Overview

The second unit of the course B329 Systems Analysis and Design is organized into six sections, of which this overview is the first.

The second section introduces the concepts of systems development life cycle (SDLC) and methodology. After the introduction and list of unit objectives, it describes the business process used to develop information systems.

The third section relates to the underlying principles of systems development. The success of almost all IS depends on these principles.

The fourth section covers the seven phases involved in a typical SDLC.

Systems development life cycle also involves several cross life cycle activities. These activities are described in section five.

The sixth section explains how Computer Aided Systems Engineering (CASE) tools can be used in different stages of SDLC. It also provides a framework and an architecture for a typical CASE tool.

The conclusion contains a summary of Unit 2, a case study, a self-test, references, a glossary of terms, and answers to the self-test questions.
Introduction

The development of an information system is a complex undertaking. The process involves a variety of stakeholders, each responsible for different aspects of the development effort. It also involves numerous interrelated activities and tasks. Because of the complexities involved in a system life cycle, it is helpful to understand the logical steps associated with various phases in the life cycle. It is also important to know how to use a methodology to physically implement the life cycle, taking into account a gamut of variables related to the activities in each phase, tools and techniques to be used for each activities, and people and groups involved in different activities.

Because modelling business processes and organizational information flow is an involved task, it is helpful to use specialized software to facilitate systems and process modelling. Therefore, it is important to understand the inner working of these software, generically called CASE tools.

You should note that some of the activities involved the system analysis and design process are sequential, while others cross the life cycle. The success of a development project nonetheless depends on the effective management of all phases of the development process. Topics related to project and process management will be covered in more detail in Unit 3 of the course.
Objectives

By the end of Unit 2, you should be able to:

1. Define systems development life cycle and methodology.
2. Describe the basic principles of systems development.
3. Describe the phases involved in a modern systems development methodology.
4. Describe cross life activities involved in the development process.
5. Discuss the use of typical CASE technology in analysis and design.
6. Describe the components of a CASE tool framework.
7. Explain how to evaluate CASE tools for adoption.
8. Explain the benefits of adopting CASE technology.
Systems development life cycles and methodologies

A systems development life cycle (SDLC) is a management tool to plan, execute and control various activities involved in the systems development process. More specifically, it is a tool for managing complex processes by breaking them down into a series of phases and activities.

There is a subtle difference between an SDLC and a methodology. Whereas SDLC relates to logical process through which the systems’ stakeholders try to solve business problems via building computer-based IS, a methodology relates to the physical implementation of the SDLC. It outlines step-by-step activities for each phase of SDLC, the role that stakeholders play in each activity and the deliverables expected from each activity. A methodology also defines the tools and techniques to be used for each activity.

As new IS under development get larger and more complex, there is a greater need to rely on formal management tools and technique to oversee the development life cycle. The factors that increase the riskiness of IS projects will be discussed in Unit 3.

Reading

‘The process of systems development’, pp. 75–84 in your text.

This reading defines systems development life cycle and methodology. You should relate the key ideas to Objective 1 of Unit 2.
Principles of systems development

Systems analysis and design is both a science and an art, similar to some engineering disciplines or architecture. We use certain generally-accepted principles to build systems. At the same time we use past experiences to continuously readjust these principles. Over the years, several of these principles have emerged to be somewhat universal. The success of almost all IS development projects depends on these underlying principles. Although some of these principles appear simple and to be common sense, disregarding them would invariably lead to a project’s failure. In total, there are eight commandments of systems development:

1. **Get the stakeholders involved.** Empirical research has shown that the user and management involvement improves the degree of systems success. As systems analysts you should be particularly careful not to take a purely ‘technical’ view of systems, which is a sure recipe for failure.

2. **Use a problem-solving approach.** As mentioned previously, design of systems is usually a complex task, which demands a systematic, well-thought out approach. First, you need to ask the right questions in order to understand the problem under study. Second, you need to be able to determine the requirements of a suitable solution. Third, you should identify the best solution among several alternatives. Fourth, you need to design and implement the solution. Finally, you should evaluate the impact of a system after its implementation with a view to refining the solution.

3. **Breakdown phases.** In the next section we will introduce a comprehensive SDLC. However, there are simpler forms of life cycles that can help you break down the major phases in the life cycle. For example you can divide a system development project into planning, analysis, design and implementation phases. Regardless of the methodology you use, breaking down the major phases of a project allows you manage various parts of the project effectively.

4. **Establish standards.** Organizations own and manage numerous information systems. It is extremely important to establish standards that help describe the activities and responsibilities associated with each phase of a systems development, outline guidelines for documentation and provide quality checks.

5. **Justify systems as capital investments.** From an accounting point of view, systems can be treated as expense or as capital investment. Whereas expenses have a short, usually one-year, accounting life, capital investments can be depreciated over longer periods of time. As such, capital investments require continual management control. By treating IS as capital investment you make sure that the systems projects are treated as investments that need to be managed cost-effectively over a long period of time.

6. **Re-evaluate the feasibility and scope of the project periodically.** IS projects are notorious in cost overrun and schedule delays. One reason is that the scope of projects usually change over time, but little effort is
made to re-evaluate their cost and schedule. Periodical re-evaluation of the feasibility and scope of an IS project allow the analysts to draw periodically a picture of the project’s status, and if necessary abort the project.

7 **Think of system integration.** Most IS need to be linked to other systems. When developing a new system, it is important to account for the effort required to make the necessary system integration, i.e., work with the existing systems.

8 **Think of growth and change.** By definition, business requirements are dynamic and as such, they change. Because IS are designed according to business requirements at a given period of time, they are subject to **entropy**. Entropy refers to the natural decay of systems over time. In other words, if left alone, IS will be unable to respond to changes in their environment and will eventually decay, rendering them useless to an organization. Therefore it is important to plan for future growth and change and design systems that are flexible and adaptable. Bear in mind that IS should be designed as ‘living’ systems, whose life starts after they become operational. During the system operation stage, the information system will require continual enhancement (improvement and further development) in order to meet constantly changing business conditions. As such, do not treat the IS development cycle as an end to itself, but rather as an ongoing process that helps the firm achieve one or more of its objectives.

**Reading**

‘Underlying principles of systems development’ pp. 79–84 in your text.

This reading introduces some general principles that should underlie all systems development methodologies. Some of these principles may seem to be common sense, but are ignored by many inexperienced systems analysts, leading to system failure. You should relate the key ideas to Objective 2 of Unit 2.
Phases in a modern systems development life cycle

The major building blocks of a system development life cycle are tasks, activities and phases. The smallest unit is a task, which represents a unit of work that can be assigned to an individual. Tasks are grouped into activities. Activities are grouped into phases. Activities consist of a set of logically related tasks designed to fulfill a well-defined, general objective. Activities may overlap within a phase. Phases do not overlap. Phases represent points in the development process at which management can make a decision to go ahead or abort the project.

SDLC has many advantages, among which are:

- A phased approach that encourages the specification of the requirements before the design of the system and the design of the system before the coding of the programs.
- Formal definitions of the phases and a work breakdown structure that result in a defined series of processes that ensure a consistent approach to the development of each and every system.
- Formal definitions of the deliverables of each phase that lead to generation of a series of documents which can be used later in testing and maintenance of the system.
- Identifiable decision points that provide management control of the development process (the decision points or checkpoints are identified by the defined phases and related deliverables.)
- Management controls that can be applied in planning, estimating, tracking and controlling the development process.

As mentioned before, there are many methodologies that can be used in the IS development process. Remember that the idea behind using a methodology is to facilitate the management of various phases and activities involved in the development cycle. In this course we will use an eight-phase methodology called FAST (Framework for the Application of Systems Techniques), phases of which are briefly described below:

1. **The preliminary investigation phase.** Also called survey or feasibility study, in the survey phase we first try to establish whether or not a project is feasible. Then we determine the budget, schedule and responsibilities of the project team. The main deliverable of this phase is a feasibility statement, outlining whether an existing system needs to be enhanced or a new system needs to be developed. If a new system is recommended then a statement of project and system scope is prepared and carried over to the next phase. The details of this phase will be discussed in Unit 8.

2. **The problem analysis phase.** In this phase, we try to understand the problem under study more thoroughly. We also need to determine whether the project is worth undertaking. The deliverable generated from this phase is a detailed study report, which includes system objectives. The details of this phase will be discussed in Unit 4 and Unit 8.
3 **The requirements analysis phase.** Also called definition phase, in this phase the analyst determines the users’ needs. Because misunderstanding users’ requirements will lead to user dissatisfaction, it is very important to spend the efforts required to outline users’ requirements. It is customary to use **modelling** techniques to document and validate users’ requirements. The product of this phase is a business requirement statement. The details of this phase will be discussed in Units 4-7.

4 **The decision analysis phase.** This phase is to identify candidate solutions, analyse those solutions for feasibility, and recommend a candidate system as the target one to be designed.

5 **The design phase.** In this phase we translate the business requirements generated in the definition phase into a design blueprint. In recent years many companies have used the **rapid application development (RAD)** approach to build a series of **prototypes**. The deliverables of this phase are the structure of files and databases, user interfaces, networks and software. The details of this phase will be discussed in Units 9-13.

6 **The construction phase.** In this phase we use rapid application development to construct and test the systems components. We also need to construct the interfaces that are required between the new systems and the existing systems. The deliverables of this phase are several prototypes. The details of this phase will be discussed in Units 9-13.

7 **The implementation phase.** Also called delivery, in this phase the analyst needs to decide the best way to move from the old system to the new one. This process requires preparing various manuals, training of the users and loading of the files and databases. The key deliverable of this phase is the production system to the users. The details of this phase will be discussed in Unit 14.

8 **The operation and support stage.** Once the system is operating, it delivers the business solution to the user community. It will still require ongoing system support for its life.

---

**Reading**

‘A system development methodology’, pp. 84–92 in your text.

This reading introduces the phases involved in SDLC. You should relate the key ideas to Objective 3 of Unit 2.
SELF-TEST 2.1

MATCHING

Match the explanation of the phase with the appropriate phase:

A  Preliminary Investigation Phase  
B  Problem Analysis Phase  
C  Requirements Analysis Phase  
D  Decision Analysis Phase  
E  Design Phase  
F  Construction Phase  
G  Implementation Phase  
H  Operation and Support Stage

1  identifies and analyzes both the business and technical problem domains for specific problems, causes, and effects

2  identifies and analyses candidate technical solutions that might solve the problem and fulfill the business requirements.

3  establishes the project context, scope, budget, staffing, and schedule.

4  identifies and analyzes business requirements that should apply to any possible technical solution to the problems

5  delivers the business solutions to end-users.

6  specifies the technical requirements for the target solution

7  decides the best way to move from the old system to the new one

8  builds and tests the actual solution (or interim prototypes of the solution)
Cross life activities

There are several cross life activities that span more than one phase of the development cycle. These activities are briefly described below:

1. **Fact-finding.** Also called information gathering, refers to activities related to collection of information about the system and its requirements. Among techniques used in fact-finding are interviews and questionnaires. Fact-finding spans almost the entire life cycle with heavier emphasis on the earlier phases.

2. **Documentation.** One of the most important activities related to systems development, documentation spans the entire life cycle. As mentioned previously, IS are designed as living systems that need to be maintained continuously, and grow and adapt periodically. In this light, it is important to document facts and specifications of a system to be used in the future.

3. **Estimation.** Historically, one of the problems with a significant portion of IS development projects has been underestimation of the required cost and time. In order to lessen the adverse effects of such miscalculations, it is helpful to use software and systems metrics to periodically revise and modify the estimation of various activities throughout the project.

4. **Feasibility analysis.** One of the most crucial questions facing IS professionals is the justification of systems. Feasibility analysis is a method that allows measurement of a system’s value. Because the scope of a system changes over time, systems analysts need to perform feasibility analysis throughout the development life cycle in order to minimize the effect of time and cost underestimation.

5. **Project and process management.** So far we have reiterated the importance of the non-technical aspects of IS projects. Many IS projects fail due to unfamiliarity of systems analysts with management tools and principles required to control and manage a diverse set of tasks and activities. We will discuss these issues in more details in Unit 3, which is entirely dedicated to this important subject.

---

**Reading**

‘Cross life cycle activities’, pp. 92–94 in your text.

This reading introduces the details of cross life activities that span different phases of the systems development process. For each phase you need to understand the purpose and the roles. You should relate the key ideas to Objective 4 of Unit 2.
Self-test 2.2

This self-test allows you to assess for yourself whether or not you have met the objectives for this section. You should attempt to answer all self-test questions before looking at the answers.

1. Discuss the overall purpose of an SDLC. What are the three major building blocks that make up an SDLC and how do they relate to one another?

2. List the eight phases of the FAST methodology. What is the main purpose and key deliverables of each phase?

3. What is the main purpose of prototyping? How does prototyping contribute to user involvement?

4. What are the cross life activities? What are the main purposes of each?

5. What are the three main functions of project management? What are the characteristics of each?
CASE tools

Information systems analysts and system developers have available to them a myriad of computer-based tools to assist in various aspects of IS development and maintenance. These software tools are commonly referred to as CASE - Computer Aided Systems Engineering (or Computer Aided Software Engineering). CASE tools generally help in two ways: 1) automate or accelerate the life cycle development process, and 2) improve the overall quality of the system under development. Quality improvements come in the form of rigorous methodologies supported by the CASE tools yielding an ‘engineering’ approach to building information systems as opposed to a loosely controlled approach.

In terms of specific benefits, preparation of systems documentation is the most obvious use of CASE tools, as documentations are required throughout the entire life cycle. Some of these documentation benefits of CASE tools include the following capabilities:

- **Graphics functions** to help develop systems models, e.g., Data Flow Diagrams. Using these functions, you can draw diagrams, print them and modify them without having to recreate an entire drawing.

- **A data dictionary template** that prompts you for standard information about data structures, data flows, data stores and data elements.

- **A report generator** that allows you to create report and screen layouts.

A prime advantage of using even basic computer facilities for analysis and design work is that you can modify your model easily. Remember, systems work is iterative and your models are very likely to change as you develop them. To be able to make those changes easily, without redoing parts that need be not changed, speeds your work enormously.

Since processes and data are modelled and communicated across all phases of the life cycle, it is important to maintain logical consistency throughout. Thus the data dictionary or CASE repository (discussed in a later subsection) serves this purpose. For example, if a data store changes, it may affect related data flows, processes and the overall data model. The need to maintain consistency and detect missing elements have become a basic feature of most CASE facilities today.

**Reading**


This section of your text provides an overview of CASE tools including some historical background.
CASE tool classification

CASE tools are generally classified into two categories, corresponding to the stages in the life cycle.

**Upper-CASE** tools address the early phases of systems analysis and design. They include modelling, analysis and some design tools. Examples of tools in this category include process modellers (Data Flow Diagram (DFD) or other methodology), documentation and project management tools.

**Lower-CASE** tools support the latter phases of the systems development life cycle. Specifically, they can address the detail design, construction, delivery and support aspects. Examples include database designers, graphical user-interface designers, automated testing programs, version management tools and help-desk tools.

It is important to note that the separation between upper- and lower-CASE tools is not well defined. CASE tool vendors provide a whole range of products that can span across one or both categories. An analyst typically will use a number of CASE tools within the life cycle, thus it is important to evaluate and select appropriate tools.

CASE repository

With the popularity of the CASE tools, many products have proliferated the market. As a result, a problem exists with making them work with each other. For example, it is important for an upper-CASE tool, such as a DFD diagrammer and analyser, to work with a lower-CASE database design tool because the information on the data elements defined in the DFD should be the same for the database design tool. Rather than re-entering the same information back into a second tool, a solution found in many modern CASE tools today incorporate a developer’s database (CASE repository). What this CASE repository does is to store, integrate and share the models, definitions and data declared for an information system across CASE tools. Thus, after modelling processes and data flows in DFD, the models and its corresponding data will be automatically updated to the CASE repository. When the database designer develops the database, the CASE tool for database design can download the appropriate data from the CASE repository.

Reading


Your textbook provides greater details on the various types of CASE tools available. Figure 3.14 shows the CASE tool architecture incorporating a repository.
CASE tool evaluation

There are a number of integrated CASE tools that span the life cycle. Unfortunately, such tools have not been popular because they typically trade off depth of functionality against flexibility. Thus it is important for an analyst and project manager to select an appropriate mixture of CASE tools that will work together.

To be sure of proper selection, the following is a list of criteria that can be used to assess CASE tools:

**Capability.** The very basic criterion for assessing a CASE tool is its suitability to the task. Simply put, features and costs aside, if a tool cannot do the job, it is of no use.

**Rating/Performance.** CASE tools, like other products, must be compared and assessed. The comparison can be in terms of features, performance, cost, etc. An analyst or project manager should review trade journals that perform assessment and comparison on CASE tool products. Such published reviews should not be adhered to strictly, but rather used as a starting point to narrow down choices and take the next step in obtaining trials of the CASE tool. Trials can be obtained from sample versions and visiting the local software distributor for a demonstration.

**Compatibility.** Compatibility in terms of platform and methodology need to be considered. With many information systems spanning multiple hardware and operating systems, it is important to ensure compatibility of CASE tools.

**Tool integration.** It is likely that no one tool may do the job for the whole systems development life cycle. Thus, selecting an appropriate suite of tools that can work together or share a common CASE repository should be considered. Otherwise making multiple CASE tools to interoperate will become a major effort.

**Reputation.** CASE tool products easily number in the thousands with each year bringing in new vendors and products. It is thus important to consider the reputation and the rating of the company and the tool that is being considered. Many stories abound of companies adopting a CASE tool only to find that the company has gone out of business and the existing data cannot be imported into other environments.

**Technical support.** CASE tools and methodologies, while helping the development process, can in themselves be complicated to work with. It is important to see that there are sufficient training tools, classes and technical support for products, and also continual upgrades of products.

**Cost.** CASE tools vary greatly in price. It is more important to consider needs and functionalities, and not just the cost alone. It has been well accepted that the maintenance costs for an IS typically surpass the original development costs over its life cycle. Thus, prioritizing IS development needs and some basic cost/benefit analysis are important in the long-term success of an IS project.
Activity
To get a good sense of the variety of CASE tools available on the market, perform an Internet search using your favourite WWW search engine. Enter the keyword ‘CASE tool’ plus any other modifier to narrow your search focus. You are likely to find hundreds of various tools and vendors.

Benefits
The systems analyst and designer are typically free in adopting (if not required by management!) CASE tools for their use in IS development projects. There are a number of advantages related to CASE tools, among which include:

• Improved productivity in using software to automate some of the tedious or detailed processes. Productivity is further improved by the incorporation of rapid application techniques including user-interface builders and code generators.

• Improved quality in automating analysis of the design and testing of the development code. Consistency checking capabilities also improves early detection of errors.

• Better documentation by means of a tighter (unambiguous) specification requirements and the structuring of information in the repository.

• Reduced lifetime maintenance through a higher quality from the enforcement of a uniform standard and a well-documented system.

• A more precise framework for modelling data, knowledge, system and constraints. By enforcing a strict discipline in detailed specifications, the CASE tool can incorporate some consistency checking algorithms to verify details of the models, thus ensuring its correctness.

Self-test 2.3
This self-test allows you to assess for yourself whether or not you have met the objectives for this section. You should attempt to answer all self-test questions before looking at the answers.

1 What is a CASE tool?

2 Why should you consider using CASE tools?

3 If you were asked to select among a number of CASE tools for a new IS development effort, what are the steps you would take to decide?

4 What is the difference between upper-CASE and lower-CASE?
Once you have attempted to answer these self-test questions, you can refer to the answer key at the end of Unit 2. Now turn back to objectives 1 to 8 of Unit 2. Do you feel you have met them? If not, turn back to the topic which is associated with that objective. Read through it again, then recheck your understanding.
A note on System Architect

System Architect by Popkin Software and System has been specified for use in this course primarily to assist you with your assignments which require modelling. You should remember that System Architect is representative of an entry-level CASE tool and lacks the full features of integrated CASE products that are used in industry. This does not mean that you will not benefit from using System Architect. However, you should not expect it to provide the major benefits which industry can obtain from the use of an integrated CASE product. Integrated CASE tools used in industry often cost in excess of $100,000 for each workstation.

ABC Snapgraphics v1.0 by Micrographx is an alternative drawing tool that is also recommended for preparing process diagrams (e.g., DFDs) in assignments. Though not a full-fledged CASE tool, it is sufficient for developing good diagrams. This software is freeware and thus is available for all to use. A special template for DFDs for use with ABC Snapgraphics also will be made available. It is available for download from the OLE server.
Summary

You should now understand the context of systems analysis and design within the SDLC. You should also know how the SDLC provides a framework for the management control of systems development. This framework is used throughout the course so that the tools and techniques can be related to one another.

Review the summary on page 111 in your textbook. Do you feel you understand the ‘Key terms’ on page 113? If not, you should review the relevant topic.

In the next unit of this course, Unit 3, you will explore project and process management.
References

B329 Systems Analysis and Design is self-instructional. Everything you need is included. You should be able to meet the learning objectives and successfully complete your assessment activities on the basis of your study units and your set textbook. The following references are not prescribed reading. However, they may be helpful if you want to further your study of the topics presented in this unit.

Systems analysis and design


Software engineering

Glossary

This glossary provides brief definitions of the main technical terms you encountered in Unit 2. You should also refer to the glossary in your textbook on pages 692–724. However, you should note that the textbook glossary covers the entire course and contains many terms that you will meet later on. The definitions given here are offered in addition to, or in place of, those in your textbook glossary. They do not correspond word for word with the definitions in the textbook, but neither do they contradict them.

Activities: Activities consist of a set of logically related tasks designed to fulfil a well-defined, general objective. Activities may overlap within a phase. Phases do not overlap.

Architecture: The design and the structure of a set of technologies.

CASE repository: A shared database for IS development that allows for the sharing of data across CASE tools.

CASE tools: Computer software that supports the life cycle development of information systems.

CASE stands for Computer-Aided System Engineering or Computer-Aided Software Engineering.

Computer-Aided Systems Engineering (CASE): The application of information technology to systems development activities, techniques and methodologies. CASE tools are programs (software) that automate or support one or more phases of a systems development life cycle. The technology is intended to accelerate the process of developing systems and to improve the quality of the resulting systems.

Configuration phase: The configuration phase identifies and analyses candidate technical solutions that might solve the problem and fulfil the business requirements. The result is a feasible application architecture.

Construction phase: The construction phase builds and tests the actual solution by using rapid application development.

Cross life cycle activities: Activities that overlap many or all phases of the methodology. In fact, they are normally performed in conjunction with several phases of the methodology.

Data dictionary template: Part of a CASE tool that prompts you for standard information about data structures, data flows, data stores and data elements.

Definition phase: Also called requirements analysis, in this phase the analyst determines the users’ needs. The definition phase identifies and analyses business requirements that should apply any possible technical solution to the problems.

Deliverables: Deliverables refer to the outputs expected from each of a phase.
**Design phase:** The design phase specifies the technical requirements for the target solution.

**Documentation:** The activity of recording facts and specifications for a system.

**Entropy:** The natural decay of systems over time.

**Estimation:** The activity of approximating the time, effort, costs and benefits of developing systems. The term guesstimation (as in 'make a guess') is used to describe the same activity in the absence of reliable data.

**Fact-finding:** Also called information gathering or data collection. The formal process of using research, interviews, meetings, questionnaires, sampling and other techniques to collect information about systems, requirements and preferences.

**Feasibility analysis:** The activity by which feasibility is measured.

**Framework:** Refers to the fundamental building blocks of a concept.

**Help-desk tools:** These are lower-CASE tools related to the support and maintenance phase of an implemented application. End-users with problems associated with software can call support personnel who will use these tools to register the problem and search for appropriate fixes to them. Reports by these tools can help management in future decision-making on maintenance activities.

**Implementation phase:** Also called delivery, in this phase the analyst needs to decide the best way to move from the old system to the new one. The delivery phase puts the solution into daily production.

**Interviews:** Fact-finding technique whereby the systems analyst collects information from individuals face-to-face.

**Lower-CASE:** CASE tools associated with the early phases of the IS life cycle. This includes tools for project management, documentation and system modelling.

**Measurement:** The activity of measuring and analysing developer productivity and quality (and sometimes costs).

**Methodology:** The physical implementation of the logical life cycle that incorporates (1) step-by-step activities for each phase, (2) individual and group roles to be played in each activity, (3) deliverables and quality standards for each activity, (4) tools and techniques to be used for each activity.

**Phases:** Phases represent points in the development process at which management makes a decision to proceed with or abort a project.

**Principles of systems development:** The principles of systems development are: get the owners involved; use a problem-solving approach; establish phases and activities; establish standards for consistent development and documentation; justify systems as capital investments; do not be afraid to cancel; divide and conquer; design systems for growth and change.
**Process management:** An ongoing activities that establishes standards for activities, methods, tools and deliverables of the life cycle.

**Project management:** The ongoing activities by which an analyst plans, delegates, directs and controls the progress to develop an acceptable system within the allotted time and budget.

**Prototypes:** A small-scale, representative or working model of users' requirements.

**Prototyping:** The act of building a small-scale, representative or working model of the users' requirements to discover or verify those requirements.

**Purchase phase:** Also called procurement, this phase relates to purchase of any required technology, including both hardware and software.

**Questionnaires:** Special-purpose documents that allow the analyst to collect information and opinions from respondents.

**Rapid application development (RAD):** The merger of various structured techniques (especially the data-driven information engineering) with prototyping techniques and joint application development techniques to accelerate systems development.

**Report generator:** Part of a CASE tool that allows you to create report and screen layouts.

**Several cross life activities:** Several cross life cycle activities include fact-finding, documentation and presentation, estimation and measurement, feasibility analysis, project management and process management.

**Software and system metrics:** Provides an encyclopaedia of techniques and tools that can both simplify the estimation process and provide a statistical database of estimates versus performance.

**Survey phase:** Also called preliminary investigation or feasibility study. It is during this phase that the project context, scope, budget, staffing and schedule are established.

**Study phase:** The study phase identifies and analyses both the business and technical problem domains for specific problems, causes and effects.

**Systems development life cycle (SDLC):** A logical process by which systems analysts, software engineers, programmers and end-users build information systems and computer applications to solve business problems and needs. It is sometimes called application development life cycle.

**Task:** The smallest unit of work that can be assigned to an individual. Tasks are grouped into activities.

**Upper-CASE:** CASE tools associated with the latter phases of the IS life cycle. This includes tools for user-interface design, database design and software testing.

**Version management tools:** Lower-CASE tools that support the development of multiple-versions of software applications. Consider the development of
an order-entry program for data entry clerks, and one for a customer-service manager; both systems will share a common base of programming source code but will have different features. Version management software allows for managing such development needs.
Answer key for self-test questions

Self-test 2.1

B 1 identifies and analyzes both the business and technical problem domains for specific problems, causes, and effects

D 2 identifies and analyses candidate technical solutions that might solve the problem and fulfill the business requirements.

A 3 establishes the project context, scope, budget, staffing, and schedule.

C 4 identifies and analyzes business requirements that should apply to any possible technical solution to the problems

H 5 delivers the business solutions to end-users

E 6 specifies the technical requirements for the target solution

G 7 decides the best way to move from the old system to the new one

F 8 builds and tests the actual solution (or interim prototypes of the solution)

Self-test 2.2

1 The purpose of an SDLC is to provide a system that will meet the business needs, is flexible and maintainable, has integrity, is reliable and is developed within budget and on time. An SDLC provides a methodology to help build such a system and to organize the literally thousands of different work assignments and end-products that must be integrated.

The smallest unit is a task. A task represents a unit of work that can be assigned to an individual and that takes less than 40 working hours to complete. Tasks are grouped into activities. Activities are grouped into phases. Activities consist of a set of logically related tasks designed to fulfill a well-defined, general objective. Activities may overlap within a phase. Phases do not overlap. Phases represent points in the development process at which management makes a decision to continue or abort a project.

2 The eight phases are: the preliminary investigation phase, the problem analysis phase, the requirements analysis phase, the decision analysis phase, the design phase, the construction phase, the implementation phase and the operation and support stage.

The purpose of the preliminary investigation phase is to define the project scope and the perceived problems, opportunities and directives that triggered the project. The project team and participants, the project
budget and the project schedule must also be established in this phase. The key deliverables of this phase are feasibility assessment, project plan and the statement of project and system scope.

The purpose of the problem analysis phase is to identify and analyse business and technical problem domains for specific problems, causes and effects. The key deliverables of this phase are feasibility analysis and a business problem statement which includes system objectives.

The purpose of the requirements analysis phase is to identify the data, process, interface and geography requirements for the users of a new system. However, most importantly, the purpose is to specify these requirements without expressing computer alternatives and technology details. The key deliverables of this phase are the final models and prototypes that are organized into a business requirements statement.

The purpose of the decision analysis phase is to identify candidate solutions, analyse those candidate solutions and recommend a target system that will be designed and implemented. The key deliverables of the phase is a formal systems proposal to system owners.

The purpose of the design phase is to transform business requirements from the definition phase into a set of technical design blueprints for construction. The key deliverables of this phase is a technical set of design specifications.

The purpose of the construction phase is to build and test a functional system that fulfils business and design requirements. Implementing the interfaces between the new system and existing production systems is also the target of this phase. The key deliverables is the functional system; however, the rapid application development (RAD) strategy of FAST results in several interim deliverables called prototypes.

The purpose of the implementation phase is to install, deploy and place the new system into operation or production. The key deliverables of this phase is the production system for the system users. Training and support are also the outputs of the delivery phase.

Once the system is operating, it delivers the business solution to the end-users. It will still require ongoing system support for the remainder of its life.

3 Prototyping is the act of building a small-scale working model of the users’ requirements to discover or verify those requirements. Its main purpose is to enhance the quality of a system specification or a system design by providing a working model of the system.

Prototyping encourages user involvement as a user ‘sees’ a working system, can operate it and suggests improvements and refinements before development is committed. In other words, the user becomes an active team member.
4 Cross life cycle activities are activities that overlap many or all phases of the methodology. They are normally performed in conjunction with several phases of the methodology. Cross life cycle activities include fact-finding, documentation and presentation, estimation and measurement, feasibility analysis, project management and process management.

The main purpose of each activities:
- Fact-finding. To collect information about systems, requirements and preferences by using interviews, questionnaires and other techniques.
- Documentation. To record facts and specifications for a system.
- Presentation. To package documentation formally for review by interested users and managers.
- Estimation. To approximate the time, effort, costs and benefits of developing systems.
- Measurement. To measure and analyse developer productivity and quality.
- Feasibility analysis. To measure how beneficial the development of an information system would be to an organization.
- Project management. To develop an acceptable system within the allotted time and budget by planning, delegating, directing and controlling progress.
- Process management. To establish standards for activities, methods, tools and deliverables of the life cycle.

5 The three main functions of project management are planning, scheduling and controlling of systems development.
- Planning involves the identification of all major segments or phases of a project.
- Scheduling relates tasks identified in planning to a time sequence.
- Controlling involves monitoring work to compare actual performance with plans and schedules.

Self-test 2.3

1 A CASE tool is a program (software) that automates or supports one or more phases of a systems development life cycle. The technology is intended to speed up the process of systems development and to improve the quality of the resulting systems.

2 CASE tools should be used in most aspects of information systems development where appropriate. CASE tools have been found to be beneficial in improving the overall productivity of analysts, designers, developers and project managers. Though CASE tools come in a wide range of forms and applications, overall the tools help to organize, manage and analyse the vast amounts of information associated with an IS development project. Greater quality of system can be achieved through the tools’ robust methodologies along with automated efforts
in the software testing phase. Since the data is organized within the system database, documentation is better managed. A resultant high quality system and documentation leads to reduced costs in life cycle maintenance.

3 I would take the following steps:

Step 1: Gather as much information as I can through trade journals on what CASE tools exist and what they can do.

Step 2: Evaluate the life cycle of the IS development effort and how such tools can facilitate the development process.

Step 3: Review publicized evaluations of required CASE tools and make a short-list of candidates based on: capability, rating/performance, compatibility, tool-integration, reputation, technical support and cost.

Step 4: Request trial versions and demonstration of the software.

Step 5: Make choice, purchase and train users on the tools.

4 Upper-CASE refers to software tools associated with the early phases of the information systems development life cycle. These tools include modelling, documentation and modelling tools. Lower-CASE refers to software tools associated with the later phases of the information systems development life cycle. These tools include user-interface designers, database designer and software testing systems.
Unit 2

IS development life cycle and CASE tools
Course Team (Original course)

Project Manager: Steve Elliot, OUHK

Course developers: University of Technology, Sydney  
Craig McDonald, Consultant  
David Wilson, Consultant

Course designer: Shannon Timmers, OUHK

Course Team (Revised course)

Course developers: Ali F Farhoomand, Consultant  
Lester W Yee, Consultant

Course designer: Linda Chung, OUHK

Course co-ordinator: John Mak, OUHK

Member: Nigel Leung, OUHK

External course assessor

Professor N V Balasubramanian, City University of Hong Kong

Production

ETPU Publishing Team

Copyright © The Open University of Hong Kong, 1998.  

All rights reserved.  
No part of this material may be reproduced in any form  
by any means without permission in writing from the  
President, The Open University of Hong Kong.

The Open University of Hong Kong  
30 Good Shepherd Street  
Ho Man Tin, Kowloon  
Hong Kong
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Objectives</td>
<td>3</td>
</tr>
<tr>
<td>Systems development life cycles and methodologies</td>
<td>4</td>
</tr>
<tr>
<td>Principles of systems development</td>
<td>5</td>
</tr>
<tr>
<td>Phases in a modern systems development life cycle</td>
<td>7</td>
</tr>
<tr>
<td>Cross life activities</td>
<td>10</td>
</tr>
<tr>
<td><strong>CASE tools</strong></td>
<td>12</td>
</tr>
<tr>
<td>CASE tool classification</td>
<td>13</td>
</tr>
<tr>
<td>CASE repository</td>
<td>13</td>
</tr>
<tr>
<td>CASE tool evaluation</td>
<td>14</td>
</tr>
<tr>
<td>Benefits</td>
<td>15</td>
</tr>
<tr>
<td><strong>A note on System Architect</strong></td>
<td>17</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>18</td>
</tr>
<tr>
<td><strong>References</strong></td>
<td>19</td>
</tr>
<tr>
<td>Systems analysis and design</td>
<td>19</td>
</tr>
<tr>
<td>Software engineering</td>
<td>19</td>
</tr>
<tr>
<td><strong>Glossary</strong></td>
<td>20</td>
</tr>
<tr>
<td><strong>Answer key for self-test questions</strong></td>
<td>24</td>
</tr>
</tbody>
</table>