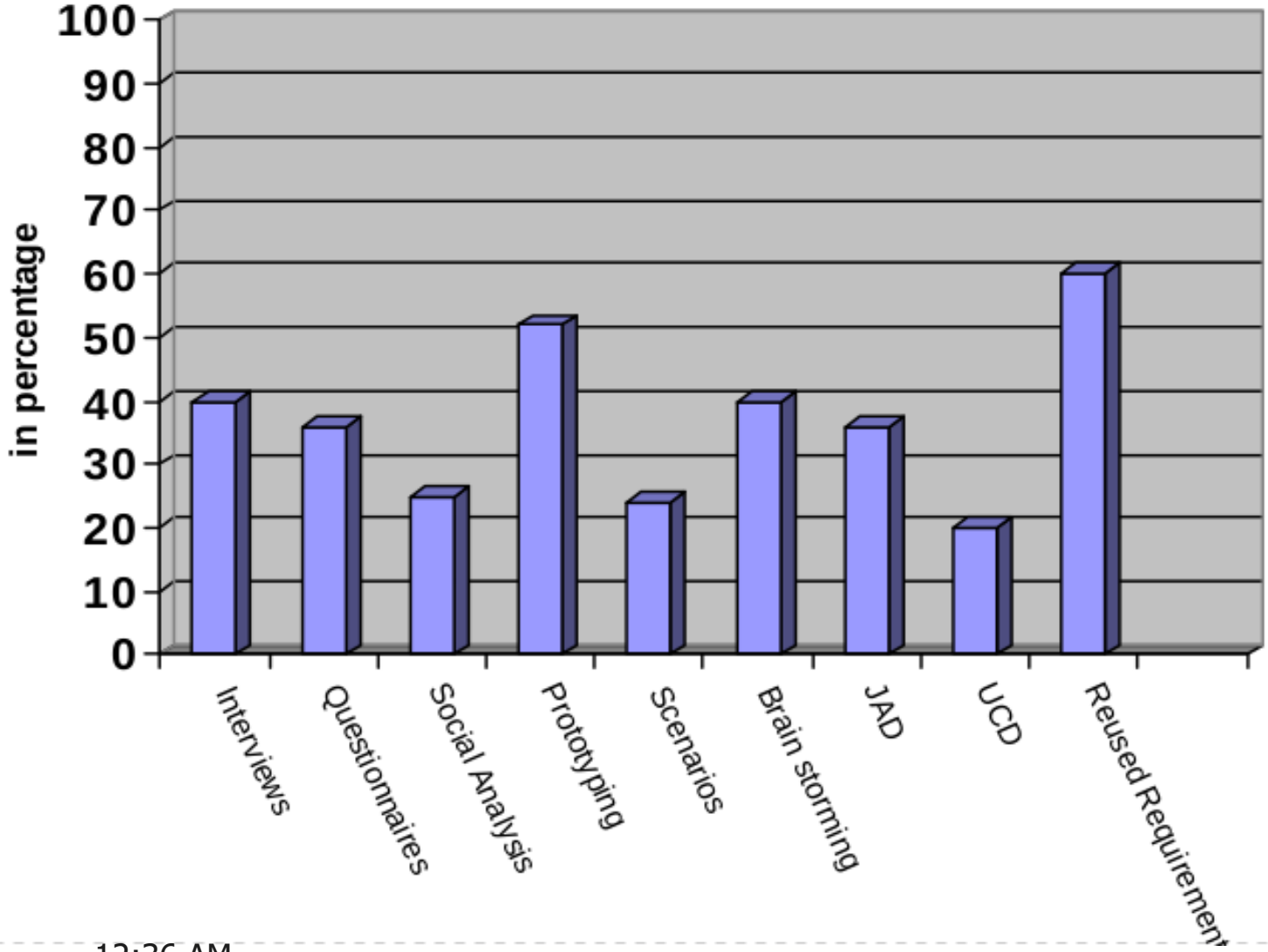
Shortcoming: Developing these scenarios can be an arduous process, involving extensive creativity

Pros and cons of req. elicitation techniques

Effectiveness of the method



Method Popularity in Industrial Usage



Thank you!

?

Requirement Specification

A specification can be viewed as a contract between users and software developers which defines the desired functional behaviour of the software artefact without showing how such functionality is going to be achieved.

Purpose of requirements specification

- producing a formal specification model
  - The specification model is used as an agreement between the software developers and the users on what constitutes the problem which must be solved by the software system
  - the specification model is also a blueprint for the development of the software system.

Input to requirements specification

- input knowledge about the problem domain supplied by the elicitation process

- input knowledge comes in a 'raw' format which must be converted to meaningful information in order to produce a formal specification model

Activities and techniques in requirements specification

- analysis and assimilation of requirements knowledge
- The process is analytical
  - because all different kinds of knowledge that the analyst elicits from the problem domain must be examined and cross-related
- synthesis and organization of the knowledge into a coherent and logical requirements model.
- Specification is synthetic
  - because the heterogeneous knowledge must be combined into producing a logical and coherent whole

Deliverables of requirements specification process

Requirements specification produces:

- user-oriented models specifying the understanding between the analyst, the customer and the user

- developer-oriented models specifying functional and non-functional properties of the software system as well as constraints on resources, design etc.

Interactions processes between requirements specification and other Requirment Engineering Process

- Requirements specification is the central process of Requirements Engineering.
- Specification controls both the elicitation and validation processes

Requirement Validation

Requirement Validation

- is an ongoing process of RE which aims to ensure that the right problem is being tackled at any time.

- the process which certifies that the requirements model is consistent with customers' and users' intents.

- The need for validation appears at the moment a new piece of information is assimilated in the current requirements model and also when different pieces of information are integrated into a coherent whole
- Any requirements model (formal or informal) is subject to validation and thus provides an input to the process

- Knowledge about the problem domain must be validated, i.e. checked for accuracy, consistency, relevance to the project etc.

- some part of the formal requirements model must be validated in parts and as a whole.

Activities and techniques in requirements Validation

Validation is a process which requires interactions between analysts, customers of the intended system and users in the problem domain.

- preparing the settings for an experiment
- performing the experiment and analysing its results

Deliverables of requirements Validation process

- Validation delivers a requirements model which is consistent and in accordance with the users expectations.

Interactions processes between requirements Validation and other Requirment Engineering processes

- Validation is ever present in all stages of Requirements Engineering. The need for validation is triggered by the acquisition of new knowledge about the problem domain (elicitation), or by the formulation (even in part) of a requirements model (specification).

- Validation is also needed during the analysis and synthesis phases of requirements specification, since information analysed must be checked for correctness and synthesised requirements must be checked for logical consistency and coherence.

Guideline on What to Validate in requirement model

- internal consistency
- non-ambiguity
- external consistency
  - External Consistency is the agreement between what is stated in the requirements model and what is true in the problem domain
- minimality
- completeness
- redundancy

Techniques for Validating Requirements

- Prototyping provides the users with a mock-up version of the software system as a means of acquiring their correct requirements.
- Simulation and animation are techniques for 'bringing life' into the requirements model, taking it through different states and generally testing it with under different scenaria in order to see how well it copes in real-life like situations.
- Natural language paraphrasing is a technique for translating the requirements model in the user's own language (i.e. natural language) in hope that this will make it easier for the user to judge the validity of the requirements.
- Expert system approaches to validation introduce the concept of an assistant , i.e. a system
- containing knowledge about the problem domain which it uses in order to bring to the

- attention of the human analyst contradictions, omissions, unresolved issues

Techniques for Validating Requirements

- Expert system approaches to validation introduce the concept of an assistant , i.e. a system containing knowledge about the problem domain which it uses in order to bring to the attention of the human analyst contradictions, omissions, unresolved issues etc. about the requirements model.

Reading Assignment

Introduction to UML

Introduction to UML

- UML is a notation that resulted from the unification of OMT (Object Modeling Technique [Rumbaugh et al., 1991]), Booch [Booch, 1994], and OOSE (Object-Oriented Software Engineering [Jacobson et al., 1992])
- The goal of UML is to provide a standard notation that can be used by all object-oriented methods and to select and integrate the best elements of precursor notations.

Introduction…

- UML has been designed for a broad range of applications (distributed systems, analysis, system design, deployment)
- System development focuses on three different models of the system

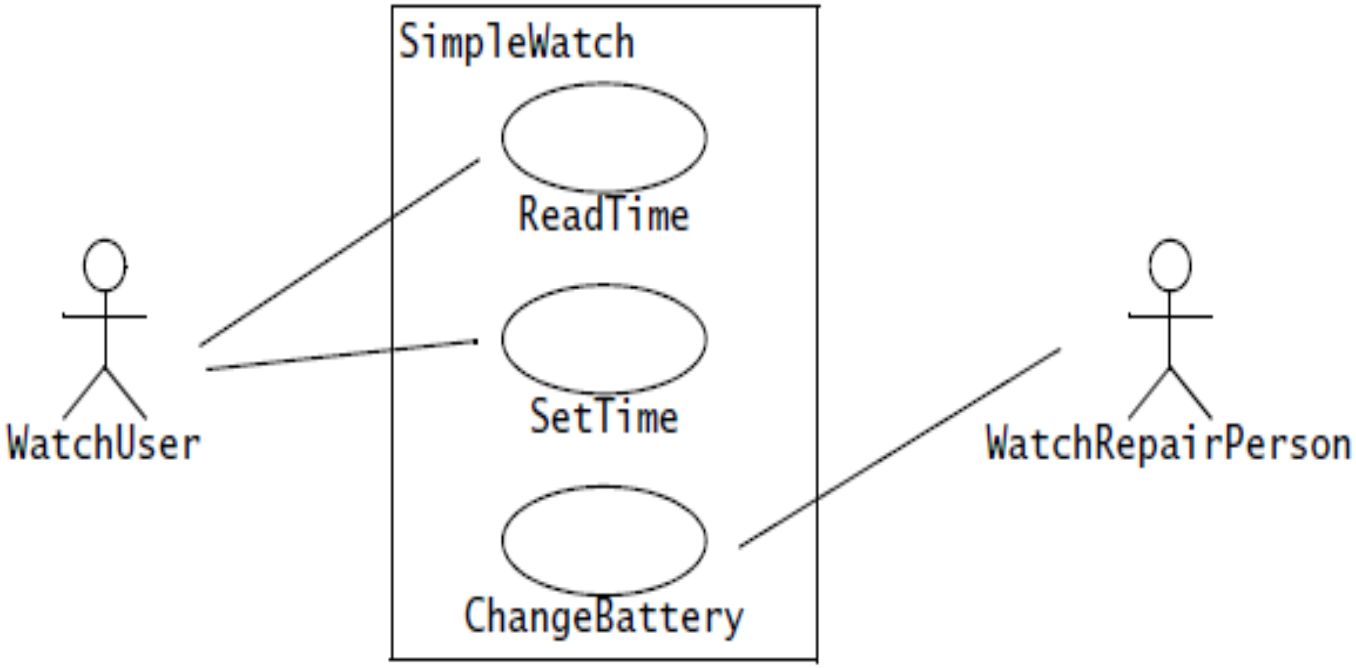
Where Can the UML Be Used?

- The **functional model**
- The **object model**
- The **dynamic model**

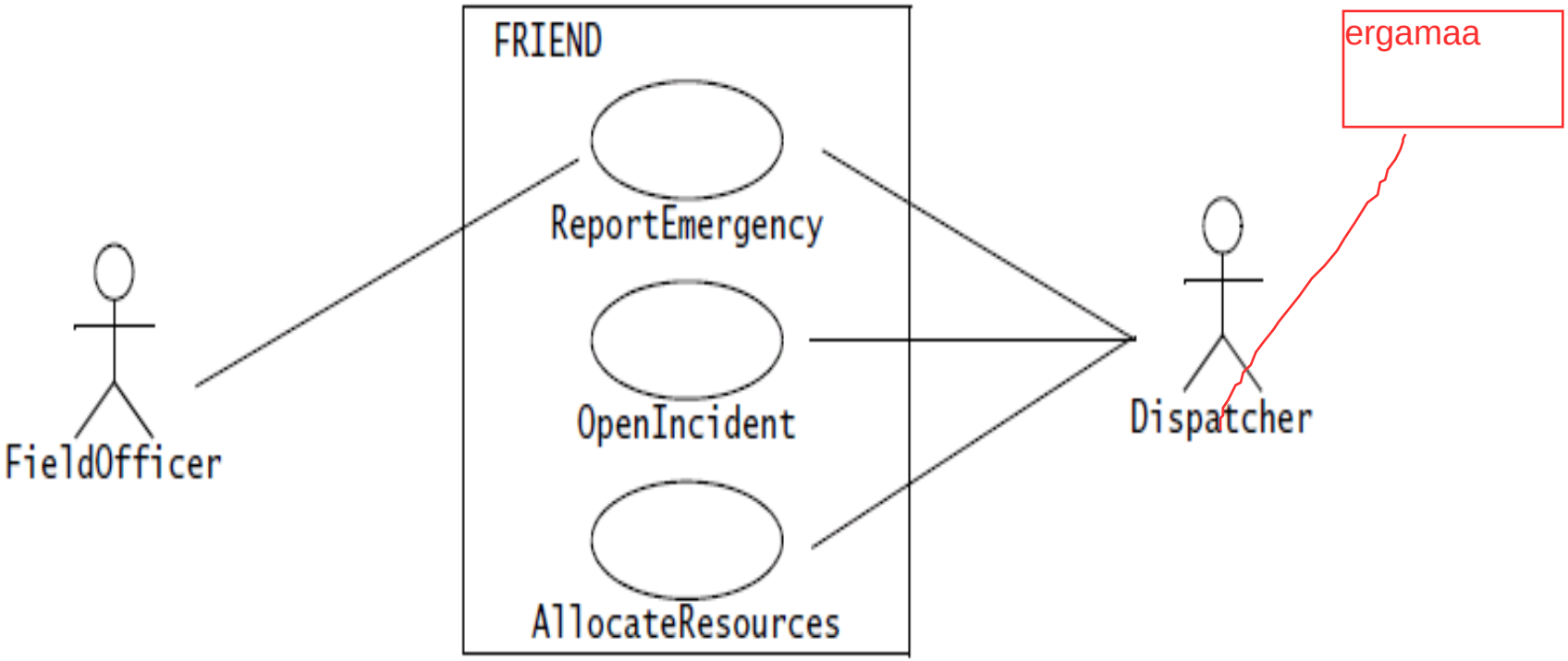
UML notations

- **Use Case Diagrams**
  - Use cases are used during requirements elicitation and analysis to represent the functionality of the system.
  - Use cases focus on the behavior of the system from an external point of view.
  - A usecase describes a function provided by the system that yields a visible result for an actor.
  - An actor describes any entity that interacts with the system

A UML use case diagram describing the functionality of a simple watch



Example…

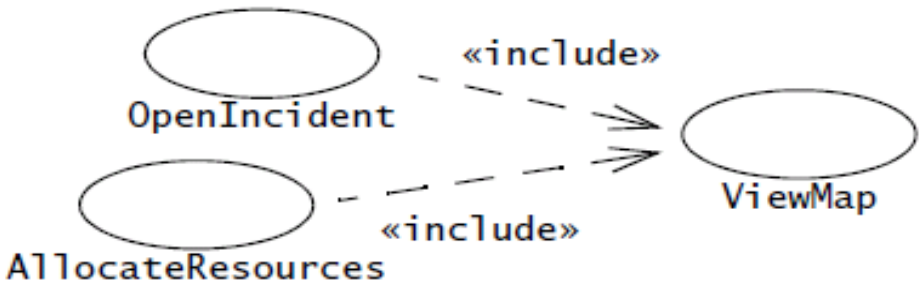


UML use case diagram for First Responder Interactive Emergency Navigational Database (FRIEND), an accident management system

An example of a use case, ReportEmergency

Use case name	`ReportEmergency`
Participating actors	Initiated by `FieldOfficer` Communicates with `Dispatcher`
Flow of events	1. The `FieldOfficer` activates the "Report Emergency" function of her terminal. 2. `FRIEND` responds by presenting a form to the `FieldOfficer`. 3. The `FieldOfficer` fills out the form by selecting the emergency level, type, location, and brief description of the situation. The `FieldOfficer` also describes possible responses to the emergency situation. Once the form is completed, the `FieldOfficer` submits the form. 4. `FRIEND` receives the form and notifies the `Dispatcher`. 5. The `Dispatcher` reviews the submitted information and creates an `Incident` in the database by invoking the `OpenIncident` use case. The `Dispatcher` selects a response and acknowledges the report. 6. `FRIEND` displays the acknowledgment and the selected response to the `FieldOfficer`.
Entry condition	• The `FieldOfficer` is logged into `FRIEND`.
Exit condition	• The `FieldOfficer` has received an acknowledgment and the selected response from the `Dispatcher`, OR • The `FieldOfficer` has received an explanation indicating why the transaction could not be processed.
Quality requirements	• The `FieldOfficer`'s report is acknowledged within 30 seconds. • The selected response arrives no later than 30 seconds after it is sent by the `Dispatcher`.

Include relationships



igure 2-15 An example of an «include» relationship (UML use case diagram).

Use case name	AllocateResources
Participating actor	Initiated by Dispatcher
Flow of events	. . .
Entry condition	• The Dispatcher opens an Incident.
Exit condition	• Additional Resources are assigned to the Incident. • Resources receives notice about their new assignment. • FieldOfficer in charge of the Incident receives notice about the new Resources.
Quality requirements	At any point during the flow of events, this use case can **include** the ViewMap use case. The ViewMap use case is initiated when the Dispatcher invokes the map function. When invoked within this use case, the system scrolls the map so that location of the current Incident is visible to the Dispatcher.

Extend relationships

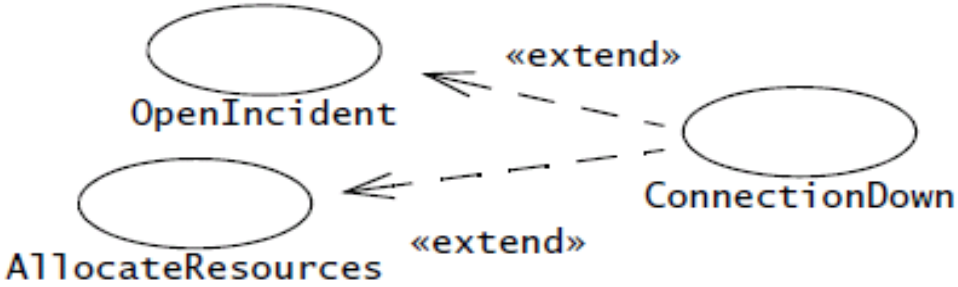
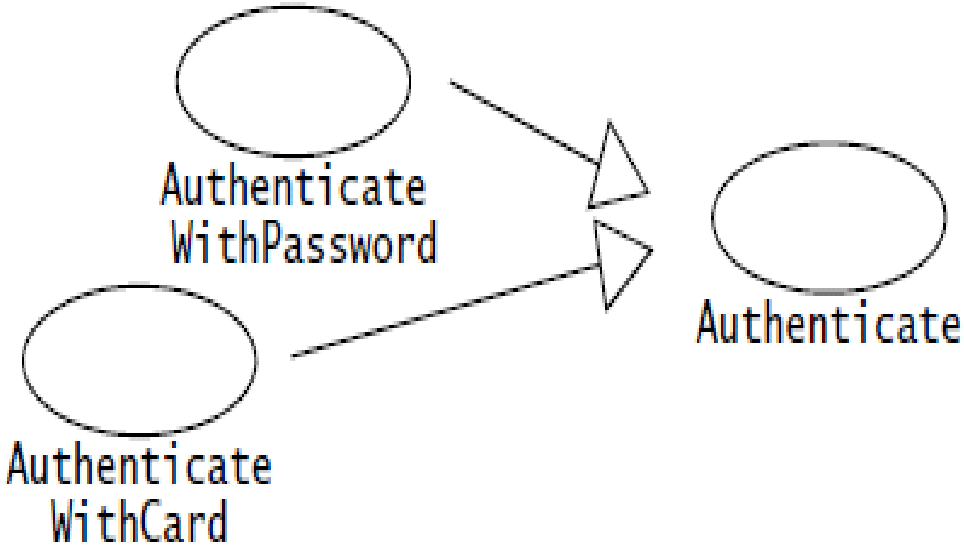


Figure 2-17 An example of an «extend» relationship (UML use case diagram).

Use case name	ConnectionDown
Participating actor	Communicates with FieldOfficer and Dispatcher.
Flow of events	. . .
Entry condition	This use case **extends** the OpenIncident and the AllocateResources use cases. It is initiated by the system whenever the network connection between the FieldOfficer and Dispatcher is lost.
Exit condition	. . .

Inheritance relationships

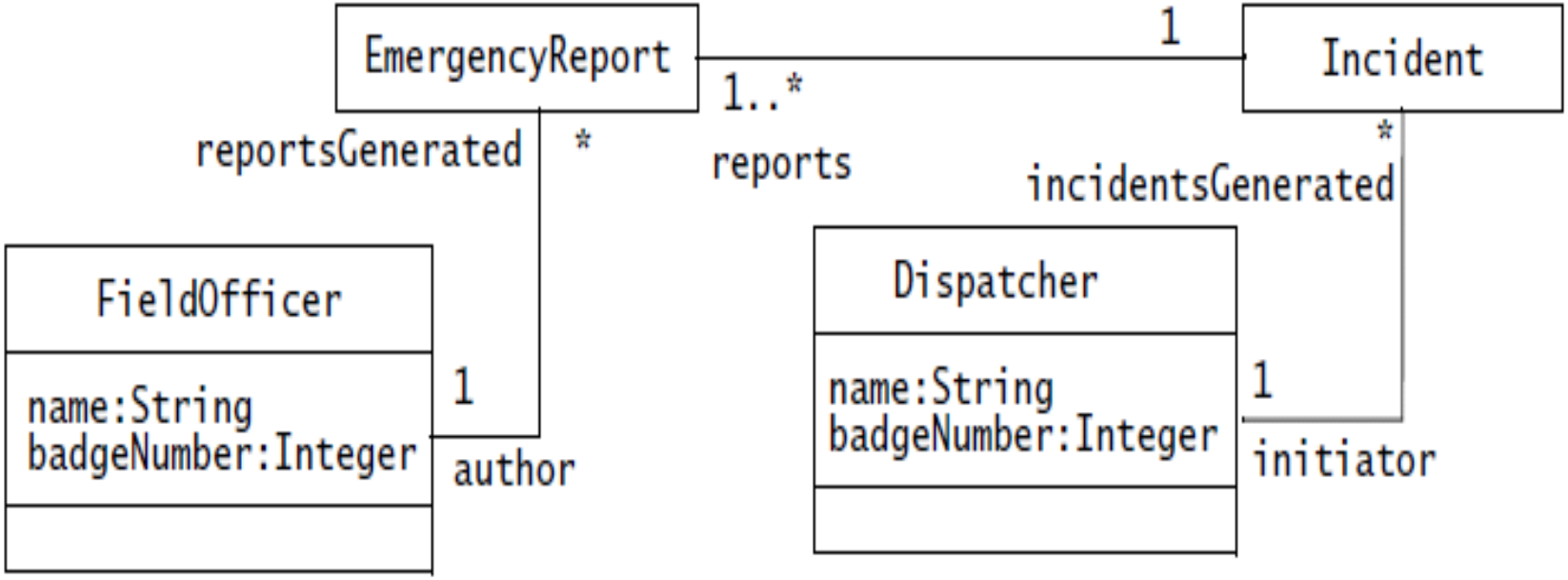


An example of an inheritance relationship

Use case name	`AuthenticateWithCard`
Participating actor	**Inherited** from Authenticate use case.
Flow of events	1. The `FieldOfficer` inserts her card into the field terminal. 2. The field terminal acknowledges the card and prompts the actor for her personal identification number (PIN). 3. The `FieldOfficer` enters her PIN with the numeric keypad. 4. The field terminal checks the entered PIN against the PIN stored on the card. If the PINs match, the `FieldOfficer` is authenticated. Otherwise, the field terminal rejects the authentication attempt.
Entry condition	**Inherited** from `Authenticate` use case.
Exit condition	**Inherited** from `Authenticate` use case.

Classes and objects

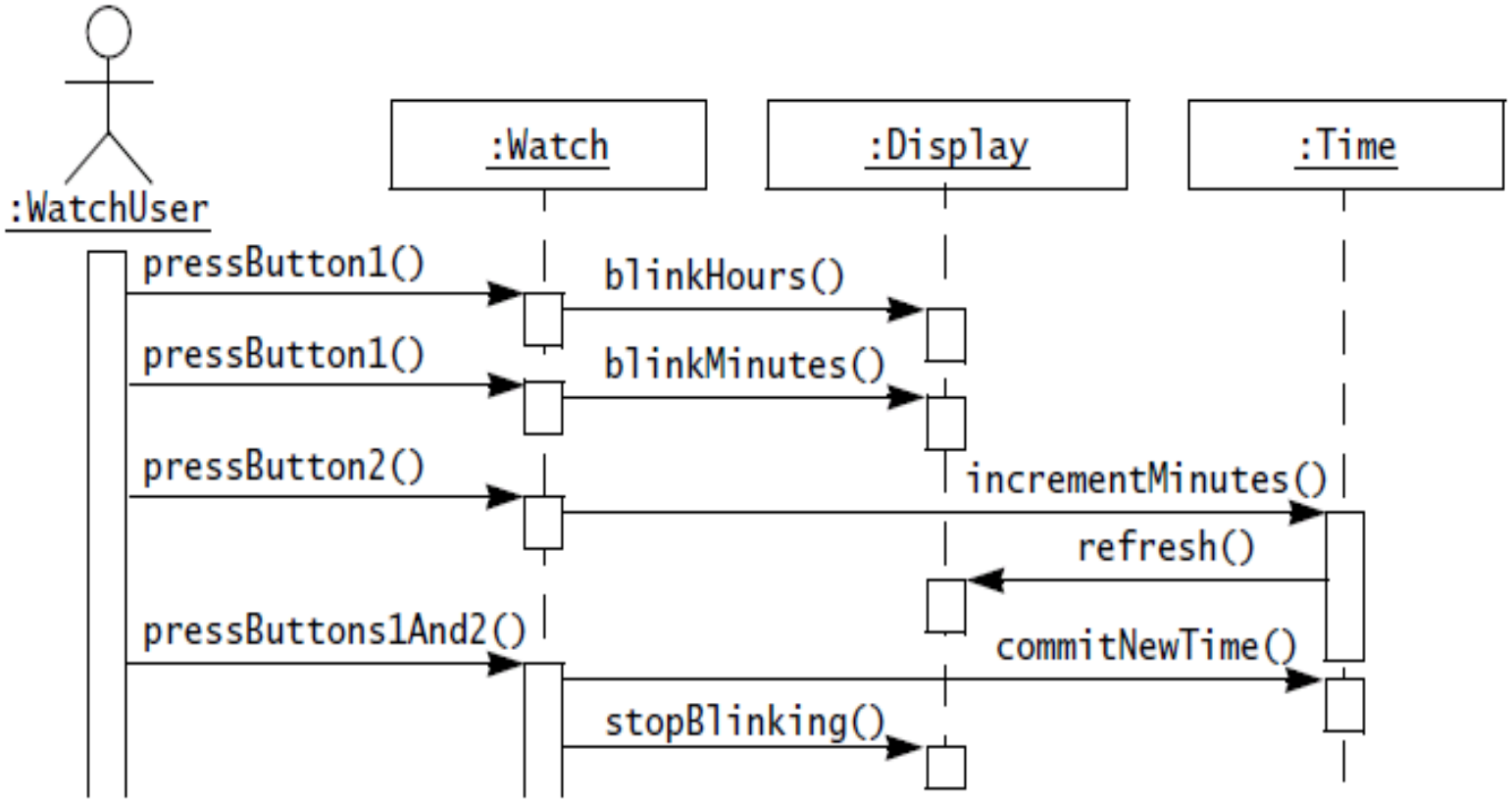
The warehouseOnFire scenario for the ReportEmergency use case.



Interaction Diagrams

- Interaction diagrams are used to formalize the dynamic behavior of the system and to visualizethe communication among objects.
- They are useful for identifying additional objects that participate in the use cases.

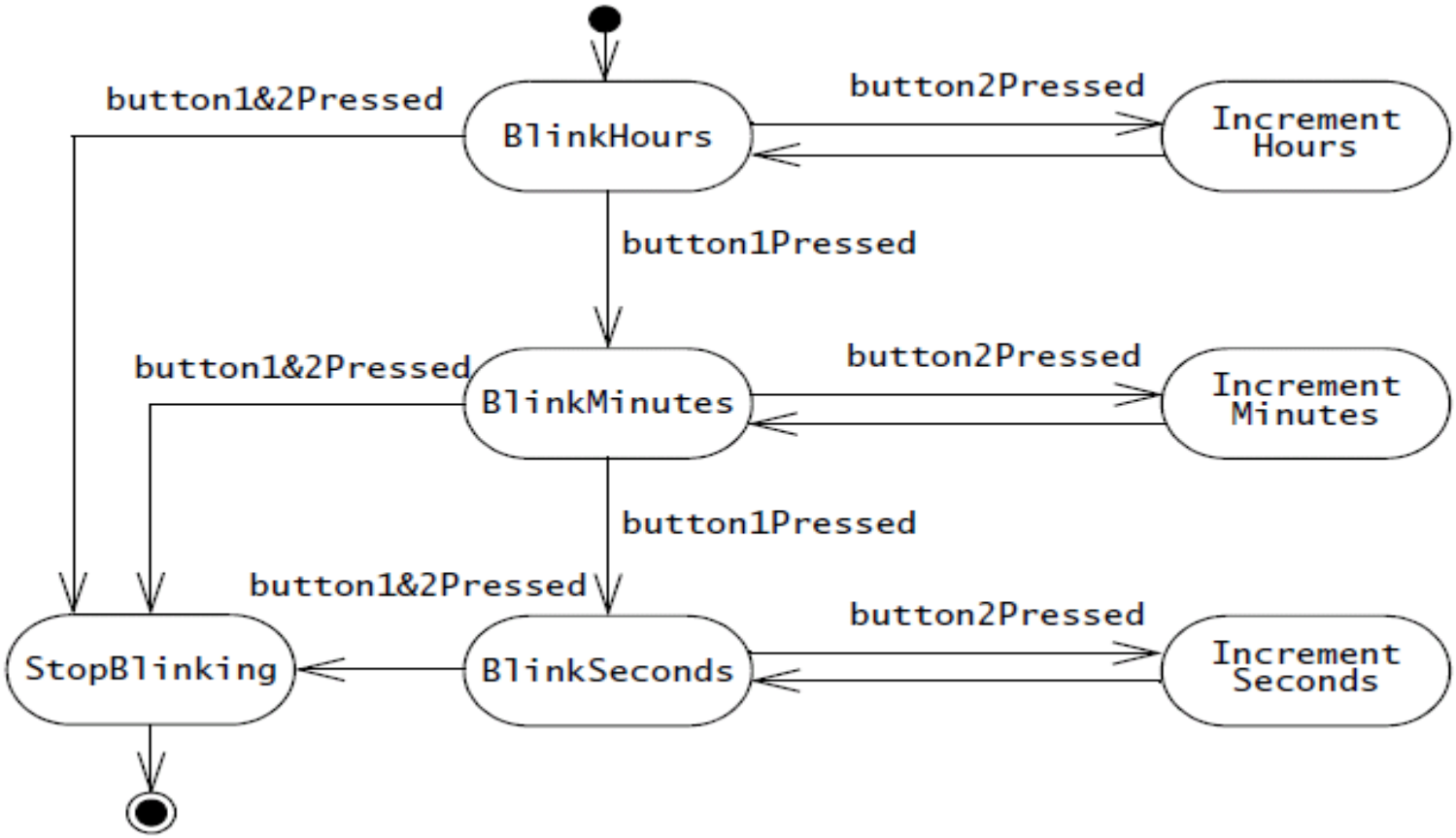
A UML sequence diagram for the Watch



State Machine Diagrams

- State machine diagrams describe the dynamic behavior of an individual object as a number of states and transitions between these states.
- A state represents a particular set of values for an object.
- Given a state, a transition represents a future state the object can move to and the conditions associated with the change of state.

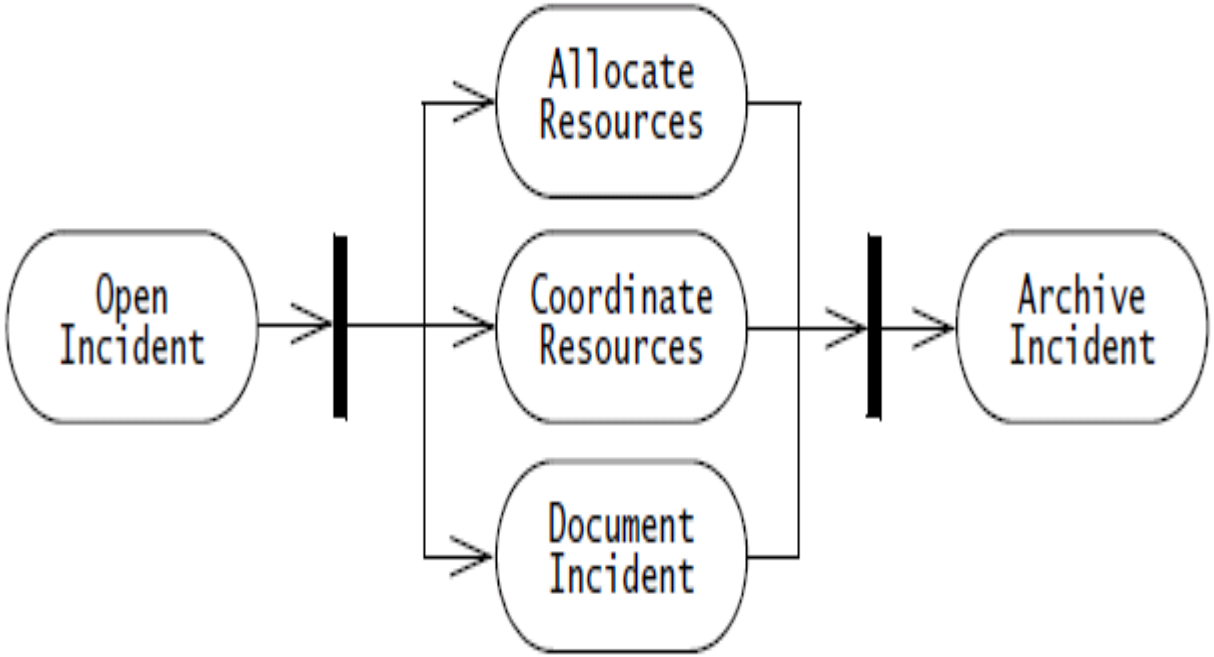
A UML state machine diagram for SetTime use case of the Watch



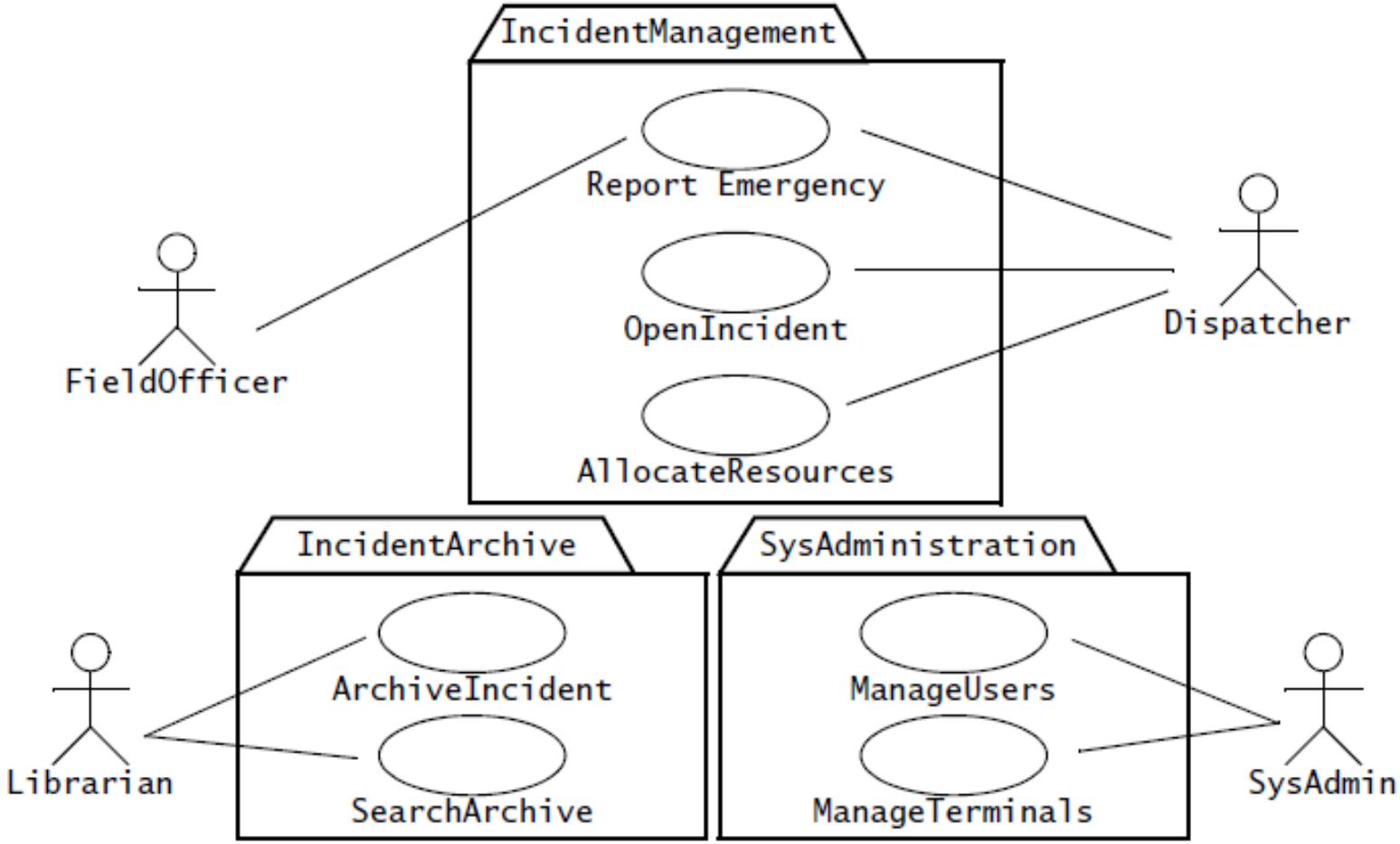
Activity Diagrams

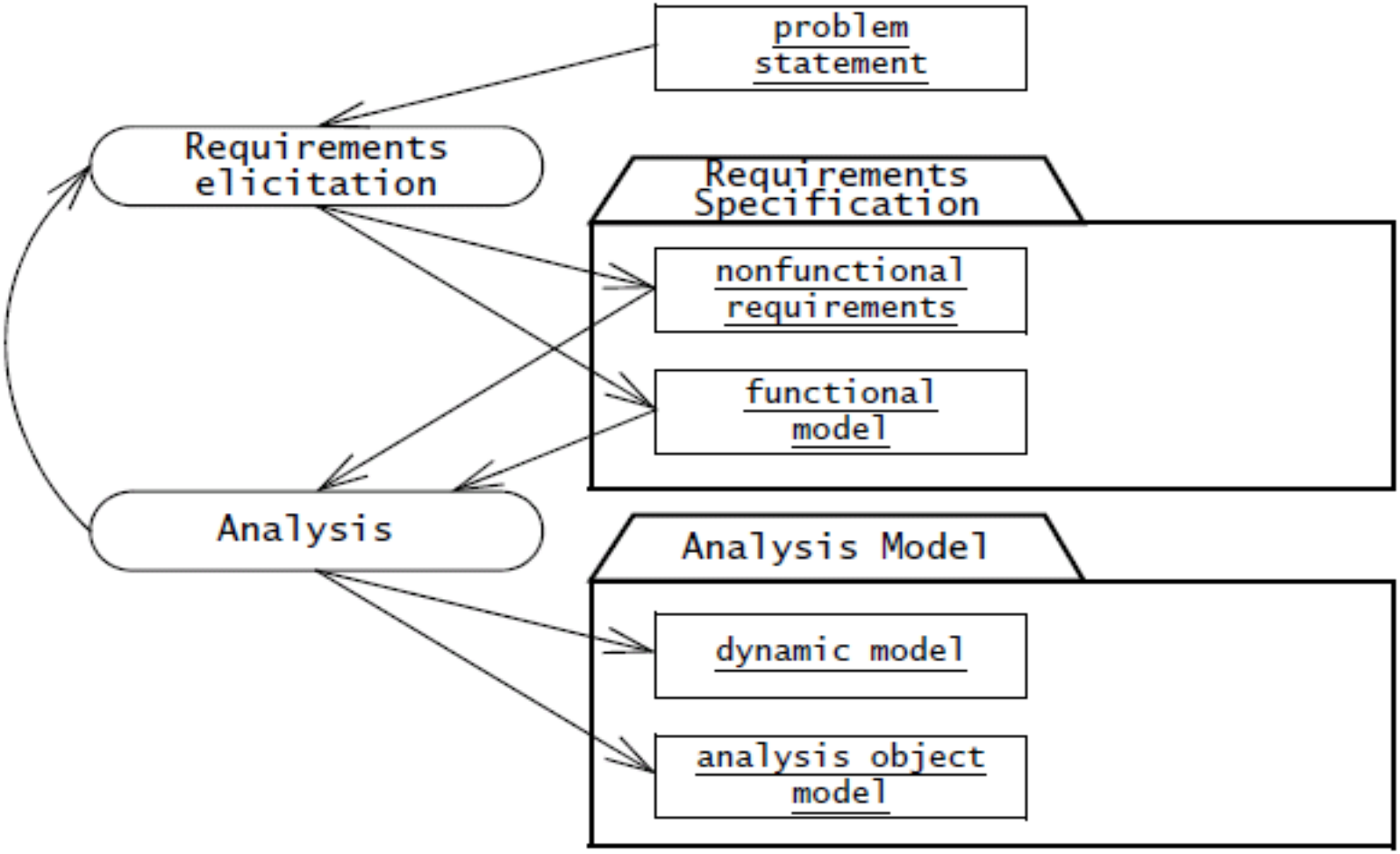
- An activity diagram describes the behavior of a system in terms of activities.
- Activities are modeling elements that represent the execution of a set of operations.
- The execution of an activity can be triggered by the completion of other activities, by the availability of objects, or by external events.

UML activity diagram



Example of packages: use cases of FRIEND organized by actors




problem statement
Requirements elicitation
Requirements Specification
nonfunctional requirements
functional model
Analysis
Analysis Model
dynamic model
analysis object model

Requirements elicitation includes the following activities:

- Identifying Actors
- *Identifying scenarios*
- *Identifying use cases*
- *Refining use cases*
- *Identifying relationships among use cases, actor and use cases*
- *Identifying nonfunctional requirements*
  - **Joint Application Design (JAD)**

Functional Requirements

- describe the interactions between the system and its environment independent of its implementation.
- The environment includes the user and any other external system with which the system interacts

Nonfunctional Requirements

- **Usability**
- **Reliability**
- **Performance**
- **Supportability**
- **Constraints or pseudo requirements**
  - **Implementation requirements**
  - **Interface requirements**
  - **Operations requirements**
  - **Packaging requirements**
  - **Legal requirements**

Completeness

- all possible scenarios through
- the system are described, including exceptional behavior

Consistency

- **it does** not contradict itself

Unambiguous

Correctness

it represents accurately the system that the client needs and that the developers intend to build

Realism

- the system can be implemented within constraints

Verifiability

- Repeatable tests can be designed to demonstrate that the system fulfills the requirements specification

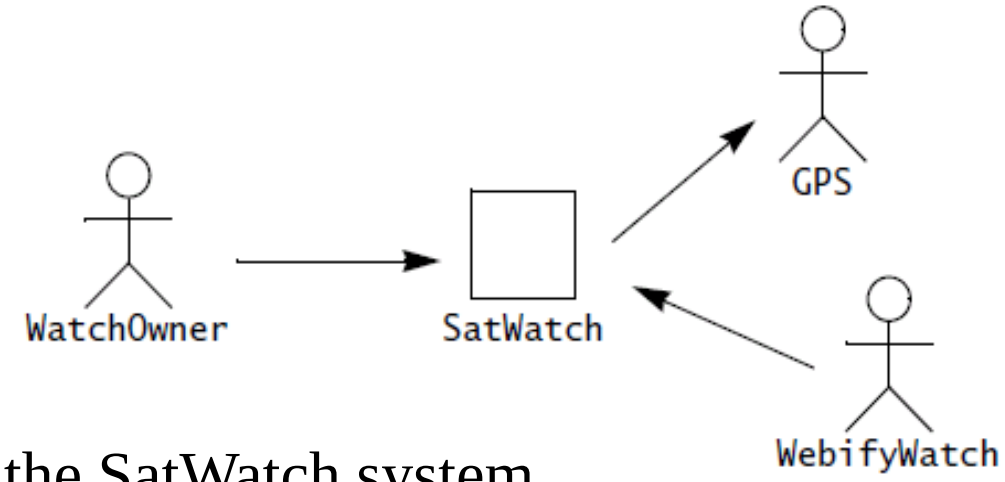
Traceability

- Each requirement can be traced throughout the software development to its corresponding system functions
- **Greenfield Engineering**
- **Reengineering**
- **Interface Engineering**

Requirements Elicitation Activities

Identifying Actors

- Actors represent external entities that interact with the system.
- An actor can be human or an external system



Actors for the SatWatch system

Requirements Elicitation Activities…

Questions for identifying actors

- Which user groups are supported by the system to perform their work?
- Which user groups execute the system's main functions?
- Which user groups perform secondary functions, such as maintenance and administration?
- With what external hardware or software system will the system interact?

Requirements Elicitation Activities…

Identifying Scenarios

- A scenario is “a narrative description of what people do and experience as they try to make use of computer systems and applications” [Carroll, 1995
- A scenario is a concrete, focused, informal description of a single feature of the system from the viewpoint of a single actor.

Scenario name	`warehouseOnFire`
Participating actor instances	`bob, alice:FieldOfficer` `john:Dispatcher`
Flow of events	1. Bob, driving down main street in his patrol car, notices smoke coming out of a warehouse. His partner, Alice, activates the "Report Emergency" function from her `FRIEND` laptop. 2. Alice enters the address of the building, a brief description of its location (i.e., northwest corner), and an emergency level. In addition to a fire unit, she requests several paramedic units on the scene, given that the area appears to be relatively busy. She confirms her input and waits for an acknowledgment. 3. John, the `Dispatcher`, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the `Incident` site and sends their estimated arrival time (ETA) to Alice. 4. Alice receives the acknowledgment and the ETA.

`warehouseOnFire` scenario for the `ReportEmergency` use case.

Requirements Elicitation Activities…

Scenarios can have many different uses during requirements elicitation and during other activities of the life cycle

- **As-is scenarios** describe a current situation
- **Visionary scenarios** describe a future system
- **Evaluation scenarios** describe user tasks against which the system is to be evaluated
- **Training scenarios** are tutorials used for introducing new users to the system

Requirements Elicitation Activities…

Questions for identifying scenarios

- What are the tasks that the actor wants the system to perform?
- What information does the actor access? Who creates that data? Can it be modified or removed? By whom?
- Which external changes does the actor need to inform the system about? How often? When?
- Which events does the system need to inform the actor about? With what latency?.

Requirements Elicitation Activities…

Identifying Use Cases

- A scenario is an instance of a use case; that is, a use case specifies all possible scenarios for a given piece of functionality.
- A use case is initiated by an actor. After its initiation, a use case may interact with other actors, as well.
- A use case represents a complete flow of events through the system in the sense that it describes a series of related interactions that result from its initiation.

The use case ReportEmergency of which the scenario warehouseOnFire is an instance

Use case name	`ReportEmergency`
Participating actors	Initiated by `FieldOfficer` Communicates with `Dispatcher`
Flow of events	1. The `FieldOfficer` activates the "Report Emergency" function of her terminal. 2. `FRIEND` responds by presenting a form to the `FieldOfficer`. 3. The `FieldOfficer` completes the form by selecting the emergency level, type, location, and brief description of the situation. The `FieldOfficer` also describes possible responses to the emergency situation. Once the form is completed, the `FieldOfficer` submits the form. 4. `FRIEND` receives the form and notifies the `Dispatcher`. 5. The `Dispatcher` reviews the submitted information and creates an `Incident` in the database by invoking the `OpenIncident` use case. The `Dispatcher` selects a response and acknowledges the report. 6. `FRIEND` displays the acknowledgment and the selected response to the `FieldOfficer`.
Entry condition	• The `FieldOfficer` is logged into `FRIEND`.
Exit conditions	• The `FieldOfficer` has received an acknowledgment and the selected response from the `Dispatcher`, OR • The `FieldOfficer` has received an explanation indicating why the transaction could not be processed.
Quality requirements	• The `FieldOfficer`'s report is acknowledged within 30 seconds. • The selected response arrives no later than 30 seconds after it is sent by the `Dispatcher`.

Simple Use Case Writing Guide

- Use cases should be named with verb phrases. The name of the use case should indicate what the user is trying to accomplish (e.g., ReportEmergency, OpenIncident).
- Actors should be named with noun phrases (e.g., FieldOfficer, Dispatcher, Victim).
- The boundary of the system should be clear. Steps accomplished by the actor and steps accomplished by the system should be distinguished
- Use case steps in the flow of events should be phrased in the active voice. This makes it explicit who accomplished the step.
- The causal relationship between successive steps should be clear.
- A use case should describe a complete user transaction (e.g., the ReportEmergency use case describes all the steps between initiating the emergency reporting and receiving an acknowledgment). Exceptions should be described separately.
- A use case should not describe the user interface of the system. This takes away the focus from the actual steps accomplished by the user and is better addressed with visual mock-ups (e.g., the ReportEmergency only refers to the “Report Emergency” function, not the menu, the button, nor the actual command that corresponds to this function).
- A use case should not exceed two or three pages in length. Otherwise, use include and extend relationships to decompose it in smaller use cases,

Requirements Elicitation Activities…

Refining Use Cases

- The use of scenarios and use cases to define the functionality of the system aims at creating requirements that are validated by the user early in the development.
- As the design and implementation of the system starts, the cost of changing the requirements specification and adding new unforeseen functionality increases

Requirements Elicitation Activities…

Identifying Relationships among Actors and Use Cases

- Relationships among actors and use cases enable the developers and users to reduce the complexity of the model and increase its understandability.
- We use communication relationships between actors and use cases to describe the system in layers of unctionality.
- We use extend relationships to separate exceptional and common flows of events.
- We use include relationships to reduce redundancy among use cases.

Extend versus include relationships

- The main distinction between these constructs is the direction of the relationship.
- For include relationships, the event triggering the target (i.e.,included) use case is described in the flow of event of the source use case.
- For extend relationships, the event triggering the source (i.e., extending) use case is described in the source use case as a precondition.
- In other words, for include relationships, every including use case must specify where the included use case should be invoked.

Requirements Elicitation Activities…

Identifying Initial Analysis Objects
Identifying Nonfunctional Requirements

Category	Example questions
Usability	• What is the level of expertise of the user? • What user interface standards are familiar to the user? • What documentation should be provided to the user?
Reliability *(including robustness, safety, and security)*	• How reliable, available, and robust should the system be? • Is restarting the system acceptable in the event of a failure? • How much data can the system loose? • How should the system handle exceptions? • Are there safety requirements of the system? • Are there security requirements of the system?
Performance	• How responsive should the system be? • Are any user tasks time critical? • How many concurrent users should it support? • How large is a typical data store for comparable systems? • What is the worse latency that is acceptable to users?
Supportability *(including maintainability and portability)*	• What are the foreseen extensions to the system? • Who maintains the system? • Are there plans to port the system to different software or hardware environments?
Implementation	• Are there constraints on the hardware platform? • Are constraints imposed by the maintenance team? • Are constraints imposed by the testing team?
Interface	• Should the system interact with any existing systems? • How are data exported/imported into the system? • What standards in use by the client should be supported by the system?
Operation	• Who manages the running system?
Packaging	• Who installs the system? • How many installations are foreseen? • Are there time constraints on the installation?
Legal	• How should the system be licensed? • Are any liability issues associated with system failures? • Are any royalties or licensing fees incurred by using specific algorithms or components?

Documenting Requirements Elicitation

- The results of the requirements elicitation and the analysis activities are documented in the **Requirements Analysis Document (RAD).**
- **This document completely describes the system in** terms of functional and nonfunctional requirements.
- The audience for the RAD includes the client, the users, the project management, the system analysts (i.e., the developers who participate in the requirements), and the system designers (i.e., the developers who participate in the system design).
- The first part of the document, including use cases and nonfunctional requirements, is written during requirements elicitation. The formalization of the specification in terms of object models is written during analysis

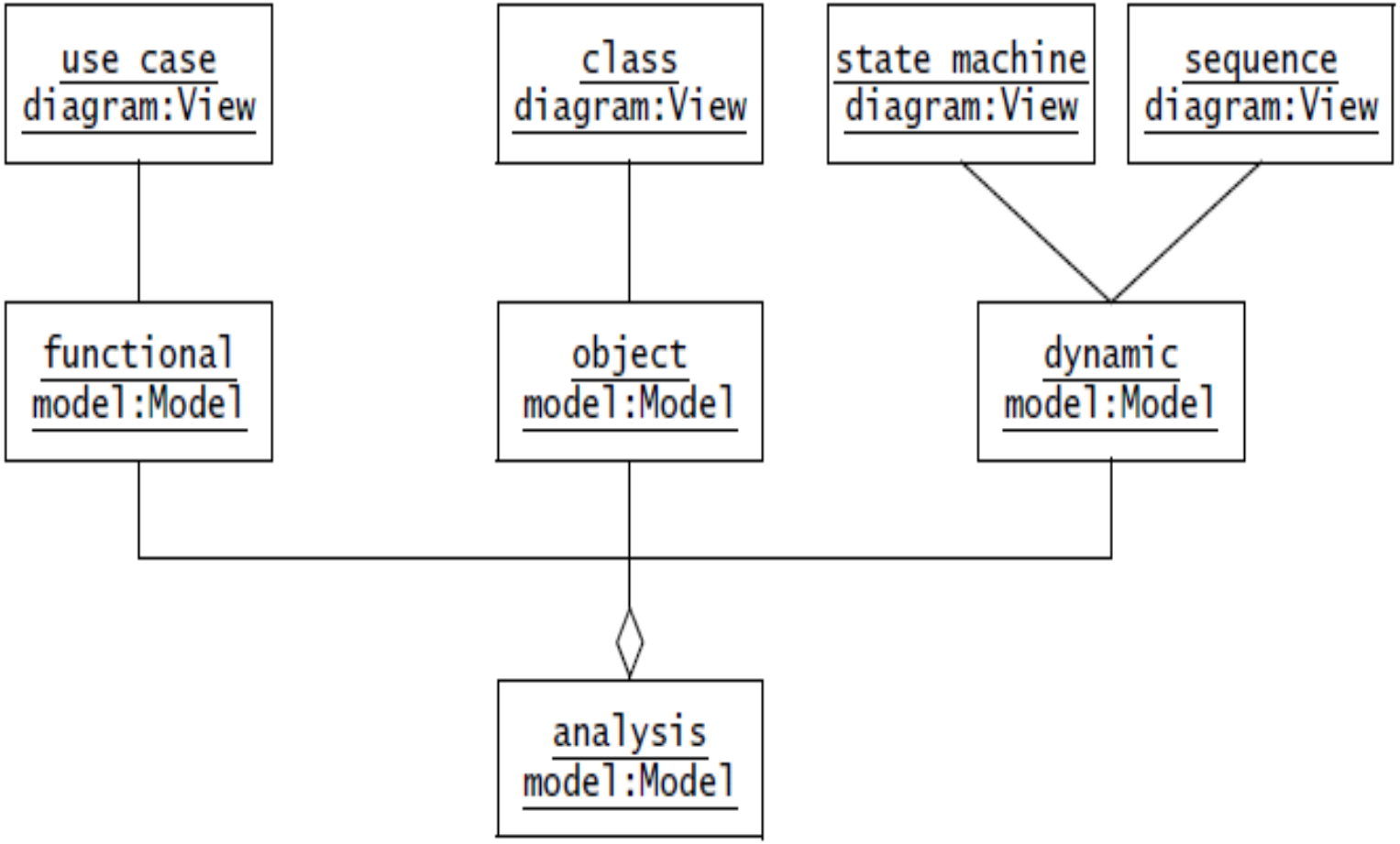
Requirement Analysis using UML

An Overview of Analysis

- Analysis focuses on producing a model of the system, called the analysis model, which is correct, complete, consistent, and verifiable.
- Analysis is different from requirements elicitation in that developers focus on structuring and formalizing the requirements elicited from users

An Overview of Analysis…

- The analysis model is composed of three individual models**: the functional model,** represented by use cases and scenarios, the **analysis object model,** represented by class and object diagrams, and the **dynamic model,** represented by state machine and sequence diagrams



The analysis model is composed of the functional model, the object model, and the dynamic model. In UML, the functional model is represented with use case diagrams, the object model with class diagrams, and the dynamic model with state machine and sequence diagrams.

Analysis Concepts

Analysis Object Models and Dynamic Models

- The analysis model represents the system under development from the user's point of view

- The analysis object model is a part of the analysis model and focuses on the individual concepts that are manipulated by the system, their properties and their relationships.

- The analysis object model, depicted with UML class diagrams, includes classes, attributes, and operations

Analysis Concepts …

- The **dynamic model focuses on the behavior of the system**
- The dynamic model is depicted with sequence diagrams and with state machines
- State machines represent the behavior of a single object (or a group of very tightly coupled objects).
-

64

Analysis Concepts …

- The dynamic model serves to assign responsibilities to individual classes and, in the process, to identify new classes, associations, and attributes to be added to the analysis object model.
- classes such as Database, Subsystem, SessionManager, Network, should not appear in the analysis model

 65

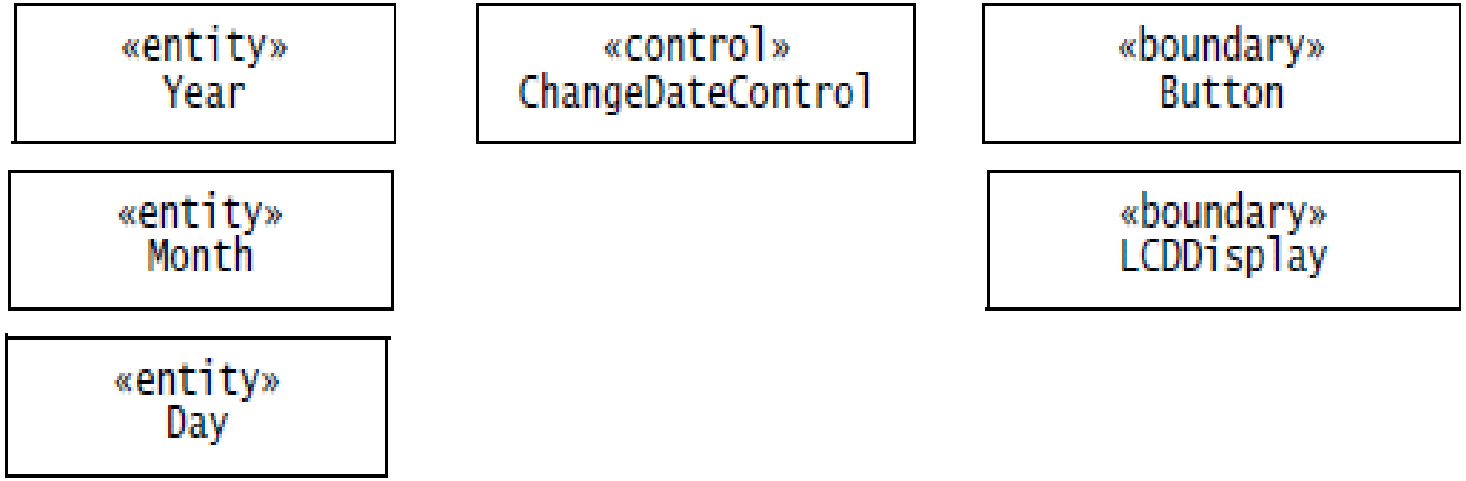
Analysis Concepts …

Entity, Boundary, and Control Objects

- **Entity objects** represent the persistent information tracked by the system.
- **Boundary objects** represent the interactions between the actors and the system.
- **Control objects** are in charge of realizing use cases.

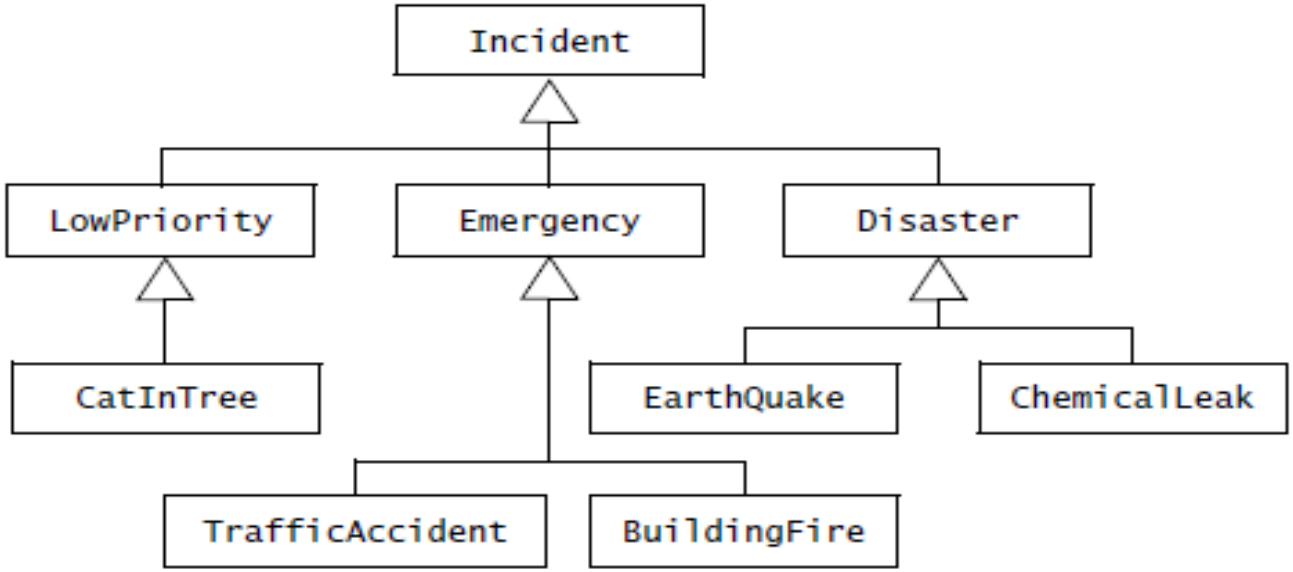
 66

Analysis Concepts …



Analysis Concepts …

Generalization and Specialization



An example of a generalization hierarchy (UML class diagram). The top of the hierarchy represents the most general concept, whereas the bottom nodes represent the most specialized concepts.

Analysis Activities: From Use Cases to Objects

- Analysis activities that transform the use cases and scenarios produced during requirements elicitation into an analysis model include:

1. Identifying Entity Objects

Heuristics for identifying entity objects

- Terms that developers or users need to clarify in order to understand the use case
- Recurring nouns in the use cases (e.g., `Incident`)
- Real-world entities that the system needs to track (e.g., `FieldOfficer`, `Dispatcher`, `Resource`)
- Real-world activities that the system needs to track (e.g., `EmergencyOperationsPlan`)
- Data sources or sinks (e.g., `Printer`).

Entity objects for the `ReportEmergency` use case.

`Dispatcher`	Police officer who manages `Incidents`. A `Dispatcher` opens, documents, and closes `Incidents` in response to `Emergency Reports` and other communication with `FieldOfficers`. `Dispatchers` are identified by badge numbers.
`EmergencyReport`	Initial report about an `Incident` from a `FieldOfficer` to a `Dispatcher`. An `EmergencyReport` usually triggers the creation of an `Incident` by the `Dispatcher`. An `EmergencyReport` is composed of an emergency level, a type (fire, road accident, other), a location, and a description.
`FieldOfficer`	Police or fire officer on duty. A `FieldOfficer` can be allocated to, at most, one `Incident` at a time. `FieldOfficers` are identified by badge numbers.
`Incident`	Situation requiring attention from a `FieldOfficer`. An `Incident` may be reported in the system by a `FieldOfficer` or anybody else external to the system. An `Incident` is composed of a description, a response, a status (open, closed, documented), a location, and a number of `FieldOfficers`.

Identifying Boundary Objects

- Boundary objects represent the system interface with the actors. In each use case, each actor interacts with at least one boundary object.
- The boundary object collects the information from the actor and translates it into a form that can be used by both entity and control objects

Heuristics for identifying boundary objects

- Identify user interface controls that the user needs to initiate the use case (e.g., `ReportEmergencyButton`).
- Identify forms the users needs to enter data into the system (e.g., `EmergencyReportForm`).
- Identify notices and messages the system uses to respond to the user (e.g., `AcknowledgmentNotice`).
- When multiple actors are involved in a use case, identify actor terminals (e.g., `DispatcherStation`) to refer to the user interface under consideration.
- Do not model the visual aspects of the interface with boundary objects (user mock-ups are better suited for that).
- *Always* use the end user's terms for describing interfaces; do not use terms from the solution or implementation domains.

Boundary objects for the ReportEmergency use case.

AcknowledgmentNotice	Notice used for displaying the Dispatcher's acknowledgment to the FieldOfficer.
DispatcherStation	Computer used by the Dispatcher.
ReportEmergencyButton	Button used by a FieldOfficer to initiate the ReportEmergency use case.
EmergencyReportForm	Form used for the input of the ReportEmergency. This form is presented to the FieldOfficer on the FieldOfficerStation when the "Report Emergency" function is selected. The EmergencyReportForm contains fields for specifying all attributes of an emergency report and a button (or other control) for submitting the completed form.
FieldOfficerStation	Mobile computer used by the FieldOfficer.
IncidentForm	Form used for the creation of Incidents. This form is presented to the Dispatcher on the DispatcherStation when the EmergencyReport is received. The Dispatcher also uses this form to allocate resources and to acknowledge the FieldOfficer's report.

Identifying Control Objects

- Control objects are responsible for coordinating boundary and entity objects.
- Control objects usually do not have a concrete counterpart in the real world.
- Often a close relationship exists between a use case and a control object; a control object is usually created at the beginning of a use case and ceases to exist at its end.
- It is responsible for collecting information from the boundary objects and dispatching it to entity objects.

Heuristics for identifying control objects

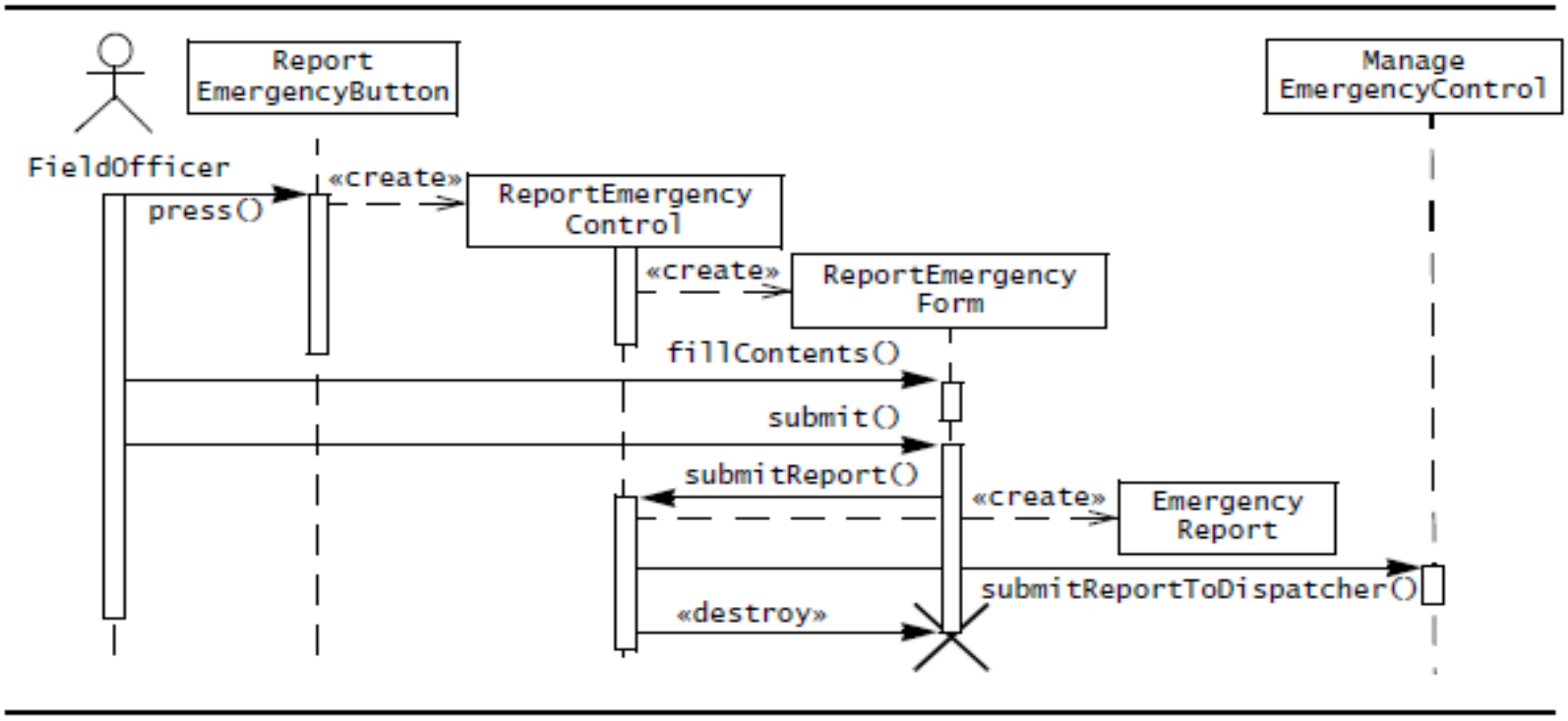
- Identify one control object per use case.
- Identify one control object per actor in the use case.
- The life span of a control object should cover the extent of the use case or the extent of a user session. If it is difficult to identify the beginning and the end of a control object activation, the corresponding use case probably does not have well-defined entry and exit conditions.

Control objects for the ReportEmergency use case.

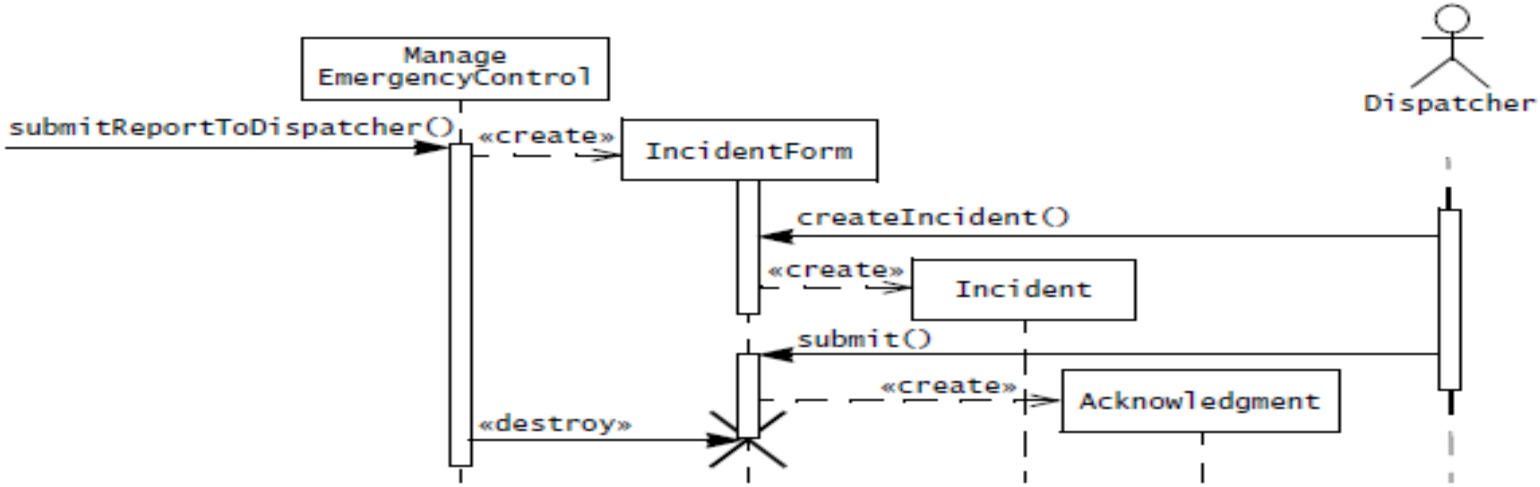
ReportEmergencyControl	Manages the ReportEmergency reporting function on the FieldOfficerStation. This object is created when the FieldOfficer selects the "Report Emergency" button. It then creates an EmergencyReportForm and presents it to the FieldOfficer. After submitting the form, this object then collects the information from the form, creates an EmergencyReport, and forwards it to the Dispatcher. The control object then waits for an acknowledgment to come back from the DispatcherStation. When the acknowledgment is received, the ReportEmergencyControl object creates an AcknowledgmentNotice and displays it to the FieldOfficer.
ManageEmergencyControl	Manages the ReportEmergency reporting function on the DispatcherStation. This object is created when an EmergencyReport is received. It then creates an IncidentForm and displays it to the Dispatcher. Once the Dispatcher has created an Incident, allocated Resources, and submitted an acknowledgment, ManageEmergencyControl forwards the acknowledgment to the FieldOfficerStation.

Mapping Use Cases to Objects with Sequence Diagrams

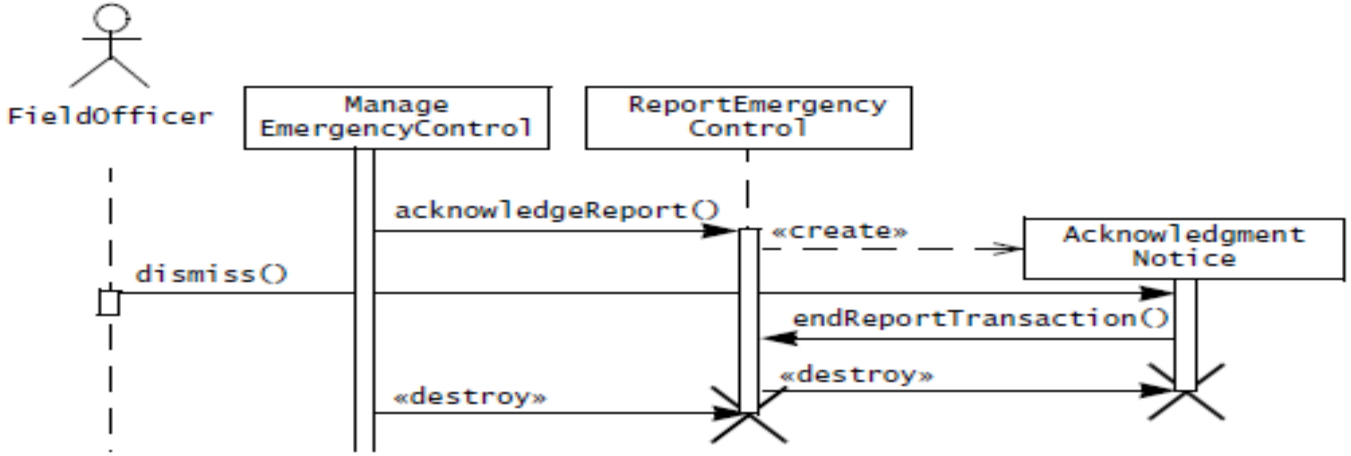
- A sequence diagram ties use cases with objects. It shows how the behavior of a use case (or scenario) is distributed among its participating objects



Sequence diagram for the ReportEmergency use case.



Sequence diagram for the ReportEmergency use case (continued from Figure 5-8).



Heuristics for drawing sequence diagrams

- The first column should correspond to the actor who initiated the use case.
- The second column should be a boundary object (that the actor used to initiate the use case).
- The third column should be the control object that manages the rest of the use case.
- Control objects are created by boundary objects initiating use cases.
- Boundary objects are created by control objects.
- Entity objects are accessed by control and boundary objects.
- Entity objects *never* access boundary or control objects; this makes it easier to share entity objects across use cases.

Modeling Interactions among Objects with CRC Cards

- An alternative for identifying interactions among objects are **CRC cards (**class, responsibilities, and collaborators)

ReportEmergencyControl

Responsibilities	Collaborators
Collects input from Field-officer Controls sequence of forms during emergency reporting	EmergencyReportForm EmergencyReport AcknowledgementNotic

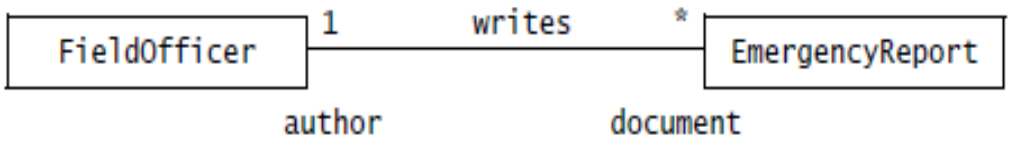
Incident

Responsibilities	Collaborators
Track all information related to a single inci-dent.	Resource

Examples of CRC cards for the `ReportEmergencyControl` and the `Incident` classes.

Identifying Associations

- An association shows a relationship between two or more classes
- Associations have several properties
  - A **name** to describe the association between the two classes



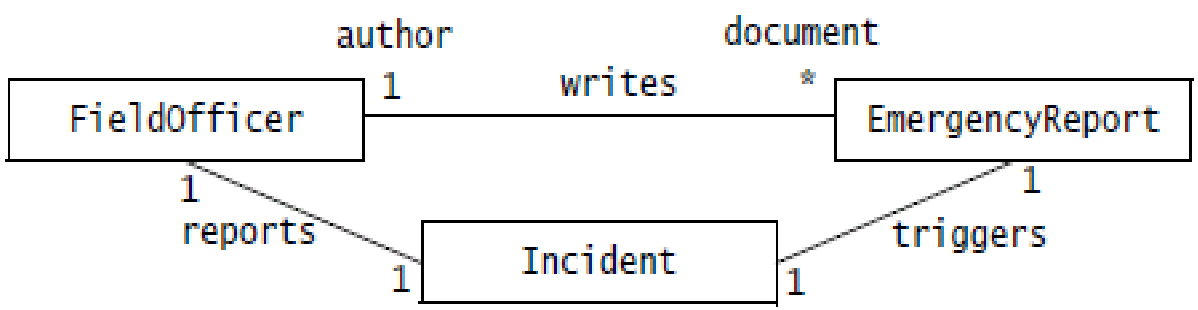
An example of association between the EmergencyReport and the FieldOfficer classes.

Identifying Associations…

- A **role** at each end, identifying the function of each class with respect to the associations
- A **multiplicity** at each end, identifying the possible number of instances (e.g., * indicates a FieldOfficer may write zero or more EmergencyReports, whereas 1 indicates that each EmergencyReport has exactly one FieldOfficer as author).

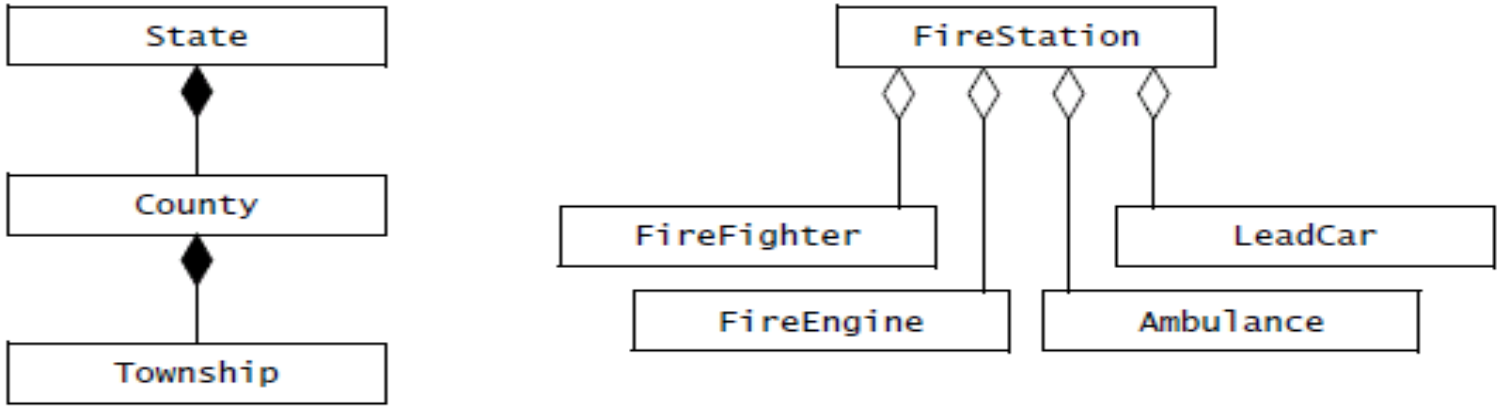
Heuristics for identifying associations

- Examine verb phrases.
- Name associations and roles precisely.
- Use qualifiers as often as possible to identify namespaces and key attributes.
- Eliminate any association that can be derived from other associations.
- Do not worry about multiplicity until the set of associations is stable.
- Too many associations make a model unreadable.



Identifying Aggregates

- **Aggregations** are special types of associations denoting a whole–part relationship**.**

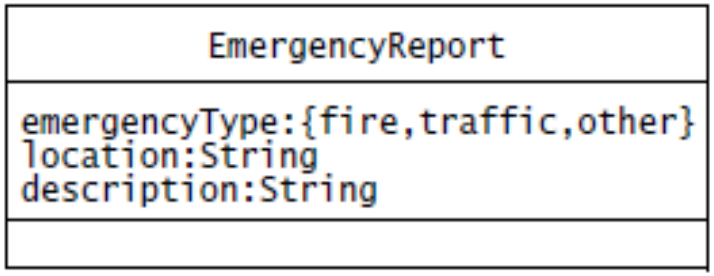


Examples of aggregations and compositions (UML class diagram). A State is composed of many Counties, which in turn is composed of many Townships. A FireStation includes FireFighters, FireEngines, Ambulances, and a LeadCar.

- A hollow diamond denotes a **shared aggregation**

Identifying Attributes

- **Attributes** are properties of individual objects



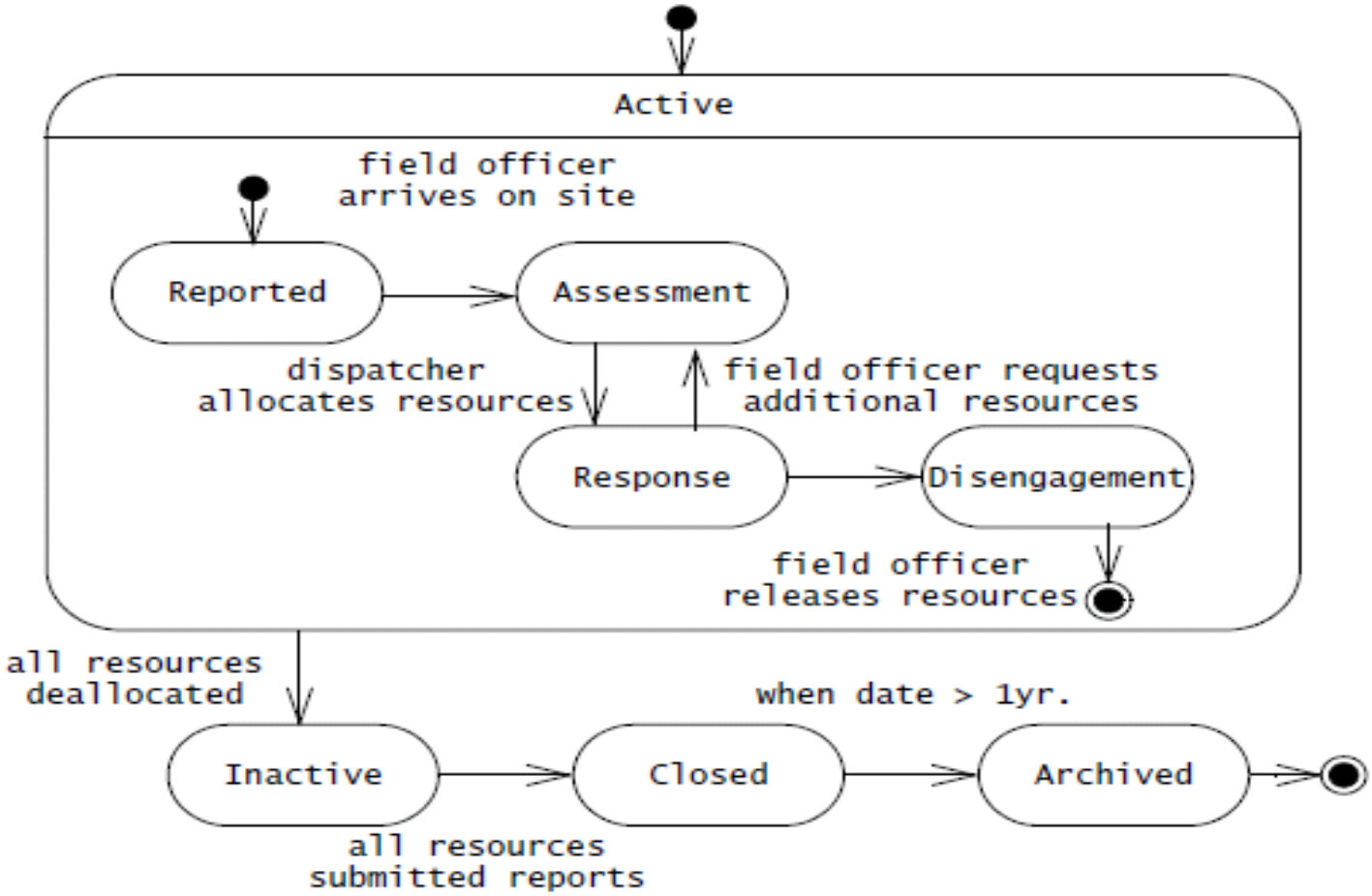
Attributes of the EmergencyReport class.

Heuristics for identifying attributes[a]

- Examine possessive phrases.
- Represent stored state as an attribute of the entity object.
- Describe each attribute.
- Do not represent an attribute as an object; use an association instead
- Do not waste time describing fine details before the object structure is stable.

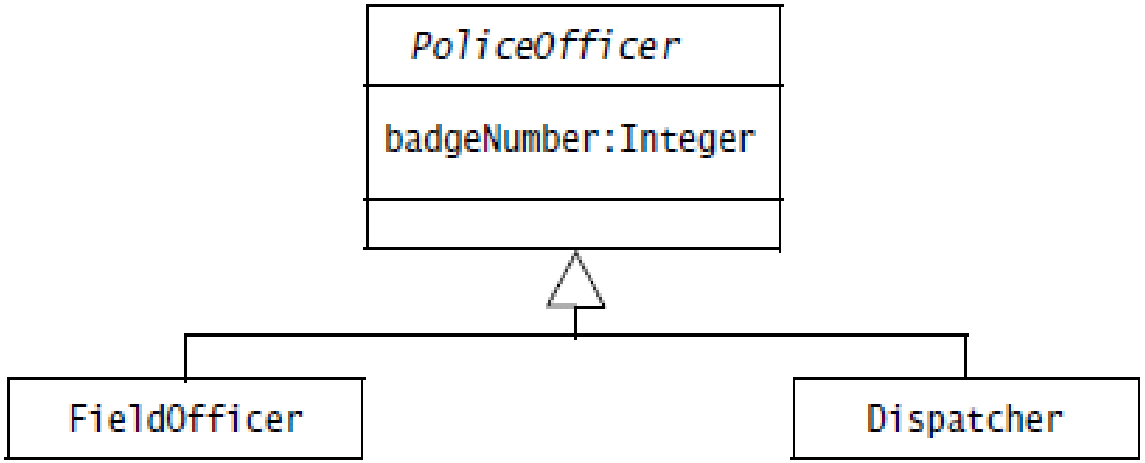
Modeling State-Dependent Behavior of Individual Objects

- Sequence diagrams represent the behavior of the system from the perspective of a single use case.
- State machine diagrams represent behavior from the perspective of a single object
- Viewing behavior from the perspective of each object enables the developer to build a more formal description of the behavior of the object, and consequently, to identify missing use cases



UML state machine for Incident.

Modeling Inheritance Relationships between Objects



An example of inheritance relationship (UML class diagram).

Technique	Good for	Kind of data	Plus	Risks
Questionnaires	Answering specific questions	Quantitative and qualitative data	Can reach many people with low resource	The design is crucial. Response rate may be low. Responses may not be what you want
Interviews	Exploring issues	Some quantitative but mostly qualitative data	Interviewer can guide interviewee. Encourages contact between developers and users	Time consuming. Artificial environment may intimidate interviewee
Focus groups and workshops	Collecting multiple viewpoints	Some quantitative but mostly qualitative data	Highlights areas of consensus and conflict. Encourages contact between developers and users	Possibility of dominant characters
Naturalistic observation	Understanding context of user activity	Qualitative	Observing actual work gives insight that other techniques cannot give	Very time consuming. Huge amounts of data
Studying documentation	Learning about procedures, regulations, and standards	Quantitative	No time commitment from users required	Day-to-day work will differ from documented procedures

DMIoT, School of Computing

Software Engineering Academic Program

Requirements Engineering (SEng3053)

By

Yayehudar T.

Chapter One

Introduction to Requirements Engineering

What are requirements?

- ✓ The software requirements are description of **features** and functionalities of the target system.
- ✓ Something required, something wanted or needed.
  - – Webster's dictionary
- ✓ Requirements convey the expectations of users from the software product.
- ❑ Requirements are defined during the **early stages of a system development**
  - as a specification of what should be implemented or as a constraint of some kind on the system.

Cont'd…

- **Requirements might describe:**
  - a user-level facility description(e.g. 'the word processor must include a spell checking and correction command')
  - a detailed specification of expected system behavior
  - a very general system property (e.g. 'the system must ensure that personal information is never made available without authorization')
  - a specific constraint on the system (e.g. 'the sensor must be polled 10 times per second')
  - information on how to carry out some computation (e.g. 'the overall mark is computed by adding the student examination, project and coursework marks based on the following formula 'total_mark = exam_mark + 2 * project_mark + 2/3 * coursework_mark')
  - A constraint on the development of the system (e.g. 'the system must be developed using Ada').

Cont'd…

- Some people suggest that requirements should always be statements of *what* a system should do rather than a statement of **how it should do it.**

- This is an attractive idea but it is too simplistic in practice.

1. The readers of a document are often practical engineers who can relate to implementation descriptions much better than they can understand very abstract problem statements.
   - ✓ You have to write requirements which are understandable to the likely readers of the document.
2. In almost all cases, the system being specified is only one of several systems in an environment.
   - To be compatible with its environment, and to conform to standards and with organizational concerns, you may have to specify implementation policies which constrain the options of the system designers.

Cont'd…

- Requirements therefore invariably contain a mixture of problem information, statements of system behaviour and properties and design and manufacturing constraints.

What is Requirements Engineering

- Requirements engineering is a relatively new term which has been invented to cover all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system.
- The use of the term 'engineering' implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, relevant, etc.
- The term 'requirements engineering' has come from a systems engineering background;

Cont'd…

- The **goal** of requirement engineering is to develop and maintain refined and descriptive 'System Requirements Specification' document.
- In practice, requirements engineering isn't sequential process, it's an **iterative process** in which activities are interleaved.
  - For example, you iterate first on the user requirements; elicitation, specification, and validation, and repeat the same steps for the system requirements.

What is a Requirements Document?

- The requirements document is an official statement of the system requirements for customers, end-users and software developers.
- Depending on the organization, the requirements document may have different names such as the 'functional specification', 'the requirements definition', 'the software requirements specification (SRS)', 'the safety/reliability plan', etc.

What is the Best Way to Write Requirements?

- There is no best way to write requirements.
  - ✓ It depends on normal organizational practice and the notations which are used by writers and readers of the requirements.
- Requirements may be stated in a language which reflects the background of the requirement source.
  - If the source is an engineer, it may be written in engineering terms;
  - if the source is a manager, it will be written in natural language.
- Most requirements are written as natural language sentences supplemented with diagrams and tables of detailed information.

How Detailed Should Requirements Be?

- Different organizations think of requirements in different ways and write them at different levels of detail.

- A requirement for controlling a motor may include details of the specific control signals to be used; a requirement for managing customer information might simply state the information to be provided.
- The system designer decides how to organize and present this information.

Cont'd…

- The level of detail that you need largely depends on the normal practice in your organization, whether or not the requirements document will be the basis of a contract for software development and the type of system which is being developed.
- ➢ If you are developing a product which you both specify and implement, you may produce a fairly general specification.
- You will add details to this as the system development proceeds.
- ➢ On the other hand, if you are contracting the system development to another company, you need a much more detailed specification which defines what they must implement.
- ❑ In some organizations, requirements are first expressed as an informal high-level description which is then developed into a more detailed specification.
- The abstract requirements are the requirements of the stakeholders of the system and the detailed requirements are a system specification.

Cont'd…

1. **Stakeholder requirements**

- sometimes called user requirements.
- These are requirements which are written from the point-of-view of system stakeholders.
- They are not usually expressed in great detail.
- They are described in natural language, informal diagrams or using some notation which is appropriate to the problem being solved (e.g. mathematical equations for a control system).

2. **System requirements**

- These are more detailed specifications of requirements which may be expressed as an abstract model of the system.
- This may be a mathematical model or may be based on graphical notations such as data-flow diagrams, object class hierarchies, etc.
- These models are always annotated with natural language descriptions.

Why is Requirements Engineering difficult?

- Businesses operate in a rapidly changing environment so their requirements for system support are constantly changing.
- Multiple stakeholders with different goals and priorities are involved in the requirements engineering process.
- System stakeholders do not have clear ideas about the system support that they need.
- Requirements are often influenced by political and organisational factors.

Software Requirements Engineering: What, why, Who, When, and How

- If software requirement is not right, companies will not end-up with software they need.

- So it is necessary to find out the what, why, who, when, and how of the system.

- **What**: The various levels and types of requirements that need to be defined
- **Why**: The benefits of having the right software requirements
- **Who**: The stakeholders of the software requirements and getting them involved in the process
- **When**: Requirements activities throughout the software development life cycle
- **How**: Techniques for eliciting, analyzing, specifying, and validating software requirements

Cont'd…

FR ,NFR and Constraints

What:

- Requirements must be determined and agreed to by the customers, users, and suppliers of a software product before the software can be built.

❑ The requirements define the "what" of a software product:

✓ What the software must do to add value for its stakeholders.

- These functional requirements define the capabilities of the software product.

✓ What the software must be to add value for its stakeholders.

- These nonfunctional requirements define the characteristics, properties, or qualities that the software product must possess.
- They define how well the product performs its functions.

✓ What limitations there are on the choices that developers have when implementing the software.

- The external interface definitions and other constraints define these limitations.

Cont'd…

WHY:

- Eliciting, analyzing, and writing good requirements are the most difficult parts of software engineering.

- According to Karl Wiegers (2004)

"If you don't get the requirements right, it doesn't matter how well you do anything else."

- One can end up doing a perfect job of building the wrong product.

Cont'd…

- There are many issues that can have a negative impact on software development projects and products if **practitioners** don't do a good job of defining their software requirements.

- These issues include:
  - Incomplete requirements
  - Lack of user involvement
  - Requirements churn(mix)
  - Wasted resources
  - Gold plating
  - Inaccurate estimates

- If the requirements are incomplete, software practitioners end up building a software product that does not meet all of the customer and user's needs.

Cont'd…

WHO:

- Stakeholders are individuals who affect or are affected by the software product and therefore have some level of influence over the requirements for that software product.
- There are three main categories of stakeholders:
  - the **acquirers**,
  - **suppliers** of the software product, and
  - **other stakeholders**.

❑ The ***acquirer*** type stakeholders can be divided into two major groups.

- The **Customers** - who request, purchase, and/or pay for the software product for their business objectives.
- The **Users(end-users)** - who actually use the product directly.

Cont'd…

- The ***suppliers*** of the software product include individuals and teams that are part of the organization that **develop**, **distribute** or **outsource the product**.
- ***Other stakeholders*** include:

✓ the **requirements analyst,** also called the business analyst or system analyst, is responsible for eliciting the requirements from customers, users and other stakeholders, analyzing the requirements, writing requirement specifications and communicating the requirements to development and other stakeholders.

✓ The ***designers*** translates the requirements into the software's architectural and detailed designs that specify how the software will be implemented.

✓ The ***developers*** implements the designs by creating the software products.

✓ The ***testers*** use the requirements as a basis for creating test cases, that they use to execute the software under specific, known conditions to detect defects and ensure the software performs as specified.

Cont'd…

- ✓ The **documentation writers** are responsible for using the requirements as inputs into the creation of the user documentation including user/operations manuals, help files, installation instructions, and training materials as necessary.
- ✓ The **project managers** are responsible for planning, monitoring, and controlling the project and guiding the software development team to the successful delivery of the software.
- ✓ **Technical support**, also called operations or the help desk, is responsible for interfacing with the user community to support the software once it has been deployed to the field.
- ✓ **Product change management**, which may take the form of a change control board (CCB), is responsible for reviewing proposed changes to the requirements, analyzing their impacts, approving/disapproving changes, and ensuring that approved changes are implemented and validated.

Cont'd…

- There may also be other stakeholders interested in the requirements.
- Examples of other requirements stakeholders include:
  - Legal or contract management
  - Manufacturing or product release management
  - Sales and marketing
  - Upper management
  - Government or regulator agencies
  - Society at large

Cont'd…

WHEN:

- Requirements development encompasses all the activities involved in identifying, capturing, and agreeing upon the requirements.
- This includes the definition and integration of the business requirements, the user requirements, and software product level requirements.

❖ The majority of the requirements development activities occur during the early concept and requirements phases of the life cycle.

✓ Continued elaboration of the requirements, however, can progress into the later phase of the software development life cycle.

Cont'd…

HOW:

- ❑ Software requirements engineering is made up of two major processes: requirements development and requirements management.
- ✓ **Requirements development** encompasses all of the activities involved in eliciting, analyzing, specifying, and validating the requirements.
- ✓ **Requirements management** encompasses the activities involved in requesting changes to the baselined requirements, performing impact analysis for the requested changes, approving or disapproving those changes, and implementing the approved changes.

Functional and Non-functional Requirements

Functional Requirements

- Functional requirements define what a system is supposed to *do*.
- Functional requirements are usually in the form of "system shall do <requirement>", an individual action or part of the system.
- They define functions and functionality within and from the software system.
- Functional requirement captures the behavioral aspects / function of the proposed automated system.
- Functional requirements are the back bone of all software products.

Cont'd…

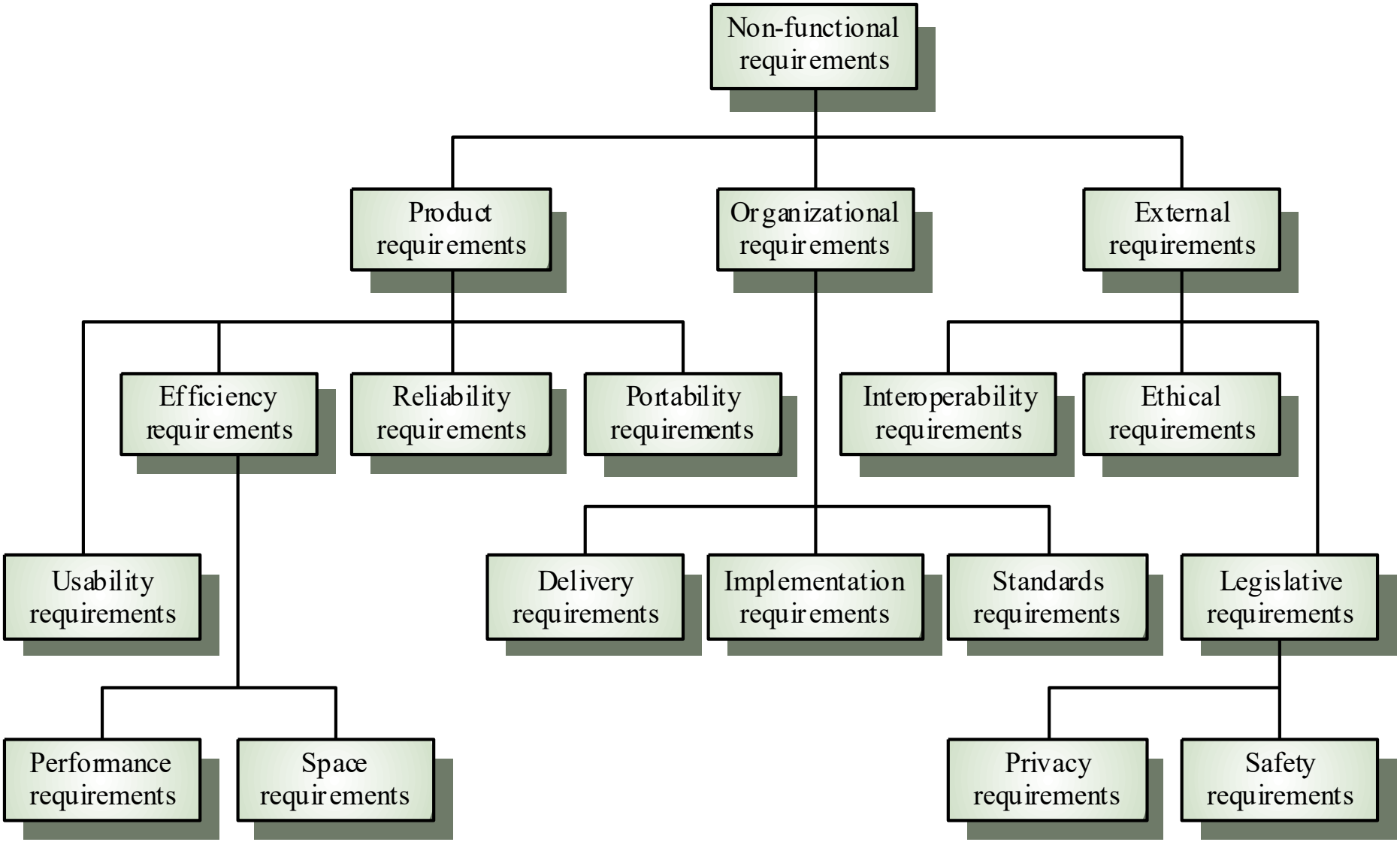
❖ **Examples of functional requirement:**

- **Requirement #1:**
  - The user shall be able to search the entire database of patients.
- **Requirement #2:**
  - The system shall provide appropriate viewers for the user to read documents in the document store.

Non-Functional Requirements

- Non-functional requirements define **how a system is supposed to *be*.**
- Non-functional requirements are in the form of "system shall be <requirement>", an overall property of the system as a whole or of a particular aspect and not a specific function.
- Non-functional requirements are often called "quality attributes" of a system.
- Other terms for non-functional requirements are "qualities", "quality goals", "quality of service requirements", "constraints" and "non-behavioral requirements".
- Informally these are sometimes called the "ilities", from attributes like stability and portability.

Types of Non-Functional Requirements (NFRs)



Product-oriented attributes

- ✓ **Performance :** (a) response time, (b) throughput (number of operations performed per second)
- ✓ **Usability:** effort required to learn, use, provide input and interpret results of a program
- ✓ **Efficiency:** minimal use of resources (memory, processor, disk, network…)
- ✓ **Reliability:** of computations, precision
- ✓ **Security:** resistance to unauthorized attempts
- ✓ **Robustness:** in the presence of faults, stress, invalid inputs…

Cont'd…

- ✓ **Adaptability:** to other environments or problems
- ✓ **Scalability:** for large number of users or quantities of data
- ✓ **Cost:** total cost of ownership (TCO) for acquisition, installation.
- **Portability:** does it work for several platforms
- **Modifiability:** addition of new functionalities
- **Reusability:** of components, code, designs, and even requirements in other systems

Process-oriented attributes

– **Maintainability:** changes to functionalities, repairs

– **Readability:** of code, documents

– **Testability:** ease of testing and error reporting

– **Understandability:** of design, architecture, code

– **Integrability:** ability to integrate components

– **Complexity:** degree of dependency and interaction between components

Thank You!

?

DMIoT, School of Computing

Software Engineering Academic Program

Requirements Engineering (SEng3053)

By

Yayehudar T.

Chapter Two

Requirements Engineering Process

What is a Requirements Engineering Process?

- A requirements engineering process is a structured set of activities which are followed to derive, validate and maintain a systems requirements document.

❑ A complete process description should include:

- ✓ what activities are carried out,
- ✓ the structuring or schedule of these activities,
- ✓ who is responsible for each activity,
- ✓ inputs and outputs to/from the activity and
- ✓ the tools used to support requirements engineering.

Cont'd…

- Very few organizations have an explicitly-defined and standardized requirements engineering process.
  - ✓They simply define the result of the process, namely the requirements document.
  - ✓The people involved in the process are responsible for deciding:
    - what to do and when to do it,
    - what information they need,
    - what tools they should use, etc.

Can You Suggest a Good Requirements Engineering Process?

- We cannot do this without knowing about:
  - your organization,
  - your systems engineering and software development processes and
  - the type of software which you develop.
- **To define a good requirements engineering process:**
  - you need to involve people from your organization who are actually involved in requirements engineering.
  - You may have to get outside help from consultants, as they can take a more objective perspective than those involved in the process.

Cont'd…

- Most of the standards which have been developed for requirements engineering are concerned with process outputs such as the structure of requirements documents.

➢ Within general software engineering standards such as the US DoD standard 2167A, there is some mention of requirements engineering activities.

❑ However, we would normally expect **a good requirements engineering process to include the following activities.**

Cont'd…

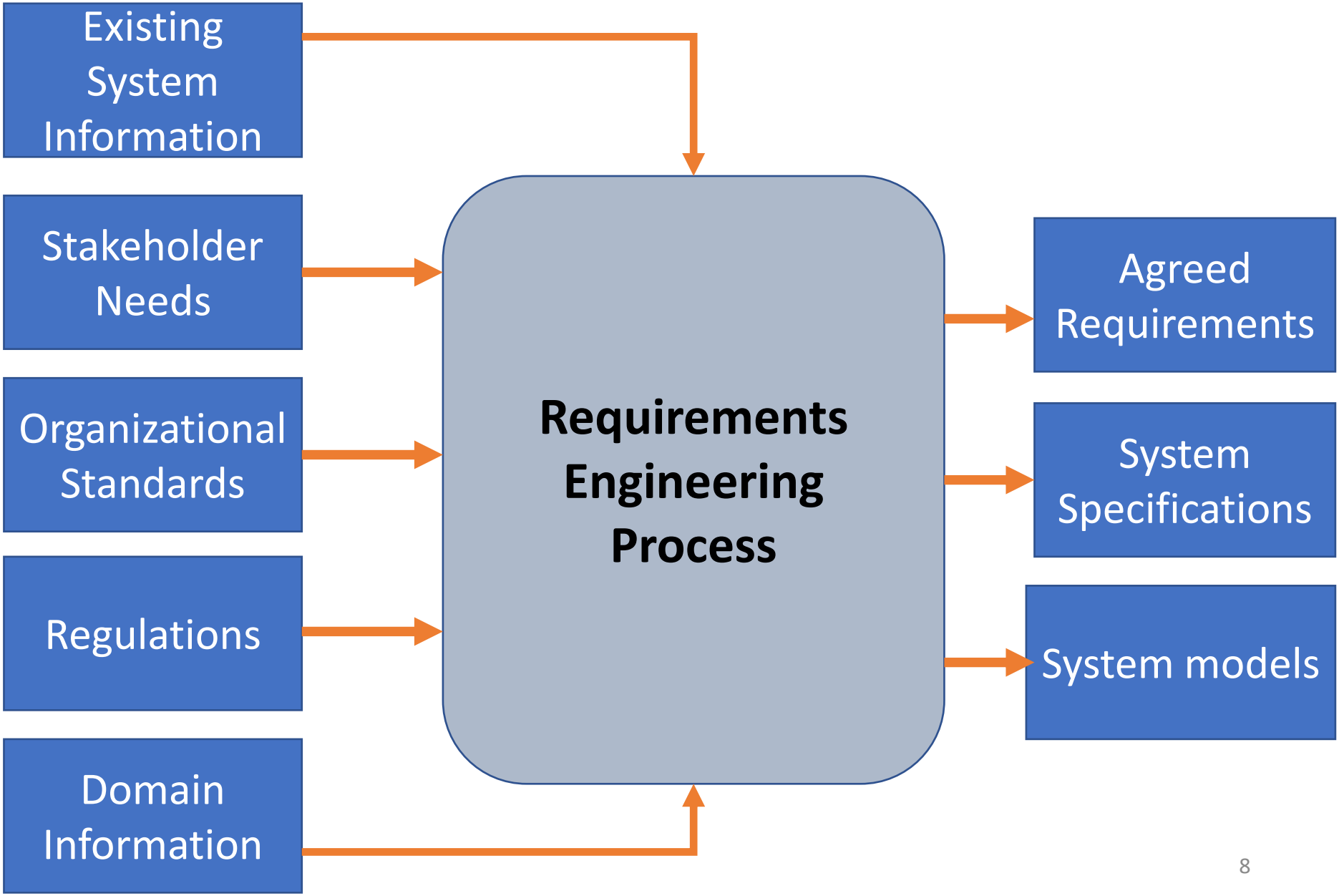
1. **Requirements elicitation**. The system requirements are discovered through consultation with stakeholders, from system documents, domain knowledge and market studies.

2. **Requirements analysis and negotiation**. The requirements are analyzed in detail, and should be some formal negotiation process involving different stakeholders to decide on which requirements are to be accepted.

3. **Requirements validation**. These should be a careful check of the requirements for consistency and completeness.

   - To support these activities, a requirements management process should be introduced to manage changes to the requirements.

RE process - inputs and outputs



Input/Output description

Input or output	Type	Description
Existing system information	Input	Information about the functionality of systems to be replaced or other systems which interact with the system being specified
Stakeholder needs	Input	Descriptions of what system stakeholders need from the system to support their work
Organisational standards	Input	Standards used in an organisation regarding system development practice, quality management, etc.
Regulations	Input	External regulations such as health and safety regulations which apply to the system.

Cont'd…

Input or output	Type	Description
Domain information	Input	General information about the application domain of the system.
Agreed requirements	Output	A description of the system requirements which is understandable by stakeholders and which has been agreed by them.
System specification	Output	This is a more detailed specification of the system functionality which may be produced in some cases.
System models	Output	A set of models such as a data-flow model. an object model, a process model, etc. which describes the system from different perspectives

RE process variability

- RE processes vary radically from one organization to another.
- **Factors contributing to this variability include:**
  - ✓ **Technical maturity**: The technologies and methods used for requirements engineering vary from one organization to another.
  - ✓ **Disciplinary involvement**: the types of engineering and managerial disciplines involved in requirements engineering vary from one organization to another.

Cont'd…

- ✓ **Organizational culture**: The culture of an organization has an important effect on all business processes and, as the culture varies, so too does the requirements engineering process.
- ✓ **Application domain**: Different types of application system need different types of requirements engineering process.

- ❖ There is therefore no '**ideal**' requirements engineering process

Process models

- A process model is a simplified description of a process presented from a particular perspective.
- Types of process model include:
  - ✓ Coarse-grain activity models
  - ✓ Fine-grain activity models
  - ✓ Role-action models
  - ✓ Entity-relation models

Cont'd...

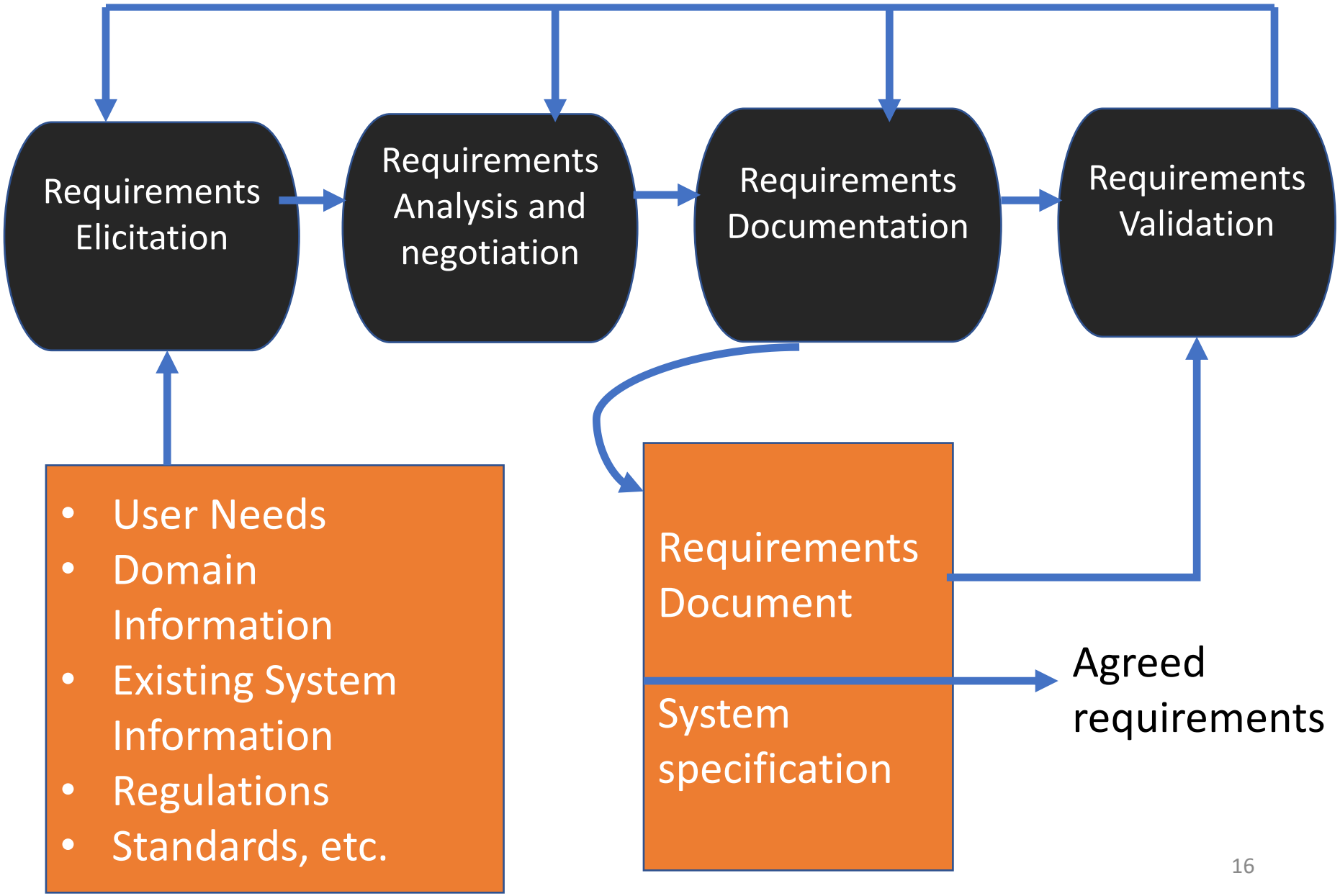
1. Coarse-grain activity models

- Shows the major activities involved in a particular process and their approximate sequencing.
  - The sequencing is approximate because, in reality, processes overlap and are interleaved - they rarely follow a simple sequential pattern.
- This type of model doesn't tell us how to enact a process but provides an overall picture of the process.

Cont'd…

- Coarse-grain activity models describe the context of the different activities in the process.
- They show other processes which provide inputs to and consume outputs from a specific process.
- The ff figure is an example of a software life cycle model which shows the context of the software requirements engineering process in the whole of the software development process.

Cont'd...


Requirements Elicitation
Requirements Analysis and negotiation
Requirements Documentation
Requirements Validation
User Needs
Domain Information
Existing System Information
Regulations
Standards, etc.
Requirements Document
System specification
Agreed requirements

Cont'd...

2. Fine-grain activity models

- These are more detailed models of a specific process.
- They may be used for understanding and improving existing processes.

Cont'd...

3. Role-action models

- These are models which show the roles of different people involved in the process and the actions which they take.
- These models may be helpful for process understanding and automation.

4. Entity-relation models

- These models show the process inputs, outputs and intermediate results and the relationships between them.
- They may be used in a quality management system and to supplement models of process activities.

RE process activities

I. Requirements elicitation

- The system requirements are discovered:
  - ✓ through consultation with stakeholders,
  - ✓ from system documents,
  - ✓ from domain knowledge and
  - ✓ from market studies.
- Other names for this process are requirements acquisition or requirements discovery.

Cont'd…

II. Requirements analysis and negotiation

- The requirements are analyzed in detail and different stakeholders negotiate to decide on which requirements are to be accepted.
- There is usually some flexibility in requirements and negotiation is necessary to decide on the set of agreed requirements for the system.

Cont'd…

III. Requirements documentation

- The agreed requirements are documented at an appropriate level of detail.
- In general, there needs to be a requirements document which is understandable by all system stakeholders.
- The requirements must be documented using natural language and diagrams.
- More detailed system documentation, such as system models may also be produced.

Cont'd…

IV. Requirements validation

- There should be a careful check of the requirements for consistency and completeness.
- This process is intended to detect problems in the requirements document before it is used as a basis for the system development.

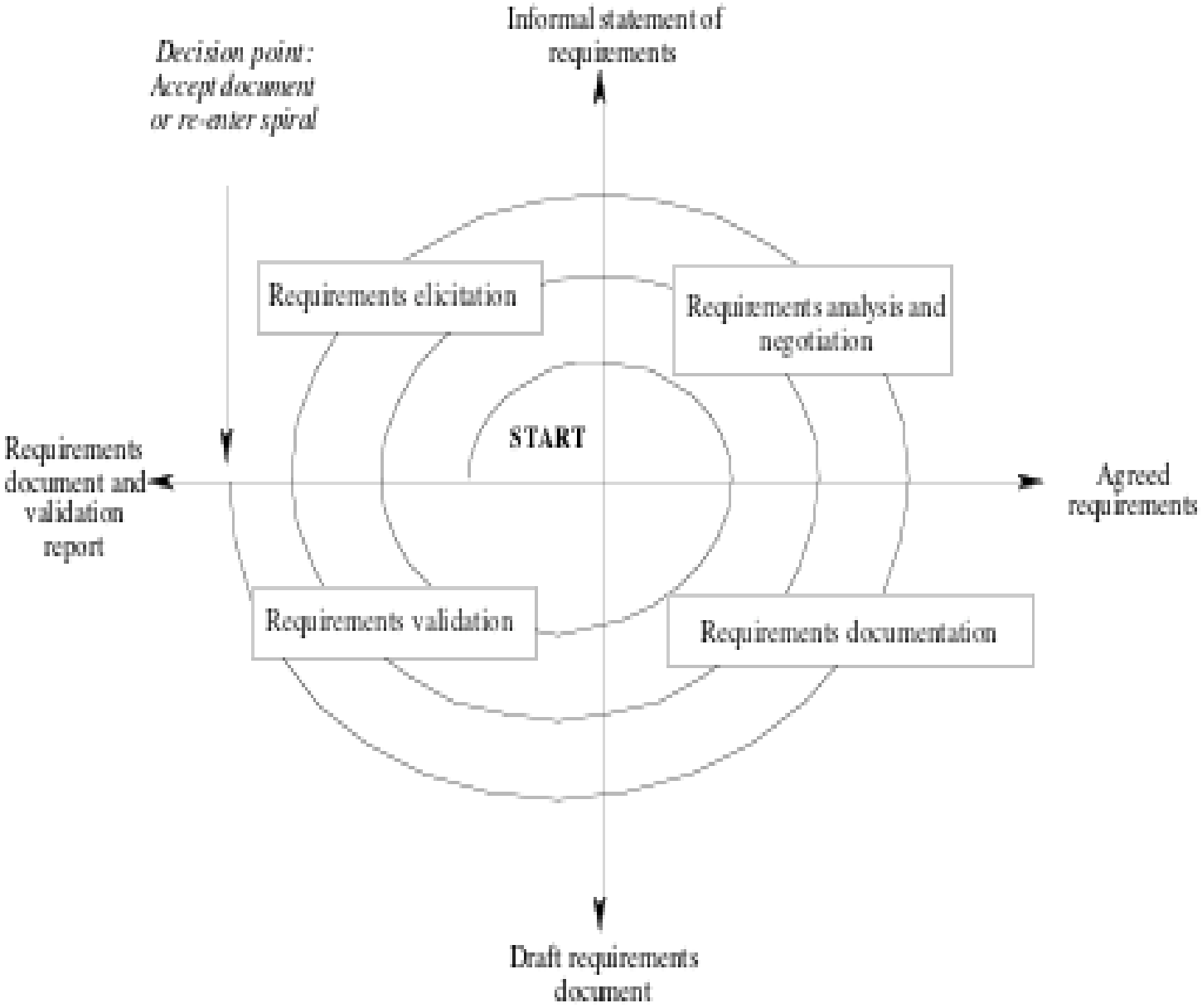
Cont'd…

- In parallel with all of the above processes is a process of requirements management which is concerned with managing changes to the requirements.

- Changing requirements are inevitable as:
  - business priorities change,
  - errors or omissions in the requirements are discovered and
  - new requirements emerge.

- Requirements management is intended to keep track of these changes and ensure that changes are made to the requirements document in a controlled way.

Spiral model of the RE process



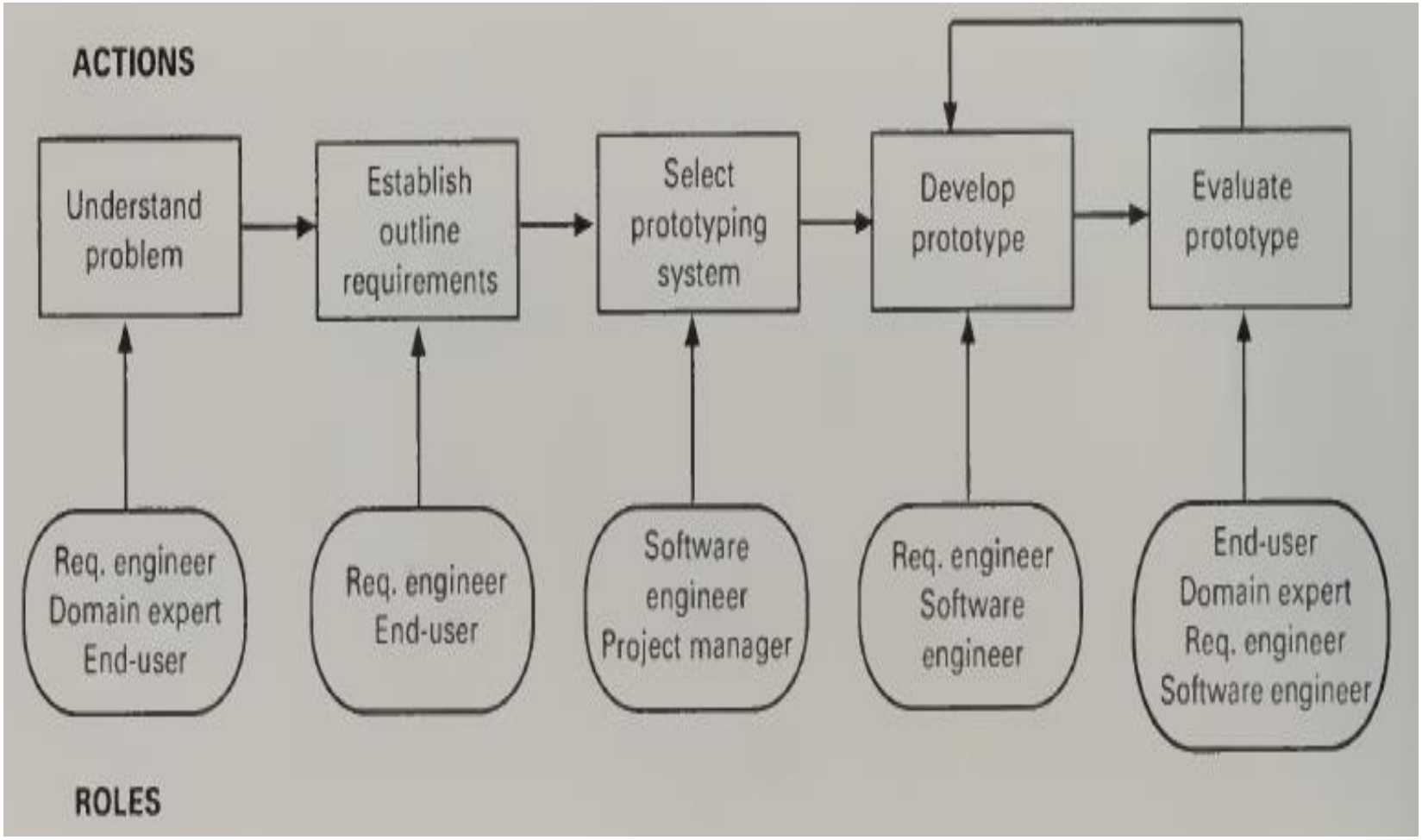
Actors in the RE process

- Actors in a process are the people involved in the execution of that process.
- Actors are normally identified by their roles, e.g., project manager, purchasing director, system engineer rather than as individuals.
- Requirements engineering involves actors who are primarily interested in the problem to be solved (end-users, etc.) as well actors interested in the solution (system designers, etc.)

➢ Role-action diagrams document which actors are involved in different activities.

RAD for software prototyping

- The following figure is a model of part of the requirements elicitation process where a prototype software system is being developed.



Role descriptions

Role	Description
Domain Expert	Responsible for providing information about the application domain and the specific problem in that domain which is to be solved.
System end-user	Responsible for using the system after delivery.
Requirements Engineer	Responsible for eliciting and specifying the system requirements.
Software Engineer	Responsible for developing the prototype software system.
Project Manager	Responsible for planning and estimating the prototyping project.

Human and social factors (RE processes)

- Requirements engineering processes are dominated by human, social and organizational factors because they always involve a range of stakeholders from different backgrounds and with different individual and organizational goals.
- System stakeholders may come from a range of technical and non-technical background and from different disciplines.
- They may come from different engineering disciplines.
- They may be responsible for management and may be internal or external to an organization.

Cont'd…

- **Examples of these different stakeholder types are:**
  - **Software engineers** responsible for system development.
  - **System end-users** who will use the system after it has been delivered.
  - **Managers of system end-users** who are responsible for their work.
  - **External regulators** who check that the system meets its legal requirements.
  - **Domain experts** who give essential background information about the system application domain.

Cont'd…

- Typically, these people come from different departments in an organization and they are usually responsible for many other things apart from the requirements engineering of a system.
- They will **not necessarily give priority** to the requirements engineering process.
- It is likely that each group of stakeholders will **have different goals**.
- They will **try to influence the requirements** so that their goals are met without necessarily taking the goals of other stakeholders into account.
- Whether or not a stakeholder succeeds in influencing the requirements often depends on personality and status and not necessarily on reasoned argument.

Cont'd…

- **Factors influencing requirements**
  - Personality and status of stakeholders
  - The personal goals of individuals within an organization
  - The degree of political influence of stakeholders within an organization.

Process support

- The need to provide some automated support for software processes has been recognized since the 1980s.
- This led to the development of a large number of CASE (Computer-Aided Software Engineering) tools which supported various process activities such as software design, configuration management and testing.
- CASE tools provide automated support for software engineering processes.
- The most mature CASE tools support well-understood activities such as programming and testing and the use of structured methods.
- Support for requirements engineering is still **limited** because of the informality and the variability of the process.

Cont'd…

CASE tools for RE

I. Modelling and validation tools: support the development of system models which can be used to specify the system and the checking of these models for completeness and consistency.

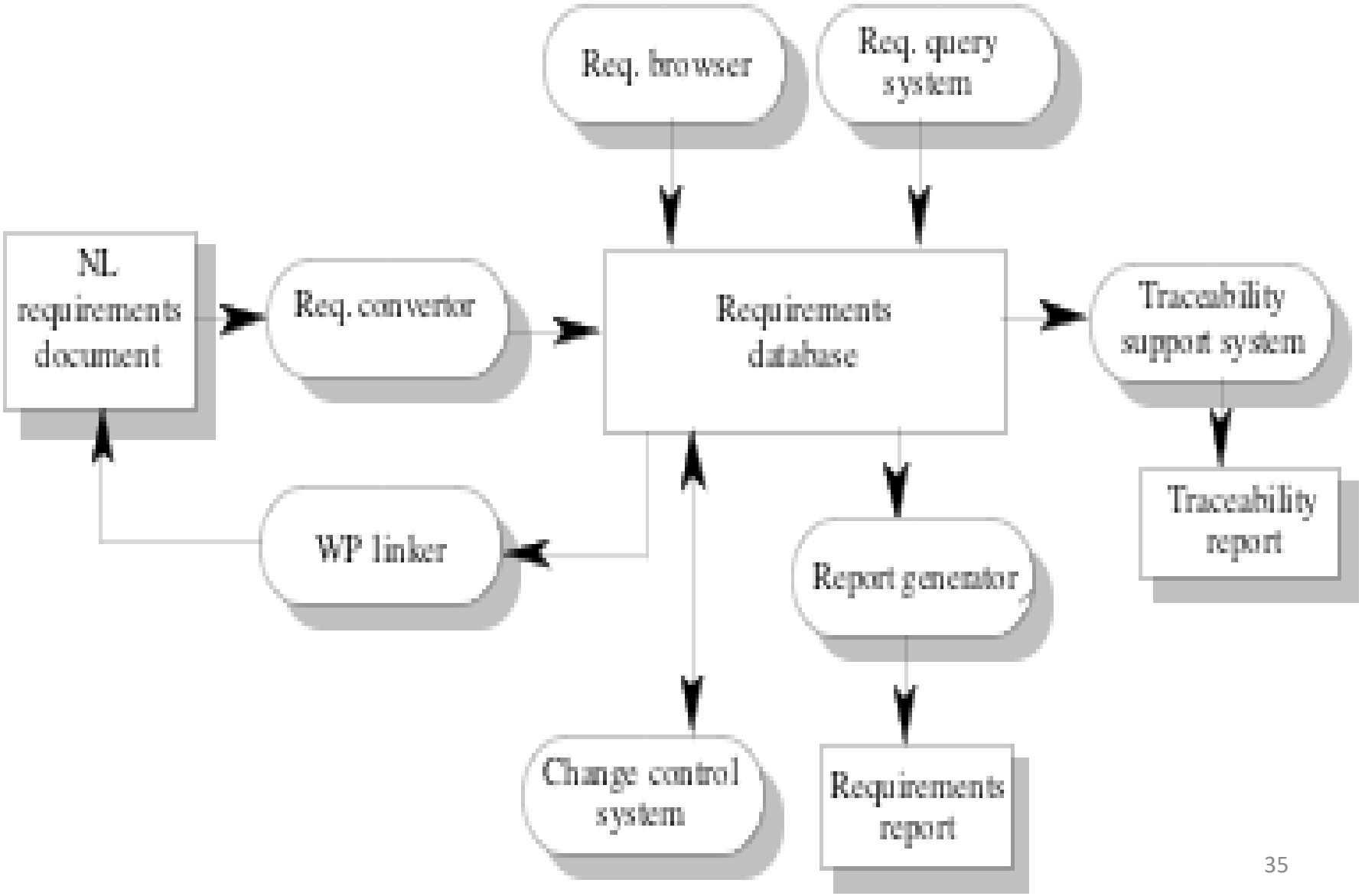
II. Management tools: help manage a database of requirements and support the management of changes to these requirements.

Cont'd…

- Tools for requirements management have been developed because of:
  - the problems of managing the large amount of data which is collected during the requirements engineering process and
  - the volatility of the requirements.
- As well as data storage and retrieval facilities, these tools:
  - provide a change management system which can record change proposals,
  - link these to the requirements affected and
  - keep track of the status of the changes.
- Requirements histories can be constructed, and some traceability support which allows the impact of changes to be assessed is usually provided.

Cont'd…

A requirements management system



Process improvement

- Process improvement is concerned with modifying processes in order to meet some improvement objectives.
- Improvement objectives
  - Quality improvement
  - Schedule reduction
  - Resource reduction

Planning process improvement

- The following questions need to be answered while planning for process improvement.
  - What are the problems with current processes?
  - What are the improvement goals?
  - How can process improvement be introduced to achieve these goals?
  - How should process improvements be controlled and managed?

Cont'd…

1. What are the problems with current processes?

- These may be identifiable problems such as **late delivery of products**, **budget over-runs**, **poor quality products**, etc.
- They may be less tangible problems such as **poor staff morale**, a reluctance in people to take responsibility or a meeting-dominated process where people spend too much time in meetings.
- Alternatively, the key problems might be problems of process understanding - no-one actually knows what processes are followed.

Cont'd…

2. What are the improvement goals?

- These should normally be related to the identified problems.
- For example, if an organization has quality management problems, its goal may be **to improve quality management procedures to satisfy the ISO 9000 certification standard** (Ince, 1994).
- If there are problems with budget-overruns, the goal may be **to reduce the amount of rework which is required in a process**. Goals must be realistic.
- There is no point in setting unrealizable goals or having unrealistic expectations about the benefits of new techniques or methods.

Cont'd…

3. How can we introduce process improvements to achieve these goals?

- This involves **assessing the existing requirements engineering processes** to find out the activities which cause most problems and identifying changes to these which will contribute to the improvement goals.

Cont'd…

4. How should improvements be controlled and managed?

- Procedures to **collect feedback on improvements**, which may be either quantitative measurements of the process or informal comments on the improvements, must be put in place.
- You should also ensure that action is taken in response to this feedback to correct any identified problems.

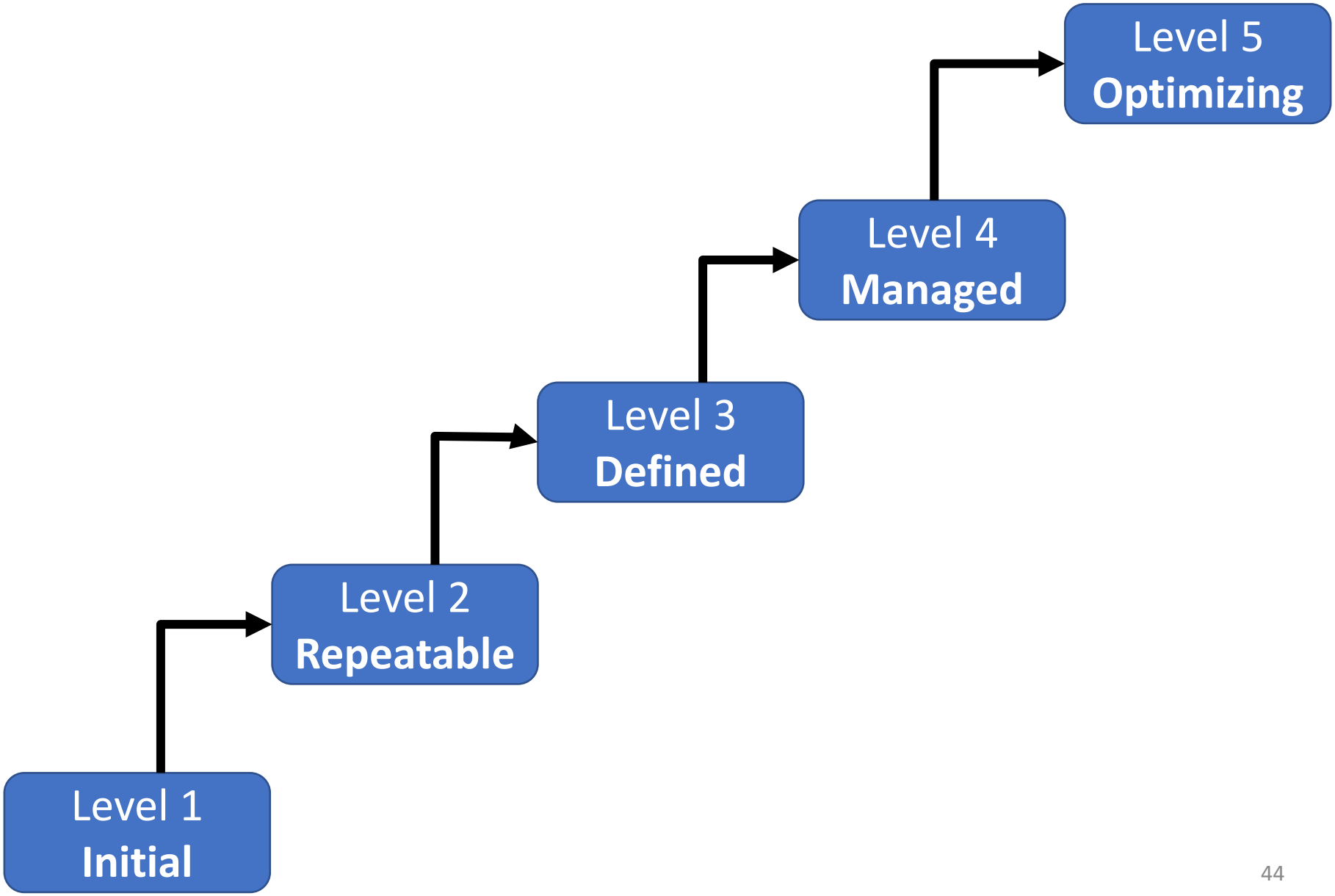
RE process problems

- Lack of stakeholder involvement
- Business needs not considered
- Lack of requirements management
- Lack of defined responsibilities
- Stakeholder communication problems
- Over-long schedules and poor quality requirements documents

Process maturity

- A particularly important factor which affects process improvement is the level of process maturity in an organization.
- **Process maturity** can be thought of as the extent that an organization has defined its processes, actively controls these processes and provides systematic human and computer-based support for them.
- The **SEI's Capability Maturity Model** is a framework for assessing software process maturity in development organizations.

Capability maturity model



Maturity levels

1. **Initial level**

   - Organizations have an undisciplined process and it is left to individuals how to manage the process and which development techniques to use.

2. **Repeatable level**

   - Organizations have basic cost and schedule management procedures in place.
   - They are likely to be able to make consistent budget and schedule predictions for projects in the same application area.

Cont'd…

3. Defined level

- The software process for both management and engineering activities is documented, standardized and integrated into a standard software process for the organization.

Cont'd…

4. **Managed level**
   - Detailed measurements of both process and product quality are collected and used to control the process.
5. **Optimizing level**
   - The organization has a continuous process improvement strategy, based on objective measurements, in place.

RE process maturity model

Level 1- Initial
- Ad-hoc requirements engineering
- Requirements problems are common

Level 2- Repeatable
- Standardized requirements engineering
- Fewer requirements problems

Level 3- Defined
- Defined process based on best practice
- process improvement in place

RE process maturity levels

- **Initial level**
  - No defined RE process.
  - Suffer from requirements problems such as requirements volatility, unsatisfied stakeholders and high rework costs.
  - Dependent on individual skills and experience.
- **Repeatable level**
  - Defined standards for requirements documents and policies and procedures for requirements management.
- **Defined level**
  - Defined RE process based on good practices and techniques.
  - Active process improvement process in place.

Good practice for RE process improvement

- RE processes can be improved by the systematic introduction of good requirements engineering practice.
- Each improvement cycle identifies good practice guidelines and works to introduce them in an organization.

Cont'd…

Examples of good practice guidelines

- Define a standard document structure
- Uniquely identify each requirement
- Define policies for requirements management
- Use checklists for requirements analysis
- Use scenarios to elicit requirements
- Specify requirements quantitatively
- Use prototyping to animate requirements
- Reuse requirements

Thank You!

?

Chapter One

Overview of Software Requirements Engineering

Introduction

Requirements

- Requirements are defined during the early stages of a system development as a specification of what should be implemented or as a constraint of some kind on the system.
- **They may be:**
  - a user-level facility description,
  - a detailed specification of expected system behavior,
  - a general system property,
  - a specific constraint on the system,
  - information on how to carry out some computation,
  - A constraint on the development of the system.

Requirements Engineering

- Refers to the process of defining, documenting and maintaining requirements in the engineering design process.
  - It is a common role in systems engineering and software engineering.
- The process of establishing the services that the customer requires from a system and the constraints under which it operates and is developed.
- The requirements themselves are the descriptions of the system services and constraints that are generated during the requirements engineering process.
- Requirements specify *what* the system is supposed to do, but not *how* the system is to accomplish its task

Cont'd…

- The **process** to gather the software requirements from client, analyse and document them is known as **requirement engineering**.
- The **goal** of requirement engineering is to develop and maintain refined and descriptive 'System Requirements Specification' document.

❖ In practice, requirements engineering isn't sequential process, it's an iterative process in which activities are interleaved.

  – For example, you iterate first on the user requirements; elicitation, specification, and validation, and repeat the same steps for the system requirements.

- Each software development process goes through the phase of requirements engineering
- The use of the term **'engineering'** implies that systematic and repeatable techniques should be used to ensure that system requirements are complete, consistent, relevant, etc.

What is a requirement?

- The **software requirements** are description of features and functionalities of the target system.
- Requirements convey the expectations of users from the software product.
- The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

❖ **Types of requirements**

- ***User requirements***
- ***System requirements***
  - Software specifications – provide more (design) detail

Cont'd…

User requirements

- Should describe functional and non-functional requirements so that they are understandable by system users who don't have detailed technical knowledge
- User requirements are defined using natural language, tables, and diagrams

Problems with natural language

- Precision vs. understandability
- Functional vs. non-functional requirements confusion
- Requirements amalgamation

Cont'd…

System requirements

- More detailed specifications of user requirements
- Serve as a basis for designing the system
- May be used as part of the system contract
- System requirements may be expressed using system models.

Cont'd…

What happens if the requirements are wrong?

- The system may be delivered late and cost more than originally expected.
- The customer and end-users are not satisfied with the system.
- The system may be unreliable in use with regular system errors and crashes disrupting normal operation.
- If the system continues in use, the costs of maintaining the system is very high.

Cont'd…

Why is requirements engineering difficult?

- Businesses operate in a rapidly changing environment so their requirements for system support are constantly changing.
- Multiple stakeholders with different goals and priorities are involved in the requirements engineering process.
- System stakeholders do not have clear ideas about the system support that they need.
- Requirements are often influenced by political and organisational factors.

Requirement Engineering Process

- The **goal** of RE process is to create and maintain a system requirements documents.
- The processes used for RE vary widely depending on the application domain, the people involved and the organization developing the requirements.
  - However, there are a number of generic activities common to all processes.

❖ It is a four step process, which includes –

A. Feasibility Study

B. Requirements elicitation and Analysis

C. Software Requirements Specification

D. Software Requirements Validation

A. Feasibility study

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

Cont'd…

Feasibility Study Implementation

- Based on information assessment (what is required), information collection and report writing.

❖ **Questions for people in the organization for information assessment and collection:**

- 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?

Cont'd…

Feasibility Study Implementation ……

- What facilities must be supported by the proposed system?
  - Feasibility study report should make a recommendation about the development to continue or not.
- Is new technology needed? What skills?
- What facilities must be supported by the proposed system?

B. Requirements Elicitation and Analysis

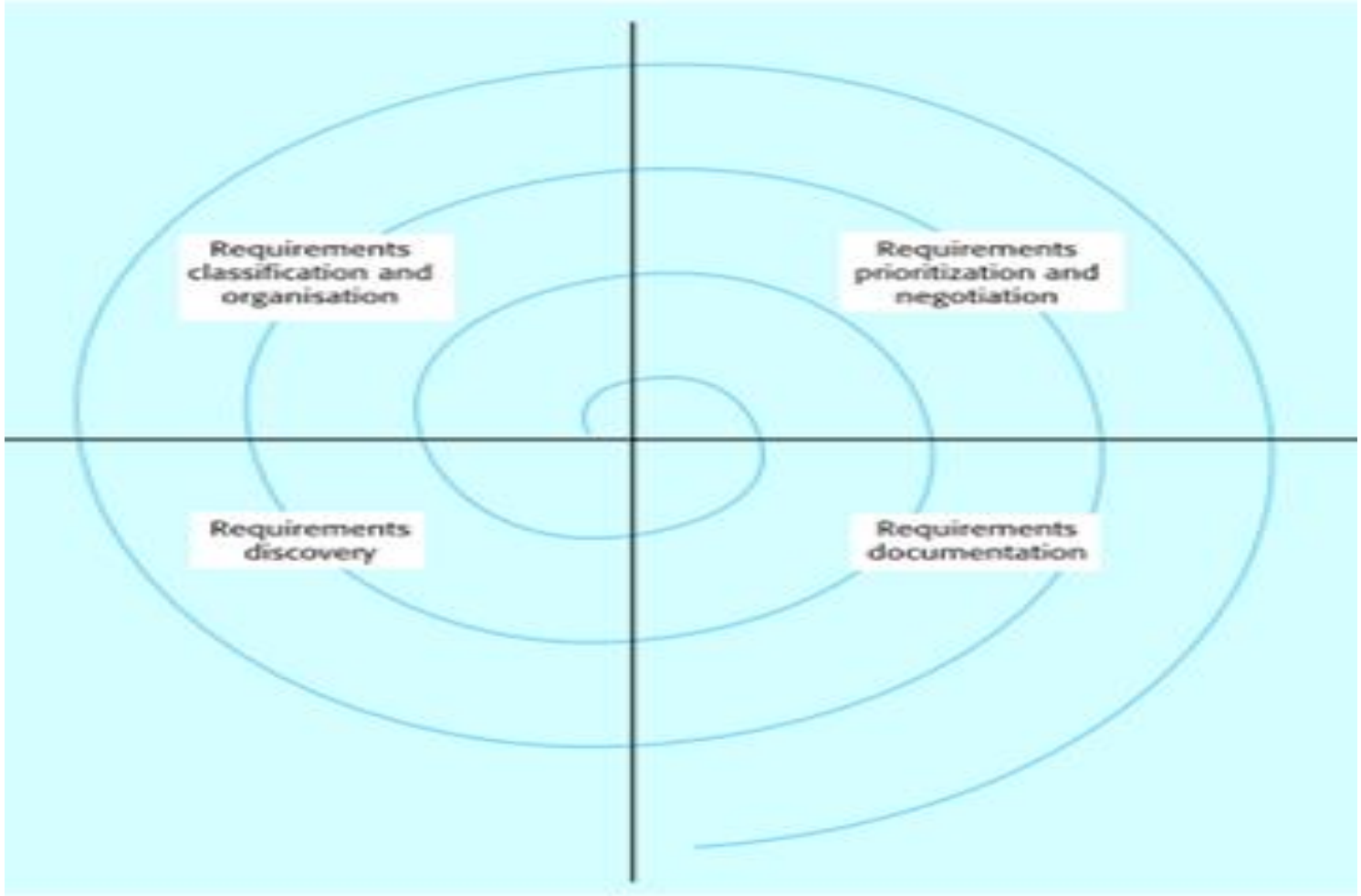
- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved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

Cont'd…

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.
- Organizational and political factors may influence the system requirements.
- The requirements change during the analysis process.
  - New stakeholders may emerge and the business environment change.

Cont'd…



The requirements elicitation & analysis Process (The Requirements Spiral)

Elicitation and Analysis Process Activities

1. Requirements discovery

- Interacting with stakeholders to discover their requirements
- 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.

- **Approaches & Techniques of requirements discovery are:**
  - Viewpoints(approach)
  - Interviewing
  - Scenario
  - Use-cases
  - ethnography

Cont'd…

2. Requirements classification and organisation

- Groups related requirements and organises them into coherent clusters.

3. Prioritisation and negotiation

- Prioritising requirements and resolving requirements conflicts.

4. Requirements documentation

- Requirements are documented and input into the next round of the spiral.

C. Software Requirements Specification

- Conversion of requirement into some standard form.
- SRS is a description of a software system to be developed.
- It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.
- It establishes the basis for an agreement between customers and contractors or suppliers (in market-driven project, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do.

Cont'd…

- It permits a careful assessment of requirements before design can begin and reduces later redesign.
  - It should also provide a realistic basis for estimating product costs, risks, and schedules.
  - Used appropriately, software requirements specifications can help prevent software project failure.

Cont'd…

- The software requirements specification document enlists enough and necessary requirements that are required for the project development.
- To derive the requirements, the developer needs to have clear and thorough understanding of the products to be developed or being developed.
  - This is achieved and refined with detailed and continuous communications with the project team and customer till the completion of the software.

Goals of Software Requirements Specification

- The Software Requirements Specification (SRS) is a communication tool between stakeholders and software designers.
- The specific goals of the SRS are:
  - Facilitating reviews
  - Describing the scope of work
  - Providing a reference to software designers (i.e. navigation aids, document structure)
  - Including features to customer requirements
  - Providing a platform for ongoing refinement (via incomplete specs or questions)

Cont'd…

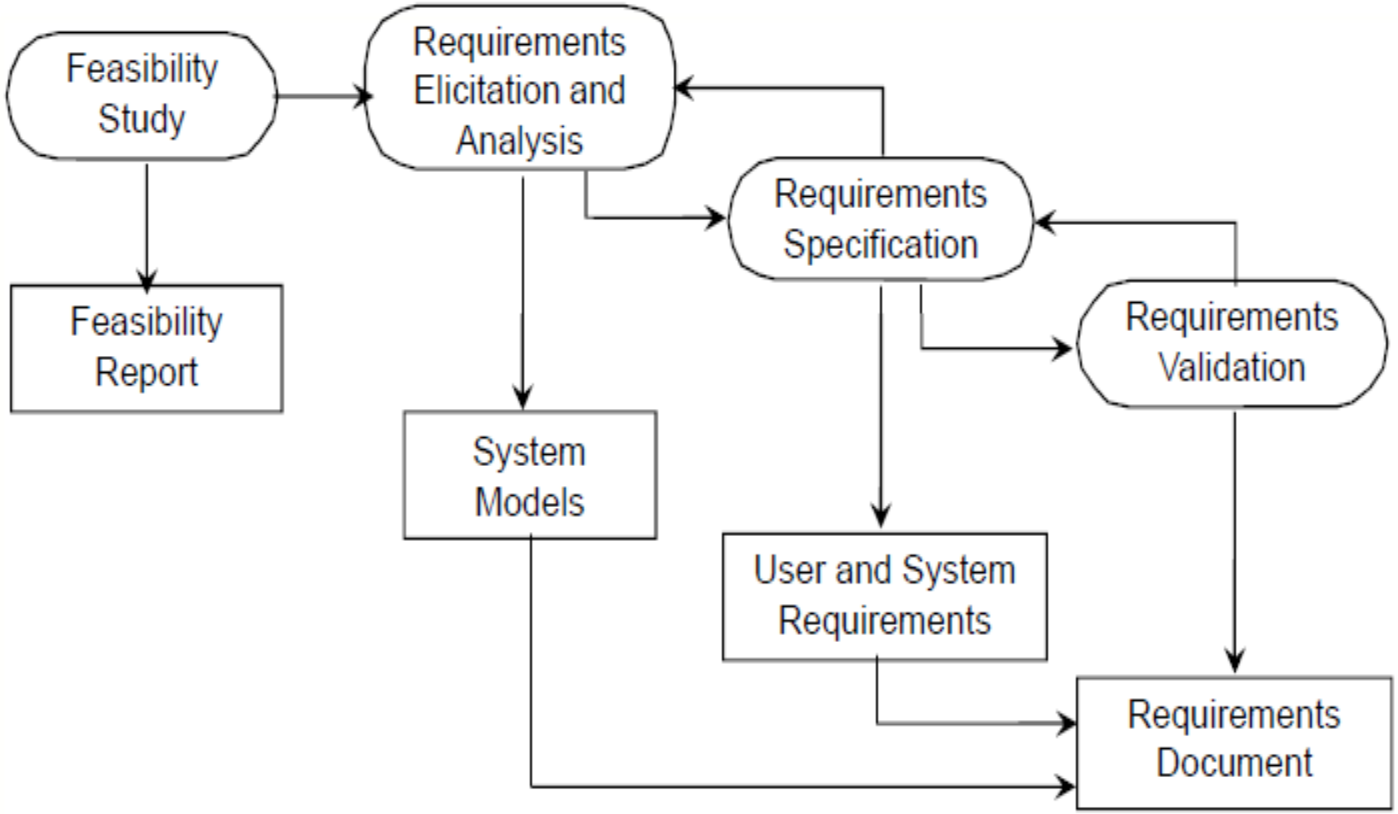


Fig 3: The requirements engineering process

D. 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?

Cont'd…

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.
  - If test is difficult or impossible to design for the requirement means it is difficult to implement that requirement

Cont'd…

❖ 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.

➢ Review Checks

- **Verifiability**: Is the requirement realistically testable?
- **Comprehensibility**: Is the requirement properly understood?
- **Traceability**: Is the origin of the requirement clearly stated?
- **Adaptability**: Can the requirement be changed without a large impact on other requirements?

Requirements Management

- 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 Changes

- The priority of requirements from different viewpoints changes during the development process.
- System customers may specify requirements from a business perspective that conflict with end-user requirements.
- The business and technical environment of the system changes during its development.
- Should apply to all proposed changes to the requirements.

➢ **Principal stages**

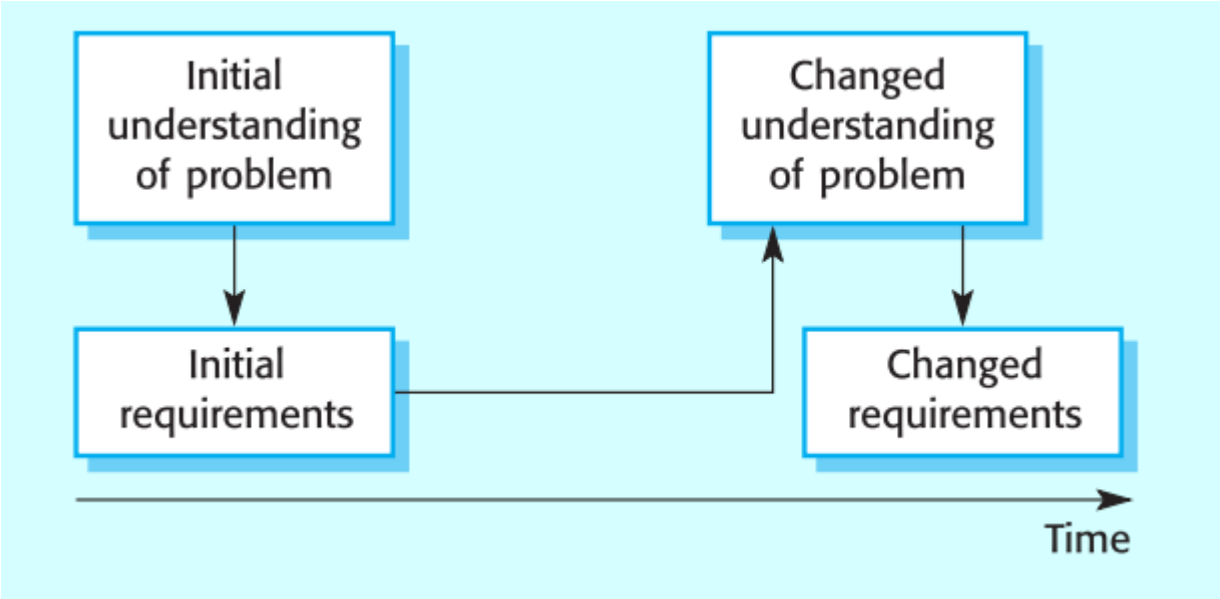
  – **Problem analysis**. Discuss requirements problem and propose change;

  – **Change analysis and costing.** Assess effects of change on other requirements;

  – **Change implementation.** Modify requirements document and other documents to reflect change.

Change Management

Requirements evolution



Enduring and volatile requirements

- **Enduring requirements.** Stable requirements derived from the core activity of the customer organisation.
  - E.g. a hospital will always have doctors, nurses, etc. May be derived from domain models
- **Volatile requirements:** Requirements which change during development or when the system is in use.
  - In a hospital, requirements derived from health-care policy

Requirements Classification

Requirement Type	Description
Mutable requirements	Requirements that change because of changes to the environment in which the organisation is operating. For example, in hospital systems, the funding of patient care may change and thus require different treatment information to be collected.
Emergent requirements	Requirements that emerge as the customer's understanding of the system develops during the system development. The design process may reveal new emergent requirements.
Consequential requirements	Requirements that result from the introduction of the computer system. Introducing the computer system may change the organisations processes and open up new ways of working which generate new system requirements
Compatibility requirements	Requirements that depend on the particular systems or business processes within an organisation. As these change, the compatibility requirements on the commissioned or delivered system may also have to evolve.

Thank You!

?

Debre Markos University, Institute of Technology

School of Computing

Software Engineering Academic Program

Requirements Engineering questions for exit exam preparation

Prepared by: Yayehudar Tamene

1. Requirements might describe:
   A. A user level facility
   B. A very general system property
   C. A specific constraint on a system
   D. A constraint on the development of the system
   ~~E.~~ All
2. Requirements should specify 'what' but not 'how'.
   ~~A.~~ True
   B. False
3. The best way to write requirements is:
   A. Using natural language
   B. Using system models
   C. Using engineering terms
   D. Using diagrams and tables
   ~~E.~~ None
4. Detailed requirements are:
   A. The requirements of stakeholders
   B. The requirements of users
   ~~C.~~ System requirements
   D. A and B
5. One of the following doesn't define Stakeholder requirements?
   A. Sometimes called user requirements.
   B. They are written from the point-of-view of system stakeholders.
   ~~C.~~ These are more detailed specifications of requirements which may be expressed as an abstract model of the system.
   D. None
6. Which of the following activities are categorized under requirements development process of requirements engineering?
   A. Requesting changes to the baselined requirements
   B. Performing impact analysis for the requested changes
   C. Implementing the approved changes.
   D. ~~All of these mentioned above~~
   E. None of the mentioned
7. Which one of the following is a functional requirement?
   A. Maintainability
   B. Portability
   C. Robustness

All are non-functional requirements representing quality of the system. Functional requirements describe what the software has to do.

D. None

8. "Consider a system where, a heat sensor detects an intrusion and alerts the security company." What kind of a requirement the system is providing?
   A. Functional
   B. Non-Functional
   C. Known Requirement
   D. None
9. Which of the following statements explains portability in non-functional requirements?
   A. It is a degree to which software running on one platform can easily be converted to run on another platform
   B. It cannot be enhanced by using languages, OS' and tools that are universally available and standardized
   C. The ability of the system to behave consistently in a user-acceptable manner when operating within the environment for which the system was intended
   D. None
10. Functional requirements capture the intended behavior of the system.
   A. True
   B. False
11. Select the developer-specific requirement?
   A. Portability
   B. Maintainability
   C. Availability
   D. A and B
12. Which one of the following is a requirement that fits in developer's module?
   A. Availability
   B. Testability
   C. Usability
   D. Flexibility

A developer needs to test his product before launching it into the market.

13. Which one of the following is **not** a step of requirement engineering?
   A. Elicitation
   B. Design
   C. Analysis
   D. documentation
14. Conflicting requirements are common in Requirement Engineering, with each client proposing his or her version is the right one.
   A. True
   B. False
15. What is the first step of requirement elicitation?
   A. Identifying Stakeholder
   B. Listing out Requirements
   C. Requirements Gathering
   D. All
16. Arrange the tasks involved in requirements elicitation in an appropriate manner.
   A. Requirements Gathering, Consolidation, Prioritization, Evaluation,
   B. Requirements Gathering, Evaluation, Prioritization, Consolidation
   C. Requirements Gathering, Prioritization, Evaluation, Consolidation
   D. Prioritization, Requirements Gathering, Evaluation, Consolidation

Requirements consolidation is the process of combining all the requirements gathered from various sources into a single document. This document is then used as a reference for the development of the software.

17. Why is Requirements Elicitation a difficult task?
   A. Problem of scope

B. Problem of understanding
C. Problem of volatility
D. All

18. Requirements elicitation is a cyclic process.
A. True
B. False

19. Requirements Analysis is an Iterative Process.
A. True
B. False

20. What does the study of an existing system refer to?
A. Details of DFD
B. Feasibility Study
C. System Analysis
D. System Planning

21. Another name for requirements document is:
A. Functional Specification
B. Requirements Definition
C. Software Requirements Specification
D. All

22. Which of the following property does **not** correspond to a good Software Requirements Specification (SRS)?
A. Verifiable
B. Ambiguous
C. Complete
D. Traceable

23. Which of the following property of SRS is depicted by the statement: "Conformity to a standard is maintained"?
A. Correct
B. Complete
C. Consistent
D. Modifiable

The SRS is complete full labeling and referencing of all figures, tables etc. and definition of all terms and units of measure is defined.

24. The SRS is said to be *consistent* if and only if
A. its structure and style are such that any changes to the requirements can be made easily while retaining the style and structure
B. every requirement stated therein is one that the software shall meet
C. every requirement stated therein is verifiable
D. no subset of individual requirements described in it conflict with each other

25. Which of the following statements about SRS is/are true?
A. SRS is written by customer
B. SRS is written by a developer
C. SRS serves as a contract between customer and developer
D. All
E. None

The SRS acts as a communication media between the Customer, Analyst, system developers, maintainers etc. Thus it is a contract between Purchaser and Supplier. It is essentially written by a developer on the basis of customer' need but in some cases it may be written by a customer as well.

26. The SRS document is also known as _____ specification.
A. black-box
B. white-box
C. grey-box
D. none of the mentioned

SRS documents are usually written by product and project managers or business analysts who directly communicate with clients or who do user research and gather future products' requirements

However, in some cases, software developers are responsible for writing the SRS document for products to be developed

27. Which of the following is included in SRS?
    A. Cost
    B. Design Constraints
    C. Staffing
    D. Delivery Schedule

Which of the following is not included in SRS ?
a) Performance
b) Functionality
c) Design solutions
d) External Interfaces
View Answer
Answer: c
Explanation: The SRS document concentrates on:"what needs to be done" and carefully avoids the solution ("how to do") aspects.

28. Which of the following is **not** included in SRS?
    A. Performance
    B. Functionality
    C. Design solutions
    D. External Interfaces

29. Arrange the given sequence to form an SRS Prototype outline as per IEEE SRS Standard.
    A. Index, General Description, Introduction, Specific Requirements, Appendices
    B. Index, Introduction, General Description, Specific Requirements, Appendices
    C. Introduction, General Description, Specific Requirements, Appendices, Index
    D. Index, General Description, Introduction,

The Index in SRS Prototype outline is a table of contents that lists the major topics and subsections of the document. It is used to help readers quickly locate information in the document

30. Consider the following Statement: "The output of a program shall be given within 10 secs of event X 10% of the time." What characteristic of SRS is being depicted here?
    A. Consistent
    B. Verifiable
    C. Non-verifiable
    D. Correct

The best explanation: An SRS is verifiable, if and only if, every requirement stated therein is verifiable.Here the given condition can be verified during testing phase.Read more on Sarthaks.com -

31. Consider the following Statement: "The data set will contain an end of file character." What characteristic of SRS is being depicted here?
    A. Consistent
    B. Non-verifiable
    C. Correct
    D. Ambiguous

32. Consider the following Statement: "The product should have a good human interface. "What characteristic of SRS is being depicted here?
    A. Consistent
    B. Non-Verifiable
    C. Correct
    D. Ambiguous

To elaborate: An SRS is verifiable, if and only if, every requirement stated therein is verifiable. The statement can only be answered on completion of the software and customer evaluation but still human interface will vary from person to person.

33. Narrative essay is one of the best types of specification document?
    A. True
    B. False

34. The system specification describes the
    A. Function, performance and constraints of a computer-based system
    B. Implementation of each allocated system
    C. Element software architecture
    D. Time required for system simulation

35. Who writes the Software Requirements Specification Document (SRS)?
    A. System Developer
    B. System Tester
    C. System Analyst
    D. None of these above

36. What is the goal of the requirements analysis and specification phase of the software development life cycle?
    A. Understanding the customer requirements and organize them in an informal document
    B. Analyzing the cost of development
    C. Determining scope of the software

The goal of the requirements analysis and specification phase of the software development life cycle is to understand the customer requirements and organize them in a formal document[1].

D. None

37. The best way to conduct a requirements validation review is to
    A. Examine the system model for errors
    B. Have the customer look over the requirements
    C. Send them to the design team and see if they have any concerns
    D. Use a checklist of questions to examine each requirement
38. Which two requirements are given priority during Requirement Management of a product?
    A. User and Developer
    B. Functional and Non-functional
    C. Enduring and Volatile
    D. All
39. Considering the example of issue/return of a book, cataloging etc. in a library management. What type of management requirement is being depicted here?
    A. Enduring
    B. Volatile
    C. Both Enduring & Volatile
    D. All of the mentioned
40. Why is Requirements Management Important? It is due to the changes
    A. to the environment
    B. in technology
    C. in customer's expectations
    D. all
41. Requirements Management is a prerequisite for Quality-Oriented Development.
    A. True
    B. False
42. Requirements traceability is one of the most important part requirement management. It may also be referred to as the heart of requirement management.
    A. True
    B. False
43. Requirements Management has a high initial start-up cost but does not need ongoing funding throughout a project.
    A. True
    B. False
44. According to a statistical report: "over 30% of all software projects are cancelled before completion and over 70% of the remainder fail to deliver expected features". What must be the reason for such a situation?
    A. Poor change management
    B. Poor requirements management
    C. Poor quality control
    D. All of the mentioned
45. Which one of the following is **not** an input to the requirements engineering process?
    A. Domain knowledge
    B. Existing system information
    C. System models
    D. Organizational standards
46. There is no ideal requirements engineering process. This is because of all **Except**:
    A. The technologies and methods used for requirements engineering vary from one organization to another.

B. The types of engineering and managerial disciplines involved in requirements engineering vary from one organization to another.
C. Different types of application system need different types of requirements engineering process.
D. As different organizations have the same culture, so too does the requirements engineering process.

47. Which of the following process models shows the major activities involved in a particular process and their approximate sequencing?
A. Role action models
B. Coarse grain activity models
C. Fine grain activity models
D. Entity relationship models

48. Which of the following is intended to detect problems in the requirements document before it is used as a basis for the system development?
A. Requirements elicitation
B. Requirements Specification
C. Requirements negotiation
D. Requirements validation.

49. Which one of the following is **not** a maturity level according to Capability Maturity Model?
A. Repeatable
B. Measured
C. Optimizing
D. Initial

50. CMM level 5 affirms that the Software Developing Company is at the stage of___.
A. Having several well defined and documented standard processes
B. Having adhoc uncontrolled processes
C. Continuous process improvement
D. Using statistical tools in the management of processes and projects

51. CMM stands for
A. Capability Management Module
B. Conservative Maturity Model
C. Capability Maturity Module
D. Capability Maturity Model

52. One of the following is **wrong** about analysis checklists.
A. The questions should be general.
B. Checklists should not include more than ten items.
C. They provide a reminder of what to look for and reduce the chances that you will forget some requirements checks.
D. None

53. Which of the following differentiates requirements validation from requirements analysis?
A. It is concerned with checking a final draft of a requirements document.
B. The validation process should be more concerned with the way in which the requirements are described.
C. It should mostly be concerned with answering the question 'have we got the requirements right?'
D. All

54. Which of the following is **not** correct about a document-based approach to developing and managing requirements?
A. It's difficult to keep the documents current and synchronized.
B. It's not easy to store supplementary information attributes about each requirement.
C. It's easy to define links between requirements and other system elements.

D. Tracking the status of both individual requirements and the entire set of requirements is cumbersome.

55. Risk management is one of the most important jobs for a
    a) Client
    b) Investor
    c) Production team
    d) Project manager

56. Which of the following term is best defined by the statement: "The underlying technology on which the system is built is superseded by new technology."? superseded=bakka bu'eera
    a) Technology change
    b) Product competition
    c) Requirements change
    d) None of the mentioned

57. Risk management is now recognized as one of the most important project management tasks.
    a) True
    b) False

58. Which of the following strategies means that the impact of the risk will be reduced?
    a) Avoidance strategies
    b) Minimization strategies
    c) Contingency plans
    d) All

59. Which of the following issues are addressed by an effective risk management plan?
    A. Risk monitoring
    B. Risk avoidance
    C. Contingency planning
    D. All

60. Typical requirements risk includes
    A. misunderstanding the requirements
    B. inadequate user involvement
    C. uncertain or changing project scope and objectives
    D. continually changing requirements
    E. all

61. which risk management activity examines the potential consequences of specific risks to your project.
    A. Risk identification
    B. Risk analysis
    C. Risk prioritization
    D. Risk monitoring

Chapter Five

SDLC and Software Development Model Approaches: Use-case-oriented modeling

Introduction

- Software development life cycle (SDLC) is a series of phases that provide a common understanding of the software building process.
- The good software engineer should have enough knowledge on how to choose the SDLC model based on the project context and the business requirements.

➢ Therefore, it may be required to choose the right SDLC model according to the **specific concerns and requirements** of the project **to ensure its success.**

Types of Software developing Model Approaches

- Waterfall model
- V-Model
- Incremental Model
- Agile Model
- Iterative Model and many more.

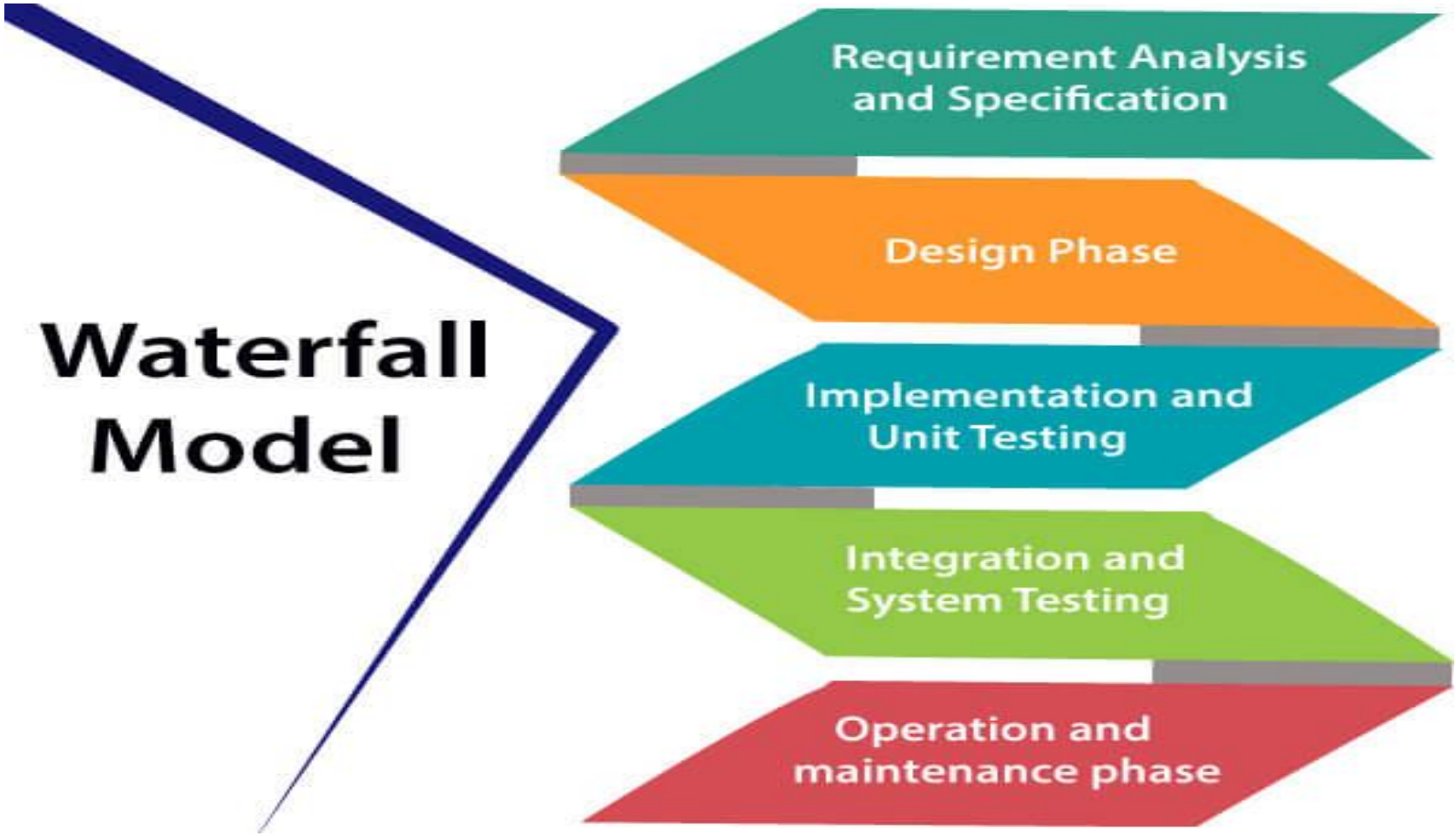
Waterfall model

- This model has five phases:
  - ✓ Requirements analysis and specification
  - ✓ design,
  - ✓ implementation, and unit testing,
  - ✓ integration and system testing, and
  - ✓ operation and maintenance.
- The steps always follow in this order and do not overlap.

➢ The developer must complete every phase before the next phase begins.

- This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

Cont'd…


Waterfall Model
Requirement Analysis and Specification
Design Phase
Implementation and Unit Testing
Integration and System Testing
Operation and maintenance phase

Cont'd…

A. Requirements analysis and specification phase:

- The aim of this phase is to understand the exact requirements of the customer and to document them properly.
- In this phase, a large document called **SRS** document is created which contained a detailed description of what the system will do in the common language.

B. Design Phase:

- This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language.
- It defines the overall software architecture together with high level and detailed design.
- All this work is documented as a **Software Design Document** (SDD).

Cont'd…

C. Implementation and unit testing:

- During this phase, design is implemented.
- If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.
- During testing, the code is thoroughly examined and modified.
- Small modules are tested in isolation initially.
- After that these modules are tested by writing some code to check the interaction between these modules and the flow of intermediate output.

Cont'd…

D. Integration and System Testing:

- This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out.
- The better output will lead to satisfied customers, lower maintenance costs, and accurate results.

➢ In this phase, the modules are tested for their **interactions** with each other and with the system.

E. Operation and maintenance phase

- Maintenance is the task performed once the software has been delivered to the customer, installed, and operational.

Cont'd…

When to use SDLC Waterfall Model?

- Some Circumstances where the use of the Waterfall model is most suited are:
  - When the requirements are constant and not changed regularly.
  - A project is short
  - The situation is calm
  - Where the tools and technology used is consistent and is not changing
  - When resources are well prepared and are available to use.

Cont'd…

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Cont'd…

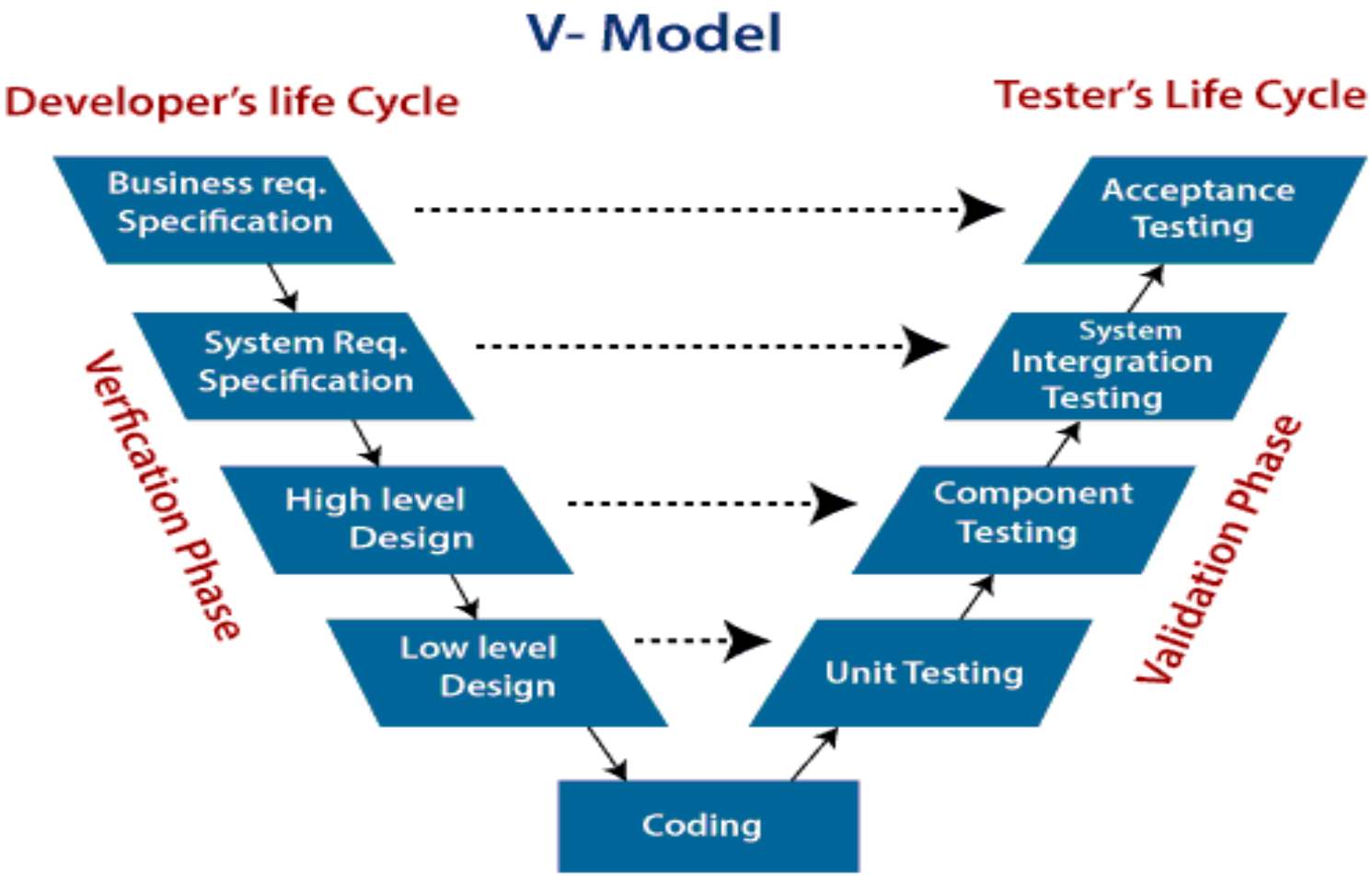
Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase.
  - For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

V-Model

- It is also referred to as the Verification and Validation Model.
- Each phase of SDLC must complete before the next phase starts.
- It follows a sequential design process same as the waterfall model.
- Testing is planned in parallel with a corresponding stage of development.

Cont'd…


V- Model
Developer's life Cycle
Tester's Life Cycle
Business req. Specification
System Req. Specification
High level Design
Low level Design
Coding
Unit Testing
Component Testing
System Intergration Testing
Acceptance Testing
Verfication Phase
Validation Phase

Cont'd…

Verification:

- It involves a static analysis method (review) done without executing code.
- It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation:

- It involves dynamic analysis method (functional, non-functional), testing is done by executing code.
- Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.
- Verification and Validation process is joined by coding phase in V-shape.
- Thus it is known as V-Model.

Cont'd…

- **There are the various phases of Verification Phase of V-model:**

A. Business requirement analysis:

- This is the first step where product requirements understood from the customer's side.
- It contains detailed communication to understand customer's expectations and exact requirements.

B. System Design:

- In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.

C. Architecture Design:

- The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc.
- The integration testing model is carried out in a particular phase.

Cont'd…

D. Module Design:

- the system breaks down into small modules.
- The detailed design of the modules is specified, which is known as Low-Level Design

E. Coding Phase:

- After designing, the coding phase is started.
- Based on the requirements, a suitable programming language is decided.
- There are some guidelines and standards for coding.
- Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

Cont'd…

- **There are the various phases of Validation Phase of V-model:**

A. Unit Testing:

- Unit Test Plans (UTPs) are developed during the module design phase.
- These UTPs are executed to eliminate errors at code level or unit level.
- A unit is the smallest entity which can independently exist, e.g., a program module.
- Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.

B. Integration Testing:

- Integration Test Plans are developed during the Architectural Design Phase.
- These tests verify that groups created and tested independently can coexist and communicate among themselves.

Cont'd…

C. System Testing:

- System Tests Plans are developed during System Design Phase.
- Unlike Unit and Integration Test Plans, System Tests Plans are composed by the clients business team.
- System Test ensures that expectations from an application developer are met.

D. Acceptance Testing:

- Acceptance testing is related to the business requirement analysis part.
- It includes testing the software product in user atmosphere.
- Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere.

Cont'd…

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Advantage of V-Model:

- Easy to Understand.
- Testing Methods like planning, test designing happens well before coding.
- This saves a lot of time. Hence a higher chance of success over the waterfall model.
- Avoids the downward flow of the defects.
- Works well for small plans where requirements are easily understood.

Cont'd…

Disadvantage of V-Model:

- Very rigid and least flexible.
- Not a good for a complex project.
- Software is developed during the implementation stage, so no early prototypes of the software are produced.
- If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Incremental Model

- It is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
- Each module goes through the requirements, design, implementation and testing phases.
- Every subsequent release of the module adds function to the previous release.
- The process continues until the complete system achieved.

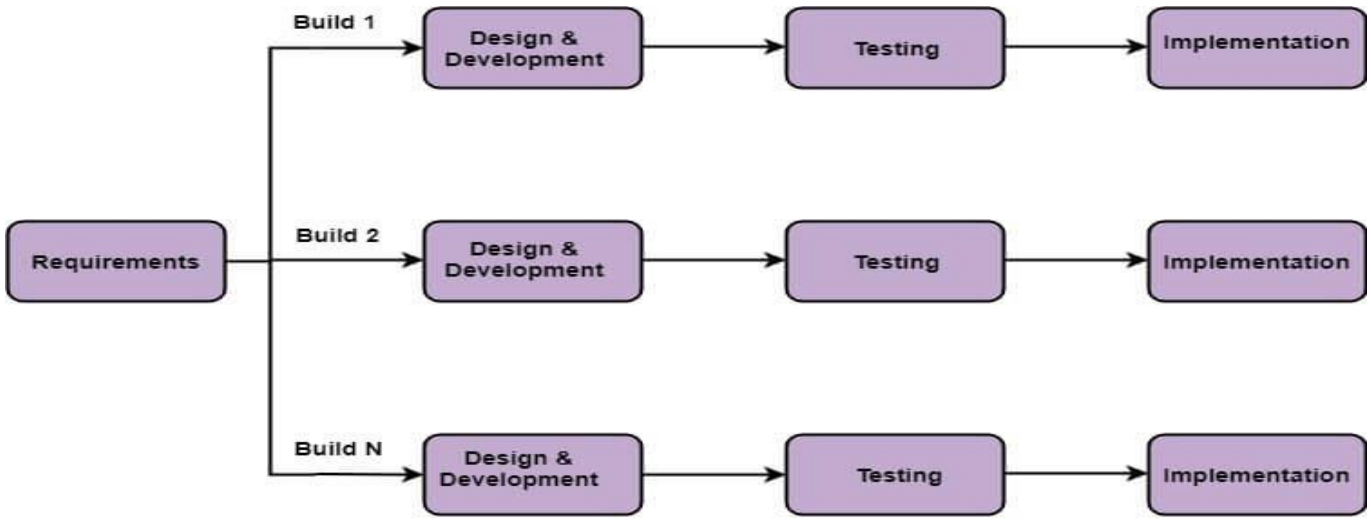


Fig: Incremental Model

Cont'd…

The various phases of incremental model are as follows:

1. Requirement analysis:

- In the first phase of the incremental model, the product analysis expertise identifies the requirements.
- And the system functional requirements are understood by the requirement analysis team.
- To develop the software under the incremental model, this phase performs a crucial role.

2. Design & Development:

- In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success.

Cont'd…

3. Testing:

- the testing phase checks the performance of each existing function as well as additional functionality.
- the various methods are used to test the behavior of each task.

4. Implementation:

- Implementation phase enables the coding phase of the development system.
- It involves the final coding.
- After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Cont'd…

When we use the Incremental Model?

- When the requirements are more.
- A project has a lengthy development schedule.
- When Software team are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

Cont'd…

Disadvantage of Incremental Model

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

Agile Model

- The meaning of Agile is swift or versatile.“
- **Agile process model**" refers to a software development approach based on iterative development.
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- The project scope and requirements are laid down at the beginning of the development process.
- Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Cont'd…

- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks.
- The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements.
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

Cont'd…

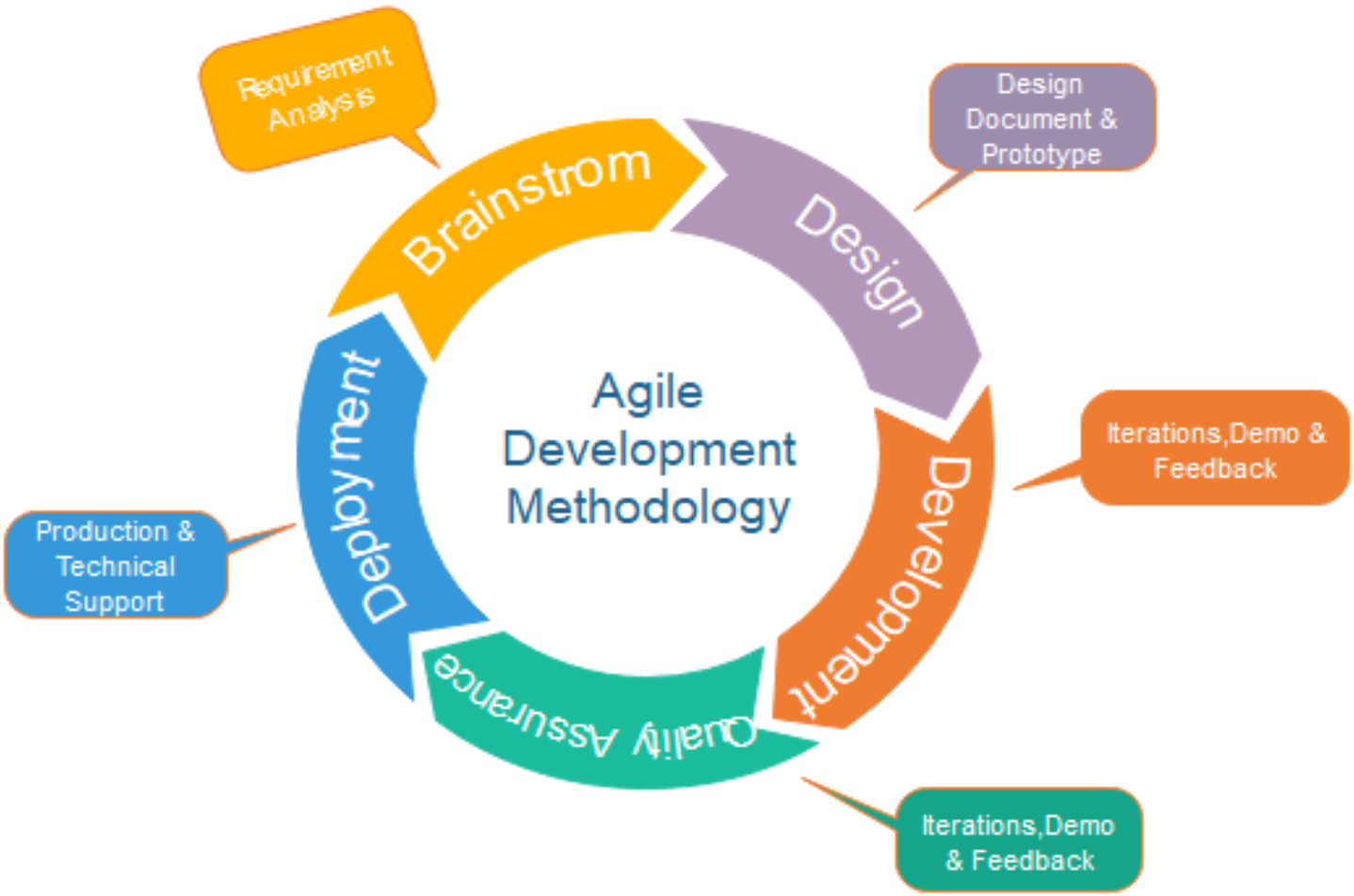


Fig. Agile Model

Cont'd…

Phases of Agile Model

1. **Requirements gathering:**

- In this phase, you must define the requirements.
- You should explain business opportunities and plan the time and effort needed to build the project.
- Based on this information, you can evaluate technical and economic feasibility.

2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements.

- You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

Cont'd…

3. Construction/ iteration: When the team defines the requirements, the work begins.

- Designers and developers start working on their project, which aims to deploy a working product.
- The product will undergo various stages of improvement, so it includes simple, minimal functionality.

4. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.

5. Deployment: In this phase, the team issues a product for the user's work environment.

6. Feedback: After releasing the product, the last step is feedback.

- In this, the team receives feedback about the product and works through the feedback.

Cont'd…

When to use the Agile Model?

- When frequent changes are required.
- When a highly qualified and experienced team is available.
- When a customer is ready to have a meeting with a software team all the time.
- When project size is small.

Advantage(Pros) of Agile Method:

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Anytime changes are acceptable.
- It reduces total development time.

Cont'd…

Disadvantages(Cons) of Agile Model:

- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers allotted to another project, **maintenance of the finished project can become a difficulty.**

Iterative Model

- In this Model, you can start with some of the software specifications and develop the first version of the software.
- After the first version if there is a need to change the software, then a new version of the software is created with a new iteration.
- Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.
- ➢ The Iterative Model allows the accessing of earlier phases, in which the variations made respectively.
- The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.

Cont'd…

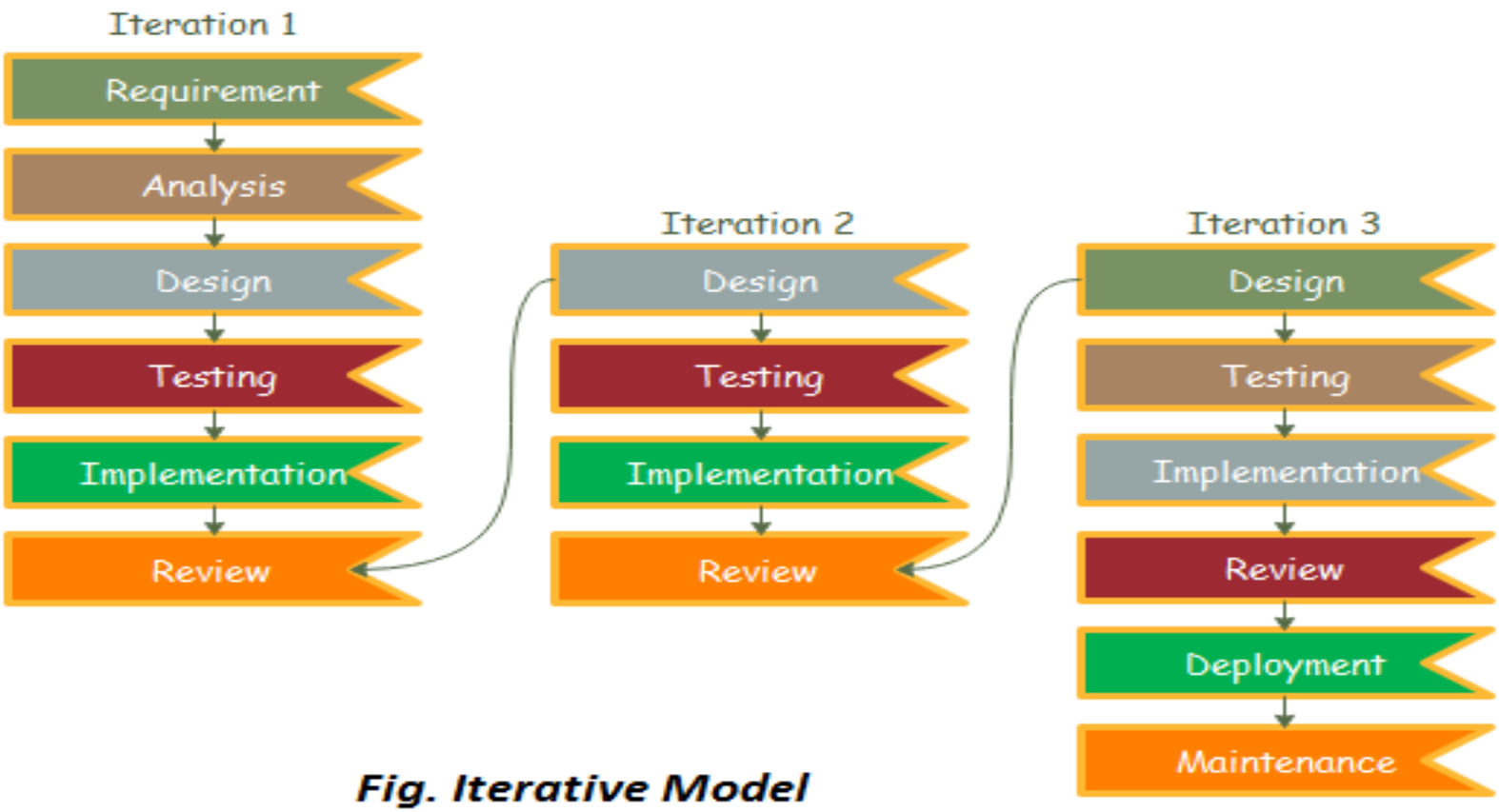


Fig. Iterative Model

Cont'd…

The various phases of Iterative model are as follows:

1. Requirement gathering & analysis:

- In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not.
- Analyst checks that need will achieve within budget or not.
- After all of this, the software team skips to the next phase.

2. Design:

- In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

3. Implementation:

- In the implementation, requirements are written in the coding language and transformed into computer programmes which are called Software.

Cont'd…

4. **Testing:** After completing the coding phase, software testing starts using different test methods.
5. **Deployment:** After completing all the phases, software is deployed to its work environment.
6. **Review:** In this phase, after the product deployment, review phase is performed to check the behaviour and validity of the developed product.

- And if there are any error found then the process starts again from the requirement gathering.

7. **Maintenance:** after deployment of the software in the working environment, there may be some bugs, some errors or new updates are required.

- Maintenance involves **debugging** and **new addition options**.

Cont'd…

When to use the Iterative Model?

- When requirements are defined clearly and easy to understand.
- When the software application is large.
- When there is a requirement of changes in future.

Advantage of Iterative Model:

- Testing and debugging during smaller iteration is easy.
- A Parallel development.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.
- Limited time spent on documentation and extra time on designing.

Cont'd…

Disadvantage of Iterative Model:

- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

Reading Assignment

- ✓ **RAD Model**
- ✓ **Big Bang Model**
- ✓ **Prototype Model**
- ✓ **Evolutionary Process Model**
- ➢ **Agile Testing Methods**

Thank You!

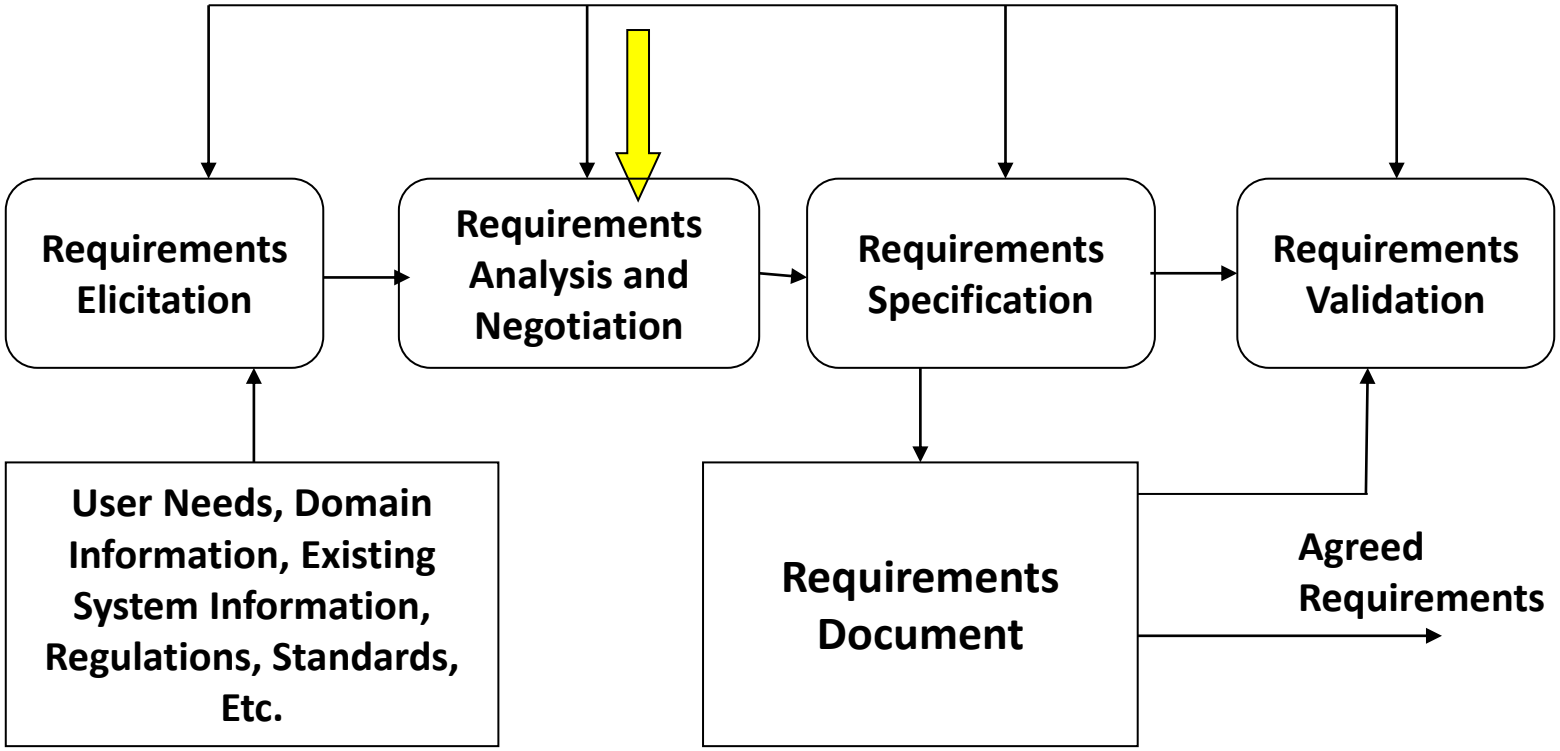
?

Chapter Four

Requirements Analysis and Negotiation

Recap of Requirements Elicitation

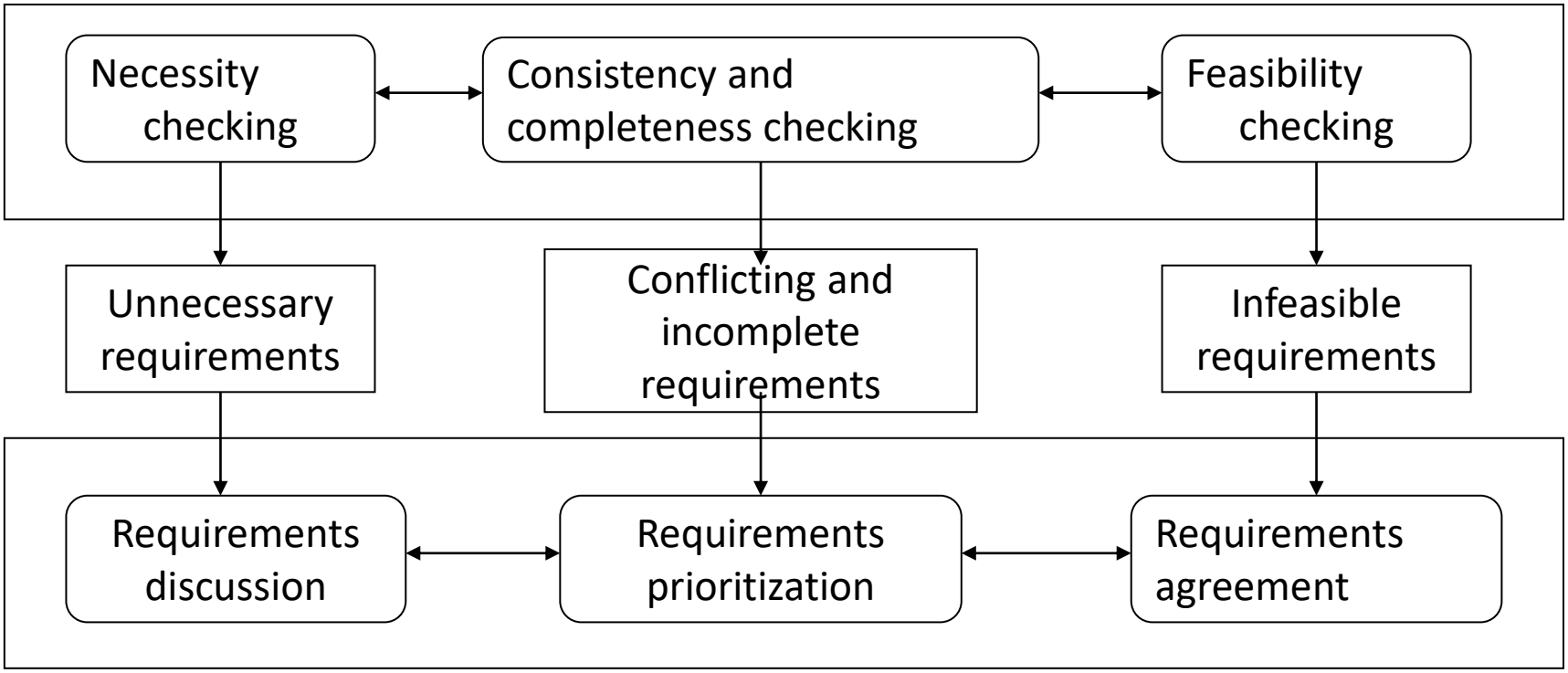
- Requirements elicitation deals with discovering requirements for a software product.
- It is an iterative process.



Requirements Analysis and Negotiation

- Requirements analysis and negotiation are inter-leaved activities and join to form a major activity of the requirements engineering process.

Requirements Analysis

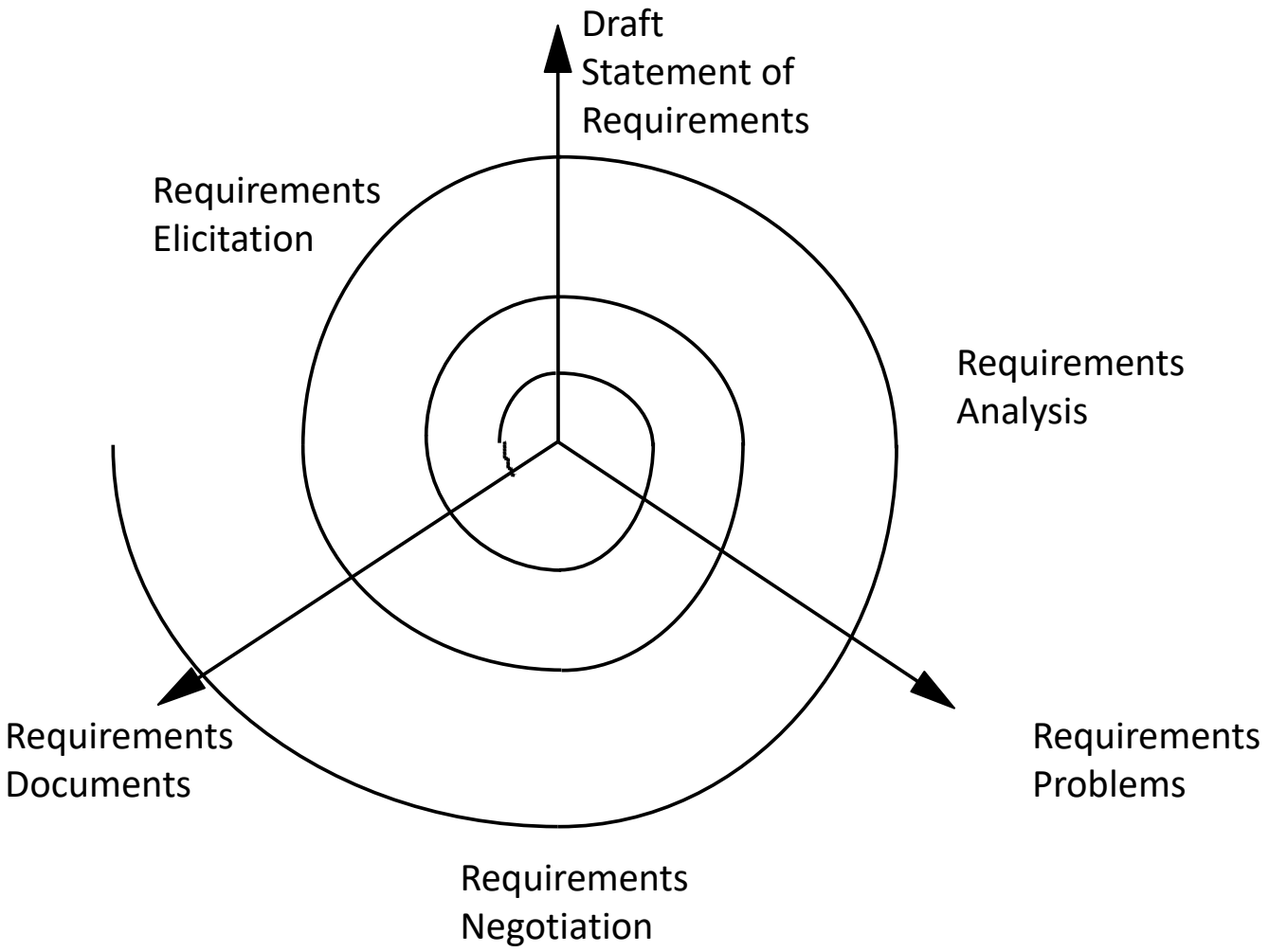


Requirements Negotiation

Requirements Analysis

- The **aim** of requirements analysis is to discover problems with the system requirements, especially incompleteness and inconsistencies.
- Some analysis is inter-leaved with requirements elicitation as problems are sometimes obvious as soon as a requirement is expressed.
- Detailed analysis usually takes place after the initial draft of the requirements document is produced.
- Analysis is concerned with incomplete set of requirements, which has not been discussed by stakeholders.

Iterative Aspects of Elicitation, Analysis, and Negotiation



Comments on Requirements Analysis

- **Analysts** read the requirements, highlight problems, and discuss them in requirements review meetings.
  - This is a time-consuming and expensive activity.
- Analysts have to think about implications of the draft statements of requirements.
- ❖ People do not think in the same way and different analysts tackle the process in different ways.
- It is not possible to make this activity a structured and systematic process.
  - It depends on the judgment and experience of process participants.

Requirements Analysis Stages

- ❖Necessity checking
- ❖Consistency and completeness checking
- ❖Feasibility checking

Necessity Checking

- The need for the requirement is analyzed.
- In some cases, requirements may be proposed which don't contribute to the business goals of the organization or to the specific problem to be addressed by the system.

Cont'd…

Consistency and Completeness Checking

- The requirements are cross-checked for consistency and completeness.

➢ **Consistency** means that no requirements should be **contradictory**

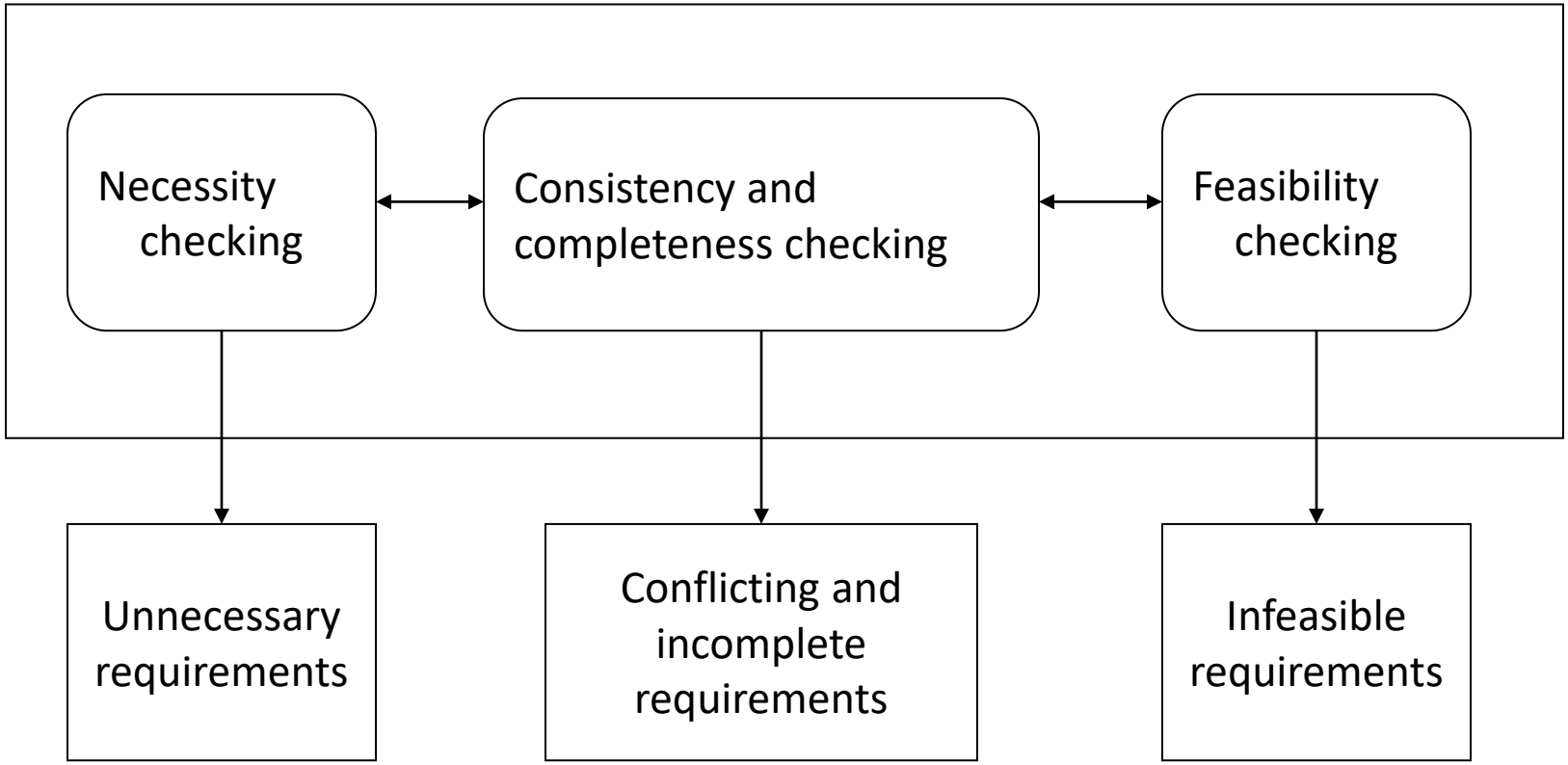
➢ **Completeness** means that **no services** or constraints which are needed have been **missed out.**

Feasibility Checking

- The requirements are checked to ensure that they are feasible in the context of the budget and schedule available for the system development.

Requirements Analysis Process

Requirements Analysis



Analysis Techniques

- **Analysis checklists**
  - A checklist is a list of questions which analysts may use to assess each requirement.
- **Interaction matrices**
  - Interaction matrices are used to discover interactions between requirements and to highlight **conflicts** and **overlaps**.

Cont'd…

Analysis Checklists

- Each requirement may be assessed against the checklist
- When potential problems are discovered, these should be noted carefully
- They can be implemented as a spreadsheet, where the **rows** are labeled with the **requirements identifiers** and **columns** are the **checklist items**
- They are useful as they provide a reminder of what to look for and reduce the chances that you will forget some requirements checks.

Cont'd…

- They must evolve with the experience of the requirements analysis process.
- The questions should be general, rather than restrictive, which can be irrelevant for most systems.
- Checklists should not include more than ten items, because people forget items on long checklists reading through a document.

❖ The following list shows a possible requirement analysis checklist:

Checklist Items and their Description

- **Premature design**
  - Does the requirement include premature design or implementation information?
- **Combined requirements**
  - Does the description of a requirement describe a single requirement or could it be broken down into several different requirements?
- **Unnecessary requirements**
  - Is the requirement ‘gold plating’? That is, is the requirement a cosmetic addition to the system which is not really necessary?

Cont'd…

- **Use of non-standard hardware**
  - Does the requirement mean that non-standard hardware or software must be used?
    - To make this decision, you need to know the computer platform requirements.
- **Conformance with business goals**
  - Is the requirement consistent with the business goals defined in the introduction to the requirements document?

Cont'd…

- **Requirements ambiguity**
  - Is the requirement ambiguous i.e., could it be read in different ways by different people? What are the possible interpretations of the requirement?
- **Requirements realism**
  - Is the requirement realistic given the technology which will be used to implement the system?
- **Requirements testability**
  - Is the requirement testable, that is, is it stated in such a way that test engineers can derive a test which can show if the system meets that requirement?

Requirements Interactions

- A very important objective of requirements analysis is to discover the interactions between requirements and to highlight requirements **conflicts** and **overlaps**.
- A **requirements interaction matrix** shows how requirements interact with each other, which can be constructed using a spreadsheet.
- Each requirement is compared with other requirements, and the matrix is filled as follows:
  - For requirements which conflict, fill in a 1
  - For requirements which overlap, fill in a 1000
  - For requirements which are independent, fill in a 0
- Consider the following example

An Interaction Matrix

Requirement	**R1**	**R2**	**R3**	**R4**	**R5**	**R6**
R1	0	0	1000	0	1	1
R2	0	0	0	0	0	0
R3	1000	0	0	1000	0	1000
R4	0	0	1000	0	1	1
R5	1	0	0	1	0	0
R6	1	0	1000	1	0	0

Comments on Interaction Matrices

- If you can't decide whether requirements conflict, you should assume that a conflict exists.
- If an error is made it is usually fairly cheap to fix; it can be much more expensive to resolve undetected conflicts.
- ➢ In the example, we are considering, we can see that R1 overlaps with R3 and conflicts with R5 and R6.
- R2 is an independent requirement.
- R3 overlaps with R1, R4, and R6.
- ➢ The advantage of using numeric values for conflicts and overlaps is that you can sum each row and column to find the number of conflicts and the number of overlaps.
- ❖ Requirements which have high values for one or both of these figures should be carefully examined.

Cont'd…

- A large number of conflicts or overlaps means that **any changes to that requirement will probably have a major impact on the rest of the requirements**
- Interaction matrices work only when there is relatively small number of requirements, as each requirement is compared with every other requirement
- The upper limit should be about **200** requirements.
- These overlaps and conflicts have to be discussed and resolved during **requirements negotiation,** which we'll discuss next.

Requirements Negotiation

- Disagreements about requirements are inevitable when a system has many stakeholders.
- ➢ Conflicts are not 'failures' but reflect different stakeholder needs and priorities.
- **Requirements negotiation** is the process of discussing requirements conflicts and reaching a compromise that all stakeholders can agree to.
- In planning a requirements engineering process, it is important to leave enough time for negotiation.
  - Finding an acceptable compromise can be time-consuming.
- ➢ The final requirements will always be a compromise which is governed by the needs of the organization in general, the specific requirements of different stakeholders, design and implementation constraints, and the budget and schedule for the system development.

Requirements Negotiation Stages

1. Requirements Discussion

- Requirements which have been highlighted as problematic are discussed and the stakeholders involved present their views about the requirements.

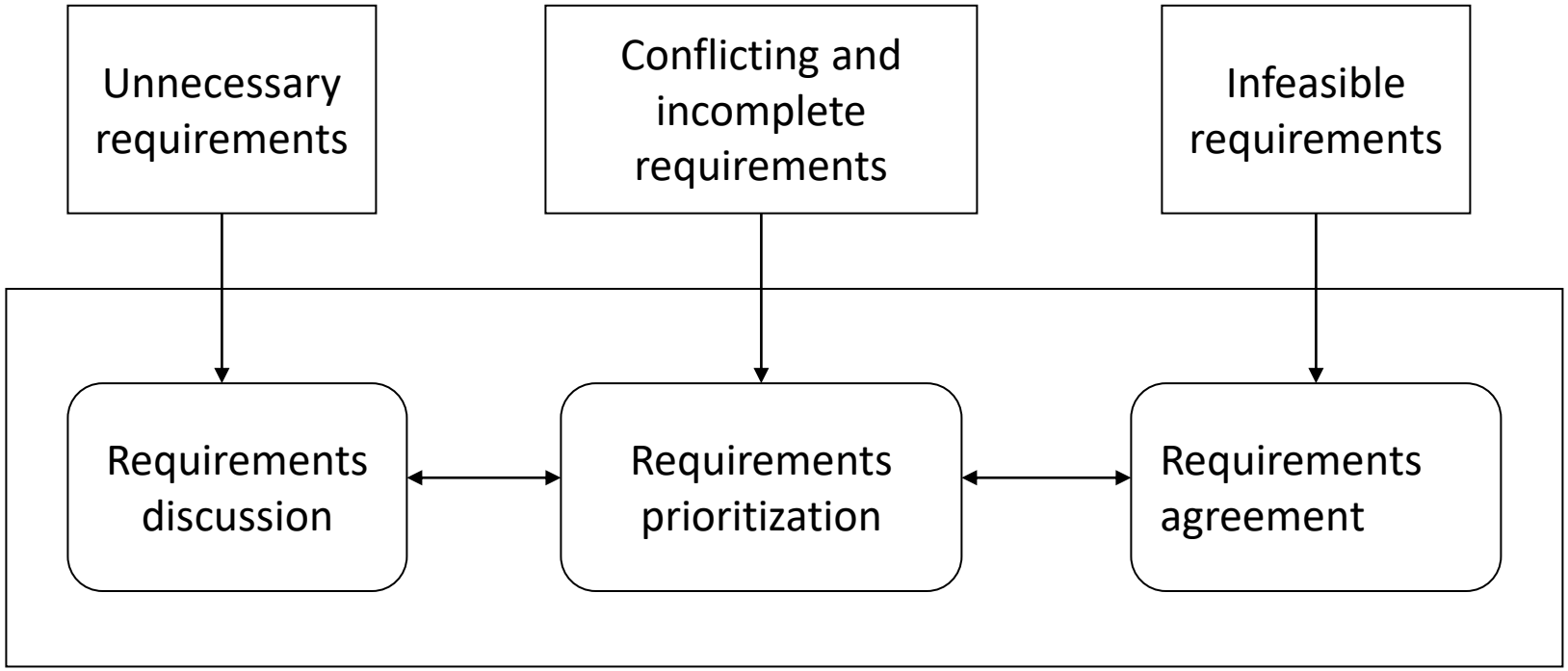
2. Requirements Prioritization

- Disputed/unclear requirements are prioritized to identify critical requirements and to help the decision making process.

3. Requirements Agreement

- Solutions to the requirements problems are identified and a compromised set of requirements are reached.
- Generally, this will involve making changes to some of the requirements.

Requirements Negotiation Process



Requirements Negotiation

Comments on Requirements Negotiation

- In principle, requirements negotiation should be an objective process.
- The judgments should be the requirements for the system and should be based on technical and organizational needs.
- Reality is, however, often different.
- Negotiations are rarely conducted using only logical and technical arguments.
- They are influenced by organizational and political considerations, and the personalities of the people involved.
- A strong personality may force their priorities on other stakeholders.
- Requirements may be accepted or rejected because they strengthen the political influence in the organization of some stakeholders.
- End-users may be resistant to change and may block requirements, etc.

Cont'd…

- The majority of time in requirements negotiation is usually spent resolving requirements conflicts.
- A requirement conflicts with other requirements if they ask for different things.
- Conflicts should not be viewed as 'failures', they are natural and inevitable – rather healthy.
- **Meetings are the most effective way to negotiate requirements and resolve requirements conflicts.**
- All requirements which are in conflict should be discussed individually.
- Negotiation meetings should be conducted in three stages.

Stages of Negotiation Meetings

1. **Information Stage**

- The nature of the problems associated with a requirement is explained.

2. **Discussion Stage**

- A discussion stage where the stakeholders involved discuss how these problems might be resolved
  - All stakeholders with an interest in the requirement should be given the opportunity to comment.
  - Priorities may be assigned to requirements at this stage.

3. **Resolution Stage**

- A resolution stage where actions concerning the requirement are agreed
  - These actions might be to delete the requirement, to suggest specific modifications to the requirement or to elicit further information about the requirement.

Requirements Error/Defect

- A deficiency in the requirements quality that can hamper software development.
- ❖ Errors and omissions find their way in different requirements documents.
- ➢ If not removed, requirements errors usually flow downstream into design, code, and user manuals.
- It is difficult to detect requirements errors once they flow downstream.
- Requirements errors are most expensive to eliminate.

Types of Requirements Errors

- Errors of omission
- Errors of commission
- Errors of clarity and ambiguity
- Errors of speed and capacity

Cont'd…

Errors of Omission

- Errors of omission are most common among requirements errors.
- Domain experts easily forget to convey domain knowledge to requirements engineers, because they consider that to be obvious and implicit.

Errors of Clarity and Ambiguity

- Second most common errors are those of clarity and ambiguity
- Primarily, because natural languages (like English) are used to state requirements, while such languages are themselves ambiguous
- For example: object

Cont'd…

Errors of Commission

- Errors of commission can also find their way into the requirements documents.

Performance Errors

- Performance, that is errors of speed and capacity, are also found in requirements.
- Primarily, these occur due to conflicting understanding or competing needs of different stakeholders.

Negative Impact of Requirements Errors

- ✓ The resulting software may not satisfy user's real needs.
- ✓ Multiple interpretations of requirements may cause disagreements between customers and developers, wasting time and money, and perhaps resulting in lawsuits.
- ✓ Negative impact on humans
  - – Unsatisfied customers and developers
  - – Lack of interest in automation of processes
  - – Blame game

Addressing Requirements Errors

- Prevention
- Removal

Prevention vs. Removal

- For requirements errors, **prevention is usually more effective than removal**
  - ✓ Joint application development (**JAD**),
  - ✓ quality function deployment (**QFD**), and
  - ✓ **prototyping** are more effective in defect prevention

- ➢ **JAD** is a methodology that involves the client or end user in the design and development of an application, through a succession of collaborative workshops called JAD sessions.
- ➢ **QFD** is a method developed in Japan in 1966 to help transform the voice of the customer into engineering characteristics for a product.
  - – For carefully listening to the voice of all the customer and then effectively responding to those needs and expectations.
- ➢ Requirements **inspections** play an important role in defect removal

Defect Prevention

- Don't let defects/errors become part of the requirements document or requirements model in the first place

➢ How is it possible?

  – Understanding application domain and business area is the first step in defect prevention

  – Training in different requirements engineering activities (elicitation, analysis and negotiation, specification, and validation) is also very important for defect prevention.

  – Allocating enough time to conduct requirements engineering activities also is very important in this regard.

Cont'd…

- **Willing and active participation of stakeholders** in different activities of requirements engineering.
  - That is why **JAD** is very useful in defect prevention as far as requirements errors are concerned.
- An overall commitment to quality and emphasis on using documented processes is also a very important.
- An overall commitment to process improvement.

Defect Removal

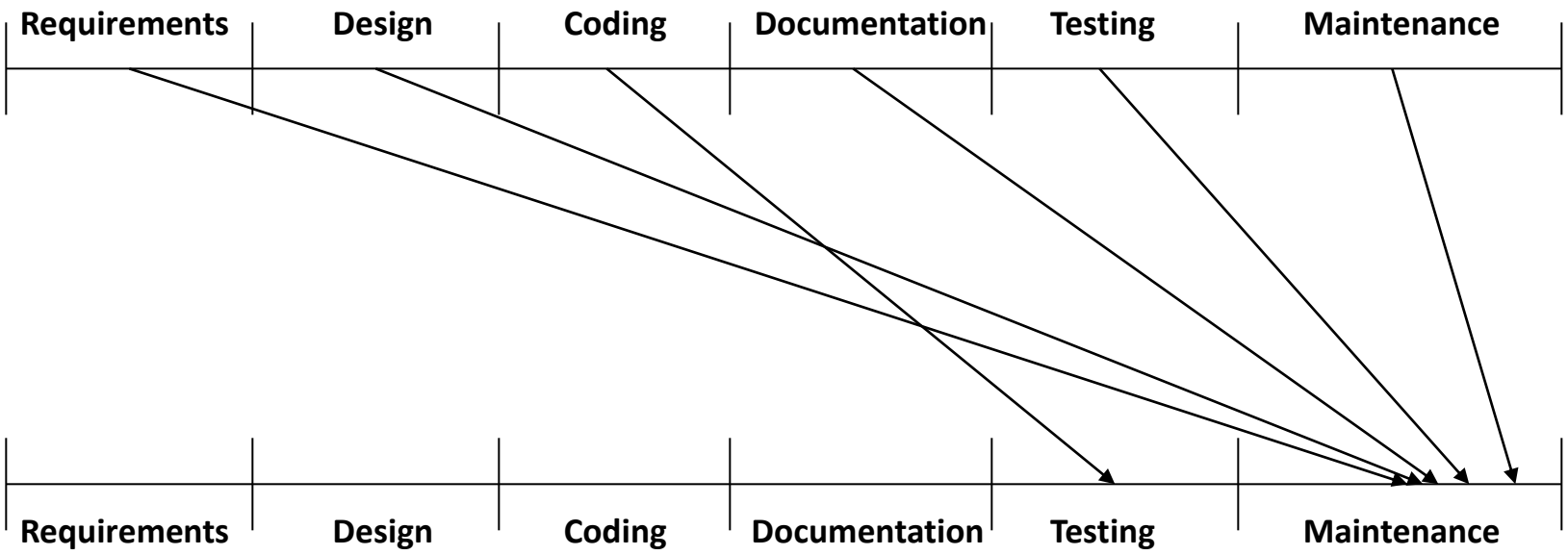
Inspections

- Inspections are conducted by a group of people working on the project, with the objective to remove defects or errors
- Every member of the inspection team has to read and evaluate requirements documents before coming to the meeting and a formal meeting is conducted to discuss requirements errors.

➢ Requirements errors detected during this inspections save lot of money and time as requirements errors do not flow into the design and development phases of software development process.

Defect Detection Without Inspections

Defect Origins

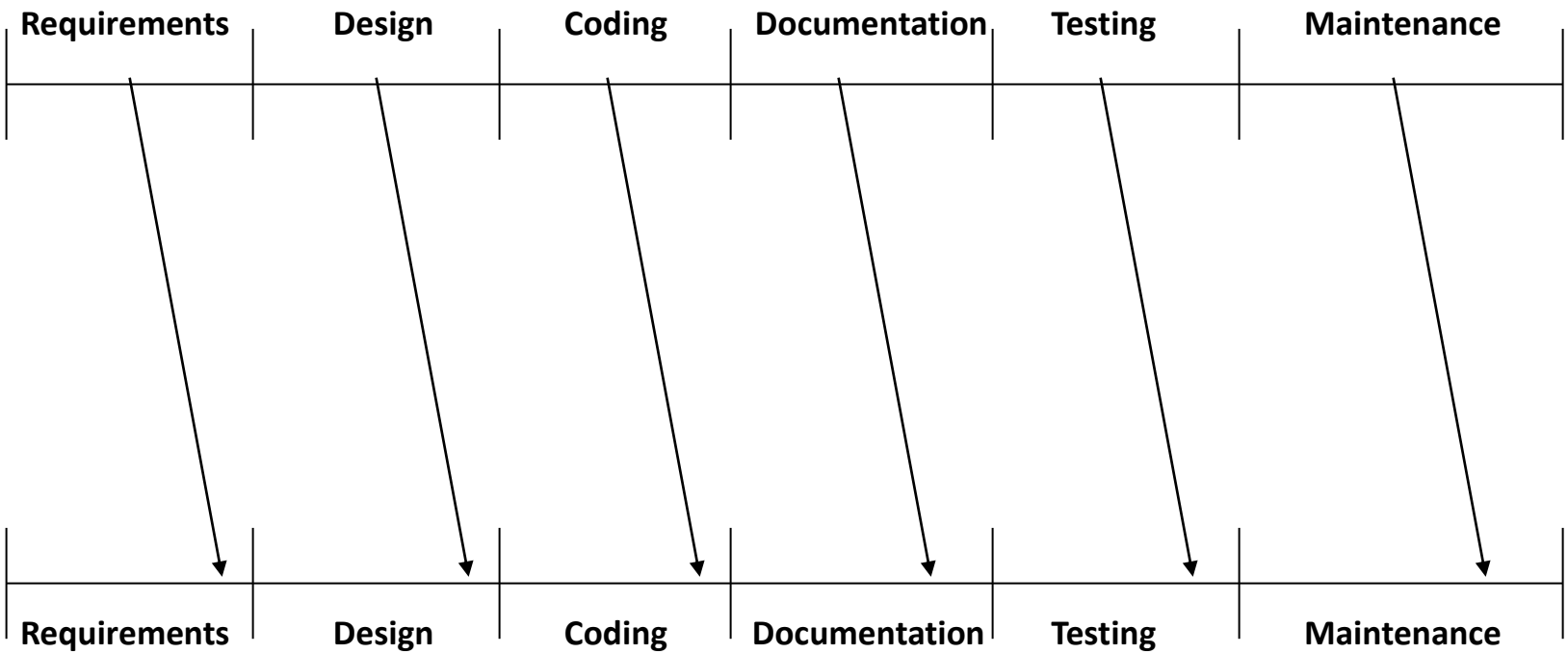


Defect Discovery

Chaos Zone

Defect Detection With Inspections

Defect Origins



Defect Discovery

According to various sources, there are different types of requirements errors that can affect the quality, feasibility, and success of software projects. Some of the most common types are:

- **Ambiguous requirements**: These are requirements that are unclear, vague, or open to multiple interpretations. They can lead to confusion, misunderstandings, and conflicts among the stakeholders, developers, and testers. They can also result in incorrect or incomplete implementations, testing failures, and customer dissatisfaction[2][3].
- **Incomplete requirements**: These are requirements that are missing, insufficient, or inconsistent. They can result in gaps, errors, or contradictions in the software design, development, and testing. They can also cause delays, rework, and changes in the project scope, schedule, and budget[2][3].
- **Unrealistic requirements**: These are requirements that are impractical, unfeasible, or impossible to implement or test. They can result in wasted time, resources, and efforts, as well as frustration and disappointment among the stakeholders, developers, and testers. They can also compromise the quality, performance, and security of the software product[2][3].
- **Volatile requirements**: These are requirements that change frequently or unpredictably during the software development process. They can result from changing business needs, market conditions, customer preferences, or regulatory requirements[3]. They can increase the complexity, uncertainty, and risk of the project[2].
- **Errors of conceptualization**: These are errors that arise because of poorly conceived problems or solutions. They can result from inadequate or incorrect understanding of the customer's needs, goals, context, or constraints[1][4].
- **Errors of specification**: These are errors that arise because of poor specification techniques or tools. They can result from using ambiguous or inconsistent language, omitting important details or assumptions, or failing to capture the functional and non-functional requirements[1][4].
- **Errors of implementation**: These are errors that arise because of poor coding and testing practices. They can result from violating the requirements specifications, introducing bugs or defects, or failing to verify or validate the software product[1][4].

Thank You!

?

Chapter Seven

Tools for Requirements Engineering

Introduction

- Although still widely used, a document-based approach to developing and managing requirements has numerous limitations, including the following:
  - ✓ It's difficult to keep the documents current and synchronized.
  - ✓ Communicating changes to all affected team members is a manual process.
  - ✓ It's not easy to store supplementary information attributes about each requirement.
  - ✓ It's hard to define links between requirements and other system elements.
  - ✓ Tracking the status of both individual requirements and the entire set of requirements is cumbersome.

Cont'd…

- ✓ Reusing a requirement generally means that the business analyst must copy the text from the original document into another document for each other system or product where the requirement is to be used.
- ✓ It's difficult for multiple project participants to modify the requirements, particularly if the participants are geographically separated.
- ✓ There's no convenient place to store proposed requirements that were considered but rejected and requirements that were deleted from a baseline.
- ✓ Identifying missing, duplicate, and unnecessary requirements is difficult.

Cont'd…

- Requirements development (**RD**) tools and requirements management (**RM**) tools **provide solutions** to all of these limitations.
  - ✓ RD tools can help you elicit the right requirements for your project and judge whether those requirements are well-written.
  - ✓ RM tools help you manage changes to those requirements, track status, and trace requirements to other project deliverables.
  - ✓ A team working on a small project might be able to get away without using any requirements tools, instead using documents, spreadsheets, or simple databases to manage their requirements.
  - ✓ Teams working on large projects will benefit from commercial requirements engineering tools.

Cont'd…

- Dozens of commercial requirements tools are available.
- RD and RM tools often aren't cheap, but the high cost of requirements-related problems can justify your investment in them.
- Recognize that the cost of a tool is not simply what you pay for the initial license.
  - The cost also includes annual maintenance fees and periodic upgrades, software installation and configuration, administration, vendor support and consulting, and training for users.
  - Cloud-based solutions eliminate some of these additional support activities and costs.
- Your cost-benefit analysis should take into account all of the expenses before you make a purchase decision.

Requirements Development (RD) Tools

- RD tools are used by business analysts to work with stakeholders to elicit and document requirements more effectively and more efficiently than with manual methods.
- RD tools can improve stakeholder collaboration by accommodating a variety of communication methods (Frye 2009).

❑ Development tools can be divided into:

✓ Elicitation tools, prototyping tools and modeling tools.

- Some of the tools in the RD category provide all of these services.
- Some of them also offer requirements management capabilities.

❖ In general, RD tools are not as mature as RM tools, and their overall impact on projects is typically less than that of RM tools.

Elicitation Tools

- Elicitation tools include those used for recording notes during elicitation sessions.
- These enable the BA to quickly organize ideas.

✓Mind-mapping tools facilitate brainstorming as well as organizing the information produced.

✓Audio pens and other recording tools allow playback of conversations or provide visual reminders of what happened during an elicitation session.

✓Some recording devices also tie the audio directly to the text that was written at the same time, enabling you to hear specific portions of the audio conversation as needed.

Cont'd…

- ✓Tools that support quality checks, such as scanning a requirements document for vague and ambiguous words, help a BA write clearer requirements.
- ✓Some elicitation tools convert requirements from text to auto-generated diagrams.
- ✓Certain tools also enable collaborative voting to help a team prioritize requirements.

Prototyping tools

- Prototyping tools facilitate the creation of work products that range from electronic mock-ups to full application simulations.
- Simple prototyping tools come with basic shapes and designs to create low-fidelity wireframes (Garmahis 2009).
- Sophisticated tools might enable mocked-up functionality that a user can click through to see just how the application would work.

➢ Some prototyping tools support version control, feedback management, requirements linking, and code generation.

- If you use a tool to create high-fidelity prototypes, make it clear to customers that the prototypes are just possible models and that the final product might be different.

Modelling tools

- Requirements modeling tools help the BA create diagrams, Establishing the business requirements, and Specifying data requirements.
  - ➢A picture is worth 1024 words.
- These tools support the use of standard shapes, notations, and syntax for drawing diagrams according to established conventions.
- They might provide templates as starting points and examples to help the BA learn more about each model.
- Often these tools automatically connect shapes in diagrams to accelerate the drawing process and to help ensure that the diagrams are drawn correctly.

➢They also enable you to create diagrams that look cleaner and more consistent than if you draw them manually.

Cont'd…

- Many requirements management tools also provide some modeling capability.
- The most sophisticated tools allow you to trace individual requirements to models or even to specific elements of models.
- For example, analysts can create swimlane diagrams in the tool, and then after they write requirements, they can trace those requirements back to specific steps in the diagrams.

➢ Keep in mind that no tool will be able to tell you if a requirement or a model element is missing, logically incorrect, or unnecessary.

- These tools enable BAs to represent information in multiple ways and to spot certain types of errors and omissions, but they don't eliminate the need for thinking and peer review.

Requirements management tools

- An RM tool that stores information in a multiuser database provides a robust solution to the limitations of storing requirements in documents.

❑Larger project teams will benefit from letting users:

  - import requirements from source documents,
  - define attribute values,
  - filter and display the database contents,
  - export requirements in various formats,
  - define traceability links, and
  - connect requirements to items stored in other software development tools.

- RM tools are more mature than requirements development tools.
  - To be fair, the problem they solve is more tractable.

Cont'd…

- It's easier to create a database in which to store requirements and provide some capabilities to manipulate them.
- Some tools combine both RD and RM capabilities into a powerful solution aid.

Benefits of using an RM tool

- Even if you do a magnificent job of eliciting and specifying your project's requirements, you can lose control of them as development progresses.
- An RM tool becomes most valuable as time passes and the team members' memories of the requirements details fade.

▪ Tasks that we can perform using RM tool:

I. Manage versions and changes

- Some RM tools provide baselining functions.
- The tools also maintain a history of the changes made to each requirement.
- You can record the rationale behind each change decision and revert to a previous version of a requirement if necessary.
- Some tools contain a change-proposal system that links change requests directly to the affected requirements.

Cont'd…

II. Store requirements attributes

- RM tools generate several system-defined attributes, such as the date a requirement was created and its current version number, and they let you define additional attributes of various data types.
- Thoughtful definition of attributes allows stakeholders to select subsets of the requirements based on specific combinations of attribute values.
- A Release Number attribute is one way to keep track of the requirements allocated to various releases.

Cont'd…

III. Facilitate impact analysis

- RM tools enable requirements tracing by letting you define links between different types of requirements, between requirements in different subsystems, and between individual requirements and related system components (for example, designs, code modules, tests, and user documentation).
- These links help you analyze the impact that a proposed change to a specific requirement will have on other system elements.
- It's also a good idea to trace each functional requirement back to its origin or parent so that you know where it came from.
- For instance, you might ask to see a list of all the requirements originating from a specific business rule so that you can judge the consequences of a change in that rule.

Cont'd…

IV. Identify missing and extraneous requirements

- The tracing functionality in RM tools helps stakeholders **identify requirements that are missing**, such as user requirements that have no mapped functional requirements.
- Similarly, they can reveal requirements that cannot be traced back to a reasonable origin, raising the question of whether those requirements are necessary.
- If a business requirement is cut from scope, then all the requirements that trace from it can also be cut quickly.

V. Track requirements status

- Collecting requirements in a database lets you know how many discrete requirements you've specified for the product.
- Tracking the status of each requirement during development supports the overall status tracking of the project.

Cont'd…

VI. Control access

- RM tools let you define access permissions for individuals or groups of users and share information with a geographically dispersed team through a web interface to the database.
- Some tools permit multiple users to update the database contents concurrently.

VII. Communicate with stakeholders

- An RM tool serves as a master repository so that all stakeholders work from the same set of requirements.
- **Some tools permit team members to discuss requirements issues electronically.**
- Automatically triggered email messages notify affected individuals when a new discussion entry is made or when a specific requirement is modified.
- This is a convenient method for visibly tracking decisions made about requirements.

Cont'd…

VIII. Reuse requirements

- Storing requirements in a database facilitates the reuse of them in multiple projects or subprojects.
- Requirements that logically fit into multiple parts of the product description can be stored once and referenced whenever necessary, to avoid duplicating requirements.

IX. Track issue status

- Some RM tools have functionality for **tracking open issues and linking each issue to its related requirements.**
- As issues are resolved, it's easy to determine whether any requirements must be updated.
- You can also quickly find a history of the issue and its resolution.
- Tracking issues in a tool enables automatic reporting on the status of the issues.

Cont'd…

X. Generate tailored subsets

- RM tools allow you to extract and view a set of requirements that fits a particular purpose.
  - For example, you might want a report that contains all of the requirements for a specific development iteration, all of the requirements that relate to a particular feature, or a set of requirements that needs to be inspected.

RM tool capabilities

- The feature tree in the following figure presents a summary of the types of capabilities commonly found in RM tools.
- You can find detailed feature comparisons of many RM tools online (for example, see Seilevel 2011; INCOSE 2010; Volere 2013).

Cont'd…

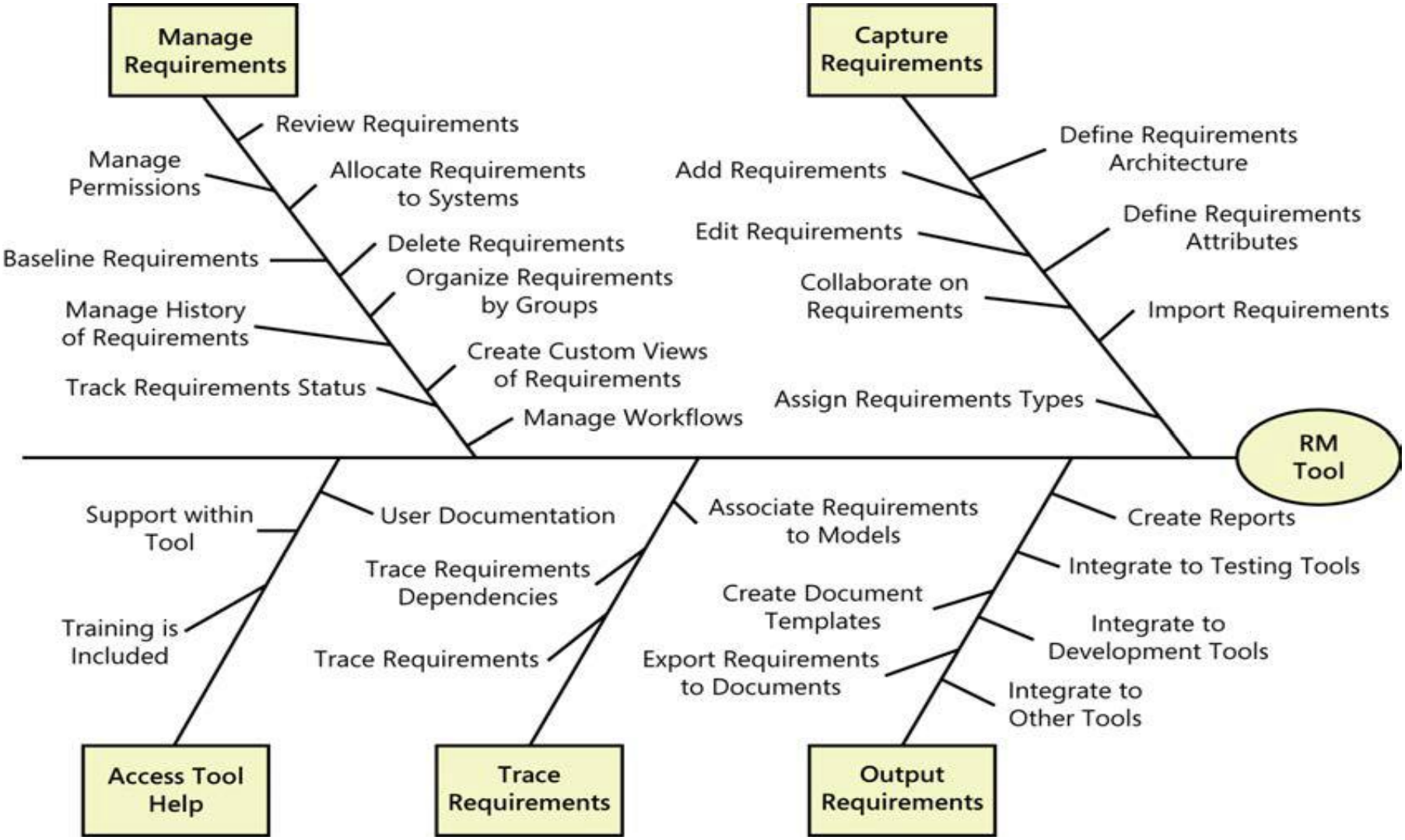


Figure: Common RM tool features

Cont'd…

- RM tools let you define different requirement types, such as business requirements, use cases, functional requirements, hardware requirements, and constraints.
- This lets you differentiate all the types of information that are typically contained in an SRS.
- Many tools allow you to configure an information architecture (which defines how requirements types and other objects relate to one another) that is customized to your practices.
- Most of the tools provide strong capabilities for defining attributes for each requirement type, a great advantage over the typical document-based approach.

Cont'd…

- RM tools typically support hierarchical numeric requirement labels, in addition to maintaining a unique internal identifier for each requirement.
- These identifiers often consist of a short text prefix that indicates the requirement type—such as UR for a user requirement—followed by a unique integer.

Cont'd…

- Requirements can be imported into an RM tool from various source document formats.
- The textual description of a requirement is treated simply as a required attribute.
- Several products let you incorporate non-textual objects such as graphics and spreadsheets into the requirements repository.
- Other products let you link individual requirements to external files (such as Microsoft Word files, graphics files, and so on) that provide supplementary information that augments the contents of the requirements repository.

Cont'd…

- Output capabilities from the tools generally include the ability to generate a requirements document in a variety of formats, including predefined or user-specified documents, spreadsheets, and webpages.
- Specification documents are then simply reports that are generated from the tool according to certain query criteria, formatted to look like a typical SRS.
- For example, you could create an SRS that contains all the functional requirements that are allocated to a specific release and assigned to a particular developer.
- Some tools provide functionality that lets users make changes in exported documents offline, which are then synchronized with the tool's database when the user is back online.

Cont'd…

- Most tools enable different views of the requirements to be generated within the tool or exported from the tool.
- Features typically include the ability to set up user groups and define permissions for selected users or groups to create, read, update, and delete projects, requirements, attributes, and attribute values.
- Setting up appropriate views and permissions facilitates the review of requirements and collaboration to improve those requirements.
- Some tools also include learning aids, such as tutorials or sample projects, to help users get up to speed.

Cont'd…

- Requirements management tools generally have robust tracing features.
- Tracing is handled by defining links between two types of objects or objects of the same type.
- Some requirements management tools include modeling capabilities that also allow the models to be linked at an element level to individual requirements or to other model elements.
- RM tools often integrate with other tools used in application development.
- For instance, you might be able to trace specific requirements to individual design elements stored in a design modeling tool, or to tests stored in a test management tool.

Cont'd…

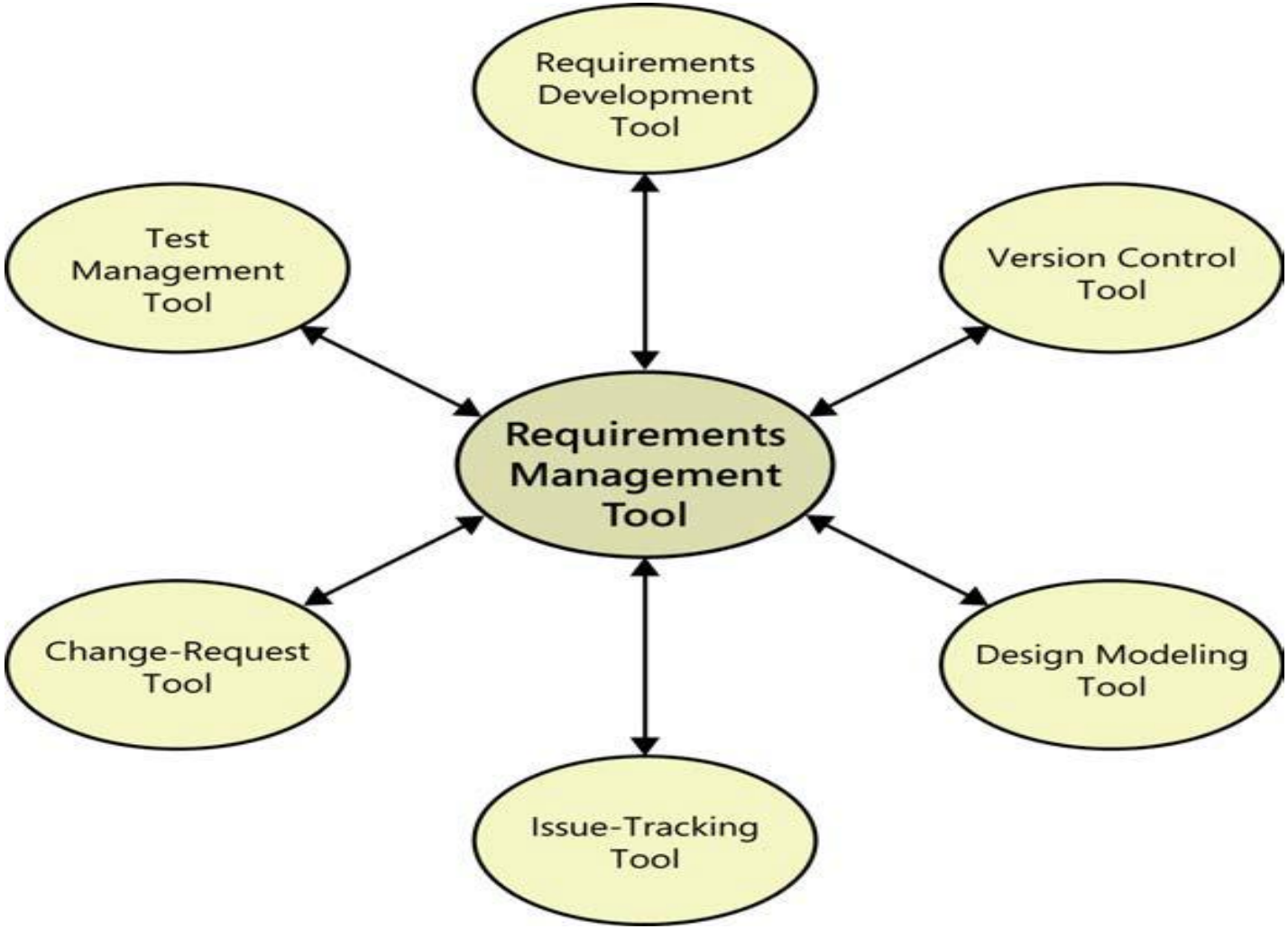


Figure**:** RM tools integrate with other kinds of software tools

Cont'd…

- When you are selecting an RM product, determine whether the tool will be able to exchange data with the other tools you use.
- Think about how you'll take advantage of these product integrations as you perform your requirements engineering, testing, project tracking, and other processes.
- For example, consider how you would define trace links between functional requirements and specific design or code elements, and how you would verify that all tests linked back to specific functional requirements have been successfully executed.

Selecting and implementing a requirements tool

- Success depends upon selecting the most appropriate tool for your organization and getting your teams to adopt it as part of their routine practices.

Selecting a tool

✓Select a tool based on the combination of desired features, platform, and pricing that best fits your development environment and culture.

The Selection process:

1. Identify your organization's requirements for the tool to serve as evaluation criteria.
2. Prioritize and weight the criteria according to what capabilities or other factors matter most to your organization.

Cont'd…

3. Set up demos or acquire evaluation copies of the tools you want to consider.
4. Score each tool against the criteria in a consistent manner.
5. Calculate a total score for each tool by using your criteria scores and the weights you assigned to them.
6. For each tool that scored well, use it on an actual project to see if it behaves as you anticipated from the objective scores.
7. To make a final selection, combine the scores, licensing costs, and ongoing costs with information on vendor support, input from current users, and your team's subjective impressions of the products.
   - Two good final questions to ask people who evaluate the tools are, "Which tool would you most want to use?" and, "Which tool would you be most upset about being forced to use?"

Setting up the tool and processes

- Recognize that it will take effort to:
  - ✓install a tool,
  - ✓load a project's requirements into it,
  - ✓define attributes and trace links,
  - ✓keep the contents current,
  - ✓define access groups and their privileges, and
  - ✓adapt your processes to use the tool.

- Configuring the tool can be complex; there is a steep learning curve just to set up a sophisticated requirements tool.

➢Management must allocate the resources needed for these operations.

➢Make an organization-wide commitment to actually use the product you select, instead of letting it become expensive shelf ware.

Cont'd…

- There's little point in using a requirements tool if you don't take advantage of its capabilities.
- Even if you select the best available tool, it won't necessarily provide every capability that your organization wants or needs.
- It might not support your existing requirements templates or processes.
- You'll still likely need to adapt some of your existing processes to incorporate the tool in them.
- Expect to have to make some changes to templates, attribute names, and the sequencing of requirements development activities.

❑ Consider the following suggestions to overcome process issues as you strive to maximize your return on investment from a requirements tool:

Cont'd…

- Assign an experienced BA to own the tool setup and process adaptations.
  - She/He will understand the impact of configuration choices and process changes.
- Think carefully about the various requirement types that you define.
  - Don't treat every section of your current SRS template as a separate requirement type, but don't simply stuff all of the SRS contents into a single requirement type either.
- Use the tool to facilitate communication with project stakeholders in various locations.
  - Set the access and change privileges to permit sufficient input to the requirements by various people without giving everyone complete freedom to change everything in the database.

Cont'd…

- Don't try to capture requirements directly in an RM tool during your early elicitation workshops.
  - As the requirements begin to stabilize, though, storing them in the tool makes them visible to the workshop participants for refinement.
- Use RD tools during elicitation activities only if you are confident that they will not slow down the discovery process and waste your stakeholders' time.
- Don't define trace links until the requirements stabilize.
  - Otherwise, you can count on doing a lot of work to revise the links as requirements continue to evolve.

A requirement management tool is a software application that helps you to document, analyze, prioritize, trace, and manage the requirements of a software project. A good requirement management tool can help you to avoid common requirements errors, improve communication and collaboration among stakeholders, and ensure that the software product meets the customer's needs and expectations.

There are many requirement management tools available in the market, each with its own features, benefits, and drawbacks. Some of the best requirement management tools are:

- **Modern Requirements**: This is a requirement management tool that integrates with Azure DevOps and TFS. It allows you to create, review, approve, and trace requirements using various techniques such as use cases, user stories, diagrams, mockups, matrices, and tests. It also supports agile and hybrid methodologies, risk management, baselining, reporting, and compliance[1].
- **Jama Software**: This is a requirement management tool that helps you to manage complex products, systems, and software. It enables you to capture and communicate requirements, goals, progress, and interdependencies throughout the development process. It also offers end-to-end traceability, impact analysis, reviews and approvals, team collaboration, risk management, test management, and compliance[2].
- **SpiraTeam**: This is a requirement management tool that integrates with other tools for project management, test management, bug tracking, and collaboration. It allows you to create and manage requirements using various formats such as text, images, diagrams, tables, and attachments. It also supports traceability matrixes, prioritization, baselining, reporting, and agile methodologies[3].

Chapter Six

Requirements Management

Introduction

- Once a system has been deployed, new requirements inevitably emerge.
- It is difficult for the users to anticipate the effect of these new requirements (if a new system is developed for these requirements) on the organization.

➢ Thus, to understand and control changes to system requirements, requirements management is performed.

- **Requirements management** can be defined as a process of eliciting, documenting, organizing, and controlling changes to the requirements.

Cont'd…

- **The principal concerns of requirements management are:**
  1. Managing changes to agreed requirements
  2. Managing the relationships between requirements
  3. Managing dependencies between the requirements document and other documents produced during the systems and software engineering process.

Cont'd…

- Generally, the process of **requirements management begins as soon as the requirements document is available**, but '**planning**' for managing the changing requirements should start during the requirements elicitation process.
- The essential **activities** performed in requirements management are listed below.

  I. Recognizing the need for change in the requirements

  II. Establishing a relationship amongst stakeholders and involving them in the requirements engineering process

  III. Identifying and tracking requirements attributes.

Cont'd…

- Requirements management enables the development team to
  - identify,
  - control, and
  - track requirements and changes that occur as the software development process progresses.
- Other **advantages** associated with the requirements management are listed below.

Cont'd…

1. **Better control of complex projects:**
   - This provides the development team with a clear understanding of what, when, and why the software is to be delivered.
   - The resources are allocated according to user-driven priorities and relative implementation effort.
2. **Improved software quality:**
   - This ensures that the software performs according to the requirements to enhance software quality.
   - This can be achieved when the developers and testers have a precise understanding of what to develop and test.

Cont'd…

3. **Reduced project costs and delays:**
   - This minimizes errors early in the development cycle as it is expensive to 'fix' errors at the later stages of the development cycle.
   - As a result, the project costs also reduce.
4. **Improved team communication:**
   - This facilitates early involvement of users to ensure that their needs are achieved.

Cont'd…

5. **Easing compliance with standards and regulations:**
   - This ensures that standards involved with software compliance and process improvement have a thorough understanding of requirements management.
6. **All the user requirements are specified in the software requirements specification.**
   - The project manager as part of requirements management tracks the requirements for the current project and those which are planned for the next release.

Requirements Management Process

- Requirements management starts with planning, which establishes the level of requirements management needed.
- After planning, each requirement is assigned a unique ‘identifier’ so that it can be crosschecked by other requirements.
- Once requirements are identified, requirements tracing is performed.

Requirements tracing

- it is a medium to trace requirements from the start of development process till the software is delivered to the user.
- The **objective** of requirements tracing is to ensure that all the **requirements are well understood** and included in test plans and test cases.

Cont'd…

- A requirement is **traceable** if you can discover:
  - ✓ who suggested the requirement,
  - ✓ why the requirement exists,
  - ✓ what requirements are related to it and
  - ✓ how that requirement relates to other information such as systems designs, implementations and user documentation.

Cont'd…

- Various **advantages** of requirements tracing are listed below.
  1. It verifies whether user requirements are implemented and adequately tested.
  2. It enables user understanding of impact of changing requirements.

Cont'd…

- Traceability techniques facilitate the impact of analysis on changes of the project, which is under development.
- Traceability information is stored in a traceability matrix, which relates requirements to stakeholders or design module.
- The traceability matrix refers to a table that correlates high-level requirements with the detailed requirements of the product.
- Mainly, **five types of traceability tables** are maintained.
  - These are listed in Table.

Cont'd…

- In a traceability matrix, each requirement is entered in a row and column of the matrix.
- The dependencies between different requirements are represented in the cell at a row and column intersection.
- '**U**' in the row and column intersection indicates the **dependencies** of the requirements **in the row on the column** and 'R' in the row and column intersection indicates the existence of some other weaker relationship between the requirements.

Cont'd…

Table : Types of Traceability Tables

Traceability Table	Description
Features traceability	Indicates how requirements relate to important features specified by the user.
Source traceability	Identifies the source of each requirement by linking the requirements to the stakeholders who proposed them. When a change is proposed, information from this table can be used to find and consult the stakeholders.

Cont'd…

Traceability Table	Description
Requirements traceability	Indicates how dependent requirements in the SRS are related to one another. Information from this table can be used to evaluate the number of requirements that will be affected due to the proposed change(s).
Design traceability	Links the requirements to the design modules where these requirements are implemented. Information from this table can be used to evaluate the impact of proposed requirements changes on the software design and implementation.

Cont'd…

Traceability Table	Description
Interface traceability	Indicates how requirements are related to internal interface and external interface of a system.

Cont'd…

Traceability Matrix

Req. ID	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		U	R					
1.2			U			R		U
1.3	R			R				
2.1			R		U			U
2.2								U
2.3		R		U				
3.1								R
3.2							R	

Cont'd…

- Note that a traceability matrix is useful when less number of requirements are to be managed.
- However, traceability matrices are **expensive** to maintain when a large system with large requirements is to be developed.
- This is because large requirements are not easy to manage.

➢ Due to this, the traceability information of large system is stored in the 'requirements database where each requirement is explicitly linked to related requirements.

- This helps to assess how a change in one requirement affects the different aspects of the system to be developed.

Requirements Change Management

- Requirements change management is used when there is a request or proposal for a change in the requirements.
- The advantage of this process is that the changes to the proposals are managed consistently and in a controlled manner.
- An efficient requirements change management process undergoes a number of stages for changes to the requirements.
- These stages are listed below.

Cont'd…

1. **Problem analysis and change specification:**
   - The entire process begins with identification of problems to the requirements.
   - The problem or proposal is analyzed to verify whether the change is valid.
   - The outcome of the analysis is provided to the 'change requester' and a more specific requirements change proposal is then made.
2. **Change analysis and costing:**
   - The effect of a change requested on the requirement is assessed according to traceability information.
   - The cost for this can be estimated on the basis of modification made to the design and implementation.
   - After the analysis is over, a decision is made whether changes are to be made.

Cont'd…

3. Change implementation:

- Finally, the changes are made to the requirements document, system design and implementation.
- The requirements document is organized in such a manner so that changes to it can be made without extensive rewriting.
- Minimizing the external references and making document sections modular achieves changeability in the document.
- By doing this, individual sections can be changed and replaced without affecting other parts of the document.

Thank You!

?