Janet Allison
Martin McKinney
Adrian Moore

CCEA AS

DIGITAL TECHNOLOGY

COLOURPOINT EDUCATIONAL

© 2019 Janet Allison, Martin McKinney, Adrian Moore and Colourpoint Creative Ltd

ISBN: 978 1 78073 116 2
eBook ISBN: 978 1 78073 281 7

First Edition
First Impression 2019

Layout and design: April Sky Design

Colourpoint Educational
An imprint of Colourpoint Creative Ltd
Colourpoint House
Jubilee Business Park
Jubilee Road
Newtownards
County Down
Northern Ireland
BT23 4YH

Tel: 028 9182 6339
E-mail: sales@colourpoint.co.uk
Website: www.colourpoint.co.uk

The Authors

Janet Allison has a BSc in Computing & Mathematics (Queen's University Belfast) and an MSc in Information Technology (Ulster University). She currently teaches computing to Level 3 and Foundation Degree students in Northern Regional College. She works as an AS-level examiner for an awarding body.

Professor Martin McKinney has an MSc in Computer Science and Applications and is Professor Emeritus of Computing (Ulster University). He is a former Head of School of Computing & Information Engineering (Ulster University) and a former Vice-Principal of Teaching & Learning (Northern Regional College).

Dr Adrian Moore is a Senior Lecturer and Course Director in the School of Computing at Ulster University. He holds a PhD and BSc in Computing and has published widely in areas including multimedia web technologies, computer networks and computer graphics.

The answers to the questions in this book are available online in PDF format. Visit www.colourpointeducational.com, and search for this book. On the page for the book you will find details of how to download the answers. If you have any problems, please contact Colourpoint.

CONTENTS

UNIT AS 1
Approaches to Systems Development

CHAPTER 1
Reasons for Systems Development

By the end of this chapter students should be able to:

- explain the impact of the 'software crisis';
- explain the need for software systems that meet the needs of organisations and/or individuals;
- explain the main factors affecting systems development: the user needs, time and cost;
- understand that a computer system consists of a user interface, processes and data;
- describe the roles of the following during systems development: the systems analyst; the project manager; and the programmer.

1.1 Introduction

To understand **systems development,** it is first useful to consider what we mean when we talk about a **system**. All of us will have come across things in the real world that we call systems. Some of these systems are extremely large and complex. Others are small – but often just as significant.

Examples of large, real-world systems are as follows.

- The **solar system** includes the Sun and all the planets and other objects that orbit it. It is an example of a naturally-occurring system.
- **The digestive system** in the human body consists of a group of organs that work together to convert food into energy and other nutrients that serve to feed the body.
- A **city transport system** includes the vehicles (such as buses, trains, trams, cars and bicycles), the associated pathways (such as roads, railway lines, tram lines and cycle lanes) and routes through the city that they serve.

- An **education system** facilitates the training and instruction of people from their infancy into adulthood. An education system itself consists of a number of smaller systems (known as **sub-systems**) such as the pre-school system, primary school system and the secondary school system.

Other systems are significantly smaller and simpler than those described above, such as:

- A **queueing system** in a café which might include an area for people to stand, different serving counters and a way for staff to bring cooked food to the correct customer.
- A **job applications** system which, at the most basic level, might simply allow applicants to submit their application online. A highly sophisticated job applications system might provide a much broader level of functionality. For example, it might also facilitate the shortlisting of applicants, the scheduling of interviews and any post-interview monitoring and reporting.

1.2 Characteristics of a System

The above examples help us to see what it is that makes something a 'system'. The three main characteristics of any system are:

- the system should have a defined **purpose**;
- the system should meet a defined **need**;
- the system should have a **number of components** which are likely to be interdependent.

Case Study

City Transport System

Consider the city transport system discussed above. We are able to state its purpose as "to enable commuters and other users to navigate around the city in a flexible, efficient and economical manner".

In doing so, this transport system meets the defined transport needs of the city and its inhabitants, namely that people and goods need to be able to move around the city in order for it to function.

The transport system has a number of components such as the buses and trains (public transport), cars, bicycles and taxis (private transport) and the roads, railways and footpaths themselves. The system can contain a number of sub-systems such as the bus system and the train system. These systems are integrated. For example, cars are dependent on the roads and trains on the tracks. There may be further interdependencies, so for example the train and bus systems may have integrated timetables to provide connecting services for train arrivals.

In summary, the city transport system meets the three characteristics of a system:

- **Purpose:** To enable commuters and other users to navigate around the city in a flexible, efficient manner.
- **Need:** So that people and goods can move around the city in order for it to function.
- **Components:** Public and private transport vehicles, roads, paths, tracks etc.

As we have discussed, large systems can generally be broken down into a number of smaller sub-systems. In some cases, a system can be so complex that it is essential to break it down in this way. An example is the system for running an airline.

Case Study

Airline System

Consider the computer system for Dalriada Airlines, which must facilitate all aspects of running the airline's operations. Rather than consider the entire system, we instead think of the 'system' as consisting of the following sub-systems:

- **Bookings system**, to facilitate booking of flights by passengers.
- **Payments system**, to facilitate the payment of flights booked by passengers.
- **Lost luggage system**, to help trace luggage that has not arrived at the correct destination.
- **Human Resources (HR)** system to deal with essential HR functions such as the payroll, hiring of staff, handling appraisal and maintaining staff records.
- **Personnel scheduler** system, to support the management of the flight rotas for staff such as the pilots and flight attendants.
- **Maintenance scheduler** system, to ensure that the planes are serviced in line with any legal/insurance requirements and to ensure that the planes are maintained in an airworthy condition.

This list in not exhaustive but serves to illustrate how complex and inter-connected systems can be.

1.3 Developing Systems

Systems development is concerned with the creation, or **development**, of systems. The term is normally used when referring specifically to **computer (software) systems**, i.e. systems where there is an element of computer processing. Hence the solar system would not be classified as a computer (software) system whereas an online banking system would clearly be classified as such.

In this context, the word 'development' is used to refer to the full range of activities from the initial identification of the need for a system, through a series of tasks that culminate in the full and final implementation of the system. This series of activities is what constitutes the systems development process and includes the following steps:

- Identify the **need** for a system by **analysing** the current system.
- **Define** the system. At this stage, the precise user requirements of the system have to be identified and agreed. Any mistakes made in the definition are likely to require a re-think at a later stage and this can be very expensive.
- **Design** the system. All systems should be carefully designed (and a systems specification produced) as any flaws in the design are difficult to alter later and can be very expensive.
- **Implement** the new software application or program based on the systems specification.
- **Test** the new system.
- **Document** the system.
- **Hand over** the system.
- **Maintain** the system.

Each of these steps will be discussed in more detail in subsequent chapters.

1.4 The Software Crisis

In the embryonic days of the software industry the process of developing software was more informal and certainly less structured than it is today. Many of the procedures that are now taken for granted had not yet been established in what was still a very young industry. For example, approaches to establishing precise requirements from customers were not as sophisticated or as precise as we would now expect. In the absence of clear and precise requirements, software often did not fully meet the user requirements.

A number of factors led to the software crisis:

- People had little historical experience of similar work on which to base future work estimates. Hence the time and cost estimates for work were often crude and inaccurate.
- The tools used to develop software were not very sophisticated and as a consequence software took much longer to produce.

- Unlike today, there were no tools to support the early identification of errors and this slowed down the process of locating and correcting faults. This also meant that systems were difficult and expensive to maintain.
- Systems were not meeting user requirements.
- Specialist testing staff were not part of the software development process, so there was no formal process for testing software for correctness.

As a consequence of these factors, the software being developed was typically of lower quality and was less robust than the systems we are familiar with today. Teams dedicated to developing the software often struggled to meet both deadlines and cost estimates. This meant that the industry was ill-prepared to meet the demand for ever larger and more complex systems. The problem became more and more acute and eventually culminated in the **software crisis**. The term was first coined at a NATO Software Engineering conference in Germany in 1968 to refer to the difficulties the software development industry was facing.

Software developers realised that in order to develop quality software on time, within budget and to more accurately meet the needs of the user, a more systematic approach to the software development process was necessary. This led to the development of **software engineering**. Software engineering is the application of a more structured approach to the design, development and maintenance of software systems, similar to the systematic approach taken in other areas of engineering such as civil engineering.

1.5 Factors Affecting Systems Development

Three main factors affect the systems development process:

- The necessity of meeting the **needs** of organisations and individuals.
- The **time available**.
- The **costs** of the project.

The needs of organisations and individuals

As previously discussed, all computer systems have both a defined **purpose** and meet a defined **need**. If we consider the example of the airline system, and focus on the booking sub-system, we can easily see that it satisfies these two criteria:

- An airline booking system has a defined **purpose** which is to allow travellers to efficiently and effectively book and pay for flights over the Internet.
- There is a defined **need** for such a system because the sheer volume of flights could not be facilitated by booking on a personal, one-to-one basis.

While it is easy to see the benefit of using software systems for organisations, organisations are also aware of the need to provide services in a manner that is easy for individual customers to use. In the case of an airline, the customers are travellers wishing to book flights. These customers are called the **users** of the system and it is important that they find the system **accessible** and **easy to use**. If they do not, they may take their business elsewhere and the system will therefore have failed to achieve its purpose. The **user interface** is how a user interacts with the system. It is considered to be of vital importance to a system's usability. Figure 1.1 illustrates how much interaction there is across the interface between the user and a simple online airline booking system.

For a system to be a success, therefore, it must meet the needs of both the organisation that operates it and the individuals who use it.

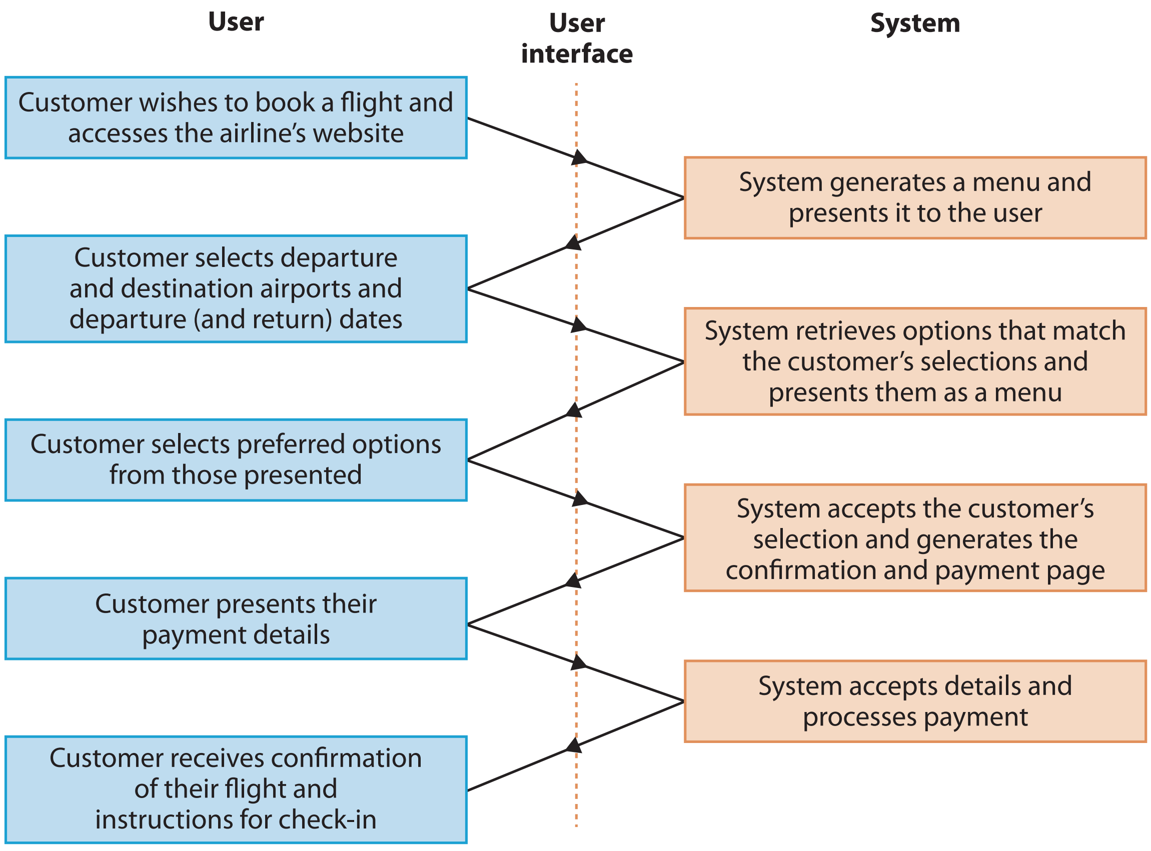
User
User interface
System
Customer wishes to book a flight and accesses the airline's website
System generates a menu and presents it to the user
Customer selects departure and destination airports and departure (and return) dates
System retrieves options that match the customer's selections and presents them as a menu
Customer selects preferred options from those presented
System accepts the customer's selection and generates the confirmation and payment page
Customer presents their payment details
System accepts details and processes payment
Customer receives confirmation of their flight and instructions for check-in

Figure 1.1: Typical online airline booking system.

The time available

The length of time available to develop a new system constrains what can be done. For example, there is little point in specifying a component that will take a year to develop if the system has to be operational within six months.

The costs of the project

Similarly, there is a limit on how much money that can be spent on a project. This will place limits on the number of people who can be employed and the type of hardware that can be purchased. Systems development must take place within the constraints of cost.

1.6 Elements of a Computer System

A computer system can be thought of in simple terms as a system that uses computing power to support a function. Typically, such systems will be characterised by the ability to:

- receive data (raw facts);
- process the data received;
- store the data received and any data generated as a result of processing;
- present the processed data as information; and
- communicate data between related sub-systems.

From this description it is clear that a computer system receives data as input, performs tasks on the data and presents processed data (information) to the user as output. Therefore, we can describe a computer system as consisting of the following distinct elements (as shown in figure 1.2):

- **Data**, provided by the user or generated by the processes.
- **User interface (UI)** which allows users to send data into the system and to view the data that has been processed.
- **Processes** which manipulate the data in the desired manner.

Each of these elements is important in its own right. For example, poor data will produce meaningless or unreliable results. A poor user interface will result in users avoiding the system, making errors and spending more time using the system. Poor processing could produce inaccurate data or take more time than necessary.

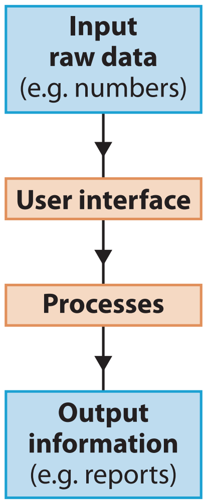

Figure 1.2: A computer system consists of data, a user interface and processes.

We can see this structure in the numerous computer systems we encounter in our everyday lives. The following case study shows how all three elements are present in a typical online banking system.

Case Study

Online Banking System

An online banking system consists of the following elements:

Data

- Account holder details.
- Transaction data.

User Interface

- Accepts user verification details (name, password and other security checks).
- Presents details of the user's account and the results of any requests that they make for information in a suitable format.
- Permits the user to enter details of transactions such as payments or transfers between accounts or transfers to other accounts.

Processes

- To retrieve our account information based on the login information.
- To analyse our data and our recent transactions.
- To allow us to make new transactions.

Task

For each of the following systems, identify examples of data, user interfaces and processes present in each case. Try to list as many as you can.

1. A fitness device worn on the wrist that continuously monitors the wearer's activity, for example their heart rate and number of steps taken, and gives feedback on the user's fitness level.
2. An automated tyre pressure monitoring system in a car, which alerts the driver if the pressure in any of the tyres drops below an acceptable level.

Computer systems are everywhere and will play an increasingly important role as time goes by. It is likely that in the future more people will live in so-called smart homes. A smart home will have a number of strategically-placed sensors about the house that can be used to monitor the behaviour of the occupant. Hence there may be a sensor on the door of the refrigerator or on the kettle. These sensors might be used to monitor the kettle being switched on or the refrigerator door being opened. Should any of these sensors be activated due to 'unusual' behaviour then this can trigger a message to the relevant parties. Unusual behaviour might even be the absence of input – hence if the kettle and refrigerator sensors have not been activated (say) by mid-morning it might be indicative that the occupant is still in bed and may be unwell.

1.7 User Interfaces

We have already seen that the needs of the user are vital to the success of a software system. Therefore, the user interface is a vital element of the computer system. The user interface forms the gateway between the system and its user. It allows the user to enter data and, after processing, it outputs the results to the user.

There are many ways of interacting with computer systems and indeed these have changed significantly over the years. In the past, much of this interaction took place via a combination of a mouse and keyboard. In more recent times user interfaces have become more sophisticated and now include elements such as voice control and touchscreen technology.

Many organisations with high security needs have replaced physical key access to restricted areas of their buildings with swipe-card access, using a card with a magnetic strip that holds encoded information. As the card is swiped through the slot, the encoded information is interpreted and, if the user is authorised, they are given access. It is likely that, in the future, interaction using gestures will become increasingly commonplace for such applications.

A common user interface used by supermarkets is the barcode. Many supermarkets today are almost totally reliant on barcodes on the packaging of their products to process sales at the checkout desk. A barcode consists of a series of parallel black and white lines and each product's data is uniquely encoded by varying the thickness of and spacing between these parallel lines. As a customer passes through the checkout, specialised barcode readers are used to scan and interpret these barcodes in order to display the product's details on the screen and process the sale. From a supermarket's point of view this:

- speeds up the checkout process;
- significantly reduces the possibility of human error;
- maintains an up-to-date record of stock levels which is vital for re-ordering from suppliers.

A barcode reader is a very different user interface from a keyboard and mouse. Indeed, every system will have a different interface. The decision of what type of interface to provide for a computer system is dictated by:

- the application;
- the environment in which it operates;
- the characteristics of its users.

Whatever the choice of user interface, it must be simple to use and extremely intuitive. Figure 1.3 contrasts two systems with very different user interface needs. Typically, users of an airport check-in kiosk only have to supply small amounts of data, so a touchscreen interface is appropriate. By contrast, a

	Airport Check-in	Hospital
Application	Check passengers onto particular flights	Update patient records
Environment	Kiosk in a busy airport	Hospital
Reason for using the system	To notify the airline that the passenger has arrived at the airport	To record patient data
Interface	Touchscreen	Screen, keyboard and mouse

Figure 1.3: Comparison of two typical user interfaces.

system used by staff in hospitals to record patient information requires many fields of data to be completed. The interface therefore needs to allow more sophisticated input, and hence a screen, keyboard and mouse are appropriate. While such a system would likely appear difficult to use to members of the public, it is only intended to be used by trained hospital staff and hence serves its purpose.

User interfaces change and evolve over time. This can happen rapidly, depending on the needs of the user that moment, or it can happen slowly as technology and society evolve. The following two case studies show two ways that this can happen.

Case Study

Satellite Navigation System

Consider a car satellite navigation (satnav) system that provides the user (the driver) with information on the route to a specified destination. The system has a number of different interfaces. The first interface the driver encounters when they turn on the satnav allows them to enter details of their desired destination. The driver must be able to concentrate fully on providing accurate destination information, so naturally this data must be entered when the car is not moving. At this point, **accuracy** is important and since they can focus their attention on the task the user will probably use a touchscreen keypad to enter details of the destination.

Once the destination has been entered and the system has calculated the appropriate route the driver can commence their journey. The satnav system will provide the driver with a series of instructions as to the precise route to be followed. However, as the driver is now focusing primarily on **driving safely** the user interface changes dramatically. A satnav interface that simply displayed the route instructions as text on a screen and required the driver to read it while driving would be dangerous. Instead, instructions are issued both visually (via an on-screen map) and orally (via a voice commentary). The main reason for the audio interface is that the user can receive instructions without taking their eyes off the road.

Case Study

Toll Road System

Toll roads are typically constructed by private organisations to provide an alternative and faster route between two locations. To pay for their investment, a fee is paid for the use of the road – this is the toll. Toll roads are increasingly common in Ireland.

In the early days of toll roads, the toll was collected manually. A barrier was constructed at some point on the road to prevent road users from progressing until their toll was paid. The toll transaction involved the driver physically handing over payment before the barrier was raised and the driver permitted to proceed with their journey.

An initial attempt to speed up the process was to have unmanned coin baskets at the barrier, that drivers simply dropped their coins into. The baskets did not issue change and were quicker to use, though drivers still had to have their coins ready in advance and wait for the barrier to rise.

As toll roads became ever more popular it was observed that the time taken for the payment process was still too long, resulting in long queues of traffic waiting to pay. Such delays were slowing down journey time and thus defeating the purpose of the roads, so an alternative mechanism was devised to speed up the payment process. On the M50 in Dublin number plate recognition is now used to identify users as they pass the toll point. Users do not have to stop and instead go online to pay their toll within a particular time frame.

1.8 System Development Roles

The development of any complex computer system requires a team of people with different skills to analyse the problem. Three of the most common roles are the **systems analyst**, the **project manager** and the **programmer**.

Systems analyst

The main role of the systems analyst is to identify the **user** and **data processing requirements** for the new or modified computer system.

Typically, the systems analyst will:

- carry out a feasibility study and produce a feasibility report;
- liaise with the client (user);
- undertake some fact-finding (using interviews, observation, questionnaires or document sampling if there is an existing system);
- define the system specification, i.e. the specific proposals for a new, modified or replacement system.

Project manager

The project manager is responsible for the **planning**, **management**, **co-ordination** and **financial control** of the project. They must ensure the project is completed on time and within budget, that the project's objectives are met and that everyone else is doing their job properly. Project managers oversee the project to ensure the desired result is achieved, the most efficient resources are used and the different interests involved are satisfied.

Typically, the project manager will:

- plan the project's schedule;
- split the project up into suitable tasks/subtasks;
- allocate resources such as personnel, hardware and software to each task/subtask;
- identify risks to the project;
- monitor the progress of the project;
- ensure the project is delivered on time and within budget;
- report back to the client.

Programmer

When creating a new system, the role of the programmer is to **develop** and **test** the system designed by the systems analyst to meet the particular needs of the client.

Typically, the programmer will:

- write or amend source code (using a high-level programming language such as Java) for the proposed system based on module specifications, algorithms or flowcharts;
- document the code (add meaningful comments) to help other programmers when the code needs to be modified;
- debug the code to correct any errors that may have been found during testing;
- ensure integration of existing software systems (if these exist);
- test the new system using a test plan;
- write operational documents for the new system; and
- work closely with the **systems analyst** and **project manager** to ensure the user requirements are met within the expected time frame.

A programmer involved in the development of code for a particular system would be well-placed to be involved in the future maintenance of code for that particular system. However, their expertise would not be confined to that one system and so they are likely to be of great value in the enhancement and/or maintenance of code for other systems that they were not originally involved in developing. The more experienced a programmer is, the more likely they are to be involved in the development and maintenance of more complex systems.