COMP11113

Information Systems Analysis and Design

Unit 1

An Introduction to System Development and System Analysis

Aims

To provide an overview of the system development process and software lifecycle models, and describe how system analysis is involved in the process and software lifecycle models, as well as explain how system analysis is carried out in the development.

Objectives

When you have completed this unit, you should be able to:

- Describe the process of system development.
- Discuss the relationship among software quality, development process, development methods and techniques, and CASE tools in system development.
- Define the software lifecycle models.
- Explain how CASE tools support system development.
- Explain the factors that affect system quality.
- Explain the role of the system analyst in system development.
- Explain the importance of the user and analysis techniques for the system analyst.
- Specify the categories of requirements.
- Explain the process of system analysis.
- Understand the approaches to collecting requirements.

Overview

This unit introduces you to the system development process that consists of different stages: system analysis, system design, system implementation, system testing, and system maintenance. System developers need to go through these stages for developing a new computer system. This unit also covers the issues relating to the system analysis that is the first stage of system development and that is important to the success of the system because it is responsible for requirements capture and system modelling and the results of the analysis will have impact on the other stages of the process.

In this module the topics are broken down into the two sections:

Section 1 examines the system development process and explains how a computer system is developed through the different stages in the process. It defines software lifecycle models and measurement of software quality.

Section 2 then focuses on system analysis and examines it in deep detail. It looks at the role of the system analyst and the user of the system. It further examines the process of system analysis and tasks of the system analyst. This section also discusses the categories of user requirements, the ways of capturing the requirements, and the specification of the requirements.

Reading of the textbooks:

John W. Satzinger, Robert B. Jackson and Stephen D. Burd, *Object-Oriented Analysis and Design with the Unified Process,* Thomson, 2005.

Alan Dennis, Barbara Wixom, and David Tegarden, Systems Analysis and Design: An Object-Oriented Approach with UML (6th edition), Wiley, 2021.

Table of Contents

1 AN INTRODUCTION TO SYSTEM DEVELOPMENT PROCESS

In this section we will provide an overview of software engineering and its topics that are related to system analysis and design. The main purpose of this section is to provide a background context from which to explore concepts relating to system analysis and design.

1.1 Objectives

After completing this section you should be able to:

- define the terms software engineering, system development process, software lifecycle model, system development technique, CASE tool, and software quality.
- describe the relationships among quality, process, technique, and CASE tool in software engineering.
- describe system development process and software lifecycle models.
- describe the role of system analysis and design in the development process.
- describe common factors that affect the software quality.

1.2 Software Engineering (SE) - A Layered Technology

Software engineering is a subject about software development and addresses important issues in software development that include software quality, development process, development methods, and CASE (computer aided software engineering) tools. It aims to support software development by using appropriate development techniques in the development throughout a development process. It can be described as a layered technology for the software development as shown in Figure 1:

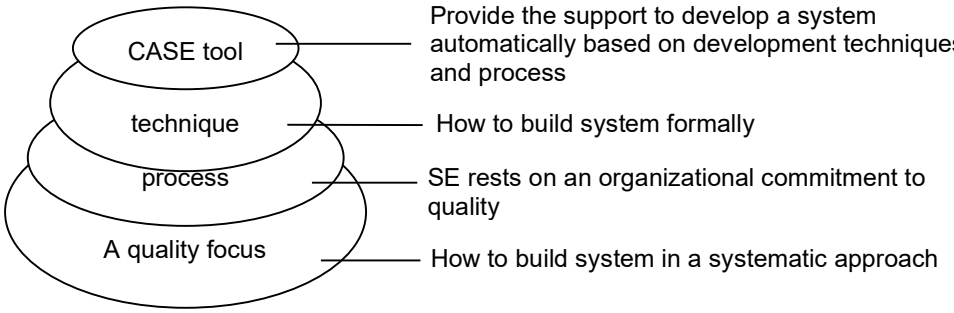

Figure 1.1: Software Engineering

The following subsections explain system development process, software lifecycle models, software development techniques, CASE tools, and software quality in detail.

1.3 System Development Process on Software Lifecycle

A system development process usually is spilt into five stages that are system analysis, system design, system implementation, system testing, and system maintenance.

- **System Analysis**
  System development begins by establishing requirements for all system elements and then allocating some subset of these requirements to system. System analysts must understand the problem, as well as the required function, performance, and interfacing.

  Specifications of a system are documented and reviewed by the system analyst with the user of the system.

- **System Design**
  System design is actually a multi-step process that focuses on four distinct attributes of the program: data structure, system architecture, procedural detail, and interface characterization.

  The design process translates system specifications into a representation of the software that can be assessed for quality before coding begins. Like specifications, the design detail is documented and becomes part of the software configuration.

- **System Implementation (Programming)**
  System implementation translates the design into a machine-readable form (i.e. source code) that is called a program written in a programming language.

- **System Testing**

System testing focuses on the logical internals of the system, ensuring that all statements have been tested, and on the functional externals.

System testing aims to find errors in software and ensure that defined input will produce actual results that agree with required results.

- **System Maintenance**
  A system will undergo change after it is delivered to the user. Change will occur because of the following factors:
  - Errors have been encountered
  - The system must be adapted to accommodate changes in its external environment (e.g. a new operating system)
  - The customer requires functional or performance enhancements.

  System maintenance reapplies each of the stages in the above to an existing system rather than a new one.

1.4 Software Lifecycle Models

Software lifecycle is the lifetime of a system from starting to use it to discarding it by the user. Software lifecycle involves a number of techniques threaded together in system development.

A software lifecycle model is a description of the overall shape of a system development project. This model captures the management control and iterative strategies for a project. Three software lifecycle models are often used by the system developer in system development. They are waterfall model, evolutionary model, and incremental model.

- Waterfall Model
  The waterfall model attempts to separate the lifecycle into a linear series of activities, each of which must be completed before the next is started. It is described in Figure 1.2.

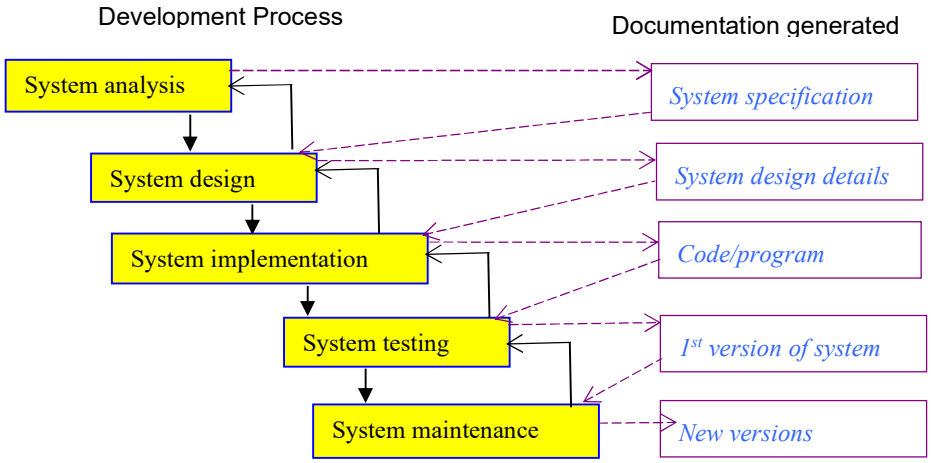

Figure 1.2: Waterfall Model

- Evolutionary Model
  The evolutionary model is a deliberate development strategy producing successively better versions of a production system. It is described in Figure 1.3.

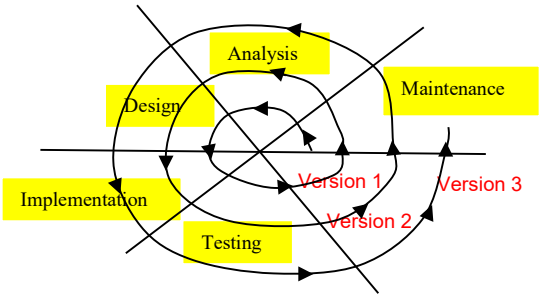

Figure 1.3: Evolutionary Model

The first version of the system might be built using rapid development tools, with successive versions being refinements to the system in terms of application performance, or extending the limits of data storage and so on.

- Incremental Model
  The incremental model addresses the need to develop production-quality software in small, manageable components called *increments*. It splits a system into small parts and develops a part of the system at one time throughout the development process, as described in the Figure 1.4.

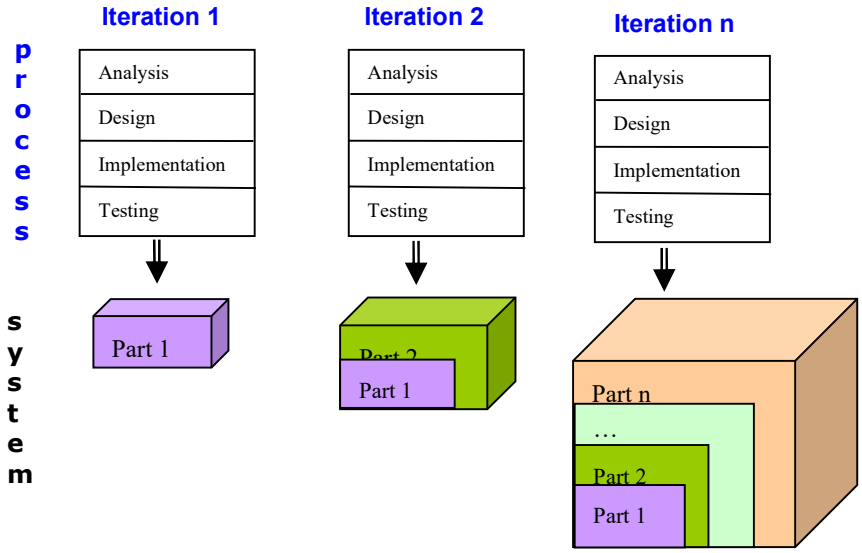

Figure 1.4: Incremental Model

This model is particularly useful in situations where the software requirements may be unstable such as for large, long-term development. It reduces the risk and improves management confidence by producing new “assets” at regular intervals.

Rapid prototyping techniques may be introduced into either the evolutionary or incremental models.

1.5 System Development Techniques

System development techniques are used to support system development. They can help

manage the complexity of a system and develop the system efficiently.

Traditional development techniques, e.g., structured techniques like dataflow diagrams.

Modern development techniques, e.g., object-oriented techniques like UML.

1.6 Case Tools Supporting Processes and Techniques

Modern CASE (Computer Aided Software Engineering) tools provide an increasingly wide range of facilities and cover most life cycle activities in system development.

CASE tools enable to develop software automatically and to improve productivity.

A CASE tool is designed for use with one or more development approaches to software development, particular notations and implementation environments.

Model and diagram support

- Checks for syntactic correctness.
- Data dictionary support
- Checks for consistency and completeness.
- Navigation to linked diagrams.
- Layering.
- Traceability.
- Report generation.
  - System simulation.
  - Performance analysis.
- CASE tools can offer a range of features to support code generation and maintenance.

1.7 Measurement of Software Quality

The following four factors can be used to measure the quality of a system:

- Correctness of system: the system must run correctly in accordance with the user requirements.
- Maintainability of system: the system can be changed easily and efficiently in a given period.
- Integrity of system: the system needs to be developed with high security and it is able to avoid any attack.
- Usability of system: the system must provide a user-friendly way of using the system.

1.8 Self-Assessment Questions

1. What is Software Engineering?

2. What is the system development process?

3. What is software lifecycle?

4. What is a software lifecycle model?

5. How many different software lifecycle models are used in system development?

6. What is a system development technique?

7. What can CASE tools support in system development?

8. What factors can be used to measure software quality?

1.9 Learning Outcomes

Having completed this section, you should be able to:

- Describe the process of system development.
- Discuss the relationship among software quality, development process, development methods and techniques, and CASE tools in system development.
- Define the software lifecycle models.
- Explain how CASE tools support system development.
- Explain the factors that affect system quality.

2 SYSTEM ANALYSIS

2.1 Introduction

In this section we will introduce system analysis in the software lifecycle models and explain what it aims to do in the system development process.

2.2 Objectives

After completing this section, you should be able to:

- Explain the role of the system analyst in system development.
- Explain the importance of the user and analysis techniques for the system analyst.
- Specify the categories of requirements.
- Explain the process of system analysis.
- Understand the approaches to collecting requirements.

2.3 What is System Analysis?

System analysis is to gather and interpret facts, diagnose problems and use the facts to improve the system. Analysis specifies what the system should do; design states how to accomplish the objective.

2.4 Systems Analyst and User

2.4.1 Role of System Analyst

• Assessment — observing, understanding and evaluating interactions of a system:

- What, why, how is being done?
- Who is doing it?
- What are the problems in doing it?

• Assistance - offering specific suggestions for improving effectiveness:

- What other ways existing for dealing with this problem?
- What kind of benefits/liabilities are associated with alternatives?

2.4.2 User Involvement in System Development

• Philosophy - system succeeds when there is heavy user involvement during all phases.

• Result - favourable user attitudes. Users regard the system as "ours".

2.5 Process of System Analysis

2.5.1 Planning

One of the main causes of projects failure is inadequate understanding of the requirements. One of the main causes of inadequate understanding of the requirements is poor planning of system analysis.

Before planning how to do the investigation and collecting information, the analyst needs to consider:

- What type of information is required?
- What are the constraints on the investigation?
- What are the potential problems which may make the task more difficult?

And then think about:

- Critical information you require before the investigation starts.
- How you will get this information.
- The fact-finding techniques which will be appropriate.
- The danger areas for the project and for your companies.

In producing the plan, a list of questions you would need to ask is made. The questions cover the scope of the investigation – the resources available, the budget, the time scale, the key business people to speak to and analysis restrictions.

The analyst must ensure to:

- understand the objectives and terms of reference agreed with the client.
- are aware of constraints which impact the analysis process.

plan the research, initial contact and other tasks to be completed during the investigation and manage time appropriately.

2.5.2 Requirements Analysis and Determination

The objectives of requirements analysis and determination are

- Provide a firm basis for the design of a future system.
- Understand the scope of the project.
- Increase user confidence that you understand the nature of the problem fully and are competent to proceed further.

To design and justify the new system, we need to investigate:

- operations and data for helping understand requirements more easily.
- problems in the current system for ensuring they will not be replicated in the new system.
- boundaries of the system for avoiding needless effort and ensure all necessary areas are examined.

Categories of requirements

- *Functional requirements* that describes what a system does or is expected to do, often referred to as its functionality.
  - Processes carried out.
  - Inputs and outputs.
  - Data stored.
- *Non-functional requirements* that describes aspects of the system that are concerned with how well it provides the functional requirements.
  - Performance criteria (e.g. desired response time).
  - Anticipated volumes of data, either in terms of throughput or of what must be stored.
  - Security considerations.

- *Usability requirements* that will enable to ensure that there is a good match between the system to be developed and both the user of that system and the tasks that they will undertake when using it.
  - Characteristics of the users who will use the system.
  - The tasks which the users undertake, including the goals which they are trying to achieve.
  - Situational factors which describe the situations which could arise during system use.
  - Acceptance criteria by which the user will judge the delivered system.

Requirements capture

- User requirements are the needs of people who will use the system.
- The analyst must understand the current system, e.g.
  - its objectives
  - how it is operating at present and how people are working now.
- The analyst has to
  - gather and document the information about what people are doing, in order to carry forward significant aspects of the current system into the new system.
  - find out new requirements, i.e., what users require in the new system that they cannot do with their existing system.

Use a fact-finding technique for investigating the current system. Record what the current system does:

- Prepare questions in advance of interview.
  - Avoid leading questions which suggest obvious answers.
  - Should direct the other person to the right area.
  - Should allow freedom of expression to provide complete answers.
- Talk to the individuals who's jobs are under investigation.
- Record the dialogue.
  - Fully record content.
  - Transcribe and organize notes as soon as possible to aid recall.
  - Summarize into clear, concise statements to confirm substance of what was discussed.
- Need to learn about the job; understand underlying problems.

Identify the problems existing in the current system and decide new requirements for the new system.

Check the established system:

- Review investigation with users
  - Continuous informal reviews during data gathering
  - Formal end of stage review
    - Confirm system boundary
    - Confirm correctness and completeness of diagrams and supporting documentation.
- Validation
  - "Are we building the right system".

Check that the system's functions are what the customer really wants.

2.5.3 Requirements Specification

Having carried out our investigation of the current system, we now want to specify the new system in terms of modelling techniques such as UML. Three system perspectives:

- A data view (what data needs to be stored) that describes the real-world system in terms of attributes and associations that must be stored.
- A process/function view (what is happening) that describes the operations carried out on that data.

- An event/state view (how data changes over time) that describes the time sequence and time constraints on individual processes, or events as triggers to trigger the execution of processes.

2.6 Self-Assessment Questions

1. What is system analysis?

2. What is the role of the system analyst and the user?

3. What are user requirements?

4. What are functional requirements, non-functional requirements?

5. Give examples of functional requirements and non-functional requirements.

6. How to capture user requirements?

7. What does a requirement specification means?

2.7 Learning Outcomes

Having completed this section, you should be able to:

- Explain the role of the system analyst in system development.
- Explain the importance of the user and analysis techniques for the system analyst.
- Specify the categories of requirements.
- Explain the process of system analysis.
- Understand the approaches to collecting requiremen