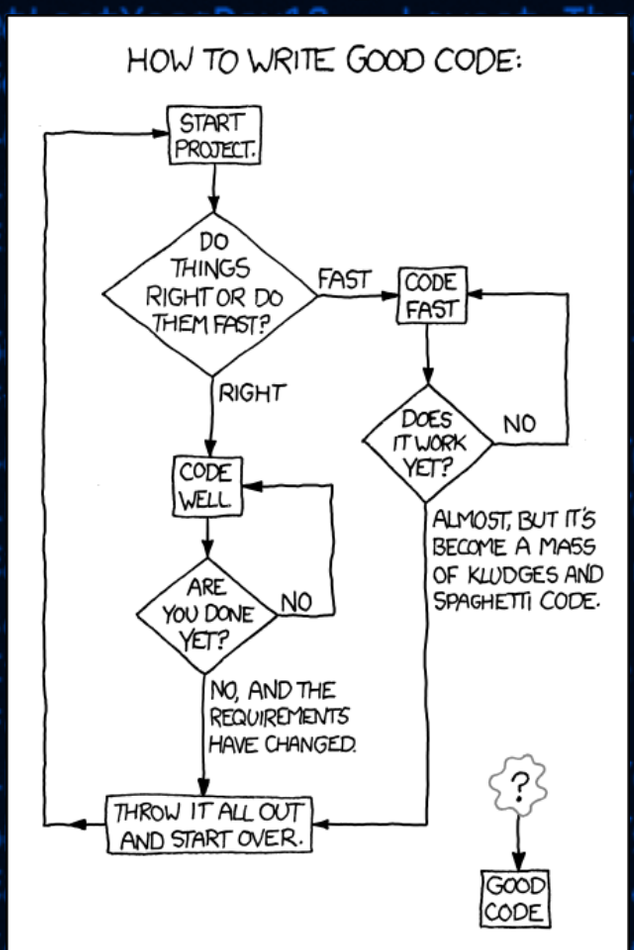
A text book for the VCE Applied Computing Study Design 2020 - 2023
&
Microsoft Visual Basic

2nd EDITION

2021 Edition

Software Development
Units 3 & 4

Vic Farrell



ISBN 978-0-6486708-1-0

Table of Contents

The 2021 edition of this textbook was written and published after the VCAA released all relevant support documentation. This version contains updates and extra support material. For any further assistance please contact Vic Farrell via the website: vicfarrell.com.au.

THANKS

To Adrian Janson.
To Elisa Brock for being an amazing proofreader right next door.

Self published using free templates from http://copiesandink.com/
Icons made by Freepik from flaticon.com
Many thanks to Randall Munroe for the use of his excellent comics www.xkcd.com

This publication is designed to meet the teaching and learning needs of teachers and students of the Study Design Applied Computing: Units 3 & 4 Software Development 2020 - 2023 provided by the Victorian Curriculum and Assessment Authority.

Microsoft Visual Studio screen grabs used with permission 2020.

Thanks to Daniel Viglietti for his Year 12 SAC code for the cover art.

ISBN: 978-0-6486708-3-4 (e-Book)
ISBN: 978-0-6486708-4-1 (Paperback)

Publisher: Amazon Kindle
Victoria, Australia
2020

For more resources: www.vicfarrell.com.au

Unit 3
Area of Study 1

Programming

In this area of study students examine the features and purposes of different design tools to accurately interpret the requirements and designs for developing working software modules. Students use a programming language and undertake the problem-solving activities of manipulation (coding), validation, testing and documentation in the development stage.
The working modules do not have to be complete solutions and can focus on limited features of the programming language; however, students are expected to fully develop the working modules in accordance with the given designs. This will prepare students for creating a complete solution in Unit 4, Area of Study 1. Validation and testing techniques are applied to ensure modules operate as intended and internal documentation is written to explain the function of the modules. Students justify the use of the selected processing features and algorithms in the development of their working modules.

Key Knowledge

Data and information
- characteristics of data types
- types of data structures, including associative arrays (or dictionaries or hash tables), one-dimensional arrays
- (single data type, integer index) and records (varying data types, field index)

Approaches to problem-solving
- methods for documenting a problem, need or opportunity
- methods for determining solution requirements, constraints and scope
- methods of representing designs, including data dictionaries, mock-ups, object descriptions and pseudocode
- formatting and structural characteristics of files, including delimited (CSV), plain text (TXT) and XML file formats
- a programming language as a method for developing working modules that meet specified needs
- naming conventions for solution elements
- processing features of a programming language, including classes, control structures, functions, instructions and methods
- algorithms for sorting, including selection sort and quick sort
- algorithms for binary and linear searching
- validation techniques, including existence checking, range checking and type checking
- techniques for checking that modules meet design specifications, including trace tables and construction of test data
- purposes and characteristics of internal documentation, including meaningful comments and syntax.

Key skills

- interpret solution requirements and designs to develop working modules
- use a range of data types and data structures
- use and justify appropriate processing features of a programming language to develop working modules
- develop and apply suitable validation, testing and debugging techniques using appropriate test data
- document the functioning of modules and the use of processing features through internal documentation.

Data and Information

Information Systems are any organisational system that sorts, finds, retrieves and displays information. Information is processed data. An example might be your mobile phone as an information system for phone contacts. You input the data - given names, family names and mobile numbers and the system places the data into the memory which can then be sorted, searched and used. Data is organised, sorted and formatted into information such as spreadsheets, magazine layouts, websites, or reports. In digital systems we mainly focus on text, numbers and images in our databases, software development and web development. Data comes in five digital types: text, numbers, images, sound and video. All of these data types are composed of fundamentally the same thing: Numbers! Digitising any data is the process of converting it into numbers.

Binary

Digital machines are built on circuitry that allows for the flow of electricity. A digital machine can detect if a circuit is ON (where there is electricity flowing through the circuit) or OFF (no electricity). This is the only way these machines can "understand" the data that is read in. So we devised a simple code to bridge the divide between humans and machines. If the circuit is OFF we will call it a zero. Computer scientists use Ø to distinguish it from the letter O. If the circuit is ON we will call it 1. This is how the connection between machines and humanity is made.

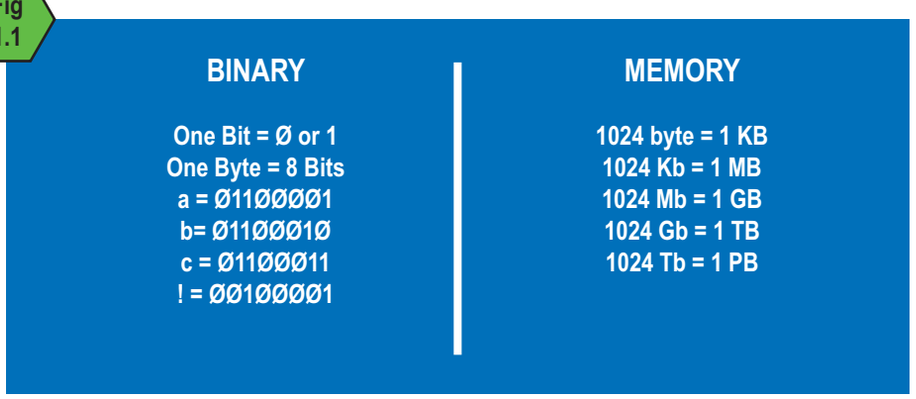
Humans have a number system based on 10 symbols (Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9) which is called the decimal system. All values can be created with these ten symbols. It is suggested the reason we chose ten symbols was due to having ten fingers. Digital machines do not have the capacity to understand anything beyond ON and OFF, so we have to construct a number system around the two "fingers" they have: Ø and 1. Since there are only two symbols, we call this number system Binary (bi means 2).

Computers use 1s and Øs to represent all the values and all the other symbols we use in text. We call each Ø and each 1 a bit and we have to combine them to create unique identifiers for each symbol. We group eight bits together to represent all the ASCII symbols (all the letters, numbers and symbols input by a keyboard). This group of eight bits is called a byte.

Memory

Bytes are too small to measure memory. We usually store hundreds of characters in a basic text file. Binary is based on the number 2. We set a kilobyte to 2^{10} (1024) bytes. In fig 1.1 you can see the relationship between a bit and a byte and how each character on the keyboard is represented as a byte. Common units of storage size are based on 2.

Fig 1.1



Data Types

It is important to be aware of common data types used in software development and their size and range. The key data types you need to know are:

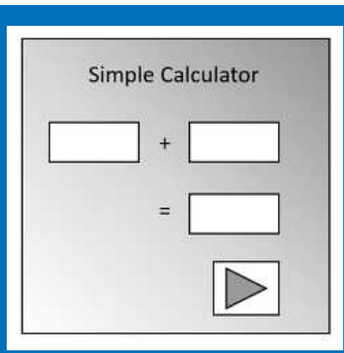
- Boolean - true or false.
- Character - a single letter, number or symbol
- Floating Point - decimal numbers
- Integer - whole numbers
- String - text of any symbols

Variables

Variables are temporary data holding-spaces used in software. These are used to read in data, process it and display the output. Here is a basic example: A simple calculator that reads in two numbers, adds them and displays the answer. A variable holds the value of each of the two numbers typed in by the user and then another variable would hold the answer. Below in fig. 1.2 is an algorithm (a plan for a solution) for the calculator using variables.

Fig 1.2

```
START
    Number1 ← Integer
    Number2 ← Integer
    Answer ← Integer
    Read in Number1
    Read in Number2
    Answer ← Number1 + Number2
    Display Answer
END
```



Records

Records are made up of fields. The fields in the example below are: ID Number, Given name, Family Name, Mobile Number, Date of Birth and Address. The records can be sorted and searched by fields so that similar records can be identified. For example all records with Date of Birth data lower than 1/1/2002 will return all records of people who are over the age of 18 years. Records are most easily managed with unique identifiers, such as ID Number. This ensures all records are unique and we can easily distinguish the possible multiple Robert Smiths recorded in the file. In figure 1.3 you can see how a tabled record can be recreated in an algorithm.

Fig 1.3

```
START
    CustomerContacts.IDNumber ← 3 (Data Type = Integer)
    CustomerContacts.GivenName ← Lady (Data Type = String)
    CustomerContacts.familyName ← Gaga (Data Type = String)
    CustomerContacts.MobilePhone ← 0488 239 765 (Data Type = String)
    CustomerContacts.DateofBirth ← 28/3/89 (Data Type = Date/Time)
    CustomerContacts.Address ← 18 Hollywood Boulevard (Data Type = String)
END
```

ID Number	Given Name	Family Name	Mobile Number	Date of Birth	Address
1	Robert	Smith	0435 238 100	2/5/78	15 String St
2	Courtney	Love	0487 234 988	6/6/72	1/1 High St

Arrays

An array is a very useful data structure for large amounts of data that needs to be sorted or searched. Arrays can only hold one type of data. Each element has an indexed address and this allows arrays to be easily manipulated. In the example below you can see a series of integers. The top row identifies each element's address from zero to nine. This array has 10 elements but is identified as Array[9]. Remember computers start counting at zero! We can edit and access the data elements out of the array using their location. We will use the example array in figure 1.4

Fig 1.4

ARRAY

0	1	2	3	4	5	6	7	8	9
16	34	12	6	28	59	72	13	43	71

Each item can be identified as:

Array[0] = 16
Array[1] = 34
Array[2] = 12
Array[3] = 6
Array[4] = 28
Array[5] = 59
Array[6] = 72
Array[7] = 13
Array[8] = 43
Array[9] = 71

A One-Dimentional Array illustrated below is a list of product data with associated indexes. Product [4] is an array with 5 string elements in it. Each element has an index number associated with it. The index allows the array to be easily searched and manipulated.

Product[0] = "eggs"
Product[1] = "milk"
Product[2] = "bread"
Product[3] = "cheese"
Product[4] = "tomatoes"

A Multi-Dimentional Array as illustrated below has more than one list. The other name for this type of array is an 2D Array. Below is a set of values and their associated soccer team positions. Team[10,1] data type is String and holds a table of two columns [0 and 1] and eleven rows [0 - 10].

If we wanted to add names to our array we could create a third column calling the array Team[10,2] and adding our names in Team[i, 2] where i is a variable that denotes each row. See figure 1.5

Fig 1.5

SOCCER TEAM ARRAY

Team(0, 0) = " 1 "
Team(0, 1) = " Goal Keeper "
Team(1, 0) = "2 "
Team(1, 1) = " Right Full Back "
Team(2, 0) = "3 "
Team(2, 1) = " Left Full Back "
Team(3, 0) = "4 "
Team(3, 1) = " Centre Half Back "
Team(4, 0) = "5 "
Team(4, 1) = " Centre Half Back "
Team(5, 0) = "6 "
Team(5, 1) = " Defensive Midfielder "
Team(6, 0) = "7 "
Team(6, 1) = " Right Winger "
Team(7, 0) = "8 "
Team(7, 1) = " Central Mid Fielder "
Team(8, 0) = "9 "
Team(8, 1) = " Striker "
Team(9, 0) = "10 "
Team(9, 1) = " Attacking Midfielder "
Team(10, 0) = "11 "
Team(10, 1) = " Left Winger "

So all the data in row one could be:

Team[0,0]=1, Team[0,1]=”Goal Keeper” and Team[0,2]= “Boubacar Barry”.

Dictionaries

A Dictionary is a data structure which has many built-in functions that can add, remove and access the elements using a unique key. Compared to alternatives, a Dictionary is easy to use and effective. It has many functions (like ContainsKey and TryGetValue) that process lookups of tabulated data. They can hold many data types while arrays can only hold one. In the example below the key terms identify the team players positions.

```
Team As New Dictionary(Of String, Integer)
    Team.Add(“Goal Keeper”, 1)
    Team.Add(“Striker”, 2)
    Team.Add(“Full Back”, 3)
    Team.Add(“MidField”, 4)
```

Hash Tables

A Hash Table is a data structure which implements all of the dictionary operations but also allows insertion, search and deletion of elements providing the associated keys for each element. Hash Tables are complex solutions that use a calculation with a prime number to find a unique location using a key, to make it easier to find when you have a large file of unsorted data. In a basic address book, you might have:

Bill, Surpreet, Jane, Nqube, Quentin

The problem with locating these names in an address book alphabetically is that will create unused spaces in the table between Bill and Quentin leaving empty wasted storage space. Also a linear search would still be required under each alphabetical section. There are two types of search methods we examine here: Linear Search and Binary Search. They require many operations to find data in a large file or database. Linear search requires every item from the beginning of the file to be checked. For example if we were looking for Quentin, it would take 5 operations to find his data. What if we had millions of items to search? A hash table can use the data itself to calculate a unique location for each item of data.

Hash Tables provide a unique identifier based on the content of the data. If we use a basic conversion of a=1, b=2, c=3 etc we could convert “Jane” to:

(J)11 + (a)1 + (n)15 + (e)5 = 32

Unfortunately if “Neaj” is added to our address book then his converted number would also be 32. So we use a Hash Function that allocates values depending on the location of each character in the string. We use a prime number which helps us create unique values. In figure 1.6 we will use 7.

Hash(Current Letter) = (Prime Number(7) * Hash from previous value) + value of current letter.

Fig 1.6

Example: JANE

Hash = 0
Hash (J) = (7 * 0) + 10 = 10
Hash (A) = (7 * 10) + 1= 71
Hash (N) = (7 * 71) + 14 = 511
Hash (E) = (7 * 511) + 5 = 3,582
Hash (JANE) = 3,582

Example: NEAJ

Hash = 0
Hash (N = (7 * 0) + 14 = 14
Hash (E) = (7 * 14) + 5= 103
Hash (A) = (7 * 103) + 1 = 722
Hash (J) = (7 * 722) + 10 = 5,064
Hash (NEAJ) = 5,064

This provides locations for each item of data based on the data content. So now if you are searching for JANE the search engine need only conduct one operation to calculate the location and go directly to the data in the file.

A common method is to use remainders to consolidate the data into multiple arrays. To calculate a remainder a value is chosen (usually a prime number) to divide the value of the data. A remainder is then output which limits the number of arrays in the matrix. In the example below, the data we have to store in our hash table is:

23, 56, 47, 29, 92, 55, 11.

So we might use 5 to divide each value to find out the remainder on each. In VB we call this a MOD function.

Mod (23/5) = 3 (4 * 5 = 20 leaving a remainder of 3)
Mod (56/5) = 1
Mod (47/5) = 2
Mod (29/5) = 4
Mod (92/5) = 2
Mod (55/5) = 0
Mod (11/5) = 1

[0]	[1]	[2]	[3]	[4]
55	56	47	23	29
	11	92		

Now we have 5 locations to store our data. We can consolidate the locations where our data could be stored. If we used this mod function on our hash table of names we would use less space in our storage files.

Storage

When developing a solution, it is important to consider what data will be input into the system. The system will only be effective if the data input is valid and correct.

When formatting and storing data it is important to consider the following issues:

- How soon do I need the data back if lost?
- How fast do I need to access the data?
- How long do I need to retain data?
- How secure does it need to be?
- What regulatory requirements need to be adhered to?

Structuring Data

A lot of hard work can be avoided by organising the data into files or data structures that best suit the purpose of the project. This is why we use databases. Databases are essentially tables of data that hold records made up of fields. Often these databases are complex stand-alone systems so software developers need to find a way of storing data in more fundamental ways such as basic text files.

A text file of unorganised values is not going to be easy to access, sort or process so there are a number of ways data can be structured. CSV files are Comma Separated Value format. Each value data point is separated from the others with a comma character. CSV is a delimited data format that has fields or columns separated by the comma character and records or rows terminated by new lines. Below in figure 1.7 an example of a table containing two records with three fields is illustrated.

Fig 1.7

	A	B	C
1	Name	DOB	Role
2	Brian Campeau	22/07/1979	CEO
3	Richard Jeffrey	19/11/1983	CFO

CSV

Name, DOB, Role,
Brian Campeau, 22/07/1979,CEO,
Richard Jeffrey,19/11/1983,CFO,

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It uses tags in the same way that HTML tags format a web page. XML tags format records and fields. These structures enable access to database-formatted data with minimum impact on the amount of storage required. Figure 1.8 is an example of XML containing two records with four fields. Each Field is called an element within the record.

Fig 1.8

```
<breakfast_menu>
    <food>
        <name>Belgian Waffles</name>
        <price>$5.95</price>
        <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
        <calories>650</calories>
    </food>

    <food>
        <name>Strawberry Belgian Waffles</name>
        <price>$7.95</price>
        <description>Light Belgian waffles covered with strawberries and whipped cream</description>
        <calories>900</calories>
    </food>
</breakfast menu>
```

breakfast_menu			
food			
name	**price**	**description**	**calories**
Belgian Waffles	$5.95	Two of our famous Belgian Waffles with plenty of real maple syrup	650
Strawberry Belgian Waffles	$7.95	Light Belgian waffles covered with strawberries and whipped cream	900

Storage Media

Data storage is the recording and storing of information in a storage medium. Recording data is accomplished by virtually any form of energy. Electronic data storage requires electrical power to store and retrieve data. Data storage in a digital, machine-readable medium is digital data. Barcodes and magnetic ink character recognition (MICR) are two ways of recording machine-readable data on paper. Electronic storage of data can be grouped into Primary, Secondary and Tertiary.

Primary Storage includes the RAM and ROM that directly support the CPU. It is volatile memory, which means all data is lost after the device is powered down.

Secondary Storage differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered down. It is non-volatile memory. Examples of Secondary Devices include; Hard Drives (these can be in-built to a computer system or be stand-alone external drives), CD/ROM, DVD, flash memory (e.g. USB flash drives or keys), magnetic tape, standalone RAM disks and Zip drives.

Tertiary Storage typically involves an automatic mechanism that will attach removable mass storage media to a storage device when required. Data is often copied to secondary storage before use. It is primarily used for archiving rarely accessed information since it is much slower than secondary storage. This is primarily useful for extraordinarily large data stores, accessed without human operators. Typical examples include tape libraries and full back-ups.

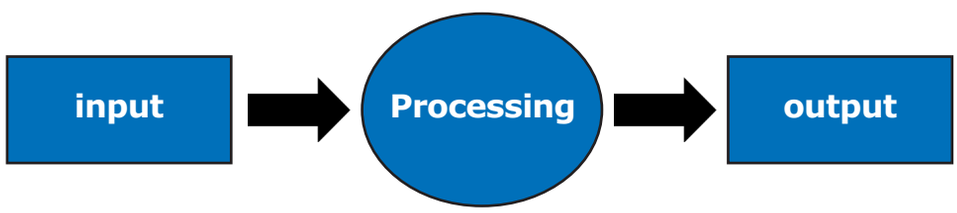
Representing Data

IPO Chart

We need to use tools to plan our software solutions. The most basic tool is the IPO chart. This chart identifies the input and output variables and the processes in between. For the Simple Calculator example see figure 1.9 below. This is a great place to start when designing a complex system solution. Identifying the data that goes in and what comes out is very helpful!

Fig 1.9

INPUT = Number1 and Number2
OUTPUT = Answer
PROCESS = Number1 + Number2 = Answer



Data Dictionary

Once we know what data needs to be handled, we need to design the variables to handle it. Creating a Data Dictionary is an important aspect of the Design Stage in developing a solution. It includes the names of the variables, the data types of the variables, the size of the data held and a description of the data. Returning to our Simple Calculator again here is a Data Dictionary in figure 1.10.

Fig 1.10

Variable Name	Data Type	Size	Description
IntNumber1	Integer	3	This is the first of two numbers to be added
IntNumber2	Integer	3	This is the second of two numbers to be added
IntAnswer	Integer	4	This is the sum of the two numbers entered

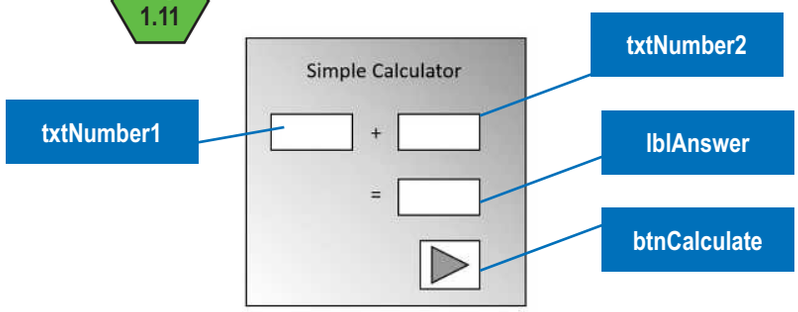
The Data Dictionary above shows the correctly named variables, their data types and descriptions. The size "3" indicates the number of bytes the variable can hold, in this case, the largest number that can be entered is 999. If we changed the size to "2" the largest number we could enter would be 99. It indicates the number of characters. We needed to put "4" as the size for IntAnswer because the largest possible value it may encounter is 999 + 999 = 1998. It is also possible to identify the scope of the variable. If it is limited to a subroutine, then the variable would be "LOCAL" or if it accessible from anywhere in the program is "GLOBAL".

A data dictionary assists the software developer to plan their program. When naming variables and setting data types it is important then to identify which Graphical User Interface (GUI) objects will handle each of the variables. Setting a consistent approach to naming variables will assist the programmer in naming and organising the objects and modules required in the solution.

Object Description Table

When designing your interface, especially in Visual Basic, you need to identify objects that will handle your variables, and in turn, your data. It is important to use Camel Case and Hungarian Notation in the naming of objects as well. You can see by the Graphical User Interface (GUI) in figure 1.11 that the two input objects are text boxes (txtNumber1 and txtNumber2). The OUTPUT will be displayed in a label (lblAnswer). The entire program will be executed when the button object (btnCalculate) is clicked.

Fig 1.11



Now that we have identified our objects, we can create an Object Description Table, see figure 1.12. This is a way of formally identifying the name, type, purpose and properties of each object. You can see in the table below that there is a column titled: Method/Event. This identifies how the objects are affected by things that happen (events) during the execution of the program or if the properties of the object are changed (methods). For example: lblAnswer is a label that will have its "text" property changed in the execution of the program. This is a method. The button btnCalculate triggers the program execution when the event_Click occurs. This is an event.

Fig 1.12

Object Name	Type	Method/Event	Properties	Description
txtNumber1	textbox	Method	Set Text: Null	Reads in IntNumber1
txtNumber1	textbox	Method	Set Text: Null	Reads in IntNumber2
lblAnswer	label	Method	Set Text: Null	Displays lblAnswer
btnCalculate	button	Event	Text: "Calculate"	When this is clicked it will trigger the execution of the calculation

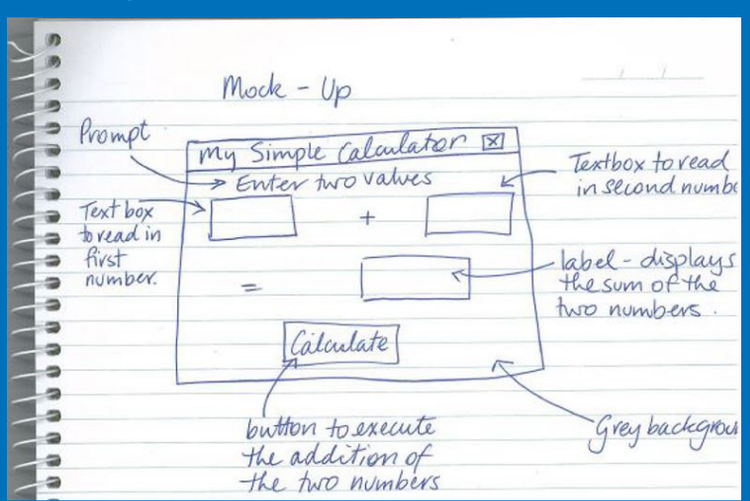
A good rule of thumb when planning a project is to create your design tools in this order:

1. Interface design as a Mock Up or Storyboard
2. From the interface design, label all the objects in the interface.
3. Create the Object description table from your labeled interface.
4. List the variables that each object handles.
5. Create the Data Dictionary from the variable list.
6. Work through the DFD in stages to organise your algorithm.
7. Write your pseudocode from the algorithm notes.

Mock Up - Design Layout

A Mock Up or Design Layout is a plan that identifies what the GUI will look like. The illustration below in figure 1.13 shows a rough hand-drawn design for the interface of the simple Calculator. It identifies each of the objects and their purpose as well as design elements. The more details you can include, the better prepared you will be once it is time to develop your solution. Perhaps even add the colour and font choices.

Fig 1.13 Mock Up



Pseudocode

Once you have decided on the GUI and the objects and variables required to solve your programming problem, it is time to define the events that take place when executed. Rather than going directly to Visual Studio and start typing up your code, it is essential that you plan your structure in an algorithm using pseudocode. Pseudocode is natural language that is easy to read but follows the structure of a programming language.

The Features of Pseudocode

- Every algorithm must begin and end with START and STOP or BEGIN and END.
- When assigning data to a variable use ← . Example: IntNumber ←5
- Show each calculation required. Example: IntAnswer ←IntNumber1 + IntNumber2
- Displaying data as output. Example: lblAnswer.Text ←IntAnswer or Display IntAnswer
- Decision Control Structures. Example: IF condition THEN Action1 ELSE Action2 ENDIF
- Case Control Structures. Example: Case where condition, CASE1 Action1 CASE2 Action2 END CASE
- Repetition Counted Loop Control Structures. Example: FOR counter ← first value to last value END FOR
- Repetition Post-Test Loop Control Structures. Example: REPEAT actions UNTIL condition
- Repetition Pre-Test Loop Control Structures. Example: WHILE condition DO actions END WHILE
- Use indentation and spaces to illustrate how control structures are incorporated
- Conditions: ==(equal), <> (not equal), > (greater than) and < (less than).

Below is an Algorithm written in Pseudocode that allows the user three attempts at getting their username and password correct. The program checks the password is correct at each attempt and prompts the user with the number of remaining attempts. Can you name six algorithm features in the example below?

```
START
   strUsername ← BillBurr123
   strPassword ← Summer1965

   FOR counter = 1 to 3 DO
      Read in strUsernameEntered
      Read in strPasswordEntered
            IF (strUsername == strUsernameEntered AND strPassword == strPasswordEntered) THEN
                  Display Prompt "Access Granted."
            ELSEIF
                  Display Prompt " Incorrect credentials. You have " (3 - Counter) " attempts left"
            ENDIF
   END FOR
END
```

Formatting and Structural Characteristics of Files

So far we have only discussed software solutions where the user enters the data via a keyboard and mouse. This is not always the most efficient method especially when we have large amounts of complex data. Sometimes data is collected from CCTV capture, such as car registration plates moving through an intersection. Collecting data out in the field may require locations to be input which could be collected via a GPS device. When dealing with large amounts of data it is important that the data is structured in such a way that it can be accessed, sorted, searched, saved and retrieved. Software solutions such as our Simple Calculator do not save data for use after the program has been turned off. To create effective, robust software solutions these programs need to save data permanently so it can be accessed again after the software has been shut down. This requires files that can hold the data.

A fundamental approach is to use a file such as a text file that you can edit in a basic text editor. This is a great approach if you are making a simple application where the client needs to update prices. Examples of how Visual Basic can access and manipulate data stored in a text file can be found in the programming Chapter 5.

If you need a more structured approach such as the use of a spreadsheet you might need to upgrade from a basic text file to an XML file. When you have data structures such as records, it is best to be working with fields. XML uses tags in much the same way HTML is structured. The figure 1.14 shows a table of five records. It is described and controlled by the XML tags. An XML file is a basic text file that is "self-descriptive'. It can easily be read by software solutions and by humans, see figure 1.14. Excel can easily produce an XML file from your data. Excel can also produce Comma Separated Value files (CSV) that can store records in much the same way. CSV files store tabulated data in a basic text file that can be imported into a spreadsheet or database.

Fig 1.14

Records in an XML File

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <data-set xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  - <record>
      <FirstName>Bob</FirstName>
      <LastName>Smith</LastName>
      <StreetAddress>16 Boing Rd</StreetAddress>
      <City>Glebe</City>
      <Postcode>2078</Postcode>
    </record>
  - <record>
      <FirstName>Bill</FirstName>
      <LastName>Jones</LastName>
      <StreetAddress>45/8 Mt Gratte Ave</StreetAddress>
      <City>Lismore</City>
      <Postcode>2424</Postcode>
    </record>
</data-set>
```

Validation

The interface between machines and humans is fraught with opportunities for data loss, mistakes and possible nefarious intentions. Validation is the process of checking the input of data. Manual validation includes spell checking, proof-reading and fact checking. These are processes the user can conduct before entering data into a system. As a software engineer, it is important to anticipate human error when reading in data to your program by including validation in the code.

Common validation techniques include:

- Existence Checking
- Type Checking
- Range Checking

Existence Checking is a validation method that checks if data is present in a variable or object. This can be an important aspect of your interface design. If the user has not included an important data element, existence checking will detect it and prompt the user to enter the missing data. For example: in a pizza ordering app, if the user has not selected a pizza type, it is impossible to process an order, so a message window would appear to prompt the user to make a pizza type choice.

Type Checking is a validation method that checks the data type of the data entered by the user. This is vital to detect typing errors. If a "K" is typed into the input object for "Quantity", type checking will detect that the data type is not numerical and return a prompt to the user to enter a value to indicate how many pizzas they want to order.

Range Checking is a validation method that checks if a value sits between two limits. A common use of range checking would be on the input of "date of birth". To ensure those logging in to an app are over 16, the range check could test for dates after 1/1/2004. Similarly, dates before 1920 would also be excluded due to the unlikelihood that users would be over the age of 100.

Programming Languages

Programming Languages are coded instructions that both a human and a machine can understand. There are two main types of programming languages: interpreted and compiled. An interpreted language requires software to interpret and run the code. These languages rely on the browser software to interpret and run the code in their window. A common example of interpreted languages are those that run inside browsers on websites such as:

- PHP
- Excel
- XML
- HTML
- JavaScript
- Perl

Compiled Languages are converted into machine code and can run independently of other software. The code can be compiled in an Integrated Development Environment (IDE) or through a text editor. They create a stand-alone software application. Languages include:

- Python
- Visual Basic
- C++
- C#
- Java

Visual Basic is a compiled Object Oriented Programming (OOP) language. It enables the user to design a Graphical User Interface (GUI) easily with a point-and-click process in the IDE. The GUI is designed by arranging input, output and processing objects on a window or form. Each object is named and its properties edited according to its association with a data structure.

Procedures, Subroutines, Events and Modules

In Visual Basic(VB) there is a hierarchy of coding segmentation. A complete Visual Basic application is called a Project. The default name is WindowsApplication1. Within a project there may be many code files called Modules. These maybe called something like Form1.vb.

Once a module has been created we write the procedures as code. In VB there are two types of procedures; functions and subroutines. Functions perform tasks that return a value. Subroutines perform tasks but may not return a value. Below is a module that adds two numbers together. It is a Private Subroutine that reads in IntNumber1 and IntNumber2, adds them and displays the answer. It is a self contained program.

```
MODULE modAddition
    Private Sub AddNumbers (ByVal IntNumber1 As Integer, ByVal IntNumber2 As Integer)
    Dim IntAnswer As Integer
    IntAnswer ← IntNumber1 + IntNumber2
    MsgBox ← IntAnswer
    End Sub
END MODULE
```

If multiple subroutines were required and the intAnswer value shared between them, the subroutines would need to be public. Below is an example of a subroutine that executes when the "Calculate" button is clicked. It reads data from GUI objects but does not return a value. Notice how the subroutine is Public.

```
Public Sub Calculate_Click (sender As Object, e As eventsArgs) _ Handles Button1.Click
        Dim IntNumber1 as Integer
        Dim IntNumber2 as Integer
        IntNumber1 ← txtNumber1.Text
        IntNumber2 ← txtNumber2.Text
End Sub
```

A function procedure is a series of mathematical procedures enclosed by "Function" and "End Function" and returns a value to the code that called up the function. Below are two functions that return the result of a mathematical process. These functions can be called up from anywhere in the program.

```
Public Function AddNumbers (ByVal IntNumber1 As Integer, ByVal IntNumber2 As Integer)
        Dim IntAnswer As Integer
        IntAnswer ← IntNumber1 + IntNumber2
END Function

Public Function DivideByThree (ByVal IntAnswer as Integer)
        Dim DblThird As Double
        DblThird ← IntAnswer / 3
        Return DblThird
END Function
```

It is now possible to create a module that will use the subroutine and the two functions. The example on the next page shows how a Module called "CalculatorApp" utilises the functions "AddNumbers" and "DivideByThree" to process the input from subroutine "Calculate".

```
Public Class CalculatorApp

    Public Sub Calculate_Click (sender As Object, e As eventsArgs) _ Handles Button1.Click
        Dim IntNumber1 as Integer
        Dim IntNumber2 as Integer
        IntNumber1 ← txtNumber1.Text
        IntNumber2 ← txtNumber2.Text

        Call AddNumbers(IntNumber1, IntNumber2)
        MsgBox(IntAnswer)

        Call DivideByThree(IntAnswer)
        MsgBox(DblThird)
    End Sub


    Public Function AddNumbers (ByVal IntNumber1 As Integer, ByVal IntNumber2 As Integer)
        Dim IntAnswer As Integer
        IntAnswer ← IntNumber1 + IntNumber2
        Return IntAnswer
    END Function

    Public Function DivideByThree (ByVal IntAnswer as Integer)
        Dim DblThird As Double
        DblThird ← IntAnswer / 3
        Return DblThird
    END Function
End Class
```

Calls up the functions inside the subroutine so they can be used multiple times throughout the module.

Control Structures

To control the order in which your functions and procedures are executed, some structure needs to be put in place. There are three main control structures:

- Sequence
- Selection
- Iteration

Sequence is simply the structure and order of instructions in the correct sequence. The example below is an algorithm that reads in two values adds them together then displays the answer.

```
START
    Read In Number1                 (Instruction  1)
    Read In Number2                 (Instruction  2)
    Answer ← Number1 + Number 2     (Instruction  3)
    Display Answer                  (Instruction  4)
END
```

Decision control structures determine the direction of the program based on condition statements. The example below reads in two values, adds them together and tests if that answer is odd or even using the Mod of 2. If there is a remainder after Answer is divided by 2 then it is odd.

```
START
    Read In Number1
    Read In Number2
    Answer ← Number1 + Number 2
    IF Answer Mod 2 > 0 THEN            (IF condition is TRUE it will do Procedure 1)
        Display "The Answer is EVEN!"   (Procedure 1)
    ELSE                                (IF condition is FALSE it will do Procedure 2)
        Display "The Answer is ODD!"    (Procedure 2)
    END IF
END
```

Iteration is a control structure that repeats a set of procedures in a loop. These loops can be controlled by counting the number of times the loop completes or by testing conditions. The first example is a Counted Loop. This algorithm will read in three guesses via an input box.

```
START
    Writeline: Guess the Number! You get three guesses!
    FOR Counter 1 to 3 DO
        Guess ← Input Box( Take a guess )
    END FOR
END
```

The second example is a Pre-Test Loop. Here a condition must be met before the loop is executed. This algorithm shows the loop commences if the condition (the guess is wrong) is true.

```
START
    Writeline: Guess the Number! Keep guessing till you get it right!
    Guess ← Input Box( Take a guess )
    WHILE Guess <> Answer
        Guess ← Input Box( Take a guess )
    End WHILE
END
```

The last example is a Post-Test Loop. This algorithm tests the outcome of the first loop and keeps looping until the guess is correct.

```
START
    Writeline: Guess the Number! Keep guessing till you get it right!
    REPEAT
        Guess ← Input Box( Take a guess )
    UNTIL Guess = Answer
END
```

Searching

Linear Search

If we were looking for "Bunmi" manually in our list of names below, we might start at the top and run our finger down the list to check each element. Linear Search does exactly that. It begins at the start of the array and uses a loop to check each element until it is found. Below is an algorithm to search for Bunmi in the list. EOF means "end of file".

```
START
    Counter ← 0
    Found ← False
    WHILE Found = False AND Not EOF DO
        IF Names (Counter) <> Bunmi THEN
            Counter ← Counter + 1
        ELSE
            Found ← TRUE
        END IF
    END WHILE
END
```

0	Robert
1	Manpreet
2	Brian
3	Julia
4	Sudha
5	Bunmi
6	Jackson
7	Maxine

This type of searching approach is fine for small lists, but if you have a telephone directory of data to search, this is very inefficient.

Binary Search

Clearly we need a better method than a Linear Search for larger files, and that is where Binary Search is useful. Unfortunately the list must be SORTED first to conduct a binary search.

You can see our array Names(7) below has been put into alphabetical order making it easier to search.

Names (0) = Brian
Names (1) = Bunmi
Names (2) = Jackson
Names (3) = Julia
Names (4) = Manpreet
Names (5) = Maxine
Names (6) = Robert
Names (7) = Sudha

First we define the lowest, highest and middle elements of the list.

LOWEST element ← Names (0) = Brian
MIDDLE element ← Names (4) = Manpreet
HIGHEST element ← Names (7) = Sudha

Now we check the name we are searching for "Bunmi" against the MIDDLE element.
Is Bunmi before or after Manpreet in the alphabet? Bunmi is before(<) Manpreet. So we now redefine our lowest, middle and highest. Because Bunmi is before Manpreet we make the old MIDDLE the new HIGHEST and then find a new MIDDLE.

LOWEST element ← Names (0) = Brian
MIDDLE element ← Names (2) = Jackson
HIGHEST element ← Names (4) = Manpreet

Again we test to see if Bunmi is > or < our new MIDDLE element Jackson. Since Bunmi < Jackson, we set the MIDDLE to be the new HIGHEST once again.

LOWEST element ← Names (0) = Brian
MIDDLE element ←Names (1) = Bunmi
HIGHEST element ← Names (2) = Jackson

Now once again we check our MIDDLE element and we have found our name "Bunmi".

Comparison of Linear and Binary Searches

A Linear Search needs to potentially conduct as many operations as there are elements in the list being searched. Bunmi was the 6th element in the unsorted list so at least 6 operations were required to find that name. However, with the Binary search only 3 operations found the same element. How many operations would take to find Robert?

Operation 1

LOWEST element ← Names (0) = Brian
MIDDLE element ← Names (4) = Manpreet
HIGHEST element ← Names (7) = Sudha
Maxine > Manpreet so MIDDLE is the new LOWEST

Operation 2

LOWEST element ← Names (4) = Manpreet
MIDDLE element ← Names (6) = Robert
HIGHEST element ← Names (7) = Sudha

Binary search is more complex to code, but it is more efficient and the most suitable solution for a large index of data. Linear is easier to code and most suitable for short files. We often identify the efficiency of a method by its worst case scenario. If the item you are looking for is the last one in a list of n number of items, the item will be found in O_n number of operations. If the list is 100 items long, it will take 100 operations. For Binary search the efficiency is increased logarithmically. An item in a list of n items will be found in O_{logn} which is about 30 operations for worst case scenario of a 100 item list.

Sorting

Despite many languages containing a 'sort' function, in major software solutions it is important to manage the sorting of data with control structures. We will look at two key sorting methods: Selection Sort and Quick Sort.

Selection Sort

The simplest sort is the Selection Sort which functions the way you would naturally sort items. It looks through the whole list looking for the smallest item and places it at the beginning of the list. This is called a PASS. The sort then passes the rest of the unsorted list for the next smallest item and adds it to the new sorted section. It COMPARES the item to the last item in the sorted section and SWAPS them so they are in order. This process repeats until the list is completely sorted. This is where arrays become very useful. We can use the index of each data item to reposition the data into a new order. We use nested loops. This allows each data item in the array to be compared with every other item in the array on each PASS. This means it can be an inefficient solution for long lists. An array(n) can require up to n^2 number of operations to sort.

A Selection Sort Algorithm to sort 100 data items in an array ArrayList(99) follows. A PassCounter repeats each PASS 97 times. For each PASS the smallest item is searched for and then put at the start of the unsorted section of the array.

```
BEGIN

    FOR PassCounter: 1 to (ArrayLength - 2) DO

        Smallest ← ArrayList(PassCounter - 1)
        SmallestIndex ← PassCounter

            FOR ItemCounter: (PassCounter - 1) to (ArrayLength - 2) DO

                IF ArrayList(ItemCounter + 1) < Smallest THEN
                    Smallest ← ArrayList(ItemCounter + 1)
                    SmallestIndex ← ItemCounter + 1
                END IF

            END FOR

            Temp ← ArrayList(PassCounter - 1)
            ArrayList(PassCounter - 1) ← ArrayList(SmallestIndex)
            ArrayList(SmallestIndex) ←Temp

    END FOR

     Display ArrayList(99)

END
```

Unsorted Array

k	e	a	m	s	q	b	j

First Pass and Swap

a	e	k	m	s	q	b	j

Second Pass and Swap

a	b	k	m	s	q	e	j

Third Pass and Swap

a	b	e	m	s	q	k	j

Fourth Pass and Swap

a	b	e	j	s	q	k	m

Fifth Pass and Swap

a	b	e	j	k	q	s	m

Sixth Pass and Swap

a	b	e	j	k	m	s	q

Final Pass to Swap - Sorted Array

a	b	e	j	k	m	q	s

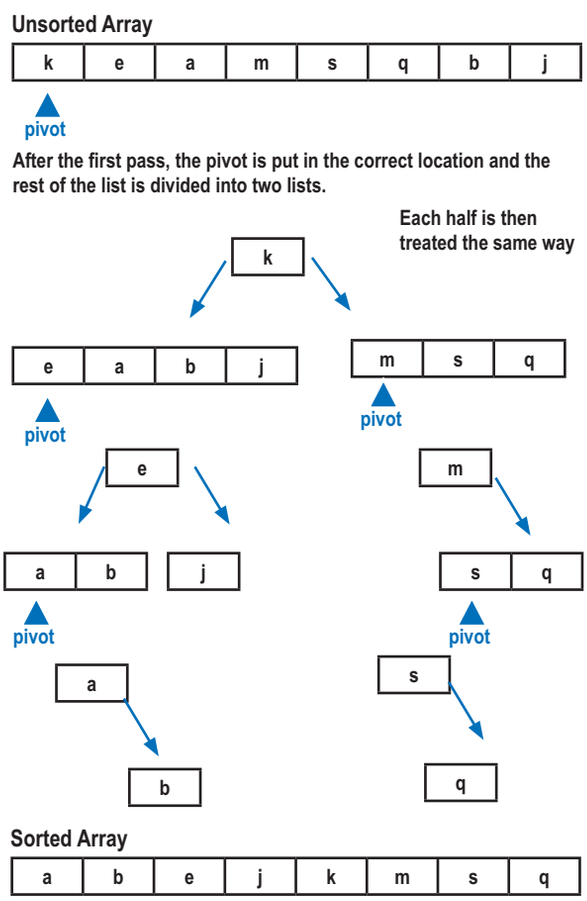
Quick Sort

Quick Sort is more sophisticated sorting technique using Divide and Conquer around a Pivot. This is a more complex sorting solution that uses recursion. Recursion is where a procedure or function calls itself. The algorithm sets StartIndex and EndIndex to the beginning and end of the array. In this situation we set the pivot to the first item. The function checks the StartIndex and the EndIndex data against the Pivot. Depending on the data it will add 1 to the StartIndex or minus 1 from the EndIndex. When data is found to be lower than the ArrayList(StartIndex) it is swapped.

You can see at the bottom of the algorithm that the Function QuickSort calls itself up to run (0 to EndIndex) then again for (StartIndex to ArrayLength). The nature of Quick Sort makes it more efficient and can sort a large list in only $O_{n \log n}$ number of operations.

The bulk of the algorithm runs a pass. Each pass places the pivot in its correct location with all the values lower than it on one side, and all the values higher on it on the other. Each of these sides are now treated as separate lists that will run the pass through. For each half the process continues to divide and conquer until each item has become a pivot and is located in its correct place.

```
START FUNCTION: QuickSort
BEGIN
   IF ArrayLength >1 THEN
      Pivot ← ArrayList(0)
      StartIndex ← 0
      EndIndex ← ArrayLength

      WHILE StartIndex <= EndIndex DO

        WHILE ArrayList(StartIndex) < Pivot DO
           StartIndex ← StartIndex + 1
        END WHILE

        WHILE ArrayList(EndIndex) > Pivot DO
           EndIndex ←EndList - 1
        END WHILE

        IF StartIndex < = EndIndex THEN
           Temp ← ArryList(StartIndex)
           ArrayList(StartIndex) ← ArrayList(EndIndex)
           ArrayList(EndList) ← Temp
           StartIndex ← StartIndex + 1
           EndIndex ←  EndIndex - 1
        End IF

      End While

   Function QuickSort (0 to EndIndex)
   Function QuickSort(StartIndex to ArrayLength)

   END IF

END
```



Testing

To test our software solution, first we need to develop a table of Test Data. The test data needs to include the kind of input that would be expected of the user. For example let's look at this basic guessing game program:

```
START
        Writeline: Guess the Number! The number is between 1 and 10!
        FOR Counter 1 to 3 DO
                Guess ← Input Box( Take a guess )
        END FOR
END
```

The expected input would be "1, 2, 3, 4, 5, 6, 7, 8, 9, 10", but what would happen if you entered "127", or "eleven", or "*^$&(?@!" ? Programmers need to build solutions that can handle any input . For example: the correct response for INPUT = 127 would be "You must enter a number between 1 and 10". The correct response to INPUT = eleven or INPUT = *^$&(?@! would be "You must enter a number not other characters".

When creating test data you must also include:

- Valid Input (Example: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
- Invalid Input (Example: eleven or 127)
- Unexpected Input (Example: (*&*%Q#w%p$)

Trace Tables

Trace Tables are designed to find logic errors when valid data is entered but incorrect output data emerges. We list all the variables across the top and enter valid input and follow the program line by line entering the changes to the values in the variables as we go. Let's look at a simple example; below is a program that sets the Answer to the guessing game at 6. All three guesses will be read in and placed into an array called Guess (2) . The FOR loop reads in the guesses into each array element. An IF statement checks the array for a correct answer and assigns points based on whether it was the first, second or third guess.

```
START
    Declare Guess (2) ← Integer
    Declare Answer ← 6
    Declare Points ← Integer
    Writeline: Guess the Number between 1 and 10!
    Writeline: You get three guesses!
    FOR Counter 0 to 2 DO
        Guess(Counter) ← Input Box( Take a guess )
            IF Guess(Counter) = Answer THEN
                Writeline: You got it right! You have (Points) number of points!
                Exit FOR
                IF Counter = 0 THEN
                        Points ← 10
                ELSEIF Counter = 1 THEN
                        Points ← 5
                ELSEIF Counter = 2 THEN
                        Points ← 1
                ENDIF
            ELSE
                Writeline: Incorrect! Try again.
            END IF
    END FOR
    Writeline: You did not guess the number!
END
```

First we need Test Data that contains adequate variation. In figure 1.15, each set of data tests a different aspect of the program, the first set of data tests a correct answer for Guess(1), the second set of data tests no correct answer, the third set of data tests a correct answer for Guess(0), etc. All possible correct solutions are tested and there are two sets that test for no correct answer. We have entered some non valid data too so we once we have added validation this can be tested.

Fig 1.15

Test Data		
Guess (0)	Guess (1)	Guess (2)
3	6	
4	2	1
6		
(	7	3
2	5	6
12	W	4

So now let's put the first set of data into a Trace Table in figure 1.16. Across the top of our trace table are all the variables used in the program and down the side identifies each operation in the code. You can see the first line contains the set Answer as 6, then each of the guesses for each iteration of the FOR loop. Finally 5 points are allocated because Guess(1) contained the correct answer.

Fig 1.16

Trace Table			
Guess (0)	Guess (1)	Guess (2)	Points
3 Continues to Guess (1) Prompts User "Higher"	6 Stops at correct Answer Prompts User "You Win"		5 Points
4 Continues to Guess (1) Prompts User "Higher"	2 Continues to Guess (2) Prompts User "Higher"	1 Prompts User "You Lose"	No Points
6 Stops at correct Answer Prompts User "You Win"			10 Points
(Input Error! Program cannot continue			
2 Continues to Guess (1) Prompts User "Higher"	5 Continues to Guess (2) Prompts User "Higher"	6 Stops at correct Answer Prompts User "You Win"	1 Point
12 Continues to Guess (1) Prompts User "Lower" despite being outside of range.	W Input Error! Program cannot continue		

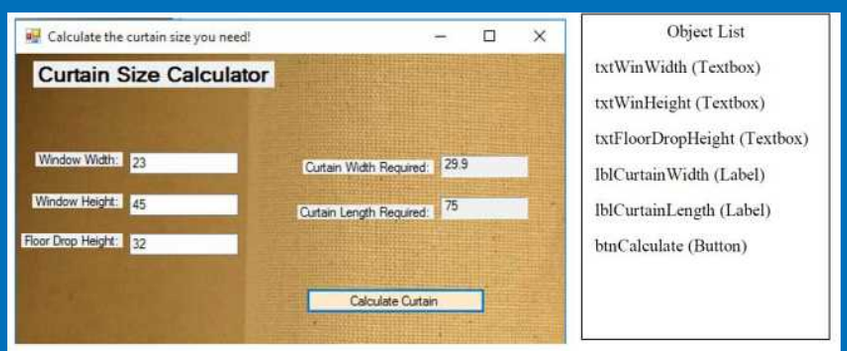
Internal Documentation

When writing software you should always include comments in your code to clearly indicate how it functions. There is nothing more annoying than coming back to a complex solution you have written in the past, only to discover you cannot remember what each variable is, or even what each section of code does. To avoid this, and to assist your clients in updating the code in the future, it is important to name your objects and variables according to type and purpose. Also, non-executable comments can be placed throughout the code to describe the function of the code.

Variable and Object Names

In Visual Basic the first stage is to build the GUI. You might use a text box to read in a data element such as Window Width, see figure1.17. To name the object, it is a common naming convention to include the object Type (Textbox) with the object's Purpose (Window Width). Object names need to be an uninterrupted string without any spaces so we use CAPITALS to break up the name. For example: txtWinWidth. This approach is called Camel Casing and makes the name easy to read. The combination of type and purpose is called Hungarian Notation. In figure 1.17 is an example of a simple application that calculates curtains given the size of a window: Curtain Calculator. A common confusion occurs when there are two similar objects - in this case, reading in the width of the window and the calculation of the curtain width. The object names clearly distinguish these two objects. Similarly the names of your variables should contain the data type and the purpose.

Fig 1.17 Graphical User Interface (GUI)



Comments

Comments are not part of the executable code. They do not slow down the operation of the program and do not affect the way the software functions. Comments do, however, make understanding the code so much easier. With fully documented code highly complex software solutions can be easily understood when you return to the program later, or if you need to hand it over to someone else for editing. The VB code in figure 1.18 shows the comments in green. They are clearly organised using headings and spaces to make the code readable.

Fig 1.18 Comments

```
Imports System                                  'Imports System
Imports System.Xml                              'Imports XML System
Public Class frmLogin                           'Public Class
    Public Shared currentUsername As String     'Declare Public Variable
    Public Shared currentLocation As String     'Declare Public Variable
    Public Shared currentName As String         'Declare Public Variable
    Public Shared UserRights As String          'Declare Public Variable

    'Declare variables
    Dim AccountUsername As String               'Declare Username Variable
    Dim AccountPassword As String               'Declare Password Variable

    'Declaring Account Name as a Sentence
    Dim AccountName As String
```

Unit 3 Outcome 1
Programming
Review Questions

1. Describe the relationship between data and variables.

2. Give an example where data might best be stored in a record.

3. Name two important features of an array.

4. How many items are included in the array: Products[14]?

5. Why would a developer wish to use a Hash Table to store their data?

6. How is the 'mod' function used to place data within a hash table?

7. How are CSV files different from XML?

8. What is the difference between primary storage and secondary storage?

9. Max Bellis has a hotel and needs a new system to calculate the final bill for each guest. Each guest's bill is calculated on the number of nights they stay and the cost of any room service. Draw an IPO chart for a solution for this billing calculator.

10. Rose Teal has an online business selling her knitting on Etsy. She needs a software solution that reads in the number of balls of wool used on a knitted product and the number of hours she spent creating the product. She wants to be paid $30 per hour and retrieve the cost per ball of wool at $5.00 each. For Rose's Knitting Calculator:

    a) Create a Data Dictionary
    b) Create an Object Description Table
    c) Create a GUI Mock Up.
    d) Write the pseudocode.

11. Identify three types of validation you could include in Rose's calculator.

12. What is the difference between interpreted languages and compiled languages?

13. What is the difference between a procedure and a function?

14. Name the three types of Control Structures. Give an example of each.

15. Describe how Linear Search works and when would it be most appropriate to use.

16. Describe how Binary Search works and why you would implement it in a solution.

17. Describe the difference between Selection Sort and Quick Sort in terms of efficiency.

18. Read the following algorithm written in pseudocode and create:
    a) A Table of Test Data
    b) A Trace Table

```
START
    Read in Name
    Read in Age
    IF Age >= 18 THEN
        Display "You may enter this site"
    Open URL
    ELSE
        Display "You are not permitted to enter this site"
    END IF
END
```

19. Write an example of a variable name using Hungarian Notation and an example of a GUI Object using Camel Case.

20. What is the purpose of including internal documentation in your code?

Applied Programming Projects

1. Create the Knitting Calculator from Question 10.

2. Create an App solution for Zhou's Coffee to Go, allowing the customer to choose the type and size of coffee as well as type of milk. The App should calculate the total price and produce an invoice of the order to the screen. All prices should be stored in an XML file to be retrieved by the App.

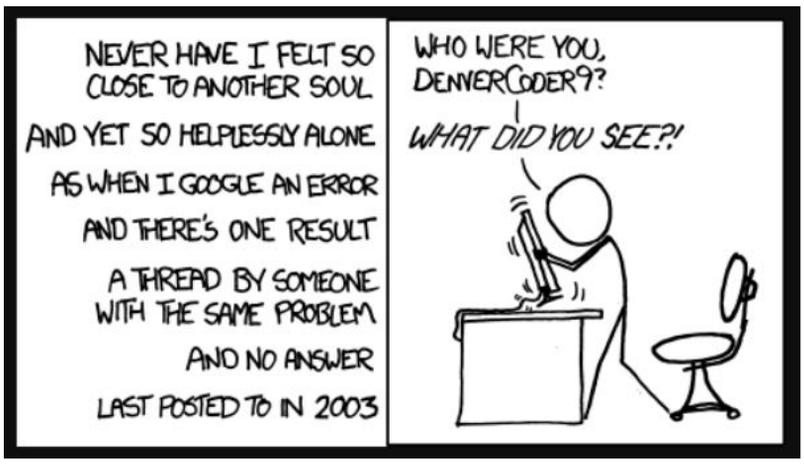
3. Create a list of items in a text file. Create an application that allows the user to click a button to run a linear search of the data and another button to run a binary search.

4. Create an application that reads in at least 100 data items from a stored file and sorts it A-Z.

5. Advanced students might like to create an application where different users have different levels of access. Hash the username and passwords of these users.

6. Produce an application that uses the time monitoring features of VB. For example you could create an application that measures the time you are on the computer and issues warnings to the user to have a screen break.

All the Visual Basic programming tutorials and tasks are in Chapter 5!



https://xkcd.com/979/

Online Support

Want to use a different language for your SAT? You can! Take care to check with your teacher first! You teacher needs to be able to support your progress through the SAT and also be able to assess your work.

- **GrokLearning.com (Python, JavaScript)**
- **CodeAcademy.com (Java, C++, C#, PHP)**
- **EdX.com**
- **ocw,mit.edu/courses/intor-programming**

https://xkcd.com/1425/

Unit 3
Area of Study 2

Analysis and Design

In this area of study students construct the framework for the development of a software solution that meets a student-identified need or opportunity. This is the first part of the School-assessed Task (SAT), involving analysis and design, with the second part undertaken in Unit 4, Area of Study 1. Students prepare a project plan that includes student-determined and teacher-provided milestones that take into account all stages of the problem-solving methodology covered in this outcome and in Unit 4, Area of Study 1. A range of methods is used to collect data for analysis. Analysis tools and techniques are used to depict relationships between data, users and digital systems and to document the solution requirements, constraints and scope as a software requirements specification. Students generate and document two or three design ideas for creating their solution. These could include annotations to indicate key functions and appearance. Evaluation criteria are developed and applied to select the preferred design idea. This design is then fully detailed, addressing the functionality and the user interface of the solution.

Key knowledge

Digital Systems

- Security considerations influencing the design of solutions, including authentication and data protection

Data and Information

- Techniques for collecting data to determine needs and requirements, including interviews, observation, reports and surveys

Approaches to problem-solving

- functional and non-functional requirements
- constraints that influence solutions, including economic, legal, social, technical and usability
- factors that determine the scope of solutions
- features and purposes of software requirement specifications
- tools and techniques for depicting the interfaces between solutions, users and networks, including use case diagrams created using UML
- features of context diagrams and data flow diagrams
- techniques for generating design ideas
- criteria for evaluating the alternative design ideas and the efficiency and effectiveness of solutions
- methods of expressing software designs using data dictionaries, mock-ups, object descriptions and pseudocode
- factors influencing the design of solutions, including affordance, interoperability, marketability, security and usability
- characteristics of user experiences, including efficient and effective user interfaces
- development model approaches, including agile, spiral and waterfall
- features of project management using Gantt charts, including the identification and sequencing of tasks, time allocation,
- dependencies, milestones and critical path

Interactions and Impact

- goals and objectives of organisations and information systems
- key legal requirements relating to the ownership and privacy of data and information.

Key skills

- select a range of methods to collect and interpret data for analysis
- select and justify the use of an appropriate development model
- apply analysis tools and techniques to determine solution requirements, constraints and scope
- document the analysis as a software requirements specification
- generate alternative design ideas
- develop evaluation criteria to select and justify preferred designs
- produce detailed designs using appropriate design methods and techniques
- create, monitor and modify project plans using software.

Problem Solving Methodology

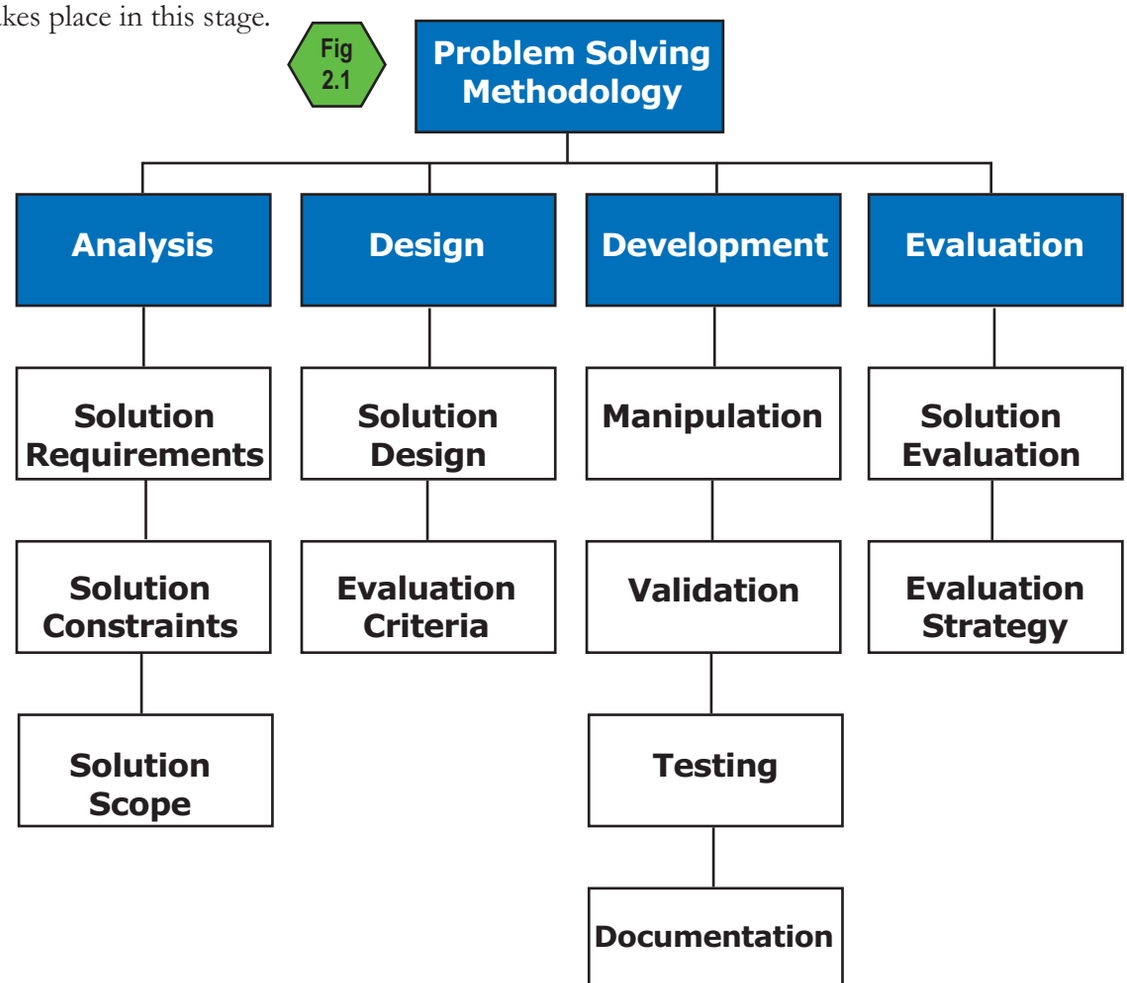
Problem Solving Methodology is the process used in this study to solve information problems. The diagram below in figure 2.1 is the framework for producing any software solution. It is crucial that all students of Software Development are familiar with the four stages: Analysis, Design, Development and Evaluation.

The Analysis stage is where the developer investigates the problem that needs to be solved with an information system. It is at this stage when the requirements of the solution are identified. These are the things the software must be able to do. It is important that other aspects of the software meet the needs of the end users and herein lie the non-functional requirements. These relate to the characteristics of the software such as ease of use. After the requirements are identified it is important to identify the constraints of the project. These are the limiting factors that affect the development of the software: cost, time, skills of users, materials and resources available. Finally the scope is an outline of what is included and what is not included in the final software to define a boundary of what needs to be in the final product.

The Design stage begins when the outcomes of the analysis stage have clearly outlined the functional and non-functional requirements, constraints and scope of the final solution. This stage incorporates the design of the solution - how it looks and how it will work. This can involve the interface design, the planning of functionality solutions by creating data dictionaries, use case diagrams, data flow diagrams and algorithms. Effectiveness and efficiency are taken into account to ensure the best possible solution. Once the solution foundation is designed, it is important that a list of evaluation criteria are developed to test against, once the solution has been completed and tested.

The Development stage is when the actual coding of the solution takes place. Important factors to be taken into consideration at this stage are data validation to be incorporated into the solution and clear documentation both as internal documentation and as other user documentation. Testing the solution for errors also takes place in this stage.

Fig 2.1



The Evaluation stage is composed of two steps: 1. The development of strategies to investigate the effectiveness and efficiency of the solution, and 2. Reporting on the outcome of the evaluation strategy. The evaluation strategy usually involves an investigation after the solution has been provided to the end users after enough time has passed for them to become familiar with its use. A combination of data collection methods can be used to collect relevant data to support the evaluation criteria list developed at the design stage. This will involve a wide range of tasks including interviews, running different scenarios, dealing with errors etc. Reporting on the outcome of the software solution is the final task which should outline the success of the new system over the one it replaces.

Analysis

A new information system is usually in response to a communication or efficiency issue arising in the operations of the organisation. To ensure the new system will meet the needs of the organisation it is important to research the requirements. To do this the developer needs to collect data from all stakeholders to ensure the solution they develop meets the needs of everyone affected and it is aligned with the organisation's goals and objectives.

Data Collection

When beginning a new software project is essential that the requirements are clearly understood, so the developer needs to investigate and research what is needed in the solution. To do this, the developer needs to collect data to analyse.

Interviews

The best place to start is to interview the client who requires the software. Interviewing involves asking individuals questions about what they require from the new software solution. This method allows for a deeper understanding of the problem that needs to be solved and allows for information to come to light that has not already been considered by the developer. Interviews allow the developer to ask follow-up questions and pursue new lines of inquiry, depending on the what comes up in the conversation.

Focus Groups

Focus Groups are a type of group interview where interaction between participants is encouraged. The person conducting a focus group plays the role of a facilitator encouraging the discussion, rather than an interviewer asking questions. This form of data collection allows for;

- discussion on complex issues,
- a wide range of key stakeholders
- free open discussion,
- personal experiences, opinions and observations.

Surveys

When a large number of people are required to provide feedback for a new system, the best forms of data collection are surveys and questionnaires. These are easy to disseminate with online surveys like Google Forms and Survey Monkey. It is cost effective and efficient in the collection of a lot of data. It is important the questions posed in surveys are carefully designed to produce meaningful and useful responses. Since survey and questionnaires only provide answers to questions that the developers have already anticipated, it is often the second stage of data collection after an interview or focus group has been conducted. Once survey data is collected it can easily be analysed by counting responses. When surveys are not issued face-to-face, legitimate and meaningful responses are not guaranteed.

Observations

Observations of end users operating the old software system can provide developers with insight as to what can be included in the new system and what can be streamlined. Observations give the advantage of seeing the context of the software in action, this allows for the developer to understand the environment and the end users. The data collected from the observation can form the foundation of the listing of Use Case analysis. Use Case illustrates how the software is used by each individual and it can be documented and described in the analysis stage as a Use Case Diagram.

Reports

There are often written reports available on the current system that is to be replaced. These can include issues with the current system, usage reports and network audits. These reports are developed by managers or network administrators of the current system and they can provide insight into the limits of the current system and what needs to be improved.

Techniques for Collecting Quantitative Data

All investigations require the collection of both qualitative and quantitative data. Qualitative data is not measurable and is often made up of opinions or case studies. Quantitative data is measurable or countable and is easily collected through the use of surveys. It is also easy to analyse due to its numerical characteristics, making the creation of graphs of the data easier.

Quantitative data is collected by counting results such as:

- "How many times do you need to look up an ID number?"
- "Would you need access online in the field?"
- "Which data is required for a Product field?"

All these questions have responses that are countable:

- Number of times, (integer).
- Number of "Yes" results, (integer).
- Product ID, Product_Name, Product Description, Product_Weight. (specific list of variables)

Techniques for Collecting Qualitative Data

Qualitative data, however is collected by recording non-measurable results. It can be collected through methods of observations, one-to-one interview and conducting focus groups. Qualitative data is about the emotions or perceptions of people and this can provide insight into human behaviour and day-to-day practice in the workplace as well as the environmental context where the software is used. Qualitative data can be investigated by asking these types of questions:

- Tell me about a time when the system was down. What led up to that situation?
- What circumstances led to data loss with the current system?
- Have you experienced better systems elsewhere? If so describe how they worked.
- Explain how the current system frustrates you. What improvements would you suggest?

These types of questions can result in detailed case studies or stories that can provide insight into functional and non-functional requirements for the new system.

Requirements

The solution requirements define what the software solution does. Solution Requirements fall into two categories:

1. Functional Requirements: what the software will do, what the input will be, what the output will be and how the data is processed, stored and transferred.
2. Non-Functional Requirements: how the software will function.

Functional & Non-Functional Requirements

Here is an example: Yang Zhou needs a new system to allow his baristas to manage coffee orders in his cafe. He has watched many errors occur between the instance when the order is taken at the counter, and when the order is filled at the coffee machine. Yang wants a system that reads in the order, calculates the cost, and produces a printed receipt for the customer and a record that appears on the screen above the coffee machine.

The Functional requirements for Yang's Cafe solution would include:

- Reads in Coffee Order details
- Calculates the total cost of the order
- Prints out the full details of the order
- Stores a record of the order that is displayed on the screen

Non-Functional Requirements relate to how the software works including:

- how easy it is to use, (Usability)
- if it is portable from one environment to another. (Portability)
- if it works reliably (Reliability)
- how it copes with all kinds of data input and use (Robustness)
- how easy it is to update and maintain (Maintainability)

The Non-Functional Requirements for Yang's Cafe solution would include:

- Easy to learn how to use (Usability)
- Easy to fix errors in data entry (Robustness & Usability)
- GUI with a touch screen can make it quicker to enter data(Usability)
- Validation of data entry (Robustness)
- Handles a wide range of inputs without providing incorrect output (Reliability)
- Data taken by wait staff on tablet is easily transferred to cash register (Portability)
- Prices are stored in an easily accessible text file for updates (Maintainability)

Solution Constraints

It is not always feasible to build all the possible features into a solution, there are usually constraints that the developer needs to take into consideration. Constraints influence what can be built and these include economic, legal, social, technical and time factors. These five factors limit the development of the solution and are out of the control of the developer or client.

Economic Constraints: As with all commercial projects, there will be a budget to adhere to.

Technical Constraints: The final organisation that will use the software solution will already have equipment requirements such as operating systems, other software and hardware that the new system must be compliant with. Depending on the nature of the organisation, security needs to be technically tailored to suit the environment. The processing speed and the memory capacity of the hardware the system will be using is a major technical constraint. The solution will have to be compatible with other systems that will remain in place.

Social Constraints: Not everyone is keen or able to up-skill with new technology. The new system will need to take into account the expertise of end users and the amount of support they will require.

Legal Constraints: There are a range of legal obligations software developers need to adhere to. These cover data management security, privacy of users and clients, and copyright ownership of the final product through clear agreements between the developers and the clients.

Time Constraints: are always prevalent in all projects. A deadline will always determine how much can be completed with the available resources.

Solution Scope

The scope of a project is the definition of what will be included and what will not be included. Factors that determine the scope of solutions vary depending on the importance of each of the requirements and are limited by the constraints. It is important for a software developer to be clear about what is included in the scope of the project and what is not. This may sound obvious but consider this example: A new app is being developed to assist the school in providing the canteen managers to be prepared for school lunch orders. The app will allow students to post their lunch order before 11am and the canteen management can prepare enough lunches with minimal waste. From the manager's point of view this is a great management tool and it has the potential to include all budgeting including developing costings for weekly orders from suppliers.

Due to time and cost constraints - it was decided by the project manager that the scope would include:

- Student orders input from mobile networked devices
- Total numbers of each lunch product required is calculated for 11am

The scope will not include:

- Budgeting components for weekly supply orders.

It is important to determine the scope of the project to tackle scope creep. Scope creep, or requirement creep, is how a project's requirements tend to increase over the life of a project. This occurs during the progress of the project where the requests from the client change or increase. Adding new requirements can drastically increase the amount of time needed to complete the task leading to disagreements on project costs.

Security Considerations

When a developer is about to embark on producing some software, it is important to investigate any security considerations that may influence the design of the solutions and how data protection can be implemented. A software developer is obligated to consider how the data will be authenticated when entered into the system and protected once saved.

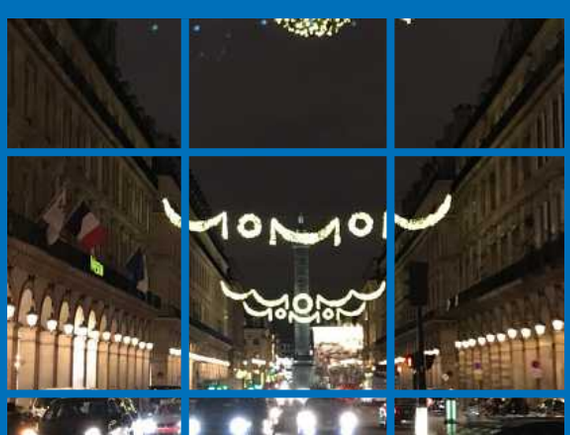
Authentication

Authentication is the process in which the personal credentials of a user are provided and then compared to those on file in a database of authorised users. This process is commonly called "signing in" or "logging on" only to the areas of a system related to their responsibilities.
There are a growing range of different options to authenticate users. The most often used and best recognised is the username and password authentication process, but with increased technological advancement we have a wider range of options. Biometrics read biological elements of the user, such as a finger print (mobile phones currently use this), face recognition or iris scanning. These can be useful if passwords are difficult to remember. It is possible a software solution will not have the resources

required for sophisticated biometrics but can offer multi-factor authentication. This is commonly used when a password is forgotten and needs to be reset. If the user has an email or mobile phone stored in the system, a new password request can send an email or text message for the user to reply to. This demonstrates to the system that the user has access to their email and mobile phone. Recently Google and Apple have joined forces to provide two factor authentication via a mobile phone app. If the user is trying to access their Gmail on a computer that is not their usual device, Google provides them with an option to authenticate on their iPhone via the Gmail app.

Systems also need to differentiate human users from non-human users. During the authentication process, the user may be asked to respond to questions that rely on interpretation. These are called captcha. They often require typing in letters or numbers that are distorted or selecting a section of an image that contains elements. Figure 2.2 below is an example that asks the user to select sections of the photo containing cars.

Fig 2.2 Captcha



Data Protection

How the data will be stored and retrieved also needs to be considered in the solution design. If data is to be stored on drives accessible via a network, security of that data needs to be implemented and integrated into the solution design. The data needs to be kept secure and backed-up. This means the data needs to be copied and stored outside of the system in case of data loss. Surge protection and an uninterrupted power supply (UPS) can provide some protection from event-based threats where electrical surges or power blackout can cause data to be lost through the shut down or damage of hardware.

Accidental threats can emerge through user incompetence. New software systems need to keep in mind that users can forget to close files, log off or accidentally delete data. Validation measures integrated into the system can provide some protection. Validation can limit the user from entering invalid data as well as implementing a process that could be harmful to the system. Some accidental threats are categorised as event-based which occur outside of the control of the users or the managers of the system. They include things like power surges or power failures. If the network connection is down, updating data can be corrupted or lost. Other non-technical events include water leaks in a building or other damage to the building that may affect the system, such as fire. It is because of these type of threats, a copy of back-ups should be kept off site.

Deliberate threats can delete data or hold it hostage if trojans, viruses, or ransomware are installed. Implementing limits to the user's access to the system may prevent these issues. If users have access to enter data, but not upload data of a particular type or size, this can protect databases from malware. There is a wide range of malware types which we will explore in Chapter 4 on Cyber Security. Keep in mind when designing a software solution, how data can be protected from accidental or deliberate threats.

Project Plan

Gantt Charts are tools that manage a project. They allow developers to organise their time and resources over the life of the project. A Gantt Chart is basically a time-line that lists all the tasks that need to be done and in what order.

Elements of a Gantt Chart

A Gantt Chart is comprised a table on the left hand side that is composed of six columns: Task Name, Duration, Start, Finish, Predecessors and Resource names. See figure 2. 3 for an example.

- Task Names - list all the tasks the developer needs to do.
- Duration - is the length of time it will take to do the task
- Start - is the start date of each task
- Finish - is the deadline date for each task
- Predecessor - is the task that needs to be completed before this task can be started.
- Resources - the list of hardware, software, personnel and other resources required for the task.

The Gantt Chart Layout

Fig 2.3 Project Plan

Task Name	Duration	Start	Finish	Pre	Resource Names
Analyisis / Project Plan	8 days	Mon 24/04/17	Wed 3/05/17		
Gantt Chart	8 days	Mon 24/04/17	Wed 3/05/17		Manager, MS Project,Printer,
Purpose, Users, Scope and Constraints	3 days	Wed 3/05/17	Fri 5/05/17		Manager
Data Collection	2 days	Mon 8/05/17	Tue 9/05/17	3	Manager,Printer, Staff
Functional & Non-Functional requirments	2 days	Wed 10/05/17	Thu 11/05/17	4	MS Work,Printer
Analysis Tools: Context Diagram, Use Case Diagram	3 days	Fri 12/05/17	Tue 16/05/17	5	MS Work,PC, Printer
SRS	0 days	Wed 17/05/17	Wed 17/05/17		

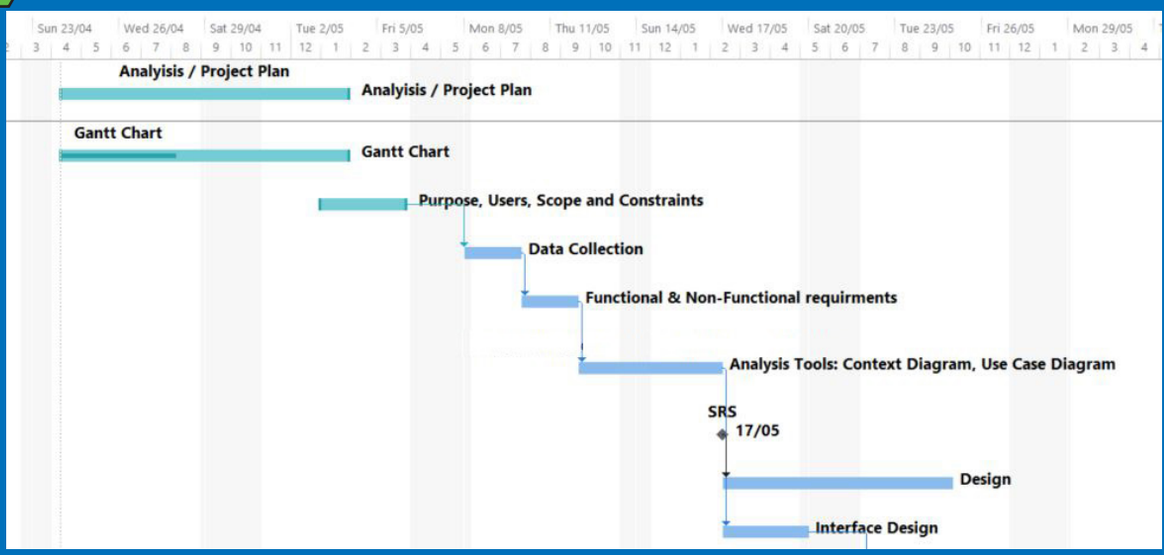
Each task listed in the table aligns with a row where the task be undertaken across the selected dates. You can see in figure 2.4 the Gantt Chart Task is displayed as an 8 day long bar, followed by the next task - "Purpose, Users, Scope and Constraints". Some bars are linked by arrows. These show the relationships between tasks and their predecessors illustrating the critical path from the first task to the final deadline. The project plan is highlighted with milestones, a key task that needs to be completed in the time-line but does not have a duration. These are indicated by a diamond on the Gantt chart. You can see the SRS Milestone located in the diagram below on the 17th May.

Students of project management often make their Gantt Chart and forget to use it throughout the life of the project. This is a mistake. The Gantt Chart is a living document that can be updated throughout the project and resubmitted for Criterion 8 of the SAT where students must demonstrate recording all adjustments to the initial plan during the progress of the entire project. Student should demonstrate a range of techniques of recording. Most project management software allows for the user to create an initial Gantt Chart and then record changes once the project begins. These edited versions of the chart can show the original plan along with the changes. To expand your techniques, keep a journal or log of your progress.

A good rule of thumb is to spend 5 minutes every day to write up the following:

- What you completed today.
- What worked well.
- What problems need to be solved.
- How did you solve your problems?
- Reflections on how you would do things differently in the future.

Fig 2.4 Gantt Chart



Software Requirements Specifications (SRS)

The Software Requirements Specification (SRS) describes the intended purpose of the software that is to be developed. It documents all aspects of the project such as the scope and constraints, the environment where it will operate and the functional and non-functional requirements. The SRS breaks down the problem into its components allowing for analysis to make way for the design of a solution.

A new information system is usually in response to a communication or efficiency issue arising in the operations of the organisation. To ensure the new system will meet the needs of the organisation it is important to research the requirements and to articulate each requirement in a detailed manner. To do this the developer needs to use data collected from all stakeholders to ensure the solution they develop meets the needs of everyone affected and it is aligned with the organisation's goals and objectives.

The Features of a Software Requirements Specification (SRS) document include:

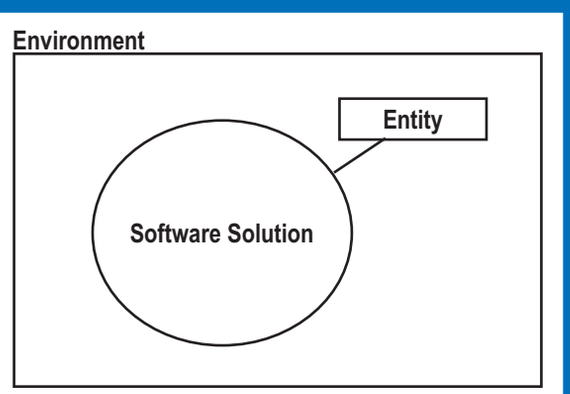
- A Table of Contents, as it will be a large document.
- Gantt Chart Project Plan. You can use GanttProject which is a free online resource.
- The Purpose of software solution clearly outlined - defining the functional requirements.
- Overview of Data Collection methods and justification – The data collection tools and results can be placed in the Appendix.
- The characteristics of the end users and how these will inform the non-functional requirements and usability constraints of the solution.
- The constraints of the project.
- The scope of the project.
- A description of the Operating Environment. This must include a Context Diagram.
- A full list of Functional Requirements clearly identified with ID. This must include Use Case Diagram and a Data Flow Diagram. These can be included in the Appendix and referred to within the functional requirements.
- A full list of Non-Functional Requirements with IDs. These can refer to the user characteristics and the environment.

Unified Modelling Language (UML) is a diagrammatic language that allows software developers to structure and visualise the designs of their solutions. We will look at three UML approaches: Context Diagram, Use Case Diagram and Data Flow Diagram.

Context Diagram

Context diagrams are tools used to depict the environment of the solution and any interfaces between the solutions, its users and any other entities such as external databases or organisations. The diagram is contained in a rectangle which represents the environment where the software will be used. (For example; An office, at home, in a school, in a vehicle etc.) The circle represents the whole software solution and the user is now called an Entity. An entity is a person or thing (it can also be an outside database). Context diagrams can be very useful when multiple users and outside organisations are connecting to the system you are building.

Fig 2.5 Context Diagram



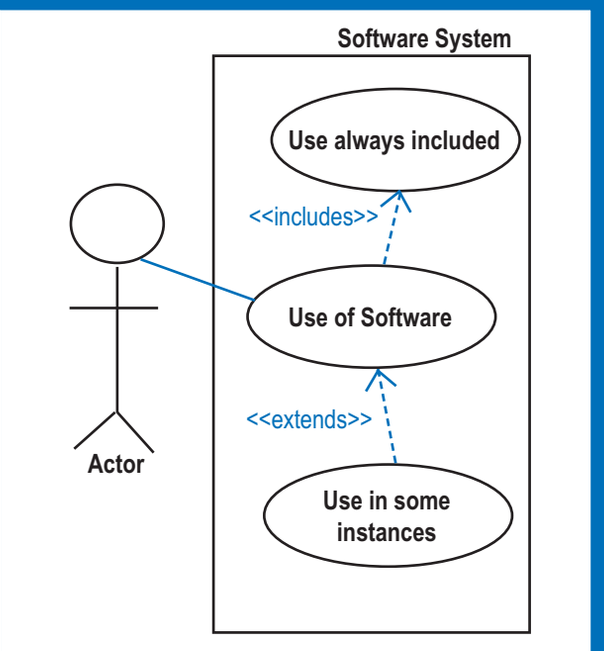
In figure 2.5 you can see the entire diagram is contained in a rectangle labelled 'Environment'. If we were planning to build an App for students to order lunches at the Canteen, the Environment would be 'School'. The Entity is also a rectangle and in the case of our canteen example this could be a student who is ordering their lunch. The arrow indicates the direction of data entering the system. Finally, the entire system is depicted as the circle. This could represent the Canteen App.

USE CASE DIAGRAMS

Use Case displays the way a software solution will be used, see figure 2.6. Rather than writing a complex algorithm to solve the problem, the first stage is to identify the system in terms of its users and what each "CASE" will be. Each CASE is how a user interacts with the solution.
You create a USE CASE in 6 STEPS:

1. Create the System boundary.
2. Identify the actors (the users roles).
3. Identify each CASE the user will interact with.
4. Create associations between Actors and CASES.
5. Add extra CASEs where there are more than one step required (includes).
6. Add extra CASEs where there may be an extra step involved on occasion (extends).

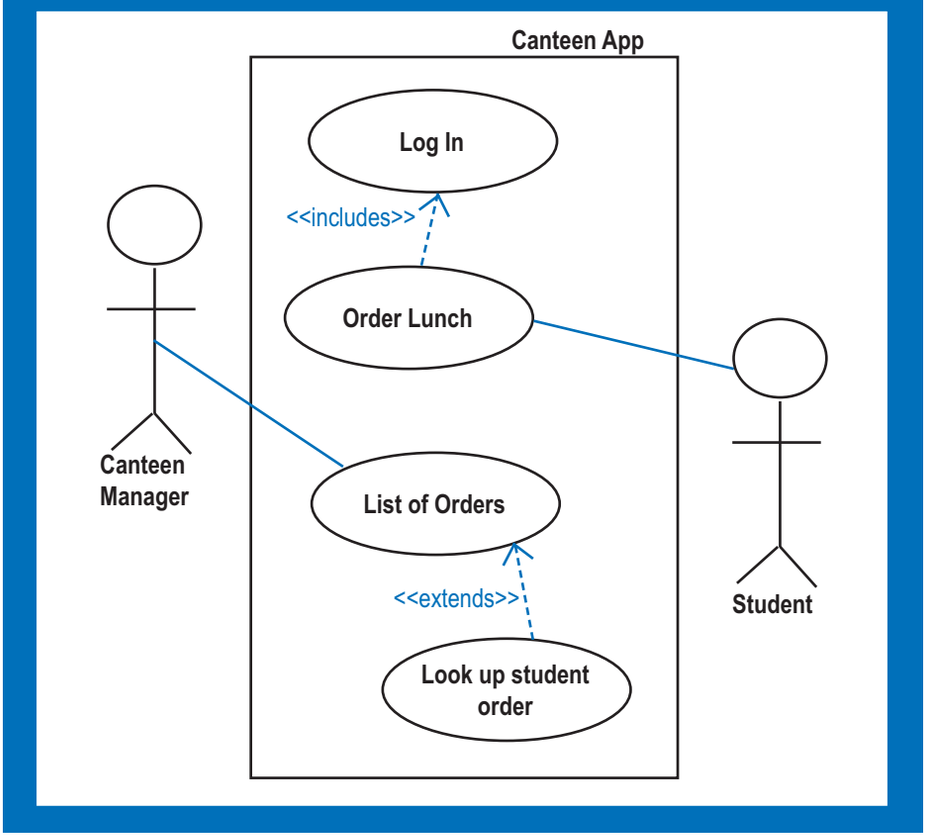
Fig 2.6 Use Case Diagram



Remember, when designing a Use Case Diagram, you need to think in terms of how the solution will be used. You can add 'log in' as an <includes> use case, but remember logging in is not the main purpose for using the software. Consider each user and what they need to use the software for. In figure 2.7 you can see an example Use Case Diagram for our Canteen App. Both of the Actors are named after their roles in relation to the software: Canteen Manager and Student.

For simplicity, the login for the Manager is omitted so you can clearly see the difference between includes and extends. Each time the student uses the app to order lunch they must login, so includes is added to the use case. The Canteen Manager mainly uses the application to get the full list of orders, but occasionally they need to look up a student's individual order in case their order is missing or there is an error. The use case of "Look Up Student Order", then, is an extends.

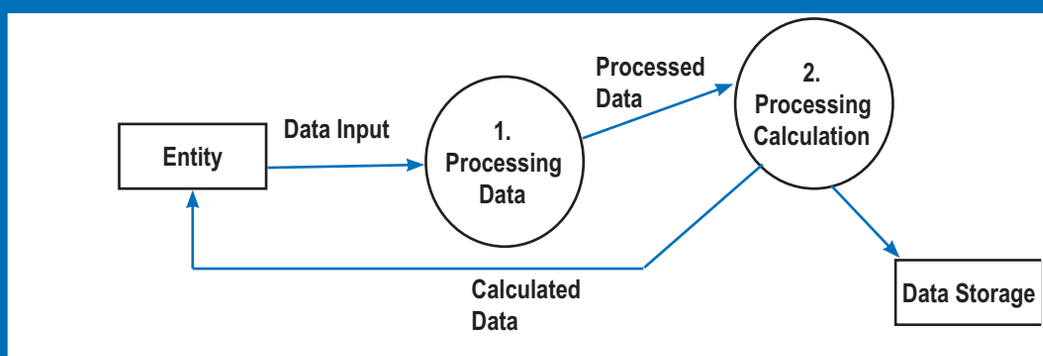
Fig 2.7 Use Case Diagram - Canteen App



DATA FLOW DIAGRAM

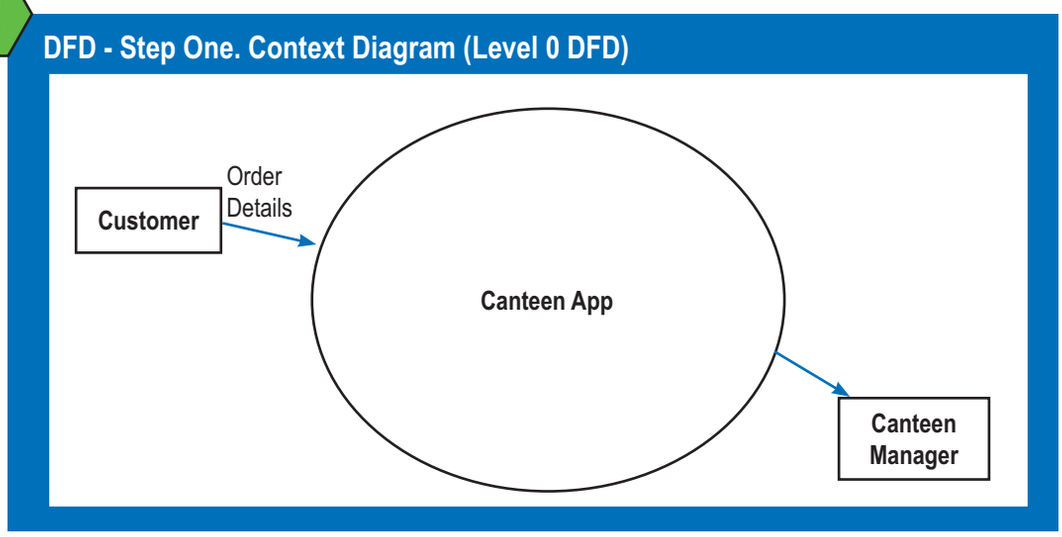
A Data Flow Diagram (DFD) shows how data will flow through the software solution. It contains four elements: Processes, Entities, Data Flows and Data Stores. Processes are depicted and numbered in circles and they change the data that flows into them, producing different data that has been processed. Entities are people or other organisations that enter or receive data from the system. Data flows are arrows labelled with short descriptions of the data showing origin and destination within the system. Finally, data stores are files or databases that store the data where it can be retrieved. In figure 2. 8 you can see how these four elements are put together

Fig 2.8 Data Flow Diagram



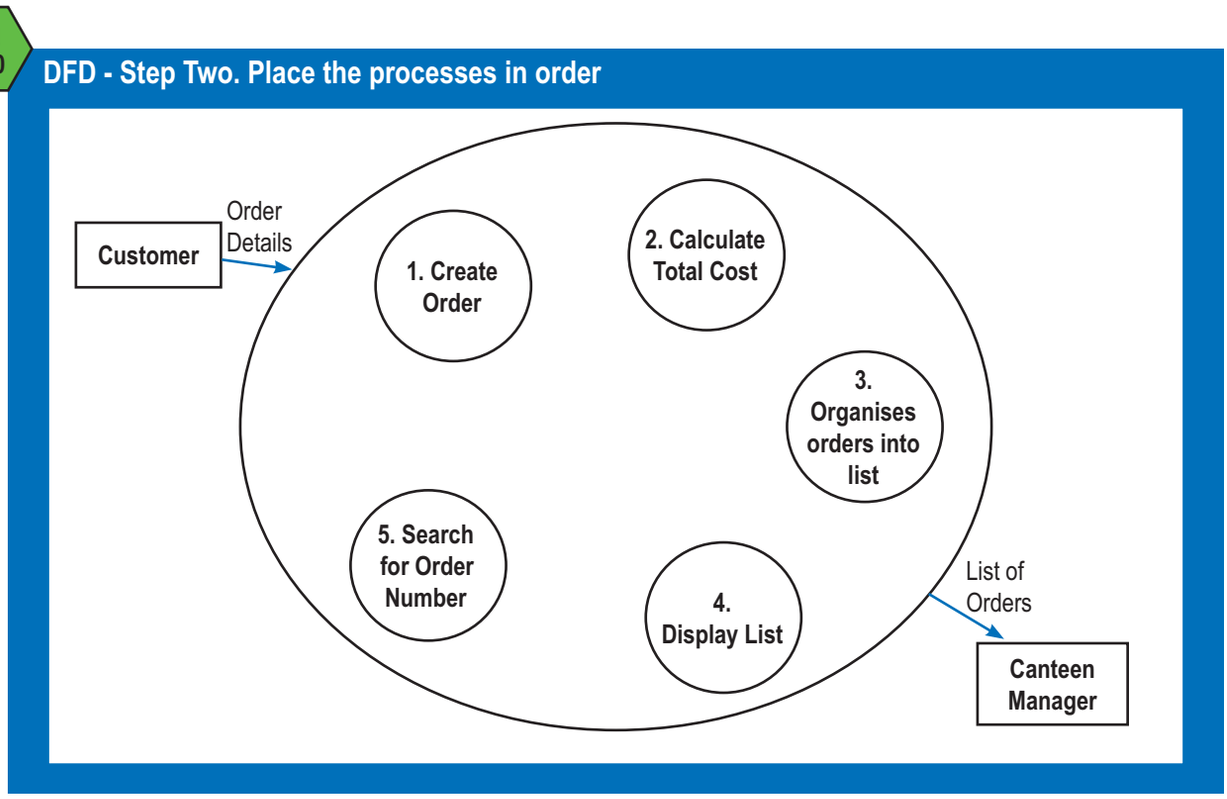
A Context Diagram is a high level DFD. A good method of DFD development is to start with the Content diagram and add processes within the system circle. In figure 2.9, you can see a Context Diagram for the Canteen App.

Fig 2.9



The next stage is to break up the Circle that represents the Canteen App into separate processes that change the input from "Lunch Order" to "Order List". Once the processes are identified, we add the data that moves between the processes, and include data stores for saving and retrieving data. The processes are numbered and should be read in a clockwise fashion starting from the top left.

Fig 2.10

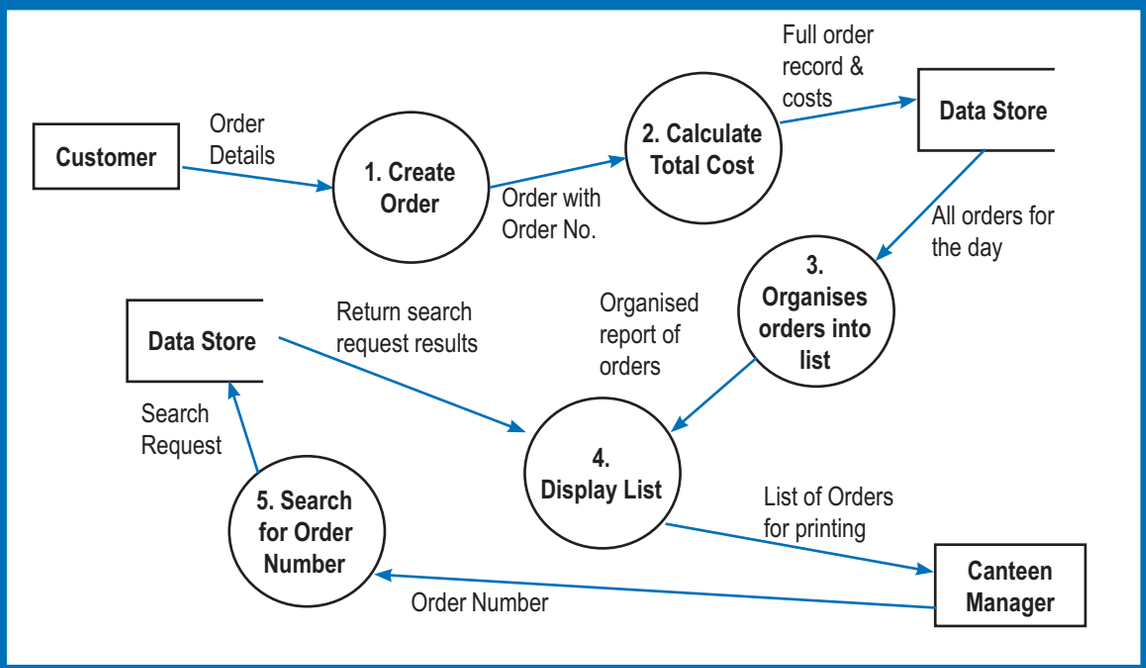


In figure 2.10 you can see how the application was broken down into five processes and placed in a clockwise order. At this stage it is still in the shape of a Context Diagram, otherwise known as a 0 Level DFD.

To complete the DFD the data flow arrows need to be added. Data stores can be added at any point in the diagram but it is important to show retrieval requests to data stores when data is being searched or accessed. Once these factors are added, you can see in figure 2.11 a completed DFD.

Fig 2.11

DFD - Step Three. Add the data flow and data stores.



A DFD should be kept simple and easy to read and limited to about 8 or less key processes that represent the solution. There are a few rules that determine an effective DFD.

1. Reading the diagram should start at the top left and move to the bottom right.
2. Data can't flow from entity to entity. (Why develop software if entities can just speak or email?)
3. Data from a Data Store must go through a process before going to another Data Store
4. A process must have data in and different data out.
5. Rejected data can be sent out of the diagram.
6. Keep the DFD to 8 processes at the 2nd level - you can add more if you need extra levels.
7. Use verbs in your process circles such as: calculate, edit, delete, sort, add.

Important Tip

EVALUATION OF PROJECT MANAGEMENT

We may well be at the beginning of our SAT project - however it is crucial that you keep records of all your decisions, ideas and changes in one location. Keeping a blog records the date of your logged entry. This will assist you in the Evaluation stage. You will be required to report on:

- reflections on how your analysis tools fed into the design process,
- evidence that you used idea development techniques (pages 40-41) to show how you solved design and development problems,
- logs of decision processes during the development stage.

If you keep a blog with a summary of your progress everyday, including different versions of your analysis and design tools, this will assist you enormously when creating your final criteria for the SAT. You will be asked to reflect on how you managed the project and the decisions you made along the way. Keeping records will assist you in this.

Design

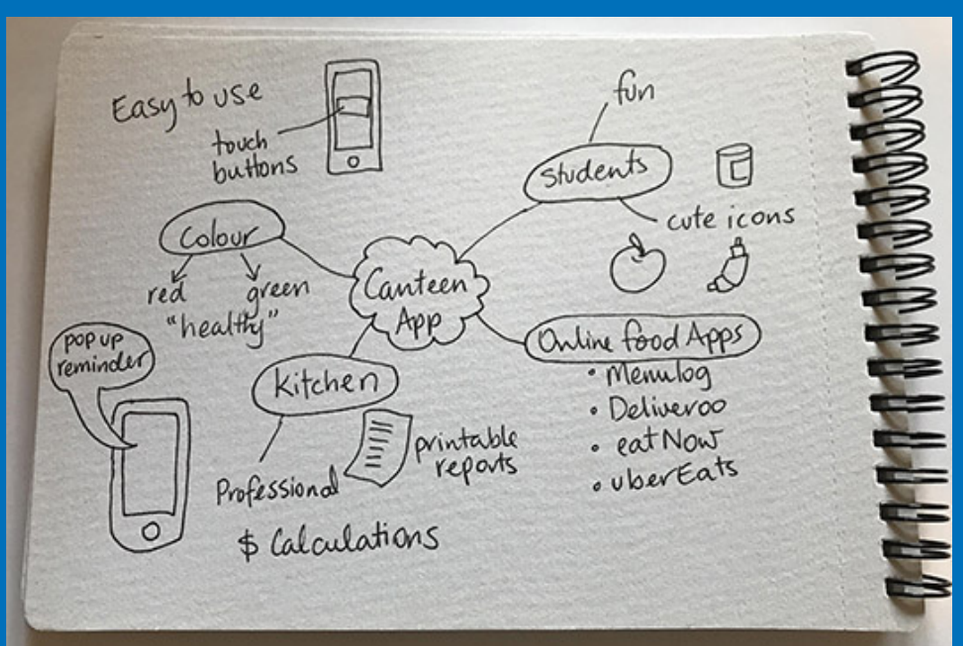
Once you have analysed the problem, investigated the requirements for the software you are about to build and you have produced an SRS, it is time to start designing a solution. You may already have ideas as to what your design will look like, and often students are keen to get started building based on their initial ideas. It is a requirement that you produce 2 - 3 feasible designs for your client to choose from, therefore, you will need to come up with more than one idea.

Some people find it easier to come up with ideas than others. There are a number of tried and true methods for generating solution design ideas.

Brainstorming

This is a method of generating as many ideas as possible. There is no editing out of crazy, infeasible suggestions because these can lead to other ideas that may well be more pertinent to the problem. This often works better with a team. See figure 2.12 for an example of this visual tool.

Fig 2.12 Brainstorming



Mind-mapping

This is a graphical technique where related concepts and key phrases are linked together. The object is to make sense of a wide range of ideas and to group them into a feasible strategy.

Role playing

This is an opportunity for the developers to put themselves in the place of end users to investigate what solutions would best suit different scenarios.

Reverse thinking

Instead of using the logical, normal manner of trying to solve a problem, you reverse it and think about opposite ideas. For example: 'how can I protect the customer data?' can change into 'how do I ensure data is able to be corrupted or stolen?' Most people find it easier to produce ideas for the 'negative challenge' simply because it is much more fun. What this can do is highlight all the issues that require attention in the software problem.

Attribute Listing

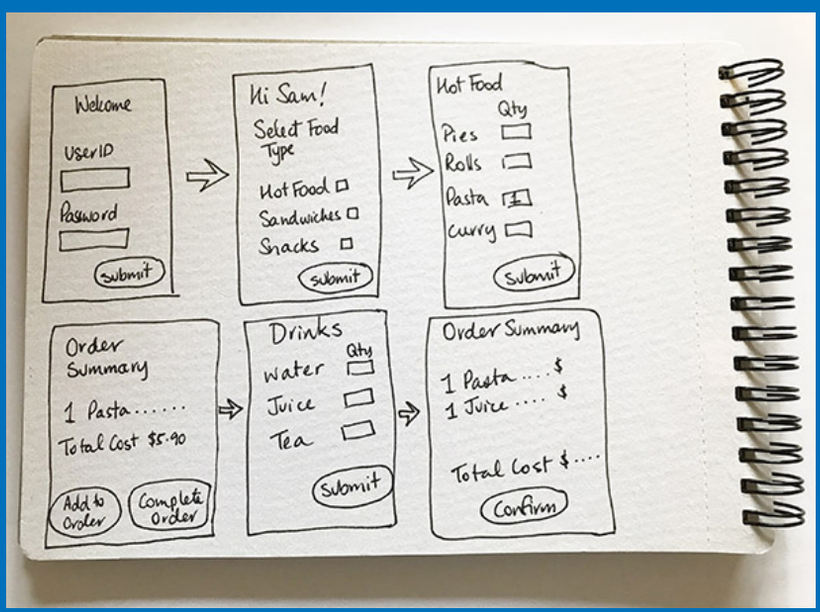
Attribute listing is an approach that investigates a system and identifies areas of improvement. To determine the aspects of the old system that need improvement, the system is broken into sections. Each section is described through a list of attributes. This is a great method of determining functional and non-functional requirements.

Storyboarding

This is also a graphical technique where the developer creates mock ups for each interface as the user moves through the system. This provides the developers an opportunity to reflect on what is required and in what order each task needs to take place. In figure 2.13 you can see how the Canteen App is broken down into different screen interfaces as the customer selects what they want to add to their order.

Fig 2.13

Storyboarding



Designing a Solution

When you have generated some ideas the next step is to use design tools in order to prepare a Design Folio. Your Design Folio supports the development stage. The folio requires the developer to produce 2 - 3 possible solutions. Students need to create mock-ups and algorithms that are distinctly different and feasible. You will not get away with creating one dud solution and one feasible solution. They both need to be fully worked out and able to be worked through to completion.

You need to produce a feasible justification for your final design and this must be complete in detail. It is composed of four deliverables:

1. Mock-Ups of Interface fully annotated with colour and other design elements and functionality.
2. Data Dictionary of all variables.
3. Object Description Table.
4. Algorithm in pseudocode.

You will find descriptions of these in Chapter 1 and examples on the support website (vicfarrell.com.au). Ideally you need to have at least two fully worked designs to show your client. This will give your client some feedback as to what is possible and they may even ask for you to adapt one of your designs or combine aspects of all them.

EVALUATION OF DESIGNS

Students, when designing their solution, often fall foul of developing a single idea and refusing to let it go. There is a science to developing a design that is both efficient and effective for the given objective and list of requirements. It is a lengthy process and requires great attention to detail in how the software will work.

Evaluating Efficiency

There are three ways to evaluate efficiency of a software solution design:

1. Speed of Processing - does it complete the task without an unfeasible delay?
2. Functionality - how well does the solution do what it was designed to do?
3. Cost of File Manipulation - how many times are files manipulated? Minimum manipulation increases efficiency?

Evaluating Effectiveness

There are many ways to evaluate effectiveness of a software solution design:

Completeness - Is all the output complete? Does the user need anything outside of the system to do the task?

Attractiveness - Have the design elements created a solution that is appealing and clearly related to the system's function?

Clarity/Readability - Is it clear enough for users to read to complete a task using the system?

Accuracy - Is the output from the system always correct?

Accessibility - Is the interface easy to use for all the users? The design of the interface needs to pay attention to the characteristics and limitations of end users.

Timeliness - Is the data entry processing and the production of output completed in a time frame that allows practical use of the information?

Communication of Message - Through affordance loaded interface elements, is the software design clearly communicating to the user?

Relevance - Is the data that has been included in the output relevant to the needs of the user?

Usability - Given the user characteristics, how easy is the solution to use?

EFFICIENCY

"A measure of how much time, cost and effort is applied to achieve intended results. Measures of efficiency in a solution could include the cost of file manipulation, its functionality and the speed of processing. Measures of efficiency in a network include its productivity, processing time, operational costs and level of automation."

VCE Study Guide 2020

EFFECTIVENESS

"A measure of how well a solution, information management strategy or a network functions and whether each achieves its intended results. Measures of effectiveness in a solution include accessibility, accuracy, attractiveness, clarity, communication of message, completeness, readability, relevance, timeliness, and usability. Measures of effectiveness of an information management strategy include currency of files, ease of retrieval, integrity of data and security. Measures of effective networks include maintainability, reliability and the security of data during storage and transmission"

VCE Study Guide 2020

It's important to be able to remember all the measures for evaluating effectiveness so here is a mnemonic to help you remember them, see figure 2.14.

Fig 2.14

Measures of Effectiveness Mnemonic

R	Clarity/Readability	Are
U	Useability	You
A	Attractiveness	A
C	Completeness	CAR 
A	Accessibility	
R	Relevance	
C	Communication of Message	CAT 
A	Accuracy	
T	Timeliness	

Factors that Influence Design

Usability

Usability is a measure of how usable the solution will be to the people it is designed for. The users define whether the solution is usable or not, so it's important to look at the user's needs and expectations of the solution. If a key function for the solution is to display a calculated variable, it needs to be clear how that variable can be accessed by the user. This can be done with clear labels, the use of affordance (using objects and icons that are generally recognisable) and the display of data in a format that the end user can actually use.

Affordance

Affordance is a property of a user interface that allows the user to easily and naturally take the correct steps to use the software. Graphical User Interfaces use a collection of recognisable images that imply to the user how they should be used. For example, a pull-down menu has a small triangle that shows the user to click to reveal the options available. An input textbox has a bevelled edge to imply something can be added to the space. Buttons are bevelled out to imply they can be "pressed" to activate or execute code.

Security

When sensitive data is being stored, processed, input and exported, the security of the data needs to be kept in mind when designing a solution. Perhaps a login and password is required to limit the access to the data to authorised users. When saving data to a file or database, measures to keep the data secure need to be put in place, including how the software solution accesses the data when required.

Interoperability

If the data from one system is to be used by another system this is an example of interoperability. For example, when you tweet a comment and a photo on Twitter, the interoperability of Twitter and Facebook allows that tweet to be accessed and displayed in both social media sites. In an SD SAT software solution, a student may use an XML file type to allow access to the data in Excel, your Visual Basic program and MS Access. Application Programming Interface (API) allows for developers to create and draw from a software library so all functions use the same platform and can interact. An example might be a fully web-based software solution such as a school moodle or portal. This solution provides an interface and processing of a wide range of purposes and end users:

- administration entering attendance
- head teaching staff managing reports
- teachers posting teaching and learning resources and content
- students reading results and timetable information.

All sections could have been built independently, but instead are part of a large over-arching framework on the same platform so data can move easily between each component.

Marketability

If a software solution is marketable, it means the solution is seen as a desired commodity or simply as an attractive product to potential users or clients. This can affect the design of the solution when there are desired integrated elements with other commonly used software already in place. Any software solution that saves the user time makes it highly marketable.

The Characteristics of an Efficient and Effective User Experiences (UX)

User experience (UX) design is the process of creating software solutions that provide meaningful and relevant experiences to users. The user experience is very important to the successful design of your solution. If the user finds the software frustrating, confusing or difficult to use this will probably be due to poor user interface design. The interface needs to be efficient and effective to allow the user to get the output they need easily an intuitively.

Clear and Familiar Design Elements

User interfaces use a range of different object elements such as buttons, combo-boxes (pull-down menus), input boxes, radio buttons, check boxes etc., which are familiar to all users who have the most basic computer experience. This familiarity with interface elements such as these, allow the interface designer to rely on the affordance the elements bring with them. Each year new students come to high school with fewer and fewer skills with actual computers. This is due to having spent most of their time with touch screen mobile devices. This is becoming an important consideration in interface design to allow for those with limited experience with WIMP interfaces (Window, Icon, Mouse, Pull-down menu). Sometimes menu buttons or icons need to provide hover information. This is text that describes the purpose of an element when the cursor hovers over it. Using icons that are familiar to the user is important. Traditionally the icon of a floppy disk is used to denote saving, however most students entering high school in 2021 have never seen a floppy disk. It is important to keep these things in mind when designing icons and buttons for familiarity and clarity. The save icon has transformed into a down-faced arrow. Google Docs has lead this change.

Responsive and Robust to User Input

A good interface needs to respond to the user. A common experience is when using an online store. If you "add" an item to your "cart" the interface responds by displaying a "1" next to the cart icon. This is providing feedback to the user that they have, indeed, added an item to the cart. Without that feedback, the user could unknowingly add an item many times. An interface needs to also be ready for incorrect, misplaced, missing or unexpected data entry. If a user tries to progress through an online form without filling in all the fields, a robust interface would provide feedback such as "You must enter data into the required field". If they enter their date of birth into the name field the interface should be able to detect that the data is not appropriate and provide feedback to the user. This is called validation. There are three types of validation that assist in making a solution robust: existence checking, range checking and type checking.

Consistent in Design

Consistency is important to allow the user to intuitively know how to navigate and use the software solution. It is important to locate key tools or elements (such as an exit or back button) in the same location on the screen for each screen. Use the same terminology throughout (example - only use "Exit" or only use "Quit", don't use a combination of both - this can be confusing.)

Efficient

Efficiency is measured in the number of actions a user is required to obtain the outcome they need. The fewer number of clicks required to access a screen the more efficient it is to use. When the same data needs to be entered in over and over again, this negatively affects efficiency. An efficient interface has functions that allow data entry, access and editing in the fewest number of actions by the user.

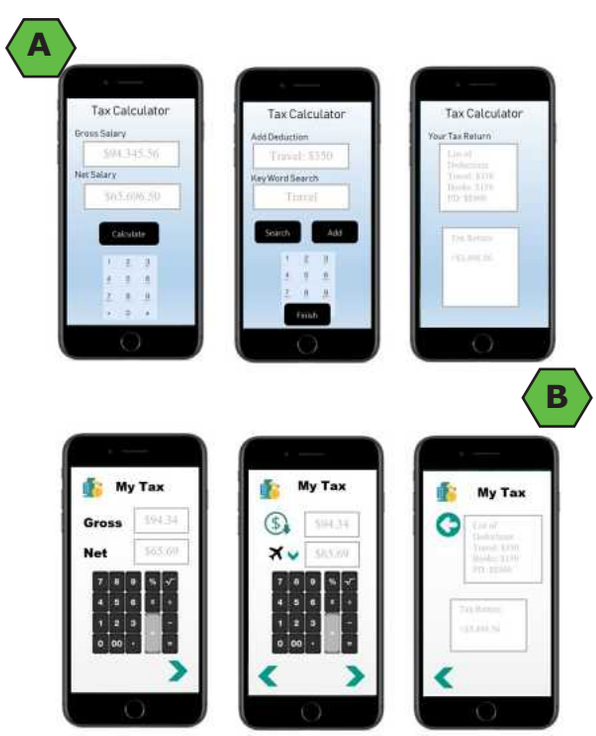
Scalable and Attractive

Your software solution may be accessed on different types of devices with varied screen sizes. Although some software may be required to be optimised for a particular screen size, there is often an expectation that the software can be responsive to the needs of different devices. Examples include providing scroll bars, scalable forms, zoom-able text and other useful features that allow smaller or larger screens to display the interface effectively for the user. An attractive interface is defined by pleasing colour choices, font styles and sizes, use of white space and layout design. A pleasing interface makes a software solution a more pleasant experience which enhances the usability of the software.

Below are examples compared:

Option A

This Tax Calculator relies on text to communicate the purpose of the interface objects. This would reduce the time required to learn how to use the application. On the other hand, the interface is not very attractive. It is not clear how to navigate between screens which could be frustrating for the user.



Option B

This interface uses a lot of graphic elements that make the interface attractive. The navigation is clearer with the use of forward and back arrows at the bottom of the screen, however, the use of icons in some instances are not clear. The deductions icon is a dollar sign with a downward arrow. This may be confusing to the user.

Writing Evaluation Criteria

The second half of the Design Stage requires that Evaluation Criteria is written. Evaluation Criteria identify how well the fully tested software performs for users.
For example:

Requirement: "Software must calculate the Tax Return."
Evaluation Criteria: "The software allows the user to calculate their tax return more efficiently than using a manual calculator by taking less time and effort."

Requirement: "Software must easy to use."
Evaluation Criteria: "Users rate the interface as attractive and easy to use on the Likert scale"

The key approach to writing these criteria is to revisit the functional and non-functional requirements. For each requirement, write an evaluation criterion. An example in the table below shows criteria written in blue below the requirement. It would be ideal to identify the strategies to evaluate the effectiveness and efficiency of the requirement too at this stage. This will make it easier to put together your Evaluation Report at the end of the SAT. Identify how each requirement can be evaluated using a strategy such as:

- observing the user interactive with the software solution,
- providing the user with a list of tasks without a tutorial,
- interviewing the user after using the solution.

No.	Requirement	Strategies	Evaluation Criteria
Evaluation Criteria 01	(FR01) User can enter Gross and Net Income into Tax Calculator.	• Observe client attempt to enter data. Note any issues. • Provide a table of data to the end user to enter. Note any issues. • Interview end user about the user friendliness of the interface.	(EC01.1)The tax calculator enters data if the Net < Gross input. (EC01.2) The Tax Calculator provides useful feedback if input is incorrect data type or size.
Evaluation Criteria 02	(FR02) User can choose from a range of deduction types and enter their deduction amount.	• Provide a table of data to an end user to enter for deductions. • Interview end user about the user friendliness of entering deductions to the software.	(EC02.1)The user is able to select the correct icon per deduction, without scrolling or asking. (EC02.2) The Tax Calculator provides useful feedback if input is incorrect data type or size.
Evaluation Criteria 03	(FR03) The user is able to select and enter multiple deduction types which are added to the total deductions.	• Provide a table of data to an end user to enter for deductions. • Interview end user about the user friendliness of entering deductions to the software.	(EC03.1)The user is able to select the correct icon per deduction, without scrolling or asking. (EC03.2) The Tax Calculator correctly adds all deductions together under the correct types. (EC03.3) The Tax Calculator provides useful feedback if input is incorrect data type or size.
Evaluation Criteria 04	(FR04) The Tax Calculator correctly calculates the tax return. Return = Gross - Deductions.	• Check output against prepared data	(EC04.1) The Tax Calculator correctly calculates the return with two decimal points. (EC04.2) The Tax Calculator provides useful feedback if he Return is calculated to be outside of the following parameters: • Return > Gross • Return < Gross - Net
Evaluation Criteria 05	(NFR01) The Tax Calculator is an attractive design	• Interview end users about the attractiveness of he interface design.	(EC05.1) The end users interviewed return and average of 4.5 on the Likert scale for the interface design.

Evaluation Criteria will be used in the Evaluation Stage after the software has been developed. It is important that the evaluation criteria is developed at the Design stage to ensure that the focus on the requirements is not lost during development. Evaluation criteria investigates the efficiency and effectiveness of the solution you build. Let's revisit these concepts in terms of what you will be investigating. Let's look at the examples for our Tax Calculator in the following table.

Efficiency	Effectiveness
TIME: The Tax Calculator allows the user to complete their tax return in less time than using other manual calculators.	**ACCESSIBILITY:** The Tax Calculator is able to be used by a wide range of end users with various needs.
COST: The use of Tax Calculator is cost effective in that it is cheaper to use than paying an accountant.	**ACCURACY:** It is easy for the end user to accurately enter data into the Tax Calculator.
EFFORT: The end user expends less effort to do their Tax Return using the Tax Calculator app that by using a manual calculator. The user does not have to make as many decisions, organise their data, press as many buttons, do many mental calculations.	**ATTRACTIVENESS:** The Tax Calculator interface has been rated an average of 4.5 on the Likert scale for attractiveness.
LEVEL OF AUTOMATION: The Tax Calculator automatically organises deductions by type at the click of a button and does not expect the user to total each type of deduction for input.	**CLARITY:** The Tax Calculator provides clear instructions through the interface design which ensure the user knows where to enter each data item.
COST OF FILE MANIPULATION: The Tax Calculator creates a new file for each tax return completed. No efforts are made to edit files already stored.	**COMMUNICATION OF MESSAGE:** The Tax Calculator provides the how the data will be processed and the what the outcome data represents so the user knows what to do with the data presented as output.
PRODUCTIVITY: The user gets more work completed using the Tax Calculator software solution than with other solutions.	**COMPLETENESS:** The Tax Calculator eaves nothing left for the user to do to complete the process of preparing a tax return. The software does not have any areas that need improving.
OPERATIONAL COSTS: The Tax Calculator may need to be updated in the future, for example tax bracket rates may change. If the amplification needs to be regularly updated, this could be an automated option, or the tax bracket information could be accessed online for the latest data.	**TIMELINESS:** The Tax Calculator may need to be updated with new data provided by government policies so that the processing is completely correct. Updates automate before the end of the financial year when users need to complete their tax.
	RELEVANCE: All data used and produced through the process is relevant to the user's needs.
	READABILITY: The text and icons are clear and easy to read.
	USABILITY: The Tax Calculator

Strategies & Collection Tools for Evaluation

Once the development stage is complete and the testing is complete, you will need to prepare for evaluation by organising strategies for investigating efficiency and effectiveness of the completed system. It is a good idea to complete a plan at the design stage so you will have the resources to do the evaluation and usability testing when required. You will need both strategies and collection tools.

A collection tool is a data collection method such as:

a) a list of interview questions,
b) an observation sheet that allow the observer to tick off criteria when it is observed,
c) a list of input data,
d) a list of instructions,
e) survey questions that allow for the likert scale (1- 5 where 1 = dislike & 5 = like).

A strategy is the method you use to collect the data with your collection tool such as:

1. Provide a list of instructions (d) to a number of end users and observe their success to complete the tasks using an observation sheet (b) to check against evaluation criteria.

2. After observing end users completing the input of provided data (c) the evaluator can interview the users from a list of prepared questions (a).

3. Many end users are provide the solution and surveyed (e).

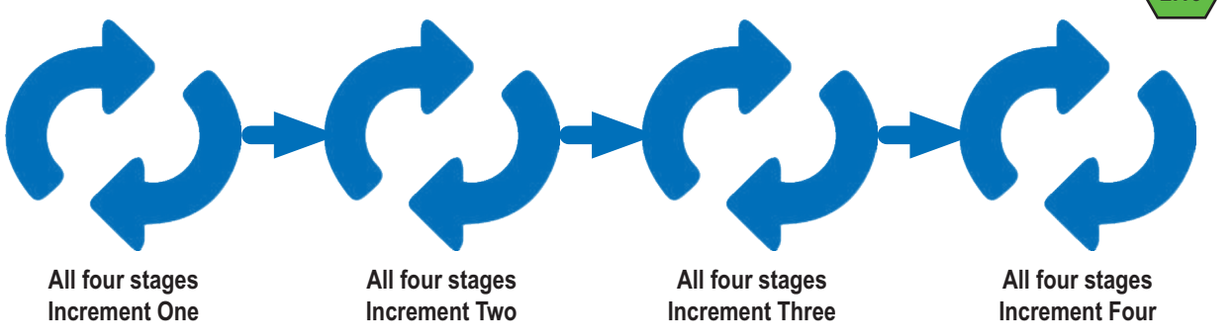
Development Models

There are a wide range of software development models. Often developers choose a particular model which determines the approach they take to solving the software problem throughout the life of the project. Software development models are the methodologies that are used for the development of the project depending on the project's aims and goals. There are many development life cycle models that have been developed in order to specify the various stages of the process and the order in which they are carried out. The study design focuses on three development models: Agile, Waterfall and Spiral.

Agile Development Model

The Agile model is the most commonly used approach to software development in recent times. Another name for the agile model is "incremental model". Software is developed in incremental or rapid cycles. This results in small incremental releases with each release building on previous function Each release is thoroughly tested to ensure software quality is maintained. It is used for time critical applications. This model provides freedom to make changes to the solution throughout the project development schedule. New changes can be implemented at very little cost because of the frequency of each new increment. In the Figure 2.15 you can see that the full problem solving methodology is a rotation until the solution meets all requirements. This approach is often taken when experimenting with new solutions, you complete each stage before moving to the next.

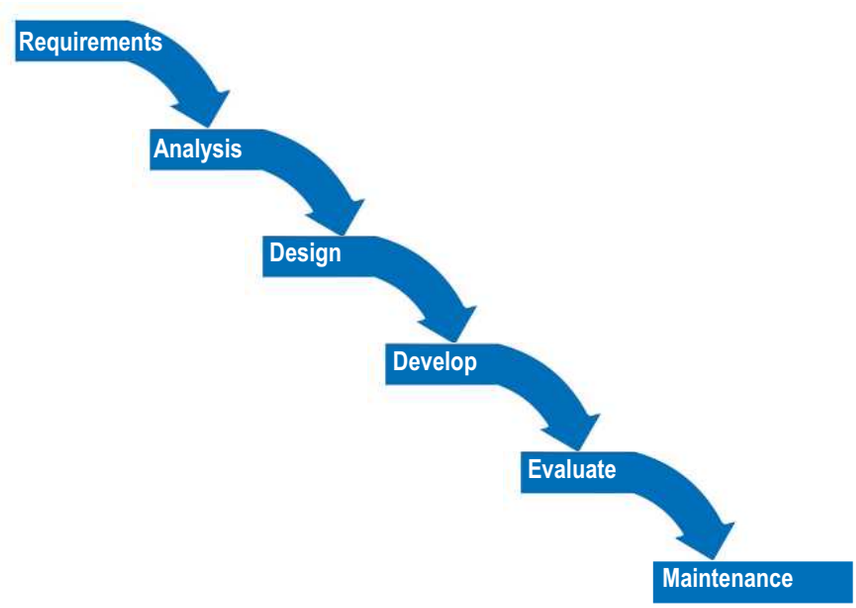
Fig 2.15



The Waterfall Model

The Waterfall Model was the first Process Model to be used extensively by software developers, and given the limited time provided to students and the nature in which the SAT is assessed, it is effectively the model most students will follow. It is sometimes called "linear-sequential life cycle" model and it is illustrated in figure 2.16. It is very simple to understand and use. In a waterfall model, each phase must be completed fully before the next phase can begin.

Fig 2.16

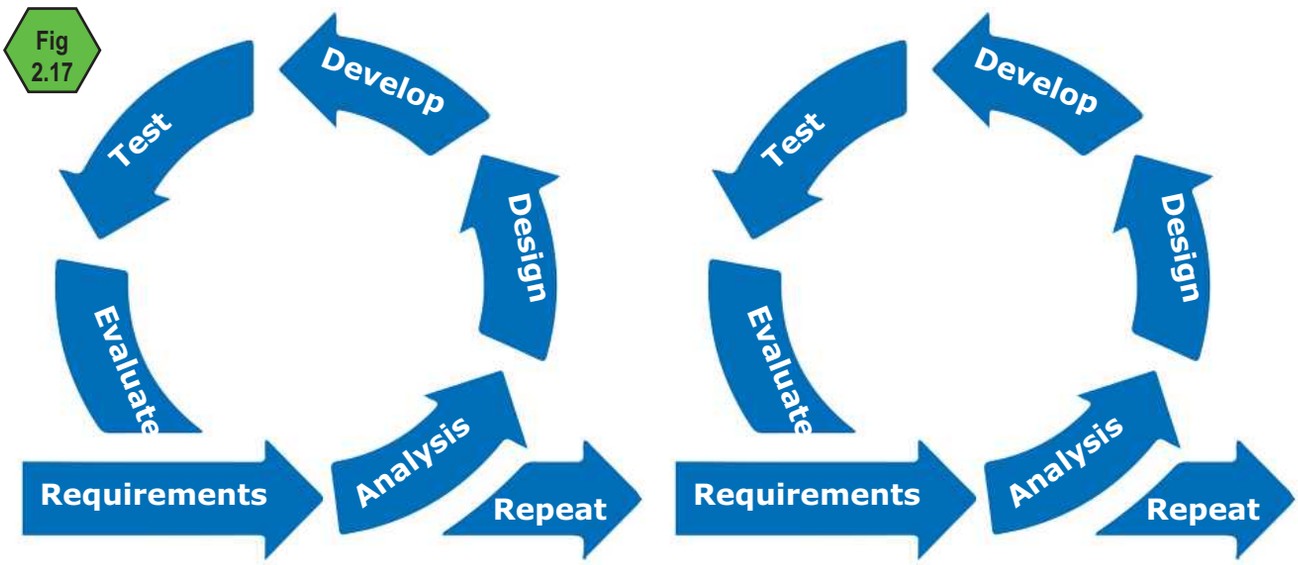


The Spiral Model

The Spiral model . The spiral model illustrated in figure 2.17 focuses more on risk analysis. It has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). In the baseline spiral, starting in the planning phase, requirements are gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. 1. Planning phase is where requirements are gathered and a 'System Requirement Specification' is developed. 2. In the risk analysis phase, a process is undertaken to identify risks of the system failing and focus on alternate solutions. A prototype is produced at the end of the risk analysis phase. If any failures are found during the risk analysis, then alternate solutions are suggested and implemented. 3. In the engineering phase, the software is developed and tested. 4. The evaluation phase allows the client to evaluate the solution currently produced before the project continues to the next spiral.

Each spiral creates the full solution! Each repeated solution refines the solution. It is not done in increments like the Agile Model.

Fig 2.17



Goals & Objectives of Organisations

An organisation will have goals and objectives that determine their operation. These feed into the requirements of any new software solution.

A goal is a statement that the organisation will strive for. Example: "To provide excellent customer service". A goal may not have tangible or measurable results. It is intended to provide direction for the operations of the organisation to achieve some general improved state. Goals assist with future planning and are purely aspirational.
An objective is a measurable aspect of the organisations goals. Example: "We will expand the 2020 Network by 10% in the next 12 months".

When an information system is developed, it is important to define goals and objectives for its operation. Goals for an information system could include:

- To enhance the customer experience online.
- To improve the management and tracking of stock.
- To report on student progress through an education program.

Information system objectives are much more specific. They identify exactly what the system will do in a measurable and tangible way.

Objectives for an information system could include:

- To provide immediate customer feedback on an e-commerce site
- To display calculated discounts.
- To increase the number of users by 20%.

LEGAL REQUIREMENTS

The steadily increasing production of data and information has lead to an evolution of traditional laws to ensure that the individuals are able to live a free and private life and to protect the authors of data, software, information systems and information from having their work stolen.

Privacy Legislation

With the onset of electronic media, social media sites and increased digitisation of all aspects of regular life, such as paying bills, banking, education and training and e-commerce. We are placing our personal details in the hands of digital information systems. Keeping personal details private has been the task of The Privacy Act since 1988 in Australia. Since then, as technology has expanded, there has been a Privacy Amendment - Enhancing Privacy Protection in 2012 and the Privacy and Data Protection Act in 2014.

There are 13 Privacy principles that relate to a range of relevant factors such as:

- the open and transparent management of personal information including having a privacy policy
- an individual having the option of transacting anonymously or using a pseudonym where practicable
- the collection of solicited personal information and receipt of unsolicited personal information including giving notice about collection
- how personal information can be used and disclosed (including overseas)
- maintaining the quality of personal information
- keeping personal information secure
- right of individuals to access and correct their personal information.

Some personal information can be sensitive and is required to be kept private such as:

- health (including predictive genetic information)
- racial or ethnic origin
- political opinions
- membership of a political association, professional or trade association or trade union
- religious beliefs or affiliations
- philosophical beliefs
- sexual orientation or practices
- criminal record
- biometric information that is to be used for certain purposes
- biometric templates.

The Australian Privacy Principles

The Australian Privacy Principles are outlined by the Office of the Australian Information Commissioner within the Privacy act and can be found on the website: www.oaic.gov.au. It is essential that all software developers comply with each and every principle listed below.

Australian Privacy Principle 1 — open and transparent management of personal information
Australian Privacy Principle 2 — anonymity and pseudonymity
Australian Privacy Principle 3 — collection of solicited personal information
Australian Privacy Principle 4 — dealing with unsolicited personal information
Australian Privacy Principle 5 — notification of the collection of personal information
Australian Privacy Principle 6 — use or disclosure of personal information
Australian Privacy Principle 7 — direct marketing
Australian Privacy Principle 8 — cross-border disclosure of personal information
Australian Privacy Principle 9 — adoption, use or disclosure of government related identifiers
Australian Privacy Principle 10 — quality of personal information
Australian Privacy Principle 11 — security of personal information
Australian Privacy Principle 12 — access to personal information
Australian Privacy Principle 13 — correction of personal information

The SPAM Act

The SPAM Act 2003 addresses the emerging issue of unsolicited emails and text messages from organisations attempting to coax potential customers to purchase goods and services. Before the Act, email in-boxes filled with unsolicited emails. Since this bill was enacted, organisations that wish to send out emails are required to include an opt-out or discontinue option for those receiving them. Organisations are now required to get permission from email owners before sending promotional material.

The Health Records Act

The Health Records Act 2001 covers the management of data by health service organisations to protect the medical data and information of patients. The Health Privacy Principles cover: the collection of data, use and disclosure of data, quality of data, security of data, openness of management policies, access to data to edit correctness, the use of unique identifiers, maintaining anonymity, the transportation of data across borders, the collection of sensitive information, and finally the transfer of data regarding health professional services from one practice to another.

Charter of Human Rights and Responsibilities Act

The Charter of Human Rights and Responsibilities Act 2006 (Victoria) and 2012 (Australia) was enacted to protect the privacy and freedom rights of Australian individuals and to govern the responsibilities of Australians and their organisations to ensure they do not encroach on those rights. Australians have a right to:

- maintain their privacy and reputation,
- express freedom of expression, religion, thought and belief.

Copyright Act

The Copyright Act 1968 protects the ownership of creative writing, software, photography, artwork, design elements, sound recordings, compositions and other information based products.

As information-based works became digitised, they became increasingly easier to copy and distribute. To protect the ownership of TV programs, movies, music and other digital information products, the Copyright Amendment Act in 2006 was enacted to stop the illegal resale of domestic and international copyright material. Exceptions were made for "fair dealing" in educational contexts and for making copies for personal use on devices such as MP3 players.

Open Source Software is software with source code that anyone can inspect, modify, and enhance. The original authors of Proprietary software are the only ones who can legally copy, inspect, and alter that software. To protect the authors in this situation, a licence agreement is written up. A licence is a legal agreement between two entities such as an organisation and its customers. Different licences allow for different access and distribution. An organisation called Creative Commons developed a few licences that protect developers and content makers:

> **Attribution:** "This applies to every Creative Commons work. Whenever a work is copied or redistributed under a Creative Commons licence, the original creator (and any other nominated parties) must be credited and the source linked to."
>
> **Non-Commercial:** "Lets others copy, distribute, display and perform the work for non-commercial purposes only."
>
> **No Derivative Works:** "Lets others distribute, display and perform only verbatim copies of the work. They may not adapt or change the work in any way."
>
> **Share Alike:** "Allows others to remix, adapt and build on the work, but only if they distribute the derivative works under the same the licence terms that govern the original work."

creativecommons.org.au/learn/licences/

Unit 3 Outcome 2
Analysis & Design
Review Questions

1. Name the activities for each of the four stages in the Problem Solving Methodology.
2. What data needs to be collected in the Analysis stage?
3. Describe two methods of collecting data.
4. What are two types of solution requirements that need to be identified at the analysis stage?
5. Name five types of constraints that can limit the development of a software solution.
6. Why is it important to determine the scope of a software solution in the analysis stage?
7. What is the difference between functional and non-functional requirements?
8. Name five non-functional requirements.
9. What are the components of an SRS?
10. Why is a Gantt Chart called a Project Plan?
11. Name the four components of a Context Diagram.
12. In a Use Case Diagram, what is the difference between 'includes' and 'extends'?
13. What is the purpose of a Use Case Diagram?
14. Name the four components of a Data Flow Diagram.
15. What is the purpose of a Data Flow Diagram?
16. When creating a Data Flow Diagram there are rules. Name them.
17. Which method of generating solution design ideas works best for you? Why?
18. A Design Folio is made of what four components?
19. What is the difference between efficiency and effectiveness?
20. Name nine measures for evaluating the effectiveness of a design.
21. What is affordance?
22. What is interoperability?
23. What is UX and why is it so important?
24. What are the three main development models and how do they differ?

25. What is the difference between an organisational goal and an objective? Give an example.

26. Describe the Australian legislation responsible for privacy.

27. Is it against the law to send unsolicited emails to sell your product? Why?

28. Describe the legislation that covers medical information and data.

29. Describe the nature of Copyright in Australia and how it differs from Creative Commons.

Applied Analysis & Design Questions

CASE STUDY PROBLEM

James Little manages live music venues around Melbourne. He prefers to work from his Android phone and needs an app to keep track of all the musicians he has on his register, their contact details and the types of music they play. The app would allow musicians to enter their availability and update any details to their performance types. James would then be able to book musicians for gigs at venues around town on specific dates and times so that the performances can be advertised leading up to the gig.

Question 1. To analyse this solution you will need to produce the following:

a) The Functional and Non-Functional Requirements,
b) The Scope,
c) The Constraints,
d) A Context Diagram,
e) A Use Case Diagram,
f) A Data Flow Diagram.

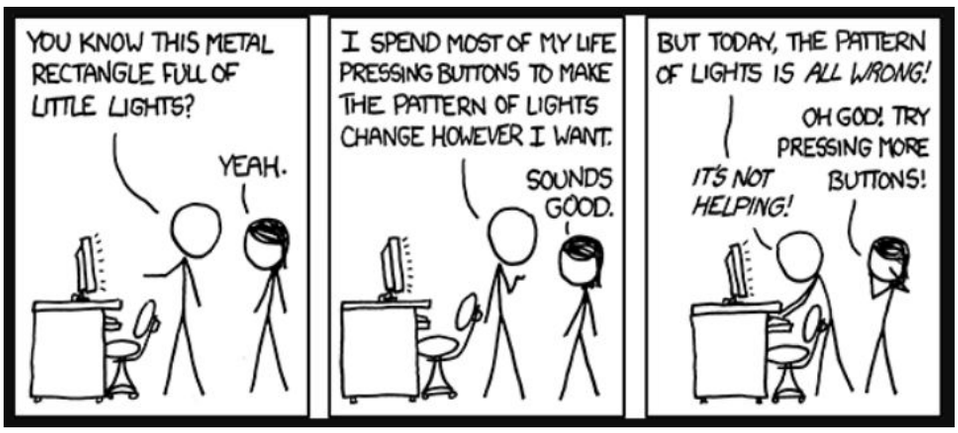
Question 2. Choose a method of generating design ideas and create the following for James' app.

a) A Mock Up and a Storyboard that shows how the user will interact with the app on the phone.
b) Annotate the designs reflecting on: usability, affordance, security, interoperability and marketability.
c) A Data Dictionary.
d) An Object Description Table.
e) An Algorithm using pseudocode.

Question 3. Evaluate the design you have produced in terms of efficiency and effectiveness.

Question 4. Write a goal and an objective for James' business.

Question 5. Identify legal requirements that might be pertinent to James' new system.

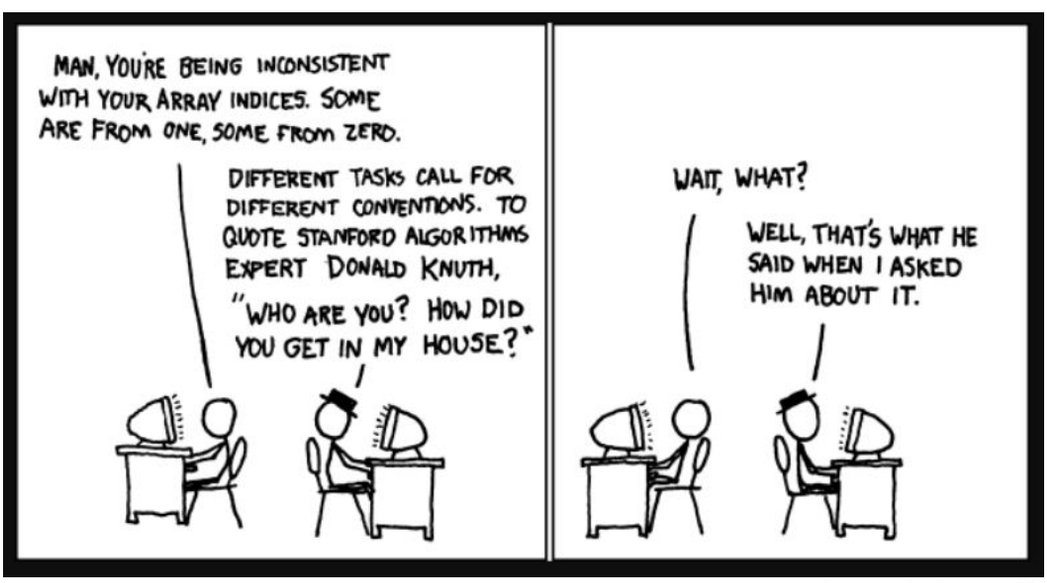


https://xkcd.com/722/

Online Support

There are plenty of resources online to assist you in developing your SRS and Design Folio. You might like to use the free drawing tool draw.io or project management tools online.

- Software.au.uk/resources (Example Evaluation Criteria lists)
- www.OpenProject.org (Free online project management tools)
- Draw.io(Free online drawing tools for diagrams)



https://xkcd.com/163/

Unit 4
Area of Study 1

Development and Evaluation

On completion of this unit the student should be able to develop and evaluate a software solution that meets requirements, evaluate the effectiveness of the development model and assess the effectiveness of the project plan. To achieve this outcome, the student will draw on key knowledge and key skills outlined in Area of Study 1.

Key knowledge

Digital systems
- procedures and techniques for handling and managing files and data, including archiving, backing up, disposing of files and data and security

Data and information,
- ways in which storage medium, transmission technologies and organisation of files affect access to data
- uses of data structures to organise and manipulate data

Approaches to problem solving
- processing features of a programming language, including classes, control structures, functions, instructions and methods
- characteristics of efficient and effective solutions
- techniques for checking that coded solutions meet design specifications, including construction of test data
- validation techniques, including existence checking, range checking and type checking
- Techniques for testing the usability of solutions and forms of documenting test results
- techniques for recording the progress of projects, including adjustments to tasks and time-frames, annotations and logs
- factors that influence the effectiveness of development models
- strategies for evaluating the efficiency and effectiveness of software solutions and assessing project plans.

Key skills
- monitor, modify and annotate the project plan as necessary
- propose and implement procedures for managing data and files
- develop a software solution and write internal documentation
- select and apply data validation and testing techniques, making any necessary modifications

DATA MANAGEMENT

File size, storage medium and organisation of files affect how your software solution accesses your data. It is important to take care to choose appropriate data structures and file types to store data for later retrieval. There are a few things that need to be ensured:

- All data conforms to internal consistency and standard classifications;
- Validity and integrity of the data;
- Allow different data sets to be integrated allowing for interoperability.

The system will only be effective if the data input is valid, correct, easy to access and consistent in its formatting and labeling.

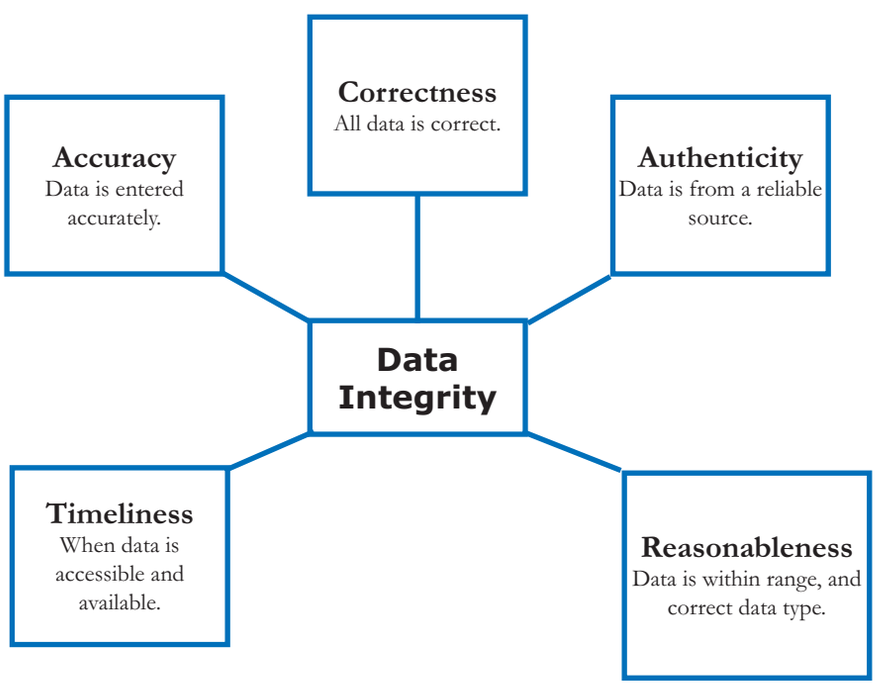
When formatting and storing data it is important to consider the following issues:

- How soon do I need the data back if lost?
- How fast do I need to access the data?
- How long do I need to retain data?
- How secure does it need to be?
- What regulatory requirements need to be adhered to?

When data is input, processed, stored and output as information it is important the quality of the data is ensured. Validation techniques make sure data entered into the system reflects the reality it is designed for.

Data Integrity is affected by five components:

1. Accuracy
2. Timeliness
3. Reasonableness
4. Authenticity
5. Correctness.



Security

Keeping data from deliberate and accidental threats is essential. Accidental threats include: users not saving, accidentally deleting files, losing or destroying a USB memory stick, sending data to the wrong destination in an email. Deliberate threats include: malware, phishing, hacking or deliberate vandalism. Event-based threats also need to be considered such as power surges or power failures and natural disasters. The best ways to protect from power related threats is to implement the use of an Uninterrupted Power Supply (UPS) and surge protection. Threats where deletion or destruction of data or hardware occurs, we need to consider storing it elsewhere. There are a range of ways of storing copies of data to protect it from loss.

Archiving

Archiving is the process of removing data from the current information system that is no longer used. It is then stored elsewhere on another server or perhaps on magnetic tape. This frees up memory space for active data in the system, but still allows access to the archived data if required.

Back-Up: Full, Differential & Incremental

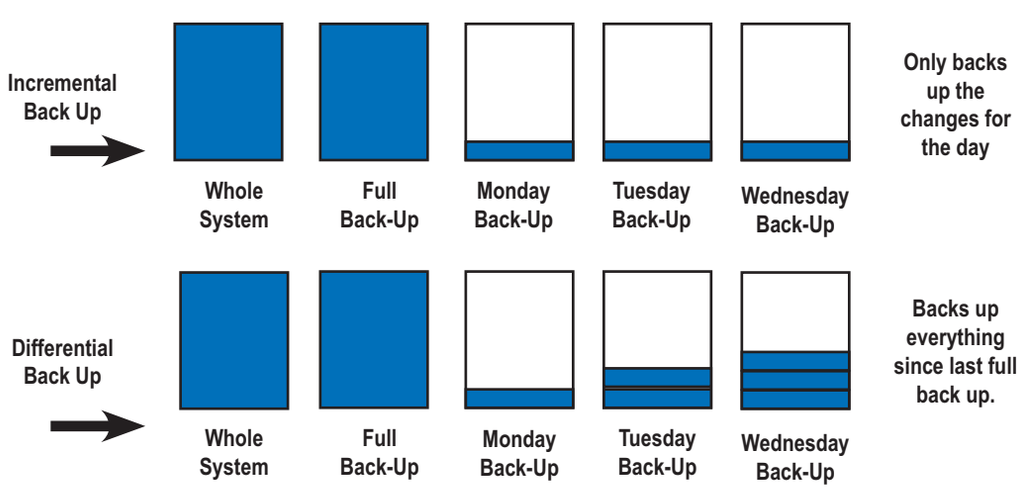
A back-up is a copy of data that is located elsewhere from the active data in use in the system. A full back-up makes a copy of ALL files on the system. This takes a long time and is often done overnight or on weekends onto magnetic tape. It is not done very often. In some instances, only once per month. Often off-site servers are used to control and store back up data. There are ways of making back-ups of only the data that is changed each day these are called Differential and Incremental. The difference is that while an incremental backup only includes the data that has changed since the previous backup, a differential backup contains all of the data that has changed since the last full backup.

Incremental back-ups build on the Full back-up. Incremental back-ups are used on a daily basis to only make copies of files that have been created or edited that day. It is more efficient to make copies of only the data that was modified that day, than to make full back-ups of the whole system on a regular basis. Often a full back-up is only run once a month or once a week, depending on the size of the system.

Differential back-ups make a copy of everything that was modified since the last full back up. In figure 3.1 you can see how a Full back-up copies the whole system. Incremental back-ups copy the changes made each day, while Differential back-ups make a copy of all the changes made since the last full back-up. Differential back-ups are sometimes called "cumulative incremental back-ups" and can be very large and time consuming to process. It could be great way to restore data lost over a period of time in preference to restoring individual incremental back-ups for each day.

Fig 3.1

Differential and Incremental Back Ups



You can imagine that with so many back-ups it is important to be using a consistent naming convention to manage each of the back-up files. Identifying the date and day each back-up occurs is crucial to getting the system up and running again after some catastrophic data loss in the system.

Data Disposal

Data disposal is necessary when it is no longer required and may contain sensitive information. Eventually all storage devices become filled to capacity and if data is not required to be archived, the data can be removed entirely. In the case of unwanted government computers, before disposal they must undergo special erasure processes to ensure that no one can access any remnants of data left behind.

Storage Media

Data storage is the recording and storing of information (data) in a storage medium. Recording is accomplished by virtually any form of energy. Electronic data storage requires electrical power to store and retrieve data. Data storage in a digital, machine-readable medium is sometimes called digital data. Barcodes and magnetic ink character recognition (MICR) are two ways of recording machine-readable data on paper. Electronic storage of data can be grouped into Primary, Secondary and Tertiary.

Primary Storage includes the RAM and ROM that directly support the CPU and it is volatile. This means all data is lost after the device is powered down. Secondary Storage differs from primary storage in that it is not directly accessible by the CPU. The computer usually uses its input/output channels to access secondary storage and transfers the desired data using intermediate area in primary storage. Secondary storage does not lose the data when the device is powered down. Secondary storage is non-volatile. After the power has shut down and restarted, the data on secondary storage can be retrieved.

Examples of Secondary Devices include:

- Hard Drives (these can be in-built to a computer system or be stand-alone external drives)
- CDROM
- DVD
- flash memory (e.g. USB flash drives or keys),
- magnetic tape
- standalone RAM disks
- Zip drives

Tertiary Storage typically it involves an automatic mechanism that will attach removable mass storage media into a storage device when required. Data are often copied to secondary storage before use. It is primarily used for archiving rarely accessed information since it is much slower than secondary storage. This is primarily useful for extraordinarily large data stores, accessed without human operators. Typical examples include tape libraries and full back-ups.

Back-Up Storage Media

Optical storage can be an option for storing backed up data. Optical storage includes CD ROMs, DVDs, UDO and Blu-Ray. It is not ideal for all situations but with the integration of jukeboxes and stack-loaders that automatically handle the loading and unloading of data, optical drives can be the most effective storage option for large quantities of video, audio, and image files. In figure 3.2 you can see the comparisons between optical drives for back up purposes.

Optical Media are random-access mediums, making it fast to access the data. It is cheap and can provide long term archiving. In recent years Facebook's data centre in North Carolina operates one of its "cold storage facilities" using Blu-Ray for long-term storage of user photos. When data needs to be accessed, a robotic retrieval system fetches a Blu-Ray disk, pulls data off the disc and writes it to a live server.

Fig 3.2

Optical Media	Description	Capacity	Costs
CD-RW	Re-Writable Compact Disc	1 GB	$1-$2 per unit.
DVD-R	Digital Video Disc Recordable	4 GB	$1-$2 per unit.
UDO	Ultra Density Optical disc - ISO cartridge optical disc. (Although now redundant, many organisations are still relying on this technology)	30-60 GB	$20 - $30 per unit.
Blu-Ray	High Density Optical Disc (Facebook uses BluRay for backup and cold storage of data.)	50 – 100 GB	$0.50-$1 per unit.

RAID Drives or Redundant Array of Inexpensive Discs are multiple hard drives that store processing data on two or more discs at once, depending on what level they are configured. RAID drives can be used in the processing of back-ups but should not be used as storage devices for backed up data. A RAID 0 is a pair of discs spinning at once managing the same processes by sharing the load, making it faster than one hard-drive alone. RAID levels increase with the number of discs they run. They work best as temporary drives while large amounts of data are being processed to a permanent Hard Disc Drive.

Hard Disc Drives (HDD) are the permanent data storage discs that can store terabytes of data and can be easily installed in any system. They can be found in servers, laptops and desktop computers. You can hear them when they spin up because they rely on the magnetic disc to be spinning for the heads to read the data. It is the random-access feature which makes it quicker to retrieve data. The mechanical aspect of the media can make HDD unreliable.

Flash/SSD Flash drive or Solid State Drives have no moving parts which make them less vulnerable to damage and data loss. The data is stored as logic gates in a microchip. These are light weight and portable but can be quite expensive. A common use of flash drives is in portable USB drives.

Both HDD and flash drives are capable of making disc images or ISO image files for archiving data. An ISO image is an uncompressed file format that can copy the entire data content of a drive. This method is not ideal for use in a disaster-recovery situation because it takes longer to unpack the data to recover the system after an disaster event.

Magnetic tape remains a common back-up media because it holds a lot of data and is very cheap. Unlike electronic media, magnetic tape is sequential. This means it is very slow to transfer data. It requires the tape be run over magnetic tape heads in a tape recorder to transfer data onto the tape, so it must run from start to finish of the length of the tape. Despite this, the capacity of recent magnetic tape formats range from 24 to 192 terabytes. This is a very good medium for archiving, or for full back-ups to be stored off-site. If you think magnetic tape is redundant, Google and Amazon made news in recent times of their disaster-recovery plans and the incorporation of tape back-ups.

Cloud Storage is a newly emerging solution for businesses. There are hundreds of Cloud Back-Up Services online that offer to store your back-up data for a monthly or annual fee. They provide a server that is accessible via the internet. This means all your data must be transmitted across the internet to access the server for storage. These organisations provide encryption to protect your data in transmission. It is important to be aware of the location of these services. If an Australian business uses a cloud back-up service located in another country, that service will not be subject to Australian legislation. Businesses can also fail financially, leaving your data unavailable. There have been plenty of instances where online data storage companies ave gone bankrupt and were unable to access their customer's data to return it.

Networks

Devices sharing data through a wired or wireless connection are a network. Each device on a network is a node and how they are configured classifies the type of network. A Local Area Network or LAN, is defined by a small localised area such as a single building, a home or office. These networks can be devices wired together, or they can rely entirely on wireless technology but they are defined as a LAN because of the limited physical area they occupy.

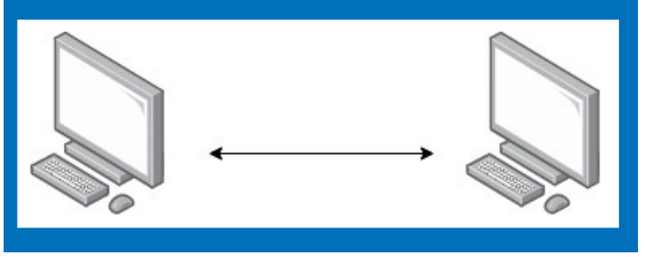
A Wide Area Network (WAN) is a network of networks. Your first experience of a WAN is at school where each building may have a LAN set up and linked together over the campus. Each building LAN may connect to a central server to appear as if the entire school campus is a single network. The WAN expands further when students and staff are able to access the school network from home (or anywhere else) over the internet. An Intranet is a service that provides network access from any location, for example, your school may have an online portal or Moodle that you can access from home through a web login page. All data is located and managed centrally and is defined by users with the authority to access it.

A LAN and WAN are network types differentiated by the area they cover. An intranet is a software service that allows users to access the WAN from any location.

Network Configurations & Application Architecture

How the devices share data determines the network architecture and how the software needs to be structures under those conditions. When setting up a network at home, devices are often connected directly to one another and share some resources such as printers. This is an example of a Peer-to-Peer network configuration. When you use torrent software for file sharing, this application architecture relies on peer-to-peer network connections. In figure 3.2 you can see the direct relationship between the two devices.

Fig 3.2 Peer-to-Peer Architecture



Most networks are Client-Server configurations where the application architecture processes the data on the server (central device) or relies on the server to provide up-to-date information. There are two types of client-server configurations;

- Rich-Client,
- Thin-Client.

Rich-Client is configured for the user to download all the software. All the data and processing occurs on the user's device, but it is still required to connect to the server to function. An example of this is when a Steam user downloads a game on their device. All the processing occurs on the device, but if the user wants to play against other users, access to the server is required.

Benefits of Rich Client:

- Fast internet access is not required to run the application.

Negative aspects of Rich Client:

- Uses a lot of memory and processing power in the user's device.
- Updates are required and may interrupt use.

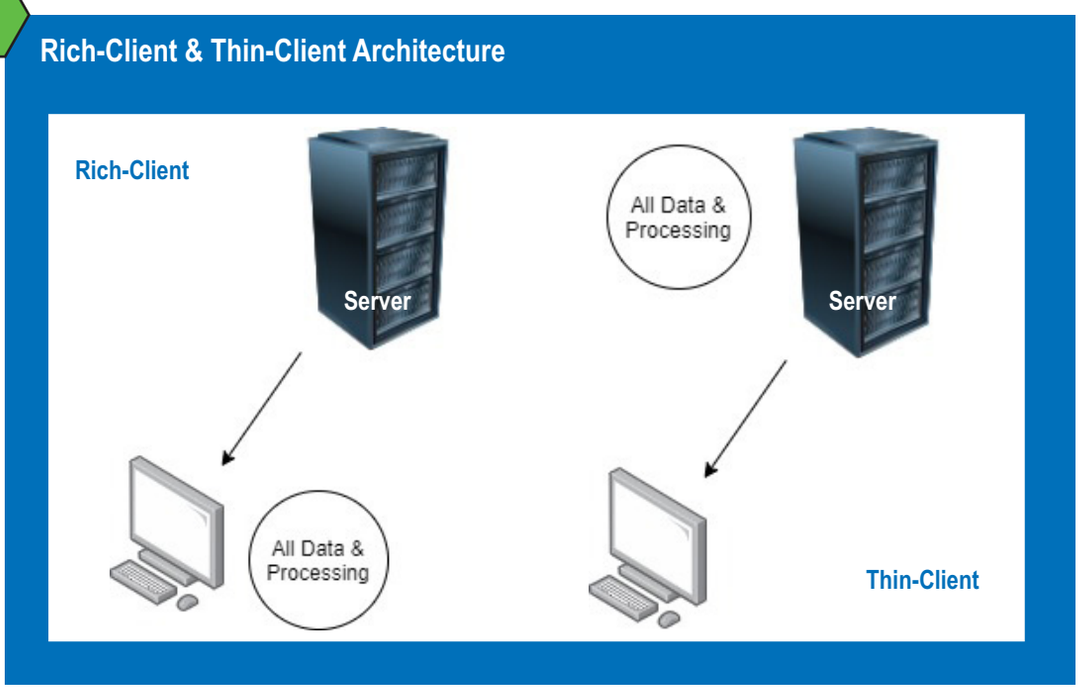
Thin-Client is configured for the user to download a light application, such as a mobile app, to their device. All the data and processing is done on the server. There is an application provided by the Bureau of Meteorology that provides weather information each day. The application simply downloads and displays the daily data from the server.

Benefits of Thin Client:
- Uses less memory and processing power in the user's device.
- Updates are required less frequently.

Negative aspects of Thin Client:
- Fast internet access is required to run the application.
- It will not access data if internet is not available.

As a software developer it is important that you understand how your software will rely on network connections and the type of network connection that is required. In figure 3.3, you can see how rich and thin clients are structured.

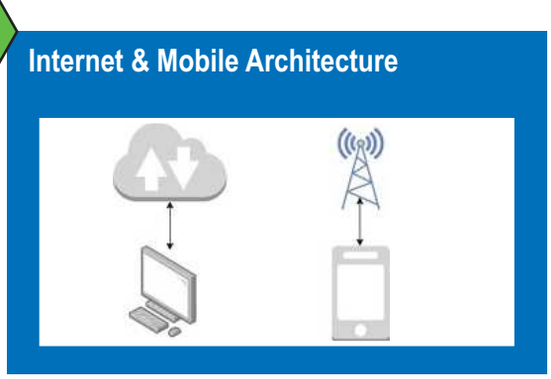
Fig 3.3



Mobile Architecture relies on mobile phone communication technology. Application that use mobile architecture can take advantage radio signals handled by mobile phone networks, or scalable software that can be interoperable with a range of different devices. See figure 3.4.

Internet Architecture relies on a web page interface and provides services to users via a browser. A commonly used service with this structure is Google. Google is a search engine with a web page interface. Users can have accounts and access their email, documents, and other services.

Fig 3.4



Servers and Protocols

Servers are dedicated devices that store data or process data. The most commonly encountered server is a central data server which all client devices can access to store and retrieve data. You can set up a hard drive at home to store files that all devices can access wirelessly. To transfer files across a LAN the protocol is commonly Ethernet.

Ethernet

Ethernet protocol is defined by the transfer medium (cable grade). When sending data across a LAN the data is broken into packets which contain information about the destination and source location, as well as data type identification and built-in error checking data. This is called Carrier Sense Multiple Access/Collision Detection (CSMA/CD). CSMA/CD is the process that governs how data moves through a LAN. Each device is able to "listen" to data signals on the Ethernet LAN and wait until the data traffic has halted before sending data packets. This protocol avoids data collisions on the network, making data transfer faster. If a data collision occurs, the sending device retransmits the data after a randomly calculated length of time.

Wi-Fi

Most LANs provide wireless access. Wi-Fi does not need a server, but there is a protocol that controls the wireless transfer of data. This is called 802.11x standard (where x identifies the various data transfer speeds). Wi-Fi uses a method called Request to Send/Clear to Send (RTS/CTS). Data is transferred via radio waves that use a collision detection method similar to Ethernet. Encryption is also required, which is covered further in Cybersecurity in Chapter 4.

FTP

When accessing a central server on a WAN where whole files need to be transferred across the internet, File Transfer Protocol is required. An FTP server is dedicated to storing and transferring whole files to client devices that have authorised access. Some software requires FTP architecture to access XML files to populate databases.

Email

Another commonly used server is an email server. The device is dedicated to storing and processing emails to send to correct recipients. There are two commonly used email protocols. Internet Message Access Protocol (IMAP) is a mail protocol used for accessing email on a remote web server from a local client. Post Office Protocol version 3 (POP3) is a standard mail protocol used to receive emails from a remote server to a local email client.

Web

Web servers host web sites and process the programmed interactivity of the website when accessed by a visitor on a browser. There are many protocols used to ensure data transfers across the internet safely. Internet Protocol (IP) enables connections between networks where all devices have IP addresses. It delivers packets from the source web host to the destination. The client browsing the internet is the destination of a web request which uses the IP addresses. IP is used in conjunction with Transfer Communication Protocol (TCP). TCP is one of the main protocols of the internet. The IP protocol deals only with packets of data while TCP enables two hosts to establish a connection and exchange streams of data. These two protocols are usually termed TCP/IP as they work together in transferring data across multiple networks. HyperText Transfer Protocol (HTTP and HTTPS Secure) are the protocols used by the World Wide Web (WWW). The WWW is not the physical network of devices, but the network of hypertext documents that are stored on the web servers. HTTPS protocol defines how any hypertext document is formatted and transmitted, and what actions web servers and browsers should take in response to various HTML instructions.

Network Devices

All network enabled devices have integrated network interfaces that allow physical connections with Ethernet or wireless connections using 802.11x. Each device has a Media Access Control (MAC) address number that identifies it on the network. Each device requires a Network Integration Card (NIC) to connect to the LAN via ethernet or via the WiFi.

Switch

A switch is a device that provides connections from multiple end-user devices and peripherals to a server. In your computer classroom you may have over 20 computers all connected by blue Ethernet cables so that you can access the internet, your email or a central server. These blue Ethernet cables are connecting each computer to a switch. The switch may have up to 48 ports, allowing 48 devices to access the network.

Router

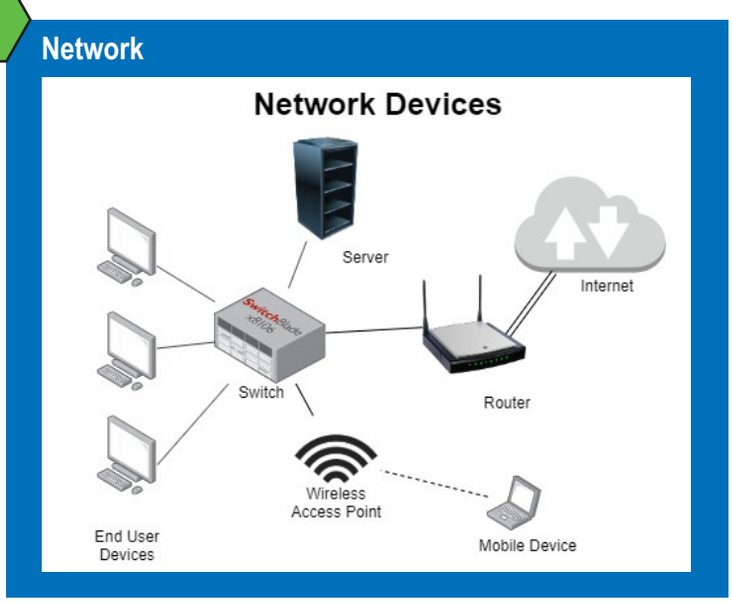
A Router is a device that connects networks of different protocols. It is most commonly used in home networks accessing the internet. It can also connect LANs to an optical fibre backbone to connect to a company or campus WAN. A router reads the address data in the transferred data packet to determine the ultimate destination. It has a stored routing table that allows it to determine how it will direct the packet to the next network on its journey towards its ultimate destination.

Transmission Media

When data is sent a long distance using Ethernet cable (commonly known as CAT5 or CAT6) the signal degrades, in much the same way as a beam of light eventually fades out. To keep the signal strong, networks use repeaters which amplify the signal. Wi-Fi radio signals also have limits to the distance they can cover, so Wireless Access Points are located where mobile devices are used. Indoors, 802.11x has a range of about 50m while outdoors it can range up to 100m. Wi-Fi is affected by environmental factors. Obstructions indoors such as brick walls can reduce the range. Interference from microwave ovens, cordless phones and other equipment that operates on the same radio frequency as the Wi-Fi can also affect the range. Ethernet cable is a form of Unshielded Twisted Pair (UTP) cable. The connectors at each end of the cable and quality of wire determine the standards. The cables are made from copper wire twisted to limit interference and carry electronic signals. The standards commonly used are CAT5 that transfers at 100Megabits per second (100Mb/s), CAT5e transfers up to 1 Gigabit per second (1Gb/s) and CAT6 transfers up to 10 Gb/s. Wi-Fi signals are radio waves transmitted at several frequency bandwidths between 2 and 5 GHz. Network speeds vary based on the Wi-Fi standard used and the environment.

Optical Fibre cable is a physical connection cable that uses light technology to transfer data. Optical fibre requires a converter to convert electronic signals from UTP to light signals. Optical Fibre is much more expensive that UTP but carries much more data for a longer distance without the need for repeaters.

Fig 3.5



USES OF DATA STRUCTURES

Data structures allow for efficient organisation, management, and storage of data. The way the data is structured determines the relationships between the data points. In Chapter One we investigated four data structures: records, dictionaries, arrays and hash tables. Each of these data structures creates different relationships between the data.

Uses for Records

Records are used to store data of different types pertaining to an entity. A common example would be a record of a customer or a staff member in a database. The person would be the entity and the record would be made up of fields such as 'name', 'address', 'phone number'. Each field has a different data type and each record has a unique identification number. Records are used when data pertaining to an entity needs to be collected, stored, retrieved or modified. Entities can be people or products. In transactional processing systems, an record can be a transaction, such as a receipt of purchase. Records can describe abstract things such as a booking time, or rooms that can be booked in a booking system. Each room would have information about its location and facilities that could be stored in a record. Each booking record would contain the date and time of the booking, the person booking the room and room identification. In this course you are required to understand how to use CSV files and XML files. Let's look at a transactional processing system(TPS) of records. A local sports club has a list of members, their contact details and scores from previous games. The club also has a calendar of games with other sports clubs in the area. The transaction would be allocating each person to a team to play each of the scheduled games. The figure 3.6 shows how different records can be used in a TPS through the use of XML files.

Fig 3.6

```
<member>
        <MemberID>342</MemberID>
        <GivenName>Sudha</GivenName>
        <FamilyName>Jespik</FamilyName>
        <MobilePh>0465 345 333</MobilePh>
        <Email>JespikS@gmail.com</Email>
        <Score>72</Score>
</member>
```

```
<game>
        <GameID>1</GameID>
        <Date>10 August 2022</Date>
        <Time>10am</Time>
        <Location>Main Park</Location>
</game>
```

```
<TeamAllocation>
        <TeamAllocationID>1</TeamAllocationID>
        <GameID>1</GameID>
        <Position1> <MemberID>342</MemberID></Position1>
        <Position2> <MemberID>332</MemberID></Position2>
        <Position3> <MemberID>093</MemberID></Position3>
        <Position4> <MemberID>266</MemberID></Position4>
        <Position5> <MemberID>781</MemberID></Position5>
</TeamAllocation>
```

It's important to organise the structure of the records so each table (XML file) can easily import and export data into your code. Chapter 5 contains instructions on how to program Visual Basic code to store and retrieve data to XML files. For extra assistance in XML and VB visit the support website: vicfarrell.com.au.

Uses for Arrays

One dimensional arrays (also called a linear array) are used for storing lists of single data types. The way they are arranged with indexes for each data item, makes them perfect for sorting and searching. The power of an array is in the way it orders the data. When a large amount of data is collected, such as the case of a wearable fitness tracker, collecting the wearer's number of steps each day as well as other biological measurements It is important that the data is stored chronologically. Arrays allow for the location of the data in the list to provide added information about the data. Other examples of linear arrays include; strings of characters to change spelling in text editing, items in a navigation menu or lists of measurements. It is possible to create multi-dimensional arrays which work the same way. Each item has two or more indexes to identify the location of the data in the array structure. In this course you are most likely to use a 2D array. In figure 3.7 you can see the difference between a linear (1D) array and a 2D array. These types of arrays are indexed. Associative arrays are not indexed. They can use any data as a key in the same way a dictionary data structure is able to use a name as a key. An associative array can use other non-numerical data as keys to locate data in the array. Chapter 5 contains examples of how to code arrays in Visual Basic.

Fig 3.7

Types of Arrays

1D Array: Names(2)

Names(0)= "Sudha",
Names(1)= "Yi"
Names(2)= "Susan"

Associative Array / Dictionary

(GK) Sudha
(DF) Yi
(FF) Susan

2D Array: TeamNames(2,1)

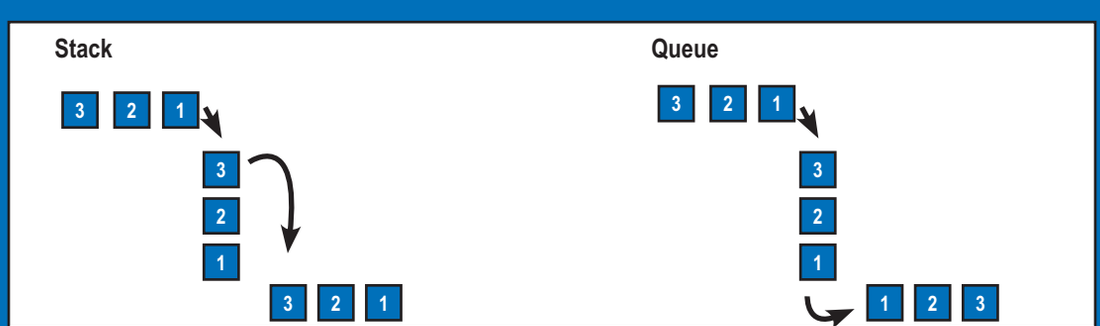
TeamNames (0,0) "Sudha" TeamNames (0,1) "Goal Keeper"
TeamNames (1,0) "Yi" TeamNames (1,1) "Defence"
TeamNames (2,0) "Susan" TeamNames (2,1) "Front Forward"

Stacks and Queues

Arrays allow for random access. The data can be accessed at any point in the data structure. Sometimes we want to restrict how data is accessed so we create sequential access data structures called Stacks and Queues. A stack is processed using the "last in - first out" (LIFO) method. Imagine you are placing large, heavy bricks in a stack, one of top of the other. You can only access the top brick at any one time. So as you stack bricks 1, 2, 3. You can only access them in the order 3, 2, 1. An example of the use of a stack is when we use an undo feature in word processing. Each task that you give the computer is stored in a stack, so that when you use Ctrl Z to undo, it removes the last task you did. A queue is processed using the "first in - first out"(FIFO) method. Imagine you are waiting in line at a store. You stand behind the person in front, The first person in the line will be served first and the last person in line will be served last. If person 1, 2, 3 line up, persons 1, 2, 3 will be served in order at the counter. An application of a queue is to manage print requests sent across a network. As the printer receives print jobs, it will store the requests in a queue and process each job in order. In figure 3.8 below, you can see how stacks and queues manage data and process requests in different orders.

Fig 3.8

Stacks and Queues



Uses for Hash Tables

A hash table is commonly used for logins and passwords. If you have ever tried to create an Access database, you might have come unstuck if you tried to create a table or field called "password". It is a restricted term to avoid hackers or bots from finding this sensitive data. A hash table is a matrix of data with numerical indexes in a similar way to a 2D array. Unlike a 2D array, there is a relationship between the numerical index and the data content. This relationship is a hash function. In this course you will be using Mod(). It calculates the remainder after a value has been divided. For example:

10/3 = 3 remainder 1.
therefore
Mod(10, 3) = 1

Not all logins and passwords are made of numbers, but thankfully all digital data is. Visual Basic can convert a character into an integer:

```
Dim Input As Char
DimConvertedInput as Integer = Convert.ToInt32(Input)
```

Then you can apply your hash function:

```
Dim Hashed as Integer = Mod(ConvertedInput, 5)
```

This will provide locations in five different lists. See figure 3.9 below to see where other values would be placed.

Fig 3.9

List(0)	List(1)	List(2)	List(3)	List(4)
15	11	7	18	9
20		22		

Here is an example of how a hash table can store passwords securely. Let's look at the following login details:

UserName: BenjaminUber
Password: 2005FoxTrain12

The username will be stored in the user records which can easily be verified. Then we convert the password "2005FoxTrain12" to a series of integers (50, 48, 48, 53, 70, 111, 120, 84, 114, 97, 105, 110, 49, 50) and add them up (1109).

Then we apply our hash function to the total:

Mod(1109,5) = 4.

This means we can store our password in List(4).

When Benjamin Uber tries to log in with his username and password later, the code will hash his password input to find out which list it should search. The code will search for a match in List(4). If Benjamin has made an error with his password, no match will be found regardless of which array is searched. The storage of passwords in a hash table removes the connection between the login username and the password. The user is the only connection and it is made when they enter their login details.

EXAMPLE USES OF DATA STRUCTURES

Problem 1

Cape Jarvis Community College PE Department needs an application to enter student scores for athletic sports. Teachers require secure logins to enter and modify these scores. Teachers also need access to the student's personal details including health and contact details in case of emergencies. The app should be able to provide measured Fitness Beep Tests and collect data from wearable fitness devices to store and calculate fitness data such as heart rate. PE teachers need to be able to search up a student by their student ID number and enter scores for sprints, distances for long jump and discus as well as heights for high jump.

Data Structures to solve this problem:

1. Records would be required to store student personal information, their student number, health and contact details.
2. 1D Arrays store each student's fitness data from the wearable device so it can be analysed or transferred into their Scores Array.
3. A Multi-Dimensional array (Scores Array) could store all the student data where each row represents each student while columns could hold their Student ID, event scores and calculated fitness rating.
4. A Hash Table could be used to store and retrieve teacher login details and Student IDs to ensure security in accessing the Scores Array and the Records.

Problem 2

Global Dash-Cam Tech are developing software to install in the new Toyota Prius. The software uses a SIM card to stay connected to the internet via the driver's mobile network. Data from the Bureau of Meteorology on temperature, rainfall and any weather warnings is downloaded and displayed on an hourly basis on the dash board. When the driver enters a route into the maps application, traffic reports are downloaded and updated into maps to warn the driver of any incidents on the route. Routes and destinations are stored for quick retrieval.

1. Weather Data is downloaded hourly and will not be required to be kept so a 1D array can store this data for display.
2. Map data is a matrix and would require a 2D array to store current map data.
3. Stored routes and destinations would contain large amounts of data that needs to be quickly accessed and the data would be best stored in a Hash Table.

Features of a Programming Language

Object-oriented programming, or OOP, is an approach to programming solutions where all computations are carried out using components called objects. Each object knows how to perform certain actions. For example when a check box is clicked it will be set to TRUE and this can be used as a control for a variable. Another example of an object is a textbox. The purpose of a text box object is to read in text which could then be assigned to a variable or other data structure. An object is a combination of code and data that can be treated as a discrete unit. An object can be a piece of an application, like a control or a form. An entire application can also be an object. Each object is determined by the behaviours of its class.

Classes

A class determines and defines the contents of objects such as variables, properties, procedures, and events. Objects are instances of classes. In Visual Basic a class statement is a fundamental building block of a programming solution. A solution may have more than one class to serve different requirements. For example, an 'EndUser' class may have different object requirements to an 'Administrator' of a large

banking system. Creating separate classes for these two users limits the inheritance of all the objects within the scope of the class. The Administrator class objects should have unrestricted access to the system while EndUser class objects should have restricted access.

Methods

Objects are determined by their operations and properties. Textboxes can read in typed data, format the data, be visible or not visible, have a green background etc. In Visual Basic we call the control of these properties from the code, methods. A method is an action which an object is able to perform. Methods of a radio button include; focus, click, default setting (true or false).

Instructions

An instruction is an action that a program should carry out. Below is an instruction to assign the variable 'strName' to the data content of object 'txtName' by method '.Text'.

```
strName = txtName.text
```

The following is a calculation where data is processed and assigned to a new variable:

```
DblTotalPrice = DblSalesPrice * IntQuantity
```

Sometimes instructions can be grouped and called a procedure. A procedure is a self-contained group of instructions that carry out a specific task. This is also called a routine, subroutine or module. In Visual Basic a procedure is a block of Visual Basic statements enclosed by a declaration statement (Function, Sub, Operator, Get or Set) with a matching End declaration. All executable statements in Visual Basic must be within some procedure. Below is an example:

```
Public Sub Main()
    Dim strName As String
    strName = InputBox("Enter your name", "Name")
    frmStart.Caption = "Hello " & strName
    frmStart.Show
End Sub
```

Functions

A function is a type of procedure; a series of instructions in a single unit to achieve some result. Usually a value is returned (but not always). A function typically requires some input (called arguments) to return a result. In the example below a size is set as a parameter, then the ShapeMenu Combo box allows the user to choose from three options; Square, Circle or Triangle. Each selection case calls up a function to draw the shape to the size by calculating the area of the shape to fill with colour.

```
START
        Read in Size
        Select Case ShapeMenu
                Case 0 Function Square ()
                Case 1 Function Circle ()
                Case 2 Function Triangle ()
        END Select

        FUNCTION Square (Size) As Single
                Return Fill Colour (Size^2) )
        END FUNCTION

        FUNCTION Circle (Size) As Single
                Return Fill Colour ( (22/7 )* (Size^2) )
        END FUNCTION

        FUNCTION Triangle (Size) As Single
                Return Fill Colour ((Size^2)/2)
        END FUNCTION
END
```

EXAMPLES OF APPLIED DATA STRUCTURES

An algorithm for managing bank accounts follows. It has three classes; two are public, and one is private. The private class can access the public class but not the reverse. This keeps the private class variables separate from other classes. The UserAccount Class has two subroutines. The first displays all the transactions from the Transaction Array through the use of a counted loop (iteration). The second transfers an amount from one account to another. This algorithm in pseudocode calls up a function "Transaction" to the do the calculation for the new balances and return the new balances.

```
Public Class Login
        Sub LoginButton
                Read in UserName
                Read in Password
                Call FUNCTION Hash(Password)
                IF (UserName= Record.UserName) AND (HashTable_Search = FOUND) THEN
                     Access Class UserAccount
                ELSE
                    Write ("Incorrect Login Details")
                END IF
        END Sub
END Class


Private Class UserAccount (UserDetails)
        Sub DisplayAccountDetails
                 Display UserName
                 Display AccountNumber
                 Display AccountBalance
                 Display Transactions()
                 FOR i = 0 to EOF
                       Display Transactions (i)
                 NEXTi
        END Sub


        Sub Transfer
                Select Account_From
                Select Account_To
                Display and Return Balance.Account_From
                Display and Return Balance.Account_To

                Read in TransactionAmount

                CALL FUNCTION Transaction(TransactionAmount)

                Display NewBalance.Account_From
                Display NewBalance.Account_To
                Balance.Account_From = NewBalance.Account_From
                Balance.Account_To = NewBalance.Account_To
        END Sub
END Class


Public Class Calculate(Amount, Balance.Account_From, Balance.Account_To)
         FUNCTION Transaction()
                  NewBalance.Account_From = Balance.Account_From – TransactionAmount
                  NewBalance.Account_To = Balance.Account_To + TransactionAmount
                  Return NewBalance.Account_From
                  Return NewBalance.Account_To
        END FUNCTION
END Class
```

Approaches to Evaluating Design Solutions

Efficiency

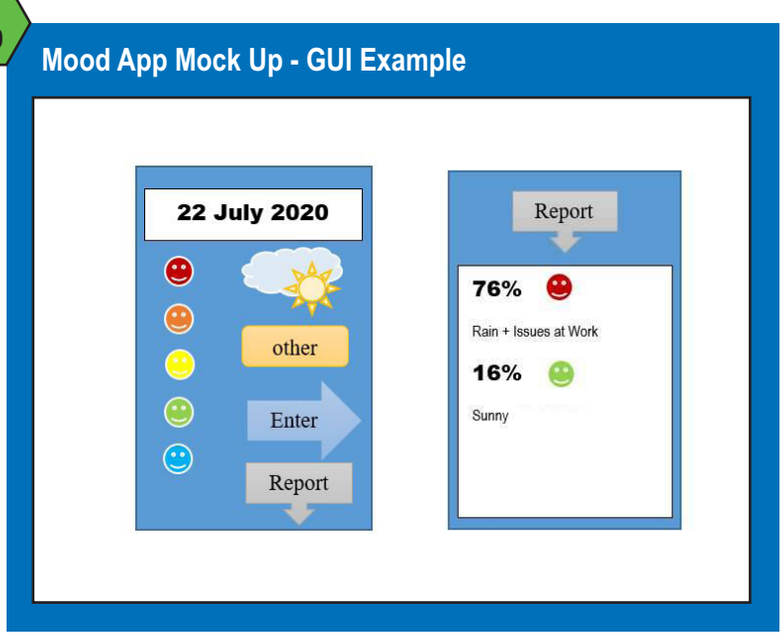
Efficiency is the avoidance of wasting time, money and effort, while effectiveness is in relation to how the solution meets it requirements. An efficient solution does its job quickly, doesn't slow down the user's work-flow by being confusing, is easy to maintain, easy to use, easy to learn and does a lot of work in a short amount of time.

Efficiency incorporates:

- speed of processing
- ease of use
- easy to learn to use
- cost of data and file manipulation in time, money and effort
- productivity
- operational costs in time, money and effort
- level of automation implicit in the system.

An efficient software solution does not slow down the user's work flow by being difficult to learn how to use, or by having confusing interface elements. An efficient solution saves time, money and effort and produces information quickly.

Fig 3.10



The Mood Phone App – Investigating Efficiency

The Mood App is a mental health data application that allows the user to enter how they feel on a rating of 1 – 5 on a daily basis. The user can also enter other data such as weather, illness or other aspects of life to investigate any patterns that may affect mood. The purpose of the application is to find any correlation between moods and other aspects of life. You can see a mock up of the GUI in figure 3.10 above. All data is ordered by date. Each date is a record that holds the mood rating, weather data and other aspects such as problems at work. A linear search is used to find patterns between the record fields. The output shows the percentages of each mood against other data that may have affected that mood.

Speed of Processing: Although easy to program, Linear Search is a very slow method of searching especially with a lot of data to investigate. A report may take longer than necessary to produce.

Ease of Use: The graphical interface appears easy to use but without any alternative text to illustrate instructions to the user, it may take some time for the user to learn.

Cost of Data and File Manipulation: records hold a lot of data and can take up a lot of space on the phone. Using an array of integers to represent options could save space and be faster to search and update.

Effort: Both the mood and the weather fields are represented graphically in the GUI and only require a touch selection. It is unclear as to how the user will enter the "Other" data and this may be more effort for the user than they are prepared to make. This could impact how much data is included.

Cost of Time: How much time does it take the user to update each day? Could the application be changed to ensure it takes less than a minute to update?

Automation: What aspects of the application can be automated to increase efficiency? A reminder notification could ensure the user enters data each day to avoid having to enter data from previous dates that have been missed. Weather data could be input from another source automatically.

Effectiveness

An effective solution is complete, functional, enjoyable to use, attractive and easy to use. Effectiveness incorporates:

- Readability/clarity – able to read the information produced due to the legibility of the interface. Both the interface and the output is understandable for the user.
- Usability – the user experience of the software's application in context.
- Accessibility – a measure of how well it can be used in any given context.
- Completeness – meets all the needs of the user. Nothing has been omitted.
- Accuracy – the data input is correct and is correctly processed. Validation ensures correct data entry.
- Relevance – is the information produced relevant to the needs of the user? How can the data and information be organised and presented in a way that maximises its usefulness to the user?
- Communication of message – Is the software presenting the data in a way that is suited to the needs of the user?
- Attractiveness – are the colour schemes, design, fonts and other graphic elements attractive and suitable for the purpose of the software?
- Timeliness – is the production of information from the software available in time for it to be useful?

Effectiveness is a measure of how well the software solves the information problem. Measures of effectiveness can be made by investigating:

- Currency of files
- Ease of retrieval
- Integrity of data and security

The Mood Phone App – Investigating Effectiveness

Readability: There are some issues with readability and clarity with the proposed application. The initial screen relies on the user understanding the icons without alternative text. The output from the Report lists percentages against moods and other data. It is not clear if the percentage is reporting on the number of times these occur together or the number of times they are entered in a selected amount of time.

Usability: There is no clear structure on the initial page to guide the user to interact. The weather icon does not contain the affordance that a button icon has in demonstrating how it is to be used. It appears that some experimenting will be required by the user to understand how it is to be used effectively.

Accessibility – the reliance on icons makes it easier for phone users to enter data rather than text boxes that need to be typed in, however the mood icons rely on the user being able to distinguish colour which may be a problem for the colour-blind.

Completeness – there appears to be no lists to choose from for the user to add "other" life aspects that may affect mood. It is left to the user to make a decision to add this to their entry. A more complete solution would provide a list for the user to select from.

Accuracy – the data entered is very vague in identifying mood in five levels without any distinctions between the types of mood. The output of this application will be very limited in identifying causes of mood swings.

Relevance – all aspects of the software must be relevant to the purpose of the software. Are all inputs relevant to the output? If the App will be an effective mood monitor will the causes of the moods be relevant data hat is required? With more research a wide range of items could be included in the interface such as work related issues, illness, social activities, financial problems etc.

Communication of Message - is the output easy to understand? The example provided shows 76% Rain and Work Issues. Does this mean that the user has logged 76% of their lower moods as related to Rain and Work Issues or are the total number of logged moods low? A clearer report would make the output more useful. For example, "Your Lowest moods appear to be related to 'Rain' and 'Work Related Issues'."

Attractiveness – the application must look professional and appealing otherwise users will not enjoy using it. If it does not look professional users may not feel that the output can be relied upon. The example does not appear to be professional in nature.

Timeliness – with a linear search you might find that the outcomes of a report may not be returned when the user requires it. Does the user have an option to choose a time to be notified for a reminder to add the daily data? This could make the application more effective if the user enters data each day.

TESTING TECHNIQUES

Each design specification outlined in the design folio must be tested to ensure the requirements are met and the output is free of errors. There are a range of things that need to be tested to ensure the software not only works, but is safe to run.

Finding errors in the code is all part of the development process. While you are coding you can detect and eliminate syntax errors easily because the code will not run with syntax errors. Syntax errors are errors in the code where the correct language rules have not been used. A common rookie error when learning Visual Basic is to forget to add the property to the object name, or to incorrectly spell the name of a variable.

Testing for Errors

Runtime errors indicate bugs in the program when it is running. These can be problems that the designers may have anticipated but could do nothing about, such as running out of memory. Logic errors are common when learning to program. Below is a VB Program that reads in the height and weight of the user and calculates the Body Mass Index (BMI). This is done by using the following calculation:

$$BMI(kg/m^2) = Weight\ (kg) / (Height^2(m)$$

There are cut offs for normal weight BMI (18 – 25) Below 18.5 is defined as underweight and above 25 is defined as overweight.

```
1    Class BMI
2       Sub Calculate _ Click
3             dblHeight = txtHeight.text
4             dblWeight = txtWeight.text
5             dblBMI = dblHeight/(dblWeight^2)
6             IF (dblBMI < 18) THEN
7                   strRating = "Underweight"
8             ELSEIF (dblBMI >25) THEN
9                   strRating = "Overweight"
10            ELSE
11                  strRating = "Normal"
12            ENDIF
13            MsgBox = (" Your BMI is " & dblBMI & " which makes you " & strRating )
14      END Sub
15   END Class
```

If you examine line 5 you will see a logic error. You can test to see what happens by running the program with appropriate data. We can develop some testing data. The testing data needs to test the boundaries of the conditions in the solution. Our conditions include:

dblBMI<18
dblBMI>25

Fig 3.11

TESTING TABLE

Height	Weight	Expected BMI	Expected Result	Actual	Signed
1.70	70	24.2	Normal	Normal	✓
1.68	51	18.1	Underweight	Normal	X
1.63	72	27.1	Overweight	Overweight	✓
1.55	59	24.6	Normal	Normal	✓
1.77	53	16.9	Underweight	Underweight	✓

In the Testing Table in figure 3.11 you can see test data for Height and Weight that calculates BMI values to test the conditions around 18 and 25 cut offs. A person with a weight of 51kg and a height of 1.68m should be in the normal BMI range (18.1). Instead we are getting "underweight". This is incorrect due to the logic error in line 5. If we correct the error with the line:

dblBMI = dblWeight/(dblHeight^2)

So far this program is not protected from inaccurate input. If Height is entered as zero, or simply omitted, we will get a runtime error because the value for calculated BMI will be undefined.

dblBMI (undefined) = dblWeight / 0 ^2

Trace Tables

In some circumstances it is important to test the logic throughout each line of code to find where the error may reside. The following program uses a text file with a list of values on each line. This data populates a 2D array called CarArray(4,1). A "For" Loop is used to collect the data from CarPrices.txt to put the values into the array along with the types of cars. It reads in the number of days a customer wishes to hire a vehicle. A ComboBox allows the customer to select the vehicle type which sets the CarType and CarPrice variables. There is a 10% discount for customers who hire a vehicle for 5 or more days. It then calculates and displays the TotalCost due. A section of the code responsible for the error follows.

```
1       Dim Prices As StreamReader = File.OpenText("CarPrices.txt")
2       Dim CarArray(3, 1) As String
3       Dim Days as Integer = Int(txtDays.Text)
4       Dim TotalCost as Integer
5       Dim CarType as String
6       Dim CarPrice as Integer
7
8       CarArray(0, 0) = "Hatchback"
9       CarArray(1, 0) = "Sedan"
10      CarArray(2, 0) = "Station Wagon"
11      CarArray(3, 0) = "Ute"
12
13       FOR i = 0 To 3
14          CarArray(i, 0) = (Prices.ReadLine())
15       NEXT
16
17      Select Case ComboBoxcarType
18         Case 0
19              CarType = CarArray(0,0)
20              CarPrice = CarArray(0,1)
21         Case 1
22              CarType = CarArray(1,0)
23              CarPrice = CarArray(1,1)
24         Case 2
25              CarType = CarArray(2,0)
26              CarPrice = CarArray(2,1)
27         Case 3
28              CarType = CarArray(3,0)
29              CarPrice = CarArray(3,1)
30      END Select
31
32      IF (Days > 5) THEN
33          TotalCost = (CarPrice * Days) – ( (CarPrice * Days) * 0.10)
        ELSE
34          TotalCost = (CarPrice * Days)
35      ENDIF
36
37      MsgBox("You have booked a " & CarType & " for " & Days & " for a total of $ " & TotalCost)
```

CarPrices.txt

30
35
40
50

Unfortunately, when we used a Testing Table this code returned errors. To investigate where they are and how they are creating problems we can use the Trace Table in figure 3.12 below. Using the Trace Table example from Chapter One we list the variables across the top and trace the value of each variable change though each line of code when we enter the test data.

Fig 3.12

TRACE TABLE

Test Input	i	CarArray		CarType	CarPrice	TotalCost
Days = 2 Ute is selected	0	(0,0) 30	(0,1)	50	(blank)	(blank)
	1	(1,0) 35	(1,1)			
	2	(2,0) 40	(2,1)			
	3	(3,0) 50	(3,1)			

You can see that the code fills the wrong section of the CarArray with the data from the text file. If we look at the code the error is in line 14. It should read:

```
CarArray(i, 1) = (Prices.ReadLine())
```

This change will mean the CarPrices will go into the second column of the array and then be transfered to the variables. All goes well until we enter 5 days. You can see in the Trace Table in figure 3.13 the 10% discount is not applied.

Fig 3.13

TRACE TABLE

Test Input	i	CarArray		CarType	CarPrice	TotalCost	Issue
Days = 5 Sedan is selected	0	(0,0) Hatchback	(0,1) 30	Sedan	35	175	X
	1	(1,0) Sedan	(1,1) 35				
	2	(2,0) Station Wagon	(2,1) 40				
	3	(3,0) Ute	(3,1) 50				
Days = 4 Sedan is selected	0	(0,0) Hatchback	(0,1) 30	Sedan	35	140	✓
	1	(1,0) Sedan	(1,1) 35				
	2	(2,0) Station Wagon	(2,1) 40				
	3	(3,0) Ute	(3,1) 50				
Days = 6 Sedan is selected	0	(0,0) Hatchback	(0,1) 30	Sedan	35	189	✓
	1	(1,0) Sedan	(1,1) 35				
	2	(2,0) Station Wagon	(2,1) 40				
	3	(3,0) Ute	(3,1) 50				

With closer inspection, you can see that line 32 does not allow Days = 5 to apply the 10% discount. It can be fixed with a correction in the condition in the IF statement.

```
IF (Days => 5) THEN
```

Types of Errors

Arithmetic Overflow

In computer programming, an overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range allocated for its storage or variable type. If, for example, you have allocated 2 bytes for the variable "IntAmount" (minimum 0 – maximum 99) and a calculation during runtime sets IntAmount to 116, there will be an Integer Overflow. There will not be enough storage space allocated for the variable.

Buffer Overflow

Sometimes large amounts of data need to be stored temporarily in memory before it is processed. Often when importing video or large images into software there can be buffering issues where the computer may slow down and wait till all the data is stored or processed. A buffer overflow can occur if a database with large files is accessed with an SQL query to search for within a range. Processing the search may require temporary storage of large files which can fill up the memory reserves causing the processing to stop.

Memory and Handle Leaks

When the software incorrectly manages memory allocations, a memory leak can slow down performance or cause the device to stop working correctly. A handle leak occurs when a computer program requests a connection to a resource on the computer, but does not release the connection when it has completed its task. This can limit access to that resource or slow down the device.

The Design Specifications

After correcting the errors, our Car Hire code produces the correct output, but does it meet other design specifications? The requirements for our software may include:

- Easy to use
- Able to edit incorrect input
- List of vehicles needs to be updated
- Cost of hourly hire charge needs to be updated

Alpha Testing

Alpha testing is performed to identify all possible problems before releasing the product to the client. The aim is to carry out the tasks that a typical user might perform. Alpha testing is carried out in a lab environment and usually, the testers are the software developers. This kind of testing is called alpha only because it is done early on, near the end of the development of the software, and before beta testing.

The developers then test the software for all design specifications. It is difficult for developers to test for ease of use, because having developed it, they already know how it works. The other three specifications can be tested by the developers.

Beta Testing

Beta Testing of the software is performed by "real users" of the software application in the context it will be used. Beta testing can be considered as a form of external User Acceptance Testing.
The Beta version of the software is released to a limited number of end-users of the product to obtain feedback on the product quality. Beta testing reduces product failure risks and provides increased quality of the product through customer validation. It is at this stage that data can be collected on how easy it is to use.

It is necessary to document the results of all testing. A test report is a formal document of all testing during both alpha and beta testing stages. It records data obtained from an evaluation experiment in an organized manner. It describes the environmental or operating conditions, and shows the comparison of test results with test objectives.

In the figure 3.14 below, you can see that each test has an ID, a date, who is doing the testing and whether the software passes or fails the test. More information is collected on the severity of the defect and a summary of the problem. The problem needs to be solved at some stage during development and must be signed off when it is fixed.

Fig 3.14

SYSTEM TESTING

Test ID	Test Date	Tester	Pass / Fail	Defect	Comments	Signed
1	12/ 10/ 22	Rakatairi	Fail	Runtime error	Check boundaries for discount	X
2	19 /10/ 22	Rakatairi	Pass		Discount working	✓

You can see in the System Testing table that Ms Rakatairi is testing the system on two dates. On the first date there was a Runtime error. A note was made regarding what needed to be checked. Later the system was tested again, passed and was signed off.

Test Data

Test data should include every possible type of input, both valid and invalid. Sometimes valid input can be unanticipated or unexpected. When the test data is comprehensive, your testing will highlight sections of the software solution where errors are likely to occur. With our Car Hire problem, we used the test data "4, 5 and 6". These tested the decision condition:

```
IF (Days > 5) THEN
```

Valid data that could be unexpected by the software system might include 50000 or 500. The software can still process this number of days despite the needs of the customer or company. What if the customer inadvertently entered some letters or a decimal number? The software needs to be able to handle all kinds of possible inputs. A full table of test data for the Car Hire Solutions could include:

0, 4, 5 ,6, 10, W, *^, 5000, Three

Validation

To ensure your software solution is able to cope with valid and invalid data, validation needs to be incorporated at each point where data is input into the system. The three main validation techniques, include existence checking, range checking and type checking. Validation ensures data is reasonable but not necessarily accurate.

To assist the user to enter valid data there are interface elements that can limit errors such as masks, pull-down menus and calendar controls. A mask illustrates what form the input data should take. A common example is --/--/---- for a date input box. A pull-down menu (called a combo box in VB) limits the user input to a pre-written list of options. Calendar controls are an easy to use interface object that makes selecting and entering date and time data a lot easier, but it does not stop incorrect data from being entered.

Existence Checking

Existence Checking tests to see if the data has actually been entered or exists in the location where it is searching. For example, if the user has not entered their name data into the "Name" field of the input box then the form cannot be processed. Validation would alert the user to enter data that has not been entered.

An example of existence checking in Visual Basic:

```
IF (strName = "") THEN
    MsgBox("Please enter your Name")
EndIF
```

Range Checking

Range Checking tests to see if the entered data is within a set range. For example, a post code in Australia consists of only 4 values. Postcodes in NSW all start with "2". If NSW is a restriction to the postcode, then a range check can test for input to be between 2000 and 2999. A common range check exists with date of birth data. This type of range check can check if the user's Date of Birth makes them eligible, over 18, or to test if the data is valid. Take care not to make your range checking too strict. Anyone entering a year over 100 years may be unusual but not necessarily inaccurate.

```
IF (Age>=18) AND (Age<= 120) THEN
    Form2.Show()
ELSE
    MsgBox("It appears the age you entered is not within the allowed range.")
ENDIF
```

Here is an example of validation for a length of values that can be used in a telephone number.

```
Dim strAllowedChars As String = "0123456789() -+"
If Len(txtTelephone.Text) = 0 Then
    Exit Sub
Else
    For i = 0 To Len(txtTelephone.Text) - 1
        If InStr(1, strAllowedChars, txtTelephone.Text(i)) = 0 Then
            MsgBox("Invalid telephone number.")
            txtTelephone.Focus()
            exit sub
        End If
    Next
End If
```

Type Checking

Type Checking tests to see if the data type is appropriate for the processes. Setting a Type Check on a postcode can further limit data entry to values, returning a warning if letters or other symbols are entered.

```
Dim intAge = txtAge.Text
If Decimal.TryParse(txtAge.Text, intAge) Then
        MsgBox("Please enter and integer for Age.")
EndIf
```

Another way to manage checking for type is using MyCheck:

```
Dim intAge = 32
Dim strName = Bob Brown
MyCheck = IsNumeric(intAge)   Returns True.
MyCheck = IsNumeric(strName)  Returns False.
```


Recording Project Progress

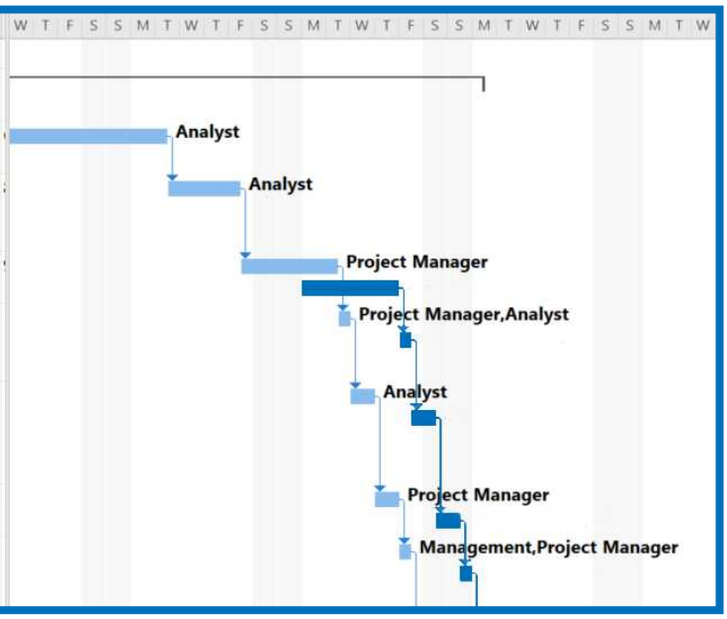
Gantt Charts

The most important document for planning a project and recording its progress is a Gantt Chart. Once the outline has been made in the Analysis stage, key goals and milestones should be clearly identified and located on a time-line. The role of the Gantt Chart is not to remain static, but to be adapted with variation in times, resources and scheduling. It is important that the project plan is monitored throughout the life of the project. When changes are made to the nature of tasks, annotations need to be included so they can be tracked. Project Managers should keep a log of all actions and changes to the plan with dates and team member identification. As part of the SAT it is advisable that the Gantt Chart is updated, and annotations are made throughout the life of the project, so that all logs are available for submission at the end of the project for the final Evaluation Stage. As the project progresses some deadlines may need to be moved for a variety of reasons. When a deadline is moved, the Gantt Chart should be updated with the original plan still available to be viewed. Although it is possible to create a Gantt Chart in a spreadsheet application, the premium software is MS Project. There are also very good free Gantt Chart software packages available, such as "GanttProject" (www.ganttproject.biz). Most specialised project management software allows for the original plan to remain, while updates to the project plan can be made and viewed at the same time. Each time an update is made ,a report can be exported and stored in a record of the project's changes. Gantt Chart software allows for tracking where each task can be identified as 25%, 50%, 75% or 100% completed. If a task is started late, the software can show both the actual tracking of the project with the original plan. In figure 3.15 you can see that the "Preliminary Budget" needed to be moved to after the weekend which affected the following dependent tasks. It is important that all changes remain clear on your final plan.

Fig 3.15

Updated Gantt Chart

Name	Duration	Start	Finish
Scope	**3.5 days**	**Thu 3/10/19**	**Tue 8/10/19**
Analysis/Software Requirements	**14 days**	**Tue 8/10/19**	**Mon 28/10/19**
Conduct needs analysis	5 days	Tue 8/10/19	Tue 15/10/19
Draft preliminary software specifications	3 days	Tue 15/10/19	Fri 18/10/19
Develop preliminary budget	2 days	Fri 18/10/19	Tue 22/10/19
Review software specifications/budget with team	4 hrs	Tue 22/10/19	Tue 22/10/19
Incorporate feedback on software specifications	1 day	Wed 23/10/19	Wed 23/10/19
Develop delivery timeline	1 day	Thu 24/10/19	Thu 24/10/19
Obtain approvals to proceed (concept, timeline, budget)	4 hrs	Fri 25/10/19	Fri 25/10/19



Action Logs

A project manager needs to keep records of the progress of the project as well as communications within the team. Team meeting minutes are important to record to ensure that everyone is later reminded of what was discussed and agreed upon at that meeting. Common things included in an Action Log might be:

- Ask teacher about how to test the validation on the password check subroutine
- Test both correct and incorrect data for record search
- Ask client for images of products to be included in database

Action Logs are "To Do" lists to keep the project on track. A project manager might allocate actions for specific members of a team. It's a good way of keeping records of each team member's responsibilities and their progress. It is also a way of documenting your work-flow in a formal way. Action Logs also include check lists and dates when things were completed. They can be used in conjunction with the Gantt Chart.

Pros & Cons of Development Models

When approaching a large task and monitoring your progress, it is likely that you will instinctively find yourself using the Waterfall Model. This uses the approach of completing one section then moving to the next section without re-visiting anything completed previously. This model works fine if the project is simple. There are three models outlined in chapter 2, and it is important that you consider how best to organise your time in the development stage by choosing a development model that would best suit your project. In figure 3.16 a comparison between the models is made. You can make use of this in your justification for choosing your model for your project. Key issues to consider when deciding on a development model includes:

- The order and number of the tasks to complete the project.
- The critical path of key tasks to define the work flow from one milestone to the next and their dependencies.
- The length of time allocated for each task is going to affect the success of the project.
- The scope, size and complexity of the project.
- The expertise of the development and management team.
- Allocation of appropriate human, hardware and software resources.
- Incorporating a contingency plan if there is an issue during part of the project's development.

Fig 3.16

Development Models

Development Model	Positives	Negatives
Waterfall	Easy to follow and understand, quick and manageable for small to medium sized projects.	Inflexible, requirements cannot be adjusted once the process has begun. Not suitable for complex or large products.
Agile	Client communication and feedback clarifies requirements. Flexible. Rapid delivery. Ideal for large complex problems.	Flexibility can use up more of the allocated time. High level of skills required.
Spiral	Highly flexible, fast development, suitable for large and complex solutions, risk management, requirements can be changed.	High level of skills and training required. Can be overwhelming to return an entire product in one cycle. More expensive.

EVALUATION

When evaluating a solution, we need to return to the evaluation criteria listed in the design stage that reference each of the requirements. In figure 3.14, Evaluation Criterion 1 (EC01) is listed in blue below the related Functional Requirement (FR01). It is at the evaluation stage that strategies need to be identified. These strategies are methods of evaluating the solution. To ensure that the application being evaluated contains all the 'choice selections' required there are three strategies that can test the efficiency and effectiveness of the requirement. The first is alpha testing and validating all 'choice selections' are included. The second is an observation of the client checking the 'choice selections' included and noting any issues.

Evaluation Criterion 2 (EC02) is listed below the related Functional Requirement (FR02) where multiple selections of any quantity are added to an order. The requirement also includes that this process needs to be a quick and easy process. There are four strategies identified. The alpha testing, the observation of an end-user, the measurement of time of an end-user order process and the interview with an end-user after they tested the solution.

Fig 3.14

No.	Requirement	Strategies	Evaluation Criteria
EC01	(FR01) Interface of the Application allows and allows for their selected products to be quickly and easily added to their order.	Provide data and instructions to an end user and observe their interaction with the software solution. Interview the end user about their experience with the solution.	**Efficiency:** The new software provides a faster solution to add products to an order than the original solution.
ECO2	(NFR02) Interface of the Application is attractive and easy to use.	Provide end users with a list of tasks to complete using the new software. Survey the participant end users regarding ow they feel about the interface attractiveness and ease of use.	**Effectiveness:** The new software has and interface design that rates 4.5 of the Likert Scale for attractiveness and ease of use.

Strategies for Evaluating Efficiency & Effectiveness

There are three ways to evaluate efficiency of a software solution:

- Measuring the speed of processing by timing how long it takes to complete a task without an infeasible delay.
- Observe end-users attempt to complete key functions on the solution. Interview the end-users on their perceptions of how well it does what it was designed to do.
- Measuring the cost of file manipulation is key to understanding how many times the data needs to be dealt with. It is possible to observe and count how many times files are manipulated, or the number of times the user must deal with a data point. Minimum manipulation increases efficiency.

There are many ways to evaluate effectiveness of a software solution:

- To evaluate completeness, users or clients can be observed checking the solution and interviewed regarding the completeness of the solution. Does the user need anything outside of the system to do the task?
- It can be confronting to evaluate attractiveness. End users can simply be asked if the design elements of the solution are appealing.
- Clarity and readability are vital and very easy to evaluate. Observing end users try to learn the new system will demonstrate how clear it is for users to read to complete a task using the system.
- Evaluating accuracy largely takes part in the testing phase, but with interoperable aspects of complex systems, it is not until the new software is put in place that accuracy is fully tested. Is the output from the system always correct for a wide range of test data?
- Accessibility can be evaluated through the use of the software by various people, devices and situations. Is the interface easy to use for all the users? The design of the interface needs to pay attention to the characteristics and limitations of end users in context.
- To evaluate timeliness, observe the work-flow of the end user and identify when data and information is required. Is the entry, processing of data and the production of output information completed in a time frame that allows practical use of the output?
- Ultimately usability must be evaluated by the client and end users, where they are observed with and without instruction. Given the characteristics of the end users, how easy is the solution to use? End users can be interviewed or surveyed for a wide range of feedback.

Quality Assurance

Quality Assurance is achieved after auditing the solution to rigorously reduce vulnerabilities and risks. Data must be kept secure and processed data must be accurate. Once the auditing of all aspects of the software is complete, the software should be more efficient and secure.

Acceptance Testing

Once Quality Assurance auditing is completed, acceptance testing is implemented before the software is released. The software is used by the end users in the context of the workplace. Set scenarios that test all functional and non-functional requirements are run through the software. The software solution achieves either a pass or a fail.

Usability Testing

Now we have developed our evaluation criteria, our strategies and our collection tools, we are ready to conduct usability testing. Usability testing is the practice of testing how easy a design is to use with a group of end users. It usually involves observing users as they attempt to complete tasks. It is often conducted repeatedly, from early development until the software is ready to be implemented.

In reporting on the usability of the software there are three main aspects that need to be included:

1. The evaluator must prepare and conduct a comprehensive usability test (using evaluation criteria) that covers all targeted requirements of the solution.

> These are our strategies outlined on pages: 46-47 and page 79. It is crucial that all requirements are tested. You must provide all the testing resources used to report on in your Usability Report. This will include outlines of the strategies and the design of the collection tools.

2. Document a comprehensive set of the results of the usability tests.

> These could be presented as tables of data from survey or interview questions, summaries of feedback, lists of observations.

3. Document a comprehensive set of the modifications to be implemented to the software solution.

> It is assumed that your solution is not perfect and some modifications are required. it is important that a high level of detail is included when listing the changes required to the software to meet the standards in the evaluation criteria.

Solution Evaluation

Once the usability testing is completed, a fuller evaluation can be made of the completed solution. To do this a few things need to be addressed. What has the usability testing revealed? Investigate the needs of the users by analysing the results from the usability testing. Be aware of exactly what the end users are trying to accomplish in using this software solution. The usability testing will have produced a list of required modifications. These need to be implemented before solution evaluation can go ahead. What are the constraints that will limit the modifications to the software?
In reporting on the evaluation there are four main aspects that must be documented:

1. Identify a comprehensive list of strategies for evaluating both the efficiency and effectiveness of the software solution.

> All the methods of data collection, the tools and the strategies used must be outlined in the evaluation report. How each method of evaluation relates to effectiveness and efficiency must be included.

2. In terms of efficiency and effectiveness identify how all specific features of the software solution meet all functional and non- functional requirements. All the evaluation criteria must be referenced.

This is best done in a table where all requirements are listed against the evaluation criteria in terms of efficiency and effectiveness.

3. Identify how the selected development model assisted in the development of the software solution.

Outline the model you used and identify the advantages using it and the disadvantages. Compare the model you used against the other two models also listing their advantages and short-comings in terms of your project.

4. Provide all evidence of critical and creative thinking through the evaluation of the analysis, design and development stage and the identification and description of improvements to the software solution.

Here is an opportunity to publish all your project logs, your Gantt chart updates and your idea development processes and documentation. In your report you must provide:

- reflections on how your analysis tools fed into the design process,
- evidence that you used idea development techniques to show how you solved design and development problems,
- logs of decision processes during the development stage.

Project Evaluation

Finally you have evaluated your software product and may have some reflections on how processes and decisions that assisted you in getting to this point. It is at this stage we need to reflect on the project structure as a whole. If you have been keeping up to date records of all the decisions, ideas and changes made throughout the project, you will be able to use your records in reflecting on the quality of the project management.

In the final criteria of the SAT, you will be asked to provide comprehensive documentation of all adjustments to the initial project plan during the project using a range of appropriate techniques. When the criteria identifies a "range" of techniques, you will be expected to provide evidence that you have monitored, modified and annotated project plans as necessary and proposed and implemented procedures for managing data and files. You might like to use the features in Microsoft Project to show the updates, or keep separate records of the Gantt chart as this are changed and annotated.

Finally you must produce comprehensive documentation of all the factors that contributed to the effectiveness of the project plan. This would include all annotations to the Gantt Chart, decisions made of changing start and end dates as well as the development model. If you found that you did not follow you identified in your project plan, then it should be evident in your record logs, and you should be able to write about the benefits of the model you decided on. This is an opportunity to write about:

a) What went well in your project and why.
b) What did not go well and why.
c) If you had the time over again, what would have been a better way to run the project.

Unit 4 Outcome 1
Development & Evaluation
Review Questions

1. How is Authenticity a different aspect of data integrity from Accuracy and Correctness?

2. Describe how Incremental and Differential back-ups differ.

3. What kind of back up storage media will you use for your SAT? Justify your answer.

4. Name three common application architectures.

5. Provide an example use for FTP.

6. What common name is given to Cat 5/6 cable?

7. What role does a router have in a network?

8. A wearable device collects data on the wearer's activity such as number of steps and their heart rate. The steps are collected throughout the day and added to provide a total. Heart rate is collected once per minute throughout the day. The user can transfer the data onto another device to access the data, but login credentials are required. Once the data is transferred, they can be processed by the app to store and summarise the user's activities and health.
   Name two kinds of data structures that could be used for this application.

9. How is a 2D array different from a Dictionary?

10. Provide an example of when you would use a Stack and when you would use a Queue.

11. How does the use of a hash table protect passwords?

12. Describe how the organisation of records can affect the interoperability of the software.

13. Give an example of an Instruction and a Function to illustrate how they differ.

14. Name three measures of efficiency.

15. Name nine measures of effectiveness.

16. The pseudocode below was designed for the athletics coach to type in 10 athlete names and their highest high-jump heights. It finds the names of the athletes that can jump over 2m in height. Create a trace table to find the errors in the algorithm.

```
START
BestJumpers(2, 1)
AllJumpers(9,1)
FOR i = 0 to 9 DO
    Read in JumperName
    Read in Height
    IF Height = 2 THEN
            BestJumpers(0, i)=JumperName
            BestJumpers(0, i)=Height
    ELSE
            AllJumpers(i, 0)=JumperName
            AllJumpers(i, 1)=Height
NEXT
END
```

	Data
John Peters	1.83
Jisha Singh	2.00
Sare Pascoe	1.92
Mike Smith	2.05
Angus Carter	2.09
Nish Quinn	1.98
Eliza Wallace	1.89
Natalie Meissner	2.10
Rose Saul	1.99
Averil Dillon	2.11

17. Describe how a data overflow can occur as a runtime error.

18. Design a tool that can sign off testing the code in Question 16

19. What feature of a Gantt Chart allows developers to reflect on the effectiveness of their plan?

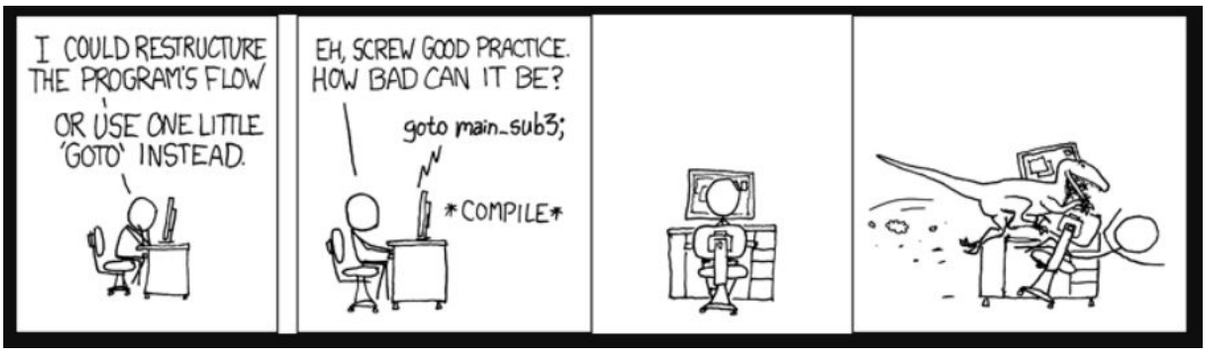
20. What is an Action Log?

21. Which development model will you choose for your SAT? Justify your answer.

22. Name three strategies to evaluate the code in Question 16.

23. Name three evaluation strategies.

24. Why does software need to be audited?



https://xkcd.com/292/

Online Support

Managing Version Control of your SAT can be problematic without GitHub! Use online testing examples to help you document your testing process.

- **SoftwareTestingHelp.com/Resources (Plenty of testing templates)**
- **guides.github.com (How to use Git Hub)**

https://xkcd.com/2109/

Unit 4
Area of Study 2

Cybersecurity

On completion of this unit the student should be able to respond to a teacher-provided case study to examine the current software .[development security strategies of an organisation, identify the risks and the consequences of ineffective strategies and recommend a risk management plan to improve current security practices.

Key knowledge

Digital systems
- physical and software security controls used to protect software development practices and to protect software and data, including version control, user authentication, encryption and software updates
- software auditing and testing strategies to identify and minimise potential risks
- types of software security and data security vulnerabilities, including data breaches, man-in-the-middle attacks and social engineering, and the strategies to protect against these
- types of web application risks, including cross-site scripting and SQL injections
- managing risks posed by software acquired from third parties

Data and information
- characteristics of data that has integrity, including accuracy, authenticity, correctness, reasonableness, relevance and timeliness

Interactions and impact
- reasons why individuals and organisations develop software, including meeting the goals and objectives of the organisation
- key legislation that affects how organisations control the collection, storage (including cloud storage) and communication of data: the Copyright Act 1968, the Health Records Act 2001, the Privacy Act 1988 and the Privacy and Data Protection Act 2014
- ethical issues arising during the software development process and the use of a software solution
- criteria for evaluating the effectiveness of software development security strategies
- the impact of ineffective security strategies on data integrity
- risk management strategies to minimise security vulnerabilities to software development practices.

Key Skills
- analyse and discuss the current security controls to protect software development practices
- identify and discuss the potential risks to software and data security with the current security strategies
- propose and apply criteria to evaluate the effectiveness of the current security practices
- identify and discuss the possible legal and ethical consequences to an organisation with ineffective security practices
- recommend and justify an effective risk management plan to improve current security practices.

Cybersecurity

Cybersecurity is the practice of protecting digital systems that store or share data over networks. Human lives are reliant on digital technology in all aspects of society. Financial institutions, medical and educational organisations all rely on networked digital data. Any device that is network enabled or any organisation that relies on networked data is vulnerable to data integrity being compromised. This can affect how efficiently and effectively the system performs. Cybersecurity ensures that systems with sensitive data meet legal obligations to keep data private and safe.
Cybersecurity involves both physical and logical controls to protect data and information systems. Physical controls limit access to the data via a barrier such as a locked door to a server room preventing unauthorised access. Software controls are based on logical limitations such as the use of login details and passwords.

Physical Security Controls

An organisation such as a school uses a complex networked information system that contains sensitive data which includes student and staff contact details, student school results, staff pay information and bank details.

Barriers

If you think about the physical security that exists around your school, you might have noticed:

- A fence or wall barrier around the school to limit access to the school.
- Locked doors on staff rooms with computer access.
- Locked doors on server rooms and IT administration staff rooms.
- Locked down laptops in class computer trolleys.
- Some staff may have special keys, PINs or swipe cards to get into computer labs, limiting who can access those rooms.
- Security cameras can provide video information if damage has been done to devices or other equipment.

Biometrics

Biometrics are methods of physically limiting who has access to a room or device. These include thumb print scanners, iris scanners and voice recognition. This is a more expensive security option and is often used in larger organisations.

Security Appliances

Security appliances are stand-alone devices that can be used as part of a security solution for an information system. No single device can protect a system from all the possible risks. There are a few types of security appliances such as routers, firewall devices, intrusion protection system (IPS) devices and virtual private network (VPN) devices.

- Routers that include integrated services (ISR) can include other capabilities such as intrusion prevention, encryption and virtual private network (VPN).
- Firewalls (such as Cisco Next Generation Firewalls) are devices that also can include router capabilities, network management and analysis.
- There are other security devices that provide VPN services, encrypting all client data exchanges. Some Cisco devices that can perform server or host computer roles in a network come already installed with anti-malware capabilities, web and email security, decryption and client control services.

Back-Up

If the system or work place is affected where the data is corrupted or lost, back-up copies can be kept off-site in a locked, fire and water proof container.

Software Security Controls

Software security is often called logic controls because the barriers to the system are not physical, but coded into the system. Some are easily experienced by the user such as user ID and password requirements to access the networked computing system while some security controls, that you are unaware of, are quietly doing their job behind the scenes.

User Authentication

A logic method of protection used ubiquitously is the personal user ID with a password. This allows the system to identify the person accessing the computer network. All students and staff at your school will be issued with unique ID numbers that you need to access the network. Unfortunately, passwords can easily be compromised especially when people are not careful, keeping their passwords written down or accessing websites that are not secure. Banks and other large organisations have begun to implement two factor authentication. This adds another level of security to a networked password access. When a profile for a user is created on a networked system, another method of communication such as a mobile phone number or external email is also stored. The system responds to the user's ID and password with a direction to check their email or phone. The system sends the user a code via their email or phone which the user must transfer into the system login screen. This ensures the person who is entering the ID and password is also able to access other authenticated systems.

Firewalls

A firewall is a software or hardware system that controls and monitors the traffic in and out of a network to authorised access only. A firewall can also control outward traffic too, limiting what users of the network can access online. You may have noticed this at your school where the school firewall can stop students from accessing gaming websites. A firewall is mostly protecting a system from hackers entering the network to do harm to the system or data stores.
There are a number of different types of firewalls:

- Network and Transport Layer Firewalls filter IP addresses, sources and destination data ports of a data packet as it moves through. This limits access from and to potentially malicious locations on the internet.
- Context Aware Application Firewalls and Application Firewalls check executable code and who is using it. This can limit the intentional or accidental distribution of malicious code. You may sometimes encounter it when you are trying to download a file at school that contains a macro – such as a spreadsheet file.
- Proxy Servers filter website requests by investigating and comparing URLs (web addresses) against a list of denied addresses. A proxy server can run to protect hypertext transport protocol (HTTP) web access or Simple Mail Transport Protocol (SMTP) email. It searches for key strings in searches and results to limit access to potentially malicious data.

Anti-Malware

Anti-Malware software can protect networked systems from a wide range of malicious software. Any software that can bypass controls, cause harm, compromise or take sensitive data is considered malware. Some anti-malware is designed to target specific types of malware which we will explore in more detail under the topic of "Risks".

Encryption

Encryption is used when data is transferred over an unsecured network such as the internet. Data can be stored encrypted also, but a key needs to be used to read the data. You can encrypt your data using the Windows feature Encrypting File System (EFS). It is linked to your account profile on the network. The process of encryption is the application of a cypher or algorithm to convert the file into an unreadable form. The file can be read only by persons with a decryption key which converts the unreadable file back to its original form. Encryption is an important security measure to protect data as it is transferred across networks.

The process involves taking your original message: for example:

"I have the microfilm, the hand over will take place in the car park"

and applying an encryption key. For example, we replace all the vowels with prime numbers and reverse all the words and replace the spaces with vowels:

"29av13h7e31htim41rc13f3lmo2htud13nha29r11vell29wik41t13o11l29cpun13ath2er31cik2rp"

This string is now impossible to read. When an encryption key is created, a Decryption key must also be developed to reverse the process back to the readable message. This is called a symmetric key encryption. Symmetric encryption can be vulnerable because often this key needs to be shared or sent which can compromise the security if intercepted.

Alternatively, an Asymmetric key encryption uses two keys - one is Public (used to encrypt) while the other is Private (used to decrypt). This is ideal for small amounts of data. Asymmetrical encryptions occur when a company wants clients or customers to communicate with them using a public key. Anyone can encrypt their message with the public key, but only the company holding the private key can read the messages.

End to End Encryption is applied in the encrypted messaging mobile phone application WhatsApp. WhatsApp users communicate through instant messaging or using phone calls over the internet. Their data passes through a WhatsApp server while transmitting from one user to the other. For many other services that offer encryption, the data is encrypted during transfer but is protected only from outside intruders like hackers. End-to-end encryption, however, keeps the data encrypted, without any possibility of decryption, even at the server and everywhere else. Thus, even if they want to, the service cannot intercept and do anything with the data. Law enforcement authorities and governments are also among those who cannot access the data, even with authorisation. Theoretically, no one can, except the parties at the two ends.

Version Control Systems

As a software student you may have managed your own version control system where you have created a file called "Transaction_System" then you created a new version called "Transaction_System2" and inevitably you will have a "Transaction_SystemFINAL" and then "Transaction_SystemFINAL2". This is a very basic approach to version control systems, but when you have thousands of files, users and systems, this type of control will not scale up. Software developers use software that monitors changes, stores the changes, the times they happen and who made the changes.

Version control systems (VCS) track changes to files or applications stored in the information system especially where many people can access and edit those resources. VCS can limit who can access or modify aspects of software or data. A commonly used VCS is a content management system used on websites. This manages the content on the website, updating and controlling text, as well as ensuring only authorised editors can access and update the content.

If incorrect updates are made, the VCS can retain information about the update and roll-back the data before the update, removing any errors. Sometimes a student at your school may accidently delete some very important file. A quick visit to your network administration manager, who has access to the VCS, can implement a roll-back of the student's profile on the network. This can mean the data stored in the student's profile is returned to the state it was in before the error was made.
Software developers use Git to manage the different versions of their builds. You can find it at the website: https://github.com/ Version control software allows developers to work in teams and it facilitates the smooth and continuous flow of changes to the code being developed. Git is the most widely used version control system.

Software Updates.

You may find software updates a bit annoying. They interrupt what you are doing and often want you to restart your device. If you don't update your software, you are leaving your device vulnerable to malicious attack.

When software is released, hackers investigate weaknesses in the software to take advantage of any device or user profile they can access. Hackers create malware that can use software vulnerabilities to spread their viruses and exploit users of the software.

The developers of the software develop patches to fix these vulnerabilities as they come to light. For example, a new version of an operating system is released and within a day or two, the developers of the system are getting reports that users are experiencing issues with viruses accessing their machines through the new system. Developers then investigate how the virus is using the new release and create an update for the operating system to ensure the virus can no longer exploit it again. The update is then released to users of the operating system.

It is always important to ensure update installations are a priority in setting up security protocols. Common software updates that need to be kept updated are:

- Operating systems
- Anti-malware
- Security software
- Browsers
- Web plugins
- Applications.

Deleting Data Permanently

In the digital world, delete does not mean delete. If you have sensitive data on devices that are being replaced and disposed of it is important that the hard drives are processed to ensure the data is permanently erased. Data deleted in the normal way can still be accessed from hard drives. Nothing is permanently deleted without a series of processes that include:

- Overriding the content with binary sequences. It is possible to miss some data which remains a risk.
- Degaussing the drive with powerful magnets so they can be re-used.
- Physical destruction of the drive using industrial shredders.
- Some solid-state drives have a built in "sanitise" command but there is no guarantee that all previous sensitive data has been completely erased.

Software Auditing and Testing

When developing a software solution, it is important to review a wide range of possible vulnerabilities built into the software. The process of investigating the quality of the software, its adherence to regulations and security requirements is called auditing. Auditing is usually done by persons external to, and independent of, the development process. An audit can simply investigate the compliance of a software product to an organisation's regulations, but this may overlook some weaknesses in the product that could be detected in more substantive testing.
Auditing the security of a software product has a three stage approach:

- Preventive: Attempt to predict potential problems before they occur and make adjustments to the software.
- Detective: Use controls that detect and report the occurrence of an error, omission or malicious act.
- Corrective: Minimise the impact of a threat by resolving problems discovered by detective controls, identify and correct the errors, and modify the processing systems to minimize future events.

To ensure a software product is secure, a few vulnerabilities need to be investigated.

Non-Validated Input

Software requires data input so it can be used. When data entered into the program is not effectively validated, persons with malicious intent can take advantage of this weakness to force the software to be used in an unintended way. Thorough testing of all data entry validation is an auditing strategy.

Buffer Overflow

All software requires allocated areas of memory. This is called buffer. If data is allowed to be written beyond the limits of the buffer, this can allow an attacker being able to run an "arbitrary code exploit". This allows an attacker to run any code they want and take over the device running the software. Some programming languages, such as C, are more vulnerable to buffer overflow than others. VB thankfully does not require special checks as it is immune. Common unsecured practices come from handling strings without validating and checking the boundaries on the size of the string in bytes. This is how extra bytes can overflow the buffer boundary.

Race Conditions

Race condition vulnerabilities arise when there are two or more operations are forced to perform simultaneously. The attacker exploits a small time gap between the moment the operations are initiated and when security controls wake up to start checking for intruders.

Weaknesses in Security Practice

Software developers may feel they should also create their own security algorithms to protect their code, but in the industry, security algorithms are developed by specialist technicians. Specialty security-ware that authenticates, authorises or encrypts, that has been created, tested and verified should be considered before attempting to develop your own solution. Often software development companies have specialist developers to devise libraries of security algorithms that can be implemented in all their software products.

Access Control Problems

When a software product does not limit user access, it is vulnerable to someone editing the software to be used in an unintended way. Software products should restrict access. If administrators are required to access one level of data, then a 'permissions-structure' needs to be put in place to keep the software product secure.

Data Vulnerabilities

Sometimes there are other vulnerabilities in a system outside of the software product. These can be caused by a wide range of attacks such as malware. Malware is malicious software that is developed to cause harm to a digital system in some way. There are a variety of ways malware can cause data breaches, man-in-the-middle attacks and social engineering.

Types of Malware

Spyware is malware developed with the intention of tracking a user's behaviour, monitoring keystrokes, capturing data, using the camera, or tracking other activity such as internet access and software use. Spyware modifies security settings to remain unnoticed. It can be downloaded or shared accidentally by the end user, being bundled in legitimate software in the same way a Trojan Horse malware is distributed. The author of spyware can access the end user's banking details, logins and passwords.

Trojan Horse malware can infiltrate a system by being attached to a non-executable files such as an image or sound file. Trojans have also been distributed as part of free games downloadable online. This type of malware can use the access privileges of the end user to access sensitive data.

Viruses are malicious executable code that can be attached to legitimate executable files. They can be distributed through USB thumb drives or external storage drives, email or networked servers. Viruses have a wide range of application and can simply be harmless. Often they are designed to be destructive

so the virus developer can watch the extent of the damage their work can implement.

Malicious Autonomous Bots can be distributed to systems through self-propagation. Spiders, crawlers or web bots all work in a similar way, they recreate themselves to be distributed onto as many devices as possible. They report back to their creator. They can wait undetected until activation by their source and can exploit access to passwords, access to bank accounts, launch DoS attacks and open more vulnerabilities to more powerful attacks.

Worms are like bots, as they replicate themselves to exploit vulnerabilities in networks. Worms slow down traffic within a network allowing other malware to take advantage of the slow moving network to create more impact attacks that may include the destruction or corruption of files.

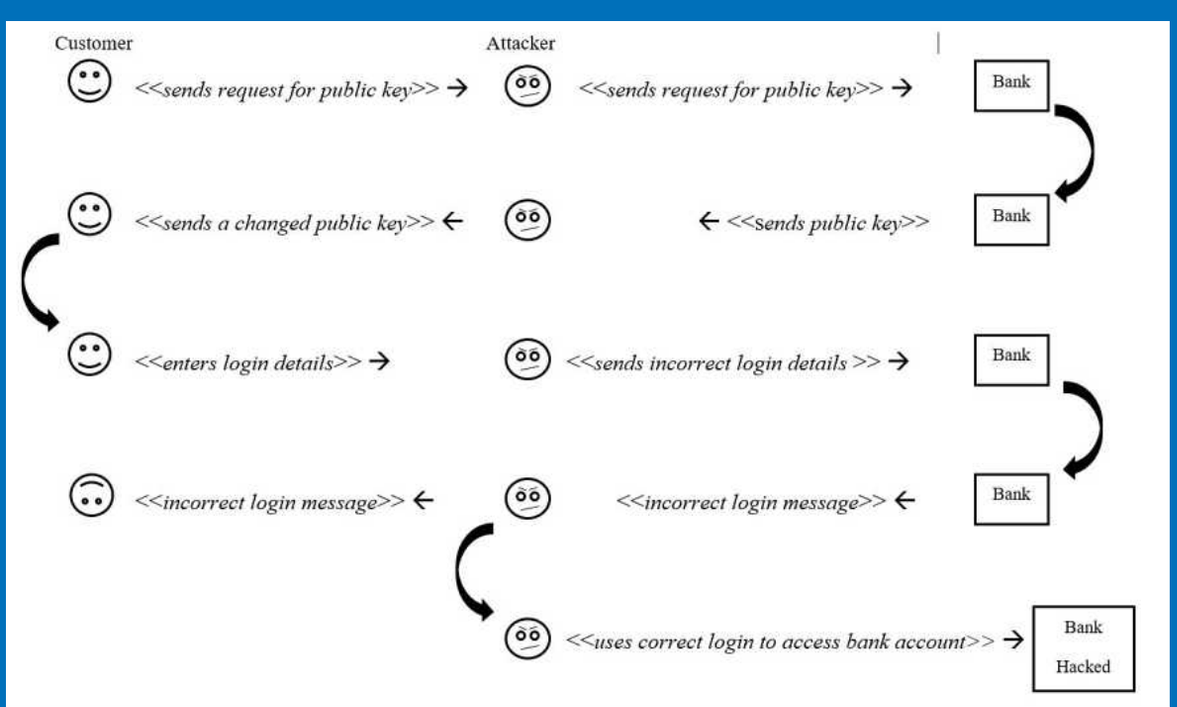
Rootkit is malware designed to modify the operating system to create a "back door". This allows unauthorised access to a system. It exploits software vulnerabilities by escalating access privileges to the device or system. Operating systems need to be wiped and reinstalled if they are infected with a rootkit virus. Keeping operating systems up to date and taking care online is best practice to secure yourself from rootkit attacks.

Ransomware is malware that encrypts all the data on a device and informs the user to make a payment for the key to decrypt the data. Ransomware can shut whole organisations out of their databases causing interruptions to operations and a lot of expense.

Man-in-the-Middle (MITM) attacks are made through the use of cryptography, where data is changed into an unreadable format with a cypher key. An attacker takes over a device without the administration's knowledge and uses encryption to change data transmitted between two parties. It allows the two parties to think they are communicating directly, but the messages are changed. A MITM attack can allow the attacker to listen or eavesdrop on communications, especially with financial organisations. This allows the attacker to steal financial records, encryption keys or other sensitive data. In figure 4.1 is an example of how MITM can intercept transmission between a bank and their customers.

Fig 4.1

Man in the Middle Attack



In figure 4.1 the customer is unaware that the attacker is sending them incorrect information. The attacker intercepts the communication to access the bank. There are three ways to protect from Man in the Middle attacks:

1. Network Security: Ensure network security through the use of virtual private network encryption.
2. Encryption: Use Wi-Fi encryption because WiFi is very vulnerable to hackers. Encrypt all hypertext transfer protocol communication online and use end-to-end encryption.
3. The use of malware protection software limits the power of hackers to access your system.

Social Engineering

The least secure aspect of a system is the human element. Humans make mistakes and poor decisions and hackers can prey on individuals' weaknesses to manipulate them. A phone call to an authorised person with an urgent network access issue could be convincing enough to get the authorised user to hand over login details. There are a few methods of social engineering.

Pretexting

Pretexting is the process of developing a believable scenario and contacting authorized people with system access. They use their story to encourage the person to provide them with sensitive information or access to a system. For example, an attacker could pose as a member of the organisation's IT support team in a call to an authorized person. They build trust with that person by offering assistance or sharing other information. Once the trust is established, the attacker asks for access to the building, the system or other sensitive information.

Phishing

Phishing is a very common type of social engineering that incorporates the use of emails. The email is sent to an unsuspecting person with a message that may incorporate a threat to encourage a reaction of urgency to act. This manipulates the receiver to act quickly and hastily such as click on a link to update antivirus software. A common type of phishing scheme is to send an email from a reputable organisation with a message saying their device has been infected. The receiver clicks on the link to download a solution to the problem and they are directed to a malicious site online. Spear phishing is an attack where a specific person is targeted. Information about the targeted person needs to be obtained, such as their interest in purchasing a particular product, such as a car. The attacker contacts the victim either directly or on a forum with information about the sale of a car. The victim opens an image of the car and inadvertently downloads and installs malware.

Baiting

Appealing to victim's interests, attackers entice users with promises of free products such as music, movies or games. In order to access the free item, users are asked to surrender login details for other services such as Uber or Google. This allows the attacker to access to the victim's accounts and even their financial records.

Something for Something (Quid Pro Quo)

This type of attack is very common. It involves attackers who impersonate IT service staff by contacting victims to offer IT assistance. The attackers make promises in exchange for the employee disabling their anti-virus program and for installing malware on their computers that appears as software updates. These updates direct the user to further compromise the system.

Minimising Potential Risks

There are many methods to ensure best practice to secure digital systems.

- Perform risk management assessments to provide system administrators with an idea how best to decide what security measures are required.
- Organisations usually have a security policy that clearly outlines staff behaviour to ensure that security protocols are not undermined.

- Put physical barriers in place to restrict access to sensitive digital systems.
- Research all users with background checks.
- Automate updates for all software, operating system, network devices and application programs.
- Put restrictions on levels of access for all users.
- Use integrated network monitoring, analytics and management tools.
- Use security devices such as routers, firewalls and intrusion prevention systems.
- Use anti malware software
- Educate the end users of the system to ensure secure use such as not opening emails from untrusted sources.
- Use encryption for all sensitive data.

Web Application Risks

Building websites has become highly complex, incorporating multiple coding languages to add functionality. Complex builds often contain databases of sensitive data that are targets for hack attacks.

Cross Site Scripting (XSS)

In the early years of the world wide web, web pages were static and made mostly of HTML code. Now websites are dynamic and interactive and use a wide range of coding languages and techniques. Most websites ask users to provide information to sign up for updates, join a group, or to log in for extra services. When the website contains forms for the user to input data, there lies a vulnerability for the whole site. People who can access the input form can type anything into the input boxes. When hackers enter code or scripting instructions into an input box, this is called Cross Site Scripting.

Adding malicious code into a web request is called injection. The most commonly used form of injection is database requests using SQL. There are two other types of cross site scripting: Server XSS and Client XSS. When a server XSS attack is performed, sensitive data is retrieved from the database through a hypertext transfer protocol request of the server. The data is unsanitised and you can see the results below of an SQL injection. Client XSS attacks exploit scripting such as HTML and JavaScript to hijack a user's session on a website by intercepting session cookies to obtain sensitive information in order to use the login credentials for unauthorised activities, phishing or distribution of malware.

SQL Injection (SQLi)

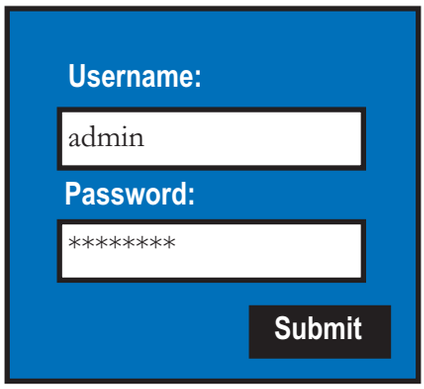
The easiest way to perform an attack on a web application is through the use of Structured Query Language (SQL). It is used for retrieving data from databases and is a type of Server XSS. If we had a database table of product names and prices called Product_Table we could use an SQL query to return the price of a product name entered into a form.

```
SELECT ProductName, ProductPrice
FROM Product_Table
WHERE (ProductName= "cherries")
```

The SQL query above is selecting field names "ProductName" and "ProductPrice" from the table "Product_Table" where the condition (ProductName = "cherries") is true. The query will find the price of the cherries and return it to the form for display.

If the web application input form is not successfully validated by excluding characters or removing direct access to the database from the web page, SQL can be typed directly in any web form to extract sensitive information. SQLi can be used where User_ID and Password are input into a web form. If the form is directly linked to the database server and an intentional error is input, the database can return details of the error that contain actual user names and passwords.

Below is an example of how an SQLi attack can collect sensitive data from a website database.



The input form uses the following SQL to check if the user name and password are stored in the data base.

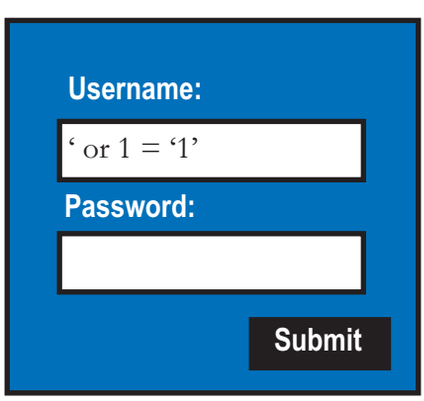
```
Username = getRequestString("UserName");
Password = getRequestString("Password");
SQL = "SELECT * FROM Users WHERE UserName = ' + Username;
```

The query reads in the two strings and selects all (*) fields from a table called **Users** where the field "UserName" must be the string read into the Username variable.

Now it's safe to say that there will be a username like "admin" in this Users table.

SELECT * FROM Users Where UserName = 'admin'		
UserID	**UserName**	**Password**
1	admin	secret123

Now if the form is expecting to read in an SQL query, a hacker can write a logic statement into the input form to return all the usernames and passwords from the database.



When an inverted comma is typed into a text box, it continues the coded SQL that has been written in the form. It can complete the query where the condition is: ' ' or 1 = '1'.

```
SQL = "SELECT * FROM Users WHERE UserName = ' ' or 1 = '1';
```

This will now return all data from the Users table where Username is null (empty) OR one equals one! One always equals one so it returns an error complete with the Usernames that are available in the database.

So the SQLi returns the following sensitive data to the screen

Failed to login as admin
Failed to login as moderator
Failed to login as guest
Failed to login as guest

Now the hacker knows the usernames, they can use the same approach to get the passwords.

SELECT * FROM Users Where Password = ' ' or 1 = '1'		
UserID	**UserName**	**Password**
1	admin	secret123
2	moderator	password123
3	guest	guest123
4	guest	guest123

Denial of Service (DoS) Attacks

Through the use of bots distributed widely on unsuspecting user devices, hackers can activate the bots to flood a website or network portal with network traffic requests. Since the requests are legitimate, the targeted server accepts all the requests until it can no longer process them. Sometimes bots send authentic requests with an invalid return address which causes the server to wait before closing the connection. When this occurs thousands of times per second, the server is no longer able to receive more requests because it is too busy, and the website can no longer be accessed by anyone. This technique can be used to shut down websites, slow down networks, target a particular person to limit their access to a service, or simply disrupt an online service.

Software from Third Parties

When developing software, on occasion it might be cost-effective to hire third party developers to solve one aspect of the build. Time constraints may mean that extra assistance is required to complete it on time. In some developments specialty skills, not held by anyone in the team, are required. Hiring a specialist can be time and cost effective.

When you acquire software from a third party the program manager needs to manage risks through the development of the project through:

- Clear legal agreements that outline ownership of the build and the terms and conditions of the partnership.
- Clear guidelines about standards outlining responsibilities and consequences if standards are not met.
- Clear and productive communication with the third party to ensure interoperability with the rest of the build.
- Clear project plan with milestones that are enforced.
- Ensuring the third party has the skill sets to deliver.
- Running quality assurance assessment.

Data Characteristics

When data is input, processed, stored, and output as information, it is important the quality of the data is ensured. Validation techniques and security procedures assist in ensuring that the data entered into the system usable.

Data Integrity is affected by various aspects:

- Accuracy
- Timeliness
- Reasonableness
- Authenticity
- Correctness

Accuracy vs Correctness

The source of the data (Data Set or Customer User) may input Accurate Data which may also be Correct, but this is not always necessarily the case. Here is an example. Julie has been handed some data to enter into the information system. She accurately transcribes the data by typing it in. Later it comes to light that the data she was provided was incorrect. Despite the data accurately entered, it still remains incorrect.

Authenticity

Data authenticity is related to the trustworthiness of the source. If UNESCO produced a Data Set on sea temperatures, you can be confident that the data is correct and it can be relied upon. The data has authenticity because UNESCO is a trusted environmental research body. The best method of ensuring authentic data is to collect it from primary sources. It is only in this instance that this method of the data collection and timeliness of the collection may interfere with Data Integrity.

Timeliness

It is common practice for data to be stored over a time to be investigated for patterns or correlations. However time can limit the integrity of the data if it is out of date, or collected at a point when the data will be affected by other issues indirectly.

Data that has a "use by date" can include:

- phone numbers
- addresses
- student progress results
- prices and preferences

The timing of data collection can also influence the integrity of the data. Consider a survey conducted to investigate community concerns about refugee numbers in Australia. If the survey is taken after a terrorist incident, this seemingly unrelated event can influence the responses provided. Many students find the use of online surveys produces poor data sets. Often respondents do not take the task of answering surveys seriously. When students do face-to-face surveys or interviews with peers, they find they get better quality answers.

Reasonableness

Reasonableness is an umbrella term for validity and relevance. Reasonable data needs to meet conditions such as ranges, data type or format. If the input field is 'date_of_birth' then only data containing 'date' formatted information will be valid. If the date entered is after the current date, it would not be relevant. Therefore, only date formatted input in a range before current date would be reasonable. It does not guarantee that it is correct.

Goals and Objectives of Software Solutions

Organisations make clear statements about their goals and objectives to assist them in making decisions, planning and creating strategies. Goals and objectives keep all new ideas, proposals and other planning issues on track to core business. It is essential that security is part of the goals and objectives in the implementation of a new information system.

- A Goal is a broad statement, that is aspiration in nature, about principles or targets the organisation may strive to achieve. The goal of new information system might be: "To provide a platform that encourages communication between departments".
- An Objective is a clear statement about measurable deliverables the organisation wants to achieve in supporting their goal. An objective of a new system might be: "To keep all personal records secure from unauthorised users".

Efficiency Objectives of Software

When an organisation or individual chooses to implement a software solution in place of a manual system already in place there are few reasons for doing so. Software can make decisions, process transactions and make calculations faster than humans. In the case of processing salaries, manual processes would take days, while a payroll software package can process wages within a few minutes. Software and databases make it faster to access the data required. Web applications make it faster to access and transfer data over large areas. When processes are faster, the business can utilize staffing to add value to their organisation by doing other operations. Example School Portal: A portal makes it faster for teachers to enter attendance so that missing students can be investigated as soon as possible by administration. When teachers want students to access resources, a portal can provide access via web application so students who are at home, can access the resources. When teachers mark student work, they can use rubrics built into the portal which makes marking quick and easy with a few simple clicks and the student gets the results immediately. Unfortunately, with more efficiency comes other vulnerabilities to the data. If the system is available on the web, then it requires extra security.

Effectiveness Objectives of Software

Software can function more effectively than manual systems. The following are key effective objectives of software solutions:

Eliminate Errors — Humans often make mistakes, while software, if correctly developed for the job, will do the right thing time and time again. Software can streamline processes and eliminate errors.

Scalability — A small scale business could easily scale up to a larger operation with the right software to support their processes. It has been argued that the development of the digital database is responsible for the increasing growth of large companies into major international conglomerates.

Integration — Software provides a wide range of integrated support. In some cases, different manual processes are seen as different problems. With the right software tools many processes can be solved with one solution. Here is an example. For many years a school portal at McFlys' Middle School only managed attendance, class portal pages and timetables. The report writing and results were done by teachers individually in applications Word and Excel. McFlys' IT department decided to create a reporting system that allowed teachers to enter their assessment results and create reports. They wanted the current portal system to share data with the new system so the teachers only needed to work with the one system. This is an example of integrated support.

Automation — Computers are excellent at automatically running processes such as calculations, categorising data, and accessing relevant information.

Security — With built in security measures, software can make it harder for unauthorised access to sensitive data. Manual systems can easily be accessed by just getting past the physical barriers to the data storage.

Legislation in Practice

Australian legislation determines constraints on software solutions. The development of software that manages, stores or uses sensitive data must adhere to the following legislation:

- The Privacy Act 1988,
- The Privacy and Data Protection Act 2014,
- The Copyright Act 1968,
- Health Records Act 2001,

The Privacy Act 1988

The Privacy Act (1988) regulates how personal information is handled. The Privacy Act defines personal information as information or an opinion, whether true or not, and whether recorded in a material form or not, about an identified individual, or an individual who is reasonably identifiable. Common examples are an individual's name, signature, address, telephone number, date of birth, medical records, bank account details and commentary or opinion about a person.
https://www.oaic.gov.au/privacy-law/privacy-act/

The Privacy Act includes thirteen Australian Privacy Principles (APPs), which apply to some private sector organisations, as well as most Australian and Norfolk Island Government agencies. These are collectively referred to as 'APP entities'. The Privacy Act also regulates the privacy component of the consumer credit reporting system, tax file numbers, and health and medical research.
https://www.oaic.gov.au/privacy/australian-privacy-principles/read-the-australian-privacy-principles

Australian Privacy Principles (APPs)

Australian Privacy Principle 1 — open and transparent management of personal information
Australian Privacy Principle 2 — anonymity and pseudonymity
Australian Privacy Principle 3 — collection of solicited personal information
Australian Privacy Principle 4 — dealing with unsolicited personal information
Australian Privacy Principle 5 — notification of the collection of personal information
Australian Privacy Principle 6 — use or disclosure of personal information
Australian Privacy Principle 7 — direct marketing
Australian Privacy Principle 8 — cross-border disclosure of personal information
Australian Privacy Principle 9 — adoption, use or disclosure of government related identifiers
Australian Privacy Principle 10 — quality of personal information
Australian Privacy Principle 11 — security of personal information
Australian Privacy Principle 12 — access to personal information
Australian Privacy Principle 13 — correction of personal information

How Organisations are Affected by the Privacy Act

Case Study: Wallan City Council collects rates from land owners each year to cover the cost of looking after roads, fixing water pipes and other services. They decided to keep a database of all the land owners and their contact details in a digital database so they can automate their rates invoices and have them sent out via email or by post. Some land owners do not live locally so the council decided to add an online component to their database to allow land owners to update their information online. While designing the solution the developers decided to offer Wallan Council the opportunity to allow rates payers to pay their invoices on the web site too.

APP1: Wallan City Council must inform the land owners that their data is being stored in a data base and they must include the purposes for which the data will be held.

APP2: Land owners may have the option to not identify themselves by using a pseudonym.

APP3: Wallan City Council can only solicit (collect) personal information that is relevant to the processes it required for. In this case, contact details and the location of the land within Wallan City Council.

APP4: Any unsolicited personal information about land owners must be destroyed within a reasonable time frame.

APP5: When data of a personal nature is being collected from the land owners, they must be informed.

APP6: If another organisation wants to know who owns a particular piece of land in Wallan, the City Council must obtain permission from the land owner to pass on their information.

APP7: A local business would like to use the Wallan City Council rate payer list to advertise a new product. The council must not disclose or use the personal contact information for direct marketing in this way.

APP8: A Chinese mining company wants to buy properties in Wallan. They approach the Council and would like the information about the land holders in the region so they can contact them to buy their properties. The Council can not disclose personal information to overseas recipients because they will not be bound by the Australian legislation.

APP9: One of the software developers suggested they use the land holder's drivers licence number as a UserID. This would be impossible because a driver's licence is a government identifier.

APP10: Each year the Wallan Council ensures that all the contact details of the land holders is accurate, up-to-date and complete by sending out emails, and letters to ask if corrections are required. If no response is received, follow up phone calls will be required.

APP11: It is important that the Council stores all personal data on a device that is physically locked with only authorised access. Access to the data via the system is restricted by a wide range of logical security measures. The data is kept safe from power surges, or accidental deletion using regular back-up procedures.

APP12: If a land holder wishes to access the data that is stored about them, the Council must provide it.

APP13: If a land owner is sent an incorrect rates invoice, the individual has the right to ensure the Council corrects any mistakes on the record.

The Privacy and Data Protection Act 2014

The Privacy and Data Protection Act is legislation that covers the handling of all personal information, other than health information, as well as covering protective data security, in the public sector in Victoria. This legislation defines Information Privacy Principles (IPPs) to identify how Victorian law enforcement will respond to breaches. The IPPs are directly related to the APPs.

Copyright Act 1968

In Australia, copyright law is contained in the Commonwealth Copyright Act 1968 (Copyright Act). Copyright is the legal ownership of creative works such as text, artistic works, music, computer programs, sound recordings, photographs and films. The rights to use creative works are granted exclusively to the copyright owner to reproduce the material, and for some material, the right to perform or show the work to the public. Copyright owners can prevent others from reproducing or communicating their work without their permission or may sell these rights to someone else.

In Australia, copyright protection is automatic. There is no need for copyright registration in Australia, nor is there a legal requirement to publish the work or to put a copyright notice on it. A work will be protected as soon as it is put into material form, such as being written down or recorded in some way (filmed or recorded on an audio tape).

In relation to software development a programmer may be paid to create a software solution, and would need to negotiate the copyright ownership of the software, as it may contain content regarding the owner's business. It is important that if your client owns the copyright for a software solution created by you, the developer, you cannot offer that same software to another client.

Health Records Act (2001)

The Victorian Health Records Act (2001) is structured around Health Privacy Principles that are concerned with personal health information by any organisation in the public or private sector in Victoria. The act is designed to "protect privacy and promote patient autonomy, whilst also ensuring safe and effective service delivery, and the continued improvement of health services."
https://www2.health.vic.gov.au/about/legislation/health-records-act

The HPPs will apply to the collection, use and handling of identifying personal information that is defined as "health information" under the Act including:

- information or opinion about the physical or mental health, or disability, of an individual,
- an individual's expressed preferences about the future provision of health, disability or aged care services to him or her,
- the nature of health, disability or aged care services that have been, or are to be, provided to an individual,
- information originally collected in the course of providing a health, disability or aged care service to an individual,
- personal information collected in connection with the donation of human tissue,

- genetic information that is or could be predictive of the health of an individual or their descendants.

Any organisation that handles this kind of identifying health information is subject to the HPPs. The Act applies regardless of the size of the business or organisation. There is no "small business" exemption. Organisations that are subject to the Act, when they handle health information, include:

- Victorian Government organisations and public bodies
- universities, schools and other educational or child based organisation.
- researchers
- blood and tissue banks
- public and private sector employers (eg. in relation to their employees' personnel records)
- counsellors
- insurers and superannuation organisations
- gymnasiums
- any other organisation that holds health information or health reports concerning its clients or customers.

No organisation may use or disclose health information about an individual for a purpose other than the primary purpose for which the information was collected, unless the individual provides consent with a couple of exceptions:

- if the use or disclosure is necessary for research, or the compilation or analysis of statistics is in the public interest.
- If the use or disclosure is necessary for training of health staff to improve health services.

Ethical Issues

The development and use of software solutions have many ethical repercussions. Developing a software solution:

- replaces manual or unskilled labour and can cause stress for people forced to up-skill,
- stores large amounts of sensitive data that requires complex and expensive protection,
- requires ongoing updates of both software and hardware,
- may not provide equal access for all,
- creates a reliance on the technology,
- may provide unhealthy working environments.

Social Impact

Often software solutions are designed to improve the efficiency and effectiveness of an organisation. If not already automated, a new software product can replace a manual system stored and maintained on paper. Replacing a system that has been in use for some time, can cause stress among employees. It is an ethical obligation of the management to up-skill staff to use the new digital system. As technology rapidly evolves and is used in the workplace, people's skills become increasingly obsolete. The continued expectation for staff to learn new skills can be confronting and uncomfortable. If management is unable to up-skill a staff member to use the new system, that person may need to be demoted, or could even lose their job.

Sensitive Data

Before digital systems, databases were primarily paper-based and could be stored behind locked doors. In these cases only authorised persons could access the data. Now with increased access due to digital networks, access to sensitive data is an expensive and complex process to restrict. Management is not only legally but ethically obliged to ensure that the data will be used only for the purposes for which it was collected. To ensure authorised staff use the data appropriately, the onus is on the management to prepare and share a "Code of Conduct" with all staff. The code of conduct must outline how access to the data is limited, and the consequences to unauthorised use of data. An authorised person can

take advantage of access to sensitive data for personal financial gain, to ruin a reputation, or simply for vandalism. Providing a hierarchy of access authorisation reduces the risk of inappropriate use of sensitive data.
It is an ethical obligation of the organisation to protect sensitive data from unauthorised access regardless of the costs involved. Commercial enterprises need to make decisions about the scope of a new software solution in managing data. With limited understanding of the potential harm that unauthorised access to sensitive data can do, management may try to cut corners in an effort to save money, leaving sensitive data at risk.

Reliance on Digital Systems

When an organisation implements a digital system to replace a manual or paper system, all operations will rely on that system being up and running, which might include online access. If the organisation is providing a greatly needed service, and the software system that has been put in place is not functioning or is offline, there is an ethical obligation to be able to provide data if requested. Software developers should be mindful that full reliance on their software may impact the organisation in a range of situations.

Interoperability with other existing systems —When developing a new system for an organisation, it is appropriate to be mindful of the existing systems in place. The client may not fully understand how a new system could cause issues with existing systems if they are unable to share files or work together in an effective and efficient way. There are many cases where organisations have purchased and implemented new digital systems that overlap with existing systems but are unable to operate together. Interoperability is the ability for systems to share data and be able to process that data. The software developer is in a position to foresee issues in the ongoing operation of the organisation and to suggest possible features in the proposed system to work seamlessly with the existing systems. If the existing systems are overlooked, the final software solution may not be more efficient in the long run.

Future proofing — Development of hardware and other system technologies continues to change quickly. Operating systems are constantly being updated and revised. It is a developer's ethical obligation to be up to date in the ongoing technology trends to ensure anything they build will still work with future versions for Network Operating Systems, computer Operating Systems and for all internet and world wide web protocols and standards. Developing a system that will soon be obsolete is unethical. Similarly, it is important that new digital systems are backwards compatible with older systems. This provides access to the software for those who are unable to update equipment or operating systems.

Updating technology and e-waste—With so many changes and advances in technology there is pressure for organisations and individuals to purchase new technology. The mobile phone industry creates new models regularly which ultimately end up in rubbish dumps. E-waste is electronic waste material, which is very difficult to recycle due to the blended materials of plastic and metal. They are too hard to extract from one another. Future proofed software that can work on any machine may be an ethical approach to ensure new hardware devices are required less frequently.

Accessibility

When systems are designed for a client's customers, it is important to ensure general accessibility is integrated into the final design. Firstly, how the day-to-day users will access the new system, needs to be considered. If a mobile solution is required because the user is moving around a warehouse or collecting data in the outdoors, the best device must be identified in terms of ergonomics and inclusive features. Ergonomics is the study of health in the work place. It is a commonly raised ethical issue because end users are often expected to sit and stare at screens for long periods of time. This practice can impact on the health and well-being of that person.

When making decisions about device features, it is possible to consider speakers and microphones to assist users who could benefit from accessing data as sound. Devices that allow for a range of contrast and zoom features on the screen can assist users in certain circumstances. Allowing for speech recognition, text-to-speech can provide accessibility to those with vision or other impairments.

Ownership

Copyright is covered under legislation and legal obligations, however, it is important that the software developer be aware of their ethical obligations beyond the legislation. Developers may be tempted to use or 'recycle' code that was developed for another client. This is not appropriate as both clients have paid for a bespoke solution. Another temptation is to use open source software code. Any shareware or freeware that is open source should not be used for a commercial product. When developers share their code online in platforms such as GitHub, it is shared under the terms and conditions of an open source licence. These licences are restricted to non-commercial redistribution or modification. It is not ethical to take open source software and charge clients for software development you have not created. Developers may also be tempted to decompile or reverse engineer other commercial software. Decompilation is the translation of executable code into a higher computing language that can be edited by a programmer. This breaches the copyright of the owner of that software because they are using another person's solution.

Evaluating Software Development Security Strategies

To evaluate the security of a software application, a list of criteria is required to identify all the risks that the software may encounter. A lack of application security is a common weakness and can make your applications prime targets for cyber-criminals to exploit vulnerabilities and steal intellectual property. It's difficult for developers to identify these potential threats while meeting their milestones to build their solution and deliver on time. Developers are not going to be able to fix everything, but some issues can be prioritised. Some issues are critical, which are issues that can be immediately exploited and cause harm to the organisation. Some issues are non-critical, but when combined with poor practices, can produce vulnerabilities. Investigating all vulnerabilities is a mammoth task so prioritisation is required, based on the risk to sensitive information, how long it will take to do, the cost, and other constraints.

There are common security practices that the software can be measured against. The organisation should have the following to identify what cannot be tolerated:

- Security Policy
- Risk Management protocols
- Best Practice education for users
- Data encryption for sensitive data
- Access Control checks.

Vulnerability Assessments

To test any potential risks the software may have, a specialised security evaluator can conduct an assessment to test for vulnerabilities. This is called a Risk and Vulnerability Assessment (RVA). In the build of a large software solution for a major organisation, a dedicated Product Security Incident Response Team (PSIRT) would be employed to investigate and report any security weaknesses. This team works within the scope of the organisation's security and risk management policies to disclose any risks that need to be fixed.

The PSIRT will run an RVA using the Common Vulnerability Scoring System (CVSS), an open framework for communicating the characteristics and severity of software vulnerabilities. The CVSS is a scoring system that can test a software product thoroughly and it includes criteria that investigates

the kill chain in cyber-defense. The Kill Chain is a series of seven stages that an attacker may employ to exploit a vulnerability on a network or software system:

1. Reconnaissance — The attacker researches information about the target.
2. Weaponisation — The attacker creates malware to send to target.
3. Delivery — The attacker uses a network, port scan, email or other method to send the malware.
4. Exploitation — The malware is executed.
5. Installation — The malware and back-doors are installed.
6. Command and Control — The attacker uses remote control over the malware.
7. Action — The attacker executes information theft, attacks on the software or by implementing additional attacks.

For each stage of the Kill Chain, a range of methods are used to evaluate the risk by scoring the vulnerability using the CVSS. Here are a few of the many scoring features:

- web content can be maliciously altered
- system files can be stolen
- user interaction can download or receive malicious content
- document parsing vulnerability
- levels of users - low-privileged user is able to exploit a virtual machine that enables an attacker to read and/or delete files
- security boundary between microprocessor privilege levels
- SQL injection vulnerability in a web application
- a simple Portable Document Format (PDF) document that allows an attacker to compromise other files
- attacker able to exhaust a shared system resource.

Once the RVA is complete, a score identifies the vulnerabilities in detail. These vulnerabilities are disclosed and reported. The reports outline how these weaknesses can be fixed to comply with the security policies of the organisation.

Penetration Testing

Penetration Testing (Pen Testing), often called 'White Hat Hacking" is the attempt to exploit the vulnerabilities in a system by using the criminal attackers' techniques. Although certainly an effective method of discovering weaknesses in the software, it cannot be relied upon entirely to ensure your software is secure. Pen Testing is used as a starting point in looking for vulnerabilities. It is the full RVA that will identify the full scope of weaknesses in a system. Both methods can be used together, sometimes referred to as vulnerability assessment/penetration testing, or VAPT.

Impact of Cyber Crime

There have been over 100 major data breaches as a result of cyber attacks in Australia in 2019 alone. Here are a few examples that occurred in the year this book was written.

Electronic Health Records

The Age newspaper reported in 2019 that medical records of a Melbourne hospital's cardiology unit "Melbourne Heart Group" were hacked. It appeared that malware from Russia or North Korea was responsible for restricting access to the patient files. The malware was a form of ransomware which encrypted the files until a ransom in cryptocurrency was paid. After the hospital paid the ransom, the decryption that was provided did not successfully restore all the files. Health records are often targeted due to the importance and the sensitivity of the data. A hospital cannot afford to lose access to their records due to the life-threatening consequences that may occur. When a human life is relying on treatment there's no time to wait for terabytes of patient data to be restored over days and weeks. Medical organisations are always willing to do what is required to restore data as soon as possible.

In recent years the Australian Federal Government implemented a new program to store all Australian health records in one place called "My Health Records". The program was initiated to consolidate all medical information about a patient in one location. The aim is to allow all medical practitioners access to all the key information about a patient, especially if they are seeing that person for the first time. It can be especially helpful in emergency situations where medical staff need to access information about the patient if they are not conscious. Concerns have been raised by privacy advocates. Many are concerned that these records can be stored long after the patient's permission for the provided data has been removed. Also there are many health situations that a patient may wish to keep private, especially with mental health or other culturally sensitive conditions that relate to sexuality. If government data is easily hacked and distributed this can have grave concerns for the privacy of all Australians.

Education Records

The Northern Territory Department of Education was targeted by the media in 2019 for failure to pass a security audit on the storage of student data. One of the key issues highlighted by the auditor's report was that there were an excessive number of administration accounts, and there were a number of terminated staff members who could have easily retained access to the system. The lack of password updates also caused concern. In the case of terminated staff members having continued access to sensitive data, this can allow disgruntled ex-employees to tamper with the system is some way. Unauthorised access to the personal information about children's enrolment, attendance rates, health and behaviour can potentially be dangerous for the children involved.

Commonly the school portal software is the main target for school hackers. These are cloud based web applications that allow teachers to manage teaching resources for each class, their student attendances and student assessment. Some commonly used products are: Simon, SchoolBox, Edumate, Compass and Synergetic. Due to the web based nature of the interface for these portals, SQLi is a leading threat. Frequently, students attempt to change results or may wish to access private data for nefarious reasons.

Employment Records

Every organisation must have a record of each employee so they can manage their salary and entitlements. This means they also hold bank account details. In 2017 a major data breach of 50 000 Australian workers' data was leaked. The information included names, passwords, ID data, phone numbers, as well as credit card numbers and corporate information including salaries and expenses. Employees of the Department of Finance, the Australian Electoral Commission and National Disability Insurance Agency as well as private sector workers were affected. It was never reported how this data was leaked, however, it seriously undermined public confidence in government data security. Once a full record has been leaked and distributed to the dark web, anyone can access that data and commit identity fraud.

Financial Records

The Australian National University was also hacked in 2019. A staff member unknowingly handed over login credentials and downloaded malware through an email in a hacking technique known as "spear phishing". The targeted staff member did not click on anything, but merely saw a preview of the attached file which was enough for the hackers to collect what they needed to get into the system. The hackers were after financial and employee records. The affected human resource records included payslips, bank account details, tax file and passport numbers, emergency contacts, and some academic records. Clearly they were targeting full personal data for the purposes of identity theft.

A new platform shared by Australian banks to process web bank account access was hacked in 2019. The platform is called New Payments Platform (NPP) and used PayID indexes for records in the system. Tens of thousands of banking customers were affected when the PayID details including account holder's name and bank account numbers were accessed. The banks were required to quickly

announce the data breach to all customers and to the media and to make restitution to all stakeholders. The hackers approached customers posing as bank staff to encourage them to provide access information to their bank accounts. The personal details of the customers including mobile numbers, email address, customer name, BSB and account numbers were disclosed on the dark web where they could be bought to be used for identity and financial fraud.

Cybersecurity Risk Management Strategies

Risk management is an institutionalised approach to:

- identifying risks,
- reducing the impact of those risks,
- reducing the probability or likelihood of risks,
- ongoing monitoring of all potential risks.

In figure 4.2 below you can see an example of each of these stages to the approach.

Fig 4.2

Risk Managment Strategies

Risks	Reduce Impact	Reduce Probability	Ongoing Monitoring
Spear-phishing of staff	Train staff not to open emails from unrecognised sources	Firewall and Email filters restrict incoming messages of a certain type.	All firewalls, Email server operating system and malware are up-dated

Effective risk management relies on a solid organisational culture of reporting vulnerabilities and an ongoing education of all stakeholders in minimising risk. Cybersecurity risk management is a combination of strategies, technologies and user education to protect data from cybersecurity attacks. Cybersecurity risk management takes the idea of real world risk management and applies it to the cyber world. It involves identifying your risks and vulnerabilities and applying administrative actions and comprehensive solutions to make sure your organisation is adequately protected.
Like all complex processes, there are many cyber security risk management models. We are going to look at Cybersecurity Capability Maturity Model (C2M2) and the National Institute for Standards and Technology (NIST) Framework Model.

Cybersecurity Capability Maturity Model (C2M2)

The model looks at 10 domains of cybersecurity in this evaluation phase.

1. Risk management: Identify, analyse, and mitigate cybersecurity risks.

2. Asset, change, and configuration management: Manage the organisation's hardware and software.

3. Identity and access management: Create and manage access profiles for entities that may be granted logical or physical access to the software.

4. Threat and vulnerability management: Create plans, procedures, and technologies to detect, identify, analyse, manage, and respond to cybersecurity threats and vulnerabilities.

5. Situational awareness: Establish activities and technologies to ensure the administrator of the system has access to the status of the system and security of the data.

6. Information sharing and communications: Establish and maintain relationships with all entities to collect and provide cybersecurity information, including information about threats and vulnerabilities, to reduce risks and to increase operational resilience.

7. Event and incident response, continuity of operations: Create plans, procedures, and technologies to detect, analyse, and respond to cybersecurity events and to sustain operations throughout a cybersecurity event.

8. Supply chain and external dependencies management: Ensure controls are used to manage the cybersecurity risks associated with services and assets that are dependent on external entities.

9. Workforce management: Create a culture of cybersecurity and to ensure the ongoing suitability and competence of personnel.

10. Cybersecurity program management: Establish and maintain an enterprise cybersecurity program that provides governance, strategic planning, and sponsorship for the organisation's cybersecurity activities.

C2M2 in Action

Jisha Dhaliwal is a chemist and has started a new small business providing specialised compounded products to treat a variety of cat and dog ailments. She has set up a website that will collect the details of the clients who order the products as well as processing their payments. Below is an example of strategies at each of the ten domains of the C2M2 model that Jisha needs to include in her ongoing risk management of the system her business will rely on.

Cybersecurity Capability Maturity Model (C2M2)	
Domains	**Strategies**
Risk Management	All potential risks are investigated on the website. This can include mitigating SQLi and other methods of accessing the data stored on her server and the web server.
Asset Change & Configuration	After investigating all hardware and software being used to manage the website, some updates to the equipment, firewall and anti-malware may be in order.
Identity and access management	Jisha has a few people working for her and she needs to identify what they can access on the system. As the business owner, she would have the administrative access to the system, while other staff who do not need to access sensitive data can be provided with limited access.
Threat and vulnerability management	Jisha needs to establish processes that will identify threats, such as the use of intrusion detection software. She needs to develop guidelines for managing any harmful event or threat to the data or system.
Situational awareness	At any time, Jisha or her network administrator should be aware of any threats to the system. A honeypot intrusion detection can alert the administrator of the website to identify a threat to the financial records.
Information sharing and communications	Jisha's system will be interacting with banking institutions to process payments. There will be requirements for data transfer to ensure it is safe. Encryption methods may need to be implemented.
Event and incident response, continuity of operations	The administrator of the website will be required to have a clear plan of action in the event of a threat to the security of the system. A detailed list of key responses needs to be developed.

Supply chain and external dependencies management:	A website is stored and managed on the web server provided by an ISP. The private server holding the client data is in Jisha's office. Data will move from the web server to the private server, which then can be accessed by the devices in the office. Transaction data is sent to the external financial organisation for processing. A clear outline of how data is moved from one location to another needs to be reported. In the case of viral malware, data may move and spread the virus through the servers in the order the data is processed.
Workforce management	The people who work for Jisha need to be educated in cybersecurity practices to ensure that human error is not the source of a threat.
Cybersecurity program management	Ongoing cybersecurity management of the whole system must be maintained by all staff. The program must be an active and updated plan of action to keep Jisha's system threat free.

National Institute for Standards and Technology (NIST) Framework Model

The NIST Framework Model is a model based on five core functions to achieve cybersecurity outcomes for a software system. The table below shows the different categories of each function. These functions do not have to be followed in series they can be implemented concurrently and continuously to maintain the security of the system. The functions are:

- Identify,
- Protect,
- Detect,
- Respond,
- Recover.

Key strategies are highlighted in the table below that Jisha might implement at each functional stage.

Functions	Strategies
Identify	Jisha creates an inventory of physical devices, software platforms and applications. Data flow is mapped throughout the website and all communications throughout the organisation are identified. All the external entity systems are identified and categorised. The cybersecurity roles and responsibilities of all Jisha's staff are identified. The context of Jisha's business is identified along with the data supply chain. Jisha's business goals and objectives are identified. A security policy is developed. All legal requirements are understood and efforts made to comply. All vulnerabilities and threats are identified on the website and in the office. All vulnerabilities and threats are identified in communication. Potential risk is identified and the impacts to Jisha's business, staff and clients are identified. Risk Management processes are defined.
Protect	Jisha's staff all have appropriate authorisation levels to access the system. The office equipment and mobile devices that access the storage of sensitive data are protected by physical boundaries. The network of Jisha's business is protected with appropriate technology. All staff are trained and expectations of their behaviour is communicated Stored data is protected by firewalls, encryption, locked doors and anti-malware. Data in transit is protected through an encryption process. A system development life cycle is developed and implemented to run continuously to ensure data is fully protected at all times. Back-up processes for all data are developed and put in place. Policies for managing, archiving and destroying data are implemented. Protection processes are constantly improved. Response and recovery plans are developed and tested. Audit logs are determined and implemented with an ongoing policy for regular review.

Detect	Detecting systems are implemented to detect attack events and other threats. The impact of threats are determined and incident alerts are established within the system. A monitoring system is implemented to detect unauthorised access or malicious code events within the network The physical environment of the network is monitored for threats. External systems are monitored to detect threats. All personnel access is monitored for threatening activity. Vulnerability scans are implemented. All of Jisha's staff are made clear of their roles and responsibilities for the detection of threats, to ensure they are accountable. All staff are expected to report all threats as they are detected. Detection processes are communicated to staff and continuously improved.
Respond	The business needs a response plan that can be easily tested and executed by all of Jisha's staff. Threat events are reported and the information shared with all stakeholders. All detection system notifications are investigated by the system administrator and the impact of the threat is made clear. Incidents are contained and forensics are performed to find the source of the threat. New vulnerabilities are identified and mitigated. These will inform new response plans.
Recover	A recovery plan is designed to be executed after a threat event. Recovery plans implement the new protection strategies and are updated. Communication with all stakeholders needs to take place and some public relations are required to re-establish Jisha's business reputation. All recovery activities are communicated to stakeholders to ensure normal business processes can proceed.

In Chapter 6 the structure for a Cybersecurity report is outlined. Using the content explored in this chapter you can analyse a case study to report on:

- the goals and objectives of the client,
- the current security practices in the case study,
- vulnerabilities in the system,
- possible threats to the data,
- consequences if vulnerabilities are exploited,
- legal obligations,
- and a risk management strategy.

Unit 4 Outcome 2
Development & Evaluation
Review Questions

1. What is a security appliance?

2. Describe three key software security controls.

3. What is the difference between symmetrical and asymmetrical encryption?

4. Describe the role of Version Control Systems in software development.

5. Why are updates so important?

6. What is the purpose of an audit in relation to software development?

7. How does buffer overflow create vulnerabilities?

8. Which malware is designed to create a "back-door" to an operating system?

9. How is it possible for a MitM attack to occur without the victim's knowledge?

10. Name four types of Social Engineering attack methods.

11. Describe how Cross Site Scripting is used to exploit vulnerable websites.

12. Why is SQLi the most common form of attack used on web sites?

13. Identify the distinctions between 'correctness', 'authenticity' and 'accuracy' in terms of data integrity.

14. Identify two businesses that use a software system. Write a goal and an objective for the system for both of these businesses.

15. Create a mnemonic for the 13 Australian Privacy Principles.

16. Why is the Copyright Act important to the software developer?

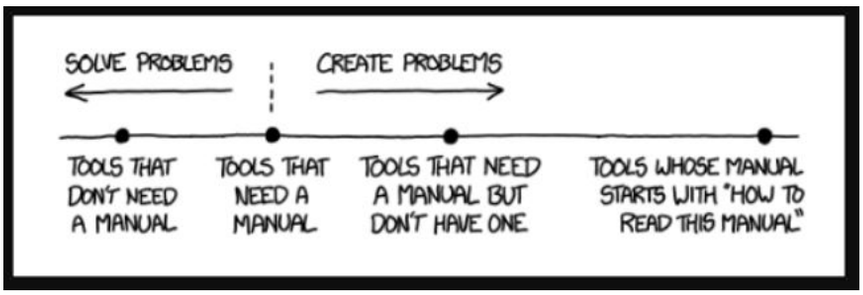
17. What is interoperability and why is it an ethical consideration?

18. What is the Kill Chain and why is it important to understand?

19. Why is Pen Testing a limited approach to investigating system vulnerabilities?

20. Investigate how many major cybersecurity breaches have occurred in Australia since the printing of this text book (November 2019). Discuss in class.

21. Choose the C2M2 or the NIST Framework Model to design a risk assessment of your school portal system.



https://xkcd.com/1343/

Online Support

Cybersecurity is not just a new buzz word, but a very real and growing need for all organisations. There is plenty of online assistance for organisations to plan their digital security.

- **esecurityplanet.com/**
- **insights.sei.cmu.edu/insider-threat/**
- **qao.qld.gov.au/report/managing-cyber-security-risks**
- **business.com/articles/cybersecurity-risk-assessment/**

https://xkcd.com/2223/

Assessment Resources

Visual Basic

PHP

Programming Requirements U3O1 & U4O1 Criteria 5 & 6

For assessment purposes, students must be familiar with all of the listed programming requirements; however, not all requirements must be addressed in each task. Teachers would be expected to select the appropriate requirements based on the key skills outlined in the study design.

In the development of the modules and solutions, the chosen programming language should provide students with the ability to carry out the development stage of the problem-solving methodology within three conceptual layers: interface, logic and data source.

Interface

Programming requirements for interface layer:

- develop a graphical user interface (GUI), for use in digital systems such as laptops, gaming consoles, mobile phones, tablets, robots.

Note that databases are not to be used in the interface layer.

Logic

Programming requirements for logic layer:

- construct and use data structures
- design and apply data validation techniques
- use program control structures: selection, iteration and sequencing
- use modularisation and code optimisation

Data Source

Programming requirements for data source layer:

- design, construct and use external storage and access technologies
- retrieve data from external sources.

It should be noted that while modules and solutions can be created in one language, other languages may be used to embellish its features

Programming in Visual Basic

Visual Studio is a Rapid Application Development Environment. It is very easy to create an interface with drag-and-drop objects and the coding is very straight forward. Many schools rely on Visual Basic for VCE Software Development because there are a lot of resources online available for support and students are able to produce a complex solution quickly, given you only have a few weeks to develop your SAT.
When Visual Studio launches it expects students to login with a Microsoft account. This account is free and the development environment is free to use once they have a Microsoft login.

This chapter does not contain all the tools and techniques you may need for your project but it does provide a starting point if you are unfamiliar with programming. You can find plenty of online Visual Basic resources online at: **www.vicfarrell.com.au**

STARTING UP

Open Visual Studio and select "Create a new project".

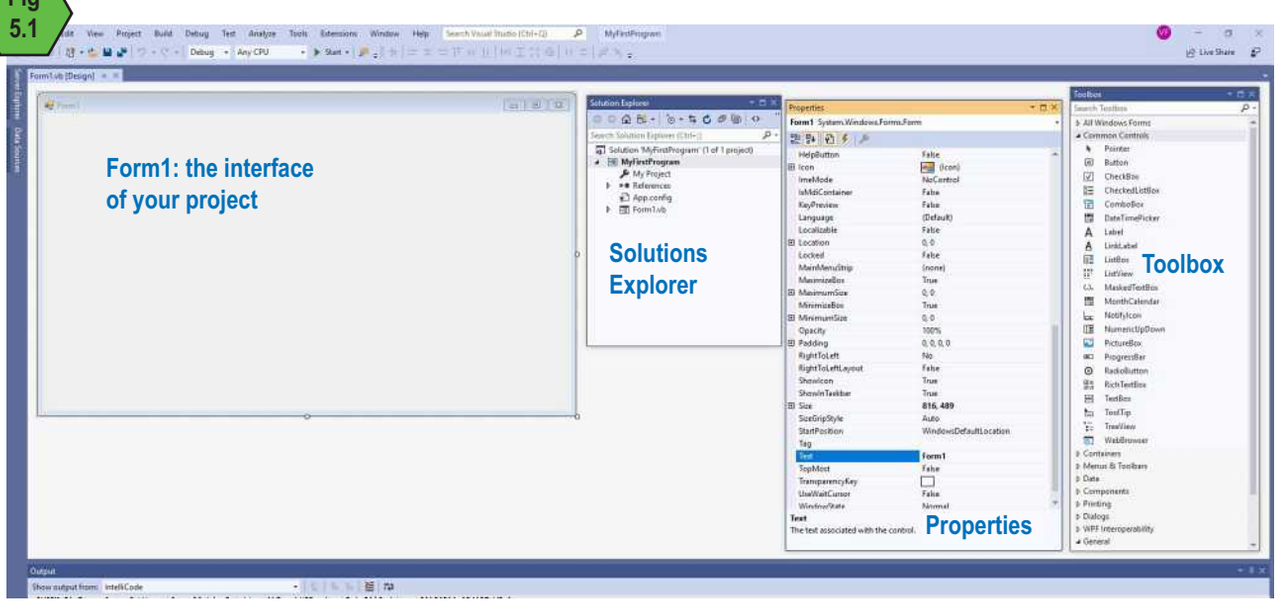
At this point you need to identify the language, the platform and project type. These can be found at the top of the screen. If you have a PC lab at your school here is the best option:

Language = Visual Studio
Platform = Windows
Project Type = Desktop

Once you select Windows Forms App (.NET Framework) select "Next". It is important that you name your project at this point and identify where you will be storing the project. Identify the location. At no time try to rename your project or move parts of the project around. Everything must remain contained in your folder. Then select "Create".

The Form is the interface where we place objects from the Toolbox window. We edit the objects with the properties in the Properties window. The Solution Explorer allows us to see where we are within the project. You can see in figure 5.1 below that Form1 is where we build our interface, the Solution Explorer window displays all the components, the Properties window shows the properties of the Form that has been selected and finally the Toolbox which allows for more objects to be placed on the form.

Fig 5.1



Activity One: Creating a basic application

We are going to make a simple calculator that reads in two values and adds them up. The first thing we do is create the interface, see figure 5.2.

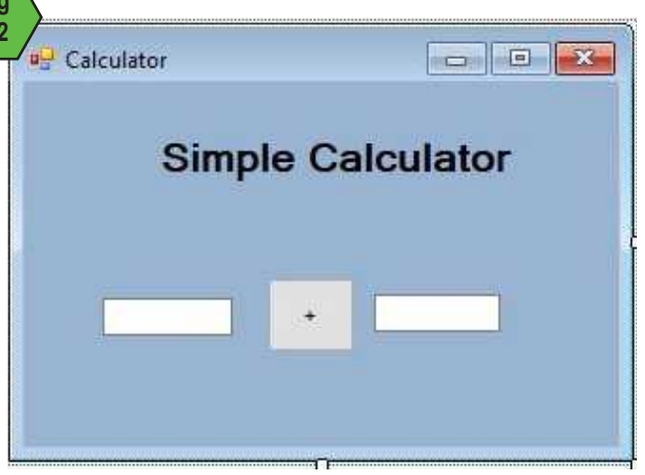
Interface Instructions

1. Select the Form and access the Properties Window.
Here you can change the background colour and the Text that shows in the top of the window. You can also change the name of the object. I changed the Text to "Calculator" and the background colour to a blue-grey.

2. Search the Toolbox for Textbox to add it to the Calculate Form.
Once the textbox is selected you go into the Properties box and change the Name to "txtNumber1".
You can go ahead and make a second textbox and call it "txtNumber2".

3. Back to the Toolbox, and add a Button to the form.
In the properties box while the button is selected on the form, rename the button "btnAdding". Change the Text property to a "+" sign.

Fig 5.2



4. Double click on the button and you can create a subroutine for the button.
Type in the code within the sub statements

```
Public Class Form1

  Private Sub BtnAdding_Click(sender As Object, e As EventArgs) Handles btnAdding.Click
    Dim intNumber1 As Integer
    Dim intNumber2 As Integer
    Dim intAnswer As Integer

    intNumber1 = txtNumber1.Text
    intNumber2 = txtNumber2.Text

    intAnswer = intNumber1 + intNumber2

    MsgBox("Your answer is " & intAnswer)
  End Sub

End Class
```

The Code

The first line indicates the class, which controls all the methods of the objects in the program.

```
Public Class Form1
```

When the button is clicked by the user, the subroutine executes. The 'Sub' is placed between the Class statements. The programming needs to be between the Private Sub and the End Sub statements.

```
Private Sub BtnAdding_Click(sender As Object, e As EventArgs) Handles btnAdding.Click


End Sub
```

Visual Basic requires that all variables are declared before they can be used, so we declare three variables, the two input values and the answer. VB uses "Dim" which is short for dimension. When we declare the variable we identify the data type it handles.

```
Dim intNumber1 As Integer
Dim intNumber2 As Integer
Dim intAnswer As Integer
```

To assign a value to the variable, we first must identify the variable that is empty and then the object and it's property that holds the data. Below intNumber1 will be assigned the value that is in the text property of the object called txtNumber 1.

```
intNumber1 = txtNumber1.Text
intNumber2 = txtNumber2.Text
```

Now, we have the two variables with data we can calculate intAnswer.

```
intAnswer = intNumber1 + intNumber2
```

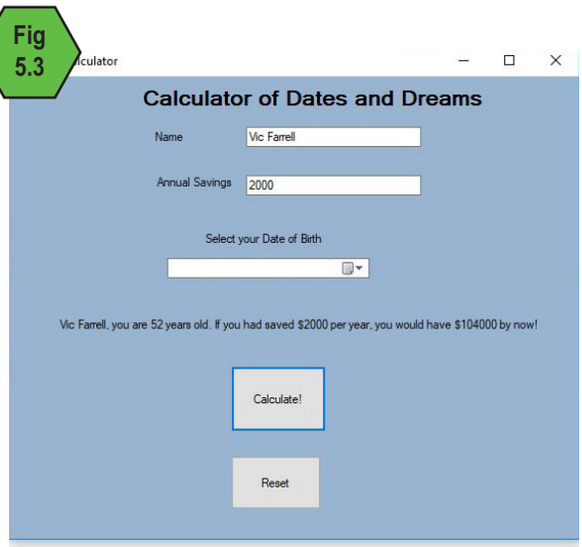
There are a number of ways we can display the answer to the user. Here we have used a Message Box which pops up as a window on the screen. You can see the text "Your answer is " must be placed within the brackets and the inverted commas. The data in variable intAnswer can then be displayed after the use of "&" inside the brackets.

```
MsgBox("Your answer is " & intAnswer)
```

Activity Two: Creating a basic application using date/time

The next activity demonstrates how you can use calendar objects and execute operations on date and time variables. See figure 5.3.

Fig 5.3



The interface comprises of the following objects:

txtName — reads in the name of the user.
txtSavings — reads in the annual savings of the user.
dtpDOB — allows the user to select their date of birth.
lblAge — displays the output.
BtnCalc — executes the calculation.
BtnReset — resets all input objects to clear.

The code to make this application run requires the following variables:
strName - (Data Type - string), entered by user.
dblSavings - (Data Type - double), a floating point number with limited decimal places entered by user.
dblWealth - (Data Type - double), the calculated savings total.
dateDOB - (Data Type - date), entered by user.
dateToday - (Data Type - date), from the computer clock.
intAge - (Data Type - integer), calculated output.

```
Public Class AgeCalculator

  Private Sub BtnCalc_Click(sender As Object, e As EventArgs) Handles btnCalc.Click
    Dim strName As String = txtName.Text
    Dim dblSavings As Double = txtSavings.Text
    Dim dblWealth As Double
    Dim dateDOB As Date = dtpDOB.Value
    Dim dateToday As Date = DateTime.Now
    Dim intAge As Integer = DateDiff(DateInterval.Year, dateDOB, dateToday)

    dblWealth = dblSavings * intAge
    lblAge.Text = (strName & ", you are " & intAge & " years old. If you had saved $" & dblSavings & " per year, you would have $" &
    dblWealth & " by now!")
  End Sub

  Private Sub BtnReset_Click(sender As Object, e As EventArgs) Handles btnReset.Click
    txtName.Text = ("")
    dtpDOB.Value = Today()
    lblAge.Text = ("...............")

  End Sub

End Class
```

Some new functions and procedures to note in this program relate to how dates are handled. Both strName and dblSavings are read in from the objects in the declaration statement. You can see that the Date of Birth is read in from the dtpDOB object via the "Value" property. The current date is retrieved by the simple function: "DateTime.Now".

The intAge variable is calculated at the declaration statement by using a function called DateDiff(). Let's look at it in detail:

```
intAge = DateDiff(DateInterval.Year, dateDOB, dateToday)
```

The DateDiff function looks at the DateInterval in Years only (DateInterval.Years). You can also choose days or months if you wanted. Then it lists the first date, then today, and it will calculate the number of years between those two dates.

dblWealth is calculated by multiplying dblSavings and intAge. This must happen after intAge has been calculated. Finally the label lblAge displays the name of the user and their age, their annual savings they entered and the calculated total accumulated wealth.

To make a user-friendly interface, a clear button must be included. BtnReset takes out all entered text from both textboxes and sets the DateTimePicker back to today's date.

Hotel Programming Task
Stage 1

Complete this programming Task to use what you learned from the first two activities:

Create a hotel booking system that reads in how many days the customer wants to stay and outputs the Check-In and Check-Out dates. Based on $89 per night, calculate the total price of a room booking.

Decision Control Structures

In VB we use IF and Select Case.

IF (Condition) Then
    Action1 (will happen if condition is True)
ELSE
    Action2 (will happen if condition is False)
ENDIF

Activity Three: Tax Calculator

This program will read in a value that is the Gross Income. Net Income is calculated by subtracting the calculated tax. The tax is calculated as a percent of the Gross but it depends on how much you earn. Australian tax brackets are:

- Less 18,200 you pay no tax at all.
- Between 18,200 and 37,000 you pay 19%.
- Between 37,001 and 90,000 you pay 32%.
- Between 90,000 and 180,000 you pay 37%.
- Over 180,000 you pay 45%.

We are going to build a very simple program that reads in the Gross Income, tests it for the tax bracket and apply that tax rate to the whole income to create a Net Income (after tax).

```
Public Class TaxCalculator
  Private Sub BtnCalc_Click(sender As Object, e As EventArgs) Handles btnCalc.Click
    Dim intGross As Integer
    Dim dblNet As Double
    Dim dblTax As Double
    intGross = txtGross.Text

        If (intGross < 18200) Then
                dblTax = 0
        ElseIf (intGross > 18200) And (intGross < 37000) Then
                 dblTax = 0.19
        ElseIf (intGross > 37001) And (intGross < 90000) Then
                dblTax = 0.32
        ElseIf (intGross > 90001) And (intGross < 180000) Then
                dblTax = 0.37
        Else
                dblTax = 0.45
        End If
    dblNet = intGross - (intGross * dblTax)
    lblNet.Text = ("Your Net Income is : $" & dblNet & ". Your tax rate is " & dblTax * 100 & "%.")
  End Sub
End Class
```

You can see that it is not enough to state only one condition for each tax bracket. The lowest tax bracket can use (intGross< 18200) but all the other brackets have two conditions. The "Else" provides the bracket for the highest rate.

Validation

Let's add validation to our Tax Calculator.

Existence Checking

If no data was entered, then we need to provide the user with a prompt to enter the missing data.
VB Example:

```
If (txtAmount.Text) <> "" then
        intAmount = txtAmount.Text
Else
        MsgBox("Please enter an Amount")
End IF
```

Data Type Checking

Sometimes a mistake can be made or data mistyped. Checking the data type is crucial.
VB Example:

```
Dim NumericCheck as Boolean
Dim intAmount as Integer
NumericCheck = IsNumeric(intAmount)
If (NumericCheck = True) then
        intAmount = txtAmount.Text
Else
        MsgBox("Please enter a value for Amount")
End IF
```

Range Checking

Sometimes the data has been added and the data type is correct but it is outside of the range. If you were to enter a very large number your program would crash and report an 'overflow'. An overflow occurs when the variable has not been defined to hold a number that large.

```
If (txtGross.Text <> 0) And (txtGross.Text < 2000000) Then
          intGross = txtGross.Text
ELSE
        MsgBox("you have entered a value too large or a zero, please enter a valid value")*
END IF
```

To incorporate all three validation methods, nested IF statements are required. The algorithm on the right shows how the existence check is the first priority, then the data type check. Finally the range check tests the data before the data can be used.

```
Algorithm

 If (data exists) Then
        If (data type is correct type) Then
                If (data is in range) Then
                        Read in the Data
                        Use the Data
                        Output the Processed Data
                Else
                        Message: "Your data is out of range"
                End IF
        Else
                Message: "Your Data Type is not Correct"
        End If
Else
Message: "Please enter some data"
End if
```

* Everyone knows people who make $2000000 don't pay tax.

Activity Four: Add Validation to Tax Calculator

The code below uses the Australian tax brackets to calculate the tax required to be paid for any given income. It also includes all three validation types.

```
Public Class Form1
    Private Sub BtnCalc_Click(sender As Object, e As EventArgs) Handles btnCalc.Click
        Dim intGross As Integer
        Dim dblNet As Double
        Dim dblTax As Double
        Dim NumericCheck As Boolean

        If txtGross.Text <> "" Then
            NumericCheck = IsNumeric(txtGross.Text)

            If NumericCheck = True Then

                If (txtGross.Text <> 0) And (txtGross.Text < 2000000) Then

                    intGross = txtGross.Text

                    If (intGross < 18200) Then
                        dblTax = 0
                    ElseIf (intGross > 18200) And (intGross < 37000) Then
                        dblTax = 0.19
                    ElseIf (intGross > 37001) And (intGross < 90000) Then
                        dblTax = 0.32
                    ElseIf (intGross > 90001) And (intGross < 180000) Then
                        dblTax = 0.37
                    Else
                        dblTax = 0.45
                    End If

                    dblNet = intGross - (intGross * dblTax)

                    lblNet.Text = ("Your Net Income is : $" & dblNet & ". Your tax rate is " & dblTax * 100 & "%.")

                Else
                    MsgBox("You have entered a Gross Earning outside of the required range")
                End If

            Else
                MsgBox("Please enter a value for your Gross Income")

            End If

        Else
            MsgBox("Please enter your Gross Income")

        End If

    End Sub

    Private Sub BtnReset_Click(sender As Object, e As EventArgs) Handles btnReset.Click
        txtGross.Text = ""
        lblNet.Text = "Your Net Income is :"
    End Sub
End Class
```

Extension Task

Adapt your program to calculate the tax rates for each bracket. Example: Gross = $85,000 - The first $18,200 will not attract tax. Between $18,201 and $37,000 ($14,500) will attract the 19% tax rate. Finally $85,000 - 37000 ($48,000) will attract the 32% tax rate. (How tax is actually calculated.)

Activity Five: Select Case

A ComboBox allows the user to select from a data list in a pull-down menu. It reduces the need for validation on typed-in data. It is controlled using a Select Case Statement in VB. When you add a combobox to the interface you can edit the options in the property "Items". These items can be manipulated and read by the code. Below is a program that allows the user to order a pizza from a list in the combobox "cbxPizzaType". They can then select a size from "cbxSize".

You can see that each item is allocated a case number. The first case is always set to zero. In the case of cbxPizzaType Case 0: the dblPrice is assigned 1.0 and the strType is assigned the SelectedItem. This means the text that is displayed for that item will be read as a string into the strType variable.

```
Public Class PizzaOrderingSystem
  Private Sub BtnOrder_Click(sender As Object, e As EventArgs) Handles btnOrder.Click
    Dim dblPrice As Double
    Dim dblCost As Double
    Dim strType As String = ""
    Dim strSize As String = ""

    Select Case cbxPizzaType.SelectedIndex
      Case 0
        dblPrice = 1.0
        strType = cbxPizzaType.SelectedItem.ToString
      Case 1
        dblPrice = 1.0
        strType = cbxPizzaType.SelectedItem.ToString
      Case 2
        dblPrice = 2.5
        strType = cbxPizzaType.SelectedItem.ToString
      Case 3
        dblPrice = 3.1
        strType = cbxPizzaType.SelectedItem.ToString
    End Select

    Select Case cbxSize.SelectedIndex
      Case 0
        dblCost = dblPrice + 10.5
        strSize = cbxSize.SelectedItem.ToString
      Case 1
        dblCost = dblPrice + 14.5
        strSize = cbxSize.SelectedItem.ToString
      Case 2
        dblCost = dblPrice + 16.6
        strSize = cbxSize.SelectedItem.ToString

    End Select

    MsgBox(" You have ordered a " & strSize & " sized " & strType & " pizza. Your order will be $" & dblCost)
  End Sub
End Class
```

Extension Task

Add images of different pizzas using picture boxes. Allow the user's choice of pizza type set the visibility of the picture box to TRUE. Let's call our picture box object of the Hawaiian Pizza "pbxHawaiian" we can set the image to invisible by setting the visibility property to FALSE. When the Case for Hawaiian is selected it can be set to TRUE:

```
pbx Hawaiian.Visible = TRUE
```

Hotel Programming Task
Stage 2

Complete this programming Task to use what you learned from activities three, four and five:

Now let's continue to build our Hotel Booking System that allows the customer to choose between types of rooms. The Basic Room at $89 per night, the Superior Room at $96 per night and the Grand Room at $107 per night. It should read in the number of guests and calculate the number of nights' stay. All inputs should be validated. Integrate ComboBoxes into your interface to ensure valid input. Provide photos of different rooms as they are chosen.

Iteration

Counted Loops in VB

Visual Basic uses a FOR loop to repeat procedures for a specific number of times. Below is a loop that will read in ten values and add them up.

```
For i = 0 to 9
        Number = InputBox("Enter a Number")
        Total = Total + Number
Next i
```

Trace Table: Counted Loop

i	Number (Test Data)	Total
0	2	2
1	6	8
2	3	11
3	5	16
4	2	18
5	4	22
6	1	23
7	3	26
8	4	30
9	7	37

Pre-Test Loops in VB

VB uses "WHILE" to pre-test iteration. See the simple example below.

```
While Total < 10
        Number = InputBox("Enter a Number")
        Total = Total + Number
End While
```

Trace Table: Pre-Test Loop

Total	Number (Test Data)
0	6
6	2
8	5

Post-Test Loops in VB

VB uses "REPEAT... UNTIL" for post-test loops.

```
Repeat
        Number = InputBox("Enter a Number")
        Total = Total + Number
Until Total > 10
```

Trace Table: Post-Test Loop

Total	Number (Test Data)
0	6
6	2
8	5
13	

We will explore pre-test and post-test loops later when we look at other data structures and accessing stored data.

Activity Six: Create a Guessing Game

The program will generate a random number between 0 and 10. There are two picture boxes PbxLose and PbxWin that are set to False which make them invisible. A counted loop will allow the user to enter 3 guesses. For each guess it is tested if the value entered is correct, higher or lower than the Answer. If they get the answer right within the three guesses the PbxWin will become visible, otherwise the PbxLose will become visible. You can see that each guess is added to a listbox called lbxGuesses if the guess is incorrect. When the game is won, the program exits the FOR loop. Can you add validation to this program?

```
Public Class Game

  Private Sub btnPlay_Click(sender As Object, e As EventArgs) Handles btnPlay.Click
    Dim Guess As Integer
    Dim CountDown As Integer
    Dim Answer As Integer
    Dim Flag As Boolean = False
    Dim RandomNumber As New Random
    Answer = RandomNumber.Next(0, 10)

    PbxLose.Visible = False
    PbxWin.Visible = False
    For i = 1 To 3
      CountDown = 3 - i
      Guess = InputBox("Guess the number! ")

      If Guess = Answer Then
        Flag = True
        Exit For
      ElseIf Guess < Answer Then
        MsgBox("Higher! You have " & CountDown & "  guesses left.")
        lbxGuesses.Items.Add(Guess)
      ElseIf Guess > Answer Then
        MsgBox("Lower! You have " & CountDown & "  guesses left.")
        lbxGuesses.Items.Add(Guess)
      End If

    Next i
        If Flag = True Then
            PbxWin.Visible = True
        Else
            PbxLose.Visible = True
        End If
  End Sub
End Class
```

DATA STRUCTURES

Arrays

Arrays in VB are very straightforward. The 1D array below can be defined in VB as:

```
Dim ValuesArraySingle(5) as Integer
```

Index	0	1	2	3	4	5
Data	245	435	67	345	543	765

The 2D array below can be similarly defined:

```
Dim ValuesArrayMatrix(5, 2) as Integer
```

Index	0	1	2	3	4	5
0	53	456	75	94	89	531
1	42	636	34	624	791	95
2	33	45	846	23	118	35

```
ValuesArraySingle(0) = 245
```

```
ValuesArrayMatrix(0,0) = 53
```

We can easily assign values to our array using pre-test loops. The Pre-Test Loop below reads in values from the user into a 1D array called ValuesArraySingle(5). The index is set to zero and each time the loop is executed one is added to the index. Once the index reaches 6 the loop will stop.

```
Index = 0
While (index < = 5)
        Value = InputBox("Please enter Value")
        ValuesArraySingle(index) = Value
        Index = index +1
End While
```

With 2D arrays we need to nest our loops. This code will read data into ValuesArrayMatrix(5,2).

```
Column = 0
Row = 0
While (Column< = 5)
        While(Row <= 2)
                Value = InputBox("Please enter Value")
                ValuesArrayMatrix(Column, Row) = Value
                Row = Row + 1
        End While
        Column= Column +1
End While
```

Records

A record can be stored in an XML file which can be accessed by VB. An XML file is simply a text file created in a basic text editor saved with the .xml extension.

This is an example Record:

ID Number	Name	Address	Phone Number
2826294	Brian Cross	18 Smith St, Urberton	0408239628

Here is the XML Structure for the same record.

```
<Members>
        <Member ID>2826294</Member ID>
        <Member Name> Brian Cross </Member Name>
        <Member Address > 18 Smith St, Urberton </Member Address >
        <Member Phone > 0408239628 </Member Phone >
 </Members>
```

VB can read from an XML file, edit an XML file and can create XML files to store data for later retrieval.

EXTERNAL STORAGE

To keep our data so we can access it after our software has been closed we need to store it into a file. We can use plain text files, XML files or store it in a database.

XML File Storage

XML files are structured like HTML – using tags:

```
<tag> data </tag>
```

You can create XML in Excel and output the data in XML format. You can also create XML in VB but it is a little more complex. For now, let's create an XML file in Notepad with records for some bank account users. Firstly we are going to create an XML file using Visual Basic. The following code imports XML system and uses a module to create each field in the record. You can use your knowledge of loops to expand on this code.

```
Imports System.Xml
Public Class WritingXML
        Private Sub btnExportXML_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
        Handles btnExportXML.Click

                Dim writer As New XmlTextWriter("product.xml", System.Text.Encoding.UTF8)
                writer.WriteStartDocument(True)
                writer.Formatting = Formatting.Indented
                writer.Indentation = 2
                writer.WriteStartElement("Table")

                createNode(1, "Beans", "$2.50", writer)
                createNode(2, "Bread", "$3.50", writer)
                createNode(3, "Milk", "$1.90", writer)
                createNode(4, "Rice", "$3.00", writer)

                writer.WriteEndElement()
                writer.WriteEndDocument()
                writer.Close()
        End Sub
```

```
Private Sub createNode(ByVal intID as Integer, ByVal strProductName As String, ByVal dblPrice As Double, ByVal writer As
XmlTextWriter)
                writer.WriteStartElement("ProductID")
                writer.WriteInteger(intID)
                writer.WriteEndElement()

                writer.WriteStartElement("ProductName")
                writer. WriteString(strProductName)
                writer.WriteEndElement()

                writer.WriteStartElement("ProductPrice")
                writer.WriteDouble(dblPrice)
                writer.WriteEndElement()
        End Sub
End Class
```

Activity Seven: Storing a Soccer Team's Data

ARRAYS

In this program we are going to manage a soccer team! We will use an array to store our player details and scores.

```
SoccerTeamArray(10, 3) As String
```

You can see the code below will fill the first column with numbers 1 - 11 and the second column with the positions on the soccer team. We still have two more columns(2 and 3) to fill.

```
Team(0, 0) = "1"
Team(0, 1) = "Goal Keeper"
Team(1, 0) = "2"
Team(1, 1) = "Right Full Back"
Team(2, 0) = "3"
Team(2, 1) = "Left Full Back"
Team(3, 0) = "4"
Team(3, 1) = "Centre Half Back"
Team(4, 0) = "5"
Team(4, 1) = "Centre Half Back"
Team(5, 0) = "6"
Team(5, 1) = "Defensive Midfielder"
Team(6, 0) = "7"
Team(7, 0) = "8"
Team(7, 1) = "Central Mid-Fielder"
Team(8, 0) = "9"
Team(8, 1) = "Striker"
Team(9, 0) = "10"
Team(9, 1) = "Attacking Midfielder"
Team(10, 0) = "11"
Team(10, 1) = "Left Winger"
```

We will need iteration to fill our array with player names and their scores. The following code will ask the user eleven times to enter each player name by prompting the position they played and then the score they have achieved.

```
For i = 0 to 10
  SoccerTeamArray(i, 2) =input("Add the player who plays position: " & SoccerTeamArray(i, 1) )
  SoccerTeamArray(i, 3) =input("Add the score for: " & SoccerTeamArray(i, 2) )
Next
```

Extension Task

Now create a search button that will read in the player number (1 - 11), and it returns that name of the player and their position.

RECORDS

If we were managing a team, we would need to keep more information about our players such as their phone number and any health issues. We may need to keep our player information in records, this means we can store our details in an XML file. Create a form that allows the user to enter the player's First Name, Second Name, Phone Number and Address into textboxes. Create a button that runs the following code. In the first section it identifies the XML file and declares the records and the document. The second section reads in the data from the text boxes. The third section creates a record in the document and the final section adds the data to the record and closes the document. This code will not run unless the XML file is created.

```
Dim xelement As XElement = XElement.Load("Players.xml")
Dim Records As IEnumerable (Of XElement) = xelement.Elements()
Dim Document As XDocument
Document =  XDocument.Load ("Players.xml")

Dim FirstName As String = txtFirstName.Text
Dim LastName As String = txtLastName.Text
Dim Phone As String = txtPhone.Text
Dim Address As String = txtAddress.Text

Dim root= New XElement("Player")
Dim FirstNameElement = New XElement("First_Name", FirstName)
Dim LastNameElement = New XElement("Last_Name",LastName)
Dim PhoneElement = New XElement("Phone", Phone)
Dim AddressElement = New XElement("Address", Address)

root.Add(FirstNameElement, LastNameElement, PhoneElement, AddressElement)
Document.Root.Add(root)
Document.Save("Players.xml")
```

The code above requires that the document has already been created so it can continue to add records. Create a text document and save it as "Players.xml". It must contain the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<Records>
    <Player>
        <First_Name> Bob </First_Name>
        <Last_Name> Phrelan </Last_Name>
        <Phone> 0450 555 444</FPhone>
        <Address> 8 High St, Waverley </Address>
    </Player>
</Records>
</xml>
```

Each time the button is clicked a new record is created in the XML file. Now put it all together to allow input for each player record to be stored in the XML file. Don't forget the "Imports System.Xml" at the beginning of your code. So we have managed to save data to a file using XML. Let's now retrieve that data. Create a Search Button and a Listbox that will display the record details. In the code n the next page, we declare the playerlist and load the Players.xml file. We are going to search by the tag name "player". We set the flag "Found" to false and read in the name of the player we are looking for from a textbox.

Once the name has been entered, a For Each loop will check the "First_Name" element for each "Player". If it is found it will set the "found" flag to TRUE and display each of the fields in the record in the listbox. If the name is not in the XML file a message will display.

```
Dim PlayerList AS New XmlDocument
PlayerList.Load("Players.xml")
Dim xmlPlayerList As XmlNodeList = PlayerList.GetElementsByTagName("Player")
Dim Found As Boolean = False
Dim Search As String = txtSearch.Text

For Each player AS XmlElement In xmlPlayerList
        If Search = (player("First_Name").InnerText) Then
                Found = True
                ListBox1.Items.Add("First Name: "  + (player("First_Name").InnerText))
                ListBox1.Items.Add("Last Name: "  + (player("Last_Name").InnerText))
                ListBox1.Items.Add("Phone Number: "  + (player("Phone").InnerText))
                ListBox1.Items.Add("Address: "  + (player("Address").InnerText))
                Exit For
        Else
                MsgBox(Search & " is not in the team")
        End If
Next
```

Hotel Programming Task
Stage 3

Complete this Programming Task to use what you learned from iteration and data structures:

Continuing with our Hotel Booking System, let's store room information in an array. Our hotel has 25 rooms and each room has the following data to be stored:

- Room Number
- Sleeps (Number of people it sleeps = 1, 2, 3, 4 or 5)
- View (1= Yes, 0=No)
- Occupied (1= Yes, 0=No)

Ensure you store details for each of the 25 rooms in an array. You must include a feature that allows the user to search for a room by Sleeps, View and Occupied.

We need to keep records of our customers and store them in an XML file. Allow the user to enter the following details:

- Customer Given Name
- Customer Family Name
- Car Registration
- Phone Number
- Room Number Allocated

When a customer checks into a room, it automatically sets the room occupation to "Yes".

Activity Eight: Modularisation

Functions and Subroutines

When we need to call up a repeated set of procedures to calculate something we call up a function. In the example below there are five global variables available to all the subroutines and functions within the class. The first subroutine is executed when Button is clicked. It reads in two values and then calls up something called "AddNumbers". If you look further below you can see a Public Sub called AddNumbers. It requires the two variables: intNumber1 and intNumber2 which need to be identified when the subroutine is called. At the initialisation of the subroutine, they need to be identified using "ByVal". This passes the variables into the new subroutine where intAnswer is calculated. Once AddNumbers() is returned, it passes intAnswer back to the Button subroutine. The next line calls up a function called "OddEvenCheck" which requires the variable intAnswer. The OddEvenCheck() function calculates a Mod 2 function to return the variable intRemainder. Now the rest of the program can execute.

If intRemainder = 0 then the intAnswer can be evenly divided by 2 and is therefore an even number. Finally the user is told if the sum of the two values is odd or even. Notice there are comments in the code to make it easier to read. Use a single inverted comma to set comments.

```
Public Class CheckSum                                   'CheckSum Program Author: V Farrell
                                                        'January 2020
Dim intNumber1 As Integer                               'Declaration Statements for input variables
Dim intNumber2 As Integer                               'intNumber1 and intNumber2 as well as the
Dim intAnswer As Integer                                'Sum of the two values and the remainder
Dim intRemainder As Integer
Dim strCheck as String

 Public Sub Button_Click
        intNumber1 = txtNumber1.Text                    'Reads in the two values
        intNumber 2= txtNumber2.Text

        '
        Call AddNumbers(intNumber1, intNumber2)         'Calls up the Subroutine "AddNumbers

        '
        Call OddEvenCheck(intAnswer)                    'Calls up the Function "OddEvenCheck


        If intRemainder = 0 Then                        'Tests if there is a remainder when divided by 2
                Check = "Even"
        Else
                Check = "Odd"
        End If                                          'Determines if the sum is odd or even

        MsgBox("The sum of your entered values is " & strCheck)
End Sub


Public Sub AddNumbers(ByVal intNumber1 As Integer, ByVal intNumber2 As Integer)     'AddNumbers is a function
        intAnswer = intNumber1 + intNumber2                                         'that adds the two input values
        Return intAnswer
End Sub

Public Function OddEvenCheck(ByVal intAnswer As Integer)    'OddEvenCheck divides the sum by 2 to find
        intRemainder = intAnswer Mod 2                      'a remainder
        Return Remainder
End Function

End Class
```

Forms as GUI

When organising the structure of our program we need to consider the order of screens and how the data is collected and displayed. Each screen is designed as a form in Visual Studio. If more than one form is used, there needs to be some planning in which order the forms are accessed.

To open Form2 use the "show" function:

```
Form2.Show()
```

To close the current Form use the "close" function:

```
me.Close()
```

You can also have control over the visibility of the objects on the form. In the example below a textbox (txtDeliveryLocation) is not visible until a checkbox has been checked:

```
txtDeliveryLocation.Visible = False

If cbxDelivery.Checked = TRUE Then
        txtDeliveryLocation.Visible = True
        strAddress = txtDeliveryLocation.Text
End If
```

You can also use the hide and show functions:

```
txtDeliveryLocation.Hide()

If cbxDelivery.Checked = TRUE Then
        txtDeliveryLocation.Show()
        strAddress = txtDeliveryLocation.Text
End If
```

Hotel Programming Task
Stage 4

Complete this Programming Task to use what you learned from modularisation.

Our Hotel Booking Software needs to be organised into the following sections with separate screens:

- Booking a room
- Checking In
- Checking Out with Extra Charges (room service, laundry service, taxi booking, mini bar, etc.)
- Changing a Booking
- Any room maintenance required (new carpet, fix a tap, holes in the wall etc.)

The system must have a user password access for hotel staff and a different level of access for the hotel manager. The manager needs to keep records of staff salaries, cleaning and maintenance costs. The staff should not have access to the management modules of the software.

Ideas for your Programming Folio

Time Zone Calculator
An application that allows the user to select from a list of major cities around the world to reveal their time zones. The user can then choose two cities and it will calculate the number of hours difference.

Activity Tracker Data Analyser
An application that can analyse data collected from a wearable activity tracker. The application could read in data from a file that contains values related to:
- UV Exposure
- Number of Steps
- Hours of Activity
- Hours of Sleep
- Heart Rate Levels
- Blood Pressure

Food Delivery App
Any application that allows the user to order food from a menu for delivery or pick up. The user should be able to order any combination and quantity of products. It should store previous orders so they can be re-ordered.

Hire Service Booking System
Any organisation that hires out equipment or vehicles requires a system to manage the data. A vehicle hire booking system would require:
- Prices per day
- Different rates for different equipment/vehicles
- Keeping track of which vehicles in the fleet are available for hire
- Calculated costs and fees.

An Arcade Game
Any kind of game where scores are saved and retrieved to reload progress.

Weather Data Analyser
Collect data from the Bureau of Meteorology on temperature and rainfall. Develop an application that can analyse the data and present the data as a graph.

Department Store Kiosk
All large shopping centers provide interactive maps for customers to find the store they are looking for. Some of the fancier shopping centers offer a valet service. Any information system built to manage the data to support customers would also be a suitable project.

Online Help

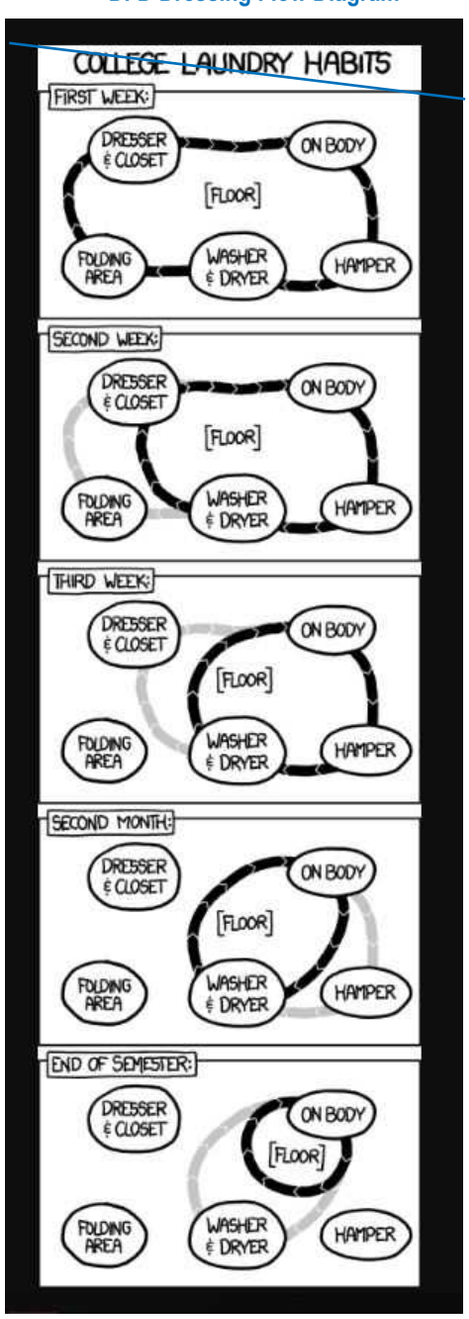
Examples of Unit 3 Outcome 1 SAC activities visit:	**www.vicfarrell.com.au**
Software Development Requirements visit:	**www.vcaa.vic.edu.au**
Help with Visual Basic and Visual Studio:	**YouTube - Passy's World of ICT** **www.VBtutor.net** **www.tutorialspoint.com**

Online Support

Sometimes you need help solving a problem with your project There are some key online help you can tap into.

- **stackoverflow.com/ (search for a solution before posting a question!)**
- **thecodingguys.net/tutorials/visualbasic/vb-tutorial**
- **ocs.microsoft.com/en-us/visualstudio/get-started/visual-basic/**

DFD Dressing Flow Diagram



https://xkcd.com/1066/

Programming in PHP

PHP stands for PHP Hypertext Preprocessor. It is a widely used web language that is

STARTING UP

You need to develop the interface as a website in HTML & CSS

Use Awardspace.com or XAMP to run a web server.

Assessment Tasks

School-Based Assessment

U3O1 Programming Folio (SAC)

U3O2 Project Plan (SAT Criterion 1)

U302 Software Requirements Specifications (SAT Criterion 2 & 3)

U3O2 Design Folio (SAT Criterion 4)

U4O1 Software Solution (SAT Criterion 5 & 6)

U4O1 Evaluation Report (SAT Criterion 7 & 8)

U4O2 Cybersecurity Report (SAC)

Assessment Structure

The Software Development Assessment Structure is covered by a number of VCAA documents on the VCAA website. This is a brief summary of the information available at the time of printing. Any updates made by VCAA after November 2019 will be summarised on the following website:

www.vicfarrell.com.au

The overall course structure weightings: VCE Examination 50% and School-Based Assessment 50%

School-Based Assessment

The school based assessment is made up of three separate tasks:

1. U3O1 SAC (Weighting 10%) Marking guidelines are provided by VCAA.
2. U3O2 & U4O1 SAT (Weighting 30%) The SAT is made up of 8 Criteria. Each Criterion is assessed out of 10 Marks. The criteria are provided by VCAA.
3. U4O2 SAC (Weighting 10%) Marking guidelines are provided by VCAA.

Deliverables

The School-Based Assessment is composed of seven separate deliverables. The time allocation listed is an approximate and should be only used as a guide to fit into your own school time constraints.

U3O1 Programming Folio (SAC)

A series of software solutions that demonstrate the programming requirements for the SAT. Students solve programming problems provided by the teacher.

Assessment: 100 Marks
Weighting: 10%
Time Allocation: Approx. 350 Minutes

U3O2 Project Plan (SAT Criterion 1)

A Gantt Chart to plan the SAT and a project journal to prepare for U4O1 Criteria 8.

Assessment: 10 Marks
Time Allocation: Approx. 250 Minutes

U3O2 Software Requirements Specifications (SAT Criteria 2 & 3)

The Software Requirements Specifications presents a summary of the analysis.

Assessment: 20 Marks
Time Allocation: Approx. 300 Minutes

U3O2 Design Folio (SAT Criterion 4)

A folio of proposed design solutions for the programming task.

Assessment: 10 Marks
Time Allocation: Approx. 200 Minutes

U4O1 The Software Solution (SAT Criteria 5 & 6)

The completed software solution.

Assessment: 20 Marks
Time Allocation: Approx. 400 Minutes

U4O1 Evaluation Report (SAT Criteria 7 & 8)

An overview of testing tools and usability testing to evaluate the solution.

Assessment: 20 Marks
Time Allocation: Approx. 250 Minutes

U4O2 Cybersecurity Report (SAC)

A structured report on the security issues of a case study provided by the teacher.

Assessment: 100 Marks
Weighting: 10%
Time Allocation: Approx. 100 Minutes

U3O1 Programming Folio (SAC)

A series of software solutions that demonstrate the programming requirements for the SAT. Students solve programming problems provided by the teacher.

Assessment: 100 Marks
Weighting: 10%
Time Allocation: Approx. 350 Minutes

VCAA Programming Requirements

In the development of the U3O1 programming folio and U4O2 SAT software solution it is required that students demonstrate the problem-solving methodology within three conceptual layers: interface, logic and data source with their chosen programming language. Chapter 5 of this text book covers all the requirements using Visual Basic. Students are not restricted to only one language for their SAT, however, it is advisable to use the programming language your teacher is most proficient in so they can assist you.

You must be able to produce the following in your chosen programming language:

1. Interface

Programming requirements for the interface layer:

Develop a graphical user interface (GUI) for use in digital systems such as desktop computers, laptops, tablets, smart phones, gaming consoles, robotic devices and smart home devices.

Note that databases are not to be used in the interface layer.

2. Logic

Programming requirements for the logic layer:

- construct and use data structures
- design and apply data validation techniques
- use program control structures: selection, iteration and sequencing
- use modularisation and code optimisation
- use classes, methods and event-driven programming functions

3. Data source

Programming requirements for the data source layer:

- design, construct and use external storage and access technologies
- retrieve data from external sources.

Level of Complexity

The Programming Folio should comprise three or more separate small tasks that demonstrate the requirements listed above. There are some common questions that arise about the level of complexity expected. Assume that there should be nested IF Statements and Iteration. Recursion would be a great demonstration of understanding and mastering complex algorithms. Some teachers expect students to use a Quick Sort or a Binary Search in one of the programming tasks.

Example U301 SAC Tasks

1. Food Ordering App

Acme Noms is a local cafeteria that wants a basic mobile device App to process lunch orders sent to the store where they can have them ready for the customer to pay for and collect when they arrive.

Functional Requirements

The mobile App needs to collect the lunch order of a customer, calculate the cost of that order and keep a record of the order in an array so that customers can re-order at the touch of a "favourite" button.

The solution is required to collect the following input data through a Validation process:

- First name of the Customer
- Food Selection
- Food Type Selection
- Drink Selection
- Time of Pick Up
- Save Order as Favourite

The solution is required to produce the following output:

- A saved a record of the customer order for use in the future
- The cost of the order
- A summary of the order displayed on the screen.

The solution needs to calculate the order based on data held in a file. Add as many types of food and drink as you prefer.

Data Example:

- Food: Pie
- Type: Beef- $3.50
- Type: Chicken - $4.00
- Food: Sandwich
- Type: Salad- $2.50
- Type: Ham - $4.00
- Drink: Water
- Type: Pump - $4.00
- Type: Mr Franklin - $2.50

Non-Functional Requirements

Within the dimensions of a mobile phone screen, the interface design must be intuitive for the user and follow the conventions of all mobile apps.

Font and colour choices must be consistent with the sample screens below. Developers must use the logo provided and muted greys with OliveDrab (#6B8E23).

Constraints

The solution is to be used on a mobile device and therefore the screen resolution and design must be appropriate for a phone or tablet. You have to use the Visual Basic programming language. Identify the dates during class time you can be working on their SAC. For example:

- Wednesday 28 February
- Thursday 1 March
- Friday 2 March
- Tuesday 6 March
- Wednesday 7 March
- Tuesday 13 March
- Wednesday 14 March
- Thursday 15 March
- Friday 16th March DEADLINE

Scope

The program needs to keep only ONE lunch order for the user. The program does not have to connect to a phone number or email to be sent through a phone network. Remember it is only a prototype. It does not transfer funds as the customer will pay on pick up.

Design Requirements provided by Acme Noms

Suggested Data Dictionary

Variable Name	Data Type	Size	Description
StrCustName	String	12	This is the first name of the customer
dblHotFoodPrice	Double (Floating Point)	4	This will handle the price of the Hot Food items.
dblSandwichPrice	Double (Floating Point)	4	This will handle the price of the sandwich items.
dblSaladPrice	Double (Floating Point)	4	This will handle the price of the Salad items.
intAmount	Integer	1	This will be the amount of each selected product.
strHotFood	String	10	The name of the Hot food product selected
strSandwich	String	10	The name of the type of sandwich selected
strSalad	String	10	The name of the type of salad selected.
dblDrink	Double (Floating Point)	4	The price of the selected drink
strDrink	String	10	The name of the selected drink
dblFavouriteFoodPrice	Double (Floating Point)	4	Price of the Saved Food Order
strFavouriteFood	String	10	The name of the Saved Food Order
dblFavouriteDrink	Double (Floating Point)	4	Price of the Saved Drink Order
strFavouriteDrink	String	10	The name of the Saved Drink Order

Assessment Deliverables

1. Create an Algorithm in pseudocode that designs the solution for the mobile app.
2. Create the Acme Noms Mobile App in Visual Basic
3. Complete the TESTING TABLE below for your solution to check ALL possible outputs.
4. Internal Documentation

Item Tested	Test Data	Expected Result	Actual Result

2. Wearable Health Monitoring Device App

The new "iCare" device is a wearable health monitoring product that collects the following data:

- Heart rates throughout the day (sampled every hour)
- Number of steps per day.
- Amount of UV exposed to per hour
- The hour you fall asleep and the hour you wake up each day
- Sedentary behaviour periods (any period >1 hour creates an alert)
- Stress levels (sampled every hour)
- Blood pressure levels (sampled every hour)

Due to the small nature of the iCare wearable device, reading and manipulating the data needs to occur on a laptop or tablet. When the user wishes to investigate their "iCare" device to get advice, they simply use the Bluetooth connection and access the data using their "iCare for your Health" Application.

The application will allow the user to login to secure their private records. The login details must be stored in an Access Database or hash table.

Once logged in, the application can access an array stored on the device that holds all the data. The application processes the data in this array to display:

- Average heart rate over 24 hours
- If the goal of 10,000 steps as achieved over the 24 hours
- At what time UV levels were >6
- Number of hours sleeping
  (Less than 7 hours = too little sleep. Greater than 9 hours = too much sleep)
- Number of hours of sedentary behaviour when not asleep
- At what times was Stress = High over 24 hours?
- At what times was Blood Pressure = High?

There is a 2D Array on the next page. You must use this as your sample data. Each sample starts at the beginning of the hour and enters data for each time period. In the first row of the "Steps" column, it stores the previous day's number of steps.

Functional Requirements

The application is required to enter the following data Input:

- The 2D array listed on page 133

At the click of a button the calculated Output must include:

- Average heart rate
- Number of steps
- Times of high UV
- Number of hours sleeping
- Number of hours of sedentary behaviour
- Times of high Stress
- Times of high blood pressure

Time	Heart Rate	UV Exposure 0 - 10	Sedentary / Active	Stress 0 - 5	Blood Pressure Low, Raised, High	Steps for the previous day	Sleep
00:00	70	0	Sedentary	0	Normal	8,457	Sleep
01:00	71	0	Sedentary	0	Normal		Sleep
02:00	71	0	Sedentary	0	Normal		Sleep
03:00	71	0	Sedentary	0	Normal		Sleep
04:00	70	0	Sedentary	0	Normal		Sleep
05:00	70	0	Sedentary	0	Normal		Sleep
06:00	71	0	Active	2	Raised		Awake
07:00	81	0	Active	2	High		Awake
08:00	75	3	Active	2	Raised		Awake
09:00	85	3	Sedentary	2	Normal		Awake
10:00	88	6	Sedentary	2	Normal		Awake
11:00	81	6	Active	5	Normal		Awake
12:00	80	8	Active	4	Normal		Awake
13:00	92	4	Sedentary	2	Normal		Awake
14:00	94	4	Sedentary	2	Normal		Awake
15:00	89	4	Active	2	Normal		Awake
16:00	90	4	Active	2	Normal		Awake
17:00	92	4	Sedentary	2	Normal		Awake
18:00	86	0	Sedentary	0	Normal		Awake
19:00	79	0	Sedentary	0	Normal		Awake
20:00	78	0	Sedentary	0	Normal		Awake
21:00	76	0	Sedentary	0	Normal		Sleep
22:00	73	0	Sedentary	0	Normal		Sleep
23:00	72	0	Sedentary	0	Normal		Sleep

Non-Functional Requirements

As a prototype, your application can use the dimensions for any laptop or tablet. Within these dimensions the interface design must be intuitive for the user and follow the conventions of all applications.
Font and colour choices must evoke a professional feel including the following Web colours "Light Slate Grey" and "Crimson".

Constraints

You must use the test data provided (the 2D array). The login details must be stored in an Access Database or hash table. One user login credentials only.
You must use the Visual Basic programming language during class time only, on the dates provided for example:

- Wednesday 28 February
- Thursday 1 March
- Friday 2 March
- Tuesday 6 March
- Wednesday 7 March
- Tuesday 13 March
- Wednesday 14 March
- Thursday 15 March
- Friday 16th March DEADLINE

Scope

Your application is only a prototype and does not need to save any data. However it must display all calculated output to the screen. Use your knowledge of VB objects to solve your interface design problem in a creative way.

Suggested Data Dictionaries

Variables	Data Type	Size	Description
IntAverageHeartRate	integer	2	The average heart rate of the day
IntSteps	integer	5	Number of steps
TimeUVWarning	time	5	Time user was exposed to UV over 6
IntHoursSleep	integer	1	Number of hours asleep
IntSedentaryHours	integer	1	Number of instances when Sedentary occurs each hour while awake.
TimeHighStress	time	5	At what time was Stress = "High"
IntSedentary	integer	5	Number of hours Sedentary
TimeBloodHigh	time	5	At what time Blood Pressure = High
IntCounter	integer	2	Used to count data items

ArrayName	Data Type	Values	Description
iCare[23, 6]	String	Times (0.00am – 1.00pm) Heart Rate (40 – 100) UV (0 – 10) Sedentary / Active Stress (0 – 5) Blood Pressure (Normal Raised High) Steps Sleep/Awake	Collected data from file

Suggested Algorithm

```
START

    Set Array iCare(23, 6) (Global)

    SubRoutine: READ IN DATA TO ARRAY

        Access data files: HeartRate, UV, Sedentary, Stress, BloodPressure, Steps and Sleep.

        While HeartRate file is not at the end
            For i ← 0 to 23
                    Read in each data item into iCare(i, 1)
        End While

        While UV file is not at the end
            For i ← 0 to 23
                    Read in each data item into iCare(i, 2)
        End While

        While Sedentary file is not at the end
            For I ← 0 to 23
                    Read in each data item into iCare(i, 3)
        End While

        While Stress file is not at the end
            For I ← 0 to 23
                    Read in each data item into iCare(i, 4)
        End While
```

```
        Read in IntSteps ← Steps file.

        While BloodPressure file is not at the end
            For i ←  0 to 23
                    Read in each data item into iCare(i, 5)
        End While

        While Sleep file is not at the end
            For i ←  0 to 23
                    Read in each data item into iCare(i, 6)
        End While

END SubRoutine: READ IN DATA TO ARRAY

SubRoutine: CALCULATE AVERATE HEART RATE

        Set ROW to integer
        Set COLUMN to integer
        Set TotalHeartRate to integer
        Set AverageHeartRate to Double

        Repeat for ROW ← 0 to ROW ←  23
            TotalHeartRate = TotalHeartRate + iCare(ROW, 1)
        End Repeat Loop

        AverageHeartRate ←  TotalHeartRate/24

        Display Message "Your average heart rate throughout the day was: "AverageHeartRate"

END SubRoutine: CALCULATE AVERAGE HEART RATE

SubRoutine: NUMBER OF STEPS

        Set IntStep

        IF InStep < 10000 is true THEN
            Display Message: You have not reached your goal of 10000 steps today"
        ELSE
            Display Message: Congratulations you reached your goal!
        END IF

END SubRoutine: NUMBER OF STEPS


SubRoutine: UV LEVELS

        Set ROW to integer
        Set COLUMN to integer

        Repeat for ROW ←  0 to ROW ←  23

            IF iCare(ROW, 2) >6 is true THEN
                    Display: "You were exposed to high UV at:" iCare(ROW, 0)
            ENDIF

        End Repeat Loop

END SubRoutine: UV LEVELS
```

```
SubRoutine: SEDENTARY

        Set ROW to integer
        Set COLUMN to integer
        Set IntCounter to integer

        Repeat for ROW ← 0 to ROW ← 23
            IF iCare(ROW, 3)← Sedentary AND (ROW, 6) ← Awake is true THEN
                    IntCounter ← IntCounter + 1
            ENDIF
        End Repeat Loop

        Display: "You have spent" IntCounter "hours in sedentary behaviour today".

END SubRoutine: SEDENTARY

SubRoutine: STRESS

        Set ROW to integer
        Set COLUMN to integer

        Repeat for ROW ← 0 to ROW ← 23
            IF iCare (ROW, 4) > 4 is true THEN
                    Display: "You experienced high stress at:" iCare(ROW, 0)
            ENDIF
        End Repeat Loop

END SubRoutine: STRESS

SubRoutine: BLOOD PRESSURE

        Set ROW to integer
        Set COLUMN to integer
        Set IntCounter to integer

        Repeat for ROW ← 0 to ROW ← 23
            IF iCare(ROW, 5)← High true THEN
                    Display: "You experienced high stress at:" iCare(ROW, 0)
            ENDIF
        End Repeat Loop

END SubRoutine: BLOOD PRESSURE

SubRoutine: SLEEP

        Set ROW to integer
        Set COLUMN to integer
        Set IntCounter to integer

        Repeat for ROW ← 0 to ROW ← 23
            IF iCare(ROW, 6)← Sleep is true THEN
                    IntCounter ← IntCounter + 1
            ENDIF
        End Repeat Loop

        Display: "You have had" IntCounter "hours of sleep in the past 24 hours".

END SubRoutine: SLEEP

END
```

U3O2 Project Plan (SAT Criteria 1)

Assessment: 10 Marks
Time Allocation: Approx. 250 Minutes

Gantt Chart

When a project plan begins, we need to consider the due date of the final submission. Once you have due dates for each of the criteria for the SAT you will have a starting point for your Gantt Chart. These will be milestones. Your tasks should be organised into the four stages of the problem solving methodology and then into each of the activities. It is very important that the tasks you identify inside each of these sections relate uniquely to your project. Do not rely entirely on the problem solving methodology as your list of tasks. Break everything down into specific outcomes that need to be completed. For example: Your SRS is made up of different sections, so you can identify them as separate tasks. Your development stage should be as detailed as possible, including all the problems you need to solve and test.

For a complete plan you must include:

- All key tasks, organised into the stages and activities of the problem solving methodology
- Relevant milestones
- Key dependencies between the tasks
- Some slack time to allow for a contingency plan
- Resources required for each task (client, printer, analysis tools etc.)
- Reasonable time allocations

For a complete, accurate and coherent Gantt Chart it should be:

- Easy to read
- Easy to use
- Modifiable
- Displaying a clear critical path
- Complete with correct milestones
- Complete with correct Dependencies
- Complete with accurate and practical sequencing.

Gantt Chart Tools

The main software package used for Gantt Charts is Microsoft Project, but there are other free alternatives such as GanttProject. GanttProject allows for importing and exporting to Microsoft Project.

https://www.ganttproject.biz/

Excel can also be used to create a Gantt Chart with free downloadable templates. One online location where serviceable templates can be found, is on TeamGantt:

https://www.teamgantt.com/free-gantt-chart-excel-template

Journal

Students who set up and commit to their journal notes at the beginning of the project are more likely to produce a better Evaluation Report at the end. Criteria 8 requires that students apply a "range of appropriate recording techniques" when logging progress throughout the project. It is advisable to set up the journal to collect regular reflections such as:

- Date
- Project progress (what was completed)
- Issues or problems that need to be solved
- Adjustments to plan and reasons for the changes
- Screen grabs of the project
- Screen grabs of the Gantt Chart
- Feedback from client, teacher, peers
- Reflection on Gantt Chart deadlines and how the plan was inadequate or useful.

It is possible to create a template for a journal to assist in keeping records.

U3O2 Software Requirements Specifications (SAT Criteria 2 & 3)

Assessment: 20 Marks
Time Allocation: Approx. 300 Minutes

The SRS Document

The SRS responds to both Criteria 2 and 3. Criterion 2 is the analysis while Criterion 3 is the interpretation of that analysis into detailed functional and non-functional requirements.

The SRS document is a formal document and should use all formal conventions such as:

- A title page with the proposed system name, the client's name and the author's name. It should have the date and any other relevant information regarding the project.
- Each page should have a footer or header with a page number and the author's name and the year the document was produced.
- An index on the second page that lists each section clearly and the related page number.
- Each diagram or table must be labeled with a title and a figure number. Each of these figures must be referred to in the text of the report. Avoid adding appendix items at the end of the report as an afterthought. If you add a diagram, discuss it in the text.
- The SRS must use study-specific language and use a formal tone throughout.

The SRS Structure

To assist your teacher in marking your work, ensure you have two clear sections: Analysis and Interpretation.

Part A: The Analysis

A.1 Introduction

- Identify the Problem the software will solve.
- Identify the purpose of the software solution you are investigating.

A.2 Intended Audience

- Describe the end users and other stakeholders in terms of relevant characteristics such as age, experience, attitudes and background.
- Identify the main end users and their needs.
- Identify any occasional and low-end users and their needs.
- Refer to the needs of the users in terms of the results collected.

A.3 Overall Description

- Describe the results of data collection. Refer to all the collection tools and the results, which should be included as figures or in the appendix.
- Ensure the data is the result of a wide range of collection techniques and methods (interviews, observations, reports and surveys).
- Describe the relationships between the data, users and the digital systems.
- Refer to analytical tools: Context Diagram, Use Case, Data Flow Diagram.
- Identify the Use Cases from the diagram in a table so they can be referred to in the Functional Requirements.

Part B: The Interpretation of Analysis

B.1 Operating Environment

- Accurately describe all relevant aspects of the technical environment (hardware, software, networking capabilities)
- Technical aspects of equipment such as memory, speed, operating systems.

B.2 Scope

Describe the scope in relation to the user needs:

- What will be included in the final software prototype.
- What will not be included in the prototype.

B.3 Constraints

- Describe the constraints to the development of the project in terms of skills, time, financial, technical, legal and social considerations.

B.4 Functional Requirements

- Ensure each requirement is allocated a key such as FR1 so it can be linked to evaluation criteria.
- Make a detailed description of each requirement. Ensure you include each input and output as well as the processes and storage of data.
- Add any notes to describe the requirement further. See the example below.

No	Requirement	Notes
FR01	Interface allows the user to select from all products available.	This may need to be organised into categories so as to limit the number of selections on the screen.
FR02	Allows for quantities to be entered for each selected product.	
FR03	Customers will be provided with a statement of their order details along with an order number.	This could be emailed or texted as well as displayed on the screen.

B.5 Non-Functional Requirements

- Ensure each requirement is allocated a key, such as NFR1, so it can be linked to evaluation criteria.
- Make a detailed description of each requirement. Ensure you relate each requirement to the factors that affect efficiency and effectiveness. Don't forget safety and security requirements.
- Add any notes to describe the requirement further. See the example below.

No	Requirement	Notes
NFR01	Interface needs to appear attractive and professional.	Business logo colours (Blue, Green, Gold).
NFR02	Robust interface - validation on personal contact data input.	• Email mask on email input text box. • Data type check on DOB or use of Date Time Picker. • Range Check on Post Code. • Existence checking on all inputs in Personal Contact Details input. • Provide feedback to user if data is incorrect or missing.

U3O2 Design Folio (SAT Criterion 4)

Assessment: 10 Marks
Time Allocation: Approx. 200 Minutes

The Design Folio document is a formal document and should use all formal conventions such as:

- A title page with the proposed system name, the client's name and the author's name. It should have the date and any other relevant information regarding the project.
- Each page should have a footer or header with a page number and the author's name and the year the document was produced.
- An index on the second page that lists each section clearly and the related page number
- Each diagram or table must be labelled with a title and a figure number. Each of these figures must be referred to in the text of the report. Avoid adding appendix items at the end of the report as an afterthought. If you add a diagram, discuss it in the text.
- The folio must use study specific language and use a formal tone throughout.

The Design Folio Structure

Part A: Introduction

Provide an overview of the folio and describe the basis for some of the decisions made about how the project will look and function. For example:

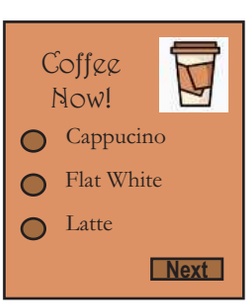
> **"In this folio the design process for the "Coffee Now!" ordering application is described and illustrated. The "Coffee Now!" business logo has been used to provide the basis for decisions on colour and fonts for the look and feel of the interface. Client, Brian Cox, identified some applications that he liked that could form a foundation for the design of the proposed software. Using the feedback collected from the client and other end users, we identified some key design elements that will inform the final design."**

Part B: Design Process

Describe the process used to generate ideas. Include the design tools used to generate ideas such as an Ishikawa diagram, Venn diagram, brainstorming notes, storyboarding, attribute listing or prompts. Below are two examples, the first is the use of a prompt and the second is an example of attribute listing.

"The Coffee Now! logo was used as a prompt to inform decisions about fonts and colours.

Colours:
Brown #994355
Tan # 5454CB
Font: Harrington"



Design 1

Design 2

Design 1 Attributes	Design 2 Attributes
Easy to read products	A graphically appealing solution
Next button easy to use	Simple first instruction
Colour is a bit flat	Cool Dark design

Part C: User Interface Design Ideas
This section must include two or three interface designs. They must be distinctively different from each other, yet all of them must be feasible. You cannot supply one good design and one poor design. Illustrations for each screen must be included for each of the designs. They must be annotated identifying all usability and graphic features. Relate your designs to factors that influence the success of the design: affordance, interoperability, marketability, security and usability.

Part D: Design Tools
Describe your two or three design ideas in terms of efficiency and effectiveness and how each design will meet the Functional and Non-Functional requirements. Refer to your design tool diagrams in the text, regardless of whether they are in the appendix or included as figures. For each of the designs you must provide:

- Object Description Table
- Data Dictionary
- Pseudocode

Part E: Selected Solution
Identify the design that will be used. Justify your decision based on function and usability.

Part F: Development Model Approaches
Identify the development model you think would best suit the project; agile, spiral or waterfall. Justify your decision.

Part G: Evaluation Criteria
The evaluation criteria are directly related to the Functional and Non-Functional Requirements. Evaluation criteria must be expressed in a way that will easily evaluate whether the client's needs are met with a completed design.

No	Requirement	Strategies	Effectiveness & Efficiency
EC01	(FR01) Interface allows a choice selection from all products available. Does the interface allow the full selection of available products?	Client to check if all selections have been made available in the interface. During Desk Checking and Final System Checking all sections are included, correct and available for ordering.	Effectiveness: • Are the selections easy to access? • Easy to read? Efficiency: • Is there a real-time feedback to show the user has made a selection?
EC09	(NFR01) Easy to use on a mobile phone. Is the scaling of the application allowing ease of use on mobile phones?	Observe users interact with application to see if they are having difficulties. Interview users or client about their experience using the application.	Effectiveness: • Correct use of the application is apparent? • Correct input is easy to add? Efficiency: • Quick to learn how to use?

U4O1 Evaluation Report (SAT Criteria 7 & 8)

Assessment: 20 Marks
Time Allocation: Approx. 250 Minutes

The evaluation report is the outcome of both testing and evaluation. When you are testing the solution you use Test Data, Testing Tables and Trace Tables. In evaluation you use a usability test comprising the strategies outlined in the evaluation criteria at the end of the Design Folio.

The Evaluation Report document is a formal document and should use all formal conventions such as:

- A title page with the proposed system name, the client's name and the author's name. It should have the date and any other relevant information regarding the project.
- Each page should have a footer or header with a page number and the author's name and the year the document was produced.
- An index on the second page that lists each section clearly and the related page number.
- Each diagram or table must be labelled with a title and a figure number. Each of these figures must be referred to in the text of the report. Avoid adding appendix items at the end of the report as an after-thought. If you add a diagram, discuss it in the text.
- The report must use study-specific language and use a formal tone throughout.

Evaluation Report

Introduction: Provide a brief overview of the success of the final solution produced.

Part A: Development Testing
- List Functional Requirements Testing Details
- Identify how the logic errors were corrected.
- Include trace tables that show how you fixed logic errors.
- Show test tables with test data, signed off against each feature.

Part B: Usability Test Methods and Results
- Functional & Non-Functional Requirements and their Evaluation Criteria from the SRS
- Identify all strategies and techniques that make up the Usability Test and how they correspond to each Evaluation Criterion.
- Include the data collection tools.
- All results from the Usability Tests.

Part C: Modifications
- Identify the reasons why all requirements are not completed.
- Identify all modifications required to the prototype to complete it.

Part D: Deliverable Evaluation
- Explain in terms of efficiency and effectiveness how specific features of the solution meet all functional and non-functional requirements. You can use the tables from your SRS to complete this section.

Part E: The Project Plan
- Write a short summary of how your Gantt Chart served your project management and identify any changes that were required. Refer to your adjusted Gantt Chart and Journal to support your summary. Explain the importance of relevant factors that influence the effectiveness of the project plan. Report on usefulness of the initial plan and how it allowed the project to be completed on time.

U4O2 Cybersecurity Report (SAC)

Assessment: 100 Marks
Weighting: 10%
Time Allocation: Approx. 100 Minutes

Students will be given a case study to examine in order to identify;
- the current software development security strategies,
- the risks and consequences of ineffective strategies,
- a risk management plan to improve current security practices.

Part A: Introduction
- Outline the goals of the organisation and information system in the case study.
- Outline the objectives of the digital system being used.
- Create a summary of the current security practices and refer to a network diagram.

Part B: Risks to Data Integrity in Case Study
- Identify the vulnerabilities in the current system.
- Identify the threats that could potentially undermine data integrity or otherwise compromise the system.
- Identify the consequences if the vulnerabilities were exploited.

Part C: Legal Obligations
- Based on the risks identified in Part B, identify all legislation with which the organisation is obliged to comply.
- Summarise how the legislation must be followed.

Part D: Recommended Risk Management Strategy
Choose a Risk Management Model to apply and identify the following:
- Create a network diagram with new security in place
- Physical Controls
- Software Controls
- Tracing Transactions and Detection Systems
- Back-Up Plans
- Response and Recovery
- Human Resource Management
- Ongoing Threat and Vulnerability Management.

Example Cybersecurity Case Study: Salt Spray Adventures

Salt Spray Adventures is a small business in Byron Bay. They take tourists for kayaking tours across the bay out to Julien Island. The tours include swimming with dolphins, paddling across the bay and picnicking at the lighthouse. They have an office on Main Street, Byron Bay where the two administrators take walk-ins, calls, and emails for tour bookings from their website. The six tour guides work out of the office and the boat house at the beach a few kilometres down the road. The guides have waterproof tablets where they collect the tour information (numbers of tourists and tour times). The guides use these to ensure they have the right people, to check they have paid and if customers have any medical conditions and to take attendance throughout the tours to ensure everyone's safety. Jarrah is a pretty laid back business owner and has set up the network and website himself. He likes the idea that tourists can come in and use the computers in the office to check emails and have a coffee. The relaxed and friendly atmosphere encourages more tourists to come in and pay for tours. The office is open plan. There are two desktop computers that the administration staff use to manage the business. All staff share the same password to access these machines. These are located behind the

counter where the printer-photocopier is placed. Around the office are lounge chairs and desks with four desktop computers for customers to use. Customers can also access the Wi-Fi. Many people know that Salt Spray is a friendly place and go there to use the Wi-Fi and print out documents, such as copies of passports and resumes.

The tour guides spend a lot of time in the boat house taking tours. Sometimes they go to the office to check on bookings but Jarrah wants them to rely on using their tablets for updates on tour booking details. It was suggested that the tablets have passwords, but due to the nature of the work environment, the tour guides don't want to be fiddling with wet hands on the screen to get the information they need in a hurry. The boat house, like the office, is a relaxed affair with computers for customers to use and free Wi-Fi.

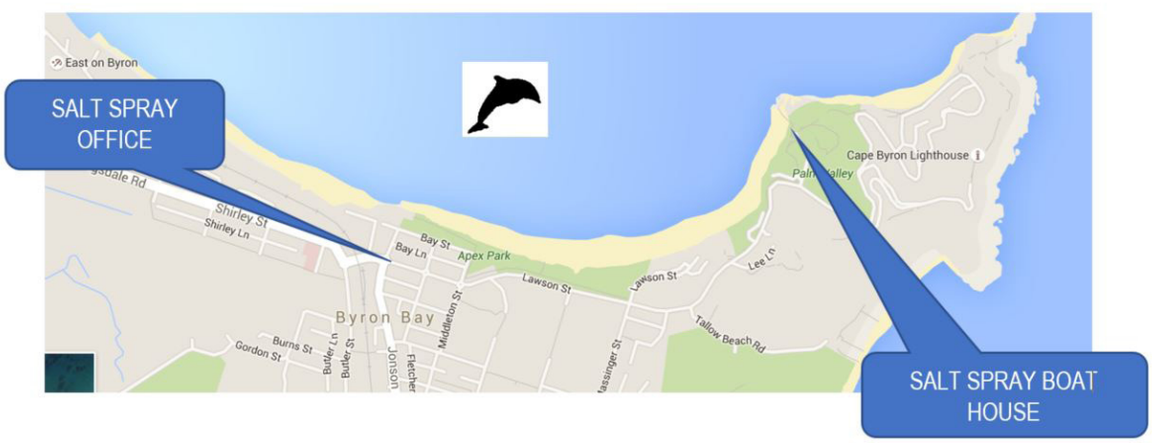
When a customer wants to book a tour, they can access saltsprayadventures.com.au to choose a time and a date. They must enter their name, current address in Byron Bay, permanent home address, mobile phone number, any medical conditions, and payment details. The data is entered into a simple HTML form and stored in a database. Many of the customers are young backpackers from around the world who are also looking to meet up with other tourists to have a good time.

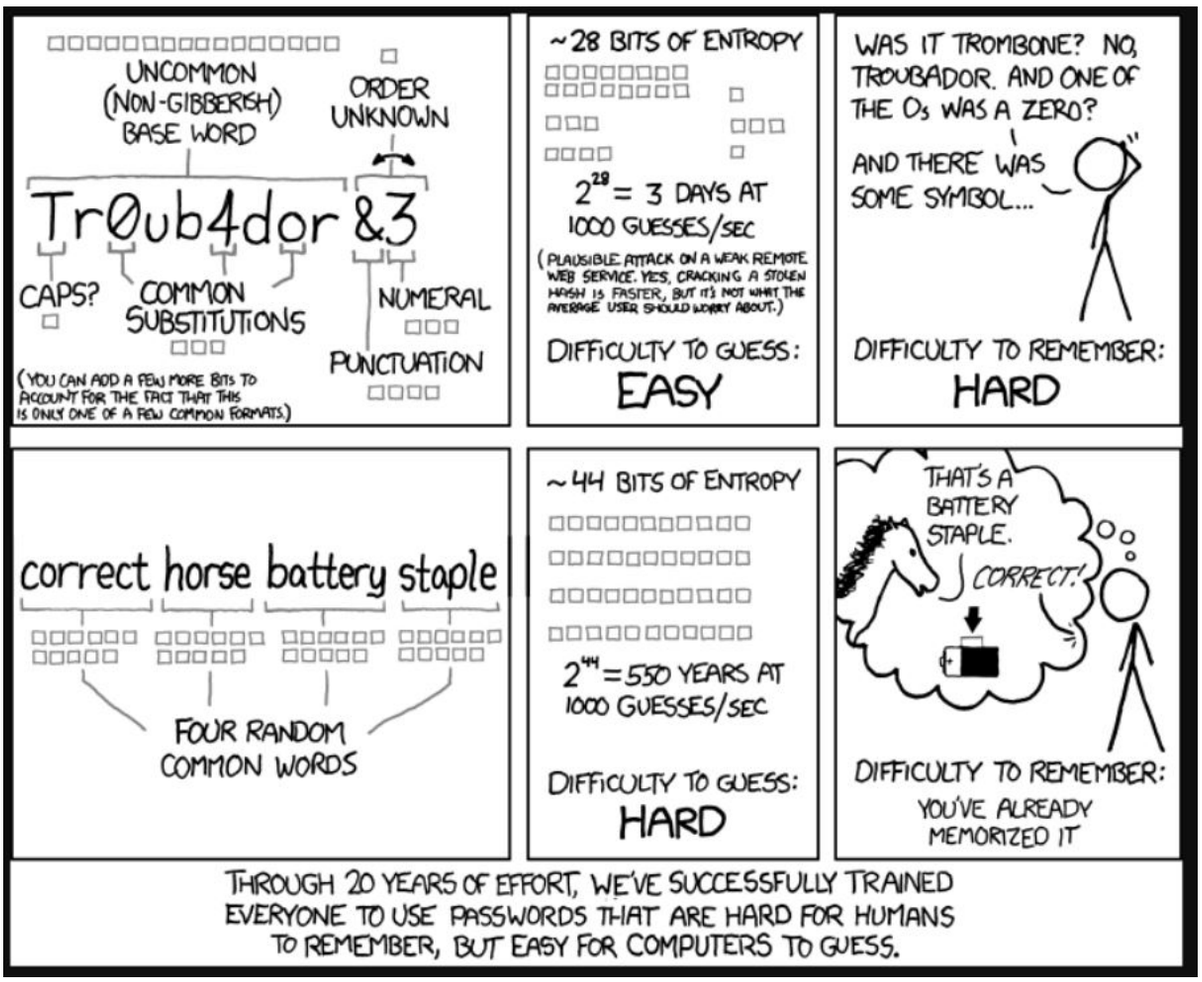
Jarrah is using the free firewall in Windows 10, and free antivirus software to cut costs. Both the office and boat house customer-access computers and Wi-Fi require one password "SaltSpray".

Jarrah's accountant, Sue came into the office one day to meet with him regarding his tax return for the business, when she saw a customer using one of the two staff computers to print out a flyer. Sue told Jarrah that looked like a security risk. Sue suggested he get a risk management assessment done on the security of his information system because he may not be complying with relevant legislation.

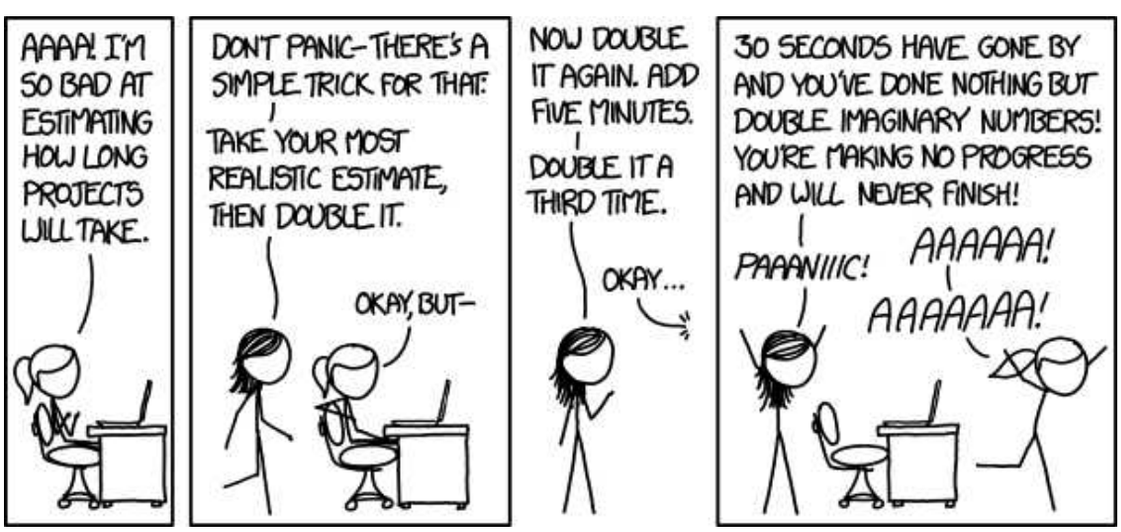
Jarrah has employed you to investigate the Salt Spray Adventures information system. Write a cybersecurity report based on the information provided.

(Despite Byron Bay being in NSW, please consider all Victorian legislation in your answer.)





https://xkcd.com/936/



https://xkcd.com/1658/

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

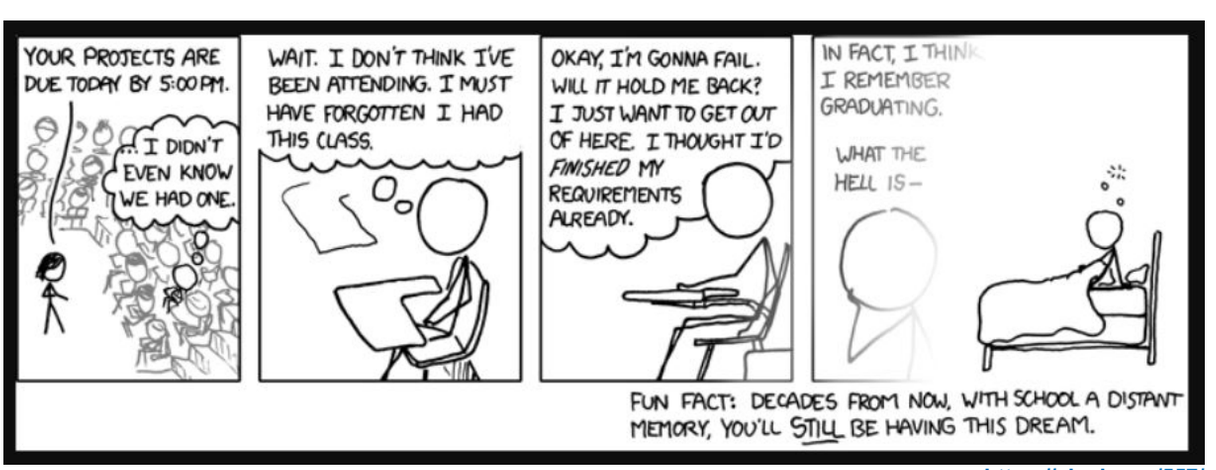
U

V

W

X

Index of VB Examples



https://xkcd.com/557/

A
Vic Farrell
Publication
2022

vicfarrell.com.au

ISBN 978-0-6486708-1-0