

CHAPTER

2 Information System Development

LEARNING OUTCOMES

By the end of this chapter, you should be able to:

1. Define methodology, model, tool and techniques;
2. Differentiate system development methodologies between Structured Analysis and Object Oriented Analysis and Design;
3. Describe other alternative approaches to system development such as Rapid Application Design and Prototyping; and
4. Life Cycle approach.

INTRODUCTION

In the early years of computing, analysis and design was considered an art. Now that the need for systems and software has become so great, people in industry and academia have developed work methods that make analysis and design a disciplined process. The goal is to help you develop the knowledge and skills needed to understand and follow various methodologies, techniques and tools that have been developed, tested and widely used over the years during system analysis and design.

To develop a complete and comprehensive system, all the elements related to the system as shown in Figure 2.1 must be taken into account.

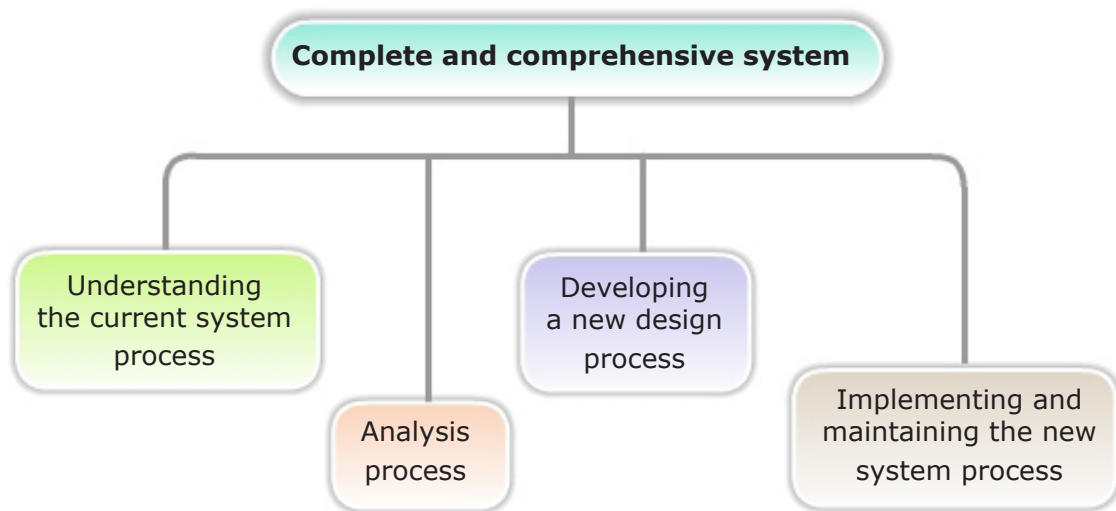


Figure 2.1: Elements related to developing a complete and comprehensive system.

This chapter, explains definition of methodology, model, tool, techniques, system development methodology, other alternatives approaches to development and finally phases in the System Development Life Cycle.

2.1 DEFINING METHODOLOGY, MODEL, TOOL AND TECHNIQUES

2.1.1 Methodology

Some of the definitions for Methodology are shown in Figure 2.2.

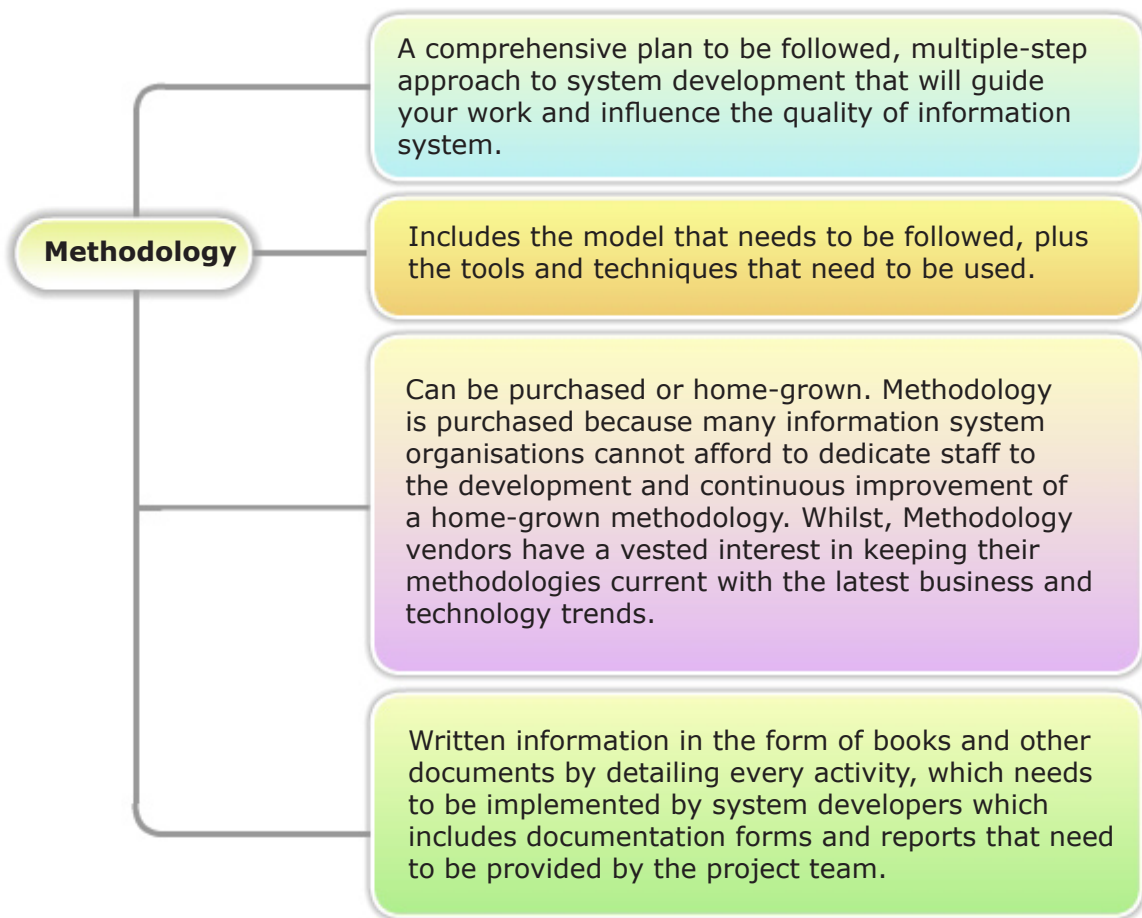


Figure 2.2: Some of definitions for Methodology.

There are also methodologies in a more shortened form, which contain general instructions on what needs to be implemented. Many methodologies today have been shaped from other methodologies, after some adaptations, to suit the needs of the new system development.

The benefits of implementing methodologies in system development process are shown in Figure 2.3.

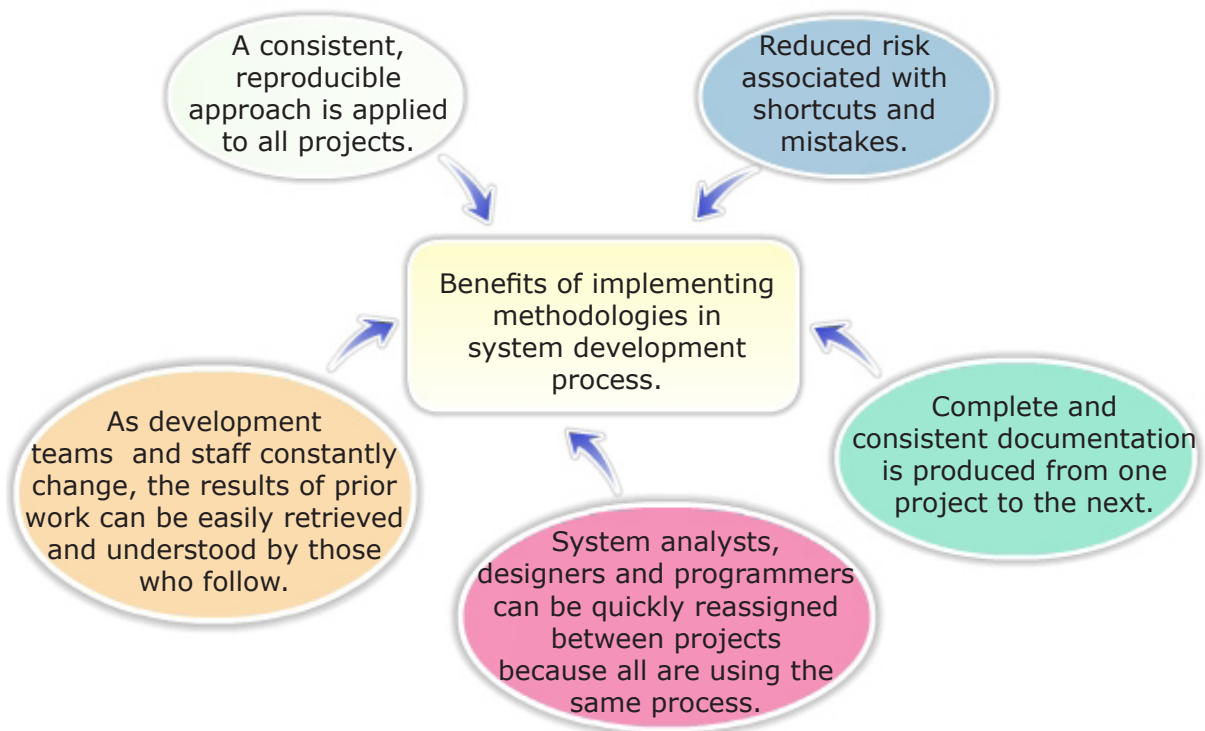


Figure 2.3: Benefits of implementing methodologies in system development process.

2.1.2 Model

A **model** is a graphical representation of a concept or process that system developers can analyse, test and modify. A model can be built for an existing system as a way to better understand those systems or for proposed systems as a way to document business requirements or technical design.

As an example, try to imagine the model of a house. To enable you to see the design aspect of the house, a housing developer would always build a small model in a three-dimensional view. For a single-storey house, the model must show the physical building and also the floor shape of the house. The real world is the terrace house, which is going to be built later, based on the model.



A system analyst can describe an information system by using a set of business data, object, network and process models as shown in Figure 2.4. A **business model** describe the information that a system must provide, for example business events and business rules would be investigated as an input to the specification of the new automated system.

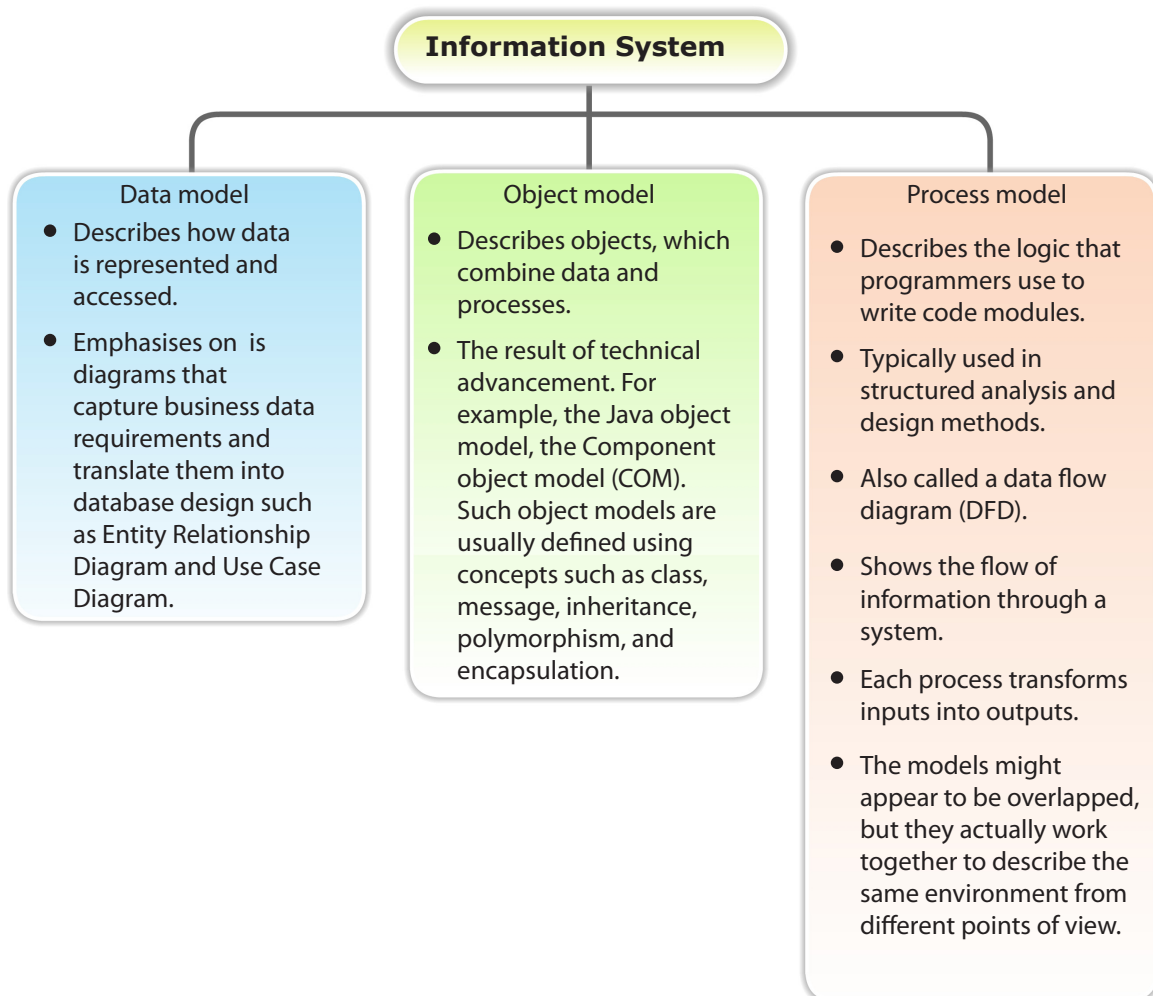


Figure 2.4: Information system described by data, object and process model.

2.1.3

Tools

Tools refer to the supporting software that are used to create models or other components that are needed in a certain project.

As an example, the software for drawing a diagram is considered a tool for drawing the picture of certain diagrams and charts. Tools also include a database application, which is used for collecting information for a certain project, such as definitions of data flows or a list of written instructions about a certain process. As an example, the specific software used for building information system called Computer-Aided System Engineering (CASE tools).

CASE tools provide an overall framework for system development and support a wide variety of design methodologies including structured analysis and object-oriented analysis. System developers often use project management tools such as Microsoft Project and special-purpose charting tools such as Microsoft Visio to create many different types of diagrams as shown in Figure 2.5.

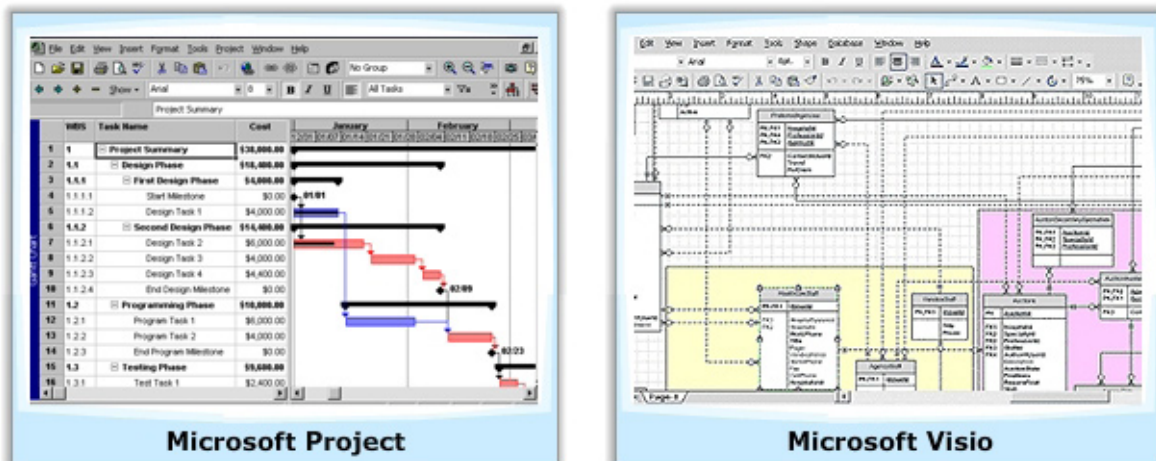


Figure 2.5: Samples of project management tool and special-purpose charting tools.

2.1.4 Techniques

Technique is a strategy which can be used for implementing a specific system development activity.

Techniques are processes that you as an analyst, will follow to help ensure that your work is well thought-out, complete, and comprehensible to others. Techniques provide support for a wide range of tasks including conducting thorough interviews with current and future users of information system to determine what your system should do. It also support planning and managing the activities in a systems development projects, diagramming how the system will function and designing the reports such as invoices your system will generate for its users to perform their jobs.

After knowing the terms that are being used in information systems development such as methodology, models, tools and techniques, do you know their relationship to each other?

Methodology is a collection of models, tools and techniques that are being used for implementing activities inside all the phases of systems development. These activities include completing various models and also producing documents. For these to be possible, systems developers use special software such as tools to support them in the implementation of activities.

To be effective, techniques, models and tools must both be consistent with an organisation's system methodology. Techniques and tools must make it easy for systems developers to conduct the steps called for in the methodology. These four elements - methodologies, techniques, models and tools must work together to form an organisational approach to system analysis and design as shown in Figure 2.6.

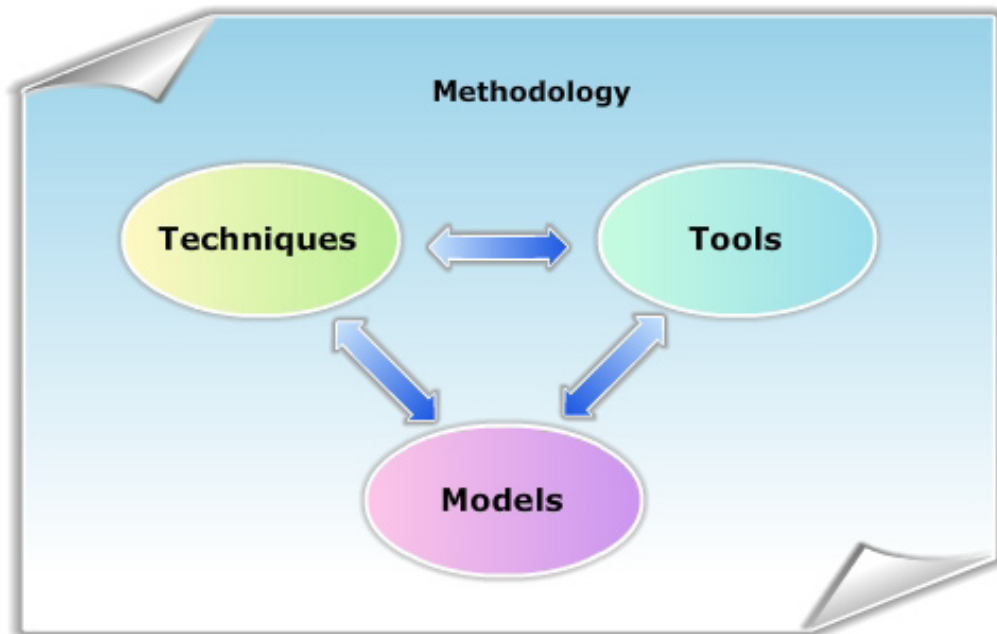


Figure 2.6: Relationship between methodology components.

ACTIVITY



What is the difference between:

- Model and Tool
- Techniques and Methodology

2.2 SYSTEM DEVELOPMENT METHODOLOGY

This section discusses various methods for developing computer-based information systems. Structured analysis is the most popular method, but newer strategy called object-oriented analysis and design also is used widely. Each method offers many variations. Some organisations develop their own approaches or adapt methods offered by software suppliers, CASE tool vendors or consultants. Most IT experts agree that no single, best system development strategy exists. Instead, a system analyst should understand the alternative methodologies and their strengths and weaknesses.

2.2.1 Structured Analysis

Structured Analysis

- A traditional systems development method which is easy to understand.
- Uses a series of phases, called the system development life cycle (SDLC), to plan, analyse, design, implement and maintain an information system.
- Evolved when most systems were based on mainframe processing, and remains a dominant systems development method.
- Uses a set of process models to describe a system graphically.
- Focuses on processes that transform data into useful information.
- A process-centred technique which breaks up a system or large program into smaller layers of modules.
- In addition to modelling the processes, it includes data organisation and structure, relational database design and user interface issues.

Figure 2.7: Characteristics of Structured Analysis

Process modelling identifies the data flowing into a process, the business rules that transform the data and the resulting output data flow. Figure 2.8 shows a simple process model that represents a school registration process with related input and output. The Register Student process accepts input data from two sources and transforms it into output data.

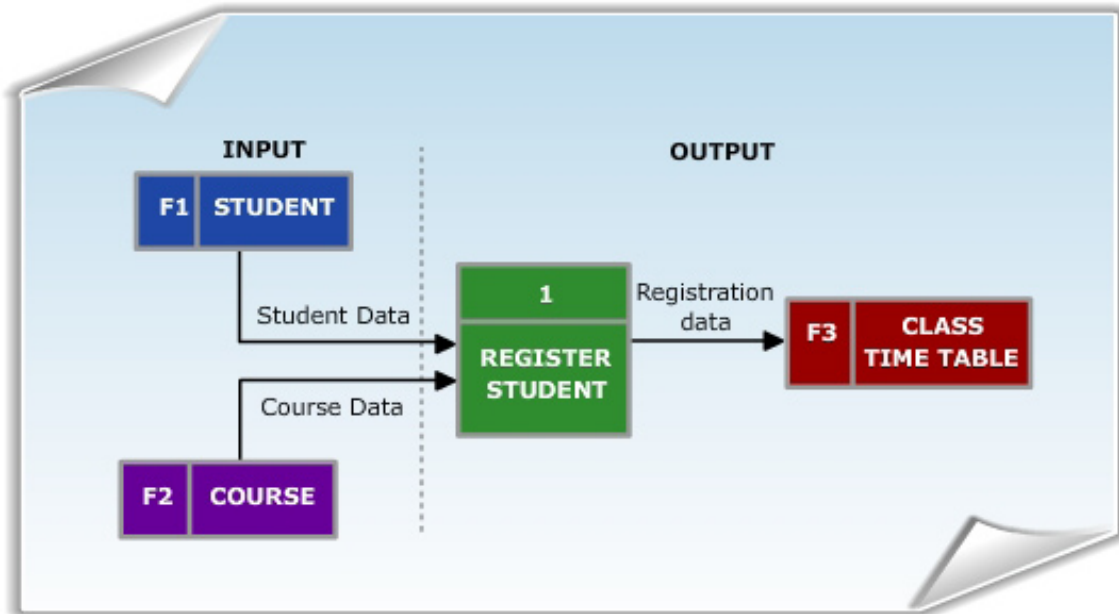


Figure 2.8: Process model for student registration system

2.2.2 Object-Oriented Analysis and Design

Whereas structured analysis treats process and data as separate components, Object-oriented analysis and design (OOAD) combines data and the processes (called methods) into single item called objects. OOAD is an approach that is intended to facilitate the development of systems that must change rapidly in response to dynamic business environments. OOAD works well in situation which complicated information systems are undergoing continuous maintenance, adaption and redesign.

System analysts use OOAD to model real-world business processes and operations. The result is a set of software objects that represent actual people, things, transactions and events. OOAD use industry standards for modelling information systems, called the unified modelling language (UML).

Objects are persons, places or things that are relevant to the system we are analysing. Each object can be viewed as an independent little machine with a distinct role or responsibility. Each object is capable of receiving messages, processing data, and sending messages to other objects.

A class defines the set of shared attributes and behaviours found in each object in the class. For example, records for students in a course section have similar information stored for each student. The values maybe different for each student, but the type of information is the same.

What makes OOAD different from structured approach is the technique of putting all of the object's attribute and methods within one self-contained structure, the class itself. This is a familiar occurrence in the real-world. For example, a packaged cake mix is analogous to a class since it has both the ingredients as well as instructions on how to mix and bake the cake.

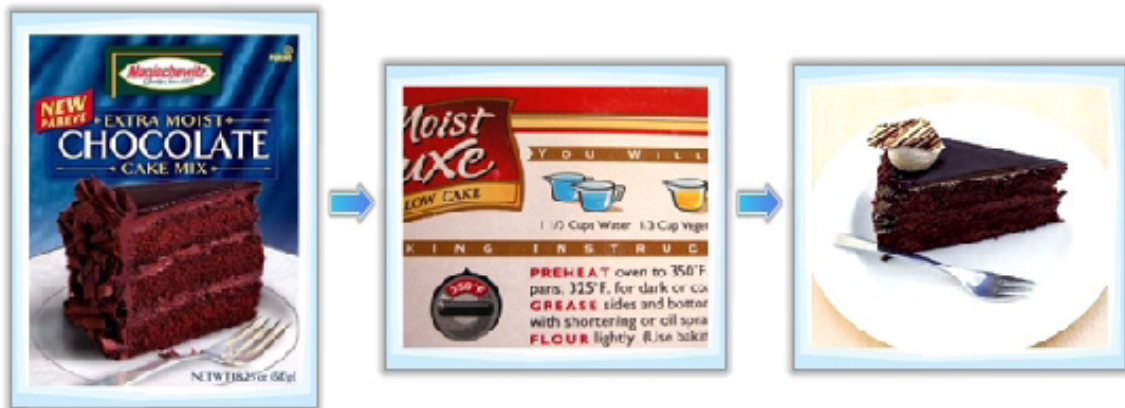


Figure 2.9 is an example of class called RentalCar. In UML, a class is drawn as a rectangle. The rectangle contains two other important features: a list of attributes and series of methods. An attribute describes some property that is possessed by all objects of the class. A method is an action that can be requested from any object of the class. Methods are also called operation. Since this approach sees an information system as a collection of objects interacting with each other, during the analysis phase, it identifies all the types of objects that are useful to the system, and to show how objects are interrelated in implementing a certain job.

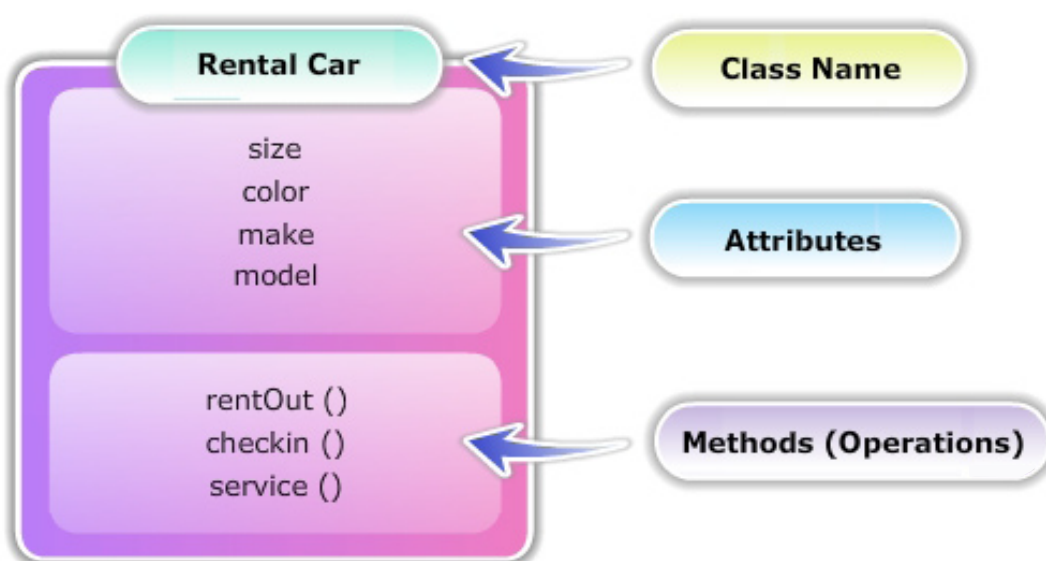


Figure 2.9: An example of UML class.

Source: Adapted from Kendall and Kendall (2008)

Object-oriented analysis (OOA) looks at the problem domain, with the aim of producing a conceptual model of the information that exists in the area being analysed. Analysis models do not consider any implementation constraints that might exist, such as concurrency, distribution, persistence, or how the system is to be built. Implementation constraints are dealt with during object-oriented design (OOD). Analysis is done before the design.

The sources for the analysis can be a written requirements statement, a formal vision document, and interviews with stakeholders or other interested parties. A system may be divided into multiple domains, representing different business, technological, or other areas of interest, each of which are analysed separately.

The result of object-oriented analysis is a description of what the system is functionally required to do, in the form of a conceptual model. That will typically be presented as a set of use cases example shown in Figure 2.10 and one or more UML class diagrams (example in Figure 2.11). It may also include some kind of user interface mock-up.

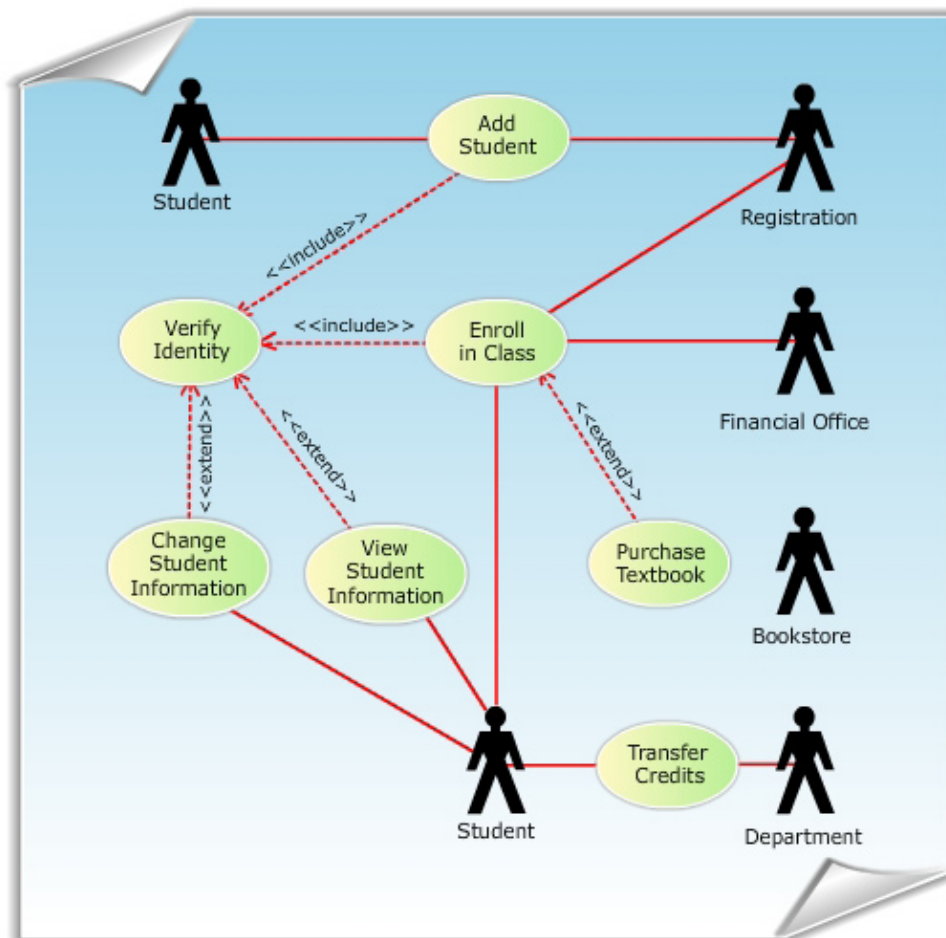
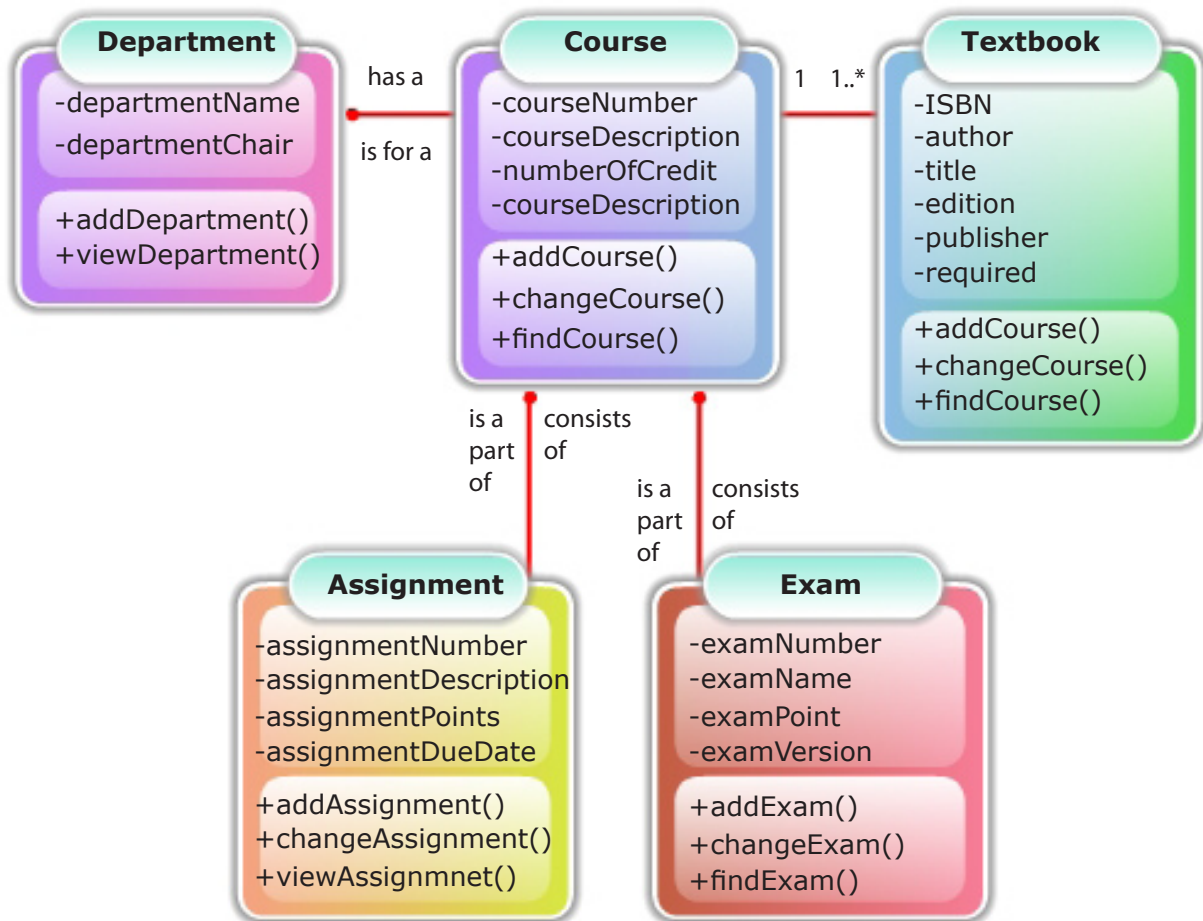


Figure 2.10: Use cases example of student enrolment.
Source: Adapted from Kendall and Kendall (2008)



Object-oriented design (OOD) transforms the conceptual model produced in object-oriented analysis to take into account of the constraints imposed by the chosen architecture and any non-functional, technological or environmental constraints, such as transaction throughput, response time, run-time platform, development environment, or programming language.

The concepts in the analysis model are mapped onto implementation classes and interfaces. The result is a model of the solution domain, a detailed description of how the system is to be built. Finally the goal of OOAD is to make systems elements more reusable, thus improving system quality and productivity of systems analysis and design.

2.2.3 Rapid Application Development

Rapid application development (RAD) is a methodology to radically decrease design and implementation time. It is to develop information systems that promise better and cheaper systems in more rapid deployment. It is used to speed-up activities and processes found inside every phase of system development, such as speeding-up the analysis phase by scheduling intensive meetings among the parties involved for the purpose of information collection. RAD involves iterative development and the construction of prototypes techniques to accelerate software systems development

Using structured techniques the developer first builds preliminary data models and business process models of the business requirements. Prototyping then helps the analyst and users to verify those requirements and to formally refine the data and process models. The cycle of models, then prototypes, then models, then prototypes and so forth, ultimately results in a combined business requirements and technical design statement to be used for constructing new systems. RAD approaches may entail compromises in functionality and performance in exchange for enabling faster development and facilitating application maintenance.

As Figure 2.12 shows the same phases that are followed in the traditional SDLC are also followed in RAD shown in Figure 2.13, but the phases are shortened and combined with each other to produce more streamlined development technique. RAD usually looks at the system being developed in isolation from other systems, thus eliminating the time-consuming activities of coordinating with existing standards and systems during design and development. Notice also that the iteration in the RAD life cycle is limited to the design and development phases. This is where the bulk of the work in a RAD approach takes place. In this way, decisions on a certain matter can be made immediately and rapidly.

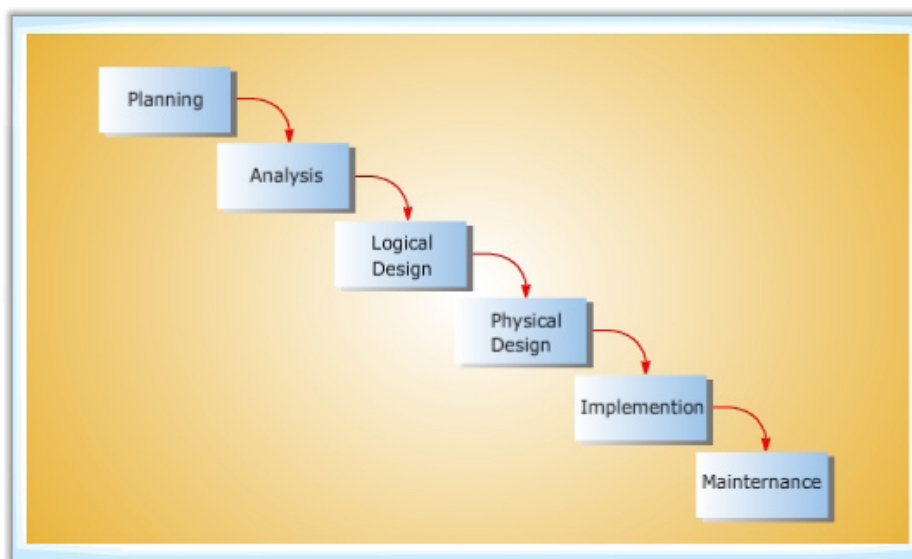


Figure 2.12: Traditional waterfall SDLC

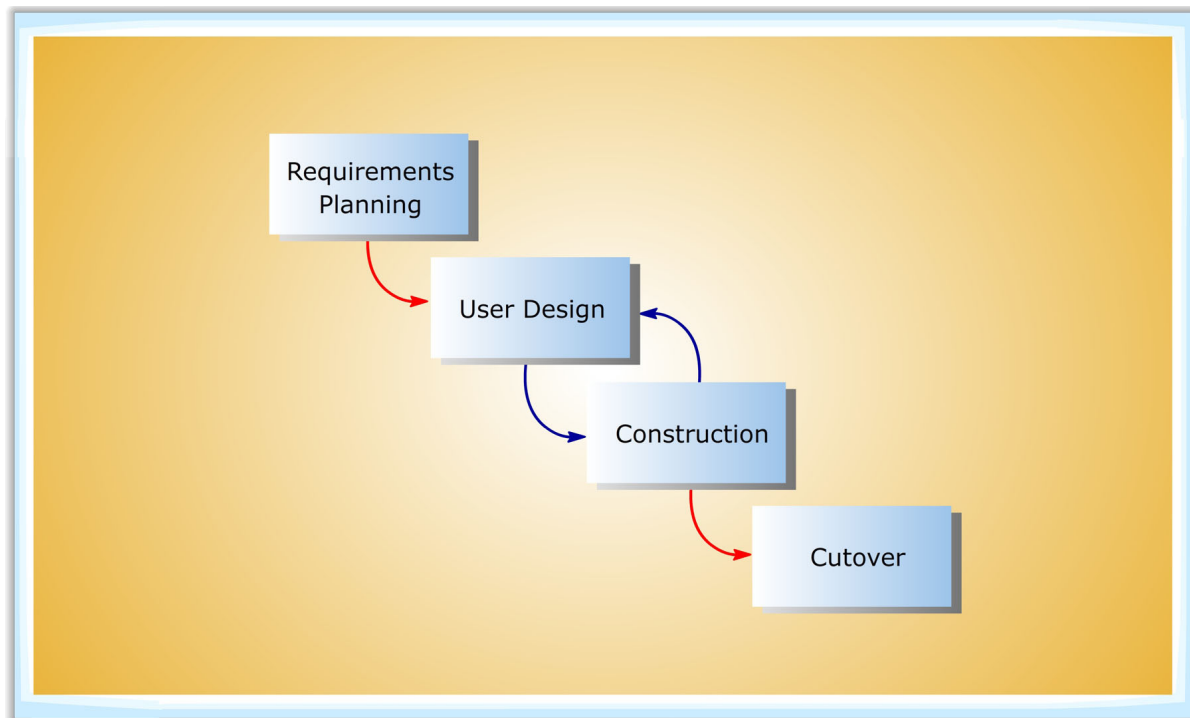


Figure 2.13: RAD lifecycle

Table 2.1 shows the RAD approach offers several advantages and disadvantages.

Table 2.1: The advantages and disadvantages of RAD approach.

Advantages	Disadvantages
It is useful for user requirements that are uncertain or imprecise.	Some argue it may encourage "code, implement, repair" mentality.
Encourages active users and management participation.	RAD prototype can easily solve wrong problem since problem analysis is abbreviated.
Projects get higher visibility and support.	May discourage analysts from considering alternatives.
Stakeholders see working solutions more rapidly.	Stakeholders reluctant to throw away prototype because they see this as a waste of time and effort in the current product.
The iterative approach is a more "natural" process because change is expected during development.	Emphasis on speed can adversely impact quality because of ill-advised shortcuts through the methodology.

2.2.4 Prototyping

A prototype is a small-scale, representative, or working model of the users' requirements or a proposed design for an information system.

A prototype typically simulates only a few aspects of the features of the eventual program, and may be completely different from the eventual implementation. The conventional purpose of a prototype is to allow users of the software to evaluate developers' proposals for the design of the eventual product by actually trying them out, rather than having to interpret and evaluate the design based on descriptions.

A prototype can be built with any computer language or development tool, but special prototyping tools have been developed to simplify the process. A prototype can be developed with visual development tools, with query screen and report design tools of a database management system and with CASE tools. Figure 2.14 depicts a prototype screen for a system.

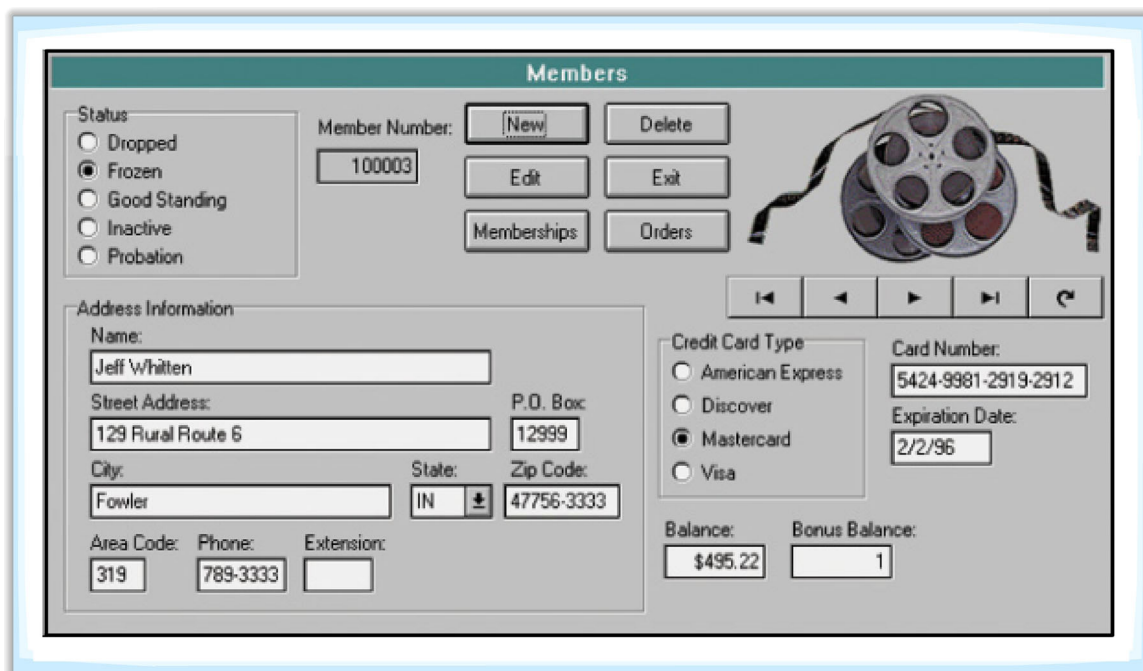


Figure 2.14: Sample prototype screen

Today many analyst and designers prefer prototyping, a modern engineering based approach to design. It is an iterative process involving a close working relationship between designer and the users. This approach has several advantages disadvantages as shown in Table 2.2.

Table 2.1: The advantages and disadvantages of RAD approach.

Advantages	Disadvantages
The software designer and implementer can obtain feedback from the users early in the project.	Prototyping does not negate the need for the system analysis phases. A prototype can solve the wrong problems and opportunities just as easily as a conventionally developed system can.
The client and the contractor can compare if the software made matches the software specification, according to which the software program is built.	You cannot completely substitute any prototype for a paper specification. Yet many information system professionals try to prototype without a specification. Prototyping should be used to complement not to replace other methodologies. The level of detail required of the paper design may be reduced but it is not eliminated.
The client and the contractor can compare if the software made matches the software specification, according to which the software program is built.	During prototyping, the scope and complexity of the system can quickly expand beyond original plans. This can easily get out of control.
It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.	
An approved prototype is a working equivalent to a paper design specification, with one exception; errors can be detected much earlier.	

ACTIVITY



- Why is it important to use methodologies when building a system?
- Why not just build the system in whatever way that appears to be quick and easy?

2.3 SYSTEM DEVELOPMENT LIFE CYCLE

System development life cycle (SDLC) is a phased approach to solving business problems. It is a common methodology for system development in many organisations; it features several phases that mark the progress of the system analysis and design effort. Structured analysis uses this method for planning and managing a system development process.

The SDLC can be thought of as a circular process in which the end of the useful life of one system leads to the beginning of another project that will develop a new version or replace an existing system altogether. For example in any given SDLC phase, the project can return to an earlier phase it necessary. Similarly if a commercial product does not perform well just after its introduction, it may be temporarily removed from the market and improved before being reintroduced. In the SDLC, it is also possible to complete some activities in one phase in parallel with some activities of another phase. Sometimes the life cycle is iterative; that is, phases are repeated as required until an acceptable system is found. The skills required of a systems analyst apply to all life cycle models.

This module follows a generic SDLC model, as illustrated in Figure 2.15. We use this SDLC as an example methodology and a way to think about systems analysis and design. You can apply this methodology to almost any life cycle. As we describe this SDLC throughout the module, it becomes clear that each phase has specific outcomes and deliverables that feed important information to other phases. At the end of each phase (and sometimes within phases for intermediate steps), a systems development project reaches a milestone. Then, as deliverables are produced, they are often reviewed by parties outside the project team, including managers and executives.

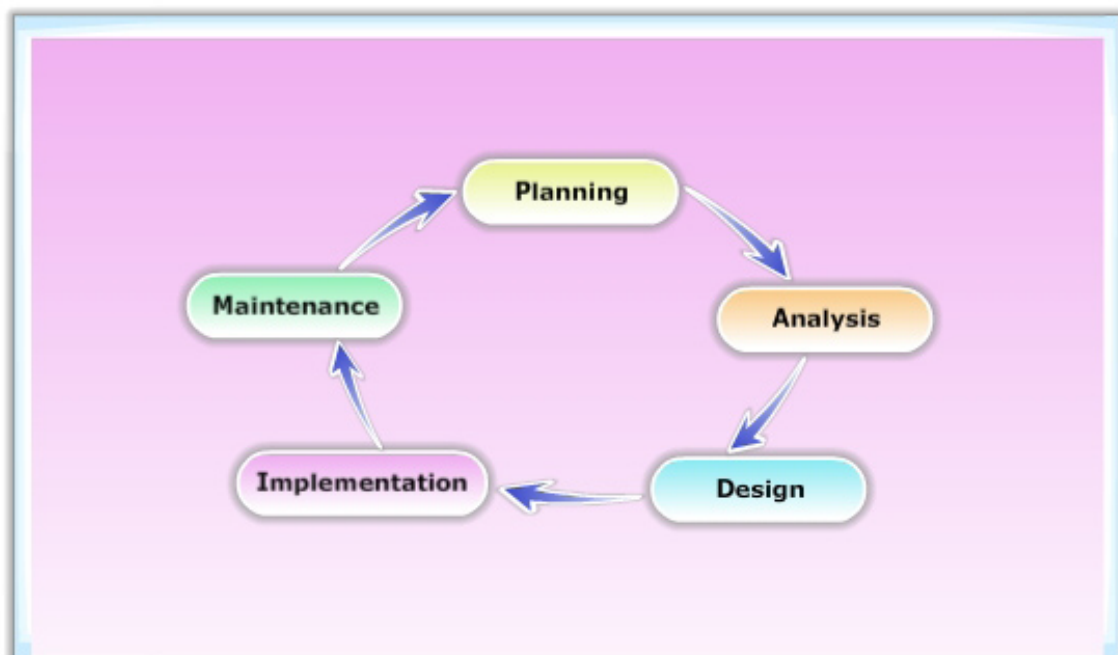


Figure 2.15: System Development Life Cycle.

2.3.1 System Planning

The first phase in the SDLC is system planning. The systems planning phase usually begins with a formal request to the IT department, called a systems request, that describes problems or desired changes in an information system or a business process. In many organisations, IT systems planning is an integral part of overall business planning. When managers and users develop their business plans, they usually include IT requirements that generate systems requests.

A systems request can come from a top manager, a planning team, a department head, or the IT department itself. The request can be very significant or relatively minor. A major request might involve a new information system or the replacement of an existing system that cannot handle current requirements. In contrast, a minor request might ask for a new feature or a change to the user interface.

The people involved in the first phase are users, analysts and systems managers coordinating the project. Activities in system planning phase are shown in Figure 2.16:

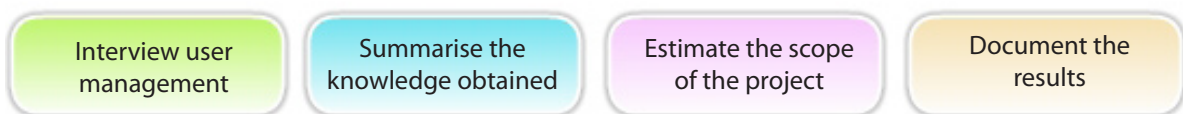


Figure 2.16: Activities in system planning phase

The output of this phase is a feasibility report. Management must then make decision whether to proceed with the proposed project. If the user group does not have sufficient funds or wishes not to tackle unrelated problems, the system project does not proceed any further. If the development process continues, the next step is the systems analysis phase.

2.3.2 System Analysis

The second phase in the SDLC is system analysis. During this phase, the analyst thoroughly studies the organisation's current procedures and the information systems used to perform organisational tasks. The main objective of this phase is to understand the business requirement and develop a logical model for the new system. The steps, which are required in phase 2, are shown in Figure 2.17:

Steps in System Analysis

1. Requirements modelling, where you investigate business processes and document what the new system must do. The information for developing this model can be obtained through information gathering techniques such as interviews, survey, document review, research, observation and sampling.
2. Development of a logical model of the business process, which includes the data model, the process model and the object model, for which the system that is to be developed should be able to support the process explained.

Figure 2.17: The steps in System Analysis.

The end product for the systems analysis phase is the system requirements document. The system requirements document describes management and user requirements, costs and benefits, alternative development strategies and proposed solution from you, as an information system expert.

2.3.3**System Design**

The third phase in the SDLC is system design. The purpose of the systems design phase is to create a blueprint that will satisfy all documented requirements for the system. At this stage, the user interface is designed and all necessary outputs, inputs, and processes are identified. In addition, internal and external controls are designed, including computer-based and manual features to guarantee that the system will be reliable, accurate, maintainable, and secure. During the systems design phase, the application architecture is also determined, which shows programmers how to transform the logical design into program modules and code.

The result of this phase is documented in the system design specification and presented to management and users for review and approval. Management and user involvement is critical to avoid any misunderstanding about what the new system will do, how it will do it, and what it will cost.

2.3.4 System Implementation

The fourth phase in the SDLC is system implementation. In this phase, the new system is constructed. Whether the developers used structured analysis or OOA methods, the procedure is the same, programs are written, tested and documented and the system is installed. If the system was purchased as a package, systems analysts configure the software and perform any necessary modifications. The objective of the systems implementation phase is to bring the system into reality by deliver a completely functioning and documented information system.

Programmers have a key role in this phase because they design, code and remove syntactical errors from computer programs. To ensure quality, a programmer may conduct either a design or a code walkthrough, explaining complex proportions of the program to a team of other programmers.

At the conclusion of this phase, the system is ready for use. Final preparations include converting data to the new system's files, training users, and performing the actual transition to the new system. The systems implementation phase also includes an assessment, called a systems evaluation, to determine whether the system operates properly and if costs and benefits are within expectations.

2.3.5 System Maintenance

The fifth and final phase in SDLC is system maintenance. The main objective of this phase is to maximise returns on investment. A system that is well designed will become a reliable system, easy to be updated and measured. A design, which can be measured, can be expanded easily to accommodate new requirements and even to add new branches to the business.

In this phase, the information technology staff will update and add value to the new system. Updates will involve correction of errors, if any, and to make changes that are appropriate with the environment at the location used, such as including value-added-tax for the tax rate. Meanwhile, activities to add values to the system just developed can be things such as adding new features and benefits. Information systems development is always a work in progress. Business processes change rapidly and most information systems need to be updated significantly or replaced after several years of operation.

In summary, maintenance is an ongoing process over the life cycle of an information system. After the information system is installed, maintenance usually takes the form of correcting previously undetected program errors. As time goes on and the business and technology change, however the maintenance effort increases dramatically.



List and explain the different phases in the SDLC.

SUMMARY

- Information system can be viewed as an organisational resource just as humans are thus it must be managed carefully, just as other resources are.
- It is very important to understand various methodologies of system development and the phases involved in system development to develop a system in a more professional and systematic manner.
- System analysis and design is a systematic approach in handling information system development such as system development life cycle (SDLC).
- SDLC can be divided into five major phases; planning, analysis, design, implementation and maintenance.
- Object-oriented analysis and design are based on object oriented programming concepts in which objects that are created not only code about data but also instructions about the operations to be performed on the data.
- Rapid Application Development (RAD) is a methodology to radically decrease design and implementation time.
- Prototyping is a small-scale, representative, or working model of the users' requirements or a proposed design for an information system.

KEY TERMS

Business model

Data model

Methodology

Model

Object

Object model

Object-orient analysis and design

Process model

Prototype

Rapid application development

Structured analysis

System development life cycle

Technique

Tools

REFERENCES

1. Kendall, E.K. and Kendall E. J (2008) 7th ed., Systems Analysis and Design, Upper Saddle River, NJ: Prentice Hall
2. Kay, R. 2002. "QuickStudy: System Development Life Cycle", May 14. Available at www.computerworld.com. Accessed 30 March 2009
3. Kroll, P. 2004 "Transitioning from waterfall to iterative development", Apr 16. Available at www.rational.com. Accessed 30 March 2009
4. Gane, Chris. (1989). Rapid System Development. Englewood Cliffs, NJ: Prentice Hall