Overview

Unit 14 is organized in seven sections. The section following this overview describes the phases involved in systems implementation. Specifically, you will learn about the activities associated with the construction and delivery phases. The third section describes the tests that are required to evaluate the targeted programmes and system. Section four examines the post-implementation review process with a view to objectively assessing the effectiveness of the implemented system. Section five focuses on system maintenance and enhancement. You will learn that a significant portion of a systems analyst’s time is spent on maintaining and enhancing the existing systems. Section six relates to issues surrounding information systems exposures and controls. Down-times (system failures) of any system affect an organization’s ability to function effectively. As such it is important to learn how to deal with systems failures and how to recover from them. The last section includes a case study that demonstrates how Orient-Pacific Insurance Corporation should implement the system under study, a summary of Unit 14, references, and a glossary of terms.
Introduction

The diverse components of today’s IS are distributed across several departments or functions in an organization which is likely to be geographically dispersed. We need to understand how system maintenance fits into the four building blocks of IS—data, processes, interfaces, geography. Not only is implementation of new systems a complex task, issues related to system maintenance and system support also require ongoing attention. IS are dynamic systems whose success depends on proper maintenance, support and enhancement.

You should note that after a system has become operational it is integrated into the exiting systems in an organization with the objective of achieving the business goals for which it has been designed. Failure to maintain and support the implemented systems properly would invariably lead to their failure. You would be surprised to learn that almost half of the time and effort of an average systems analyst is spent on maintaining or supporting the existing systems. So it is very important to gain an intimate insight into the issues related to system support and maintenance.
Objectives

By the end of Unit 14, you should be able to:

1. Describe the activities involved in the implementation process in terms of the construction phase of the development life cycle, including testing networks, databases, programs, and systems.

2. Describe the activities involved in the implementation process in terms of the delivery phase of the development life cycle, including conversion, installation, and training.

3. Describe software tests that are required to ensure the overall system integrity.

4. Explain the purpose of post-implementation review.

5. Discuss system maintenance and system enhancement.

6. Discuss IS exposures and controls.
Phases involved in systems development

As in Units 4 and 9, in this Unit we will follow the phases in the FAST methodology to describe the activities involved and the deliverables produced in each of the two development phases.

Systems development refers to the construction and delivery of the new system into the day-to-day operation (also called production). Like the phases involved in systems analysis and systems design, the necessary documentation associated with each activity involved in systems implementation is produced and delivered to the project repository for future reference.

Construction

The purpose of this phase is to build and test the new system, and to implement the interfaces between the new system and existing systems. The activities involved in this phase are:

- **build and test network**

  Using the specifications generated in the design phase, we need to build the required networks and test them to ensure that they work properly. The details of the installed network are documented in the project repository.

- **build and test databases**

  At this stage we need to build the required databases. You should note that because databases are shared by more than one application, we need to build them before programming. The key deliverable of this stage is an unpopulated database, which will eventually be populated, used and maintained by the applications. The revised database schema and test model details are placed in the project repository.

- **install and test new software packages**

  Depending on the systems design and integration requirements and program documentation, we may need to install, integrate and test new software packages.

- **write and test new programs**

  Finally at this stage we are ready to write any required in-house programs. Using the technical design statement, programming plan and data test, systems builders, in conjunction with systems designers, write the programs. Other deliverables at this stage may include a set of reusable software components, which are placed in a software library, and program documentation, which is placed in the project repository. Some of you are likely to have written small or large programs based on certain required specifications. Have you ever thought of all the steps
that it takes to reach the programming stage? Do you remember the numerous phases, activities, and techniques that you have learned in the previous 13 Units? You would be surprised to learn that many IS projects fail because some of the initial stages of the development cycle have been skipped and have not been given the requisite attention.

Having written the specified programs, you need to test them thoroughly. Because of the importance of software testing in the overall success of IS projects we will discuss different types of software test in the section following the implementation phase.

**Implementation**

Now that the programs are written, it is time to put the new system into production. This is primarily the job of the systems analyst. The activities that involve system delivery are described below.

- **conduct system test**

  In the last activity related to the construction of the system we tested individual programmes. At this stage we need to test the overall system to ensure that all the new and old components work together. Details of such tests will be covered in the next section.

- **prepare conversion plan**

  Transition from the old to the new system requires planning, a job that is usually done by the project manager. The key deliverable of this activity is a conversion plan, which highlights the databases that are required to be installed, the required training and documentation, and a conversion strategy.

  System installation involves activating a new system while disconnecting an existing one. There are four installation strategies:

  - **Abrupt cut-over** or immediate replacement, is when the old system is terminated and the new system is installed.

  - **Parallel conversion** with a single cut-over point or with a gradual shift to the new system.

  - **Location conversion** is used when the same system needs to be installed in multiple locations. The system is first installed in one location, using abrupt cut-over or parallel strategies. After the site has approved the systems, it is then installed in other locations.

  - **Version conversion** is used when development and installation are jointly phased — additional capabilities are developed and installed as incremental steps. When installation activity is phased separately from development — a fully functional system is developed but is installed in phases, with not all functions being adopted at once.
• install databases

During installation, you convert data from existing sources to create new production databases and files. Data conversion tasks vary considerably depending on existing data. If existing data is complete, accurate and already in computer-readable form, then conversion is straightforward and quick. However, existing data may be incomplete, inaccurate and stored in a variety of places and forms. Then, data conversion is a challenge.

Data conversion invariably involves constructing a small, temporary system to capture current data, verify and transform that data and create new databases and files. Despite the small and temporary nature of a conversion system, it must be rigorously tested (why invest care in developing a new system, if data used to load it is inaccurate and incomplete?). Also, you may not be able to convert all data automatically —some manual intervention may be necessary before conversion can be performed. You must also verify all converted data: are they accurate? Are they in the correct (new) format? Are they faithful to their original form?

• train system users

Users are the ultimate clients of the system. It should be clear that the success of the system depends to a large extent on their ability to use the system effectively. Therefore it is very important to train the users adequately and properly. The key deliverable in this activity is user training and documentation. It is extremely important that these manuals are written clearly and in a non-technical way so that the users can understand them.

• convert to new system

At last! The system is converted and the ownership of the system is transferred to the end users. Changeover marks the transfer of your system to production mode. Users assume full responsibility for the system.

Reading

‘The construction phase’ and ‘The implementation phase’, pp. 607–17 in your text. This reading, which relates to Objectives 1 and 2 of this unit, provides a detailed explanation of the activities involved in the construction and implementation phases.
1 Why is it important to build and test databases before starting on any programming activities?

2 Abrupt cut-over is one of the commonly used strategies for converting from an existing to a new production information system. By employing this strategy, the old system is terminated and the new system is placed into operation on a specific date. State the advantages and disadvantages of this strategy.

3 User training can be performed one on one but this method is not as popular as group training. Why?
Software testing

The goals of software testing are to:

- force a program or module to work incorrectly
- discover causes of these errors
- revise program or module code to eliminate those errors.

Note that it is a destructive process. This contrasts with a common misconception that the goal of testing is to prove that a program or module functions correctly. The objective is failure, not proof that a program or module works. This may appear to be a semantic difference, but it is fundamental to your approach to software testing.

Levels of testing and testing strategies

Once individual system components are available (coding completed), actual software testing commences. A complete schedule of testing involves six broad levels: stub testing, unit testing, integration testing, function testing, systems testing and acceptance testing. Stub testing, unit testing, function and integration testing are primarily the responsibility of information systems developers. Systems testing and acceptance testing are primarily the responsibility of users.

You should not think that all software testing tasks commence as soon as system components are available. Much preparation is necessary before the actual software testing can be undertaken. Software testing is itself a mini-project (within the overall development project) and commences with the development of a high-level test plan. This document defines the scope of testing, identifies components to be tested, by whom and within what timeframe. It is produced during the analysis and design phases and is confirmed in the implementation phase. An initial task within this plan is test case creation. Each test case is a set of input data, expected results (from executing that data) and acceptance criteria (functional requirements that have been agreed as the basis for accepting a system). Test cases are developed such that all system requirements and combinations of requirements are tested. These test cases are to be completed in time for actual software testing to commence.

Stub (module) testing

Stub testing is the responsibility of programmers. A programmer codes a module and then starts testing the module. First, ‘desk checking’ involves mentally tracing simple input data through a module. The aim is to check that the logic is correct and that output is as expected. Finally, machine checking involves compiling and running a module. The aim is to check that a module actually runs as expected.
Unit or programme testing

After all the modules are stub-tested, the program is tested as an integrated unit.

Integration and function testing

Integration and function testing are the responsibility of systems analysts and designers. Integration testing checks that individual modules connect together to form sound and workable programs. It focuses on interface errors. Function testing then checks whether user requirements for inputs and outputs have been met. It focuses on an entire system.

Software testing is a progressive exercise. Each level of testing relies on the previous level. First, all modules must be individually tested and necessary corrections made. Then, modules are integrated and tested in small groups; again corrections are made. Only then can function testing examine an entire system. To start at this level would be impractical and unmanageable. In turn, function testing is a necessary prerequisite to system and acceptance testing.

System and acceptance testing

System testing and acceptance testing are the responsibility of users. System testing checks operational and procedural relationships of data with the numbers of records expected in a full production situation. It brings together users, documentation (user and operational manuals), test data files, software and hardware, to determine whether your system functions as required (as specified in your Requirements Specification). Acceptance testing is a live ‘dress rehearsal’ immediately prior to operation. It is a final opportunity for users to accept, modify or reject the system.

It is virtually impossible to design a set of test cases that completely simulate all potential situations that can arise. So the key to successful testing is to systematically design test cases that maximise the potential to detect errors in software. Test cases are created for as many real-life situations as possible. These should include:

- valid and invalid data (including out-of-scale values that cause problems and challenge assumed maximums and minimums)
- normal processing routines
- error-handling routines
- variations of different input and output formats and combinations
- addition and deletion of records on master files
- file storage and retrieval algorithms
- start-up and termination procedures
- back-up and recovery procedures
- documentation, particularly user and operations manuals.
A common failure is to design test cases for expected problems. While it is impossible to anticipate every situation, you should try to identify the unexpected. Your efforts in producing exhaustive test cases will result in fewer failures for your system in production.

**Self-test 14.2**

1. Why should we force a program to work incorrectly?

2. How can you test a higher-level module before coding its lower-level modules?

3. Why do we need systems testing?
Post-implementation review

At this point, a new system has been integrated into an organization. Manual and automated processes are working together as part of routine operations. Anticipated benefits are being realized. The successes and failings of the system and development process should now be reviewed. Senior management needs to assess the effectiveness of decisions that led to the implementation of a system.

The purpose of a post-implementation review is to objectively assess a completed project. There are two specific areas included in such a review. Firstly, the performance and success of the development process are reviewed. Secondly, the performance and success of the system itself is reviewed.

First we concentrate on the performance and success of the development process. This review assesses the technical performance of systems developers. In doing so, it deals with what happened during the development and why it happened. Typical questions addressed are:

• Were stated aims and objectives of a project achieved?
• How do estimated and actual development costs compare?
• How do estimated and actual schedules and time-scales compare?
• What are the reasons for variations between estimated and actual performance?

This review documents successes so that they can be repeated during future projects. It also documents failures so that they can be avoided. To be of value to information systems management, this review must provide an honest, accurate and objective evaluation of the successes and failures. To do this, it must be conducted immediately after development is complete (without affecting implementation activities) and should involve all development staff.

Second, we concentrate on the performance and success of the system itself. This review assesses the extent to which specified requirements are being met. This is usually carried out some three to six months after the system has been installed — so that the system has completed several cycles and is stable. Typical questions addressed are:

• Are operational expectations of a system being met?
• Have anticipated benefits been realised?
• Are there any residual problems with a system?
• Are there any unanticipated requirements that are not being satisfied?

This review documents whether or not users received what they wanted. In doing so, it identifies problems and omissions. To be effective, this review involves users, user management, developers and information systems management. User management can then determine whether corrective
maintenance or system enhancements are needed. Corrective maintenance may involve improving response time or turnaround time of an automated system. System enhancement may involve new features and requirements being incorporated into a system. If extensive, such enhancement should be treated as a new development project.

**Self-test 14.3**

1. Why is it necessary to perform a post-implementation review?

2. Why should we document successes as well as failures when assessing the new system?
System maintenance and enhancement

So far we have reiterated the fact that initial development work is only a minor part in the useful life of a system. Once we have performed post-implementation review of a system it enters the maintenance phase. Time and cost expended on a system in maintenance activity can be many times that expended during initial development. We cannot emphasize enough that, you, as a systems analyst, will eventually spend a significant portion of your working time on maintaining systems.

Maintenance is necessary to correct errors, to adapt a system to changes in business procedures, or to incorporate new requirements. Maintenance work should be viewed as development work in ‘miniature.’ The same structured approach that has been described in this course for the initial development of systems should be followed for maintenance work.

It is reasonable to expect that maintenance activity and cost will increase over time. A newly implemented system will meet its specified requirements for some period of time. As time progresses, new requirements are likely to emerge. Business procedures are likely to change. In other words, demand for maintenance work increases as a system becomes older.

When a system becomes operational, it becomes a candidate for enhancement, update or revision. This type of adaptive maintenance, which is called system enhancement, usually calls for modifications or expansion of the existing applications caused by changes in business requirements or technology.

Reading

‘Systems maintenance’, pp. 628–32, ‘System enhancement — correcting errors’, pp. 633–38 in your text. These readings, which correspond to Objective 5 of this unit, highlight the importance of system maintenance and enhancement.

Self-test 14.4

1. What is a repository? How is it different from a program library and a business database?

2. Why is system enhancement necessary?

3. What causes bugs?
IS exposures and controls

Management is responsible for establishing and maintaining adequate internal controls of business processes in order to minimize the risks of various exposures. As organizations increasingly depend upon information systems, similar controls need to be designed into information systems themselves. Traditional security, audit and control mechanisms take on a new and different form in a computer-based system. New systems must be developed upon a solid framework of operational and accounting controls.

Types of Exposures

Exposures are sources of risk to the overall integrity of an information system. Typical exposures are:

- **Errors and omissions**: These include accidental or intentional errors and omissions in application programs, data communications, personal computers, software and manual procedures.

- **Theft**: This refers to safeguarding an organization’s assets (including information assets) from unauthorized copy or removal by persons legitimately within an organization (employees, vendors) or outsiders. It includes embezzlement, fraud and outright theft.

- **Unauthorized access**: This includes unauthorized access to databases, data centres and personal computers. Legitimate users may have access to specified areas only (e.g., to inventory data but not payroll data) or be restricted in their use of data (e.g., they may read data but not update it).

- **Breach of privacy**: This protects against accidental or intentional release of confidential information beyond normal business requirements. This includes personal information about an individual (e.g., prevention of releasing names and addresses to mailing companies) and private information about an organization (e.g., prevention of releasing detailed sales figures to competitors).

- **Lost data**: This includes batch data never input to a system and online data ‘forgotten’ by restart and recovery procedures.

- **Disasters and recoveries**: This refers to interruptions, both natural (e.g., flood, fire, earthquake) and intentional (e.g., terrorist activity, arson) that disrupt a data centre, equipment or data communications. Refer to p. 584 of your text for a summary of activities involved in system recovery.

This list is not exhaustive. Also, it could be developed at a lower level of detail. You have to determine what is appropriate for your system and your organization.
Types of controls

You need to appreciate both application and system controls in order to design effective, efficient and well-controlled systems:

- **application controls** are designed into a specific application system; these are particular to a system and are specified by a systems designer.

- **system controls** are provided by operating software (for example, the operating system, database or data communications software) and physical security controls; these are usually provided by a central data centre and are available to all application systems.

You need to be aware of system controls to ensure that your system utilizes centrally provided controls. You need to understand application controls to ensure that your system design will provide consistent, accurate, timely and relevant information. Controls are classified as:

- **preventative controls** that discourage, restrain or stop an unwanted act or event

- **detective controls** that reveal or report an unwanted act or event

- **corrective controls** that remedy, recover or make good an unwanted act or event.

**Self-test 14.5**

1. What systems backup is necessary?

2. Describe the role of controls in safeguarding information systems.

**Activity 14**

1. Match the term with the appropriate definition.
   - (1) parallel conversion
   - (2) systems testing
   - (3) stub testing
   - (4) abrupt cut-over
   - (5) audit testing
   - (6) validation testing
   - (7) staged conversion
   - (8) unit or program testing
   - (9) location conversion
   - (10) verification testing
A The test performed on individual modules, whether they be main program, subroutine, subprogram, block, or paragraph is called _______.

B A test whereby all of the modules that have been coded and stub tested are tested as an integrated unit is called _______.

C _______ is a test that ensures that application programs written in isolation, work properly when they are integrated into the total system.

D _______ is a conversion strategy where, on a specific date, the old system is terminated and the new system is placed into operation.

E _______ is a conversion strategy where both the old and new systems are operated for some period of time to ensure that all major problems in the new system have been solved before the old system is discarded.

F _______ is a conversion strategy where a new system that will be used in numerous geographical locations is converted at one location before being farmed to the other sites.

G _______ is a conversion strategy wherein each successive version of the new system is converted as it is developed.

H _______ is an acceptance test that runs the system in a simulated environment using simulated data to identify errors and omissions regarding end-user and design specifications.

I _______ is an acceptance test that runs a system in a live environment using real data to validate a systems performance, peak workload processing performance, human engineering, methods and procedures, and backup and recovery.

J _______ is an acceptance test aimed at certifying that the system is free of errors and is ready to be placed into operation.


2. Match the term with the appropriate definition.

(1) systems support
(2) unit testing
(3) version control
(4) systems testing
(5) software metrics
(6) regression testing

A The on-going maintenance of a system(s) after it has been placed into operation, including program maintenance and system improvements.
B Ensures that the stand-alone program not only fixes the bug, but does so without side effects.

C Ensures that the entire application, of which the modified program was a part, still works.

D Extrapolates the impact of the changes on program and application throughput and response time from the before and after results using the test data.

E The process whereby a librarian (usually software-based) keeps track of changes made to programs thus allowing recovery of prior version of the programs in the event that new versions cause unexpected problems.

F Mathematically proven measurements of software quality and productivity.

Summary

You should now understand the importance of system implementation and system maintenance and support. You have learned about the activities involved in the last two phases of the FAST methodology. You were also exposed to the importance of software testing. You should now understand that software testing is an important systems development activity. It uncovers design limitations, coding errors, unforeseen system requirements and faulty procedures. The aim is to reveal these deficiencies so that they can be corrected before your system goes into production. Careful testing requires careful planning of test activities and resources.

You also noted that after the installation of a system we need to perform post-implementation review to assess the overall effectiveness of the system in terms of achieving the original goals for which it has been designed. Furthermore, you learned that the ‘real’ life of a system starts after it enters the production mode, and as such systems analysts need to spend a fair amount of time maintaining and supporting the existing systems. Finally, you gained an understanding of the exposures that may compromise the overall integrity of an information system, ranging from theft to unauthorised use to system crash to nature disaster. You learned about the importance of controls and how to use them to prevent or minimize the associated risks.

Unit 14 concludes this course. We hope that you have found the material in the study units useful. We believe that systems analysis and design is an extremely important discipline, whose professional stature will continue to climb in the future. In order to excel in this area you need to ensure that you develop a solid understanding of the theoretical and conceptual aspects of IS development before putting your knowledge into practice. You also need to keep abreast of business and technological developments in the area. In light of the rapidly changing business world and technology arena, you will be faced with an extremely challenging job. You, as a systems analyst, are in the forefront of technological as well as business developments. As such, you are expected to understand business problems and opportunities and provide solutions using appropriate technology. This course has provided you with the necessary tools for such a challenging job. You now need to put these tools to practice.
References

Control and reliability design


Software testing


Software quality assurance


Glossary

This glossary provides brief definitions of the main technical terms you encountered in Unit 14. The definitions given here are offered in addition to, or in place of those found in your textbook.

**Abrupt cut-over:** On a specific date (usually a date that coincides with an official business period such as month, quarter, or fiscal year), the old system is terminated and the new system is placed into operation.

**Application controls:** Controls designed into a specific application system; these are particular to a system and are specified by a systems designer.

**Corrective controls:** Controls that remedy, recover or make good an unwanted act or event.

**Detective controls:** Controls that reveal or report an unwanted act or event.

**Location conversion:** When the same system will be used in numerous geographical locations, it is usually converted at one location (using either abrupt or parallel conversion). As soon as that site has approved the system, it will be farmed to the other sites. Other sites can be cut over abruptly because major errors have been fixed.

**Parallel conversion:** Under this approach, both the old and new systems are operated for some time period. This is done to ensure that all major problems in the new system have been solved before the old system is discarded.

**Preventative controls:** Controls that discourage, restrain or stop an unwanted act or event.

**Stub testing:** Testing of individual modules.

**System controls:** Controls provided by operating software (for example, the operating system, database or data communications software) and physical security controls; these are usually provided by a central data centre and are available to all application systems.

**Systems enhancement:** When a system becomes operational, it becomes a candidate for enhancement, update or revision. This type of adaptive maintenance, which is called system enhancement, usually calls for modifications or expansion of the existing applications caused by changes in business requirements or technology.

**Systems implementation:** The construction of the new system and the delivery of that system into production (meaning day-to-day operation).

**Systems maintenance:** The purpose of system maintenance is to correct errors, to adapt a system to changes in business procedures, or to incorporate new requirements.

**Systems support:** The ongoing maintenance of a system after it has been placed into operation. This includes program maintenance and system improvements.
**Unit testing:** A test whereby all the modules that have been coded and stub tested are tested as an integrated unit.

**Unpopulated database:** This is a database with its structure implemented but with no data presence.

**Version conversion:** Version conversion is used when development and installation are jointly phased — additional capabilities are developed and installed as incremental steps.
Answer key for self-test questions

Self-test 14.1

1  Building and testing databases must precede any programming activities because databases are the resources shared by the computer programs to be written.

2  Abrupt cut-over is a rather high-risk approach when compared to other conversion strategies because there may still be major problems that will not be uncovered until the system has been in operation for at least one business period. This is the disadvantage of employing abrupt cut-over. However, the advantages of this conversion strategy are that it involves no transition costs and it may be desirable if a business policy or government mandate becomes effective on a specific date and the system could not be implemented before that date.

3  Group training is generally preferred to one-on-one training because it is a better use of time and it encourages group learning possibilities. Through practice with large groups where common problems and issues can be addressed effectively, users can master the skills more quickly and effectively.

Self-test 14.2

1  By forcing a program to work incorrectly, we can discover errors and learn about the shortcomings of the module, and by correcting the errors the program becomes more capable of handling the complex day-to-day situations.

2  One can test a higher-level module before coding its lower-level modules by stimulating the lower-level modules which are often called stubs. Stub modules are subroutines, paragraphs and the like that contain no logic. They only need to show that they have been correctly called by printing an alert statement, and then pass the control back to the parent module.

3  Systems testing is needed because it ensures that application programs written in isolation work properly when they are integrated into the total system. A single program works properly does not mean that it works properly with other programs. Thus, an integrated set of programs should be run through a systems test to make sure one program properly accepts, as input, the output of other programs.
Self-test 14.3

1 The purpose of the post-implementation review is to objectively assess a completed project. By evaluating the system while it is in production, improvements can be made to meet the new requirements.

2 When assessing a system, we should document successes and failures because information systems management aims to give an honest, accurate and objective evaluation of the performance of the system. Successes are documented so that they can be repeated whilst failures can be learnt and avoided on future projects.

Self-test 14.4

1 A repository is a collection of places where all documentation associated with the application and project is kept. It stores all system models and detailed specifications.

   A program library stores the actual application programs that have been placed into production while a business database looks after the operational data created and maintained by the production application programs.

   A repository differs from a program library and a business database in that, it is not manipulated by users in daily routines. Contents of the repository are not altered unless application changes i.e., changes in models and specifications occur. However, a repository sometimes helps users in operating the system by supplying documentation of applications.

2 System enhancement is necessary because we need to modify or expand the application system in response to constantly changing requirements.

3 Bugs are the errors encountered when using the system. They can be caused by:
   • miscommunication of requirements.
   • design flaws.
   • situations that are not anticipated and, therefore, not tested.
   • unanticipated misuse of the programs.

Self-test 14.5

1 Systems backup is required to prevent possible loss of data, which may be as a result of natural disasters, theft, or system failure.

2 Control consists of all the methods, policies, and organizational procedures that ensure the safety of the organization’s assets, the accuracy and reliability of its accounting records, and adherence to management standards. For computerized information systems, controls consist of both manual and programmed procedures. Controls that
safeguard information system security are especially important in today’s on-line networked environment.

**Activity 14**

1. (1) E
   (2) C
   (3) A
   (4) D
   (5) J
   (6) I
   (7) G
   (8) B
   (9) F
   (10) H

2. (1) A
   (2) B
   (3) E
   (4) C
   (5) F
   (6) D
Implementation and support
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 coordinator: John Mak, OUHK

Member: Nigel Leung, OUHK

External course assessor

Professor NV Balasubramanian, City University of Hong Kong

Production

ETPU Publishing Team

Copyright acknowledgements

The following copyright materials are reproduced under the terms of a licence issued by Hong Kong Reprographic Licensing Society. No further copying is permitted.

Information in Activity 14.

Acknowledgement

The OUHK gratefully acknowledges a grant of $100,000,000 from the Hong Kong Government designated for course development.

B329 Systems Analysis and Design was developed in part using resources provided by this generous grant.

Copyright © The Open University of Hong Kong, 1998, 1999.

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>Phases involved in systems development</td>
<td>4</td>
</tr>
<tr>
<td>Construction</td>
<td>4</td>
</tr>
<tr>
<td>Implementation</td>
<td>5</td>
</tr>
<tr>
<td>Software testing</td>
<td>8</td>
</tr>
<tr>
<td>Levels of testing and testing strategies</td>
<td>8</td>
</tr>
<tr>
<td>Stub (module) testing</td>
<td>8</td>
</tr>
<tr>
<td>Unit or programme testing</td>
<td>9</td>
</tr>
<tr>
<td>Integration and function testing</td>
<td>9</td>
</tr>
<tr>
<td>System and acceptance testing</td>
<td>9</td>
</tr>
<tr>
<td>Post-implementation review</td>
<td>11</td>
</tr>
<tr>
<td>System maintenance and enhancement</td>
<td>13</td>
</tr>
<tr>
<td>IS exposures and controls</td>
<td>14</td>
</tr>
<tr>
<td>Types of Exposures</td>
<td>14</td>
</tr>
<tr>
<td>Types of controls</td>
<td>15</td>
</tr>
<tr>
<td>Summary</td>
<td>18</td>
</tr>
<tr>
<td>References</td>
<td>19</td>
</tr>
<tr>
<td>Glossary</td>
<td>20</td>
</tr>
<tr>
<td>Answer key for self-test questions</td>
<td>22</td>
</tr>
</tbody>
</table>