SOFTWARE DEV
KEY KNOWLEDGE POINTS

U3O1

- stages of the problem-solving methodology
- key tasks associated with planning software projects, including identifying, scheduling and monitoring tasks, resources, people and time
- a brief overview of the concept of the OSI model for network protocols
- purposes and functions of the physical layer (Layer 1) of the OSI and the relationship of the physical layer to the Transmission Control Protocol/Internet Protocol model
- appropriateness of interviews, surveys and observation as methods of collecting data to determine needs and requirements
- features of functional and non-functional solution requirements
- constraints that influence solutions
- the functions, technical underpinnings and sources of worms, Trojans and spyware that intentionally threaten the security of networks
- factors that determine the scope of solutions
- tools and techniques for depicting the interfaces between solutions, users and the network, including use cases, via the Unified Modelling Language
- features of context diagrams and data flow diagrams that allow data flows to be depicted
- composition of an SRS and purposes of documenting an analysis in this form.

U3O2

- stages of the problem-solving methodology
- characteristics of data types: integer, floating point number, Boolean, character, string
- types of data structures, including one-dimensional arrays, records and files
- methods of expressing software designs using data dictionaries and data structure diagrams, object descriptions and pseudocode
- formatting and structural characteristics of efficient and effective input and output
- needs of users and how these influence the design of solutions
- criteria for evaluating the efficiency and effectiveness of solutions
- processing features of programming languages, including instructions, procedures, methods, functions and control structures
- purposes and characteristics of internal documentation
- techniques for checking that coded solutions meet design specifications, including construction of test data.

U4 AOS 1

- stages of the problem-solving methodology
- types and characteristics of mobile computing devices, including PDAs, mobile phones, laptops, gaming consoles
- procedures and techniques for handling and managing files, including security, archiving, backing up and disposing of files
- methods of organising files to suit particular software needs, including serial and random access
- ways in which file size, storage medium and organisation of files affect access of data
- characteristics of efficient and effective user interfaces
- factors affecting solution design, including user interface, user needs, processing efficiency, development time, technical specifications of mobile devices
- naming conventions for solution elements
- methods and techniques of expressing software designs
- forms and uses of data structures to organise and manipulate data, including two-dimensional arrays, stacks and queues

- validation techniques, including existence checking, range checking and type checking
- techniques for searching, including binary search, and techniques for sorting, including bubble sort and quick sort
- techniques for checking that coded solutions meet design specifications, including construction of test data.
- purposes and characteristics of internal documentation
- forms and types of user documentation, including printed, online Internet site (forms) and quick start guide, tutorial, content sensitive help and manual (types)
- applications and purposes of utilities in a programming environment
- legal obligations of programmers
- security measures designed to protect the integrity and security of data and information.

U4 AOS 2
- technical underpinnings of intranets, the Internet and virtual private networks
- characteristics of wired and wireless networks
- techniques for measuring the reliability and maintainability of networks, including audits, error logs and software tracking tools
- criteria and techniques for testing the security of networked environments
- characteristics of efficient and effective solutions
- strategies and techniques for acquiring evaluation data about the quality of solutions
- criteria and techniques for testing acceptance by users of solutions
- types of training for the users of solutions, and techniques for measuring the suitability of training programs
- types of system support documentation offered to users and criteria for determining their appropriateness for users
- practices that cause conflict between stakeholders who use, or are affected by, solutions that operate within networked environments
- suitability of setting codes of ethics, imposing sanctions, education programs and the use of decision-support frameworks as strategies for managing ethical dilemmas.

SOME (UNSORTED) USEFUL STUFF

Exam hints:
Always link back to the case study
Don't leave the exam early. You will have made a mistake. Go an find it.
They shouldn't get you to draw anything
Remember, just say what makes sense to you
DON'T MAKE ANY ASSUMPTIONS – STATE EVERYTHING
You won't lose marks for having too much information
The difference between functional and non-functional attributes will be stressed.
Discuss – Show both points of view

All ways put answers in the context of the question, no generic answers
Evaluation has similar data collection to the analysis stage
"Strategy" keyword in exam means answer has to include a number of steps and time is a factor

Describe the differences: just define each of the two things well.
Justify: explain why your choice is correct by contrasting with other options (say why others are wrong)
Use dot points so you can see how many points you have made so far.

Errors in example algos are usually logic off by one errors.

Operating system: controls the hardware, allocates resources to application software, manages security and communications.

Variable types: Int, Bool, float, arrary, string, Char?

DATA STRUCTURES:
In stacks, remember that on pop, the "top" value will be reduced by one, but the top value is not removed.
CHAPTER 1:
PSM:

Analysis	Design	Development	Evaluation
Solution Requirements Solution Constraints Scope of Solution	Solution Design Evaluation criteria	Manipulation (coding) Validation Testing Documentation	Strategy Report

Need to learn types and characteristics of mobile computing devices, including PDA's, mobile phones, laptops and gaming consoles

Key tasks of planning sofware projects including identifying, scheduling and monitoring tasks, resources, people and time.

Requirements can be classed as either functional or non-functional
Constraints can be cost, processing power, requirements that users have, legal requirements, security required, compatibility with existing hardware or software. There can be others.
Scope defines the boundaries of the software solution – what it will and won't do and any benefits.

Design involves working out how data that is required will be named, stored, processed, validated and manipulated

Documentation includes both internal (in the code) and external.

Strategy is how the software will be evaluated
Reporting is how the comparison will be reported

CHAPTER 2:
Open Systems Interconnection model
Has 7 layers. They are:
Media Layers:
Physical – concern with a machines interaction with a media (ie is plugged into a usb cable)
Data Link – MAC addresses are passed on this layer, Ethernet, switches and bridges operate here
Network - establishes way to send packets between nodes
Host layers:
Transport – concerned with flow control of packets also concerned with transmission of packets
Session – controls connections between computers
Presentation – controls how the data is presented to the application layer (ie decrypted)
Application – application specific: certain protocols operate on this layer such as FTP and HTTP

DPI is the amount of pixels in an inch. For example if the display is 2" by 2" and 400 px by 400 px, the DPI is 400/2 = 200 DPI

TCP IP is a data protocol for internet transfers, expand this

CHAPTER 3 ANALYSIS:

Determining system requirements:
Usually by data collection. Data can be quantitative (numbers) or qualitative (rainbows and feelings).
Surveys and questionnaires: used to get data from employees, clients, managements and others. Questions can be multi choice, sliding scales or long answer. Often cheap, but processing can be difficult
Interviews and focus groups: provides richer, qualitative data which may explain other data from other data collection method
Observations: can be used to see how the system works in a given snap shot of time. Provides an unbiased view however people who are observed often act differently.
Document reviews: reviewing documents produced by the system can be a cheap way of obtaining data.

Functional requirements: Are directly related to what the program will do, what inputs it will have, what outputs it will generate and how it will behave.

Non-functional requirements include
Response rates how quickly does the program respond to actions made by users?
User friendliness how clear, intuitive, logical and accessible is the solution?
Reliability how long can the system perform its required functions in its environment
Portability how easily can the solution be moved from one environment to another?
Robustness how well does the solution deal with garbage input
Maintainability how easily can the solution be fixed if problems occur?

Constraints: cost, requirements of users, expertise. Others include legal (copyright, privacy), speed required, capacity, time frame and availability of equipment.

All of the features of a software solution that have been discussed so far – timelines, resources, functional and non-functional requirements, constraints, scope, are documented in the form of a **Software Requirements Specification (SRS)**.

An SRS will be made up of an introduction, description of proposed solution, the specific requirements and a description of the environment in which it will operate.

CHAPTER 4: ANALYSIS TOOLS:

Use Case diagram: the purpose of a use case is to depict the functional aspects of a system within an organisation, and people's relationship with these. [1] If there is an arrow on the line, it implies that this user specifically initiates the use case, however if there is no arrow, it just implies there is a relationship.
Actor: a role external to the system. Described by a noun, often a organisation or a role in the organisation. (Can be other things also such as a machine)

1 You can find a lot more information here : domain.mhs.vic.edu.au/student/Faculties/InfoTech/Yr12SD/Staff %20Documents/Theory/SD-Use-Cases.ppt

Use case a representation of a function within the system. Described by <verb> <noun>, no “and”s
System boundary drawn around all the use cases and represents the edge of the system
Associations/Communications line between actor and use case shows link.
Includes links between use cases, showing that the functionality of one use case is used by another
extends links between use cases, showing that the functionality of one use case is enhanced by another

Data Flow diagram: their purpose is to visually represent the flow of data through a system.
Entity person agent or company outside the organisation
Data flow represents transfer of data. can be specific (address, surname) or general (Order details)
Data store a storage location within organisation. Only holds data and does not process
Process any activity that changes data

DFD rules:
Left to right, top to bottom
No Entity-Entity, Data Store-Data Store or Entity-Data Store/Data Store-Entity data flows
Avoid Black holes

Context diagram: This is a special type of DFD. It only displays the organisation and data flows from outside it.

CHAPTER 5 DESIGN

Data dictionary usually lists names of variables, their type, size in charaters, scope and a description of their function.
Data Structure Diagrams can be used to represent entities and their relationships. Entities are rectangles, and relationships are lines with annotations. Inverted arrows are used to represent a one to many relationship and must face downwards.
Object descriptions a table with object names, types and descriptions.
Top Down Design is breaking a problem in to smaller parts to complete.
Input-Process-Output charts contain three columns (as in title)
Annotated diagrams can show screen design and convey the user interfaces
Story Boards can be used to map the transition through several screens.

Algorithm structures
Sequence : algorithm that is one thing after another
Selection : conditional. Most commonly used as If-Then-Else
Repetition/Iteration : A number of tasks are performed a number of times or until a condition is met.

Pseudo code syntax
Start... Stop or **Begin … End** are used to indicate beginning and end of any program
Actions are just written line by line, and indented as normal in a programming language

```
If conditional
Then
        Action 1
        Action 2
Else
        Action 3
        Action 4
End if
```

This is correct syntax for an If statement. The Else and actions 3 and 4 are optional.

```
Case of Variable:
        value 1: Action 1
        value 2 to value 3: Action 2
        value 4+ : Action 3
End Case
```

This is correct syntax for a case statement, which is a simplification of a large if statement.

Count ← *number* assignment of *number* to variable "Count"

```
For Count ← first to last
        Action 1
        Action 2
Next Count
```

A fixed iteration loop that increases Count each time.

```
Repeat
        Action 1
        Action 2
Until Condition
```

repeat until condition evaluates to true

```
While Condition do
        Action 1
        Action 2
End While
```

test at beginning loop.

Other useful terms

Process
Display
Increment
Add, Multiply, Divide, Subtract
Calculate
Sum
Input, Output
<>,<,>.<=,>=

Arrays are declared as variables followed by a () with the parameters inside. A 2-d array will have two parameters, separated by a comma (NEED TO CONFIRM THIS)

Measures of Efficiency:
Ease of use: how is it is to use? How easy is it to learn to use
Speed of processing: is data able to be entered quickly, is output generated quickly
Functionality: how well are the functions of the user interface and how well do they address the tasks for which they have been designed.

Measures of Effectiveness:
Completeness: how complete is the output data? Is it everything the user needs or do they need to consult other sources
Readability/clarity: how easy is it to read and understand the output data?
Accuracy: is the information accurate; is it correct, does it often need to be adjusted?
Timeliness: is information produced within a relevant time frame
Relevance: does the information contain the required detail or elements, is the output relevant to the users request
Attractiveness: how attractive is the user interface?
Accessibility: how easy are the UI's functions easy to find and use?

Communication of message: how clearly is the solution presenting the information it is producing
Usability is the interface intuitive and easy to use?

Needs of users effect design:
The solution must cater for needs of users.
Age: older people require larger fonts, small children need large controls, Icons that do not assume a prior understanding or knowledge are desirable
Vision Impaired: are people colour blind? Then don't mix red/greens or provide alternate colour schemes. Do they have poor vision? Then provide a large high contrast UI.
Physical Impairments: these may influence ability to use mouse/keyboard, changing or limiting the interaction with the solution
Language: Does the solution cater for those with a different language background
Education: level of education should effect level and type of language used (ie any jargon)
Experience/Ability: level of experience of users will affect the amount of explanation the solution needs (ie less help pages) or how complicated the solution can be
Motivation: less motivated users need the solution to do things as easily as possible.

Other things that affect design:
Processing efficiency: The way the data is processed should effect what the user can see. (Ie if the data isn't available yet there should not be a "view data" option)
Development time: How long do we have to work on this? How much time can we spend on this?
Tech Specs: Limitations or characteristics of a device can influence design.

User Interfaces should be:
Clear an interface which is easy to read and understand.
Concise an interface that has succinct explanations and dialogs
Responsive as fast as possible, but also gives feedback (ie progress bars)
Familiar similar to other successful UI's
Consistent the Interface is similar across all the screens
Efficient tasks can be executed with a minimum of user effort
Attractive looks good (for the user)
Scalable allows for addition of new features so they will not change the layout adversely
Forgiving allows users to roll back mistakes (Ctrl Z). This gives a sense of security.

CHAPTER 8: ALGORITHMS

Selection Sort: examine every element of array, find the smallest one, put it at the start, repeat.
Bubble Sort: compares each pair of numbers and "bubbles" the largest number to the end. It typically takes n-1 passes. It can be modified to check if any swaps have been made on a pass and if not it know it is sorted.
Binary search: Check middle of array. If the value is not what you are looking for, look in either the top of bottom half of the array (if value was too small or too large respectively) and take the middle value again. Repeat until the value is found. The array must be sorted for this to work.

CHAPTER 9: DATA/FILES

Need to know Procedures and techniques for handlig and managing files, including security, archiving, backing up and disposing of files.

Archiving: data is archived as no-one wants to throw away data but do not want the associated costs with keeping it available (Ie on a database server)

Backing up: Backing up is usually done at night as they take time and no files are allowed to be open.
Full: All data is copied to the backup media. Uses lots of space, however is easy to restore.
Differential Backup: A full backup is made and then the only other backup which is kept is a copy of all the files that have been changed (A differential backup). This can mean that the backup is up to date without taking up heaps of space. However, the differential backup is slow to restore and the differential backup may grow till another full backup needs to be made.
Incremental Backup: A full backup is made, and then subsequent (incremental) backups are made from the differences since the last incremental backup. This makes the process very fast HOWEVER, it is much more complex to restore as the full backup must be made and then all the incremental backups need to made.

RAID: Redundant array of inexpensive disks. Increases reliability and read speeds, however is not really a good alternative to backing up your data.

Files are not often disposed of but rather will be archived.

Attributes of files that will affect access of data
This is not a 1 way relationship as the need to have files accessed in a certain way will change the way it is structured.
A serial file consists of sets of data of the same record type (but usually the records are of different lengths) that is stored *in the same order in which the records were created*. This is like a cassette tape, where you must fast forward to the position you want.[2] You must read in from the start to find anything.
Random access files must be stored on a random access storage device. They often have an index file that lets you directly seek and go to the start of a desired record. This is like a cd where you can just change tracks (or like MySQL). Majority of files are random files. It makes it faster find records
Storage media also effects how files are accessed, as they other attributes such as latency, reliability, cost, ease of use etc.

Validation: These three types can be used in order.
Existence Checking: Check that the value exists (is not null)
Type Checking: Check that the variable is of the correct type.
Range Checking: Check that the value is in the correct range of values.

CHAPTER 10: DOCUMENTATION AND TESTING

Types of errors:
Syntax: the implementation of the solution was syntactically incorrect for the language, and the program is not a valid program in that language.
Logic: The algorithm or implementation of it is wrong (Ie program fetches wrong db entries). These are much harder to detect than syntax errors which will normally be caught by an IDE or compiler. "A logic error produces unintended or undesired output or other behavior, although it may not immediately be recognized as such."[3] They can also be referred to as semantic errors.
Run Time: When something that was not planned for occurs, such as an index out of bounds of the array. "These typically occur when your program attempts an operation that is impossible to carry out."[4]

2
3 http://en.wikipedia.org/wiki/Logic_error
4 http://msdn.microsoft.com/en-us/library/s9ek7a19%28v=vs.80%29.aspx

Examples of errors:
Arithmetic over/under flow: This occurs when a number becomes to small or large to fit in its variable type.
Memory leak: This occurs when a program keeps creating variables and loading things into memory and does not remove them from memory. Eventually large amounts of memory are consumed.
Handle leak: Similar to the memory leak, this occurs when the program assigns handles to resources and does not remove them, creating multiple handles for every resource.
Buffer overflow: a buffer is a temporary memory location to store data that is either input or output before it is processed. Sometimes the input can be too large and fill the buffer completely.
Stack overflow: Occurs when 2 subroutines reference each other and fill up the stack, causing a program crash.
Deadlock: Occurs when two or more processes are waiting for each other to finish.
Off by one: Very common in loops – issue is normally with a one or zero based array, or a <= sign instead of a <
race hazard: A process that needs the data from another process executes before the other one
Type conversion: Some types cannot be casted to each other (ie string to float) and others (float to int) cause the variable to change dramatically.

Testing table contains colomns item tested, test data, expected result, actual result.
Good values to test are boundary values, values inside and outside the allowable range.

Watches and breakpoints some IDEs allow you to step through code line by line, and also "watch" variable values as the program executes.
Large file sizes Testing large file sizes is important as it may occur regularly once fully developed.
Benchmarking most commonly occurs in the analysis stage and checks the organisations actions against best practise. In the development stage it is a means of documenting the capabilities of the solution.
Live Data the program should be tested on live data from the day to day operations to see how it copes.

Types of documentation:
User guide/manual is generally not too technical in nature and will have step by step instructions of how to use the solution.
Technical Reference Manual this will go into in depth detail for someone who will be maintaining the solution. While it probably won't discuss the code it will discuss issues like compatibility, security, system files and network use.
Installation manual Similar to the Technical Reference Manual, this will describe how to install the system and get it fully working. Will often contain FAQ and a troubleshooting section.
Quick Start Guide this is a short and simple guide to help the user start the program up and execute many of the functions of the solution.
Help Guide this could also take the form of a FAQ or troubleshooting guide and is often based on diagnostic help, providing detailed assistance so the users can solve any difficulties they are having with the software.
Tutorials these take the user step by step through the main functions of the solution. They can easily be placed inside the solution.
Context sensitive help this is a type of help which often is triggered based on a certain event inside the program or it can be triggered when the user accesses the help menu in the program. Context sensitive help is not viewed in a complete form as it would not be useful in that form.

CHAPTER 11: THE LAW

The Acts:
Privacy Act 1988 applies to Government organisations, local councils or any organisation contracted by a government organisation. It has 11 Information Privacy Principles
Information Privacy Act 2000 applies to the private sector. It is an addition to the Privacy Act. It is mandatory for organisations with a turnover greater than 3 million dollars per year, organisations that store medical data or sell or distribute personal data. It has 10 National Privacy Principles.
Health Records Act 2001 applies to any organisation that deals with medical information. It is another addition to the Privacy Act. It establishes 11 Health Privacy Principles.
Copyright Act 1968 has all the copyright laws. In Australia you do not need to apply for copyright or place a (c) on your work – copyright is automatic.
Copyright Amendment Act 2006 was used to make legal format shifting of music for personal use and recording of TV for personal use. It also brought our Copyright laws into compatibility with the US copyright laws.
Charter of Human Rights and Responsibilities Act 2006 (VIC) has 3 sections that are relevant to us. They are
Section 13: Privacy and Reputation each individual has a right to privacy and the right to not have their reputation unlawfully attacked.
Section 14: Freedom of thought, conscience, religion and belief each person has freedom as specified in the title and may express these in any (legal) way.
Section 15: Freedom of expression an induvidiual has the right to express an opinion in any medium. However these expressions must be legal (ie not a breach of privacy, not hate speach)
Spam Act 2003 states that emails should only be sent to those who have consented to receive then, and that should be able to remove themselves from a mailing list and any organisation that sends an email must identify that they are sending the message.

Code of Ethics many organisations have a code of ethics. This isn't legally binding but sets the standard for members of the organisation. Usually not mandated. It is a guideline, and is something you would sign up for. Code of Ethics can help unethical situations by: allowing people to know what to prioritize in an ethical dilemma and will help people explain why they cannot do something.

CHAPTER 12: SECURITY AND HAXXo0RZ

Security Threats: actions device and events that threaten the integrity and securit of data and information in or between information systes (Paraphrased from study design)

Internet Security can be classed as risk management as no-one can establish 100% security but trying to attain this can cost large sums of money.

Security is concerned with Protection from threats and Detection of Threats. Detection can be done via a honey pot.

Firewalls filter traffic based on a set of rules and can be software or hardware based.
Malwarez:
Viruses: a program designed to replicate itself. Often transferred by secondary storage devices. A virus is often attached to legitimate programs so it can be executed to deliver its payload.
Worms: usually designed only to consume bandwidth in a network, a worm is designed to rapidly replicate. If it does have a payload it will usually turn the computer into a zombie.
"Computer worms use the network to send copies of themselves to other PCs, usually utilizing a security hole to travel from one host to the next, often automatically without user intervention. Because they can spread so rapidly across a network, infecting every PC in their path, they tend to

be the most well-known type of malware, although many users still mistakenly refer to them as viruses."[5] Some of the most famous worms have travelled through email.
Trojans: Are programs designed to look like a different program. Trojans will deliver a payload but do not replicate.
Spyware: It is a type of malware specifically designed to collect information. It is often a payload of one of the other types of malware. *Adware* is a subset of malware that will display adds to the user to generate income for the owner of the Adware.

Zombies and Botnets are collections of compromised computers that can do things for the person in control. For example, they can deliver DDOS attacks, send spam and mine bitcoins.

Encryption
Symmetric Key one key is used to encrypt and decrypt data. It is fast however has issues such as how the original key is sent to the reciever.
Asymmetric Key a public key is used to encrypt data – this is sent to everyone who wants to everyone who wishes to communicate with the server, but only the private key can decrypt data. It is slow.
Hybrid Encryption is a better solution. It uses asymmetric key encryption to send a symmetric key to the other party. The rest of the transfer uses symmetric key encryption.
TLS and SSL. Secure Socket Layer is a part of Transport Layer Security. It employs Hybrid Encryption and a handshake procedure to authenticate the server and then encrypt all of the data.

Networks
Hubs and sniffing. On a hub, as every node gets all the data, you can easily sniff packets to listen in on any communications.
Switches and MAC address spoofing. Switches cannot be sniffed, but if you spoof the MAC address of another computer, you can receive their data. This can easily be detected as the real computer will stop receiving data.
Wireless. On open wireless networks, people can easily sniff your packets. HTTPS is a good option to avoid this and a better one would be using a VPN.

Logical Security is software measures to counter unauthorised access such as logins, levels of access and password policies.
Physical Security is about preventing unauthorised physical access to hardware. **Biometrics** can be used to do this, as it involves identifying an individual based on unique characteristics.

Testing security
Analysis of Detected Attacks
This involves analysing log files and the like, the following questions are useful for unsuccessful attacks:
"How many different types of attacks were detected?"
"How were they detected and were they repelled in sufficient manner?"
"Is there anything that could be done to reduce the number of attacks?"
"What information was able to be gathered about where the attacks originated?"
(From Software Development Core Techniques and Principles, by Adrian Janson, pg 173)

For successful attacks, the following questions are useful:
"Where did the attack originate?"
"Did the attacker exploit an existing security weakness or is there a new vulnerability that needs to be attended to?"
"What data or information was stolen?"

5 http://lifehacker.com/5560443/whats-the-difference-between-viruses-trojans-worms-and-other-malware

“Did the attacker appear to seek a specific resource or was the intrusion an opportunistic one?” (From Software Development Core Techniques and Principles, by Adrian Janson, pg 173)

Penetration Testing has consultants trying to break into the system. If they know nothing about the system it is called “Black Box testing” but if they know as much as an insider would then it is “white box”. The employees are the greatest weakness to security.

Security Audits involve security professionals examining and testing? a network, providing a report on issues with the network.

CHAPTER 13: EVALUATION

Evaluation generally takes place 6 months after deployment.

First a <u>strategy</u> is determined, then it is executed and the results are <u>reported</u>.

Quality can be evaluated using the efficiency and effectiveness measures from Chapter 5. There are two techniques used to compare these measures with the goals of the solution. They are Acceptance testing and Quality Assurance.

Quality Assurance is usually a large set of testing to make sure that a new solution is working as planned. It is used to ensure that it meets the needs of the designers and the organisation.

Acceptance testing involves the employees and management in ensuring everything works. Representatives of the organisation are given a chance to test the solution and give feedback on any problems or issues. Specification Creep must be dealt with at this point because the testing is intended to show the original goals have been met. If other goals have been added, additional testing must be used to address these.

Training
Off-site training occurs at an external venue. This minimises any distraction their actual work would involve. Of course, issues may arise with keeping the business running while this occurs.
On-site training is conducted at the organisation. The advantage is that they learn to use the solution in the environment they will be using it.
Train the trainer a select few can be trained up to train others. It can be cheaper than training everyone although the trainer will have to focus more on training than working for some time.
On-line training is on a electronic device. It is cheap and easy to implement. It has many difficulties such as distraction and lack of motivation.

Not all training methods are suitable and can be determined from the environment, availability of money, time frame and number of employees involved.

Documentation
Each of the previously mentioned types of training are suitable for different things.

Decision Support Framework: this is a framework for solving conflicts within an organisation. It may set out a conflict resolution process that parties must go through. It aims to avoid conflicts and provide a process to resolve them when they occur.