



# Introduction to Systems Development

NQF Level 2

R Jonker & MWH Smit

STUDENT'S BOOK

**TVET FIRST**

# Introduction to Systems Development

NQF Level 2

Student's Book

R Jonker & MWH Smit



**Introduction to Systems Development NQF Level 2  
Student's Book**

© R Jonker & MWH Smit, 2011

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, photocopying, recording, or otherwise, without the prior written permission of the copyright holder or in accordance with the provisions of the Copyright Act, 1978 [as amended]. Any person who does any unauthorised act in relation to this publication may be liable for criminal prosecution and civil claims for damages.

First published in 2011 by  
Troupant Publishers [Pty] Ltd  
PO Box 4532  
Northcliff  
2115

Distributed by Macmillan South Africa [Pty] Ltd

ISBN: 978-1-9203-3487-1  
Web PDF ISBN: 978-1-4308-0279-2

It is illegal to photocopy any page of this book without written permission from the publisher.
---

**Acknowledgements**

Microsoft product screenshots used with permission from Microsoft.

While every effort has been made to ensure the information published in this work is accurate, the authors, editors, publisher and printers take no responsibility for any loss or damage suffered by any person as a result of reliance upon the information contained herein. The publisher respectfully advises readers to obtain professional advice concerning the content.

While every effort has been made to trace the copyright holders and obtain copyright permission from them, in some cases this has proved impossible due to logistic and time constraints. Any copyright holder who becomes aware of infringement on our side is invited to contact the publisher.

**Note:** Any reference to Further Education and Training (FET) in this book should be taken to mean Technical and Vocational Education and Training (TVET).

To order any of these books, contact Macmillan Customer Services at:

Tel: (011) 731 3300

Fax: (011) 731 3535

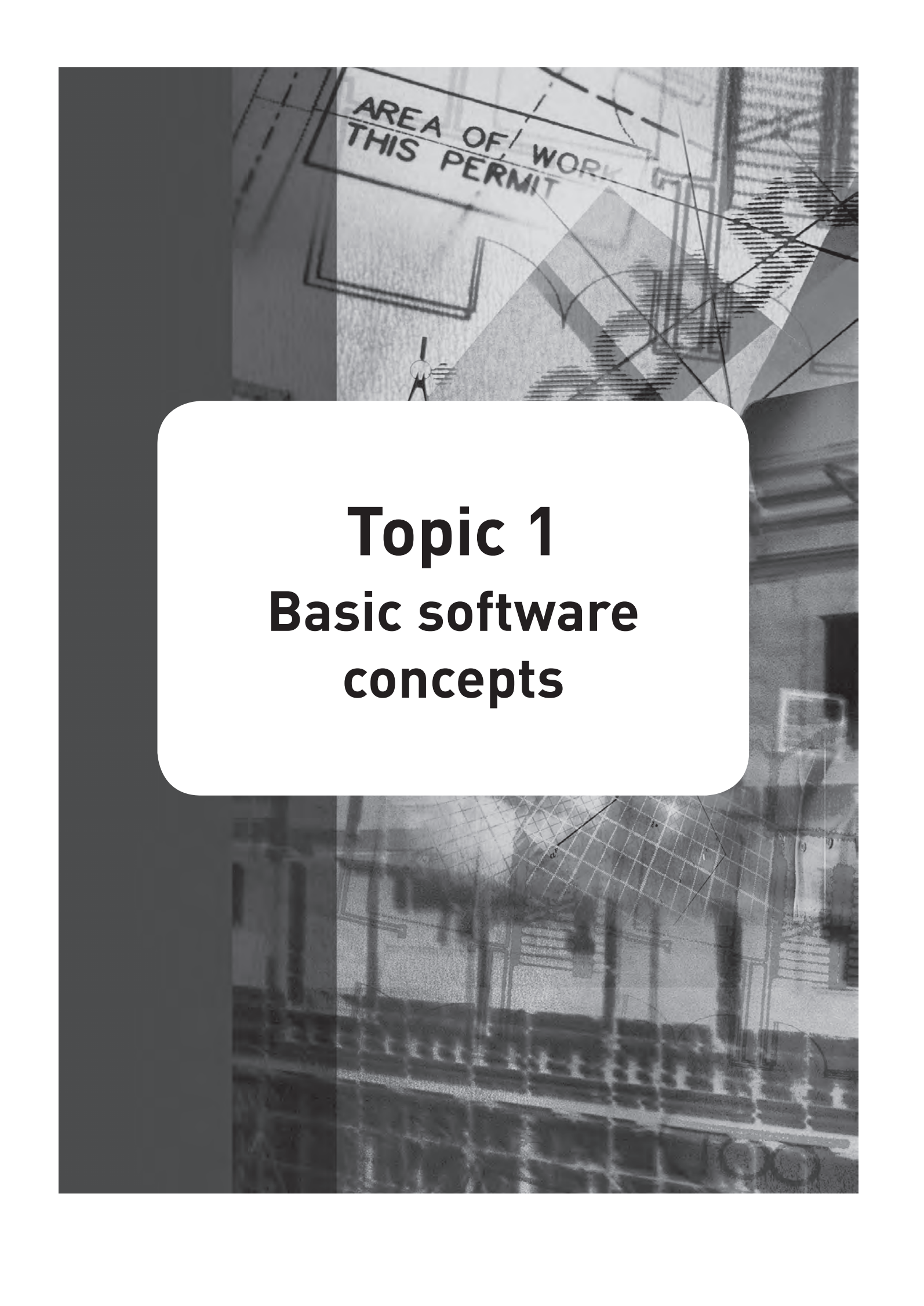
E-mail: [customerservices@macmillan.co.za](mailto:customerservices@macmillan.co.za)

# Contents

<b>Topic 1 Basic software concepts</b>	<b>1</b>
<b>Module 1 Types of software</b>	<b>2</b>
Unit 1.1 What is software?	2
Unit 1.2 Different types of software and their functions	3
Unit 1.3 Software tools	4
Unit 1.4 Software versions	7
<b>Module 2 Application software</b>	<b>10</b>
Unit 2.1 Features common to all types of application software	10
Unit 2.2 Different types of application software and their use	12
Unit 2.3 Purpose and use of features common to types of application software	18
Unit 2.4 Installing application software	22
<b>Module 3 System software</b>	<b>29</b>
Unit 3.1 System software	29
Unit 3.2 Operating systems	29
Unit 3.3 Utility programs	31
Unit 3.4 Language translators	33
<b>Module 4 Operating systems and environments</b>	<b>35</b>
Unit 4.1 Different types of operating systems	35
Unit 4.2 The environment in which operating systems operate	37
Unit 4.3 The history of operating systems in terms of proprietary and open-source	38
<b>Topic 2 Software development and programming languages concepts</b>	<b>49</b>
<b>Module 5 Generations of programming languages</b>	<b>50</b>
Unit 5.1 The various generations of programming languages	50
Unit 5.2 Features of programming languages	58
Unit 5.3 Strengths and limitations of programming languages	58
<b>Module 6 Uses of popular high-level programming languages</b>	<b>60</b>
Unit 6.1 Which high-level programming languages are the most popular?	60
Unit 6.2 Uses of high-level programming languages	66
Unit 6.3 Advantages and disadvantages of high-level programming languages	66
<b>Module 7 Object-oriented and visual programming</b>	<b>68</b>
Unit 7.1 Object-oriented and visual programming methodologies	68
Unit 7.2 Concepts of visual programming languages	72
Unit 7.3 Concepts of object-oriented programming	76
Unit 7.4 The relationship between visual programming, RAD, object orientation and OOP	78
Unit 7.5 Object-oriented programming in terms of the reuse of classes and the implementation of objects	79
Unit 7.6 Examples of object-oriented programming languages	80
<b>Module 8 Developing a computer program</b>	<b>82</b>
Unit 8.1 Basic steps in developing a computer program	82
Unit 8.2 Basic steps in a computer program development life cycle (PDLC)	82

<b>Module 9 Software development tools</b> .....	<b>86</b>
Unit 9.1 Software development tools .....	86
Unit 9.2 Uses of software development tools .....	86
<b>Topic 3 Computer data storage</b> .....	<b>93</b>
<b>Module 10 Computer data types</b> .....	<b>94</b>
Unit 10.1 Data types .....	94
Unit 10.2 Categories of coding systems .....	96
<b>Topic 4 Basic computer programming</b> .....	<b>107</b>
<b>Module 11 Problem solving</b> .....	<b>108</b>
Unit 11.1 Introduction .....	108
Unit 11.2 The steps in problem solving .....	108
<b>Module 12 Producing and documenting an algorithm</b> .....	<b>112</b>
Unit 12.1 Algorithms .....	112
Unit 12.2 Identifying inputs, processes and outputs for an algorithm .....	113
Unit 12.3 Drawing an IPO chart .....	115
Unit 12.4 Algorithmic structures needed to produce a feasible algorithm to solve a given problem .....	116
<b>Module 13 Producing and documenting pseudocode</b> .....	<b>121</b>
Unit 13.1 Pseudocode .....	121
Unit 13.2 The difference between pseudocode and an algorithm .....	122
Unit 13.3 Solving a problem using pseudocode techniques .....	122
<b>Module 14 Alternative design methods for documenting specifications or solutions</b> .....	<b>130</b>
Unit 14.1 Alternative methods for documenting and specifying a solution .....	130
Unit 14.2 Decision tables .....	143
Unit 14.3 Decision trees .....	143
Unit 14.4 Use case diagrams in UML .....	144
<b>Module 15 Implementing a programming language</b> .....	<b>147</b>
Unit 15.1 Using a programming language to implement the designed solution .....	147
Unit 15.2 Using the IDE of the programming language to write the source code according to the designed solution .....	151
Unit 15.3 Compiling and debugging the developed program for syntax and logical errors .....	158
Unit 15.4 Test the correctness of the program using sample data .....	204
<b>Topic 5 Principles of computer program quality assurance and project viability</b> ..	<b>209</b>
<b>Module 16 Basic principles of program quality assurance</b> .....	<b>210</b>
Unit 16.1 Good programming documentation principles .....	210
Unit 16.2 Quality assurance principles .....	221
Unit 16.3 Verification and validation .....	224
<b>Module 17 The principles used to determine project viability</b> .....	<b>227</b>
Unit 17.1 Viability of developing software .....	227
Unit 17.2 Project viability in terms of processing the information or data .....	229
<b>Glossary</b> .....	<b>236</b>





# **Topic 1**

## **Basic software concepts**

# Module 1

## Types of software

### Overview

At the end of this module, you should be able to:

- explain the term 'software'
- distinguish between various types of software and their purposes (applications and systems software)
- distinguish between open-source and proprietary software
- explain why the same software has different versions.



### Did you know?

John W. Tukey first used the term 'software' in 1957 to describe the concept of reading different sequences of instructions into the memory of a device to control computations.

## Unit 1.1 What is software?

Computer **software**, or simply software, consists of programs and other operating information used by a computer. Instructions and data are stored electronically in programs and enable your computer to carry out specific actions. The physical components of the system, such as the hard drive, printer, monitor, keyboard and mouse, are called **hardware**.

Software consists of:

- machine-readable code that enables the computer to perform a specific task or instruction
- all documentation that forms a vital component of each project, such as the specifications, the design, legal and accounting documents, as well as all the user manuals.

Software is generally written in a **high-level language** that is easier and more efficient for the programmer to use. A high-level language is a programming language that has instructions resembling a human language, such as English. A **low-level language** is a programming language that is close to machine code in form. The term 'high-level language' does not imply that the language is superior to low-level programming languages. In fact, in terms of depth of knowledge of how computers operate, the opposite may be true.

High-level programming languages are more abstract, easier to use and more portable across platforms than low-level programming languages. In general, high-level languages make complex programming simpler, while low-level languages tend to produce more efficient code.

With a high-level language, complex elements can be broken up into simpler, although still fairly complex elements. This allows programmers to avoid having to continually recreate the same programming code. However, the cost of this convenience is often less efficient code overall. For this reason, code that needs to run particularly quickly and efficiently may be written in a low-level language, even if a high-level language would make the coding easier.

A high-level programming language **translates** (converts) the program into a low-level machine language that consists of **binary values** (0, 1).

### Words & Terms

**software:** programs and other operating information used by a computer

**hardware:** the physical components of a computer, such as the hard drive, printer, monitor, keyboard and mouse

**high-level language:** a programming language that has instructions resembling a human language, such as English

**low-level language:** a language that is close to machine code in form

**translate:** to convert from one computer language into another

**binary values:** having two values or states; this describes a number system that has a base of two and uses 1 and 0 as its digits

These binary values are then loaded into the computer's **random access memory** (RAM) before being sent to the **central processing unit** (CPU) to perform the specific instruction.

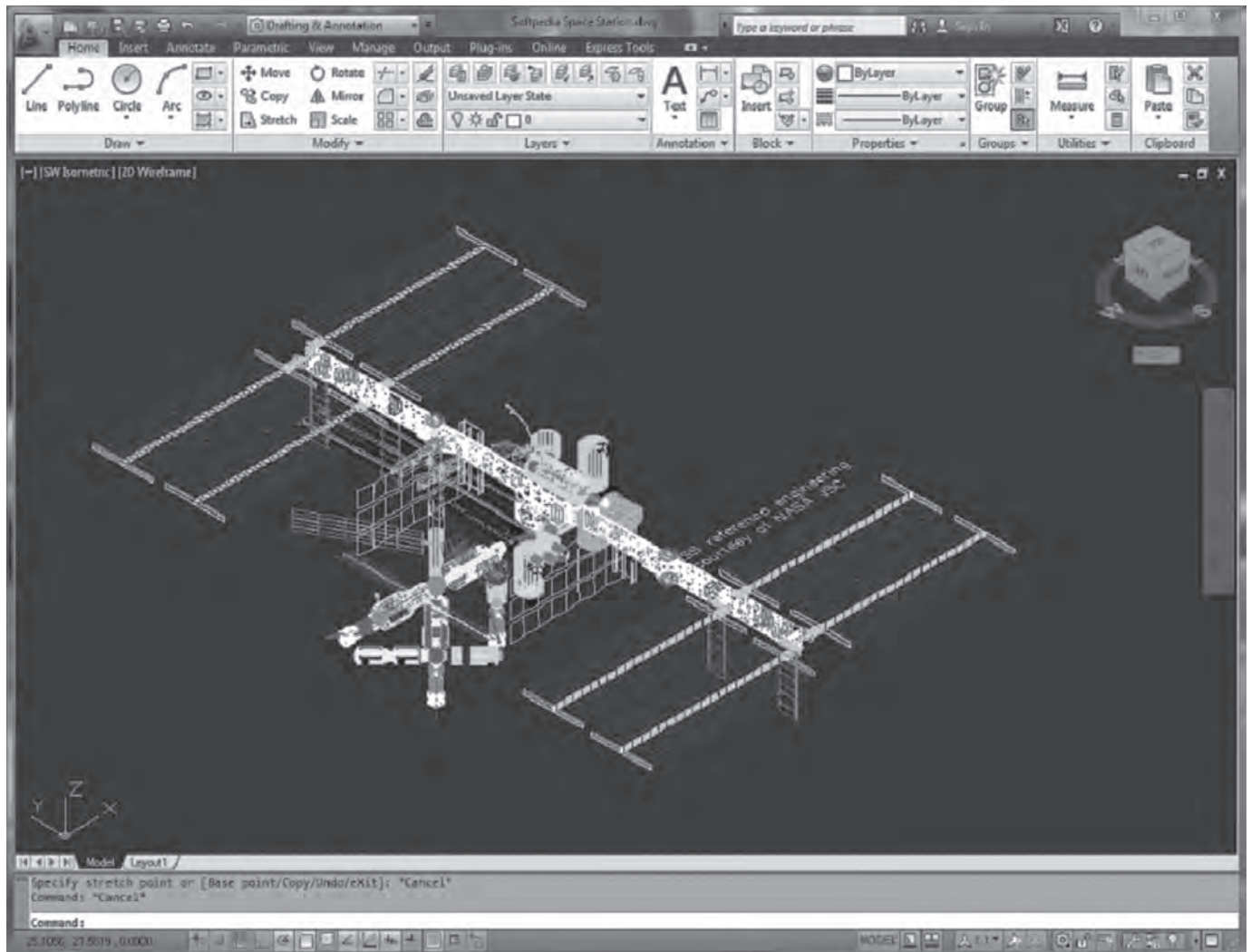


Figure 1.1 An example of software in use – AutoCAD 2011®

## Unit 1.2 Different types of software and their functions

Software can be divided into subclasses depending on the use and purpose of the program. Some software programs perform tasks that the user wishes them to perform, for example word processing, while other programs (such as antivirus software) perform tasks on different programs to achieve a desired result.

In the recent past, sales of all types of software have increased, providing the user with a wide variety of both general software and software designed for specific tasks. These include the following:

- Software tools are either basic single programs created to perform one task, such as word processing, or advanced software tools, such as a software bundle.

### Words & Terms

**random access memory (RAM):** computer memory used by programs to perform necessary tasks while the computer is on; it holds the program code and data during computation

**central processing unit (CPU):** interprets instructions and processes data contained in computer programs



- Application software (also called end-user programs) performs a specific task for the end user. Application software can be divided into numerous categories, for example media players, games, spreadsheets and word processors.
- System software refers to programs that manage a computer's resources, scheduling and the monitoring of computer events. Operating systems are a form of system software.
- Utility software is designed to help manage an operating system. It performs a single task or a range of small tasks to fine-tune the operating system and hardware.
- Language translators translate a high-level language into a low-level language (assembly language or machine language).

## Unit 1.3 Software tools

### Introduction

Most people would agree that the software market has changed dramatically in the last few years. In the past, users would buy what is called **proprietary software** from a **software vendor**. The software vendor would then bring out updated versions to replace older versions on a regular basis. If users wanted the updated version, they had to buy it and the software vendor thus made more money. This system provided some competition among the software companies, which helped to improve the quality of the software.

Today, however, users may also obtain free software, called **open-source software**. Users pay only for maintenance of the software. In this way, users have the advantage of having the latest version of the software for free, while the software company still makes money by charging for the maintenance of their software.

Open-source software is becoming more popular and there is now a wider range available. This software has become more reliable and **compatible** with other software applications. It has also become more easily available as a result of the internet. Open-source software has a big following on the internet where users worldwide use online forums to improve it.

The success of open-source software has proved to be a challenge for software vendors. The software products produced by these companies now have to be of an even higher standard, as they are not only competing against other software vendors, but against a global software community looking for alternative options to proprietary software.

Good examples of open-source versus proprietary software are Linux versus Microsoft Windows and Mozilla Firefox versus Internet Explorer.

### Proprietary software

Software vendors develop proprietary software for specific tasks. The source code is hidden from users who are not allowed to make any changes to it. If changes are needed, the software vendor will make the

#### Words & Terms

**proprietary software:** software programs that users must buy

**software vendor:** a software company that develops and sell proprietary software

**open-source software:** software programs that users can obtain for free

**software compatibility:** compatible software can interact with different versions of the same software, as well as with other software products



necessary changes. Proprietary software comes with obligatory, reliable and professional technical support that is normally provided via online chat rooms, forums or call centres staffed by software experts. Software vendors provide regular updates and bug fixes, and are constantly improving their software products. Microsoft Excel is an example of proprietary software (see Figure 1.2).

Advantages of proprietary software include the following:

- Customer support is obligatory – vendors help users with any software problems by providing reliable and professional support.
- There are regular updates to the software.

Disadvantages of proprietary software include the following:

- Closed standards may hinder compatibility with other software or between different operating systems.
- It is expensive to buy the licence for the software.
- The software may not be changed or customised without the vendor's permission.

No.	Customer name	Contact number	Purpose of visit	Time in	Description of customer	Start time
1	Nzibande Mr		2c Cindy	8:33	Beige jacket	8:36
2	Pedzisayi		Open acc	8:47	Beige jacket	8:49
3	Moeletsana		2c Julia	8:54		9:19
4	Mr Malan		H Loan	8:56	Black jacket	9:00
5	Mr Modimo		P Loan	8:58	Black jacket	9:00
6	Mr Phasa		Savings	9:03	Green jersey	9:03
7	Ms Skosana		Close acc	9:17	Black jacket	9:18
8	Ms Beukes		P Loan	9:42	Black jersey	9:43
9	Mr Strydom		P loan	9:48	Blue jacket	9:49
10	Mr Nzimande		P loan	10:02	Brown top	10:02
11	Ms Njobweni		2c Julia	10:04	Brown jacket	10:17
12	Miss Moriri		2c Julia	10:11	Scarf	10:11
13	Mr Le Roux		Overdraft	10:14	White yellow	10:24
14	Mrs Nompangisane		New acc	10:23	Fur jacket	10:24
15	Mr Clark		Mona	10:24	Black jacket	10:24
16	Mr Ndadza		Open acc	10:27	Green jacket	10:28
17	Mr Mavuso		2c Maria	10:28	Grey black jers	10:46
18	Mrs Moleya		New acc	10:32	Green shirt	10:32

Figure 1.2 Microsoft Excel 2007 © Spreadsheet

## Open-source software

Open-source software, as implied by the name, gives users access to the software's source code. This allows users to see the source code and change the code to suit themselves. These software programs are therefore flexible and can be customised to any specification using current programming standards that ensure compatibility across platforms and other software applications. Figures 1.3 and 1.4 show examples of open-source software.

Advantages of open-source software include the following:

- Open standards allow the software to be more compatible with other software or between operating systems.
- Users can download open-source software for free.
- Users can also customise the software to suit their purposes.

Disadvantages of open-source software include the following:

- There is no obligatory support, but users can find online support forums where other users may be able to help them solve any problems.
- Updates may be erratic.

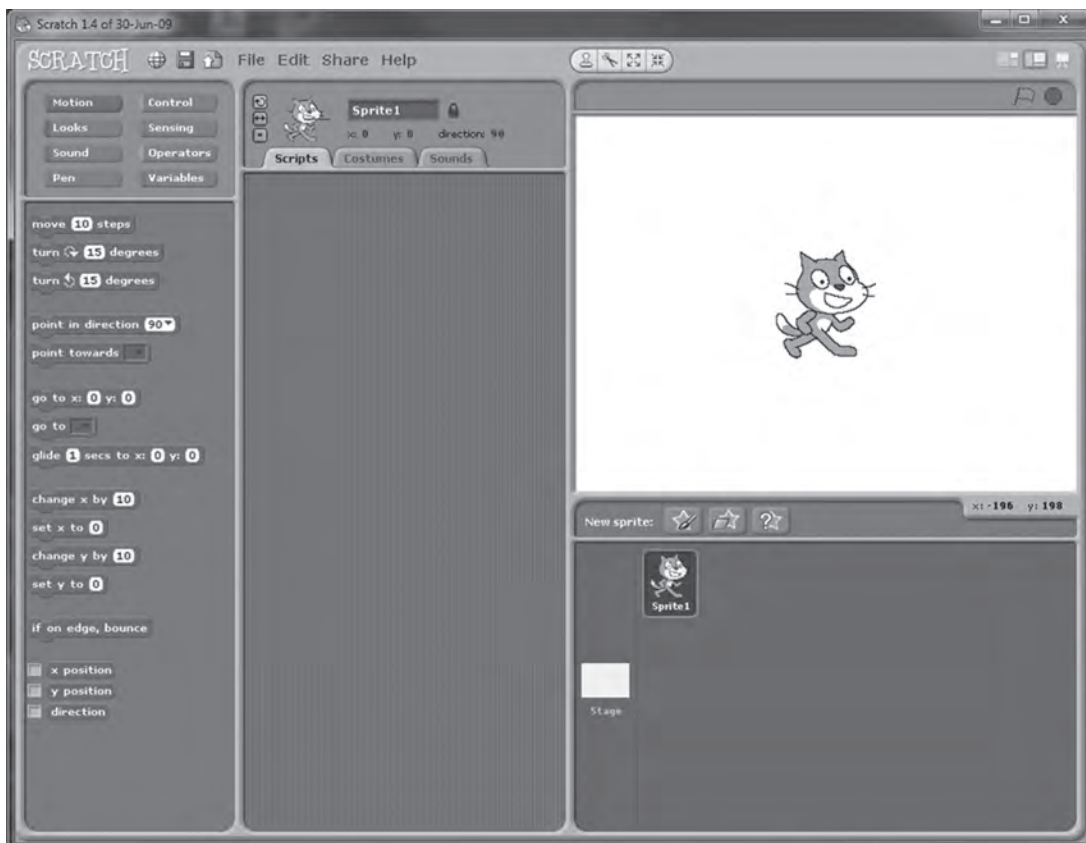


Figure 1.3 Open-source software development tool – Scratch®

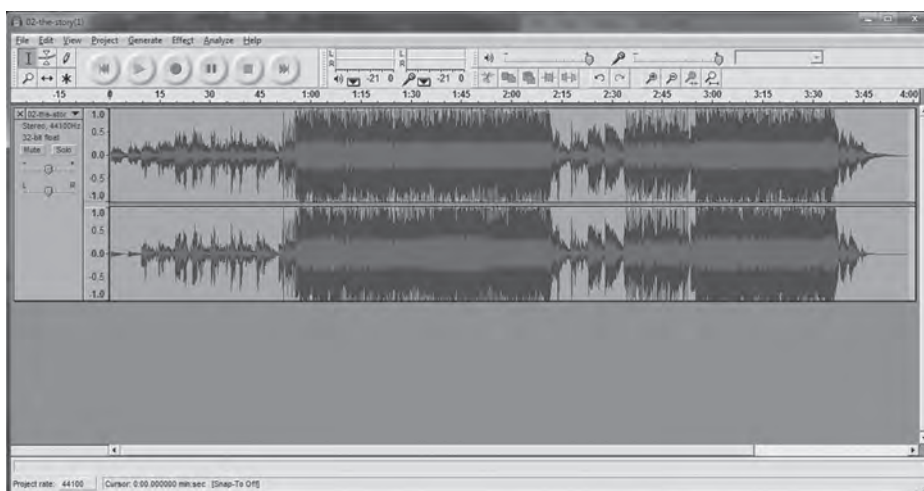


Figure 1.4 Open-source sound editing program®



### Assessment activity 1.1

#### Discussion. Work in groups of at least four.

1. Discuss the differences between open-source and proprietary software-using the following as guidelines:
  - a) Open-source versus closed-source programming code
  - b) Customer support
  - c) Security issues with regard to open-source and closed-source code.
2. Use the internet to make a list of proprietary software programs and their open-source equivalents.
3. Divide your group in two. Download two of the open-source programs you named in question 2. Make sure that you have the propriety equivalents of these programs. Now list the advantages and disadvantages of the programs. One group can evaluate the open-source programs, while the other group evaluates the proprietary software. Try to be objective and do not just look at the price of the software.

## Unit 1.4 Software versions

Before we go into detail regarding different types of software, let us first examine the use of **versions** in software. A version of something is a little different from others of the same type (a new model of a car is a version of the same car).

Few software programs are perfect from the start. Software vendors therefore have to update and improve their software programs continually. This results in different versions of the same software program. The process followed is described below:

Reasons for version changes:

- **Pre-release stage:** When software developers develop a software program, they give each new phase of the project a new version number. They also document the changes made to the version number to keep track of the modifications made during the development.
- **Testing phase:** During the testing phase, the version of the program is referred to as the **beta version**. A beta version of software is supposed to be very close to the final product, but, in practice, it is a way of having users to test the software under real conditions.
- **Post-release phase:** After testing, an **alpha version** of the software program is released with a new version number. When problems or bugs appear, the software developer makes the necessary modifications and then labels the software with a new version number. This process goes on until the end of the software program's life cycle.

Software developers therefore use version numbers to:

- identify different phases in the development of a program and the modifications made during each phase
- identify and document modifications made to a program once it has been released.



#### Words & Terms

**version:** a new version of a software program is an updated form of the same program; differences can be major or minor

**beta version:** a pre-release version of software that has gone through some tests, but is not yet ready to be sold

**alpha version:** software that is ready for the market



### In the workplace

The speed with which operating systems and applications software have been developed and keep on changing has been mind-boggling. However, the changes are often not as far-reaching as the companies who market them would have us believe. For instance, a word processing program you use today will carry out much the same functions as the program you used 15 years ago, although there will have been many 'improved' versions since then.

However, users cannot afford to ignore the changes. If a business does not update its software regularly, it will find that, after a few years, the software is completely out of date and people in other companies can no longer read their documents.

When software developers give different version numbers to the software to keep track of all the changes made to programs, they use different ways of numbering their developments. Here is a list of some version schemes that developers use:

- **Year of release:** Use of the year in which the software was released, for example Macromedia Dreamweaver 2004
- **Alphanumeric codes:** Use of alphabetic and/or numeric values, for example Adobe Photoshop CS2
- **Roman numerals:** Use of roman numerals, for example Apple's Mac OS X
- **Numeric:** Use of numeric values, for example Winamp 5.0.8.

The most popular scheme, and one we will discuss in more detail here, is the numeric version scheme. The advantage of a numeric scheme is that it is flexible and it gives information about the software's history. The development in a numeric version scheme will typically be as follows:

- A software program must first be tested before it can be released. The first version will be 0.1.0, also known as the beta version. As testing progresses, so the versioning continues up to the point of release.
- When the software is released, the version becomes 1.0.0. This is the first alpha version. When a few bugs are fixed during maintenance work, the version becomes 1.0.1. When a minor improvement is then made to the version, it becomes 1.1.1. Finally, when a major update is done, it becomes version 2.0.0.

As the developers continue to improve a software program with minor or major updates, the version will keep us up-to-date on how it developed over time. An easy way to remember numeric versions is by remembering this simple equation: *Major.Minor.Maintenance*.

### ?? Did you know?

Microsoft has used different version schemes to denote the releases of their operating system software, Windows. The first Windows had a numeric scheme – Windows 1.0 to Windows 3.11. Later, this changed to Windows 95, Windows 98 and Windows 2000, using the year of release scheme. Finally, it changed to the alphanumeric scheme – Windows ME and Windows XP.



## Assessment activity 1.2

### Work on your own.

1. a) What is computer software? (3)  
b) What is computer hardware? (2)
2. Explain the difference between high-level and low-level programming languages. (3)
3. Name three types of software and briefly explain what each does. (6)
4. a) Explain the differences between open-source software and proprietary software. (4)  
b) List the advantages and disadvantages of each of these two different types of software. (10)
5. Create your own software program with a unique name and complete the following:
  - a) Name four beta versions.
  - b) Name four maintenance releases.
  - c) Name four minor releases.
  - d) Name three major releases using the numeric versioning scheme. (15)
6. Explain in one paragraph why software developers use version schemes for software. (4)
7. List four version schemes that developers use and explain them each in one sentence. (4)
8. Name two advantages and two disadvantages of using versioning schemes. (4)

**Five marks** for the following:

**Neatness:** whether written by hand or typed, the print should be neat

**Creativity:** any use of colour, pictures and individual thought

**Elements of planning:** due date met, headings in a different style or colour, logical order of answers, relationship of parts to each other.

**Total marks: 60**

# Module 2

## Application software

### Overview

At the end of this module, you should be able to:

- identify and demonstrate the different features common to all types of application software
- list and describe different types of application software and their use
- explain the purpose and use of each of the types of features common to all types of application software
- outline the processes for installing application software.

## Unit 2.1 Features common to all types of application software

### What is application software?

**Application software**, also called end-user programs, performs a specific task for the end user. By way of contrast, **system software** is computer software designed to operate the computer hardware and to provide a platform for running application software. There are many types of application software, such as media players, games, spreadsheets and word processors. Each type of application software performs a certain task for the user, whether for recreational use or for use in business applications.

Multiple applications bundled together is referred to as an **application suite**. Microsoft Office is a good example as it includes a word processing program, MS Word, a spreadsheet program, MS Excel and other programs that are bundled together into one application suite. Each program is used in a similar way, making it easier to learn and use the application software. An advantage of a suite is that the programs in the suite can interact with each other, for example a spreadsheet from MS Excel can be embedded in an MS Word document. Application software cannot run without an operating system (discussed in Module 4). Figure 2.1 shows an example of application software (Corel PaintShop photo Pro X3).



### Words & Terms

**application software:** performs a specific task for the end user; also called an end-user program

**system software:** operates the computer hardware and provides a platform for application software

**application suite:** multiple applications bundled together

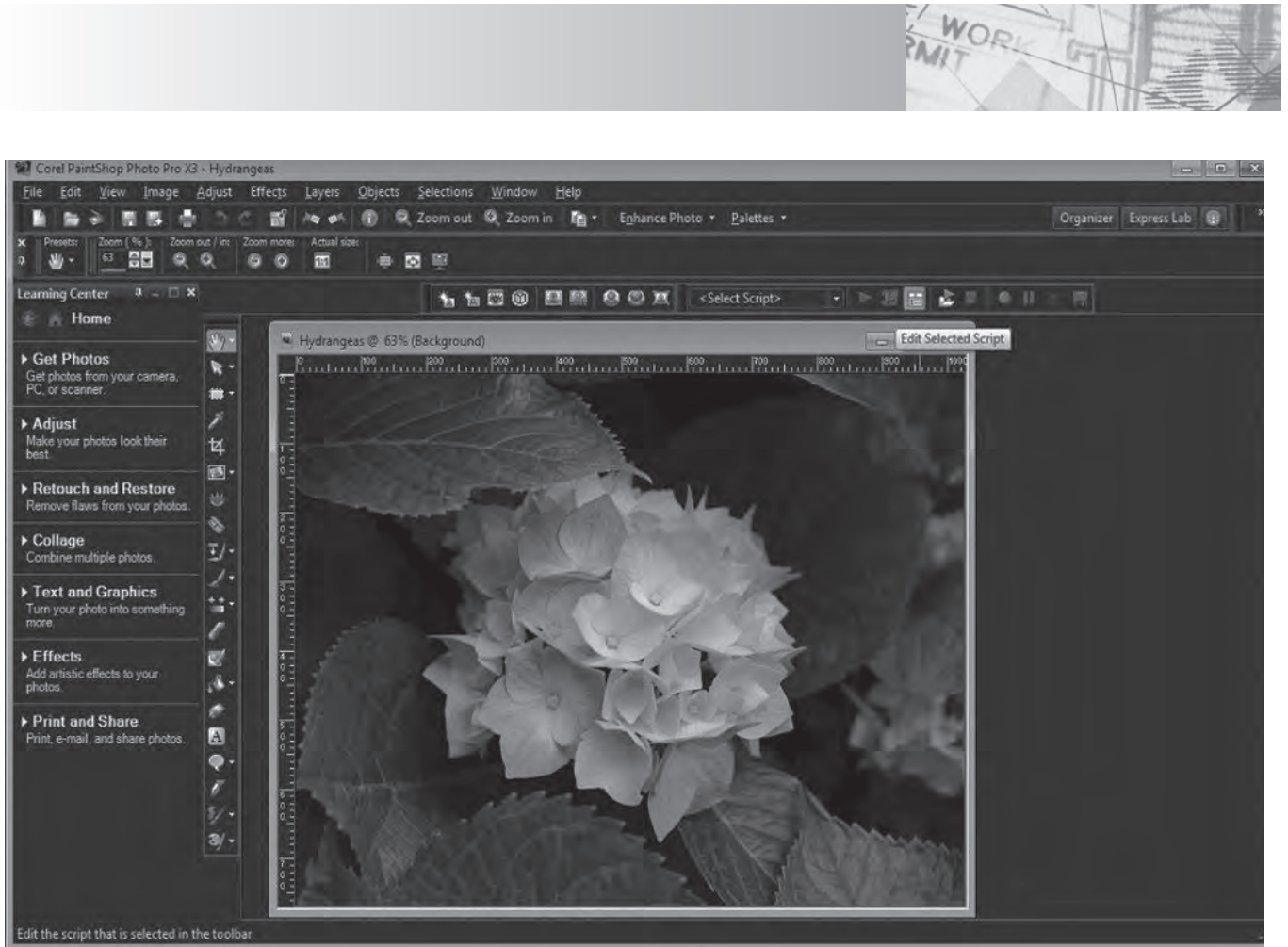


Figure 2.1 Example of application software

## Features of application software

Figure 2.2 shows the toolbars of four programs from the Microsoft Office 2007 suite. Just by looking at them, the user can see that they share common features. A **feature** is a distinctive attribute or aspect of a program. These programs were developed by the same company and are designed to look the same. This not only makes it easier for users to learn a new program, but it is also easier to take a document from one program and use it in another program.

Other software development companies also use the same style of menus as shown in Figure 2.2. The reason for this is that these companies know that most of their users probably have one of these programs. By using a menu with which users are familiar, they make it easier for new users to learn their own programs. Section 2.3 explores features common to application software in more detail.

### Words & Terms

**feature:** a distinctive attribute or aspect of a program



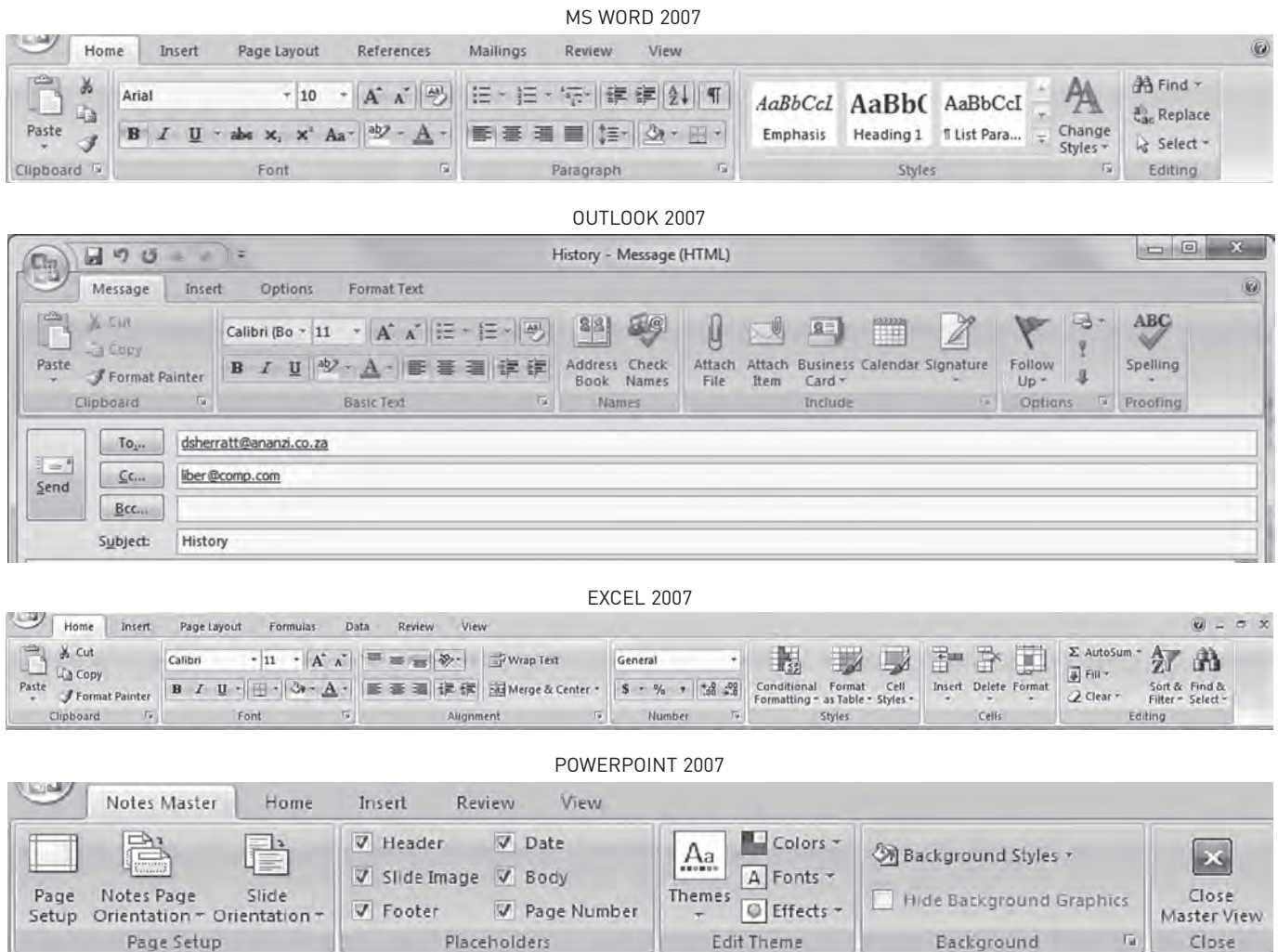


Figure 2.2 Menus of programs in the Microsoft Office 2007 suite

## Unit 2.2 Different types of application software and their use

### Word processing software

Word processing software enables users to create, edit and store a document electronically. Users can save their documents on a computer hard drive, a CD or DVD, or a flash drive. The advantages of a word processing program are that you can edit your work, delete mistakes, move sections of text around, and insert new words and sentences. When you have completed your document, you can save it for future reference, print it on a printer to obtain a hard copy or send it to a recipient via email.

Word processing programs, or word processors, vary considerably, but they all support the following basic features:

- **Insert text:** Allows you to insert text anywhere in the document.
- **Delete text:** Allows you to erase characters, words, lines or pages by pressing the backspace or delete keys.



- **Cut and Paste:** Allows you to remove or cut a section of text from one place in a document and insert or paste it somewhere else.
- **Copy:** Allows you to duplicate a section of text or copy a section of text from another document.
- **Page size and margins:** Allows you to define various page sizes and margins. The program will automatically readjust the text so that it fits onto the page if your margins are too big or small. This helps when working with different sizes of paper.
- **Search and replace:** Allows you to direct the word processor to search for a particular word or phrase. You can also replace one group of characters with another wherever that the first group appears.
- **Word wrap:** The word processor automatically moves to the next line when you have filled one line with text, and it will readjust text if you change the margins.
- **Print:** Allows you to send a document to a printer to obtain a hard copy.

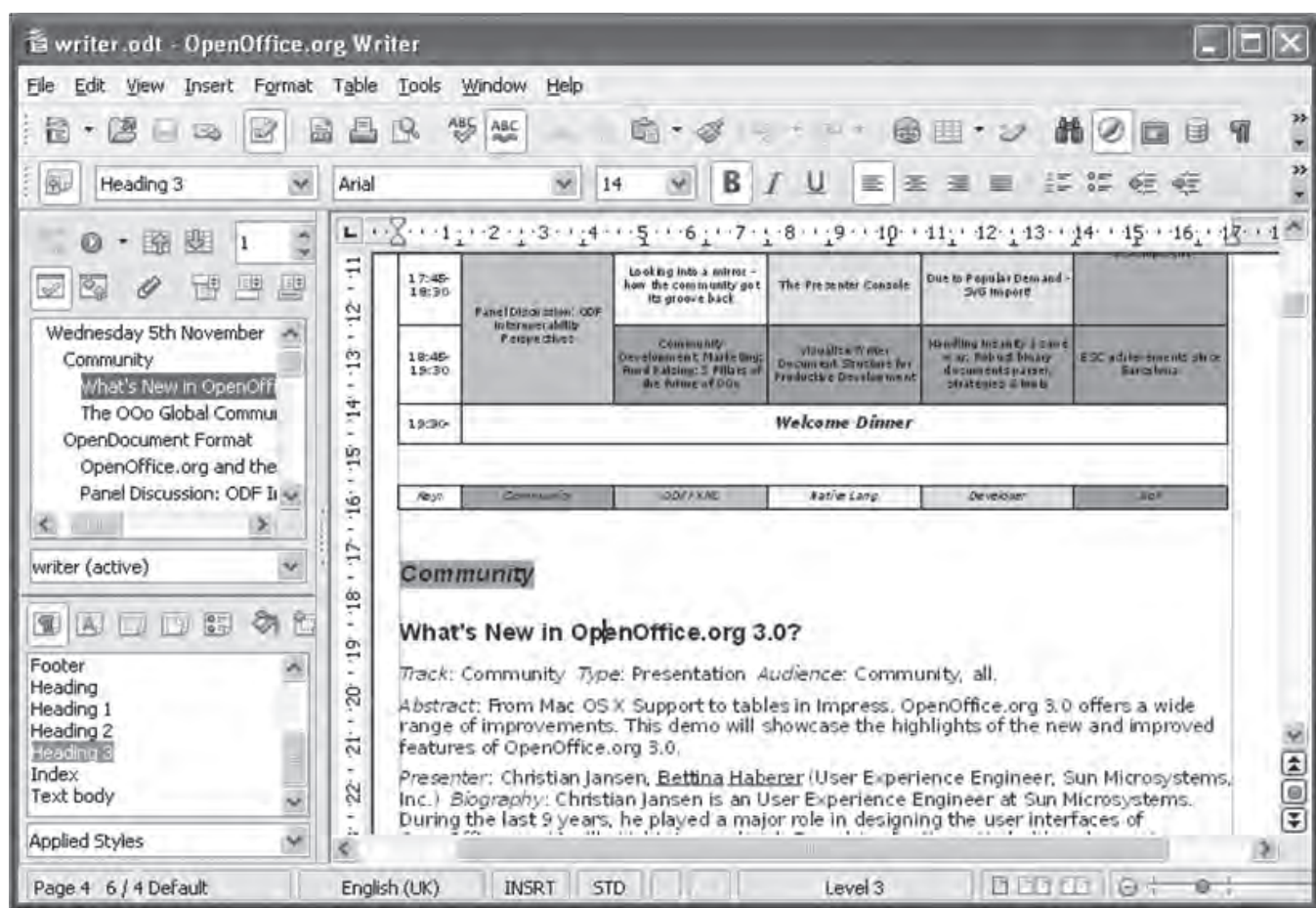
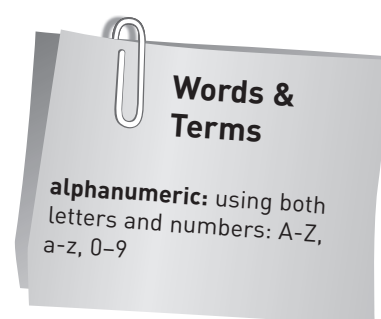


Figure 2.3 A word processing application – OpenOffice.org

Word processors that support only these basic features listed above are called text editors. Most word processors, however, support additional features that enable you to manipulate and format documents in more sophisticated ways. These more advanced word processors are sometimes called full-feature word processors. Full-feature word processors usually support the following added features:

- **File management:** Word processors have file management capabilities that allow you to create, delete, move and search for files.
- **Font specification:** Allows you to change fonts in a document. For example, you can specify bold, italics and underlining. Most word processors also let you change the font size and the typeface.
- **Footnotes and cross-references:** Automates the numbering and placement of footnotes and enables you to cross-reference other sections of the document.
- **Graphics:** Allows you to embed illustrations and graphs into a document. Some word processors let you create the illustrations within the word processor; others let you insert an illustration produced by a different program.
- **Headers, footers and page numbering:** Allows you to specify customised headers and footers that the word processor will put at the top and bottom of every page. The word processor automatically keeps track of page numbers so that the correct number appears on each page.
- **Layout:** Allows you to specify different margins in a single document and to specify various methods for indenting paragraphs.
- **Merges:** Allows you to merge text from one file into another file. This is particularly useful for generating many files that have the same format but different data. Generating mailing labels is a classic example of using the merge feature.
- **Spell checker:** This allows you to check the spelling of words. It will highlight any words that it does not recognise.
- **Windows:** This feature allows you to edit two or more documents at the same time. Each document appears in a separate window. This is particularly valuable when working on a large project that consists of several different files.
- **WYSIWYG (what you see is what you get):** With WYSIWYG, a document appears on the screen exactly as it will look when printed.

Different types of word processors include WordPad and MS Word. An open-source option is Open Office's word processor.



## Spreadsheet software

Spreadsheet application software enables the user to create a worksheet in a two-dimensional matrix or grid containing rows and columns. Each cell in the grid can contain **alphanumeric** text, numeric values or formulas. The formula defines how the content or value in each cell can be calculated or defined. For example, you can insert values in two cells and use a formula to multiply the two values and insert the answer in a third cell.

Most spreadsheets also allow for the creation of graphs and charts from the values in the cells. As the values in the cells change, they will automatically change the graph or chart as well. Spreadsheets are mostly used for accounting purposes because of their ability to recalculate the entire sheet automatically after changing one or more values. They are also used for a wide range of other applications where calculations are required.



Most spreadsheet applications also allow users to link data from different spreadsheets. For example, you can link all the monthly income/expenditure spreadsheets for a 12-month period to create one spreadsheet containing all the totals. You can then use graphs to give an overview of the year's income and expenditure.

As with word processing software, you can also save spreadsheets electronically to a hard drive, CD or DVD, or a flash drive. You can print and email spreadsheets. Many of the features listed above for word processing software are also applicable to spreadsheets.

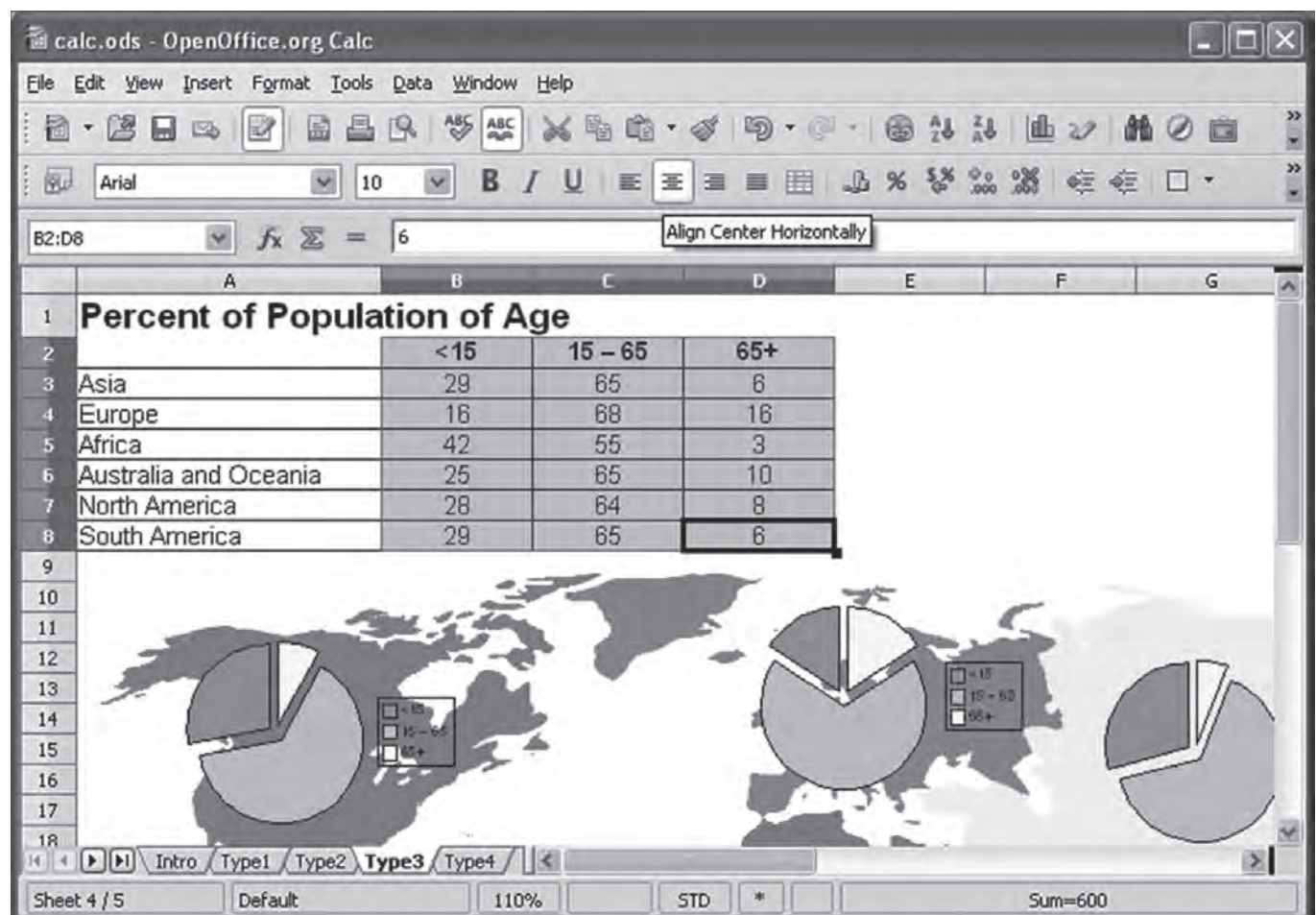


Figure 2.4 A spreadsheet application – OpenOffice.org

Different types of spreadsheets include MS Excel and Lotus 123. OpenOffice Calc is an example of an open-source application.

## Presentation software

Presentation application software is used to create and display information in a slideshow format. The program can create slides combining text, images, graphs and audio/video clips. The program can then manipulate the transition between slides to create a slideshow (see Figure 2.5).



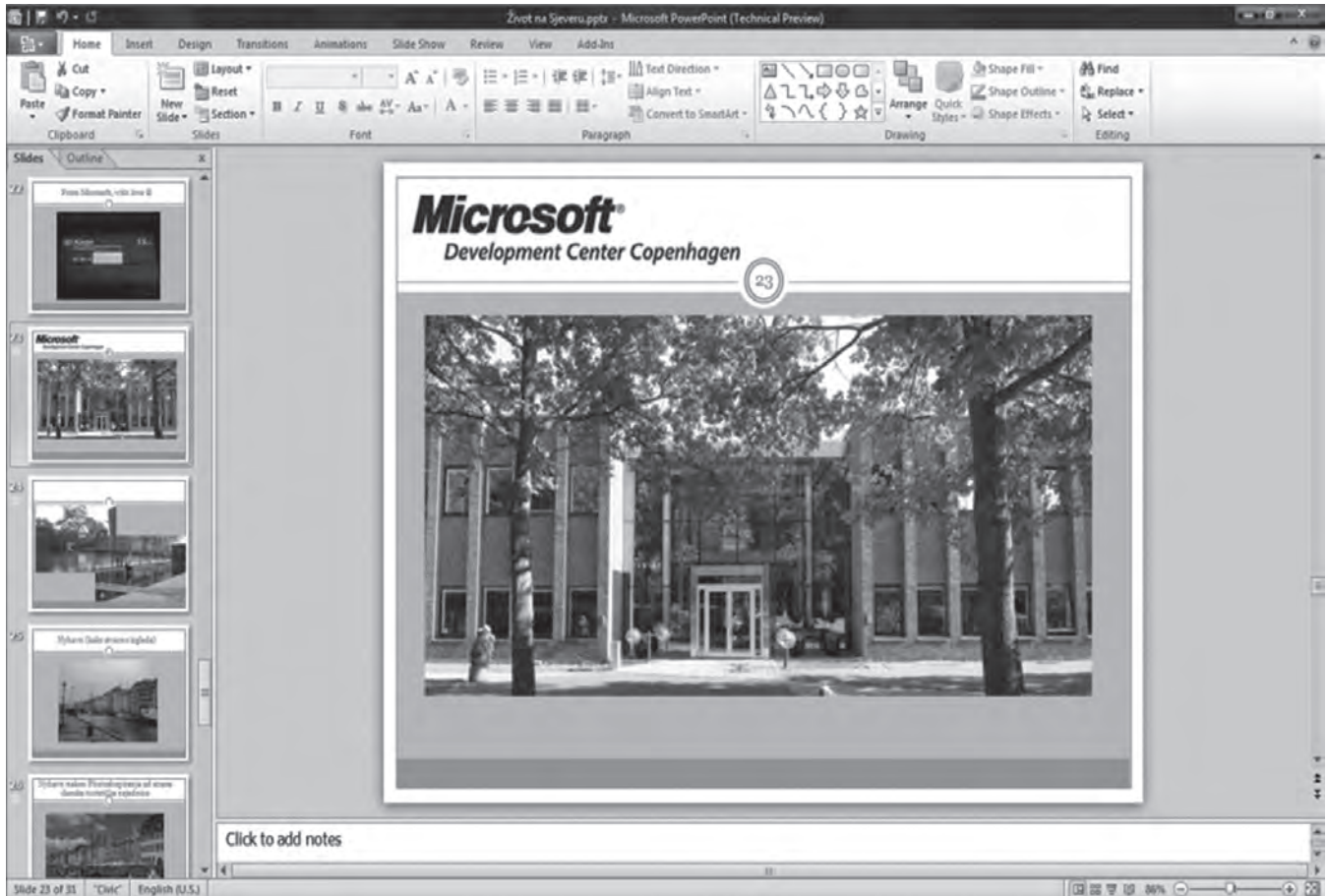


Figure 2.5 Microsoft PowerPoint 2010®

People use presentation software to help with visual aspects when delivering a presentation, lecture or speech. Business people use it to show graphs and statistics. Today, the most common way of displaying presentations is by means of a video projector. An example of presentation software is MS PowerPoint. An open-source version is Open Office Impress.

## Database software

Database application software, often abbreviated to DB, is an application designed to store, organise and manage large amounts of data easily. Traditional databases are organised by fields, records and files. A **field** is a single piece of information, for example your student number. A **record** is a collection of fields. For example, the details on your college registration form, such as your name, surname and ID number, would be separate fields. Together, these fields form a record of your personal details. A **file** is a collection of records. We can describe it as a book containing all the details of all the students.

To manage or access the information in a database, you need a program that enables you to enter data, organise the data and retrieve information from the database easily and effectively. To do this, a database needs a database management system (DBMS). The DBMS

### Words & Terms

**field:** a single piece of information

**record:** a collection of fields

**file:** a collection of records



can retrieve information from the database as required by the user through queries or reports. This allows the user to retrieve only that information required to perform a certain task. For example, if a student wants to find out about only his/her final results from the previous semester, the student counsellor just has to run a query for that specific information and not the student's entire academic record. Examples of database software include MS Access, Oracle and SQL (see Figure 2.6).

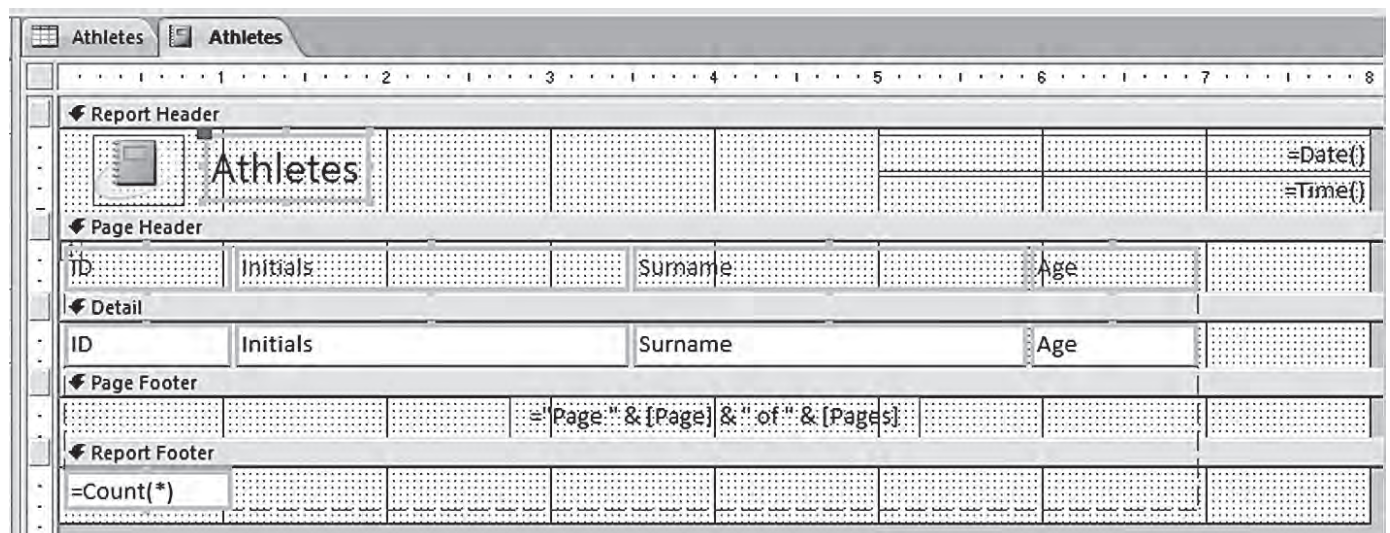


Figure 2.6 A database application – Microsoft Office 2007 Access®

## Multimedia software

The term 'media' traditionally refers to forms of printed material. Multimedia, however, is a combination of text, audio, video, animation and still images. If you take a piece of text and add music and a still image next to the text, you have created a basic multimedia file. Multimedia applications allow you to create, edit, record and play various forms of multimedia depending on which multimedia application you use.

Different types of multimedia applications are MS Media Player, Real Video, Win-Amp, Adobe Flash and Apple iTunes.

### Think about it

Gone are the days when web pages just had text. Today, most web pages contain various forms of multimedia. These include audio files, video clips, animations and **Adobe Flash**, commonly known as Flash. Some Flash animations allow for interaction – this is called interactive media.

### Words & Terms

**Adobe Flash:** multimedia software that adds animation, video and interactivity to web pages.

### Assessment activity 2.1

#### Work on your own.

1. Define application software and an application suite in your own words. (4)
2. Give two examples of where you would use a word processing application in your daily life. (4)
3. Pastel/Lotus 123/AutoCad/Norton AntiVirus/MediaPlayer/Visual Studio 2005/Power Point  
From the list above, choose the program you would use to do each of the following. You could possibly use one program for more than one application:
  - a) Draw a plan of your dream home.
  - b) Plan the budget for building your dream home.
  - c) Protect your computer from viruses.
  - d) Listen to your music on the computer.
  - e) Create a new program for your company.
  - f) Create a materials list with prices for your dream home.
  - g) Give a presentation of a new venture at your company. (7)

Total marks: 15

## Unit 2.3 Purpose and use of features common to types of application software

There are many different types of application software available for the end user, but all share some common features. In this section, we shall discuss some of the more general features common to all application software.

### In the workplace

In practice, most people in business do not create their own programs, but work with Microsoft Word for word-processing purposes (for example when writing a document), and with Microsoft Excel for spreadsheet purposes (for example when drawing up a budget). Other applications that are widely used are Microsoft Outlook for emailing and Microsoft Explorer for internet browsing. Microsoft PowerPoint, a presentation program, is also widely used. For anything beyond these standard applications, a business must either invest in more expensive applications or commission a software development company to write a customised program.

The fundamental reasons why various software applications share the same features are:

- usability
- functionality
- productivity
- compatibility.





## Usability

**Usability** refers to how user-friendly or easy a program is to use. When end users buy software, they want a program that is easy to use with a simple interface to perform the tasks they want to do. Usability thus denotes the ease with which people can perform a certain task by using a particular tool.

## Functionality

**Functionality** is the set of product features and functions of a software application and its functions. However, functionality does not stop at what the software application is capable of doing. If a product has all the required capabilities, but takes hours to perform a simple instruction, is it still functional? If the user wants to perform a task, but is not sure how to do so, is there support to help the user? Software functionality is more than just the application's ability to perform a given set of instructions; it also refers to how easy it is to perform that given task.

How does one improve functionality in an application? The **graphical user interface (GUI)** is a type of user interface that allows users to interact with the computer with images rather than text commands. A GUI uses images, icons and text to represent the information and actions available to the user. The user interacts with the computer by manipulating the images and icons to perform the desired actions. Most users are not interested in the source code of the software application – they want an interface with a good layout and easy-to-use functions. The interface should not be cluttered with unnecessary information. The user interface is responsible for the efficient input of data and clear and understandable output from the system.

**Help tools** and **online support** are the backbone of all modern software packages:

- A **help tool** is an onscreen instructional program that provides further information about how a function works. It tells you how and when to use a specific instruction. Modern help tools are sophisticated, with an index and search function, making it easy for the user to understand and apply any function that they do not understand. You can press **F1** on the keyboard in Microsoft Windows operating systems and applications to call up the help function.
- **Online support** and **live support** give extra information when the help tool is inadequate. Online support is support for software supplied over the internet. It normally has a page listing frequently asked questions (FAQ) and extra information on the product's functions. When this is still inadequate, users can call the live support call centre where qualified staff help users with their problems by explaining what to do in a step-by-step manner.

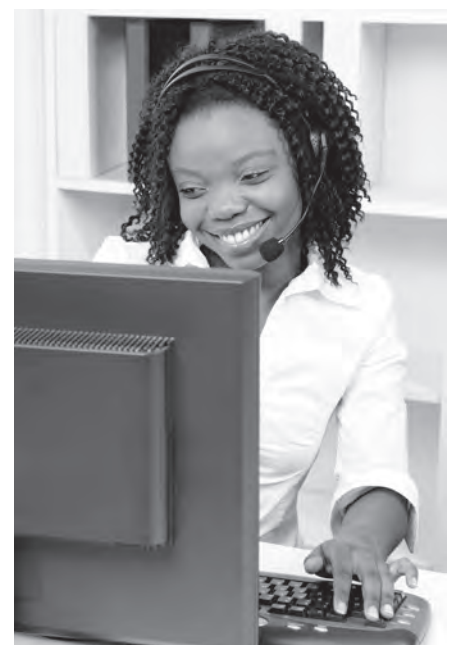
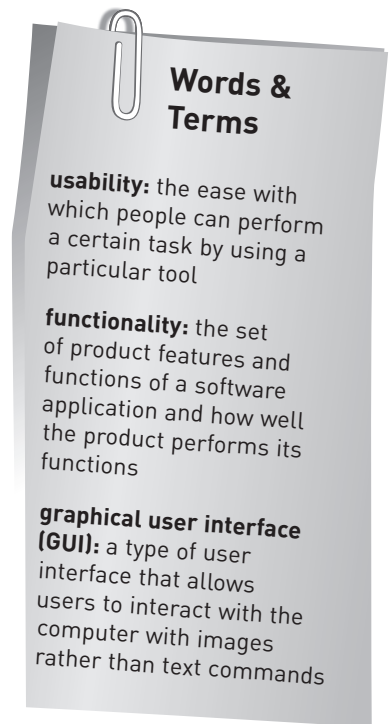


Figure 2.7 A live support call centre operator



## Productivity

Equipment is productive when it achieves a significant result. In software terms, we say software is productive when it creates a large amount of output for every unit of input – simply put, this occurs when you get more out than you put in. A good software product that is easy to use and has functions that increase the user's **productivity** results in satisfied clients.

What types of functions create better productivity? Software packages have different functions, but they do share some general features. We shall point out a few of them to give you a perspective on how they simplify the end user's life:

- A well-designed interface with easy-to-use functions will increase a user's productivity immensely.
- A **toolbar** is a bar with tools or functions that enable the user to be more productive. A toolbar contains a cluster of functions in groups. When the user clicks on a tool, it performs a certain task.
- **Shortcut keys**, like toolbars, are an aid to help the user perform certain tasks more quickly by using a combination of keys. For example, by pressing *Ctrl* + *S* in a Windows-based application, the user saves the document he or she is currently working on.

## Compatibility

**Compatibility** is the ability of a software package to work on different types of system software, to interact with different software packages and, most important of all, to run on different types of hardware configurations. When designing a software program, a software developer must take compatibility into account – not just compatibility with hardware, but with other software too (system software as well as other applications).

In software development there are two important types of compatibility, namely backward compatibility and forward compatibility:

- Backward compatibility (or downward compatibility) is the ability of a software package to interact with older versions of the same product. It refers to the relationship between two components rather than being an attribute of just one of them.
- Forward compatibility is the ability of a software package to accept instructions and data formats in later versions of itself.

### Words & Terms

**help tool:** an onscreen program that provides further information about how any functions in an application work; it tells you how and when to use a specific instruction

**online support:** support for software supplied over the internet

**live support:** a qualified person who gives advice directly over the phone

**productivity:** the amount of output created per unit of input

**toolbar:** a bar with tools or functions that enable the user to be more productive

**shortcut keys:** an aid to help the user perform certain tasks more quickly by using a combination of keys

**compatibility:** the ability of a software package to work on different types of system software, to interact with different software packages and to run on different types of hardware configurations

## In the workplace

Samantha works at a design company specialising in graphic design and image manipulation. The company uses different application software for various projects. A client comes to Samantha and asks her if she could help him manipulate a photo of his father into a drawing, which the client wants to frame as a birthday present for him on his 80th birthday.

Samantha tells the client about a new application that takes a raster image and converts it into a vector image, thus creating the illusion that it is a drawing. The client is excited and gives Samantha the photo of his daughter.

After Samantha scanned the photo, she opened the program. After loading the client's image, she used the software to create a vectorised model of the client's photo. You can see the result in Figure 2.8.



Figure 2.8 The original scanned photo and geometric model using vector graphics

### Assessment activity 2.2

1. Explain the concepts application software and application suite. (4)
2. Choose an item from Column B to match a description in Column A. Write only the letter (A – L) next to the question number (2.1 – 2.10). (10)

	Column A		Column B
2.1	Microsoft Word	A	Application programs
2.2	Microsoft Excel	B	For browsing the World Wide Web
2.2	Microsoft 7/Vista/XP	C	For making a list of CDs in a collection
2.4	Norton, AVG, Panda	D	For sending/receiving emails
2.5	Microsoft Outlook	E	For typing a letter or CV
2.6	Microsoft Access	F	Operating systems
2.7	Internet Explorer	G	Utility programs
2.8	Calculator, System Restore, Disk Defragmenter	H	For creating a presentation
2.9	Microsoft Powerpoint	I	For preventing viruses, spam
2.10	Smartdraw, Adobe Reader, Quicktime	J	For doing calculations and drawing graphs
		K	For making drawings
		L	Sound editing

Total marks: 14

## Unit 2.4 Installing application software

### What is software installation?

**Software installation** is an automated process in which the user answers questions asked by an **installation wizard**. This program installs the software application by responding to input from the user. The user manual normally has detailed instructions on how to execute the installation process.

### The installation process

A program consists of many files. When the program is installed, these files are stored on the computer's hard drive. The program does not use all these files while running, or at **run time**. Some files will be used for a short period only and then they will be released from memory. These files are called **run-time modules** or **dynamic linked library (DLL) files**. The printer driver is an example of a DLL file. Some of these files may need to be located in different directories because they are shared by different programs.

#### Words & Terms

**software installation:** an automated process in which the user answers questions asked by an installation wizard

**installation wizard:** a utility program that installs the software application by responding to input from the user

**run time:** the period during which a file is being used

**run-time modules or dynamic linked library (DLL) files:** files that are used for a short period only and are then released from memory, for example the printer driver



During the installation process, a program directory is created to store the main files of the program. The registry may need to be updated with file references. Computers that run on the Windows operating system have a **Windows registry**. This is a database that controls virtually every aspect of your computer's operation. Database files also need to be created. Because users do not know exactly where each file should be stored, the installation process is therefore automated by means of an installation wizard.

### Words & Terms

**Windows registry:** a database that controls virtually every aspect of your computer's operation and that is found on all Microsoft Windows operating systems

## Installing a software program

### a) Welcome screen

Figure 2.9 shows the Welcome screen. This screen allows the user to exit the installation process before it starts. If the user selects Cancel, the installation process is aborted. To continue, click on Next.



Figure 2.9 Welcome screen

### ?? Did you know?

The installation process of a new program is controlled by one file, the SETUP.EXE file. The purpose of this file is to do most of the thinking during the installation process for you. When the user double-clicks on this file, the program installation starts and the wizard asks the user a series of questions.

### b) Licence agreement

Figure 2.10 shows the **licence agreement**. The user uses the scroll bar on the right-hand side of the screen to view the agreement. The licence agreement offers the user the option of either accepting the terms and conditions of the software company or rejecting them. If the user decides to reject the agreement, the installation process will be aborted. If the user accepts the agreement, the Next button becomes active.

### Words & Terms

**licence agreement:** a contract between the user and the software development company



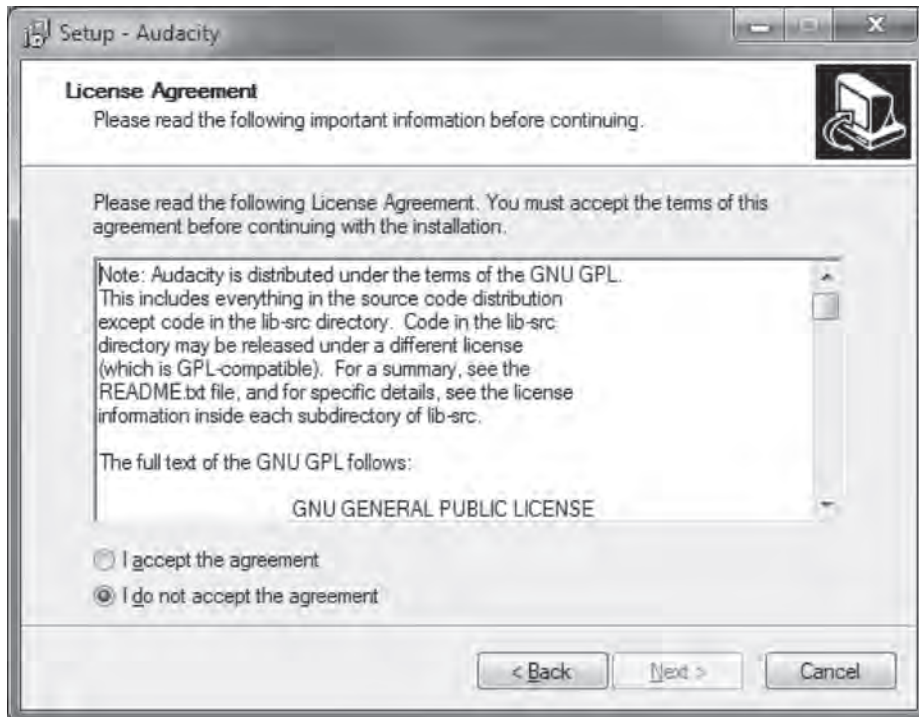


Figure 2.10 Licence agreement

### c) Information screen

Figure 2.11 shows the information screen. In some cases, the installation process offers more information, such as the email address of the developer of the program. Click Next to continue with the installation.

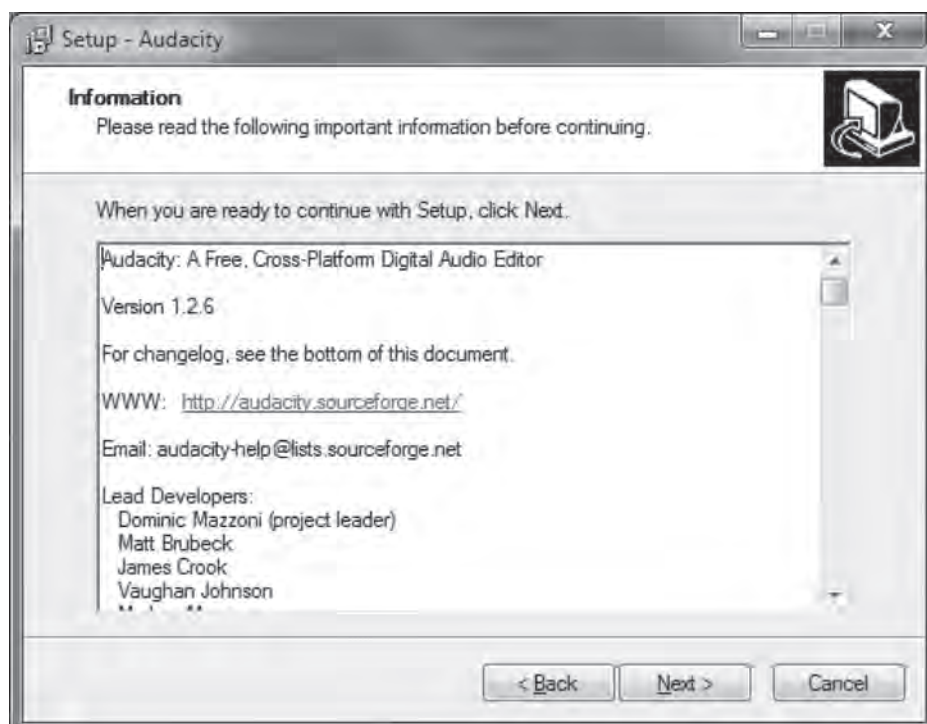


Figure 2.11 Information screen



#### d) Setup destination

Here, you need first to understand the difference between 32-bit and 64-bit machines. To explain this, think of a single-lane road on which cars may travel at 120 km/h. However, if there are many cars, then they will all have to slow down to, say, 100 km/h. If there is another lane, the traffic flow will be better and many cars will be able to travel at 120 km/h. A 64-bit machine means that there are 64 lanes for cars to travel in. The newer machines are all 64-bit machines. Unfortunately, some software does not know how to take advantage of the increased number of lanes. A 64-bit machine with Windows 7 helps programs to use all the lanes to full advantage and run faster.

Figure 2.12 shows the Setup destination screen. The directory where installation will take place is now determined. The user can click on the Browse button to find an alternative directory in which to install the program. If a 32-bit program is being installed on a 64-bit machine with Windows 7, the installation will take place in the Program Files (x86) directory.

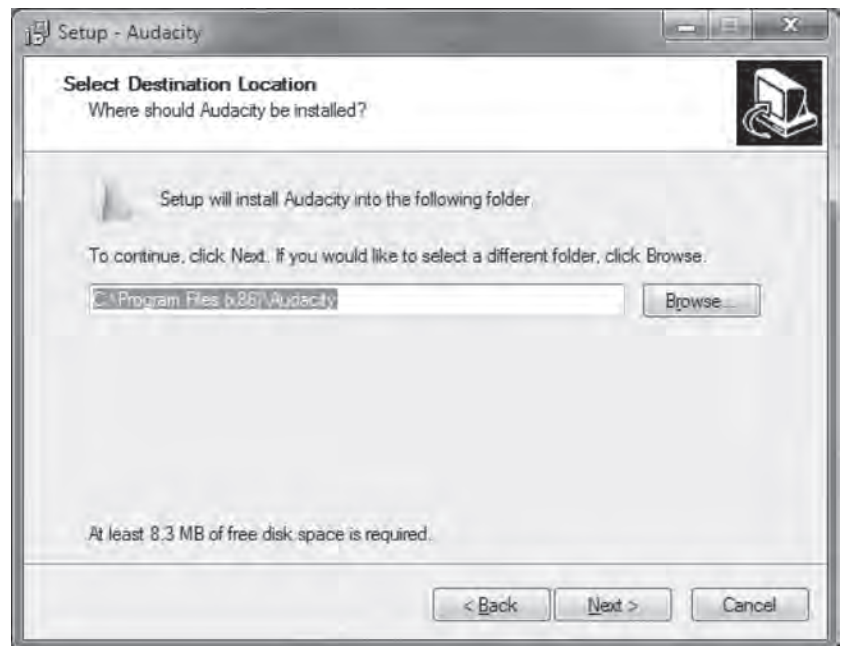


Figure 2.12 Setup destination screen

#### e) Folder exists warning

Figure 2.13 shows the Folder Exists warning. If the program is being reinstalled, the wizard will warn you that the directory already exists.

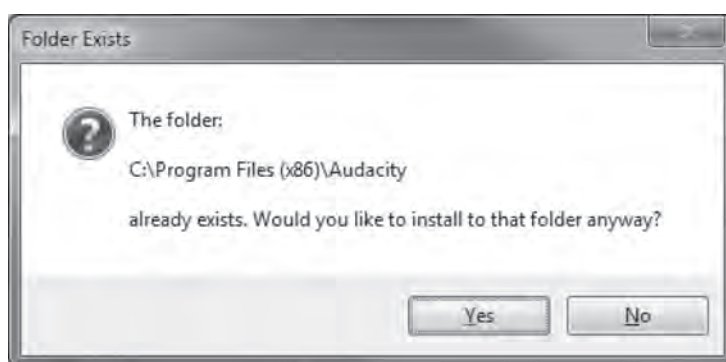


Figure 2.13 Information screen

#### f) Additional tasks

The user can perform additional task such as:

- creating desktop icons
- creating a program group in the Windows startup folder
- creating an uninstall program
- associating project files
- creating a quick-launch button on the taskbar.



Figure 2.14 Additional tasks

#### g) Ready to install

The wizard now informs the user that it has gathered enough information to do the installation. When the user clicks Install, the wizard will install the program. If the user clicks Cancel, the installation will be aborted.

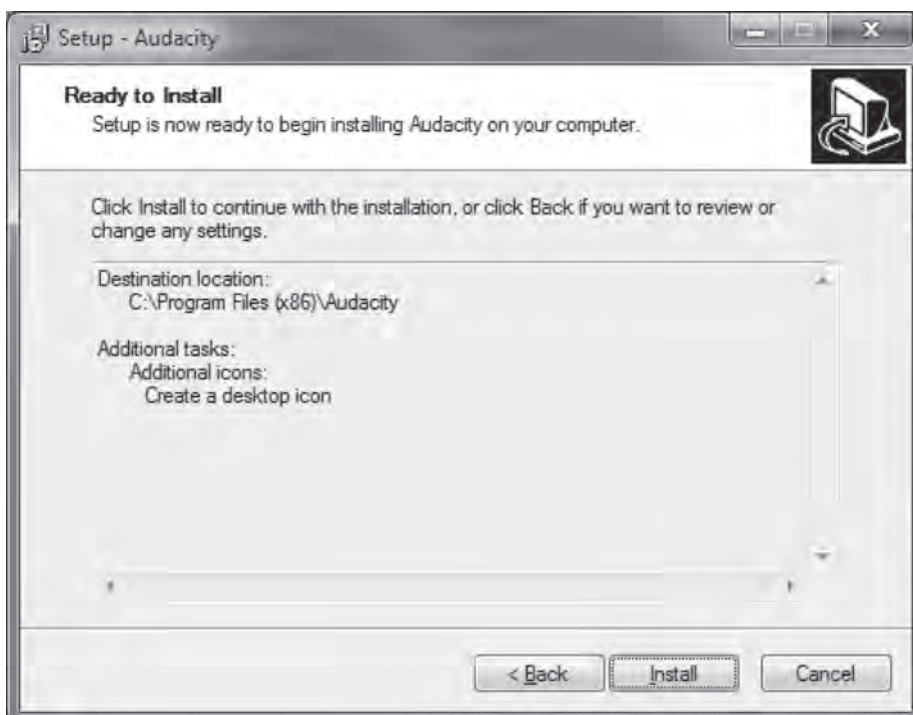


Figure 2.15 Ready to install



## h) Installation complete

After installation, the wizard informs the user of the successful installation. Click on the Finish button to end the wizard and run the program.



Figure 2.16 Completed installation

### Assessment activity 2.3

#### Work on your own.

1. a) Choose a specific application software package and identify four core features of this software. (4)  
b) Explain the purpose and use of these four core features. (4)
2. Explain the installation process. (4)
3. Consider at the word search puzzle below and circle those words in pencil that would best explain the following descriptions. The first answer is given as an example.
  - a) We use this to give instructions to a computer to do certain things.
  - b) The components of a computer that I can touch are called the \_\_\_\_.
  - c) When we talk to or program a computer, our human language must be converted into a language that a computer understands, so we need a \_\_\_\_ program.
  - d) The main processor where all the thinking is done.
  - e) The memory that is volatile and loses its information.
  - f) The memory that does not lose instructions when the power goes off.
  - g) The programs we create are also called \_\_\_\_.
  - h) These programs help us make the computer work better.
  - i) When you buy software, it is called \_\_\_\_ software.
  - j) When a program can work on most types of systems without problems.



- k) When a person helps us with problems we have with a computer.
- l) When we give programs new names and numbers (e.g. Vista, XP, Version 2) to know which program we are dealing with \_\_\_\_ .
- m) We can use this program to do maths and statistics (Microsoft Excel is an example of these programs).
- n) This software is used when giving lectures and presenting slides.
- o) This software is free and has no copyright.
- p) A type of memory.
- q) Use this program to store information (Microsoft Access is an example of these programs).
- r) Using sound, videos and pictures in a program.
- s) When a program is functional, we talk about its \_\_\_\_ .
- t) Is the program going to work? Can I use it to solve my problems?
- u) Will this program make the solution of my problems faster?
- v) Can a new program work with old data from my old program?
- w) Can an old program read the data of my new program?
- x) When we place new programs on a computer using a setup program.

I	R	O	T	A	L	S	N	A	R	T	A	A	H	G	E	Y	A	C	P	U	W
N	O	I	T	A	L	L	A	T	S	N	I	Z	Q	Y	V	J	K	S	P	A	M
D	R	C	G	K	K	R	J	O	P	T	T	W	H	F	Y	V	V	X	Z	C	U
R	H	D	A	U	H	W	N	M	Y	R	A	T	E	I	R	P	O	R	P	O	L
A	D	V	Y	T	I	L	I	B	A	S	U	T	G	X	D	G	K	K	V	M	T
W	U	I	D	K	O	O	I	U	A	P	P	L	I	C	A	T	I	O	N	P	I
R	H	M	H	U	O	P	E	N	S	O	U	R	C	E	R	K	A	R	Z	A	M
O	I	I	Y	T	I	L	I	B	I	T	A	P	M	O	C	T	H	H	B	T	E
F	Y	R	O	M	E	M	S	S	E	C	C	A	M	O	D	N	A	R	C	I	D
P	J	H	B	A	C	K	W	A	R	D	L	B	A	W	B	W	F	H	G	B	I
Z	B	T	D	I	P	R	E	S	E	N	T	A	T	I	O	N	B	X	D	I	A
D	A	T	A	B	A	S	E	C	I	J	L	J	W	H	M	H	L	T	Q	L	Q
F	D	W	Z	V	C	N	N	R	U	P	G	I	Q	Q	B	Y	C	A	L	I	K
J	T	E	K	B	P	E	Q	L	G	Y	T	I	L	I	T	U	Z	W	E	T	Y
N	B	G	V	S	O	F	T	W	A	R	E	O	V	J	P	T	Q	A	G	Y	M
X	Q	P	N	D	O	G	B	T	P	R	O	D	U	C	T	I	V	I	T	Y	M
F	O	G	N	I	N	O	I	S	R	E	V	F	F	V	T	N	V	Z	G	K	X
M	U	N	E	R	I	C	M	A	R	M	O	R	U	Z	N	Z	B	P	F	P	D
V	Y	Y	T	G	Y	S	P	R	E	A	D	S	H	E	E	T	E	N	N	R	B
L	S	Y	I	K	U	P	T	F	U	N	C	T	I	O	N	A	L	I	T	Y	X
T	L	S	R	J	H	C	C	Z	E	W	V	M	H	A	R	D	W	A	R	E	T
J	U	S	P	T	I	N	K	K	J	S	U	P	P	O	R	T	W	Y	D	R	A

Total marks: 36

# Module 3

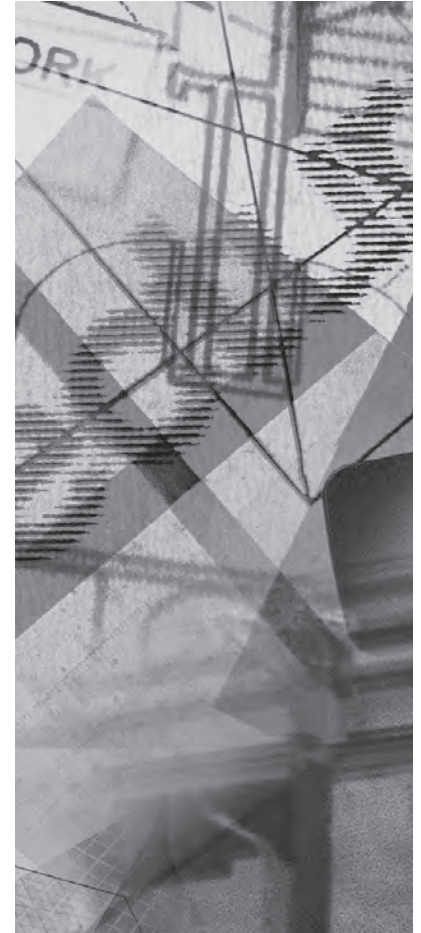


## System software

### Overview

At the end of this module, you should be able to:

- describe the term system software
- define an operating system in terms of the tasks it performs in a computer
- define utility programs in terms of their use
- define language translators in terms of their purpose, with examples.



### Unit 3.1 System software

**System software** refers to programs that manage a computer's resources and scheduling, as well as monitor computer events. System software forms an essential part of a computer, because it links the hardware and the software.

The different types of system software include the following:

- An **operating system** is a type of system software that manages all the hardware and software resources of the computer.
- A **utility program** is a specialised system software program that performs one specific task. For example, when you buy a new hardware component, such as a DVD drive, monitor or digital camera, it is usually accompanied by a CD. This CD contains the installation files of the specific component that must be installed for the hardware to work properly on your operating system. These installation files are system software as they help the operating system to manage the component's resources.
- A **language translator** converts a high-level language into a low-level language.

Here is a simple example of how system software works. You have completed your document in a word processor and you want to print your work. You click on the print function on the toolbar of the word processor interface. The word processor (application software) communicates with the printer driver (system software) which, in turn, tells the printer to print your document.

### Unit 3.2 Operating systems

As mentioned above, an operating system is a type of system software. A computer cannot function without an operating system as it manages the both the hardware and software resources of the computer. It performs basic tasks such as controlling and allocating memory, prioritising the processing of instructions and controlling

#### Words & Terms

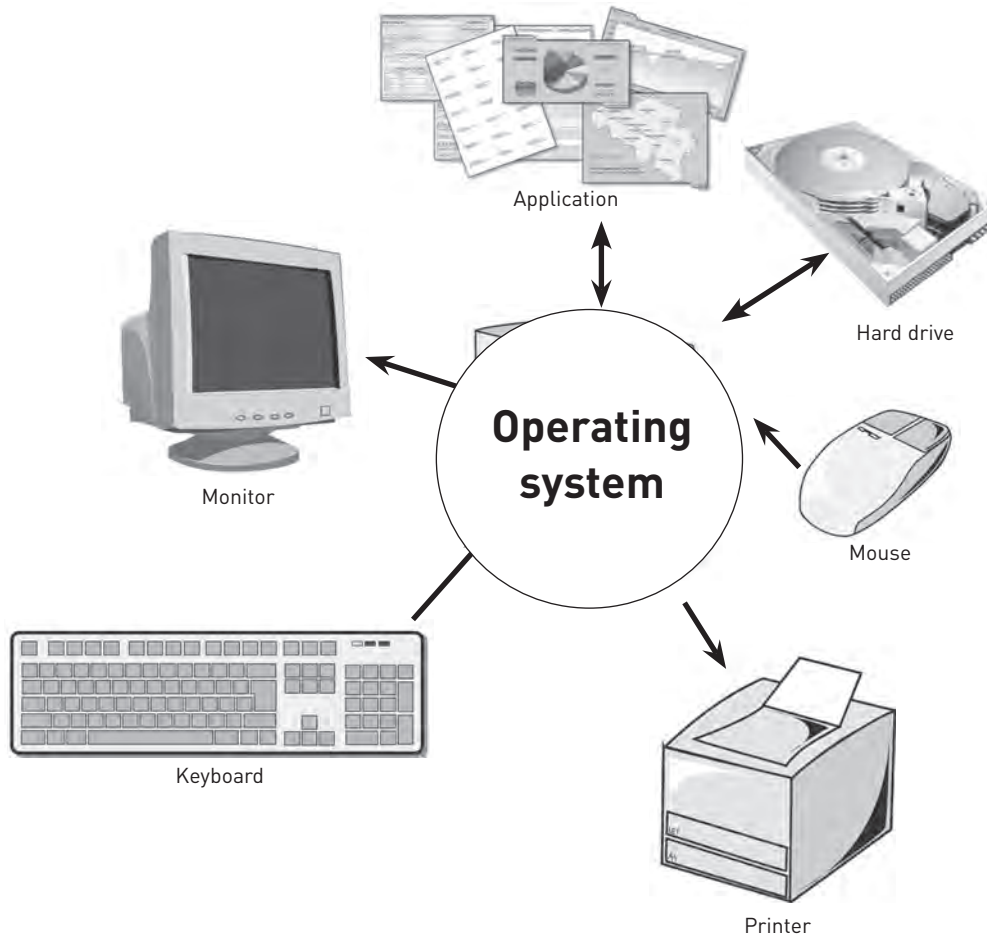
**system software:** programs that manage a computer's resources and scheduling, as well as monitor computer events; forms an essential part of a computer, because it links the hardware and the software

**operating system:** manages the hardware and software resources of the computer

**utility program:** a specialised system software program that performs one specific task

**language translator:** converts a high-level language into a low-level language

input and output devices. The operating system also handles file management, for example when you want to copy, delete or move files.



*Figure 3.1 The operating system plays a central role in any computer*

An operating system handles the following functions:

- **User interface:** The computer needs a method of communicating with the user. The user interface allows the user to talk to the computer via the keyboard or mouse. The user interface displays prompts on the screen to which the user responds by typing in commands or clicking on the relevant icons. Icons are the graphics that the user sees on the screen.
- **Job management:** The user wants to have one or more things to happen on the computer. The operating system has the task of spooling the jobs (putting the jobs in a queue). The user then sees the expected result at the right time. Job management thus controls the order and time in which programs are run.
- **Task management:** Multi-tasking refers to running of multiple, independent computer programs on the same computer, giving the appearance that it is performing the tasks at the same time. Since most computers can only do one or two things at once, this is generally done via time-sharing. This means that each program uses a share of the computer's time to execute. Task management therefore executes multiple programs simultaneously.



- **Data management:** The ability to access data stored on disks is a central feature of all operating systems. Computers store data on disks using files. Files are structured in specific ways to allow faster access, higher reliability and to make better use of the drive's available space. The way in which files are stored on a disk is called a file system and enables files to have names and attributes. It also allows them to be stored in a hierarchy of directories or folders arranged in a directory tree. Data management thus keeps track of the data on storage devices.
- **Device management:** Most of the peripherals that we buy come with a disk. This disk contains the driver for the device. The manufacturer of the device develops the driver separately from the designer of the operating system. The operating system will therefore not know how to work this device unless we provide it with instructions. The instructions are stored on the disk and are called device drivers. The task of the operating system is to manage these devices.
- **Security:** The security of a computer depends on a number of technologies working properly. A modern operating system provides access to a number of resources that are available to software running on the system and to external devices, such as networks. Multi-user systems are protected by passwords to keep out unauthorised users. The firewall is also part of the operating system security.

The operating system provides a software platform on which other programs (application software) can run. Application software is written to perform according to a specific operating system's characteristics.

The operating system may have the following characteristics:

- multi-user (allows two or more users to run programs at the same time)
- multi-processing (supports running a program on more than one central processing unit or CPU)
- multi-tasking (allows more than one program to run concurrently)
- multi-threading (allows different parts of a single program to run concurrently)
- real time (responds to input instantly).

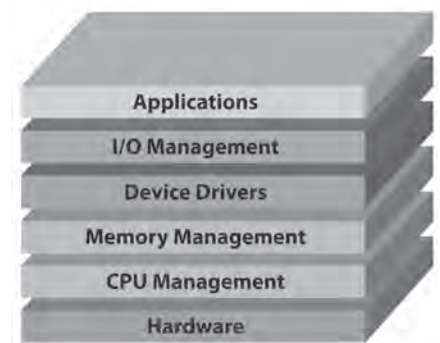


Figure 3.2 The functions of an operating system

## Unit 3.3 Utility programs

Utility software is designed to help manage the operating system. A utility program performs a single task or a range of small tasks to fine-tune the operating system and hardware. Some operating systems have integrated utility software to help the system perform tasks. The program functions like application software, but on a smaller scale.

There are various types of utility software. Examples include disk defragmenters, antivirus programs and the Control Panel of the operating system.

A disk defragmenter detects computer files whose contents have been stored in disjointed fragments and then relocates and stores the fragments together to increase efficiency. This can best be explained by using two tables. Table 3.1 represents the disk in a fragmented



state. There are three files stored on the disk. The data of file 1 is represented by the numbers (1, 2, 3, 4, 5 etc.). The data in the second file is represented by letters (aa, bb, cc, dd, ee, ff, gg, hh). The data of the third file is represented by (++, ++, ++, ++, ++, ++, ++). See Table 3.1 below.

1	2	3	4	5	aa	bb	
11	cc	22	dd				ee
++	++	33		44	55		
	++	++	++	++	++		
66	77		ff	gg	hh		

Table 3.1 Representation of a disk in a fragmented state

As you can see, the data in the files is broken up, or fragmented – the data is scattered over several clusters of the disk. There are also some open spaces on the disk. This makes the reading of the data slow and will influence the speed of the computer. The process of fixing fragmentation is called defragmentation (removing fragmentation). Table 3.2 shows what the disk will look like after disk defragmentation.

1	2	3	4	5	aa	bb	cc
dd	ee	ff	gg	hh	++	++	++
++	++	++	++	11	22	33	44
55	66	77					

Table 3.2 A representation of a disk in a defragmented state

Antivirus programs search the computer for viruses. Figure 3.3 shows the the antivirus program Trend Micro.

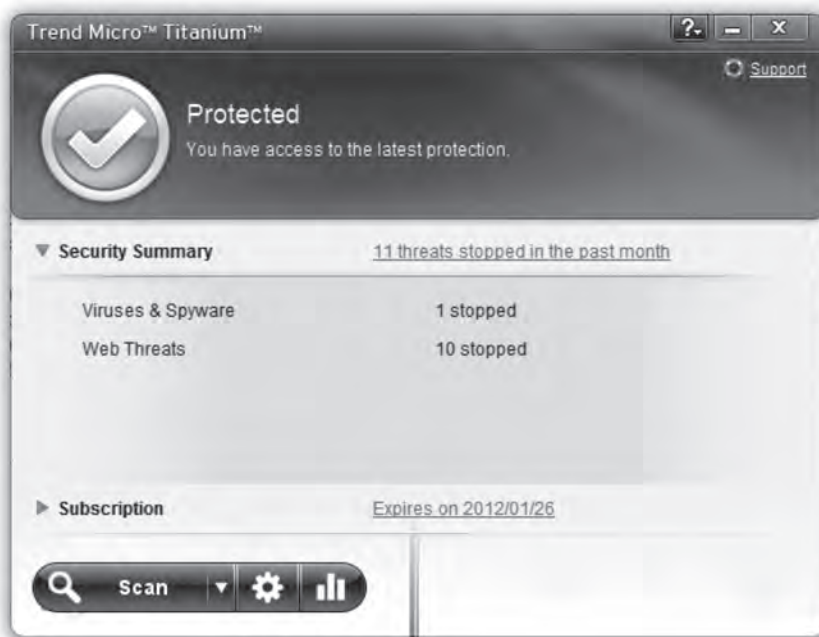


Figure 3.3 A utility program – the antivirus program Trend Micro®

The operating system provides the user with a Control Panel, which includes commands and initialisation files that enable the user to customise the behaviour and the appearance of the user environment. Figures 3.4 and 3.5 show the Control Panel from Windows 7.



Figure 3.4 The Control Panel from Windows 7

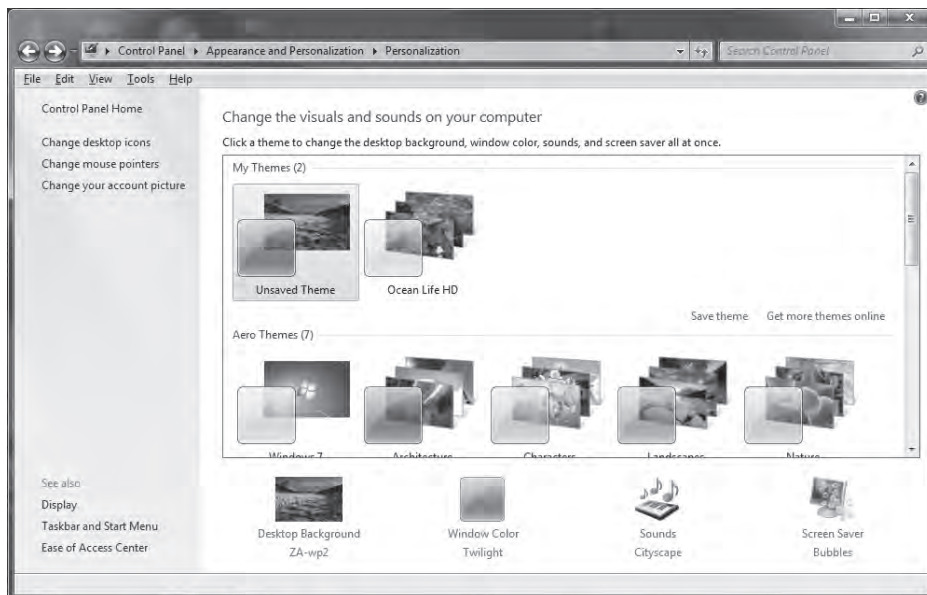


Figure 3.5 The Control Panel for display options from Windows 7

## Unit 3.4 Language translators

Translators convert a high-level language into a lower-level language (assembly language or machine language).

### Words & Terms

**compiler:** a program that translates high-level source code into a low-level object code, normally assembly or machine language

**source code:** text written in a computer programming language

**object or machine code:** the final product of the translated or compiled code

There are two main types of language translators in computers, namely compilers and interpreters:

- A **compiler** is a program that translates high-level **source code** into a low-level object code, normally assembly or machine language. Source code is text written in a computer programming language. Computer programmers tell the computer what actions to perform by means of the programming language. **Object or machine code** is the final product of the translated or compiled code. To **compile** means to translate from human language into machine language. It does not matter in which high-level language a program is written, it will always end up as object or machine code (an executable program).
- An **interpreter** is a program that analyses and executes other programs line by line. This is in contrast to a compiler that does not execute its source code, but translates it into another language, usually executable machine code, which is then executed.

It takes longer to run a program using an interpreter than to run compiled code, as the interpreter must analyse each statement in the program every time it is executed, whereas compiled code simply performs the precompiled actions.

### Words & Terms

**compile:** translate from human language into machine language

**interpreter:** a program that analyses and executes other programs line by line

### Assessment activity 3.1

#### Work in pairs.

1. Define:
  - a) system software
  - b) operating system
  - c) utility program
  - d) language translator.
2. a) What are the functions of an operating system? (8)  
 b) What are the characteristics of an operating system? (6)
3. Give two examples of utility programs. (10)
4. What are the two main types of language translators? (2)  
 Give a brief description of each. (6)
5. Explain the following:
  - a) source code
  - b) object or machine code
  - c) compiled code.
6. Explain the difference between system software and utility software. (6)
7. The \_\_\_\_ refers to the instructions typed by the programmer and that are compiled and then executed as an executable file. (2)
8. The two languages that are the closest to the languages that a computer understands are called \_\_\_\_ and \_\_\_\_ . (1)
9. Name the peripheral devices in Figure 3.6 that are needed to form a computer system. Write the number and the name of the device next to it. (2)  
 (7)

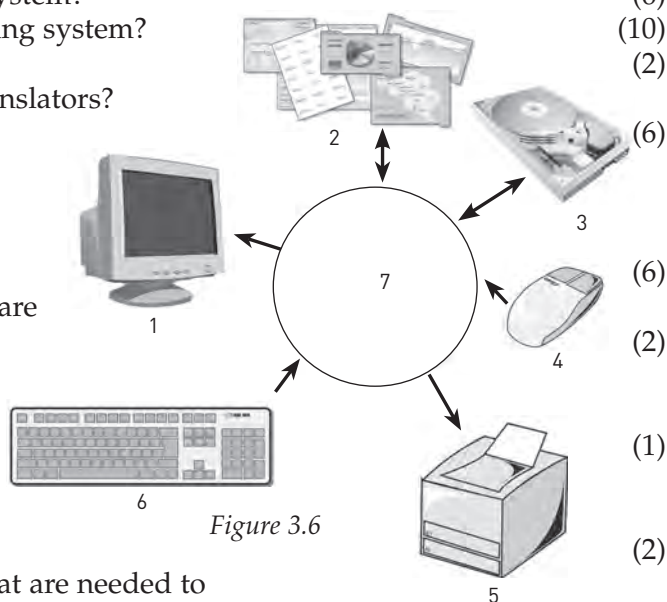


Figure 3.6

Total marks: 50

# Module 4



## Operating systems and environments

### Overview

At the end of this module, you should be able to:

- name and describe different types of operating systems
- describe the environment in which the types of operating systems operate
- outline the history of the different operating systems in terms of proprietary and open-source.



### Unit 4.1 Different types of operating systems

We defined an operating system in Module 3. From this discussion, you saw how important an operating system is to a computer. With this in mind, you can also see that a company that controls the operating system, controls the computer market. Microsoft Windows has almost 90% of the software market. It not only offers a wide range of operating systems for personal computers and mainframes, but also a variety of application software that uses its operating systems. The main competition for Microsoft Windows in the market for operating systems are Unix, Linux and the Apple Macintosh operating system (Mac OS).

#### Unix

The Unix operating system is widely used in both servers and workstations. It was developed in the 1960s and 1970s by three employees of AT&T Bell Laboratories, Ken Thompson, Dennis Ritchie and Douglas McIlroy, using a high-level language (C).

Unix was designed to be a portable, multi-user, multi-tasking and timesharing environment.

Bell Labs distributed Unix as source code so that anyone could modify and customise it for their own purposes. Today, the trademarks 'Unix' and 'Single Unix specification' belong to the Open Group.



Figure 4.1 The UNIX® logo



### Assessment activity 4.1

What is Unix time? Do research on the internet and write a report on what it is and its implications. Has there been any other similar event?

Total marks: 16

## Linux

Linux is a Unix-like operating system and is also open-source. Linux runs on a number of hardware platforms. The Linux kernel was developed mainly by Linus Torvalds.

Linux can be used on both PC and Macintosh platforms. Its security, reliability and the fact that it is free has made it very popular in server environments. It is also gaining popularity in the desktop market.

## Macintosh

Apple Computer developed the Mac OS for use in their computer systems. It is a graphical user interface-based operating system that Apple designed to be user-friendly and to set itself apart from other more technically challenging operating systems. The Mac OS made the graphical user interface popular.

?? Did you know?

Tux the penguin is the logo and mascot of Linux, based on the image created by Larry Ewing in 1996.



Figure 4.2 Tux the penguin



Figure 4.3 Mac OS X® 2006 graphical user interface



## Microsoft Windows

Microsoft Windows is a family of operating systems, varying from servers and embedded devices to personal computers. Microsoft first introduced Windows in November 1985 as an add-on for MS-DOS in response to the growing trend towards graphical user interfaces popularised by the Apple Macintosh. Microsoft Windows eventually came to dominate the computer market.

Microsoft has taken two routes in operating systems. One route is for the home user with more graphics and less functionality (for example, Windows XP Home®). The other route is for the professional IT user with more functionality in networking and security.



Figure 4.4 Microsoft Windows®

## Unit 4.2 The environment in which operating systems operate

Older operating systems, such as MS-Dos and Unix, used a command line interface. This meant that the user had to type in the instructions. Modern operating systems, such as Mac OS and Windows XP, use a graphical user interface (GUI), often pronounced 'Gooney', in which the user simply clicks on an icon or menu to run a program.

Although Apple first introduced the GUI, Microsoft overtook Apple when it also switched to using a GUI interface. Microsoft's operating systems benefited from not being tied to one manufacturer. Microsoft was willing to license its operating systems to different hardware manufacturers, with the result that most computers sold now come with Windows pre-installed.

Microsoft also sells a large range of proprietary software available exclusively for the Windows family of operating systems. Microsoft created a situation called **vendor lock-in**, also known as proprietary lock-in. This situation occurs when a customer is so dependent on a supplier for products and services that it becomes too expensive to change to another supplier, due to the substantial switching costs. Suppliers create this situation by creating different versions of the same architecture that cannot interface with other systems.

The rapid way in which Microsoft Windows dominated the market is closely tied to the spread of personal computers (PCs). It is difficult to believe that PCs are barely 30 years old. More than 1,5 billion PCs have now been sold in the world, 25% of them in the USA. In the past five years, between 2006 and 2010, cumulative PC sales reached 1,3 billion units, or nearly as much as in the first 30 years.

The way Microsoft marketed its product has led to numerous court cases against the company in both the USA and Europe. Many people accuse the company of being anti-competitive and of running a world-wide monopoly. However, it must be said that it does make life much easier if most people use the same operating system. It means that you do not have to relearn everything when you join a new company, as was often the case in the years before Microsoft became dominant.

### Words & Terms

**vendor lock-in:** when a customer is so dependent on a supplier for products and services that it becomes too expensive to change to another supplier due to the substantial switching costs; also called proprietary lock-in

#### Assessment activity 4.2

Add drawings and pictures where you can when answering the following questions.

1. In an introductory paragraph, describe the term system software using at least two examples. (4)
2. Describe three common characteristics of an operating system. (3)
3. In one paragraph, define utility programs in terms of their use. (4)
4. Define language translators in terms of their purpose, with two examples. (4)

**Tota marks: 15**

#### ?? Did you know?

William Henry 'Bill' Gates, born 28 October 1955, is the cofounder, former chairman, former chief software architect and former CEO of the Microsoft Corporation. Bill Gates has topped the Forbes Magazine list as the richest man in the world for the last 12 years in a row, with a net worth of \$50 billion. His wealth briefly surpassed \$100 billion in 1999, making him the first \$ billionaire (someone with a net worth of \$100 000 000 000).



*Figure 4.5 Bill Gates*

## Unit 4.3 The history of operating systems in terms of proprietary and open-source

Because operating systems are the software that make the hardware usable, their development is closely linked to the development of hardware. Since the 1940s, operating systems have evolved through a number of distinct generations that correspond approximately to the decades. The following is a brief overview of these generations.

### Early history – the 1940s and 1950s

In the 1940s, the earliest electronic digital systems had no operating systems. Computers of this time were so primitive that users entered programs into the computer one bit at a time on rows of mechanical switches.

Eventually, machine languages consisting of strings of the binary digits 0 and 1 were introduced that sped up the programming process. The systems of the 1950s generally ran only one job at a time and allowed only one user at a time to use the machine. Each user wrote all the code necessary to implement a particular application, including the highly detailed machine-level input/output instructions.

Soon, the input/output coding needed to implement basic functions was consolidated into an input/output control system (IOCS). Users wanting to perform input/output operations no longer had to code the instructions directly. Instead, they used IOCS routines to do the real work. This greatly simplified and sped up the coding process. The implementation of the input/output control system was the beginning of today's concept of an operating system.



General Motors Research Laboratories began designing what many consider to be the first operating system for their IBM 701 mainframe in early 1956. Its success helped establish batch computing – the grouping of the jobs into a single deck of cards, separated by control cards that instructed the computer about the various specifications of the job. The programming language that the control cards used was called job control language (JCL).

The batch processing system greatly improved the use of computer systems and helped demonstrate the real value of operating systems by managing resources intensely. This type of processing, called single-stream batch processing, became the state-of-the-art in the early 1960s.

## **The 1960s – timesharing and multi-programming**

The operating systems of the 1960s were also batch-processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. It was rare that a single job used all of a computer's resources. The operating system designers realised that running a mixture of various jobs appeared to be the best way to optimise computer use. This process is called multi-programming. Here, several programs simultaneously compete for system resources.

The operating systems of the 1960s, while capable of multi-programming, were limited by the memory capacity. This led MIT to implement the first timesharing system in 1961, called the Compatible Time-Sharing System (CTSS). The demonstration version allowed just three users to share the computer at a time and reduced the turnaround time to minutes, and later to seconds. It also demonstrated the value of interactive computing. MIT then developed the Multics operating system as the successor to CTSS. Although Multics was not successful, it gave rise to perhaps the most versatile operating system existing even today, namely the UNIX system.

## **The 1970s – Development of UNIX**

The UNIX operating system is particularly noteworthy because this is the only system that has been successfully implemented in every kind of computer, from microcomputers to supercomputers.

From 1965–1969, Bell Labs participated with General Electric and Project MAC at MIT in the development of Multics system. Originally designed for the mainframe, Multics was a large and complex system. As the project progressed, it became clear that although Multics was likely to deliver the variety of services required, it would be a huge, expensive system and very difficult to develop. For these reasons, Bell Laboratories withdrew from the project in 1969.



This, however, did not dissuade some members of the Bell Labs' research staff to work on a far less ambitious system. The group, led by Ken Thompson, wanted to create a simple computing environment for programming research and development. They later named this UNIX, a pun on Multics, according to one of the co-developers, Dennis Ritchie.

The design of UNIX evolved over a period of a few months in 1969 based on a small set of primitive concepts. By the early 1970s, UNIX was working well to the satisfaction of the designers, providing remarkably powerful facilities for a single user on a minicomputer, called the PDP-7. However, the designers still had difficulty convincing the computer community of its merits.

In 1973, Dennis Ritchie, a former Multics teammate, joined the UNIX team. Like any other operating system before it, the first version of UNIX was written in Assembly language, which made it machine-dependent. Ritchie designed a new language, called C, especially for the UNIX. The use of C made UNIX portable, or machine-independent so that it could be implemented on any computer system. In fact, this was the first time an operating system was written using a higher-level language than Assembly language.

AT&T, the parent company of Bell Labs before telephone deregulation of 1983, was not allowed to compete in the computer industry, so it made the UNIX systems available to universities at a nominal fee. More importantly, AT&T also distributed its source code. The minimal design of UNIX and its simplicity compared to the complex operating systems of the mainframe made it hugely popular with universities and research laboratories. By the 1980s, UNIX had become the standard operating system among computer professionals. Today, UNIX, in its open-source projects, can be found in NetBSD, OpenBSD and FreeBSD. Linux is also a derivative of UNIX.

## The 1970s – The development of the microchip and the personal computer (PC)

In the 1970s, the development of the microprocessor, popularly known as the microchip, changed the nature of the computer by being the enabling technology for personal computers. In this personal computing environment, operating systems became elevated to a higher level of importance, and by the 1990s had become the dominating factor in the software industry. A microchip is an integrated circuit consisting of thousands of transistors on a small silicon chip.

In 1975, the first microprocessor-based computer, the Altair 8800, was introduced by MITS. It had no display, no keyboard and not enough memory to do anything useful. The only way the Altair could be programmed was to enter programs in pure binary code by flicking the small hand switches on the front, a situation reminiscent of the early computers of the 1940s. It had very little that could be considered truly useful for a user.

The news of the introduction of Altair in the market made computer experts and entrepreneurs immediately jump in to take advantage



Did you know?

The first integrated circuits were produced in 1962 for the military. They cost about \$50 and contained an average of half a dozen active components per chip. After that, the number of components on a chip doubled every year. By 1970, it was possible to make LSI (Large Scale Integration) chips that contained thousands of active components.



of the opportunities of adding hardware and writing software. In 1975, two of these computer experts, Bill Gates and his childhood friend, Paul Allen, decided to develop the BASIC language for Altair. After obtaining permission from Ed Roberts, the owner of MITS, Bill Gates and Paul Allen formed a partnership, which they named Micro-Soft (the hyphen was dropped later). After six weeks of intense programming, they delivered a BASIC programming system to MITS. They refused to sell it to MITS though, instead licensing it in return for a royalty.

The Altair 8800 with its add-ons and BASIC software transformed electronics, as computer hobbyists showed strong enthusiasm. During the first quarter of 1975, MITS received \$1 million in orders for Altair. The royalty from this provided a substantial cash flow for Microsoft, then only a tiny company.

From 1975, the personal computer industry saw rapid expansion. The peripherals, such as keyboards, disk drives and monitors were added to the bare-bone Altair models.

In 1976, the first operating system for these Intel-based personal computers was written by a system programmer named Gary Kildall. As a consultant for Intel, he developed an operating system called CP/M (Control Program for Micros) for Intel 8080. While doing this, he realised that the floppy disk would make a good mass storage device. He realised that a disk had several advantages over magnetic or paper tape. First, it was faster. Second, the user could both read and write data to it. Its primary advantage was that a disk had random access. Users did not have to run through the entire spool of tape to get to a specific piece of data.

To accomplish this, however, required some special programming, something which IBM did in the 1960s for its mainframe computers, called Disk Operating System (DOS). However, a personal computer disk operating system had little to do with mainframe operating systems. What was needed was rapid and accurate storage and retrieval of files from a floppy disk. A typical file would, in fact, be stored in a set of fragments, inserted at whatever free space was available on the disk. The operating system for a personal computer needed to be designed in such a way that it could find these free spaces, put data there, retrieve it later on, and then reassemble the fragments. With this concept, he extended the CP/M operating system for Intel 8080 that it could also manage disk drives. He called this specialised code the BIOS – Basic Input/Output System.

## The 1970s – the rise of Apple Computer

Despite the increasing popularity of microcomputers among computer hobbyists and professionals, they did not appeal to non-experts and households. In 1975, one company, Apple Computer, was established that changed the microcomputer from being a challenging tool for the computer professionals and hobbyists to a useful personal computer for households and non-expert individuals.

In 1975, two computer enthusiasts, Steve Jobs and Steve Wozniak, founded a company called Apple Computer. What distinguished Apple from others was its vision and determination to make the microcomputer a consumer product for a much greater market of consumers. With this vision, Apple I came out in 1975 and in 1977, a much improved version of Apple, called Apple II was released. Although Apple I's BASIC was written by Steve Wozniak, the company decided, to contract the writing of the operating system out to Microsoft for a better version.

In 1979, Apple Computer added the first spreadsheet software for microcomputers, called VisiCalc. It also added a word processing program. Apple's success convinced many others of the feasibility of such a computer. One of the firms that decided to enter the personal computer industry was IBM, the most dominant firm in the computer industry at that time, and this changed the computer industry dramatically.

## **The 1980s – IBM's entry into the personal computer industry and its effect on operating systems**

While the 1980s saw the development of features such as distributed processing and client-server processing, it is the personal computer segment that had the major impact on the computer industry. In this decade, the personal computer and its operating system played a significant role, and became the dominating segment in the computer industry.

The critical event in this decade started with IBM's entry into the personal computer market in 1980. When the IBM negotiating team visited Microsoft to close the deal on BASIC, it sought Bill Gates' help in recommending what to do about the operating system. Bill Gates was eager to accommodate IBM's needs, and offered to provide one to IBM who, without seeing the actual product, entered into an agreement. Bill Gates, with his experience of the advantages of royalty rather than outright selling of BASIC for Altair, insisted on a royalty for each copy sold.

Microsoft, however, did not have an actual operating system ready, nor did it have the resources to develop one to beat IBM's deadline. However, Gates knew that Tim Paterson had developed an operating system for the Intel 8086 chip, known internally as QDOS, which stood for 'Quick and Dirty Operating System.' Microsoft initially paid \$15 000 for the rights to use the product and later paid a larger sum of money for the complete rights. Microsoft, after slight modifications, named it MS-DOS.

During 1991, the first personal computers by IBM began to come off the IBM assembly plant. Within the next few weeks, the IBM personal computer became a runaway success, exceeding almost everybody's expectations.



During 1982–1983, the IBM personal computer became an industry standard. IBM's decision to allow it to have an open architecture meant the other firms could copy its design. This encouraged other manufacturers to produce computers with the same architecture, which came to be known as clones, in order to take advantage of the huge demand that market was experiencing.

One company that benefited the most out of this was Microsoft. Almost every model of IBM PC and its clones were supplied with its MS-DOS operating system. As hundreds of thousands and eventually millions of machines were sold, money poured into Microsoft. This revenue stream allowed Microsoft to diversify into computer application software without having to rely on external venture capital. It also allowed Microsoft to cross-subsidise some of the software that initially did not succeed. For example, in mid-1983, Microsoft began to develop a word processing software package called Word. Word was initially not a successful product and had a negligible impact on the market leader at that time, WordStar. However, the cash flow from the MS-DOS allowed Microsoft to continue to market the product at a loss until the opportunity came later to bundle it properly with its new generation of operating systems, Windows.

One major deficiency of MS-DOS was that it was not very easy to use. The user interacted with the operating system through a command line interface in which instructions to the operating system had to be typed by the user in an exact way. If there was even a single letter out of place or a character was missing or mistyped, the user had to type the line again. This problem, called a lack of user-friendliness, prevented the PCs from being truly acceptable as a consumer product. In 1984, Apple solved this problem when it introduced its Macintosh model.

## The 1980s – Introduction of Macintosh

Apple was the only major microcomputer manufacturer that did not switch to producing IBM-compatibles, instead choosing its own path. In 1984, it unveiled its Macintosh model, which was far superior to any of the IBM PCs or its clones in terms of user-friendliness. It used a technology called a graphical user interface (GUI) and a pointing device called a mouse.

But, after the initial enthusiasm, the sales were disappointing. The problem was the lack of sufficient software packages and other add-ons. One reason for this was Apple's policy to keep Macintosh's architecture closed. This closed architecture meant that hardware and software developers found it difficult to create their own Macintosh add-ons and software without the close cooperation of Apple. The Apple Macintosh operating system, or Mac OS, developed as follows:

- 1984 – System 1-4
- 1987 – System 5
- 1988 – System 6
- 1991 – System 7
- 1997 – Mac OS 8
- 1999 – Mac OS 9



- 2001 – Mac OS X.
- The latest of the OS X versions is 10.7 Lion.

In order to reposition itself, Apple invited several leading software firms to develop software. But, a lack of sufficient level of demand for Mac software caused software developers to be discouraged. The only major firm that did accept the offer to write software for Mac was Microsoft.

By taking the offer from Apple to write programs for them, Microsoft found an environment insulated from the highly competitive IBM-compatible market, where it was facing intense competition for its application software. By 1987, Microsoft, in fact, was deriving half of its revenue from its Macintosh software. More importantly, working on the Macintosh gave Microsoft firsthand knowledge of the GUI technology, on which it based its new Windows operating system for the IBM PC.

## **The 1980s – Launching Microsoft Windows**

Microsoft started its own graphical user interface (GUI) project in 1981, shortly after Bill Gates had visited Steve Jobs at Apple and seen the prototype of the Macintosh computer under development.

Microsoft Windows was heavily based on the Macintosh user interface. In 1985, shortly after Windows was launched, Microsoft signed a licensing agreement to copy the visual characteristics of the Macintosh, thereby avoiding legal trouble for version 1.

Although a million copies were sold, most users found the system to be little more than a gimmick and the vast majority of users stayed with MS-DOS. Part of the reason was that the microprocessor at that time – Intel 80286 – was not fast enough to support GUI technology. Only in the late 1980s when the next generation of microprocessors – the Intel 386 and 486 – became available, did the GUI become much more supportable. At that time, Microsoft introduced its Windows 2.0. Windows 2.0 popularity also provided Microsoft with the opportunity to bundle its Excel spreadsheet software and its word processing software, Word. This allowed it to increase their market share considerably, and it eventually became the market leader in these applications.

## **The 1990s and beyond – the dominance of Microsoft in the operating system market**

In 1990, Microsoft introduced Windows 3.0 all around the world with extravagant publicity. Windows 3.0 was well received. Microsoft continued to take advantage of its dominance in operating systems by bundling the application software with operating systems and by taking advantage of its intimate knowledge of the source code of the operating system.

In April 1991, IBM announced a new version of OS/2 – release 2.0. However, despite the sound technical merits of OS/2, IBM continued



to lose ground against Microsoft – partly because of a lack of appealing software and partly because of failure to market it effectively.

Failure of OS/2 resulted in furthering the dominant position of Microsoft Windows. This position was further reinforced in 1995 with its release of Windows 95 which was an immediate success. Since then, it has introduced several other versions of Windows, including Windows 2000 and Windows XP.

**Source:** Moumina, A. & Bergin, T. 2001. *History of operating systems*.

The following is a history of the various versions of Windows:

- **Windows 3.0:** In 1990, Microsoft released Windows 3.0 with a streamlined GUI and improved security. The company also launched Microsoft Office in the same year.
- **Windows 95:** In the mid-1990s, at the start of the internet boom, Microsoft released Windows 95. This was a 32-bit operating system with a completely new interface.
- **Windows 98:** Microsoft released Windows 98 in June 1998. It included new hardware drivers and better support for the FAT32 file systems. Internet Explorer was also integrated into Windows 98.
- **Windows XP:** In 2001, Microsoft released Windows XP. This version had a 64-bit system that could pass the 4GB RAM barrier that 32-bit systems have. A third service pack was released in 2008. XP broke the record for Microsoft's longest lasting flagship OS as it ran from 2001 until 2007.
- **Windows Vista:** Vista had a new restricted user mode called User Account Control. It included new applications such as Windows DVD Maker, Internet Explorer 7 and Windows Media Player 11.
- **Windows 7:** This version was released in 2009 to replace Windows Vista. It introduced a large number of new features and is available in six different editions to cater for different users:
  - Windows 7 Starter
  - Windows 7 Home Basic
  - Windows 7 Home Premium
  - Windows 7 Professional
  - Windows 7 Ultimate
  - Windows 7 Enterprise.

**Note:** This history of the Microsoft operating systems does not include all their operating systems and server operating systems, but just some of the major Windows OS releases.

## Other open-source operating systems

### a) Berkeley Software Distribution (BSD)

Berkeley Software Distribution (BSD) is a Unix-based operating system developed between 1977 and 1995 at the University of California in Berkeley. Today, the term is used to refer to the following BSD descendents:

- **NetBSD:** a free Unix-like operating system via BSD Unix
- **OpenBSD:** created by one of the original founders of FreeBSD
- **FreeBSD:** a free Unix-like operating system via BSD Unix.

**?? Did you know?**

By January 2011, Microsoft had sold more than 300 million copies of Windows 7.



Figure 4.6 Windows 95<sup>®</sup>



Figure 4.7 Windows 98<sup>®</sup>



Figure 4.8 Windows XP<sup>®</sup>



Figure 4.9 Windows Vista<sup>®</sup>



Figure 4.10 Microsoft Windows Logo<sup>®</sup>

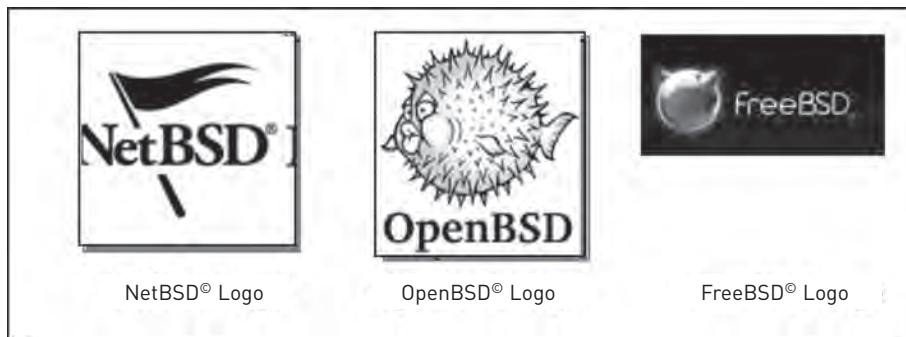


Figure 4.11 Berkeley Software Distribution (BSD) open-source operating systems

## b) Linux

Linux was a personal project of Linus Torvalds who wanted to create a new operating system kernel. It was initially released in 1991. Torvalds created the Linux kernel based on Unix, and at the time did not know about the BSD projects. From its basic form and under a prohibitive commercial distribution licence, Torvalds' small idea grew and expanded. At first, he wanted to name it Freax, but one of his co-workers at the University of Helsinki in Finland, who volunteered to help with the **FTP** download server at the time, disliked the name and without Torvalds' consent renamed it Linux. Linus had considered the name Linux but found it too egotistical, but consented to the name change.

The most popular name in Linux distributions, also known as 'distros', has a particularly South African flavour. Mark Shuttleworth founded the **Ubuntu** Foundation in 2005. Ubuntu is easy to install, performs common tasks just as easily and has regular updates. This has propelled Ubuntu to the top of Linux distributors. The main reason for its popularity is that it offers the same high quality as other commercial distributors, but Ubuntu and its support are free to anyone from businesses to personal users, on the same terms. There is also an educational version of Ubuntu that targets younger users. This version is called Edubuntu.

The basic difference between the BSD (Berkeley Software Distribution) projects and Linux is that each BSD project has a tightly controlled design, while Linux is free-form. Linux is the operating system that is often used in businesses on servers, especially as **proxy servers**. Linux is also the popular choice of computer users who want to experiment with and tweak their personal copy.



Figure 4.12 Mark Shuttleworth

## Words & Terms

**FTP (file transfer protocol server):** a server containing files that can be exchanged and manipulated over the internet

**ubuntu:** an African word meaning 'humanity to others'

**proxy server:** a server that businesses use as an intermediary between users in the company and the internet. From the proxy server, IT administrators can implement security such as firewalls, as well as blocking users from viewing certain websites as per company policy, for example Facebook, Youtube and pornographic web pages

## Did you know?

Tim Berners-Lee created the first web browser on a NeXT Computer.



## ?? Did you know?

Virtual memory is a memory management technique that tricks a computer into thinking it has more memory (RAM) than it actually has. Virtual memory uses space on a computer's hard disk to accommodate the additional memory it needs.

## ?? Did you know?

Software pirates are users who download or copy software with the intention of using the software product without paying for the software licence. Piracy is illegal and a user can be fined or even jailed for pirating software.

## ?? Did you know?

In a 2001 study done on Red Hat Linux version 7.1, it was found that the operating system contained roughly 30 million lines of source code. Using a constructive cost model, the study estimated that if the development had been done by a software vendor in the USA, it would have cost approximately \$1 billion.

In association with the non-profit organisation One Laptop per Child, Red Hat also worked on a design and produced an inexpensive laptop running a slimmed down version of Red Hat's Fedora Linux operating system with the aim of providing every child in the world with a laptop so that they can have access to open knowledge and learning. Visit [www.laptop.org](http://www.laptop.org) for more information.

### Assessment activity 4.3

1. Name two different operating systems. (2)
2. Describe the environment in which the operating system operates. (3)
3. Briefly outline the history of three different operating systems. (9)

**Total marks: 14**

## Summative Assessment

1. Define the following terms:
  - a) Software (2)
  - b) Productivity (2)
  - c) Version (2)
  - d) Graphical user interface (2)
  - e) Operating system (2)
  - f) Background compatibility (2)
  - g) Forward compatibility (2)
2. What are the main functions of an operating system? (12)
3. Name and describe three different types of operating systems. (9)
4. Name two ways in which to improve productivity in application software. (4)
5. What are the four fundamental features of application software? (8)
6. What is application software also known as? (1)
7. Name the different software categories. (5)
8. Give an explanation of system software. (3)
9. Name and explain the different operating systems characteristics. (10)

**Total marks: 66**

## Think about it

1. What operating system would you choose and why?
2. On what would you base your decision?