

In this chapter you will learn to:

- evaluate the extent to which a proposed system will meet user needs
- evaluate the effectiveness of using existing software
- identify the parts of the proposed system that require software to be designed and developed
- identify a relevant approach for a given problem
- develop and interpret design specifications from a user's perspective
- recognise the difference between the user's and developer's perspectives and the communication issues that may arise
- differentiate between forms of systems documentation and the purposes for which each is used
- describe a system by interpreting its diagrammatic representation
- create a diagrammatic representation for a system using appropriate modeling tools
- effectively communicate with users regarding a proposed software solution
- identify a range of criteria on which the quality of the product will be judged
- identify relevant processes for a given criterion that will result in a quality product

Which will make you more able to:

- differentiate between various methods used to construct software solutions
- identify and evaluate legal, social and ethical issues in a number of contexts
- construct software solutions that address legal, social and ethical issues
- identify needs to which software solutions are appropriate
- apply appropriate development methods to solve software problems
- apply a modular approach to implement well structured software solutions and evaluate their effectiveness
- apply project management techniques to maximise the productivity of the software development
- create and justify the need for the various types of documentation required for a software solution
- select and apply appropriate software to facilitate the design and development of software solutions
- assess the skills required in the software development cycle
- communicate the processes involved in a software solution to an inexperienced user
- use and describe a collaborative approach during the software development cycle
- develop and evaluate effective user interfaces, in consultation with appropriate people.

In this chapter you will learn about

#### **Defining the problem**

- identifying the problem
  - needs of the client
  - functionality requirements
  - compatibility issues
  - performance issues
  - boundaries of the problem

#### **Issues relevant to a proposed solution**

- determining if an existing solution can be used
  - social and ethical considerations
  - consideration of existing software products
  - customisation of existing software solutions
  - cost effectiveness
  - licensing considerations
- selecting an appropriate development approach if there is no appropriate existing solution

#### **Design specifications**

- specifications of the proposed system
- developer's perspective in consideration of:
  - data types
  - data structures
  - algorithms
- user's perspective
  - interface design
  - social and ethical issues
  - relevance to the user's environment and computer configuration

#### **System documentation**

- representing a system using systems modeling tools, including:
  - IPO diagrams
  - context diagrams
  - data flow diagrams (DFDs)
  - storyboards
  - structure charts
  - system flowcharts
  - data dictionaries
- algorithms used to document the logic in modules and subroutines
- test data and expected output

#### **Communication issues between client and developer**

- the need to consult with the client
- the need to incorporate the client's perspective
- the need for the developer to enable and consider feedback
- the need to involve and empower the client during the development process

#### **Quality assurance**

- the need to explicitly define the criteria on which the quality of the product will be judged
- putting in place management processes to ensure that quality criteria will be met
- an ongoing process throughout development to ensure the quality criteria will be met

## DEFINING AND UNDERSTANDING THE PROBLEM

Defining and understanding the problem is the first stage of the software development cycle. In many ways, this initial stage is the most important. Formally identifying the precise problem to be solved and determining if an existing solution is viable or a new solution is needed are vital steps that are often overlooked and perceived as trivial by some large software developers. Studies have shown that some 30% of software projects are abandoned well into the development cycle. The majority of these cancelled projects are well over budget and clearly were not feasible from the outset. Software developers must take steps to carefully define and understand the problem before moving onto the planning and design phase.

In this chapter, we look at important considerations when identifying the precise problem to be solved. We examine how to identify and specify the problem so it meets the needs of the client – including the importance of client-developer communication. We also look at different methods of documenting and modelling the proposed new system. Finally we consider quality assurance. Quality assurance is an ongoing process during development which monitors and assesses the standard of the developing solution against a set of quality criteria.

### DEFINING THE PROBLEM

Problems can come from a number of different sources. It may be that an outside company is looking to contract a software developer to construct a solution for their business. It could be a new idea for a product. In either case, the problem itself needs to be defined precisely so that both the developer and the client understand what will be done.

Some questions that need to be asked to help define and identify the problem are:

- What are the client's needs which will be met by this product?
- Compatibility issues with other existing software and hardware?
- Possible performance issues, particularly for internet and graphics (or video) intensive systems?
- What are the boundaries of the new system?

These questions need to be answered to clearly identify the problem prior to creating a set of design specifications.

### NEEDS OF THE CLIENT

A need is an instance in which some necessity or want exists. The implication is that some form of solution is required to meet this need. Without actually articulating the needs clearly, it will be difficult to develop a clear picture of the precise problem to be solved. In most cases the needs of the client are the primary reason software projects

are developed. Often such needs are initially expressed as general statements, such as “We need to be able to monitor sales made at each of our stores”. Clearly such general statements require significant consultation to determine how such monitoring can be performed. Through consultation and analysis a set of functionality requirements is created. Meeting these requirements is the central aim of the entire software development process.



Consider the following:

A window cleaning business, known as ‘Mr Window Cleaner’, has just completed an advertising campaign during which they have dropped 10,000 fridge magnets in letterboxes throughout the northern and eastern suburbs of Sydney. They claim, on the magnets, that a window cleaner can be on the job within 3 working days. The magnet drop is so successful that they are inundated with work. They are so busy that they are unable to even call back each potential customer within 3 days, let alone get a window cleaner to the job.

Let us consider the needs from this scenario.

The potential customer needs:

- To have their windows cleaned professionally
- To be contacted to organise a time for the work to be done
- To have the work completed within 3 working days
- To be charged an acceptable fee

The business ‘Mr Window Cleaner’ (the client) needs:

- To make a profit
- To fulfil customers’ needs
- To contact customers quickly
- To complete each job within 3 working days
- To continue to service existing customers



#### **GROUP TASK Discussion**

A software developer may immediately see possible methods of meeting these needs. For example, setting up a computerised voice mail service to track client enquiries, developing a database of clients, automated scheduling of work, etc. The software developer is really expressing their ‘need’ to be part of the solution. Is the software developer ultimately going to meet the needs of ‘Mr Window Cleaner’ and their customers by implementing systems such as those mentioned above? What do you think? Consider other non-software alternatives open to the business as part of your discussion.



#### **GROUP TASK Discussion**

Do the customers really require their windows cleaned within 3 working days? Suggest ways of determining if meeting this need should be a high priority for the business.

Often needs are expressed in non-specific, even emotive terms. For example, we need to improve customer relations, or we want to improve our turnaround times. Analysing needs is an important aspect of beginning to define the problem and for large systems a professional systems analyst is used. A number of tools and techniques are available for accomplishing the analysis.



### Systems Analyst

A professional who analyses systems, determines requirements and designs new information systems.

- Surveys - Completed by all key personnel. For example, managers, IT personnel, users and clients. Survey's are an excellent tool for gathering information from large numbers of people, however they often limit the detail and explanations participants are able to provide.
- Interviews - Personal interviews will allow participants to more freely express their needs. Detailed responses and explanations are possible, however performing interviews is labour intensive and hence expensive.
- Time management studies - How much time is really spent on each specific function. Actually watching and recording the time, nature and order of tasks as they are performed will often highlight priority areas.
- Business analysis - Examining different aspects of a business' activities in search of areas where improvement can be made. Often such analysis will include a cost benefit analysis to ensure a profitable return on the software investment is likely.

Many developers have great ideas for new products. This is all well and good, but is there a need for this product? Developing a new product of any type without establishing a need for the product is like putting the proverbial cart before the horse. Many great inventions have remained as ideas because no market or need exists for the product. Conversely, many fine products appear which fail economically because they don't meet the needs of the market place.



Consider the following:

A software development company is exploring the possibility of producing a computer controlled, in-home system. This system will use a personal computer to control the activation and operation of various electrical appliances within the home. It will be able to control the home's security system, lights, air conditioning, TV, video and various kitchen appliances. For example, when you leave the house the system will put all appliances into a sleep mode, and the air conditioning will automatically cool the house prior to your normal time of arrival in the evening.



### GROUP TASK Discussion

How could the software development company establish if there was a need amongst potential customers for a system such as this?



### GROUP TASK Debate

The company finds that potential customers don't really understand how the new computer controlled system can be of benefit. Should the company attempt to educate the public about this product and hence, hopefully, create a need, or should they cancel the project?

## Functionality requirements

Functionality requirements describe what the system will do. They are what you are aiming to achieve. Requirements are statements that if achieved, will allow needs to be met. The requirements of a system give direction to the project. Throughout the design and development of a system the functionality requirements should continually be examined to ensure their fulfilment. The final evaluation of a project's success or failure is based on how well the original requirements have been achieved.



### Requirements

Features, properties or behaviours a system must have to achieve its purpose. Each requirement must be verifiable.

To be able to evaluate the degree of success or failure of requirements requires that the requirements be stated in such a way that they can be readily verified. In terms of software development, this often means they must be quite scientific or mathematical statements. For example, a requirement may be: 'The system will be able to redraw the screen at least 12 times per second' or '95% of orders will be filled within 24 hours'. Each of these requirements can be precisely tested.



Consider the following:

A small mail order business has grown to a stage where it is no longer viable to use a manual ordering system. The business has established a list of needs as follows:

1. Orders must be processed more efficiently.
2. Overdue accounts need to be dealt with more effectively.
3. Items on backorder should always be filled before new orders.
4. Customers need to feel confident their personal details will remain confidential.
5. Personal attention to customers must be maintained.

After further discussion and analysis a set of functionality requirements is developed as follows:

- 90% of orders will leave the premises within 48 hours. Customers whose orders are not filled in this time are contacted by phone.
- Overdue accounts are generated automatically after 30 days.
- Accounts over 60 days are contacted personally. If a resolution is not found they are referred to a collection agency.
- Stock to be entered into the computer system before it is placed in the warehouse.
- Backorders are always checked before items are allocated to orders.
- Password protection on the customer details file.



### GROUP TASK Discussion

Describe the essential differences between the above needs and the functionality requirements that follow.



### GROUP TASK Discussion

Can you suggest further requirements that may assist in ensuring personal customer attention is maintained?



### Compatibility Issues

Software of various types runs on a variety of operating systems, browsers, hardware configurations and even a range of different devices, such as smart phones and tablet computers. Users are able to configure their device in a variety of manners, for example screen resolution, colour depth and font sizes. Today many software products require and use internet or LAN connections. Such connections operate at varying speeds and will often encounter errors or even complete loss of connectivity. When designing software developers must ensure their products are compatible with a wide range of likely user devices and network conditions.

Some examples of common compatibility issues include:

- Problems with different versions of the intended operating system, particularly as users upgrade their operating system.
- COTS products which experience issues when the underlying application is updated.
- Not all graphics cards are supported by many graphics code libraries.
- Screens which do not resize to support different screen resolutions.
- Labels for screen elements which overlap or are no longer aligned correctly when fonts are enlarged.
- Loss of server connection causes fatal errors over wireless networks.
- Browsers which do not correctly implement HTML standards set by the W3C (World Wide Web Consortium).

Any existing hardware, software and communications systems must be documented and then taken into account when defining the problem to ensure the solution will be compatible with these existing resources.

### Performance Issues

Often the specifications of the computers used by developers far exceed those likely to be present in a typical user's computer. In addition multi-user applications and applications that access large files and databases will perform very differently under real world conditions. Testing environments and actual tests should simulate real world conditions. We consider system testing in detail as part of chapter 6, however all testing, including performance testing should be ongoing throughout development.

When defining and understanding the problem it is common for performance requirements to be included. For example, "Online payments must be approved within 5 seconds" or "The application will load in less than 3 seconds". For custom systems written for known hardware and communication lines it is possible to ensure such time sensitive requirements are met with reasonable certainty, however when developing for a broad and largely unknown audience it is much more difficult. Nevertheless it is critical that developers use the most efficient algorithms for the task. If processing has the potential to take some time then visual aids should be used on the user interface to indicate the likely time required. In some cases time consuming processing can be performed in the background whilst the user performs other tasks.

Some common examples of performance issues include:

- The computer appears to be not responding after some function has been initiated. In fact it is busy processing a time consuming task.
- Users experience poor response times. This is often seen with networked applications where during data entry users spend most of their time waiting for the application.

## BOUNDARIES OF THE PROBLEM

Boundaries define the limits of the problem or system to be developed. Anything outside the system is said to be a part of the environment. The system interacts with its environment via an interface. Input and output to and from the system takes place via an interface. The keyboard provides an interface that allows humans to input data into a computer system. An internet service provider (ISP) provides an interface between computers and the internet.

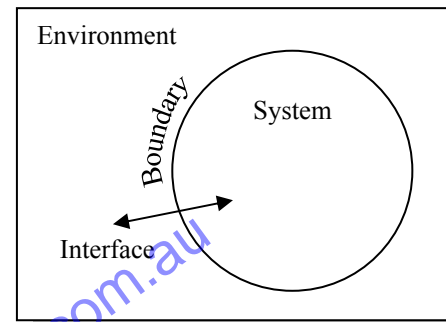


Fig 3.1

*The boundary defines the limits of the system. Interfaces allow communication with the environment.*

It is vital to determine the boundaries of a problem to be solved. Determining the boundaries is, in effect, determining what is and what is not a part of the system. Items that are not part of the system, that is items that are in the environment, need to be considered if they have an influence on the system. For items in the environment to influence the system there must be an interface with the system. Items in the environment can affect the system, however the system cannot alter its environment.

When defining a problem it is important to define the boundaries for the problem so that the customer has realistic expectations of the limits of the new system.



Consider the following:

A company that specialises in the production of custom kitchen cupboards is computerising its cabinet making processes. Each cabinet-maker will have a dedicated terminal that will provide him or her with details of the next cupboard they are to construct. Once construction is completed, the tradesperson enters the time of completion into his or her terminal and awaits allocation of their next job. The terminals use a proprietary operating system. You, as a software developer, are employed to develop the software for this system.

There are a number of limitations or boundaries apparent to you as a software developer, namely:

- Control over the dimensions of cupboards delivered via your software to the cabinet-makers. The software can only send the dimensions entered by the kitchen design staff.
- Ability to determine the time taken to build individual cabinets is dependant on the cabinet-makers efficiently entering jobs as complete at the correct time.
- As the terminals to be used by the cabinet-makers use a proprietary operating system, the functioning of the software is dependant on the capabilities of this operating system.



### GROUP TASK Discussion

Explain how each of these boundaries may affect the overall success of the final system. Use examples of possible scenarios to illustrate your answers.

## ISSUES RELEVANT TO A PROPOSED SOLUTION

Although a new solution may well meet the current identified needs there are other areas that should be considered prior to investing time and money in the development of a new software product. Will a new solution provide enough advantages to the organisation to be worth the time, effort and money involved in its development and implementation?

Points to consider from the client's perspective include:

- Will the existing system be able to perform the required tasks in the foreseeable future? If so, then maybe the new system should be cancelled until a later date.
- Will the proposed new system meet future needs? If not, then maybe the requirements need to be changed to reflect longer term needs.
- Have existing similar solutions been examined? Perhaps a suitable solution already exists or an existing software package can be customised.
- How crucial is the new system to the total organisation? Perhaps resources would be better utilised in other areas.

Some points to be considered by the software developer include:

- What expertise is required to complete this project? Do we already have this expertise and if not then can we get it?
- What resources are required to develop this project? Do we have these resources and if not can we get them?
- Will we be able to support this product in the future?
- Can we retrain/redistribute existing staff and resources to this project? How would this affect other projects?

Different methods and approaches to the solution of a problem should be considered prior to commencing design and implementation. It is important that each method or approach to the solution be targeted at achieving the same set of requirements. This allows a fair judgement to be made as to which method or approach is preferable. It is wise to research existing software products as often an existing product will be a more cost effective solution. Today many software products can be customised to tailor their functionality to suit different user needs. For instance open source products can be altered to include new functionality or existing functions can be customised. Specialised functionality, such as artificial intelligence, can be outsourced or existing code modules can be obtained.

Often a method of solution is developed without consideration of other possible solutions to the problem. For example, adding another lane to a motorway may seem the obvious way to reduce traffic congestion. However, a better solution may be to encourage businesses to relocate from the city centre, or perhaps an upgrade of the public transport system should be considered. If a software solution is required there are usually a number of possible solutions to the same problem. Many consultancy services are set up for precisely this purpose. Their job is to examine different methods of approach and solutions to particular problems.



### GROUP TASK Discussion

There are numerous commercial software packages which include code and even complete programs developed by other companies. Identify example software where this is the case. Explain why the development company may have decided to use code from others within their software package.



When comparing development of new software with the use of existing software solutions we should consider the following areas:

- Social and ethical considerations
- Legal issues including licensing considerations
- Customisation of existing software products
- Cost effectiveness

#### **Social and ethical considerations**

Some broad social and ethical areas that require consideration include:

- Changing nature of work for users
- Effects on level of employment
- Effects on the public

The social and ethical issues will vary depending on the problem at hand.

#### **• Changing nature of work for users**

The introduction of most new software products has an effect on the nature of work for users of the system. We must consider the effect the new system will have on the users. Input from potential users is critical at this stage. Ultimately, these are the people who will spend the most time with the completed product so their input is of untold value.

If the product is to be implemented across large numbers of sites, perhaps industry associations and/or unions need to be involved in the consultation process. Often, employees will be on contracts that list the tasks expected of them, and often these contracts will require re-negotiation due to changes in the nature of employees' work. Employees may need to be retrained to acquire skills for their new or changed position.

#### **• Effects on level of employment**

In the 1970s, a major concern was that computers were going to substantially replace the work done by many people. In many ways this has occurred, though in general computers have created many more positions that have replaced many labour intensive and repetitious jobs. One of the main reasons for creating a new software system is to reduce a business' costs. One of the largest costs for most businesses is salaries and wages, hence many new software products are designed to reduce the overall level of employment in a particular area. Developers and their clients must take this into account, both in terms of cost savings but also in terms of the effect on employees made redundant.

#### **• Effects on the public**

Large software systems can have a substantial effect on the general public. The introduction of automatic teller machines (ATMs), involved retraining the entire population. Many older people were reluctant to make use of this new banking system. The internet has changed the way many of us communicate and obtain new knowledge. Computerised billing systems have allowed us to pay bills without ever having to enter the premises of the billing company.



*Fig 3.3  
The introduction of ATMs involved retraining the entire population*

Effects on the public can be both negative and positive. It is vital that consideration be given to how a system will affect the public during this initial stage.



Consider the following:

During the early 1980s most large supermarkets changed from a manual pricing system to a computerised system using the barcodes on individual items. More recently supermarkets and other retailers have installed self-checkout systems where only a single employee is required to monitor 10 or more checkout stations.



#### **GROUP TASK Discussion**

What effect have these changes had for supermarket employees? Explain your answers.



#### **GROUP TASK Discussion**

How has the new pricing and checkout systems affected shoppers? What measures have been introduced by supermarkets to deal with these problems?

### **Legal issues including licensing considerations**

All parties must consider issues related to copyright (see chapter 1), both in terms of protecting the rights of the developer and the legal rights of the customers who will purchase and use the product. Software that is used to store and access private and personal information will need to include safeguards against unauthorised access.

When considering the use of an existing software solution it is worth taking the time to contact other users of the product, either directly or via online user groups. Consider also the level of support provided by the developer and whether it is possible to request new functions. The terms and conditions of the licence agreement should be read and understood. Open source licences, although free of direct costs usually do not provide direct support. Some open source products are also distributed commercially in which case professional support and training is routinely available. For businesses open source licences are not always the most cost effective solution. The extra effort required to support and troubleshoot any issues can tip the balance in favour of commercial products.



#### **GROUP TASK Debate**

“Surely the use free of open source software will always be a cheaper alternative to commercial solutions.”  
Do you agree? Debate using examples to support both sides.

### **Customisation of existing software products**

Customisation of an existing software product is often a cost effective strategy for obtaining new functionality. Many software developers spend much of their time modifying their own existing products to suit the specific needs of individual clients. For example, Parramatta Education Centre writes custom database applications often using Microsoft Access to build the data entry forms and printed reports. Much of the routine work performed by the company involves upgrading existing products to include additional features. In this case Microsoft Access is being used to create and update different COTS (Customised Off-the-shelf) products.

Open source software is also routinely customised to add new features. For example, phpBB is a popular open source product used to build online forums. Many of the forums based on phpBB have been customised to suit the unique requirements of particular forums. In many cases the modifications are built as add-ons which are then made available to other users of the base software product. For example, an add-on for phpBB awards points to users based on the number and quality of posts they make. Other add-ons allow paid advertisements relevant to the subforum being viewed to be embedded within screens.

The ability to customise software helps the original developer of the product widen their market. It is now common for tools to be included in many commercial products which allow the end user to create customised screens and other functionality through the use of wizards and various drag and drop design screens. Other companies who specialise in software for particular industries offer their own software customisation service. For example, GeoCivil is a software product designed for use by civil engineers. The developers (Geocomp Consulting) offer a customisation service where GeoCivil can be altered to suit particular requirements.



#### **GROUP TASK Research**

Using the internet (or otherwise) research examples of software which can be customised. Will customising existing software products always be a more cost effective solution compared to developing a new product?

#### **Cost effectiveness**

Usually one of the constraints of a new software system will be that it falls within a certain budget. If it is not a requirement then it will most likely be an absolute constraint. This area aims to accurately assess and predict the costs involved in the development of a new software project compared with using or modifying an existing product.

Some areas of costing that should be considered include:

- Development hardware costs? Any new hardware that needs to be purchased/leased to enable development.
- Development software costs? What software is required to develop the new system? Possible purchases may include programming languages, CASE tools, database management systems, graphics creation tools, etc.
- Development personnel costs? Essentially the salaries of the people on the development team. Also consider support staff, trainers, management and office staff.
- Outsourcing costs? Will parts of the project be outsourced to other businesses? What are the costs involved?

Once estimates and quotations have been obtained, an overall development costing can be compiled. This total development cost is then compared to the allocated budget for the project and the total cost is assessed.

How is the budget for the project obtained? Often clients for whom the project is developed will provide a budget. A budget for a project is the amount of money that is allocated for the development project. Some factors that need to be considered when formulating a budget include:

- The available capital.
- The predicted sales for the finished product.
- Cost savings as a result of the product's implementation.
- Predicted future upgrade and maintenance costs for the product.
- Ongoing customer support costs.
- Other costs. For example, special hardware required to operate the system, training costs for new future users.

It is very common for large projects to go over budget during their development. This puts management in a difficult position: a large amount of money has been spent and to scrap the project would result in large losses. To help overcome this eventuality, many budgets will include extra funds for unforeseen circumstances. It is common to add some 20% to initial costings for unforeseen circumstances. Some large projects have been known to go over budget by as much as 200%, this being particularly the case when developing new products that use new untested technologies. The American 'Star Wars' defence system is an example of a project that went way over budget and finally did not realise its stated objectives.

Recommendations resulting from questions of cost effectiveness will often involve redefining the requirements of the systems so development can proceed within budget. Perhaps some aspects of the product will be earmarked for inclusion in future upgrades once the product has an established market.



Consider the following:

A local council wishes ratepayers to be able to pay their bills on the internet. The council advertises for expressions of interest from software developers. The council has limited funds to allocate to the project and has already decided on a budget for the implementation of the system. An information pack from the council outlines the requirements of the system. As a software developer, you are interested in tendering for this job. You are currently considering applying to develop this product.



#### **GROUP TASK Discussion**

Develop a list of questions that need to be answered to determine whether the software can be developed using the available budget.



#### **GROUP TASK Discussion**

Suggest some criteria the council may use to decide which software developer should be given the contract to develop this product.

### **SELECTING AN APPROPRIATE DEVELOPMENT APPROACH**

If an appropriate existing solution cannot be found then new software will need to be developed. This may involve developing the entire solution or it may involve writing code to customise an existing solution. When new software is developed one of the first tasks is to identify a suitable software development approach. Chapter 2 of this text describes a variety of software development approaches together with their strengths and weaknesses. In many cases a combination of approaches will be appropriate.



HSC style question:

For each of the following scenarios, recommend and justify a suitable software development approach (or combination of approaches).

- A breeder of an exotic type of cat wishes to record details of all cats and their kittens so they can create a family tree for each kitten they sell.
- A gift store wishes to commence trading over the internet. The owner has some design ideas for the website based on their current marketing, however they have concerns about protecting financial transactions performed over the internet.
- The manufacturer of a new printer requires a USB driver to be written that makes use of all the printer's additional features not available in standard USB drivers.

#### Suggested solutions

- If the cat breeder is computer literate and able to locate and use proprietary software, it is likely they will be able to locate a suitable package through the internet that meets her needs exactly. There are plenty of 'family tree' packages that she could use for her purposes. The approach is therefore use of an application package that she may be able to customise to her specific needs. This would require use of an existing package together with the end-user approach to customise the product to meet her specific requirements.

Another possible approach would be for her to utilise an integrated database management system, such as Microsoft Access to store the details – this would also constitute an end-user approach. If she is not capable of defining and correctly implementing the required relationships, she could ask an IT person to develop the necessary application for her using the forms and coding language of the package, thus constituting a RAD approach.

- The recommended approach here would be a prototyping approach to ensure that the web interface meets the client's needs exactly. The final interface designs would be developed in an iterative process using the owner's initial design ideas, with the developer progressively refining the design to incorporate both the processing requirements and effective design skills. The iterative process requires that the owner and developer work together on the design, with the designer listening carefully to the feedback from the owner at each point to ensure that the final agreed design reflects their requirements exactly.

The concerns about security can be addressed by the developer incorporating a set of routines that ensure appropriate security. These routines could either be developed specifically for the gift store website (probably using a structured approach) or more likely, the developer would make use of existing encryption routines or services which are well documented and tested.

- The logic required for this driver is very specific and the requirements can be defined precisely in advance. A structured approach is therefore suitable. Because this software will be distributed with new printers, it is very important that the solution be a rigorous solution, designed with the use of very formal stages. The result being a well designed, efficient, fully tested and documented system.

*Correct answers could conceivably recommend a variety of different approaches. Marks would largely be awarded for justifying the chosen approach for the given scenario.*



**SET 3A**

1. Needs can be described as:
  - (A) an instance where some necessity or want exists.
  - (B) what you are aiming to achieve.
  - (C) the limits of a system.
  - (D) factors with which the system must comply.
2. Why should requirements be verifiable?
  - (A) To ensure the project remains within budget.
  - (B) So the level of success or failure of the project can be evaluated.
  - (C) So developers and clients reach agreement with regard to what the system will do.
  - (D) To select the best approach to solving the problem.
3. Which of the following is an example of a performance issue?
  - (A) Problems when the operating system is upgraded.
  - (B) Features the new system must have are missing.
  - (C) Poor communication between developers and clients.
  - (D) Slow response to user inputs.
4. Which of the following is an example of a functionality issue?
  - (A) Problems when the operating system is upgraded.
  - (B) Features the new system must have are missing.
  - (C) Poor communication between developers and clients.
  - (D) Slow response to user inputs.
5. Which of the following is an example of a compatibility issue?
  - (A) Problems when the operating system is upgraded.
  - (B) Features the new system must have are missing.
  - (C) Poor communication between developers and clients.
  - (D) Slow response to user inputs.
6. Boundaries can be described as:
  - (A) an instance where some necessity or want exists.
  - (B) what you are aiming to achieve.
  - (C) the limits of a system.
  - (D) factors with which the system must comply.
7. Items within the boundary, outside the boundary, and communicating across the boundary. Which of the following lists these in the correct sequence?
  - (A) system, interface, environment.
  - (B) interface, system, environment.
  - (C) system, environment, interface.
  - (D) environment, interface, system.
8. Which of the following is true with regard to developing new software compared to using an existing solution?
  - (A) Developing new software is cheaper than purchasing existing software.
  - (B) Developing new software is always preferable to using existing software.
  - (C) In many cases developing new software will better meet the unique needs of the client.
  - (D) Often existing software can be used at no cost to the client.
9. Consideration of development costs in regard to hardware, software, personnel and outsourcing is part of establishing:
  - (A) cost effectiveness.
  - (B) functionality requirements.
  - (C) system boundaries.
  - (D) customer needs.
10. Which of the following types of software generally lacks personalised customer support?
  - (A) Commercial software.
  - (B) Customised off-the-shelf software.
  - (C) Commercially distributed open source software.
  - (D) Free open source software.

11. Imagine your Software Design and Development class as a system. It contains physical components such as windows, a door, lights, the computers, tables and chairs. Students in the class and the teacher, this textbook, lighting, sunlight, electricity, network connections, certain times during the week when the class is timetabled, all these things contribute to the system.
- List all the items that are within the control of the class system.
  - List all the items that are in the environment of this system.
  - List any items that allow the class system to interface with its environment.
  - Describe the boundaries of this system.

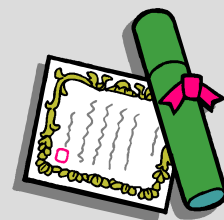
12. Defining the problem precisely upfront is often thought to be a waste of time and resources. Do you agree? Give reasons for your answer.

13. The water board currently reads water meters manually. An employee walks from house to house recording the meter reading in a notebook. These readings are later entered into a computer at the office. Because the meter readers do not have access to previous readings, they are unable to determine abnormal usage for any individual meter. If they had access to the previous reading they could determine major water leakages as well as properties that have apparently used no water. This information would allow the meter reader to check the operation of the meter whilst on the site.

A new computerised system is being contemplated.

- What are the needs from the water board's point of view?
  - Transform these needs into more concrete possible functionality requirements for a new meter reading system.
14. Currently students completing the HSC in NSW receive their ATAR in December and offers of university places soon after. Many students would like to have an idea of the ATAR result they are likely to achieve after they complete their Trial HSC exams. The main purpose of this 'Trial HSC ATAR' would be to assist with decisions in regard to university. Students would also like to know any courses where extra work would best help in maximising their final ATAR.

- What are the needs of the students?
- Assume you are considering creating a software package to meet these needs. What would be the requirements for such a system?
- Outline any social and ethical issues that would arise as a result of such a system.



(Note: Parramatta Education Centre markets a product, 'The ATAR Estimator', available only to schools, that performs these functions).

15. A group of your friends have approached you with an idea for a new software product. This product will essentially be an integrated student information system. It will incorporate a student diary/homework system, a mark keeping system and a note taking system. You are currently considering the possibility of developing this product.
- Is this problem worth solving? Describe advantages and disadvantages this new system would have compared to existing, traditional pen and paper techniques.
  - Do other existing software solutions already exist that solve this problem? Research to determine if similar products already exist.

## DESIGN SPECIFICATIONS

Developing a set of design specifications is one of the most important steps before the actual design is planned. The design specifications form the basis for planning and designing the solution. The aim of the design specifications is to accurately interpret the needs, requirements and boundaries identified into a set of workable and realistic specifications from which a final solution can be created. These design specifications should include considerations from both the developer's perspective and the user's perspective.

Developers' perspective	Users' perspective
Consideration of: <ul style="list-style-type: none"> <li>• Data types</li> <li>• Data structures</li> <li>• Algorithms</li> <li>• Variables</li> <li>• Software design approach</li> <li>• Quality assurance</li> <li>• Modelling the system</li> <li>• Documentation</li> </ul>	Consideration of: <ul style="list-style-type: none"> <li>• Interface design</li> <li>• Appropriate messages</li> <li>• Appropriate icons</li> <li>• Relevant data formats for display</li> <li>• Ergonomic issues</li> <li>• Relevance to the user's environment and computer configuration</li> <li>• Social and ethical issues</li> </ul>

Evaluation throughout the development process is undertaken with reference to these specifications. Specifications relevant to the user's perspective can be evaluated by the user, whereas specifications relevant to the developer can be evaluated by the developer.

### DESIGN SPECIFICATIONS FROM THE DEVELOPER'S PERSPECTIVE

These specifications will create a standard framework under which the team of developers must work. These are specifications that will not directly affect the product from the user's perspective but will provide a framework in which the team of developers will operate. The methods used to model the system - for example, use of system flowcharts, dataflow diagrams, IPO diagrams - will be specified. The method and depth of algorithm description to be used, how data structures, data types and variable names are to be allocated and any naming conventions that should be used should all be specified. A system for maintaining an accurate data dictionary needs to be specified. In other words, a framework for the organisation of the development process is set up so that each member of the development team will be creating sections of the solution that look, feel and are documented using a common approach (and of course so that each module will operate correctly with other modules). Often CASE tools will be utilised to ensure the team complies with these design specifications.

Once these specifications have been developed a system model can be created to give an overview of the entire system. These system models will lead to the allocation of specific tasks for completion by team members, whilst at the same time an overall direction of the project can be visualised.



#### GROUP TASK Discussion

What are each of the items listed under 'Developers' perspective' above?  
Why is it important to specify details of each of these items?

## DESIGN SPECIFICATIONS FROM THE USER'S PERSPECTIVE

Specifications developed from the user's point of view should include any design specifications that influence the experience of the end-user. Standards for interface (or screen) design will be specified to ensure continuity of design across the project's screens. For example, use of menus, use of frames to group items, use of colour, placement of common elements. The wording of messages, design of icons and the format of any data presented to the user need to be determined and a set of specifications created. For example, the font to be used for prompts and user inputs, the size and nature of icons used, the tone of the language used.

Ergonomic issues should also be considered and taken into account as part of the design specifications. Questions such as: Which functions will be accessed most often? Which functions require keyboard shortcuts? How should movement between screen elements be accomplished? What is the order in which data will be entered? Are the screens aesthetically pleasing? Such questions need to be examined and a standard set of design specifications generated.

The user's environment will also have an effect on the specifications created. Perhaps the users are familiar with existing applications. If so, these applications should be examined and some of their design elements incorporated so that transfer of skills from existing applications can take place. For example, using keyboard short cuts that are already known from other applications. Operating system settings should be used to determine the following: colour of windows, typefaces used and printer settings. Consideration of hardware necessary to execute the new product including: processor speed, RAM, video memory and hard disk space. Potential users need to be consulted to determine the acceptable specifications.

System models can assist in determining user based design specifications, in particular, screen designs and concept prototypes. It is vital that software developers acknowledge the contribution users make to the development of design specifications. Communication and in particular feedback from users is especially important at this early stage.



Consider the following:

A franchise based lawn mowing company wishes to have a software product developed to ensure all its franchisees quote and invoice their clients using a standard approach. The software needs to run on very basic MS-Windows machines with very limited RAM. The feasibility study has shown that the computer skills of many of the franchisees are very limited to non-existent.



### GROUP TASK Discussion

List and describe some of the design specifications from the user's perspective that need to be incorporated into this project.



### GROUP TASK Discussion

Concept prototypes are one method for gathering user specifications, however the limited computer skills of the users may limit the usefulness of such prototypes. Brainstorm and discuss other possible strategies that could be used to gather user specifications.

Evaluation Copy

Evaluation Copy

www.pedc.com.au

Pages 119 to 132

Not included in this sample chapter

Evaluation Copy www.pedc.com.au

Evaluation Copy www.pedc.com.au

www.pedc.com.au

Evaluation Copy www.pedc.com.au

Evaluation Copy www.pedc.com.au

pedc.com.au

pedc.com.au



ItemsSold. As these parameters use the symbol  $\sigma$  they are not control parameters and hence they do not alter the order of processing (or flow of control) within the algorithms. Rather these parameters are used to pass data into and out of subroutines.



#### GROUP TASK Discussion

Structure charts can be created from existing algorithms or even from existing source code. Examine the algorithms above to determine how they convert to the initial structure chart.



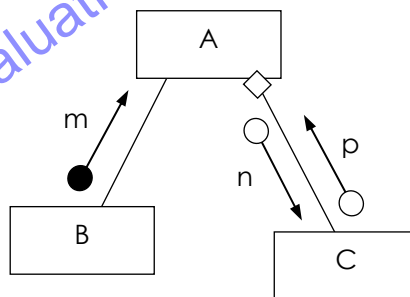
#### GROUP TASK Discussion

The PartID entered by the user is utilised by all the subroutines. Describe how the PartID is passed to each of the subroutines.



HSC style questions:

Consider the following structure chart when answering Question 1, 2 and 3.



1. Which of the following correctly identifies elements on the structure chart?
  - (A) m, n and p are processes, A, B and C are parameters.
  - (B) n and p are control parameters, m is a parameter, A, B and C are processes.
  - (C) m is a parameter, n and p are data items, A, B and C are variables.
  - (D) m is a control parameter, n and p are parameters, A, B and C are processes.
2. Which of the following best describes the structure chart?
  - (A) A calls B which returns a Boolean value for m. m is used to decide if C is called. If C is called it returns p after processing the value n.
  - (B) B calls A with the Boolean value m. m is used to decide if C should call A. If A is called it returns n after processing the value p.
  - (C) A calls B which returns a Boolean value for m. C is called repeatedly using n values. Each time C is called it returns a value for p.
  - (D) A calls B with a value for the parameter m. C is optionally called with the parameter n and returns a value for p.

3. Which of the following algorithms for A best reflects the above structure chart?

(A) BEGIN A  
        $B = \underline{m}$   
       IF  $m$  THEN  
           Get  $p$   
            $C = \underline{B}(n)$   
       ENDIF  
   END A

(B) BEGIN A  
        $\underline{B}(m)$   
        $\underline{C}(n, p)$   
   END A

(C) BEGIN A  
        $m = \underline{B}$   
       IF  $m$  THEN  
           Get  $n$   
            $p = \underline{C}(n)$   
       ENDIF  
   END A

(D) BEGIN A  
        $\underline{B}(m)$   
       IF  $m$  THEN  
            $\underline{C}(n, p)$   
       ENDIF  
   END A

#### Question 4.

The following algorithm updates the high score data at the conclusion of a game.

```
BEGIN UpdateHighScores(PlayerName,Score)
  LoadHighScores
  PlayerIndex = FindPlayer(PlayerName)
  IF PlayerIndex < 0 THEN
    InsertPlayer(PlayerName, Score)
  ELSE
    IF Score > Player(PlayerIndex).HighScore THEN
      UpdateScore(PlayerIndex, Score)
    ENDIF
  ENDIF
  WriteHighScores
END UpdateHighScores
```

Create a structure chart for the UpdateHighScores algorithm.

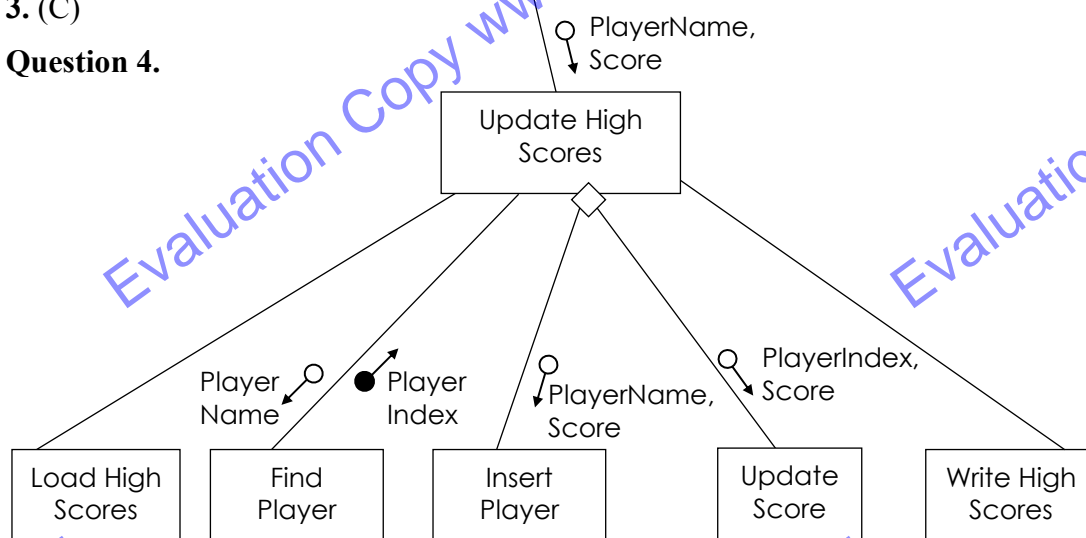
#### Suggested solutions

1. (D)

2. (A)

3. (C)

#### Question 4.



## INPUT PROCESS OUTPUT (IPO) DIAGRAMS

IPO diagrams are used to describe the data entering a process, the nature of the processing performed on the data and the resulting information leaving the process. IPO diagrams are used to further expand on the lower-level processes described on data flow diagrams and structure diagrams.

There are many ways of drawing IPO diagrams or charts. An IPO diagram can be a graphical diagram used to show the flow of control from an input through a process

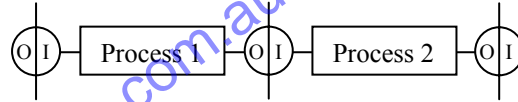


Fig 3.14

Graphical interpretation of an IPO diagram.

transforming the input into an output (see Fig 3.14). The resulting output will become the input for a further process or processes. In other words, the diagram shows a number of processes with the focus on the links between their outputs and inputs. In this course IPO diagrams use a modified table format, with a section for inputs into a particular process, a section for an explanation of the processing and a section to list the outputs from the processing. Fig 3.15 shows three commonly used methods of drawing IPO diagrams. Each of the methods is conveying virtually identical information: it is just the format that is different.

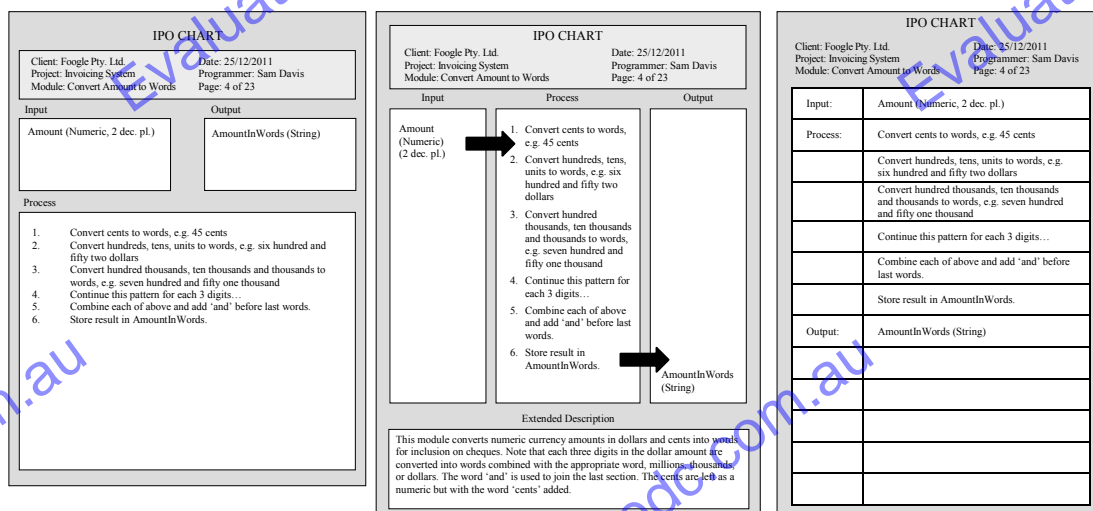


Fig 3.15

Three commonly used methods of drawing IPO diagrams or charts.

Different software development companies will use different formats for their IPO diagrams. IPO diagrams form the specifications that are handed to programmers. The programmer uses the information to write algorithms and finally the source code for the module. The process section of the IPO diagram need not explain all the logic of the solution but must include the precise nature of the tasks the subroutine or module must accomplish. In the above example, a precise explanation is given of how to convert amounts to words, however no indication is given in regard to decisions and repetitions that may be required in the final algorithm.

The recommended method for drawing IPO diagrams in this course, is to use three columns: one each for input, process and output. The inputs and the outputs are aligned next to the process that accepts the input or the process that results in the output.

Fig 3.16 shows an example of this method using the same information as was used in the previous Fig 3.15 examples. On the IPO diagram at right, the input, Amount is shown next to the first process that requires that input. The output, AmountInWords, is shown next to the process that ultimately calculates the value of that variable. Further information in regard to the source and destination of the inputs and outputs can also be included in the input and output columns.

IPO CHART		
Client: Foogle Pty. Ltd. Project: Invoicing System Subroutine: Convert Amount to Words		Date: 25/12/2011 Programmer: Sam Davis Page: 4 of 23
Input	Process	Output
Amount	Convert cents to words, e.g. 45 cents	
	Convert hundreds, tens, units to words, e.g. six hundred and fifty two dollars	
	Convert hundred thousands, ten thousands and thousands to words.	
	Continue this pattern for each 3 digits...	
	Combine each of above and add 'and' before last words.	
	Store result in AmountInWords.	AmountInWords

Fig 3.16

Recommended method for drawing IPO diagrams.



Consider the following:

A business is invoiced for supplies from its vendors. The structure chart below models the tasks involved in processing and paying these invoices. A series of IPO diagrams are now being prepared.

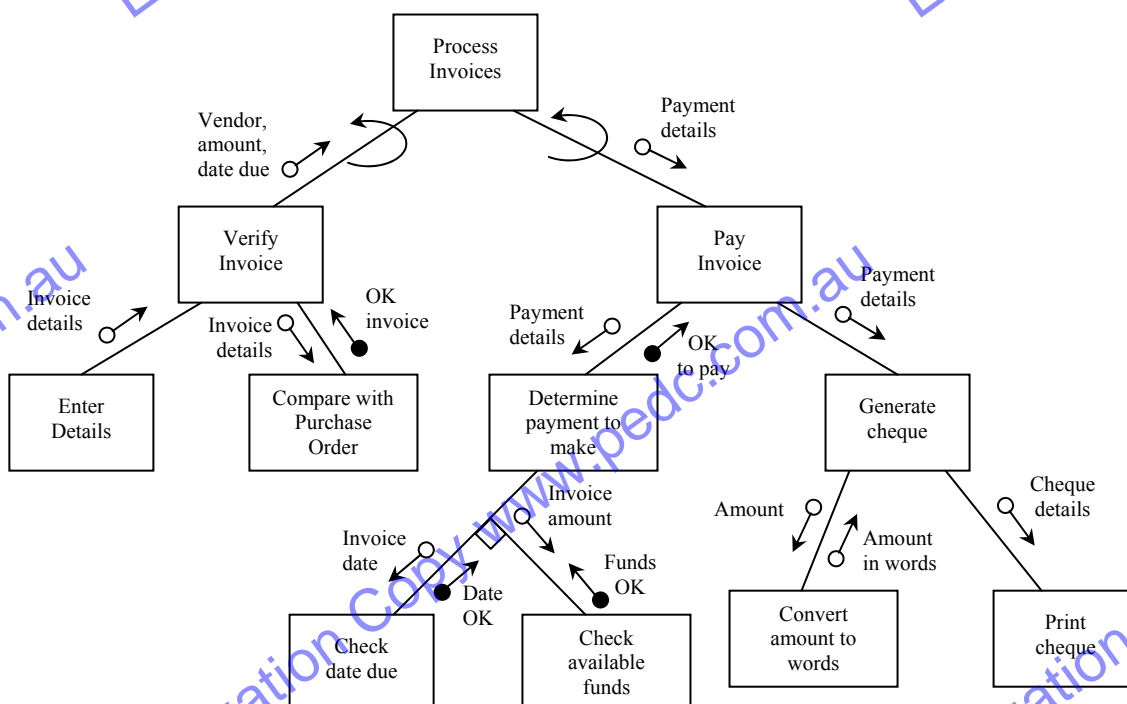


Fig 3.17

Structure chart for processing invoices.



### GROUP TASK Discussion

Describe the processing modelled on the above structure chart (Fig 3.17). Identify each input into the module and the most likely data source.

An IPO chart has been developed for the Compare with Purchase Order task from the structure chart in *Fig 3.17* above.

IPO DIAGRAM		
Client: Foogle Pty. Ltd. Project: Invoicing System Subroutine: Compare with Purchase Order		Date: 29/12/2011 Programmer: Janine Fendall Page: 9 of 23
Input	Process	Output
	Set OKInvoice to True	
PO Number, Invoice Item Name, Cost, Quantity (from VerifyInvoice)	Read PO details from PO file using PO number.	
Purchase order Item Name, Cost, Quantity (read from PO file)	Check if Item Names match and Item Costs match and Item Quantities match.	If False then OKInvoice is set to False
	Add Invoice Item Cost to Invoice Total	
	Repeat above 3 processes for each item or while OKInvoice is True.	
Purchase Order Total (read from PO file)	Check if Invoice Total matches PO Total	If False then OKInvoice is set to False

*Fig 3.18*

*IPO Chart for the 'Compare with Purchase Order' task in Fig 3.17.*



#### **GROUP TASK Describe**

Describe in words what happens in the Compare with Purchase Order task modelled in the above IPO diagram.



#### **GROUP TASK Discussion**

The IPO diagrams in both *Fig 3.16* and *3.18* describe tasks (subroutines) from the structure chart in *Fig 3.17*. Create IPO diagrams for at least two of the other subroutines on the structure chart in *Fig 3.17*.



#### **GROUP TASK Discussion**

Discuss the link between structure diagrams and IPO charts. Use the above IPO chart as an example during your discussion.

### **DATA DICTIONARY**

When developing a software solution to a problem, it is vital that all variable names or identifiers are carefully documented. This ensures that all members of the development team are aware of identifiers that have been used, their data type, storage size, display size and format, scope of usage and various other details. A data dictionary is the repository where this information is held. Data dictionary functions are available with most software development CASE tools. If the data dictionary is an integral part of the development environment then confusion due to duplicate identifiers can be eliminated. For smaller development projects, the data dictionary is often maintained on paper with the assistance of a database or word processor.



#### **Scope**

The extent to which a variable is available for use. Global variables are available for use by all subroutines in a program. Local variables are available only within the individual subroutine.



A data dictionary should include a thorough description of every variable and data structure used by the program and each field in each database and file used or accessed by the program. A separate data dictionary is commonly created for each module, database and file used by the program. Often a data dictionary will be a table containing columns for:

- Identifier name or data item.  
This is the name used to access the variable within the source code.
- Data type, such as integer, string floating point or Boolean. For data structures the general nature of the structure is used, for example Array(string) specifies an array which contains a sequence of strings. When specifying records use a heading row followed by a separate row for each field within the record.
- Format. Refers to the display format where this is applicable, for example XXNNN for a string indicates a total of 5 characters where the first two characters are alphanumerics and the last three characters must be digits, such as BF345.
- Number of bytes required for storage. For strings each character will use a single byte if ASCII is used and two bytes if Unicode is used. For example, a string containing 5 characters will require either 5 or 10 bytes of storage.
- Size for display. The number of characters required to display the contents of the variable on a screen or on paper. This value will include decimal points, dollar signs, percentage signs or any appropriate characters added to the raw data when it is displayed. For instance, dollar and cents values less than 10,000 will require up to 8 characters to display as \$9999.99 contains 8 characters. However, if the data type is single precision floating point then just 4 bytes are required for storage.
- Description or purpose. A brief outline of the logical role the variable plays within the subroutine. May include other relevant details such as whether it is a parameter, input by the user or stored in a file.
- Example. A typical example of an actual data item usually in its display format. This is particularly important for dates, currency and other formats that can be shown in a variety of different ways.
- Validation. Refers to the range of allowable values. For instance, a person's age in years must be positive and less than say 130. Dates are often restricted based on the current date, that is often they must be less than or equal to today's date or for delivery dates they must be after today's date.
- Scope of the variable. Is the variable local to its subroutine or is it available more widely?

The exact columns used will be determined by the nature of the data and how it is used in the source code. A variable that is only used internally (never displayed to users) has no need for a format and size for display, hence these can be left blank. The example data dictionary above in Fig 3.19 displays a sub-form (with the tabs general and lookup) which allows detailed entry of information specific to the particular data type of the current field. This data dictionary describes a table in a relational database, therefore it includes an icon of a key to indicate the primary key field in the table.

Field Name	Data Type	Description
Ans1	Text	Incorrect answer 1
Ans2	Text	Incorrect answer 2
Ans3	Text	Incorrect answer 3
Correct	Text	The Correct answers text
QuesText	Memo	The Question
Explanation	Memo	Explains why an answer is correct, also why any possible answers are incorrect
QuesID	AutoNumber	Links a Question to its Topics
Stimulus	Text	The filename for the stimulus .avi for video, .bmp for picture (must be one of these 2)

**Field Properties**

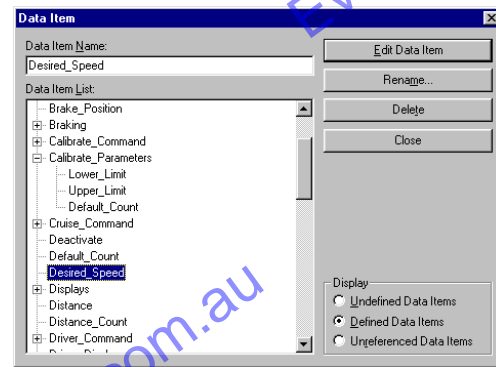
**General** | **Lookup**

Field Size: Long Integer  
 New Values: Increment  
 Format:  
 Caption:  
 Indexed: Yes (No Duplicates)

A field name can be up to 64 characters long, including spaces.  
 Press F1 for help on field names.

Fig 3.19  
A data dictionary from MS-Access

The screen shown in *Fig 3.20* is part of a data flow diagram CASE tool. This screen allows the modification of the attributes of each data item used in the data flow diagram. As data flows are added to the diagrams the data items are automatically added to the data dictionary. Clicking on the 'Edit Data Item' button opens a screen allowing entry of details describing specific data items, such as data type, range, and a description of the data item's purpose.



*Fig 3.20*

*Data dictionary screen from the Axiomsys CASE tool.*



Consider the following:

The data dictionary below describes all the identifiers used in the Convert Amount to Words subroutine from the Invoicing System structure diagram shown in *Fig 3.17*.

Identifier (Data Item)	Data Type	Number of storage bytes	Description	Example
AmountInWords	String	255	Returns the currency amount in words.	Six thousand, two hundred and fifty dollars and 45 cents.
Amount	Floating point	4	Input parameter.	6250.45
TempDigit	Integer	2	Stores each digit as it is extracted from Amount.	6
DigitWord(19)	Array(string)	10 * 20	The word associated with each digit.	DigitWord(5)="five".
TenMultipleWord(9)	Array(string)	10 * 10	Word for each multiple of ten.	TenMultipleWord(3)="thirty"
Ten3Word(4)	Array(string)	10 * 5	Word for each 3 <sup>rd</sup> power of ten.	Ten3Word(2)="million"
PlaceCounter	Integer	2	Counter incremented for each digit in Amount.	1
TempResult	String	255	Stores the amount in words during processing.	and 45 cents.

*Fig 3.21*

*Data dictionary for the 'Convert Amount to Words' subroutine.*

The programmer created this data dictionary whilst developing the source code. For this particular subroutine format and size for display have not been included as the only output is the final amount in words, which is a string up to 255 characters long. Data dictionaries are stored, along with other documentation, to assist in future maintenance and upgrading of the software product.



#### GROUP TASK Discussion

How could a data dictionary, such as the one above, be of assistance to future maintenance personnel?



#### GROUP TASK Discussion

Explain the relationship between structure charts, IPO charts and data dictionaries. Use the above data dictionary as an example to illustrate your answer.

## SCREEN DESIGNS AND CONCEPT PROTOTYPES

Screen designs provide the interface between the user and the software. Considerations in regard to placement of items, consistency of design, order, grouping and nature of screen elements, use of colour, graphics and fonts, and language used all contribute to the effectiveness of a screen's design. Concept prototypes (see chapter 2) are used to implement screen designs so potential clients can evaluate them. Limited or concept prototypes are designed to gauge the effectiveness or ineffectiveness of screen designs. A concept prototype can be thought of as a working screen design.

Companies that produce operating systems publish standards for screen design to ensure consistency of design between applications. The screen in Fig 3.22 utilises a number of features complying with Microsoft's standards for Windows screen design. Radio buttons are round, check boxes are square and contain a tick when selected, frames are used to link related items, the screen's title appears in the bar at the top of the screen, a cross is used to close the screen, etc. Adhering to these screen design standards results in reduced learning times for users. Today, many screen elements are similar in function and design across the range of GUI operating systems, however their colour, use of fonts, and format can make them look very different. Compare the above Windows screen with the Apple screen in Fig 3.23 and the Linux screen in Fig 3.24. Many of the elements are similar, however the overall effect is different.

Many integrated CASE tools include areas where preferences in regard to screen design are stored. When source code is generated, the screens will adhere to the preferences entered into the CASE tool. This encourages consistency of design across all screens in a particular application.

Many large-scale business applications use a text-based user interface. Text-based screens are often designed using a paper grid containing a cell for each character that makes up the screen.

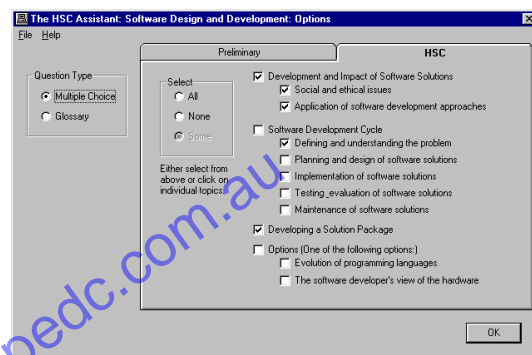


Fig 3.22

Option screen from 'The HSC Assistant: SDD' multimedia CD-ROM for Windows.

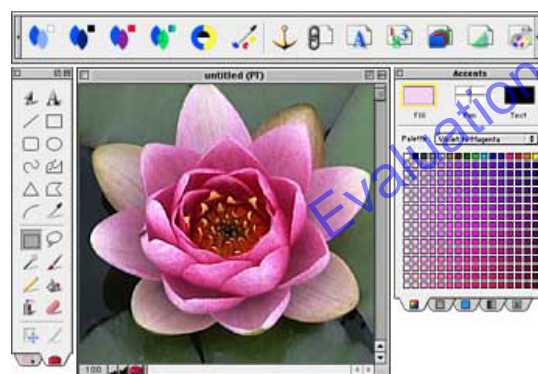


Fig 3.23

Screen shot from the paint module in Appleworks 6 for the Macintosh.

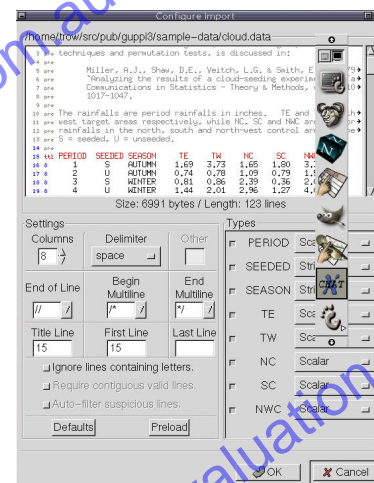


Fig 3.24

Screen from an application for the Linux operating system.



### GROUP TASK Activity

Use the internet to find information about standards for screen design for the operating system you most commonly use. Compile a list of your findings.

## STORYBOARDS

A storyboard is used to describe the screens used in a project together with how each screen interacts with other screens. Effectively, a storyboard for a software product is a collection of screen designs together with a diagram describing the links between the screens. Storyboards are commonly used in movie production where a series of cartoon-like screens are produced. For movie production, the sequence in which screens are encountered is always linear, as the movie needs to be watched from start to finish to make sense. This is not the case with most software. The interactive nature of software means that storyboards must carefully document the possible flow of data between screens.

Storyboards for software products usually include a series of screen designs together with a diagram illustrating the possible navigation paths between screens. This diagram will show each screen as a rectangle with arrows leading to and from other screens. If the screens are small and simple and the screen designs fit on a single page then the links can be included without the need for a separate diagram. It is important to develop a logical navigation system between screens so users do not become lost in a maze of screens. Navigation buttons are normally included on every screen in a consistent position to assist users.

There are many ways of organising the navigation between screens when developing a new application. The most common method is to utilise a hierarchical structure where a main menu directs the user to the major screens of the application. Each of these screens will have further links to lower-level screens in the hierarchy.



Consider the following:

The diagram below shows a hierarchical screen structure for a company's web site. Although the structure is basically hierarchical, a link is provided on each screen to return the user to the home page.

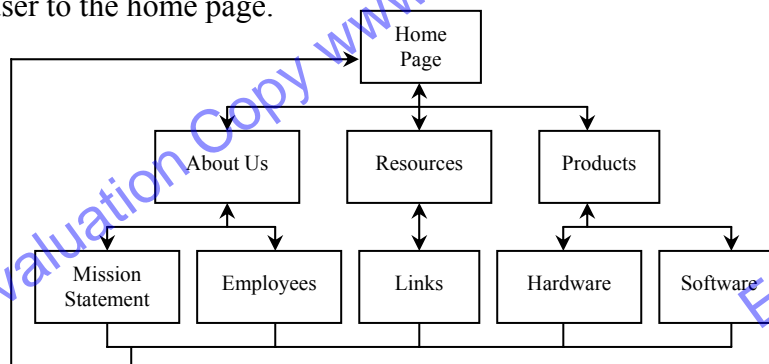


Fig 3.26

*Hierarchical navigation between screens is common for web sites.*



Fig 3.25

*A storyboard for a movie is always linear.*



### GROUP TASK Discussion

Is there any way of viewing the Hardware page without first viewing the Products page? Suggest reasons why the developer may have done this.





Consider the following:

The storyboard below describes the navigation between the screens of a school's reports package, together with three screen designs that formed part of the original concept prototype for the project.

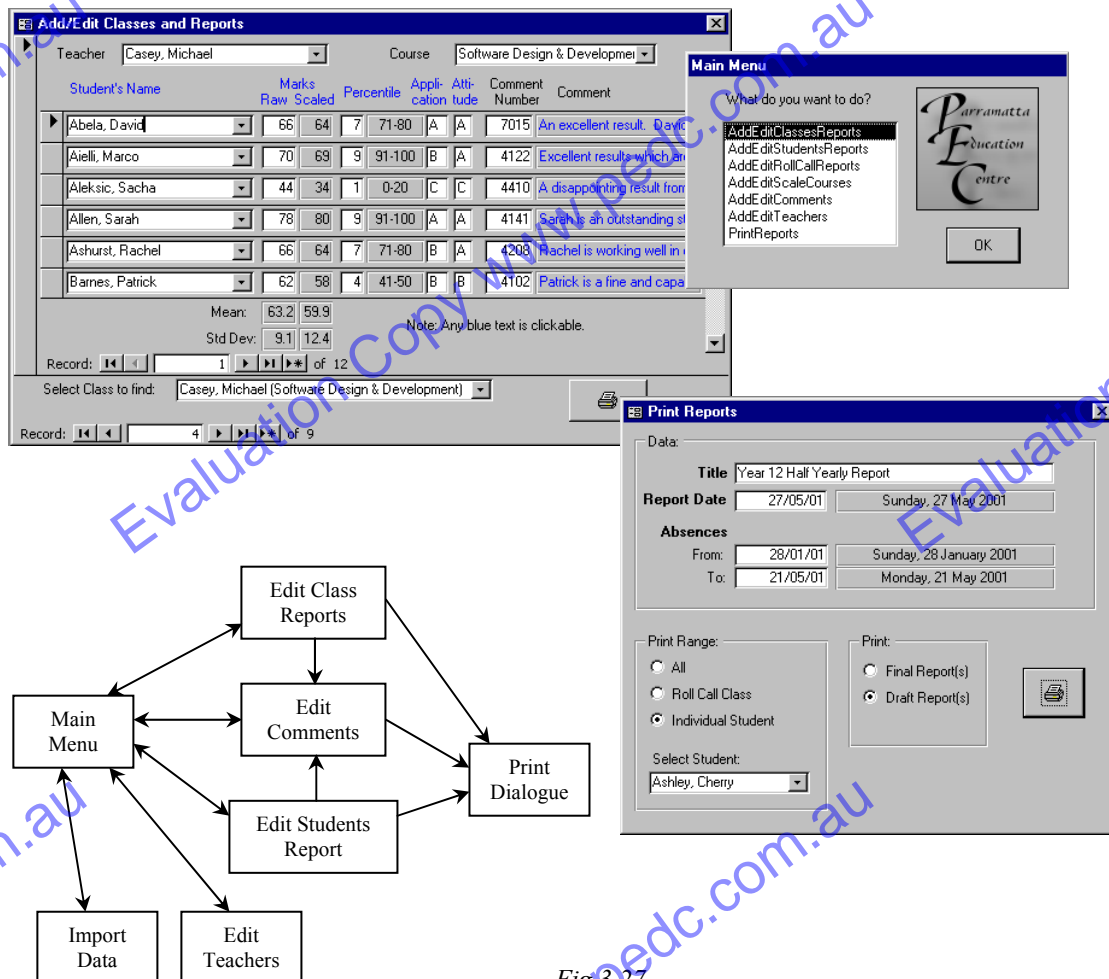


Fig 3.27

Storyboard for a school's computerised report system.



### GROUP TASK Discussion

List and describe the screen elements shown on the above three screen designs. Do you have any suggestions for improving their design?



### GROUP TASK Activity

Initial screen designs are often done on paper. Create a new design for the Main Menu using your knowledge of sound screen design.





HSC style question:

A graphics conversion program is being developed to convert images between the BMP, JPG and GIF formats. The program will convert all graphic files of a particular format that are found within a specified directory into one of the other graphic formats. For example, if C:\Images is the chosen directory and the user wishes to convert from BMP to JPG then all BMP images in the C:\Images directory are converted into JPG images.

- Create an IPO diagram to describe the overall operation of this graphics conversion program.
- Create a screen design for the graphics conversion program.

#### Suggested solutions

(a)

Input	Process	Output
Directory name File type old File type new	<ul style="list-style-type: none"> <li>Get directory name</li> <li>Get input (old) and output (new) file types (JPG, BMP or GIF)</li> <li>Determine and store file names found in the directory that are of the relevant input file type</li> <li>Convert each image in turn into the desired output file type, saving using the same name but with the new file extension</li> </ul>	Converted graphic files in new file type

(b)

**SET 3C**

1. A model of a system can best be described as:
  - (A) a system flowchart representing logic and movement of the data.
  - (B) a representation of a system showing structure and functionality.
  - (C) a data dictionary describing the data types used by a system.
  - (D) a structure chart showing the navigation between screens.
2. IPO diagrams explain:
  - (A) the data types used.
  - (B) the screen designs.
  - (C) how inputs are transformed into outputs.
  - (D) the flow of data.
3. Data flow diagrams explain:
  - (A) the data path through the system.
  - (B) the timing of the data path through the system.
  - (C) the manual processes component of the system.
  - (D) the sequence of processing.
4. Structure charts are used to:
  - (A) model the hierarchy of the processes within the system.
  - (B) show the algorithms that the system will use.
  - (C) determine user requirements.
  - (D) all of the above.
5. A data dictionary should include:
  - (A) identifiers.
  - (B) data types.
  - (C) number of bytes of storage.
  - (D) all of the above.
6. The interface between the end-user and the software is largely determined by:
  - (A) the operating system.
  - (B) the screen designs.
  - (C) the clients evaluating the system.
  - (D) the source code.
7. Concept prototypes are useful because they:
  - (A) result in reduced learning times for users.
  - (B) use all the same elements.
  - (C) assist in gauging the effectiveness of the screen designs.
  - (D) are all text based screens.
8. A collection of screen designs together with a diagram describing the links between screens is referred to as a(n):
  - (A) storyboard.
  - (B) IPO diagram.
  - (C) structure chart.
  - (D) hierarchical diagram.
9. Data dictionaries and other documentation should be stored because:
  - (A) it assists in the maintenance of the software.
  - (B) it assists in the upgrading of the software.
  - (C) it assists in understanding the processes used by the software.
  - (D) all of the above.
10. Which representation is generally used at a higher level giving an overall view of the system?
  - (A) system flowcharts.
  - (B) storyboards.
  - (C) data flow diagrams.
  - (D) IPO diagrams.
11. Identify each of the symbols used in system flowcharts. Give an example of a situation where each symbol may be used.
12. Draw a system flowchart to model a system of your choice. Use at least 6 of the symbols identified in question 11.
13. Structure charts have many different names. Compile a list of these and explain in your own words how each of these terms describes the nature of a structure chart.
14. Traditional board or card games have very definite rules. For this reason they are good examples to use when learning to apply the logic and structure that is involved in software development. Using this concept, draw a structure diagram for a board or card game with which you are familiar.
15. Data flow diagrams are particularly useful for describing systems that are data oriented. An EFTPOS (Electronic Funds Transfer Point Of Sale) terminal is used to process withdrawals directly from customers' accounts. These terminals are now commonplace in most retail stores. Design a data flow diagram to model the operations of a typical EFTPOS terminal.

## QUALITY ASSURANCE

Quality assurance (QA) is a set of processes that occur throughout the development and manufacturing of any type of product. These QA processes ensure standards and requirements are met and continue to be met such that the final products are of a high quality. For most manufactured products quality assurance is largely about monitoring the manufacturing process to ensure parts continue to meet tolerances and standards over time. Random samples can be removed from the production line for thorough testing. Software is somewhat different as it is the design and development process that determines quality. Software quality is a measure of the extent to which desirable features, including functional requirements, are included such that the software reliably and efficiently continues to meet the needs of users. Software Quality Assurance (SQA) aims to provide users with confidence in the integrity of software. SQA ensures that software will function reliably as intended and is free of errors and other vulnerabilities. Vulnerabilities include both intentional and unintentional faults, such as security flaws, inclusion of untested code and of course a variety of other bugs which may result in loss of data or reduced reliability.



### Software Quality

The extent to which desirable features are incorporated in to software so it can reliably and efficiently continue to meet the needs of users.

How can software quality be judged throughout the design and development of software? A set or list of measurable criteria is required. The criteria will be different and have different focus depending on the nature and size of the software project. Software which will operate critical systems, such as the electricity grid or an aircraft's flight controls, requires a particular focus on reliability and accuracy whereas a graphics editor may focus more on usability. The following list includes a range of areas that should be assessed or at least considered when developing quality assurance criteria.

- **Efficiency.** The software should perform its functions by making the best use of the computer's resources. Judging efficiency includes assurance that good programming techniques are used such as the use of algorithms which optimise processing time, well structured code and thorough testing. In addition the selection of an appropriate language for the problem and the use of operating system and CPU features such as multi-tasking, distributed and parallel processing. Efficiency should be judged using a range of possible system configurations, such as different versions of operating systems, amount of RAM, video RAM and various CPUs. Code that accesses networks, particularly the internet will often greatly affect efficiency and should be thoroughly examined and optimised.
- **Integrity.** The correctness of data within the system. Integrity is improved when data validation is thorough and the ability to access the system illegally or inappropriately is tightly controlled. Data integrity will often reduce over time as time dependant data becomes out of date. Systems to check such data should be included, for instance checks that phone numbers, addresses and email addresses are current. During development integrity can be judged based on the security mechanisms for accessing the system and also on data validation as data is entered into the system.

- **Reliability.** The ability of the system to continue to perform its functions over time. Other reliability factors that should be considered include the time taken to recover from a failure and the ability of the system to be updated to reflect changes in requirements or changes in the environment in which the software operates. For instance, a new version of an operating system may cause issues for the software or a hard disk may fail. Such problems should be able to be fixed in a timely manner if the software is to perform reliably.
- **Usability.** The ability of software to be learnt and used by users. The design of the user interface is critical when judging usability. Factors such as the consistency of the design and the correct use of screen elements and colour. Other considerations include recovery from errors, warnings or undo features when data changes of significance are made and also features for advanced users such as shortcut keys and the ability to record macros which can be replayed to repeat commonly performed tasks. For instance a macro might generate a weekly report, print it to a pdf file and email it to the appropriate recipients.
- **Accuracy.** The software performs its functions correctly and according to its specifications. Accuracy can only be assured when the code has been thoroughly tested. Algorithms can be assessed to ensure correctness for all possible paths and sets of inputs. The code should be well commented so that accuracy can be ensured during future upgrades.
- **Maintainability.** Most successful software will undergo changes throughout its lifetime. Maintainability is a measure of the ease with which these changes can be made. Source code should be well documented, modular in design and test data or test routines should be retained so thorough testing can be performed after any code changes have been made. There are three main reasons code is routinely altered – to correct errors, to implement new functionality or to improve performance. Code should always be written with future maintenance in mind. Often the logic within code appears obvious when first written but is difficult to comprehend later when modifications are required.
- **Testability.** The ability to test all aspects of the software is critical to each of the other quality assurance factors. Testing occurs at various times and levels as software development progresses. Individual subroutines and then modules should be thoroughly tested, the system should be tested to ensure it performs according to requirements and finally the client performs acceptance testing to ensure requirements have been met with a live system.
- **Re-usability.** The ability to reuse code in other related systems. Once a fully tested routine or module has been produced it can be reused as part of future software projects. Often minimal effort is required to write a more general case of a required subroutine which is much more suited to reuse. Most programmers maintain a library of subroutines and modules that perform commonly used functions. There is no need to reinvent the wheel and furthermore the accuracy of future software is improved when thoroughly tested routines are reused. There are many commercial and free libraries of code available for most languages. Portability is another significant aspect of reusability. Many languages are compiled so the same code can be used for a variety of operating systems.

**GROUP TASK Discussion**

Does each of the above areas improve software quality for software developers or for their clients? Discuss.



Quality and often people's perception of quality has an enormous impact on the success of all products including software. It is critical that management processes are put in place to ensure quality criteria is met throughout the software development cycle. Quality assurance is an ongoing process throughout design and development which ensures the identified quality criteria are met.



Consider the following new software products:

- Software driver for a new web camera.
- Record system for breeders of exotic animals.
- Software for a new hand held GPS.
- New version of Microsoft Word.



#### **GROUP TASK Discussion**

For each of the above software products identify and justify quality assurance criteria likely to be of particular importance.



#### **GROUP TASK Discussion**

For each of the quality criteria identified above, discuss suitable management processes that could be used to ensure the quality criteria is met during the software development cycle.



HSC style question:

- (a) Define the term “Quality Assurance” as it applies to software development.
- (b) Maintainability is one aspect of quality software. Explain how maintainability can be assured during the development of software.

#### **Suggested solutions**

- (a) Quality assurance is an ongoing process occurring throughout the software development cycle to ensure software is of high quality. Quality criteria are set and are then monitored during development. The quality of the product is judged according to the extent to which the quality criteria have been met.
- (b) Maintainability is the ease by which a software product can be altered to correct errors, improve performance or add new functions. To assure maintainability the source code should be well documented with comments and the use of intrinsic or meaningful variable names. In addition system models such as structure charts and DFDs should be kept and updated. Also a strict system of testing should be implemented and these tests kept so they can be rerun when any changes are made to the code. All these aspects of maintainability should be specified as requirements to all developers. They should be checked and monitored on an ongoing basis during the development process so that maintainability is assured.



**CHAPTER 3 REVIEW**

1. The first stage of the software development cycle is:
  - (A) choosing a method that is appropriate to the construction of the solution.
  - (B) defining and understanding the problem.
  - (C) selecting the most suitable personnel for the project.
  - (D) developing the design specifications for the system.
2. Items that are not part of a system are said to be:
  - (A) part of the environment.
  - (B) part of the boundaries of the system.
  - (C) part of the interface.
  - (D) part of the limits of the system.
3. The difference between boundaries and interfaces is:
  - (A) boundaries are factors with which the system must comply; interfaces are the limits of the system.
  - (B) boundaries are what you are aiming to achieve; interfaces are what is keeping you from achieving them.
  - (C) boundaries are the limits of the system; interfaces are how the system communicates across a boundary with its environment.
  - (D) boundaries are within the system; interfaces are within the environment.
4. A software is difficult to maintain and has usability issues. Which of the following would have helped minimise these issues during the initial development processes?
  - (A) thorough system testing.
  - (B) quality assurance.
  - (C) careful problem definition.
  - (D) needs analysis.
5. Social and ethical issues requiring consideration as part of a software development project include:
  - (A) legal issues.
  - (B) effects on employment levels.
  - (C) the changing nature of work.
  - (D) all of the above.
6. Joe and Mary meet after work for dinner and discuss their fears regarding the new system that is being implemented at their place of work. Mary believes that she will be made redundant when the new system is operational. Her fears are commonplace in the given scenario and fall under the category of:
  - (A) social and ethical issues.
  - (B) legal issues.
  - (C) technical issues.
  - (D) operational issues.
7. The diagram that describes the top-down design and sequence of processing is called a(n):
  - (A) IPO diagram.
  - (B) storyboard.
  - (C) structure chart.
  - (D) data flow diagram.
8. Processes can best be described as:
  - (A) the destination of data.
  - (B) a model of a system.
  - (C) the interface between the user and the system.
  - (D) the actions that take place on the data within the system to transform inputs into outputs.
9. A need is often expressed in emotive, non-specific terms. Which need listed is a clear example of this?
  - (A) Our image in the marketplace needs improvement.
  - (B) Our profit margin needs to increase by 15% over 18 months.
  - (C) The new system needs to be implemented by July 15.
  - (D) Our call out service fee should increase by \$5.00.
10. A requirement should be able to be readily tested to determine whether it has been achieved. Which requirement cannot be readily tested?
  - (A) 95% of deliveries are shipped within 3 working days.
  - (B) Clients are contacted at least once every 6 months.
  - (C) Customer perceptions of our company are vastly improved.
  - (D) Data entry productivity is increased by 15%.

11. A software company is examining the possibility of developing a product that will enable high-speed internet access in hotel rooms. The software product will provide high-speed networking capabilities as well as monitoring each guest's online time. The online time will be sent to the hotel's main computer where charges will be debited to the guest's account. At the present time, hardware is available to complete this task. A cable network already exists into each room to deliver cable television. The new product would involve a dedicated internet connection being installed in each of the hotel's rooms.

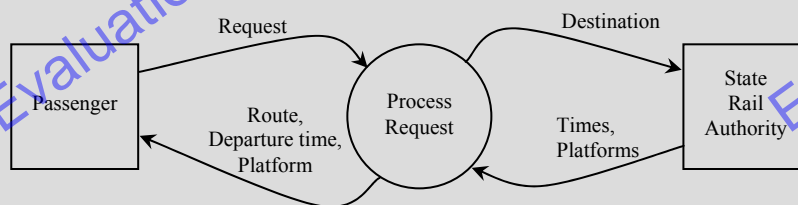


Describe at least four aspects of this proposal that should be closely examined as part of defining and understanding the problem.

12. Design specifications are developed to ensure the development process adheres to certain standards. These specifications determine a framework in which members of the development team will operate.

List and describe at least 6 items that should be considered as part of the design specifications for a software development project.

13. The SRA (State Rail Authority) is looking to develop a computerised timetable system that will allow passengers to enter their destination and obtain details of the route, time of departure and platform of the next available train. A context diagram is provided below. Use this context diagram as an aid in the creation of a level one data flow diagram for this scenario.



14. As part of the development team for a new software product, you have been given the following IPO diagram. Use this IPO diagram to develop an algorithm that performs this task.

Client: Build Soft Pty. Ltd.  
Project: Encryption System  
Module: FindTwoPrimes

IPO DIAGRAM

Date: 18/10/2001  
Programmer: Youcan Doit  
Page: 3 of 9

Input	Process	Output
Lowest as an integer, the two primes must be greater than this value	Loop around for each integer greater than Lowest.	
	For each integer, loop around from 1 to the square root of the integer.	
	Check if any of these numbers divide evenly into the number being checked.	
	If no numbers go in evenly then the integer is a prime so keep it.	
	Do the same to find a second prime.	Prime1, Prime2

Notes: A prime number only has factors of 1 and itself. E.g. 1 and 7 are the only integers that go into 7 so 7 is a prime number. 1, 2, 3 and 6 go into 6 so 6 is not a prime number.

15. In question 13, we created a data flow diagram for a computerised timetable system for the State Rail Authority. Assuming that this system will be implemented using a touch screen, design a series of screens for this product. Remember, people with various forms of disability will use this system. Justify your choice of screen design elements.