CHAPTER 1: ERGONOMICS

The term ergonomics refers to the study of the relationship between people and their work environment.

ERGONOMICS

- Effects of prolonged use of software:
  - Repetitive Strain Injury (RSI)

- Procedures to prevent and minimize injury:
  - Lighting (incorrect lighting may cause headaches).
  - Indoor Climate (if the climate is incorrect it can cause sleepiness etc.)
  - Noise (Excessive noise makes it hard to concentrate).

- Work Routine (The way the job is done. Directly effects performance.)
  - Job Design
  - Workload
  - Work Pressure

- Touch Typing (Gives the advantage of being able to type and keep eyes on the document.)

- Ergonomically Placed Furniture.
  - Desk
  - Chair

- Ergonomically Placed Equipment
  - Screen
  - Keyboard
  - Mouse

- Ergonomic issues relating to software
  - GUI rather than remembering commands
  - Acceptable response times in software
Preliminary Coarse Summary

○ 'User friendly'
  ■ Learnability
  ■ Flexibility
  ■ Robustness
○ Appropriate messages to the user
○ Consistency

INTELECTUAL PROPERTY

● Software license agreements
  ○ License terminology
  ○ Legal aspects
  ○ Use of software covered by a software agreement
● Origins of software design ideas
  ○ Scientific and military in mind.
  ○ Evolution of existing concepts
    ■ Xerox and the original GUI
  ○ New and existing approaches
    ■ Visicalc and spreadsheets
● Sources of code and conditions that apply
  ○ Internet, books, magazines, freeware and shareware.
  ○ Publication of code does not give a person the right to use it.

INCLUSIVITY

● The need for software design and development to be inclusive
  ○ Cultural
  ○ Economical
  ○ Social
  ○ Gender
  ○ Disability

CHAPTER 2: HARDWARE AND SOFTWARE

HARDWARE

● The function of hardware within the computer system
  ○ Input
  ○ Processing
  ○ Control
  ○ Storage
  ○ Output

● Peripheral Devices (devices other than the CPU, often attached to the computer, include: input, output and storage devices.)
● Input Devices
Preliminary Coarse Summary

- Keyboards
- Pointing Devices (mouse)
- Character Readers (barcode wand)
- Scanners
- Digital Cameras
- Video Input (webcam)
- Sound Devices (MIDI)

**Output devices**
- Visual Display Units (monitor)
- Hard Copy Devices (printer)
- Speakers.

**Process and Control**
- CPU is responsible for processing and control.
- CPU is made up of the control unit, the arithmetic logic control unit and registers.

```
Control unit
CONTROL

Arithmetic
logic unit (ALU)

PROCESS

Registers
STORAGE
```

- Control Unit co-ordinates the operations of input, processing, output and storage.
- Arithmetic Control Unit (ALU) is the part of the CPU that does all logic and mathematical calculations.
- Registers are temporary storage areas for small amounts of data or instructions needed for processing.

**Storage**
- Primary storage stores data and programs before and after they have been processed by the CPU.
- Main types of primary storage includes:
  - read-only memory (ROM)
  - random-access memory (RAM)
- Secondary Storage
  - More permanent than RAM (hard-disk drive, or floppy disk)
    - Magnetic tape, magnetic disks, floppy disks, hard-disks, CD-ROM (compact disk-read only memory) CD-R.

**SOFTWARE**

Software can be classified as system software or application software. System software provides the algorithms for the computer to manage its resources and to communicate with peripherals. Application software is responsible for the instructions which allow the computer to perform certain
Preliminary Coarse Summary

- System Software: (Two groups).
  - Operating Software
    - Provides the computer with programs that allow it to communicate with the outside world and manage its resources.
    - Running when the computer boots up until it is shut down.
    - Provides link between application and the input and output devices.
  - Utility Software
    - Allows the user to perform simple tasks such as formatting disks, deleting files and searching through files.
- Application Software
  - Responsible for the computer to be able to perform certain tasks.
  - Falls into two categories:
    - Mass-produced programs
      - Designed for general use. IE, word processors, spreadsheets.
    - Custom software
      - Developed for a single customer.
      - Specialized for the task the customer requires. IE. An airline's booking system.
- Generations of programming languages
  - Low-level languages (processor dependent).
    - First-generation languages
      - Directly able to be understood by computers (binary).
    - Second-generation languages
      - Replaced binary with mnemonic codes (or short code words) to represent instructions.
  - High-level languages (processor-independent).
    - Third-generation languages
      - Processor-independent; uses variables with sequences, branches and loops.
    - Fourth-generation languages
      - Processor-independent; uses form filling, computer-aided graphics and screen instructions.
    - Fifth-generation languages
      - Processor-dependent; uses artificial intelligence techniques.
- Event-driven versus sequential approach
  - Sequential approach follows the recipe formula
  - Event-driven programming caters to different happenings.
- The need for translation
  - The instructions contained in high-level languages (second-generation and
Preliminary Coarse Summary

above) cannot be directly understood by the processor. The instructions have to be converted from human-friendly code (source code) into machine-readable form (object code).

- Compilation is the process by which the source code is converted into object code and stored to be executed.
- Interpretation is where the source code is converted into object code and executed immediately.
- In incremental interpretation sections of the code which are used repeatedly are translated and stored for use by the interpreter. This means that the interpreter does not need to repeat the translation process many times.

**Characteristics of different operating systems**
- Command-based interface relies on text to communicate with the computer.
- Graphical User Interface (GUI) uses windows, icons, a mouse and pointers. Also referred to as a WIMP interface. The GUI also displays documents in a form which closely resembles their final form, this is known as WYSIWYG.

**The Relationship Between Hardware and Software**

- **Processing of software instructions by hardware: the 'fetch-execute' cycle**
  - The fetch execute cycle can be divided up into nine steps:
    1. fetching the instruction from primary storage
    2. decoding the instruction into an operation code and data address
    3. copying the operation code into the instruction register
    4. copying the addresses of the data into the address register
    5. using the address register to copy the data into the storage register
    6. sending the operation code and data to the ALU
    7. executing the instruction on the data
    8. sending the result to the accumulator, ready for the next instruction
    9. storing the results in primary storage.

- **The initiation and running of an application**
  - The operating system will load the program into main memory, when it is loaded the OS passes control to the now-loaded program. The program will more than likely take input from the user and then perform the task.

- **The existence of minimum hardware requirements to run some software**
  - When purchasing software you are usually provided with minimum hardware requirements. These are the specifications that need to be met by the computer that is to run the application so it can be run effectively and to its full potential. If the computer does not meet minimum hardware requirements chances are that the program will not run.

- **Elements of a computer system**
  - Hardware – physical devices needed to perform the required job.
  - Software - applications
  - Data – facts that are manipulated by the computer in order to present information
  - Procedures – tasks performed by, and rules put in place for the users of the computer system.
Preliminary Coarse Summary

- Personnel – people involved with the computer system. These include:
  - direct users
  - indirect users
  - computer operators
  - data entry operators
  - information systems managers
  - maintainers – (computer technicians, technical support staff, training specialists, computer consultants.)
  - developers – (system analysts, computer programmers, hardware engineers.)
- Developing software
  - When software is being developed each of the system elements has its part to play.

CHAPTER 3: SOFTWARE DEVELOPMENT APPROACHES

THE STRUCTURED APPROACH TO SOFTWARE SOLUTIONS

Can be divided into five stages – defining the problem, followed by planning, building, checking and modifying the solution. These steps form the software life cycle.

- Defining the problem
  - Manager of the business and systems analyst discuss the problems faced with the current system. From these discussions, comes a document that clearly states the requirements of the new system.
- Planning the solution
  - This phase has to identify the needs of the user. Processes and procedures of the existing system are observed in order to determine how the inputs are going to be converted into outputs. This is conducted through tools such as interviews, questionnaires and observations.
- Building the solution
  - The next step in the method of software development is to hand over specifications to the design team. They will break the required processing up into smaller pieces called modules to ensure that those responsible for the design of the processing steps have a small, understandable process to work on. Each module is then turned into a set of steps, known as an algorithm. The algorithms are then passes on to the programmer for coding.
- Checking the solution
  - The checking procedure is designed to detect errors. This makes the task of finding errors in the later stages much easier. Once this is completed, the program is ready for the customer.
- Modifying the solution
  - When the new system is in place, continual evaluation by users takes place to determine whether it meets the requirements.
- Usefulness of structured approach
  - Useful for large and/or complex applications.
  - Efficient
  - good for teams of developers.

THE PROTOTYPING APPROACH TO SOFTWARE SOLUTIONS

A prototype is a working model that is usually used to gain information about how the elements of the system work together.

- Prototyping as an information-gathering tool
  - When a prototype is used as an information-gathering tool, it is often at the beginning of
Preliminary Coarse Summary

the structured approach to software development. Often used when the user is unsure what is required for the application. Little regard is paid to error checking. The prototype is passed to the user for evaluation under working conditions. Suggestions are made by the user for improvements, which are then implemented and sent back to the user for further evaluation. When the user is satisfied, the prototype is incorporated into the requirements of the new system.

- Prototyping as a software development approach
  - Not suitable for large-scale projects.
  - 'Evolutionary' prototyping will generally produce software in shorter time and at lower costs to the client.

RAPID APPLICATION DEVELOPMENT (RAD)

- Software engineering process design that leads to faster application development. A number of different approaches may be used:
  - CASE (computer-aided software design)
  - reuse of code
  - use of templates.
- When RAD approach is taken, user is often directly involved with the programmer.

END-USER DEVELOPMENT

- The emphasis on this type of development is on rapidly arriving at the solution. There are no formal stages, just the user taking advantage of the application's capabilities and customising applications to their own situations.
- Produced at a fraction of the cost (no outside involvement.)
CHAPTER 4: DEFINING THE PROBLEM AND PLANNING SOFTWARE SOLUTIONS

DEFINING THE PROBLEM
There are three main steps in producing a solution to a problem. They are:
1. Understand the problem
2. Work out a way to solve the problem
3. Check the solution of the problem

IPO chart
- IPO (input, processing, output) chart is used to understand the process and data interactions within the system.
- First the outputs that are required are taken into account, we need to know what it is that we want the system to do before we look for the necessary inputs and processes.
- The inputs that are needed for the desired outputs are looked at.
- Deciding on the process that we need to convert the input into the desired output is needed.

An example of an IPO chart:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>PROCESS</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimensions of area to be tiled</td>
<td>Calculate area.</td>
<td>Total cost of job</td>
</tr>
<tr>
<td>Tile type</td>
<td>Calculate cost of the tiles.</td>
<td>Cost of materials</td>
</tr>
<tr>
<td>Cost of tiles per square metre</td>
<td>Calculate cost of the other materials.</td>
<td>Cost of labour</td>
</tr>
<tr>
<td>Amount of other materials (groat and adhesive) needed per square metre</td>
<td>Calculate cost of all materials by adding the cost of the tiles to the cost of the other materials.</td>
<td></td>
</tr>
<tr>
<td>Unit costs of other materials</td>
<td>Calculate time required by the tiler.</td>
<td></td>
</tr>
<tr>
<td>Area of tiles that can be laid in an hour</td>
<td>Calculate cost of the tiler’s labour.</td>
<td></td>
</tr>
<tr>
<td>Tiler’s hourly rate of pay</td>
<td>Calculate total cost of the job by adding the cost of all materials and the cost of the labour.</td>
<td></td>
</tr>
<tr>
<td>Output total cost of the job, cost of materials and the cost of the labour.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.5 Stage 3, the processes required to achieve the outputs are included in the table.

ABSTRACTION/REFINEMENT
- Top-down design
  - In this approach, large complicated problems are broken down into a series of smaller, easier-to-solve problems. The solution to one of these problems is called a module.
  - The basic concept of the process is the decomposition of a solution into smaller and smaller units until each unit can be expressed as one instruction. An alternative name of this is stepwise refinement.
Modification of an existing solution

- When modifying a solution it is sensible to use the parts of the program that are performing well instead of starting again from scratch. This may include adding a function to a program, which is a perfect example of how writing a completely new system and throwing out the old one is ridiculous.

DATA REPRESENTATION

- Digital computers use the binary number system to represent data items and instructions. The binary number system consists of only two digits, 1 and 0.
- Conversions to and from the binary number system

  - The steps for converting numbers to binary are:
    1. Write down your chosen number and divide it by two.
    2. Repeatedly divide the answer by two until you reach 0, each time writing down the answer and writing the remainder in the R column. (If there is no remainder, write 0).
    3. For example:

```
   2) 103   R
      2) 51    1
      2) 25    1
      2) 12    1
      2) 6     0
      2) 3     0
      2) 1     1
      2) 0     1
```

    4. The digits in the remainder column are then read from the bottom to the top to give the answer. This means that 103 is equal to 1100111.

  - To convert binary into base ten, the following procedure is used:
    1. Think of the digits with the powers of two above them like this:

```
  2^6  2^5  2^4  2^3  2^2  2^1  2^0
  1    1    0    0    1    1    1
```
    2. Write the decimal values for each of the powers of 2.
3. Add up the numbers with 1s below them and ignore those with 0s below.

\[
\begin{array}{cccccccc}
64 & 32 & 16 & 8 & 4 & 2 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

\[
1100111 = 64 + 32 + 4 + 2 + 1
\]
\[
= 103
\]

Conversion to and from the octal number system

- Octal number system consists of eight numbers 0, 1, 2, 3, 4, 5, 6, and 7.
- Convert decimal number to binary (much easier to convert from binary than base ten).
  1. Group the digits in threes from the right-hand side of the binary number, adding zero's if necessary to create the last group of three:
     \[
     1100111 \text{ becomes } 001 \ 100 \ 111
     \]
  2. Use the equivalents already known for the digits 0 to 7 known in binary, replace each group of three with the appropriate digit. In the example, 001 represents 1, 100 represents 4 and 111 represents 7, so our octal equivalent of 103 is 147.
- To convert octal numbers back to decimal numbers, we follow similar steps to the conversion from binary to decimal:
  1. Write the number with the power of eight above each digit
     \[
     \begin{array}{ccc}
     8^2 & 8^1 & 8^0 \\
     1 & 4 & 7 \\
     \end{array}
     \]
  2. Change the powers of eight to their decimal equivalents.
     \[
     \begin{array}{ccc}
     64 & 8 & 1 \\
     1 & 4 & 7 \\
     \end{array}
     \]
  3. Add up the values.
     \[
     1 \times 64 + 4 \times 8 + 7 \times 1 = 64 + 32 = 7
     \]
     \[
     = 103_{10}
     \]

Conversion to and from the hexadecimal system

- The hexadecimal system is based on 16 numbers.
- The conversion is very similar to octal conversions. Once the value is in binary, the digits are grouped into sets of four from the right-hand side and then the bit patterns are changed to their hexadecimal equivalents.
- To convert 103 to hexadecimal, first convert to binary. The bit pattern 1100111 is now written as two groups of four, 0110 and 0111. 0110 = 6 and 0111 = 7, therefore 103 = 67.
- To convert the hexadecimal number back to base ten, we look at the number. With 67, the right hand side digit represents the single units and the next digit to the left stands for 16s. Therefore 67(base 16) = 6 * 16 + 7 * 1 or 103(base ten)

DATA TYPES

- Data often needs to be represented on paper so that we can understand what needs to be done to process it. These structures are called abstract data types and are beneficial to the
programmer. Abstract data structures can be classified as simple data types or structured data types.

- **Simple data types** include:
  - **Boolean data type** - the boolean data type is the simplest data type available and can represent one of only two possible values: true and false.
  - **Character data type** – the character data type represents the smallest item of data that an individual normally uses. The most common code used to represent characters is the ASCII code.
  - **Integer data type** – stores numerical data types. With an integer you can store a number between -32768 and 32768.
  - **Floating point data type** – used to represent numbers that lie between integers (fractions)
  - **String data type** – is used to represent a sequence of characters, such as word, but keeps its identity as a single data element.
  - **Date and currency formats** – special purpose types.
- **Structured data types**
  - **Arrays** – numbers of related data items of the same type stored together
  - **Record** – numbers of related data items of different types stored together.
  - **File** – numbers of records stored together.

**Structured algorithms**

- The algorithm – an algorithm is a finite set of steps, with a single beginning and a single end, which solves a problem.
- Methods of algorithm description – algorithms are usually represented as pseudocode and flowcharts.
- Pseudocode is one method of algorithm description involving the use of the English language. It relies on indentation and using keywords to highlight the structure of the algorithm. It is very popular because it is in a text form that can be easily modified using a word processor. An example of pseudocode that shows the steps required to make a cup of coffee.

```
BEGIN make_coffee
    Fill a kettle with water.
    Boil the water in the kettle.
    Add coffee to the cup.
    Pour boiling water into the cup.
END make_coffee
```

- Flowcharts – a flowchart is a pictorial or graphical method of describing an algorithm. Flowcharts are often favoured because it is easier to follow the structure in a picture than in words. The basic elements of a flowchart are a set of symbols that contain messages, and interconnecting lines with arrows. The four most commonly used symbols are:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Symbol 1" /></td>
<td>Terminal: start and finish. There should be only one line in, or one line out.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol 2" /></td>
<td>Process: description of a process or action. It has one line in and one line out.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol 3" /></td>
<td>Subprogram: process described by its own flowchart to perform a particular task.</td>
</tr>
<tr>
<td><img src="image" alt="Symbol 4" /></td>
<td>Decision: it has one line coming in at the top, and two lines leaving it.</td>
</tr>
</tbody>
</table>

*Table 4.5* Some of the common symbols used in flowcharts.
Standard Constructs – Algorithms are composed of standard constructs or control structures: a sequence of instructions, a selection between two or more alternative courses of action, and a repetition of a set of instructions number of times (an iteration).

- Sequence – The simplest of all the constructs, is a set of instructions following one after the other.
- Selection – The process of decision making is known as a selection; a particular course of action is followed according to the data at a particular time and a set of selection rules. It is used to determine which step should be executed next. There are two types of selection: binary selection and multiway selection.
  - Binary – The simplest selection structure will only allow two courses of action. The most common form of binary selection is in the form of an 'IF condition THEN action ELSE alternative action'.
  - Multiway – Where a number of branches are needed to make the point of the program, using a number of binary selections becomes inefficient and complicated. In order to classify a data element in one decision and take the appropriate action, a multiway selection is used. Example of a multiway selection using pseudocode:

```
BEGIN MAINPROGRAM
  CASEWHERE signal is
    Red: stop the vehicle.
    Amber: stop the vehicle.
    Green: pass through traffic lights.
    OTHERWISE: proceed with caution.
  ENDCASE
END MAINPROGRAM
```

- Iteration (repetition or loop) – the steps to be repeated are referred to as a the body of the loop. It is very important that each loop contains a condition to stop the loop going on forever. The condition can be checked, or tested, at the beginning or end of the loop, being respectively known as a pre-test loop or a post-test loop. A special construct known as a counted loop is often used when a loop has to be executed a pre-determined number of times.
  - Pre-test repetition
    - In a pre-test repetition or guarded loop, the condition is checked at the beginning of the loop before the steps to be repeated are executed. For example:

```
BEGIN
  WHILE car is travelling
    Keep seat belt on
  ENDWHILE
END
```

- Post-test repetition
Preliminary Coarse Summary

- In a post-test repetition the condition is checked at the end of the loop after the steps to be repeated are executed. For example:

```
BEGIN
  REPEAT
    water plants
  UNTIL ground is soaked
END
```

- Counted loop
  - For when we need to execute a set of steps a predetermined number of times. For Example:

```
BEGIN
  set end to 10
  FOR counter goes from one to end
    print counter
  NEXT counter
END
```

CHECKING ALGORITHMS

- Desk Checking
  - Desk checking involves working through the algorithm, keeping track of the values of the variables and the outputs of the algorithm as the steps are manually gone through with sample data items. The data items are known as test data. When the algorithm is checked, a table is used known as an input/output table. An example of one for the following algorithm is below:

```
BEGIN
  set end_number to user input
  set counter to 0
  set sum to 0
  WHILE counter < 5
    set counter to counter + 1
    set sum to sum + counter
  ENDWHILE
  display counter, sum
END
```

<table>
<thead>
<tr>
<th>Variable</th>
<th>Values</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>end_number</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>counter</td>
<td>1, 2, 3, 4, 5</td>
<td>15</td>
</tr>
<tr>
<td>sum</td>
<td>1, 2, 3, 4, 5</td>
<td>15</td>
</tr>
</tbody>
</table>

*Figure 4.40* The desk check table after the check has been performed.

- Standard algorithms
Preliminary Coarse Summary

- There are some common processes which appear in a wide range of problems. Rather than rewriting the algorithm each time it is required, a standard one can be used. This saves both time and effort.

- **Swap two data items**
  - This algorithm has an input of two data items called `item1` and `item2`. At the conclusion of the algorithm, the values of the data items would have swapped.

```plaintext
BEGIN swap items
    set temporary to item1
    set item1 to item2
    set item2 to temporary
END swap items
```

CHAPTER 5: BUILDING SOFTWARE SOLUTIONS

CODING IN AN APPROVED PROGRAMMING LANGUAGE

- **Metalanguages** – languages used to describe the rules of another language.
  - Examples of metalanguages are BNF (Backus-Naur Form)/EBNF (Extended Backus-Naur Form) and syntax structure diagrams (railroad diagrams)
  - Representation of syntax
    - the syntax of a language refers to the rules which govern the way that the elements of that language can be joined to form valid statements. Syntax can be joined to create statements, and statements can be combined to form compound statements.
    - Syntax structure diagrams (railroad diagrams) – graphic diagrams that show that trace a path from left to right, there are branches that shows the allowable features.
  - Backus-Naur Form and Extended Backus-Naur Form
    - BNF – is a text based system and as such cannot draw upon shapes to define of refer to elements. Several symbols are used in BNF to indicate the structures seen above.
    - EBNF – is also a text based system. It is much simpler than BNF as it does not hav the need for repetition of some definitions.
  - The syntax of control structures
    - Sequence – the coding of a sequence is basically the same in all languages. Each instruction is placed in a position that corresponds to its place in the algorithm.
    - Selection - When implemented in a language selections will have the same basic structure. The binary structure starts with the keyword `IF` and a multiway selection uses the keyword `CASEWHERE`, just as they do in pseudocode.
    - Iteration – the three types of repetition appear in most languages in a form that is not too different from its representation in the approved form of pseudocode.
  - Syntax of data type definitions
    - languages are structured in such a way that the storage required for variables is made available when needed. This is done in a variety of ways, the most common is declaring
Preliminary Coarse Summary

the variables.

ERROR-CORRECTION TECHNIQUES

● Types of coding errors
  ○ Syntax errors – syntax errors, or compile-time errors, are identified by the programmer as written statements that do not conform to the rules of the language. Common causes are:
    - missing or wrong statement punctuation
    - typographical errors in reserved words
    - failure to complete groupings
    - incomplete program statements
  ○ Run-time errors – a run-time error in a program may not be evident at the point at which it occurs, but it may surface later, causing all kinds of trouble in the program.
  ○ Logic errors – most logical errors should be identified during the algorithm description stage of the design process. This kind of error is much harder to rectify and involves using techniques such as setting breakpoints and tracing the flow of control.
  ○ Stubs – Small modules that represent parts of the program which have yet to be written are called stubs.
  ○ Debugging output statements – the process of error detection does not end with the correct working of the algorithm and error-free coding. It is often necessary to trace the values of variables as they change during the execution of a program in order to detect where the problem comes from.
  ○ Flags – A flag is a variable used to indicate or 'flag' whether an event has occurred or not.

LIBRARIES OF CODE

● Reusable code – when writing modules we should look at keeping them as independent from the main program as possible. In this was we can reuse the module with minimum effort. When a module has proven successful it should be added to the library.

USER INTERFACE DEVELOPMENT

- The user interface is the only means by which the user communicates with the application.

● Screen design
  ○ Designs need to be consistent. They need to have uniform commands and message placement.
  ○ The work needs to be legible so as the user can distinguish and interpret the elements on the screen.
  ○ Elements of a screen design should be chosen to fit best the action they are to perform. Elements include, radio buttons, choice boxes, dialog boxes, and icons.

DOCUMENTATION

● Types of documentation – documentation can be categorised as external or internal documentation.
Preliminary Coarse Summary

- Process documentation is used to assist with the maintenance of the system, includes algorithms, test data, and code.
- Product documentation is used to assist the user and system administrators with the operation of the system, and includes manuals, tutorials and online documentation.
- Internal documentation- is contained in the programs. It assists programmers in modifying the program for maintenance and further development. It includes intrinsic documentation and attached comments.

CHAPTER 6: CHECKING THE SOFTWARE SOLUTION

TEST DATA
A set of data known as test data has to be formulated to use for the testing process. They are chosen carefully to test all parts of the algorithm or program.

- **Requirements of test data**
  - a good set of test data should achieve the following three aims:
    1. to test all parts of the program
    2. to test each of the paths that can be taken during execution
    3. to test on each side of any boundary value as well as the value itself.
  - Both the algorithm and the coded solution should be tested for errors using the set of test data.

- **The test data dictionary**
  - a dictionary of data items is written to list all test data items, their expected outputs and the reasons for their inclusion in the set.

- **Testing both algorithms and coded solutions with test data**
  - The main aim of testing is to cause and discover errors, not to show that the program functions correctly.
  - Before beginning, it is good to devise a testing strategy.
  - Desk checking is a testing process in which the steps of the algorithm or program are followed manually, at the same time keeping track of the values of variables on paper.
  - White-box testing is a process of thoroughly testing a module, rigorously following each of the execution paths.

- **Error Types**
  - Arithmetic error – are those that occur in the processing of numerical data, the most common being divisions by zero and problems due to the order of calculation and truncation.
  - Comparison errors – occur in the attempt to compare two different types of data elements.
  - Control logic errors – occur within the selection and repetition structures of the program.
  - Data structure errors – include those in the initialisation of variables, test for proper indexing of arrays and consistency in the use of identifiers.
  - Input/output errors – include those associated with data validation, and opening and closing files as well as special cases such as empty records or end-of-data markers.

- **Bottom-up and top-down testing**
  - Bottom-up testing – testing from the lowest levels (simplest) of the module hierarchy upwards to the highest level (the main or control module).
  - Top-down testing – involves first testing the main or driver module, then working down the hierarchy to the lowest level of modules.
EVALUATION OF DESIGN
- consists of two processes – verification and validation

  ● Verification is the process in which the software is tested to see whether it performs its functions correctly
  ● Validation is the process where the software is evaluated against the original specifications to see whether it will perform the tasks specified in the problem definition.
  ● Methods of evaluation
    ○ Peer checking – one of the most effective ways of evaluating a solution is for a peer to check a design.
    ○ Structured walk-through – is a clearly defined process in which a number of peers take on particular roles. They are: the presenter (submits the product for review), coordinator (responsible for running the walk-through), scribe (responsible for keeping record of results), maintenance representative (concerned with ease of maintenance), standards leader (responsible for consistency of product and ensuring standards are met) and customer representative (ensures product meets needs of user).
    ○ Desk checking

EVALUATION OF IMPLEMENTED SOLUTIONS

  ● Checking the solution to see if it meets the original design specifications
    ○ It is imperative that the solution meets the specifications outlined in the original design.
  ● User feedback – evaluation given by a user can be more valuable than that from the development team. User feedback should be sought on the following:
    ○ the operation of the system
    ○ the ease of use and sustainability of the user interface
    ○ response times for various tasks
    ○ the overall 'feel' of the software application
    ○ ease of learning of the software
    ○ functions that may be improved to that have been omitted.
  ● Social and ethical perspectives – the software needs to be evaluated in terms of:
    ○ Copyright
    ○ Ergonomics
    ○ User Friendliness
    ○ Inclusivity

CHAPTER 7: MODIFYING SOFTWARE SOLUTIONS

REASONS FOR MAINTENANCE CODING
The maintenance of existing code is known as maintenance coding.

  ● Maintenance coding may be required for a number of reasons: to remove a bug, to improve efficiency, to change the way a program works, to allow the program to cope with changed data, to meet new requirements of the organisation or to comply with government regulations.
  ● Changing user requirements – as a user becomes more familiar and competent in using a computer program, shortcomings in its operation may become evident.
Preliminary Coarse Summary

- Upgrading the user interface – when software is originally developed, the user interface receives a lot of attention. It is often found though that the user interface could benefit from redesign.
- Changes in the data to be processed – changes will often lead to a need to modify the software.
- Introduction of new hardware or software – during the life of the software, technology is improving both the hardware and the OS software. Maintenance will be required for the software to run on new OS’s.
- Changing organisational focus – as the organisation evolves, its focus changes. As the focus changes the software must adapt with it.
- Changes in government requirements – it may be necessary to modify organisation's software in order to comply with new requirements.
- Poorly implemented code – maintenance is a good time to evaluate the code for its efficiency and properly overcome earlier problems rather than relying on a software patch.

SOCIAL AND ETHICAL IMPLICATIONS
Modification of software must be performed within legal and ethical bounds.

- Plagiarism – within the context of software development and modification, plagiarism includes the use of code within an application without acknowledging its source and the reverse engineering of code to determine the algorithm which is then recorded.

FEATURES IN SOURCE CODE THAT IMPROVES ITS MAINTAINABILITY

- Documentation within the code – One of the easiest ways to make sure that code can be easily modified and updated is employing internal documentation in the form of comments and choosing meaningful identifier names.
- Presentation of a coded solution – the manner in which it is presented can have a large bearing on the ease of maintenance. Factors include: the use of standard control structures, a clear and uncluttered mainline and the allocation of one logical task to each subroutine.

INTERPRETATION

- Interpretation of a program can be done with the aide of the documentation to the program. The documentation consists of product documentation such as user manuals and process documentation associated with the software such algorithm descriptions,

DOCUMENTATION
Documentation can tell us about the control structures used and the ways in which data has been processed. Using supplied documentation it is easy to identity the control structures and explain how variables have been used.

CHAPTER 8: DEVELOPING SOFTWARE SOLUTIONS

IMPLEMENTING PROJECTS
Steps when creating a program:

- Defining the problem
Preliminary Coarse Summary

- Understanding the problem
- Identification of inputs, processes and outputs
- Planning
- Identification of a suitable development approach
- Design of appropriate algorithms
- Determination of appropriate data structures
- Identification of relevant subroutines
- The design of test data and expected output
- The desk check of algorithms
- Identification of existing code that can be used
- Building the solution
- Implementation of the solution in an appropriate language
- Testing the solution using test data
- Documenting the solutions
- Checking the solutions
- Evaluation of the completed solution
- Modifying the solution
- Changing the solution to meet specifications

PROJECT MANAGEMENT TECHNIQUES

- Identification of tasks
- Techniques to assist project management
  - Gantt charts – time management chart (graphical tool)
  - Log books – tracks development of Project
- Allocation of resources
- Identification of major milestones and stumbling blocks
- Regular backup
- Response to difficulties
- Regular reporting
- Evaluation
- CASE tools

PROJECT DOCUMENTATION

- Algorithms
- Gantt Charts
- Manuals
- System documentation
- Data Dictionaries
- Diaries or logbooks

SOCIAL AND ETHICAL WORK ISSUES RELATED TO PROJECT WORK

- Ease of use
- Accessibility of technical language
- Copyright
- Ergonomics