# Universiti Teknologi PETRONAS (UTP) E-Learning and content development

By

MAZLIN MOHD RAZALI

Dissertation submitted in partial fulfillment of
the requirement for the
Bachelor of Technology (Hons)
(Information and Communication Technology)
JAN 2007

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

# Universiti Teknologi PETRONAS (UTP) E-Learning and content development

By

Mazlin Mohd Razali

A project dissertation submitted to the
Information and Communication Technology Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
BACHELOR OF TECHNOLOGY (Hons)
(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,

(Mr. Saipunidzam bin Mahamad)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

Jan 2007

# CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contain herein have not to been undertaken or done by specified sources or person.

MAZLIN MOHD RAZALI

# ABSTRACT

paper describes about the implementation of open source and content development in UTP earning. As the technology evolves and the need for open source environment arises, a new digm is needed to allow E-Learning to move into this open source environment. A few open ce programs were analyzed and Moodle was chose as the most suitable open source to elop UTP's new E-Learning. Moodle appears to be one of the most user-friendly and flexible n source courseware products available and has excellent documentation, strong support for rity and administration, and is evolving towards IMS/SCORM standards. SCORM was eloped with the guidance of a large consortium of academic, technological and governmental nizations. SCORM is maintained by the Advanced Distributed Learning (ADL) Initiative. objective of implementing open source and content development in developing UTP's E-ning is mainly to help initiate an advance Information and Communication Technology ) environment for student. Methodology of developing this project has already been defined probably pose much similar with traditional information-development process. Tools irement such as the Moodle software itself and other supportive software and hardware are tified to develop this project.

# ACKNOWLEDGEMENT

:t of all, praise goes to Allah S.W.T for giving the author the strength, skills, knowledge and d health in completing the Final Year Project, titled Universiti Teknologi PETRONAS (UTP) ,earning and content development. Warmest gratitude to Mr. Saipunidzam bin Mahamad for generosity, unwavering support and guidance during the long sessions of advice that make author achieve the objectives meant to at the conception of this project. It was fortunate to e worked with such a wonderful person. Thank You.

;pest appreciation goes to Mesiniaga, especially Mrs. Asmawiyah Amat Sajuri, which is ly supportive person, giving good cooperation and giving such useful information and lelines regarding the information on content development that make the task manageable and thwhile.

cial thanks goes to the Final Year Project Coordinator, Mrs. Aliza Sarlan and committee for their patient and understanding in guiding and assisting the author in this project. Not ;etting, all Computer Information System (CIS) lecturers for their valuable expertise, guide support in completing this project. Your comments, suggestion and constructive criticism le a real difference in the quality of this project. Thank You.

t but not least, sincere thanks to all the people who have been involved directly or indirectly completing the Universiti Teknologi PETRONAS (UTP) E-Learning and content elopment. The contributions made throughout the development of this project really valuable ie author.

.in, thank you for your encouragement and continuous support.

## T OF FIGURES

# Γ OF TABLES

# TABLE OF CONTENT

# CHAPTER 1

# INTRODUCTION

## .1. Background

Universiti Teknologi PETRONAS (UTP) is using Blackboard as their E-Learning. lackboard is a learning management system (LMS) software partially owned by Microsoft. It s popular software used around the world and licensed annually. Blackboard is very costly vhere over $1 million spent each year on the Blackboard licenses. Other approach to reduce the ost of spending money for Blackboard licenses, UTP is developing E-Learning by using open ource.

In general, open source refers to any program whose source code is made available for e or modification as users or other developers see fit. Historically, the makers of proprietary ftware have generally not made source code available. Open source software is usually veloped as a public collaboration and made freely available.

Other than that, open source is similar in idea to free software but slightly less rigid than e free software movement. Users of open source software are generally able to view the source de, alter and re-distribute open source software. There is however less of an emphasis in the en source movement on the right of information and source code to be free and in some cases mpanies are able to develop proprietary products based on open source ones.

## 1.2. Problem Statement

Currently used E-Learning is costing UTP in term of paying for the Blackboard licenses. Since UTP Blackboard license will be end soon, so UTP is moving to open source environment. Open source offers a radically different and exponentially better software development model. Companies can improve their products greatly and significantly increase their market share. Overall, open source is good for everyone.

## 1.3. Objectives

1. To customize Moodle to suit UTP preferences.
2. To evaluate the implementation of Moodle using open source.
3. To implement SCORM in Moodle content development.

## 1.4. Scope of Study

The implementation of UTP E-Learning is using Moodle, which is open source software and e-learning platform. Moodle is a software package designed to help educators create quality online courses. It is an open source web-based course management system designed based on the principles of social pedagogy. Moodle produces Internet-based courses and websites; it requires a web server with integrated PHP which is the scripting language and database support. Moodle supplements the traditional face-to-face learning by providing online-classes, is user-friendly, simple, efficient, compatible, easy to install on any platform that supports PHP. It requires an underlying mySQL database. The entire Moodle system is managed by an admin user during setup. The administrator is permitted to customize the site colors, fonts or layout to suit their needs. The Moodle is then added with the required activity modules. The source code is easy to modify to suit the student's needs.

For content development, basically, the author is developing a courseware as a learning material for the e-Learning users of Universiti Teknologi PETRONAS (UTP). This courseware will guide the user in the learning process. It can be accessed from the Content Management System (CMS), Moodle. In order to develop the courseware to be interoperability, accessibility, durability and reusability, the author will be implementing the Sharable Content Object Reference Model (SCORM). SCORM is a suite of technical standards that enable web-based learning systems to find, import, share, reuse, and export learning content in a standardized way.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1. E-LEARNING IMPLEMENTATION USING OPEN SOURCE

### 2.1.1. Introduction

Before we can intelligently discuss the advantage and disadvantages of open source and differences between open source and proprietary software, we need to clarify what the term really means. Many people equate open source with free of charge, but that's not necessarily the case. Open source code can be and is the basis for products such as RedHat and dozens of other commercial distributions of Linux that range in cost from a few dollars to a few thousand.

Open source also does not mean unlicensed. In fact, there are a whole slew of licenses under which open source software is distributed. Some of the most popular include GPL the GNU Public License, BSD, and the Mozilla Public License. The Open Source Initiative (OSI), a non-profit corporation, has developed a certification process for licenses.

GPL refers to a type of open source license called the 'GNU General Public License'. Copyright protects a creator's right to control copies and changes to a work, whereas the GPL license protects a user's right to copy and change a work. The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software, to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it. For example, if you distribute a program, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. We protect your rights with two steps:

- Copyright the software,
- Offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations. Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

### 2.1.2. Open source system

The name itself tells the story, open source software means the source code or the programming often written in C, C++ or even assembler language is available to anyone who wants it, and can be examined, changed and used to write additional programming. This is in contrast to closed or proprietary software such as Microsoft Windows, for which the source code is a closely guarded trade secret except when it's leaked to the public. Open Source is any software whose license provides:

- Right to access the source code
- Right to modify it
- Right to copy it

The advantages of using the Open Source software to implement solutions are:

- Availability of the source code and right to modify it which reduces dependence on the supplier and increases flexibility.
- There is no cost for license or for copies which allows the installation of the software in as many computers as necessary without extra costs.
- Open Source products are used by many users which proves their quality.
- Additional support can be provided by many existing communities.
- The development is easy and technical support is usually immediate.
- The development of the software is based on a co-operative model which increases the quality.
- Programmers are usually also users themselves so they take extra care to ensure the quality and functionality of the software.

The disadvantages of using the Open Source software are there is no written guarantee of the quality even though it is proved to be a high quality product. The developers are not liable for any that the software might cause.

There is a lot of open source software in the market. Here are differences between ATutor, Claroline and Moodle:

| MOODLE | ATutor | Claroline |
|---|---|---|
| • most user-friendly and flexible open source<br>• excellent documentation<br>• strong security and administration<br>• evolving towards IMS/SCORM standards<br>• A strong envelopment and user community | • Very few LMS' that support learning object storage<br>• can import external content in IMS/SCORM format<br>• many features and rates highly for usability, including accessibility for learners with disabilities | • Has a broad user base but still have problem over some features<br>• Similar to Moodle and ATutor alternatives |

**TABLE 2.0:** Difference between ATutor, Claroline and Moodle

### 2.1.3. Moodle

Moodle is one of the most user-friendly and flexible open source courseware products available. It has excellent documentation, strong support for security and administration, and is evolving towards IMS/SCORM standards. The key to Moodle is that it is developed with both pedagogy and technology in mind. It is a robust development and user community. It is also great with languages although some development may be needed for robust handling of MathML and enhanced tracking features. Still, this program receives a high recommendation. Moodle is a student-centered course management system designed to help educators who want to create quality online courses. The software is used all over the world by universities, schools, companies and independent teachers.

### 2.1.4. ATutor

ATutor is a promising system that provides good documentation, ease of installation, and strong potential for development. While the user interface may not seem intuitive to many, the overall functionality is good (and wide open/modular) and the development team is committed to standards. The system is also install-friendly and receptive to new language versions.1. ATutor is one of very few LMS' that support learning object repositories. ATutor is very strong on standards and can import external content in IMS/SCORM format. ATutor scores highly for openness. It is written in a modular format. It has many features and rates highly for usability, including accessibility for learners with disabilities.

### 2.1.5. Claroline

Has a broad user base but there are questions over particular features, such as the gradebook, that are likely to be desirable by consortium partners. Claroline is similar to Moodle and ATutor alternatives. The development pathway indicates an increasing commitment to open standards.

### 2.1.6. Conclusion

After all, Moodle is the best among all. Moodle's features are very rich where most of ATutor's and Claroline are more basic. Also, additional functionality offers much more in Moodle than in others. It seems that ATutor's and Claroline focus on accessibility takes away some of the attention from enriching and adding of features. Some technical issues came up during installation of ATutor and Claroline, while Moodle installs and upgrades smoothly time and again. Moodle appears much more comfortable and intuitive to navigate and work with than ATutor.

## 1.2. CONTENT DEVELOPMENT

Content development is the process of researching, writing, gathering, organizing, and editing information for publication on web sites. Web site content may consist of text, graphics, pictures, recordings, movies or other media assets that could be distributed by a hypertext transfer protocol server, and viewed by a web browser.

### 1.2.1. What is SCORM?

SCORM, which stands for "Shareable Content Object Reference Model", is a standard for developing, packaging and delivering high-quality training materials for online training courses. SCORM was developed based on the concept of using a common e-learning standard to modernize education and training courses. Now, with the SCORM standard, and more specifically SCORM compliant platforms, you can find, import, share, reuse and export learning contents through compliant web-based learning systems.

In short, SCORM is a set of technical standards that have been developed with the specific purpose of enabling the reuse of Web-based learning content across multiple environments and products. If your lesson content is SCORM compliant, then you can use the content in any course management system or learning management system that supports SCORM without having to make any modifications to the content. Additionally, the SCORM standards allow your content to interface automatically with specific components of the course management system. It's not unusual for people involved in authoring and developing online courses to question the need and rationale for building SCORM-compliant courses.

### 1.2.2. Why SCORM?

SCORM is a compilation of technical specifications that enable interoperability, accessibility and reusability of Web-based learning content. E-learning content and learning management systems built to the SCORM specification allow for sharing of courses across federal agencies and even with other entities, such as nongovernmental colleges and universities.

SCORM also allows for sharing of smaller pieces of e-learning courses, called learning objects. Learning objects include course modules, book chapters, videos or even smaller, discrete pieces of information. The result will be faster, more personalized learning experiences, e-learning experts said. Those learning objects could also be easily incorporated into other e-learning courses.

SCORM also allows courses to be updated without starting over. If the course has been built to the SCORM specification, it's relatively inexpensive to change just the piece pertaining to the decision, said Elliott Masie, president of the Masie Center, a Saratoga Springs, N.Y., and e-learning think tank. There are significant benefits to adopting SCORM:

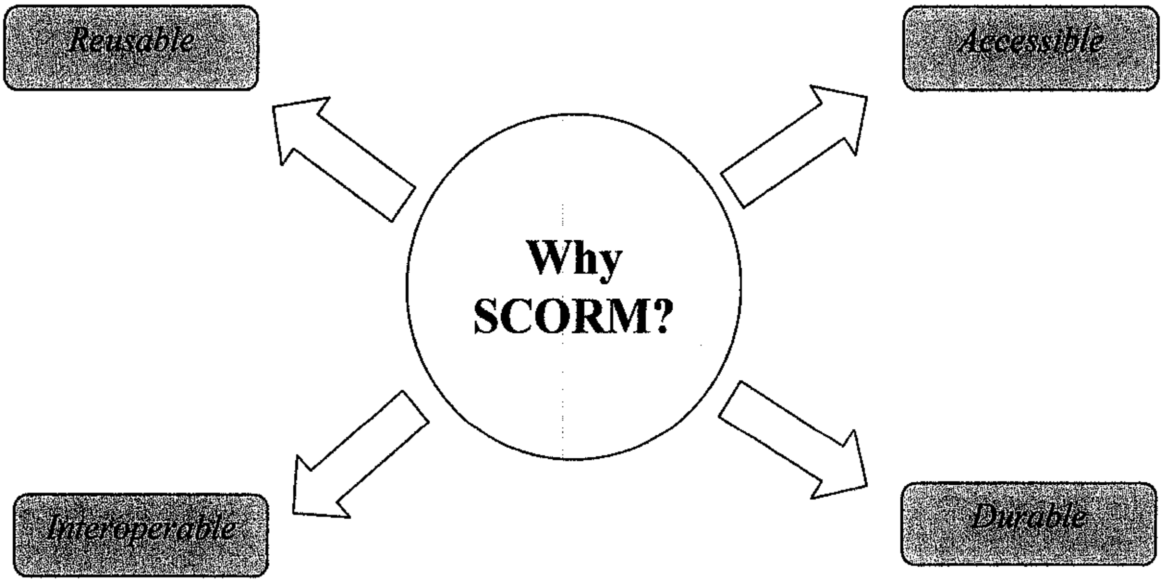


**Figure 1.0:** Reason why use SCORM

- *Reusability: Lowers costs because learning contents can be used many times.*

  SCORM incorporated contents can be easily incorporated into multiple applications, allowing for learning contents to be easily modified and used by different tools and platforms. For example, SCORM compliant learning contents produced by an authoring tool such as Macromedia Flash can be reused, modified by another set of SCORM compliant tools. This allows you to spend less effort in creating new contents from scratch and focus more on incorporating existing contents to meet the specific needs.

- *Accessibility: Increased access to learning information.*

  The SCORM standard offers you the ability to access learning contents from anywhere and at anytime. It enables you to customize the delivery of your learning contents from one remote location and deliver them too many other locations. It allows you to search, identify, access and retrieve contents that are widely distributed in SCORM compliant repositories. This way, it is easier for learners to identify, retrieve, and incorporate valuable content from many trusted sources.

- *Interoperability: Systems works efficiently together*

  SCORM provides an extension for compliant courseware and similar tools to communicate under a common interface. With consistent content production specifications, learning contents developed using one set of tools can be easily identified, and launched with a different set of tools. In other words, SCORM compliant learning contents can be authored and launched by any SCORM compliant authoring tools, providing maximum flexibility.

- *Durability: Learning contents will not easily become obsolete.*

SCORM is becoming the technological standard for the e-Learning industry because SCORM compliant contents are preserved and portable even when base technology changes. This means learning contents do not need to go through a redesign or redevelopment process, even when technology changes.

# CHAPTER 3

# METHODOLOGY

## 3.1. PHASES OF PROJECT DEVELOPMENT

The methodology that will be use in this project contains many of the same elements as traditional information-development process. The four phases of the methodology follows:

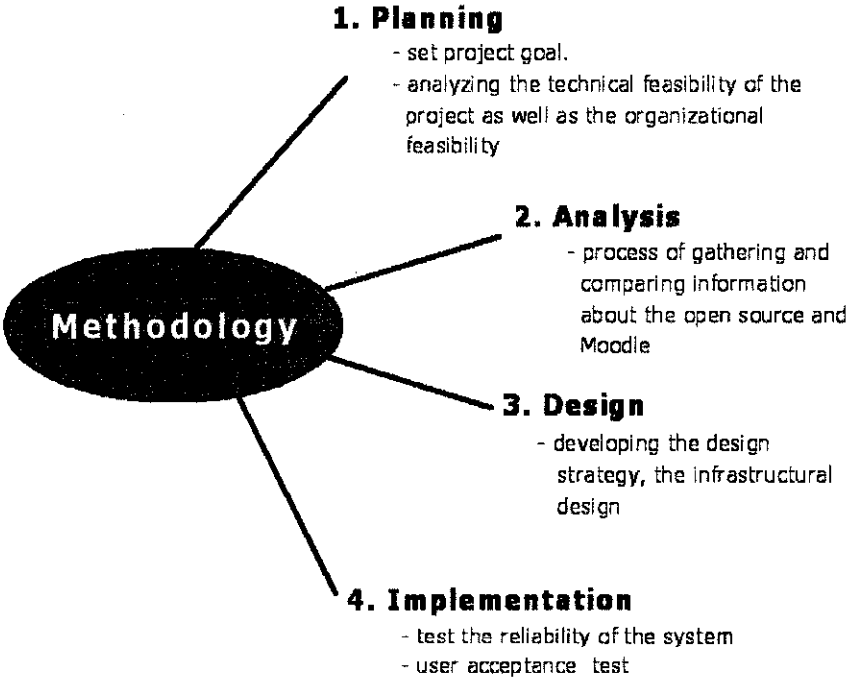

Methodology
1. Planning
- set project goal.
- analyzing the technical feasibility of the project as well as the organizational feasibility
2. Analysis
- process of gathering and comparing information about the open source and Moodle
3. Design
- developing the design strategy, the infrastructural design
4. Implementation
- test the reliability of the system
- user acceptance test

**Figure 3.0:** Methodology of the project development

### 3.1.1. Planning Phase

In the planning phase, goals for the project are set by choosing several factors. The goals include anticipating and deciding on the target audience, purpose and objectives for the information. Other than that, the tasks that will be involved are the project initiation where I need to select the project topic and come out with a project proposal. Then, next task will be the project management. This project management task includes analyzing the technical feasibility of the project as well as the organizational feasibility.

After the analysis made, I will come out with the preliminary research work where I need to study on several previous study made on the same project topic or scope of the system that I will develop. This includes articles, journals, and surf for information in the internet or books on the technology being used that is Open Source.

### 3.1.2. Analysis Phase

To develop the project, the process of gathering and comparing information about the open source and Moodle will be done. Any queries about the information will be discussed for further understanding. The tasks include developing analysis plan where here the requirements gathering take place. The user requirements will be gathered in various ways such as surveys, interview or discussions. The objectives of this requirement gathering is to ensure that the project developed meet the user satisfaction as well as to see the factors that contributes to the system development.

After the requirements gathering, the next task will be the use-case modeling. This includes developing uses cases, structural model for example class diagram and developing the behavioral model such as the sequence diagram. Those models and diagrams are used to understand the system flow and also to determine the object and the interaction of the object with each function. This also used as guidelines in developing the system afterwards.

### 3.1.3. Design Phase

In the design phase, the programming part takes place but before that, the first task that needs to be completed is the system architectural design. This includes developing the design strategy, the infrastructural design such as the system network model, the hardware and software specifications and also the security plan for the system.

Then, the next task will be the user interface design which then followed by the programming management. This will take quite some time as the building source code takes place.

According to Moodle specification, there will be some requirements to be taken into consideration. SCORM in content development is also collection of specifications that enable interoperability, accessibility and reusability of web-based learning content. The design phase also emphasizes the user interface of the content.

### 3.1.4. Implementation Phase

The system will be build using all the available tools. In developing the system, the author uses the Macromedia Dreamweaver software to manipulate the index page which also known as the landing page of the E-Learning system. The author also will be using Macromedia Flash MX software and SCORM as the standard for content development.

In this phase, testing also will be conducted to test the reliability of the system. Then, if the management of changes will take place if there is any changes need to be made. Finally, the system will be assessed and the study of the user acceptance will be made after the system is completed and tested.

## 3.2. TOOLS, REQUIREMENT AND UTILITIES

### 3.2.1. Moodle

Moodle is primarily developed in Linux using Apache, MySQL and PHP also sometimes known as the LAMP platform, but is also regularly tested with PostgreSQL and on Windows XP, Mac OS X and Netware 6 operating systems. The requirements for Moodle are as follows:

- Web server software. Most people use Apache, but Moodle should work fine under any web server that supports PHP, such as IIS on Windows platforms. PHP does impose requirements on versions of web servers, however these are complex and the general advice is to use the newest version possible of your chosen web server.
- PHP scripting language. PHP 5 is supported as of Moodle 1.4. From Moodle version 1.6, the minimum version of PHP will be 4.3.0.
- A working database server, MySQL or PostgreSQL are completely supported and recommended for use with Moodle. MySQL is the choice for many people because it is very popular, but there are some arguments in favour of PostgreSQL, especially if you are planning a large deployment. The minimum version of MySql required for Moodle 1.5 is 3.23. Moodle 1.6 requires MySQL 4.1.16. The minimum version of PostgreSQL is 7.4 and it is widely used with 8.0 and 8.1.

### 3.2.2. Implement and Development

- 64 MB RAM (recommended)
- 160 MB free Fixed Disk
- Windows 98, ME, Windows NT, 2000, XP (Recommended)

### 3.2.3. Content Development

- Macromedia Flash MX
- Adobe Photoshop
- SCORM

### 3.2.4. SCORM

- Macromedia Dreamweaver MX
- SCORM Runtime Wrapper extension
- RELOAD (Reusable E-Learning Object Authoring and Delivery) Editor

# CHAPTER 4

# RESULT AND DISCUSSION

## 4.1. ARCHITECTURE

### 4.1.1. System Architecture

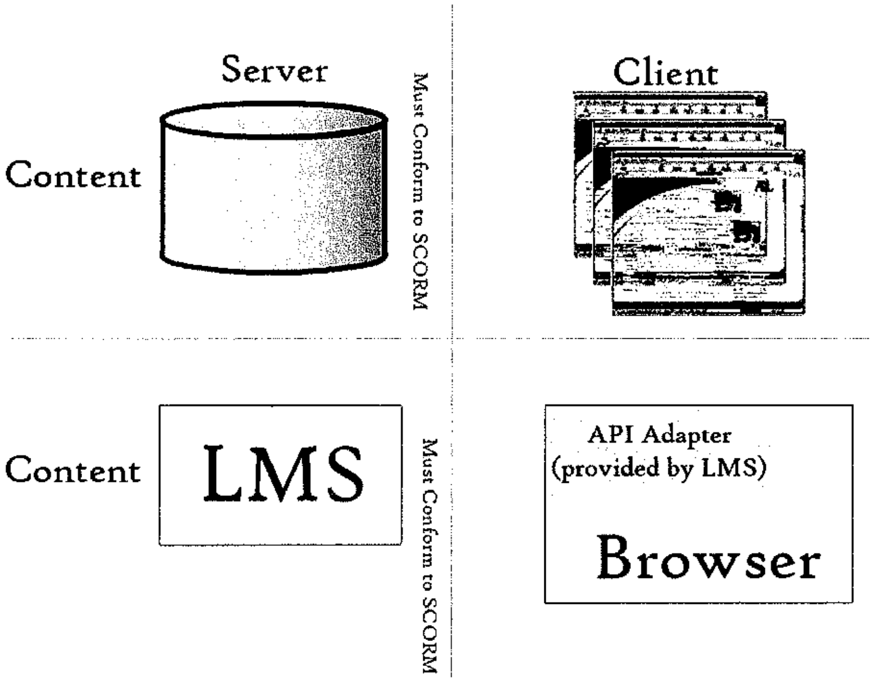


**FIGURE 4.0:** Diagram on how SCORM works

### 4.1.1.1. Content Development

#### 4.1.1.1.1. SCO

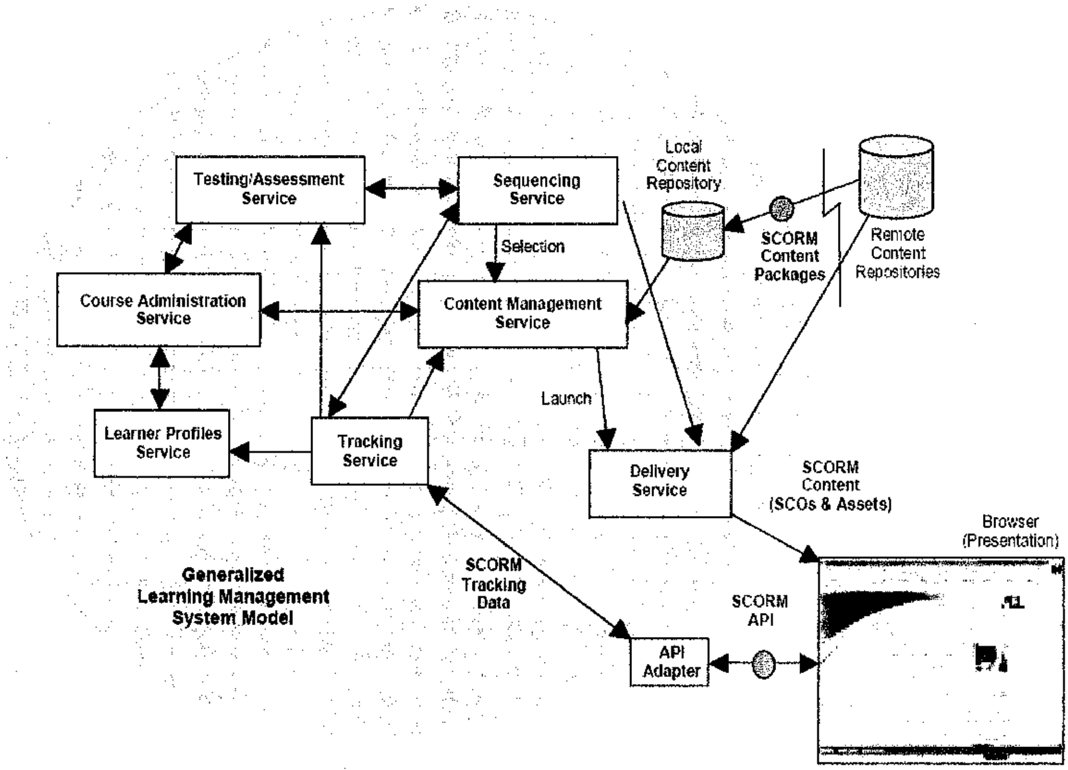


**FIGURE 4.1:** Highly generalized model of a Learning Management System (LMS)

E-Learning content is delivered as a single unit called a Sharable Content Object or SCO. SCOs are independent, self-contained, transportable packages that represent the lowest level of granularity that is tracked by a learning management system. In other words, SCOs are mini-applications that when launched from an LMS report, among other things, one score and pass/fail status. Your SCOs should contain all the necessary files required to function, have or require no knowledge of any other SCO or external information, and as a matter of design, should remain relatively small such that they can be easily reused.

And since the specification indicates that SCOs shall be transportable, you could infer that they should also be platform independent, although this is not explicitly stated. Server dependencies such as Coldfusion, Perl, ASP or server-based databases are problematic as a particular LMS might be running on a server that does not have those services available. So unless you are working in a tightly controlled internal environment, your SCOs should only be using client-side technologies such as HTML, Flash and Javascript.

#### 4.1.1.1.2. Content Package

Now before we get into the details of the SCO, realize that we must to be able to transport it, even if that only means getting it from our desktop computer where we created it to the server where our LMS is hosted. To do so, we use a Content Package. Going back to our wall socket analogy, imagine that the content package is the box in which the toaster is packed and transported. In reality, it is simply a single compressed file containing all the files such as html, images, and flash swf, necessary for the SCO to function. The content package may contain more than one SCO and a single manifest file which describes the contents of the package. The LMS will use the manifest file to properly find and import each of the SCOs.

#### 4.1.1.1.3. Web-Based and Script Enabled

While there is no specific requirement that learning management systems be web-based, SCOs must be. This requirement gives developers a common platform, the web browser to design and develop their content to function within, thereby removing unnecessary barriers to compatibility and transportability. Of course, even with the current maturity of web-browsers, they are not all created equal and SCORM makes no mention of a specification to be followed to ensure web-browser compatibility. Compliance with the latest W3C standards however, is a safe course of action to ensure compatibility with a wide range of current web-browsers.

And in order for a SCO to communicate data to and from the LMS as well as potentially provide interactivity and quizzing functionality, it's going to require more than just client-side HTML. SCORM specifies that SCOs use an ECMA Standards compliant scripting language to communicate with the LMS. The most notable ECMA compliant language is one you've probably heard of - JavaScript.

### 4.1.1.1.4. SCORM

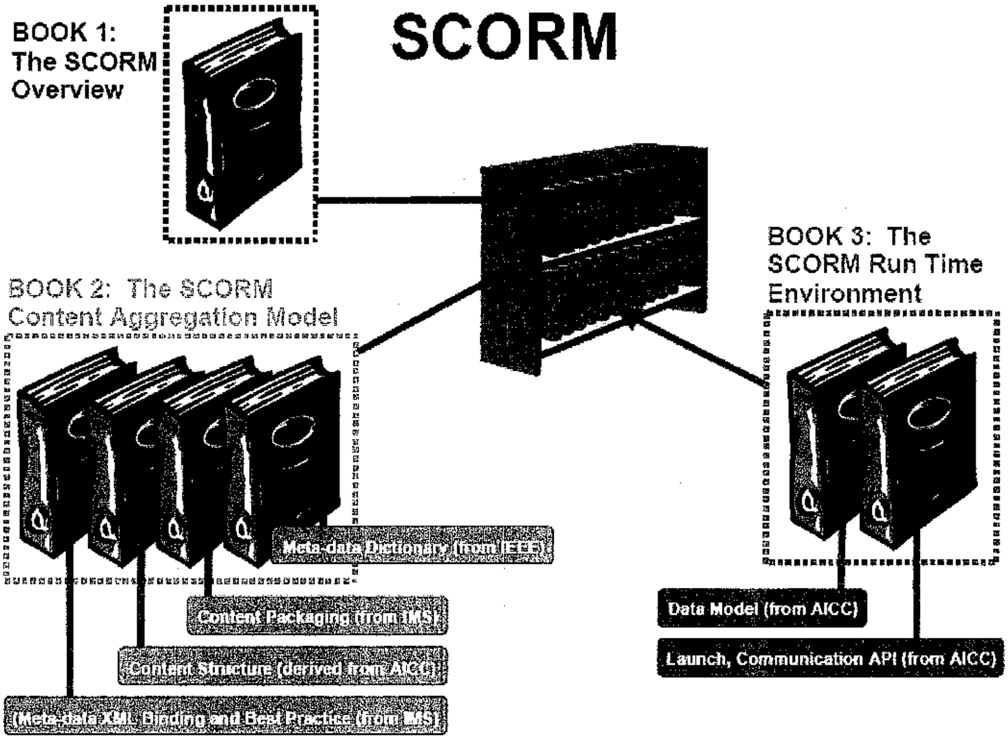

SCORM
BOOK 1:
The SCORM
Overview
BOOK 2: The SCORM
Content Aggregation Model
BOOK 3: The
SCORM Run Time
Environment
Meta-data Dictionary (from IEEE)
Content Packaging (from IMS)
Content Structure (derived from AICC)
Meta-data XML Binding and Best Practice (from IMS)
Data Model (from AICC)
Launch, Communication API (from AICC)

**FIGURE 4.2:** The SCORM as a collection specification

SCORM is a set of specifications that describes:

- How to create Web-based learning content that can be delivered and tracked by different SCORM-compliant learning management systems.
- What a SCORM-compliant learning management system must do in order to properly deliver and track SCORM-compliant learning content.

The SCORM specification is actually a set of specifications profiles based on various other industry standards and specifications. The SCORM specification does not cover all aspects of a learning enterprise; for example, it does not specify how tracking information is stored and what reports are generated, what pedagogical or training models should be used, or how learner information is compiled.

SCORM is a collection of specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reuse of Web-based learning content. A SCORM-compliant learning management system (LMS) is a system that:

- Can accept any SCORM compliant content as described and make it available to users of the LMS. SCORM does not specify how this is done, but conformance requires that a documented process exist to do it. Some LMS automate the process of publishing the content to make it available to the learners. Others require manual copying of files and possibly the manual tweaking of database tables.
- Deliver the content to a learner, by providing a runtime environment within which it launches the content. The content is always launched in a web browser window. The runtime environment must meet certain technical requirements defined by the SCORM specification and conformance requirements document.

The SCORM specification does not require that the LMS user interface behave in any particular way, except for the fact that the user must be able to choose and launch any SCO designed by an item in the content organization.

What a LMS does with the tracking information that may be communicated by a SCO is also not specified by SCORM. Obviously, often the reason to have a LMS in the first place is to collect this tracking information and use it for learning management or reporting purposes, but whether and how to do this is left to the LMS implementers and vendors to decide.

Note that although SCORM was designed primarily to support content delivery and tracking through a web server, it does not require the use of the internet or even of a network. It is possible to have a SCORM implementation that uses a closed intranet. You may even have purely local, offline implementation. For example, with an offline runtime environment, you should be able to play SCORM compliant content on a laptop or workstation with no network connection whatsoever. The offline runtime component may or may not include a "mini-LMS" capable of recording tracking information. Such an implementation may perhaps include synchronization with a server-based LMS when a mobile computer is reconnected to a network. How to do this is however beyond the scope of SCORM.

A Web-based learning object that can be included in a package for delivery by a SCORM compliant learning management system as an individual activity is called a Shareable Content Object (SCO). In practice, SCO come in two main flavors:

- A minimal SCO. This is HTML content or a service that can be launched in the browser window, and uses the SCORM Application Program Interface (API) for minimal communication with the learning management system. An LMS can track the time between the launch and the normal termination of such an object. Most generic content that can be launched in a browser window, but does not contain links to another learning object, can usually be turned into a SCO by "wrapping" it into a SCO. This could be an HTML page, an Adobe Acrobat file, or a text file. A collection of HTML pages is also acceptable, as long as they only link to each other and not to other learning objects.
- A data-enabled SCO. This is like a basic SCO, but also uses the SCORM API to get or send data to the LMS. The data may include tracking data, learner information.

In order to implement SCORM-compliant learning objects and SCOs it is useful to understand a little bit of the runtime environment in which they will be used. To distinguish this runtime service from the rest of the LMS and its login, authentication, storage and reporting services, we will refer to the runtime environment and the processes that manage it as the runtime service. The distinction between LMS and runtime service can be useful because an LMS and the user experience may have.

The runtime service can be implemented in part on the learner's computer and in the other part on the server side. In an offline situation, the server side of the environment is emulated by an application that runs on the learner's computer. The learner's computer of the runtime service:

- Is provided by the LMS;
- Is implemented as a Web page or frameset in a browser window;
- Presents some necessary runtime user interface components to the learner, such as a table of content and/or navigation buttons;
- Launches learning objects in a "stage window" that is either part of the frameset, or a separate window that it creates as needed;
- Includes an API object instance named "API" that can be found and called by JavaScript
- Is generated or regenerated as needed by the server side.

The server side of the runtime service and the rest of the learning management system are completely invisible to the scripts in SCORM learning objects. The user interface components of the runtime service, such as a table of content or navigation buttons, are also completely invisible to SCORM learning objects. How the learner's computer of the runtime service is implemented, and how it communicates with the server are totally invisible to the content. The only parts of the runtime service with which a SCORM learning object can detect and communicate are:

- The stage window
- The API adapter
- The functions provided by the API adapter

The learning objects are specifically prohibited from navigating to other learning objects within the stage window. Only the runtime service can load another learning object in the stage window. The runtime service uses the content organization defined in the content package as its guide to manage navigation between learning objects.

#### 4.1.1.1.5. Moodle

- *Moodle should run on the widest variety of platforms*

  The web application platform that runs on most platforms is PHP combined with MySQL, and this is the environment that Moodle has been developed.

- *Moodle should be easy to install, learn and modify*

  The PHP scripting language is easy to get into if you've done any programming using any other scripting language. PHP is also easy to install and is widely available to the point that most web hosting services provide it as standard.

- *It should be easy to upgrade from one version to the next*

  Moodle knows what version it is as well as the versions of all plug-in modules and a mechanism has been built-in so that Moodle can properly upgrade itself to new versions

- *It should be modular to allow for growth*

  Moodle has a number of features that are modular, including themes, activities, interface languages, database schemas and course formats. This allows anyone to add features to the main code base or to even distribute them separately.

## 4.2. Project Survey

A survey about MOODLE and content development of the project was conducted. Ten Universiti Teknologi Petronas (UTP) students, who already tested the project, participated. The survey basically covers whether the MOODLE and the project content development of the project are user-friendly or not, interactivity, functionality and usability. The survey results are shown as below:

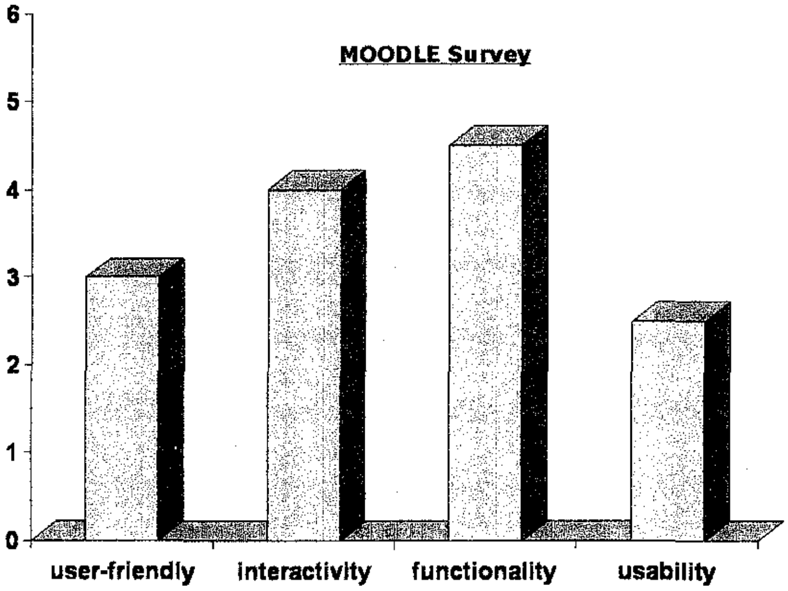


**Figure 4.3:** Survey done by student UTP for MOODLE

For average Moodle survey chart, participants for the survey are averaging 3.0 rating point to the level of user friendly of the Moodle program; while for interactivity of the Moodle, the participants' average rating is 4.0. For functionality and usability, the participants averaged 4.5 and 2.5 rating point respectively.

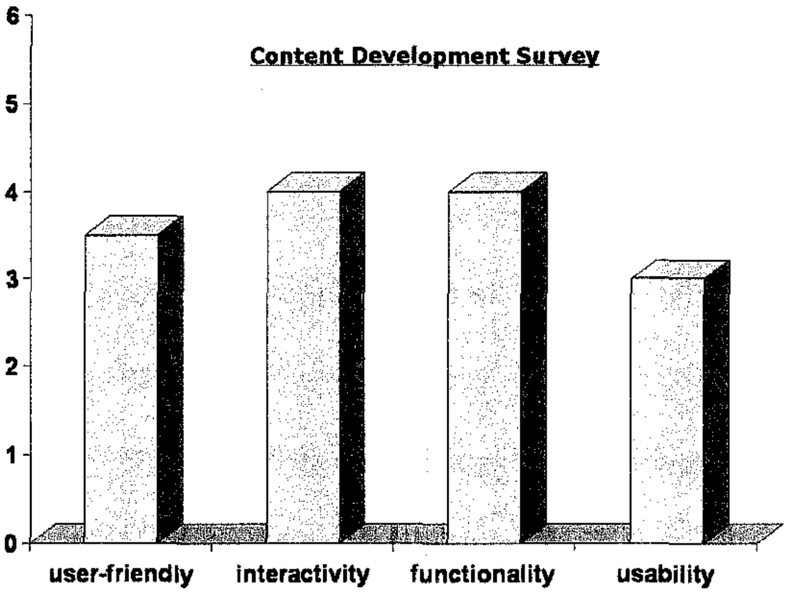


**Figure 4.4:** Survey done by student UTP for Content Development

For average content development survey chart, the participants scored an averaged 3.5 rating point on whether or not the content development is user friendly to them. A similar average of 4.0 rating point was calculated for interactivity and functionality of the content development. Lastly for usability, the participants rated an average 3.0 point.

# CHAPTER 5

# CONCLUSION

In conclusion, the main focus for this report is to develop a new version of e-learning by using open source software which is Moodle software. Proper comparisons with others open source software and also ongoing and very detailed development process and methodology executed in order to make sure that Moodle will serve its purpose as UTP new breed of e-learning. This report also did focus on content development for the E-Learning. SCORM is used as a standard for proper communication between LMS and content. All tools requirement are identified for development and implementation of Moodle and content development.

We've also covered regarding Sharable Content Objects can be summarized as follows:

- SCOs are mini-applications containing the content itself.
- SCOs are self-contained and transportable - all required files are included and compressed into a single content package.
- Content packages contain a single manifest file and may contain more than one SCO.
- SCOs are web-based and script-enabled - consisting of HTML, Javascript and any other client-side technology (images, Macromedia Flash, etc).
- SCOs are the smallest level of granularity tracked by an LMS - they report, among other things, only one score and status.

Lastly, we've examined the SCO and its general operation, the API and how it provides the communications portal between the LMS and the SCO and the manifest file and how it facilitates the importation of the SCO into the LMS. In other words, we've only scratched the surface. To fully understand SCORM and its functionality, you should download the full specification from the ADL's website and get familiar with it. Begin working with our sample content package and study as many other SCORM compliant SCOs as you can. The code is all client-side, so you will be able to open it with a basic text editor such as Notepad.

# REFERENCES

[1] Joan Van Duzer, Instructional Technologist College of Professional Studies Humboldt State University. A Comparison of Satisfaction with Online Teaching and Learning Tools.

[2] Bruce Perens, June, 1997, "Open Source Definition." (http://www.opensource.org/)

[3] *"Journal of* An Open Source software evaluation model with a case study on Course Management Systems*"*, Tilburg University, Karin van den Berg, August 2005

[4] Analysis Of RFP For e-Learning System, A software engineer perspective, By Huzaifa Al Nahas June 2003

[5] Anders Berggren[1], Daniel Burgos[2], Josep M. Fontana[3], Don Hinkelman[4], Vu Hung[5], Anthony Hursh[6], Ger Tielemans[7] *Moodle Community Learning Design Book Study Group*

[6] *Christos Bouras*, Computer Engineering and Informatics Department, University of Patras, GR-26500 Rion, Patras, GREECE, Research Academic Computer Technology Institute,

[7] Postion Statement from Singapore, Lim Kin Che, Chairman, Learning Standards Technical Committee, 5 Oct 2005

[8] Brent Lee And Frank Fucile From Malaspina University-College

[9] eLab Report, By Luca Botturi November 2004

[10] Firouz Anaraki Director, Network Operation Center College of Internet Distance Education Assumption University of Thailand, *Developing an Effective and Efficient eLearning Platform Using Open Source Software*

[11] The eLearning Guild™ Advisory Board Ruth Clark, Conrad Gottfredson, John Hartnett, Bill Horton, Kevin Moore, Eric Parks, Brenda Pfaus Marc Rosenberg, Allison Rossett

[12] *Short listing of Learning Management System software*, Part I of LMS Evaluation Open Source e-Learning Environment and Community Platform Project, Richard Wyles, 15 March 2004

[13] SumTotal Standards Projects, 2003-2005 by SumTotal Systems, Inc

[14] Severna Park, Maryland, USA © 2007 ICS Learning Group

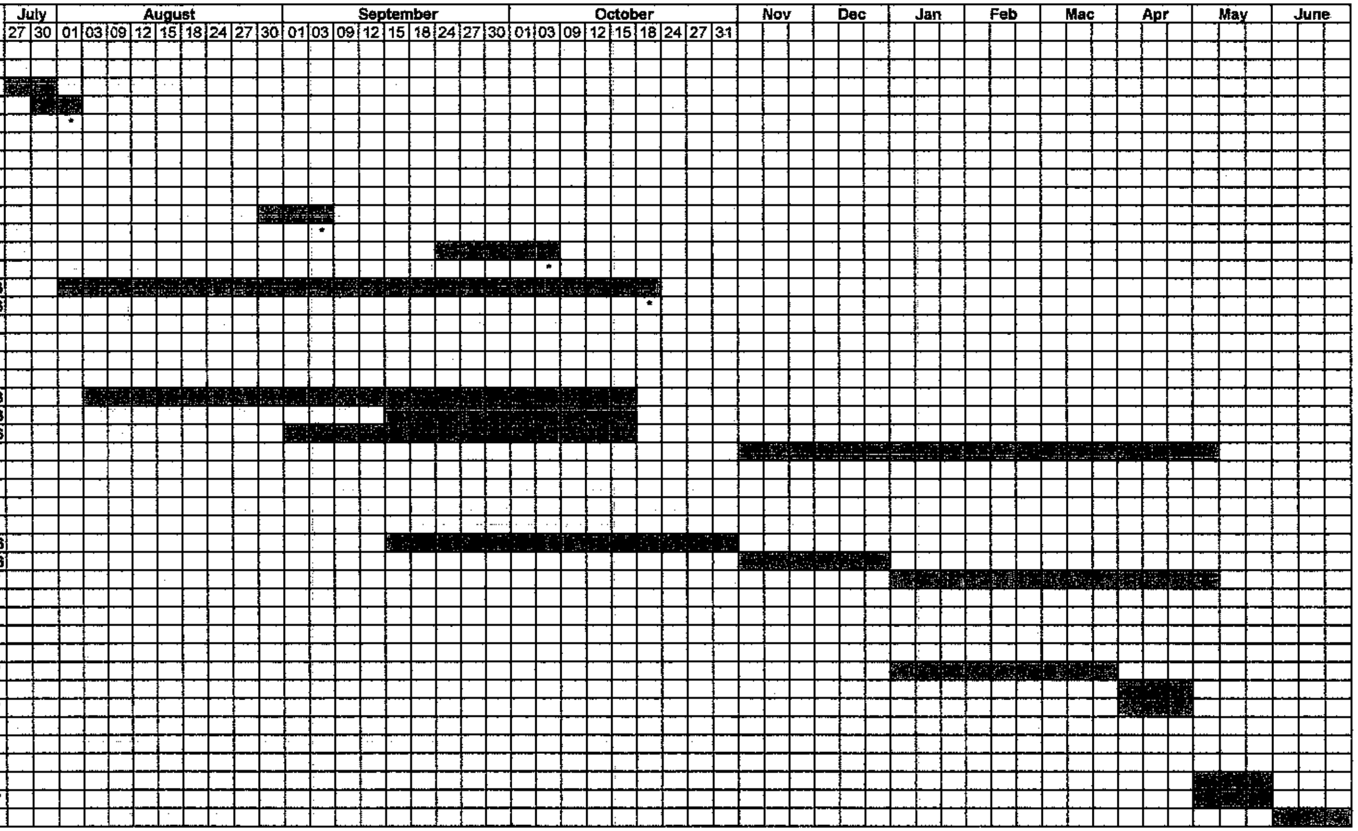
# APPENDIX A

## Project Timeline

Name : Mazlin Mohd Razali
ID : 4640

**Final Year Project - Universiti Teknologi PETRONAS (UTP) E-Learning implementation using open source and content development**
Project Timeline - 2006/2007

| Task | Duration | StartDate | EndDate |
|---|---|---|---|
| **Planning** | | | |
| Brainstorming | 4 days | 27/8/2006 | 30/8/2006 |
| Prepare Project Proposal | 2 days | 30/8/2006 | 1/8/2006 |
| Project Proposal Due | 0 days | 2/8/2006 | 2/8/2006 |
| **Documentations & Presentation** | | | |
| Preliminary Report | 7 days | 29/8/2006 | 3/9/2006 |
| Preliminary Report due | 0 days | 6/9/2006 | 6/9/2006 |
| Interim Report | 10 days | 24/9/2006 | 3/10/2006 |
| Interim Report due | 0 days | 4/10/2006 | 4/10/2006 |
| Prepare Weekly Report | 84 days | 2/8/2006 | 20/10/2006 |
| Oral Presentation Phase 1 | 3 days | 16/10/2006 | 19/10/2006 |
| **Research** | | | |
| Open Source System | 72 days | 3/8/2006 | 15/10/2006 |
| MOODLE | 30 days | 1/9/2006 | 15/10/2006 |
| Content Development | 45 days | 1/9/2006 | 15/10/2006 |
| SCORM | 180 days | 1/11/2006 | 30/4/2007 |
| **Development** | | | |
| User Interface | 45 days | 15/9/2006 | 31/10/2006 |
| System Design | 60 days | 1/11/2006 | 31/12/2006 |
| System Development | 120 days | 1/1/2007 | 30/4/2007 |
| **Testing** | | | |
| Unit Testing | 90 days | 1/1/2007 | 31/3/2007 |
| System Integration Testing | 7 days | 1/4/2007 | 7/4/2007 |
| User Acceptance Testing | 7 days | 8/4/2007 | 15/4/2007 |
| **Final Report & Presenation** | | | |
| Final  Report Compilation | 30 days | 1/5/2007 | 31/5/2007 |
| Final Oral Presentation | 0 days | 1/6/2007 | 1/6/2007 |



# APPENDIX B

# API Wrapper Source Code

```
/*******************************************************************************
** FileName: APIWrapper.js
** This file is part of the ADL Sample API Implementation intended to provide
** an elementary example of the concepts presented in the ADL Sharable
** Content Object Reference Model (SCORM).
**
** The purpose in wrapping the calls to the API is to (1) provide a
** consistent means of finding the LMS API implementation within the window
** hierarchy and (2) to validate that the data being exchanged via the
** API conforms to the defined CMI data types.
**
** This is just one possible example for implementing the API guidelines for
** runtime communication between an LMS and executable content components.
** There are several other possible implementations.
**
** Usage: Executable course content can call the API Wrapper
**      functions as follows:
**
**    javascript:
**         var result = doLMSInitialize();
**         if (result != true)
**         {
**           // handle error
**         }
**
**    authorware
**         result := ReadURL("javascript:doLMSInitialize()", 100)
**
*******************************************************************************/
var _Debug = false;  // set this to false to turn debugging off
                // and get rid of those annoying alert boxes.

// Define exception/error codes
var _NoError = 0;
var _GeneralException = 101;
var _ServerBusy = 102;
var _InvalidArgumentError = 201;
var _ElementCannotHaveChildren = 202;
var _ElementIsNotAnArray = 203;
var _NotInitialized = 301;
var _NotImplementedError = 401;
var _InvalidSetValue = 402;
var _ElementIsReadOnly = 403;
var _ElementIsWriteOnly = 404;
var _IncorrectDataType = 405;


// local variable definitions
var apiHandle = null;
var API = null;
var findAPITries = 0;

/*******************************************************************************
** Function: doLMSInitialize()
** Inputs:  None
** Return:  CMIBoolean true if the initialization was successful, or
**          CMIBoolean false if the initialization failed.
**
** Description:
** Initialize communication with LMS by calling the LMSInitialize
** function which will be implemented by the LMS.
*******************************************************************************/
function doLMSInitialize()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSInitialize was not successful.");
    return "false";
  }
  var result = api.LMSInitialize("");

  if (result.toString() != "true")
  {
    var err = ErrorHandler();
  }

  return result.toString();
}
/*******************************************************************************
** Function doLMSFinish()
** Inputs:  None
** Return:  CMIBoolean true if successful
**          CMIBoolean false if failed.
**
** Description:
** Close communication with LMS by calling the LMSFinish
** function which will be implemented by the LMS
*******************************************************************************/
function doLMSFinish()
{

  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSFinish was not successful.");
    return "false";
  }
  else
  {
    // call the LMSFinish function that should be implemented by the API

    var result = api.LMSFinish("");
    if (result.toString() != "true")
    {
      var err = ErrorHandler();
    }

  }

  return result.toString();
}
/*******************************************************************************
** Function doLMSGetValue(name)
** Inputs:  name - string representing the cmi data model defined category or
**             element (e.g. cmi.core.student_id)
** Return:  The value presently assigned by the LMS to the cmi data model
**       element defined by the element or category identified by the name
**       input value.
**
** Description:
** Wraps the call to the LMS LMSGetValue method
*******************************************************************************/
function doLMSGetValue(name)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSGetValue was not successful.");
    return "";
  }
  else
  {
    var value = api.LMSGetValue(name);
    var errCode = api.LMSGetLastError().toString();
    if (errCode != _NoError)
    {
      // an error was encountered so display the error description
      var errDescription = api.LMSGetErrorString(errCode);
      alert("LMSGetValue("+name+") failed. \n"+ errDescription);
      return "";
    }
    else
    {

      return value.toString();
    }
  }
}
/*******************************************************************************
** Function doLMSSetValue(name, value)
** Inputs:  name -string representing the data model defined category or element
**          value -the value that the named element or category will be assigned
** Return:  CMIBoolean true if successful
**          CMIBoolean false if failed.
**
** Description:
** Wraps the call to the LMS LMSSetValue function
*******************************************************************************/
function doLMSSetValue(name, value)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSSetValue was not successful.");
    return;
  }
  else
  {
    var result = api.LMSSetValue(name, value);
    if (result.toString() != "true")
    {
      var err = ErrorHandler();
    }
  }

  return;
}
/*******************************************************************************
** Function doLMSCommit()
** Inputs:  None
** Return:  None
**
** Description:
** Call the LMSCommit function
*******************************************************************************/
function doLMSCommit()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSCommit was not successful.");
    return "false";
  }
  else
  {
    var result = api.LMSCommit("");
    if (result != "true")
    {
      var err = ErrorHandler();
    }
  }

  return result.toString();
}
/*******************************************************************************
** Function doLMSGetLastError()
** Inputs:  None
** Return:  The error code that was set by the last LMS function call
```

```
**
** Description:
** Call the LMSGetLastError function
*******************************************************************************/
function doLMSGetLastError()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSGetLastError was not successful.");
    //since we can't get the error code from the LMS, return a general error
    return _GeneralError;
  }

  return api.LMSGetLastError().toString();
}
/*******************************************************************************
** Function doLMSGetErrorString(errorCode)
** Inputs:  errorCode - Error Code
** Return:  The textual description that corresponds to the input error code
**
** Description:
** Call the LMSGetErrorString function
*******************************************************************************/
function doLMSGetErrorString(errorCode)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSGetErrorString was not successful.");
  }

  return api.LMSGetErrorString(errorCode).toString();
}
/*******************************************************************************
** Function doLMSGetDiagnostic(errorCode)
** Inputs:  errorCode - Error Code(integer format), or null
** Return:  The vendor specific textual description that corresponds to the
**          input error code
**
** Description:
** Call the LMSGetDiagnostic function
*******************************************************************************/
function doLMSGetDiagnostic(errorCode)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSGetDiagnostic was not successful.");
  }

  return api.LMSGetDiagnostic(errorCode).toString();
}
/*******************************************************************************
** Function LMSIsInitialized()
** Inputs:  none
** Return:  true if the LMS API is currently initialized, otherwise false
**
** Description:
** Determines if the LMS API is currently initialized or not.
*******************************************************************************/
function LMSIsInitialized()
{
  // there is no direct method for determining if the LMS API is initialized
  // for example an LMSIsInitialized function defined on the API so we'll try
  // a simple LMSGetValue and trap for the LMS Not Initialized Error

  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nLMSIsInitialized() failed.");
    return false;
  }
  else
  {
    var value = api.LMSGetValue("cmi.core.student_name");
    var errCode = api.LMSGetLastError().toString();
    if (errCode == _NotInitialized)
    {
      return false;
    }
    else
    {
      return true;
    }
  }
}
/*****************************************************************************
** Function ErrorHandler()
** Inputs:  None
** Return:  The current value of the LMS Error Code
**
** Description:
** Determines if an error was encountered by the previous API call
** and if so, displays a message to the user.  If the error code
** has associated text it is also displayed.
*****************************************************************************/
function ErrorHandler()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Unable to locate the LMS's API Implementation.\nCannot determine LMS error code.");
    return;
  }

  // check for errors caused by or from the LMS
  var errCode = api.LMSGetLastError().toString();
  if (errCode != _NoError)
  {
    // an error was encountered so display the error description
    var errDescription = api.LMSGetErrorString(errCode);

    if (_Debug == true)
    {
      errDescription += "\n";
      errDescription += api.LMSGetDiagnostic(null);
      // by passing null to LMSGetDiagnostic, we get any available diagnostics
      // on the previous error.
    }

    alert(errDescription);
  }

  return errCode;
}
/*****************************************************************************
** Function getAPIHandle()
** Inputs:  None
** Return:  value contained by APIHandle
** Description:
** Returns the handle to API object if it was previously set,
** otherwise it returns null
*****************************************************************************/
function getAPIHandle()
{
  if (apiHandle == null)
  {
    apiHandle = getAPI();
  }
  return apiHandle;
}
/*****************************************************************************
** Function findAPI(win)
** Inputs:  win - a Window Object
** Return:  If an API object is found, it's returned, otherwise null is returned
**
** Description:
** This function looks for an object named API in parent and opener windows
*****************************************************************************/
function findAPI(win)
{
  while ((win.API == null) && (win.parent != null) && (win.parent != win))
  {
    findAPITries++;
    // Note: 7 is an arbitrary number, but should be more than sufficient
    if (findAPITries > 7)
    {
      alert("Error finding API -- too deeply nested.");
      return null;
    }

    win = win.parent;

  }
  return win.API;
}
/*****************************************************************************
** Function getAPI()
** Inputs:  none
** Return:  If an API object is found, it's returned, otherwise null is returned
**
** Description:
** This function looks for an object named API, first in the current window's
** frame hierarchy and then, if necessary, in the current window's opener window
** hierarchy (if there is an opener window).
*****************************************************************************/
function getAPI()
{
  var theAPI = findAPI(window);
  if ((theAPI == null) && (window.opener != null) && (typeof(window.opener) != "undefined"))
  {
    theAPI = findAPI(window.opener);
  }
  if (theAPI == null)
  {
    alert("Unable to find an API adapter");
  }
  return theAPI
}
```

# APPENDIX C

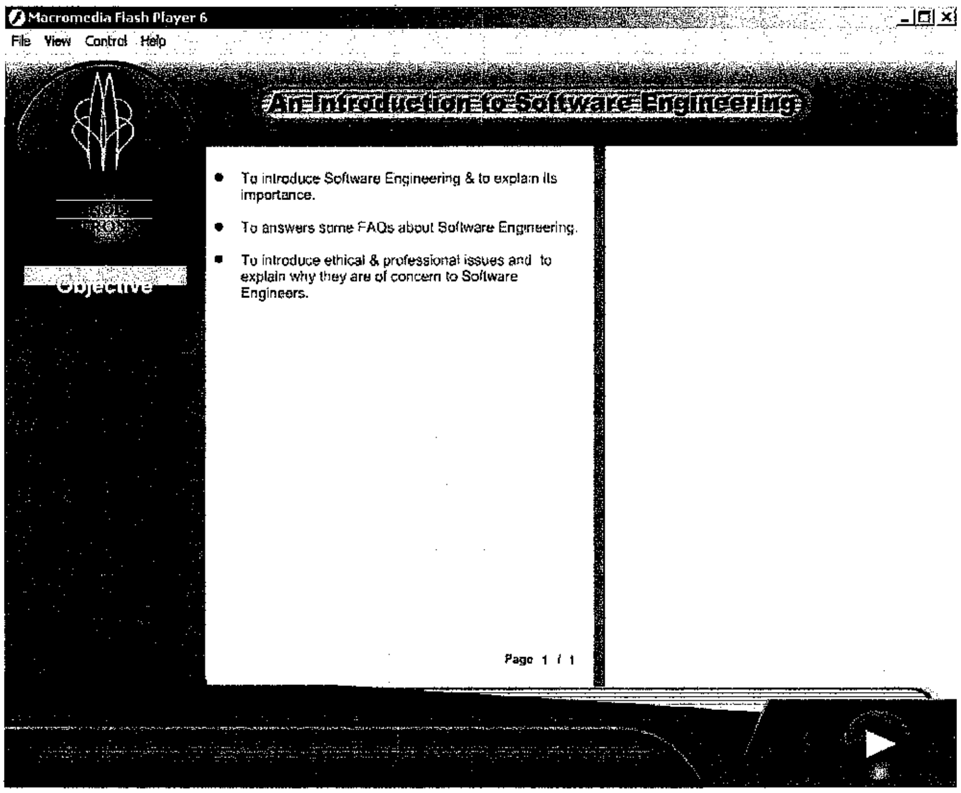
## ScoFunctions Source Code

```
<html>
<head>
<script type="text/javascript" language="JavaScript">

var API = null; /* SCORM API */

/* look up through the frameset hierarchy for the SCORM API */
function findAPI(win)
{
        while ((win.API == null) && (win.parent != null) && (win.parent != win))
        {
                win = win.parent;
        }
        API = win.API;
}

/* init the SCORM API */
function initAPI(win)
{
        /* look for the SCORM API up in the frameset */
        findAPI(win);

        /* if we still have not found the API, look at the opener and it's frameset */
        if ((API == null) && (win.opener != null))
        {
                findAPI(win.opener);
        }
}
</script>
<title></title>
</head>

<body onLoad="init()" onUnload="finish()">

</body>
</html>
```

# APPENDIX D

## Screenshot

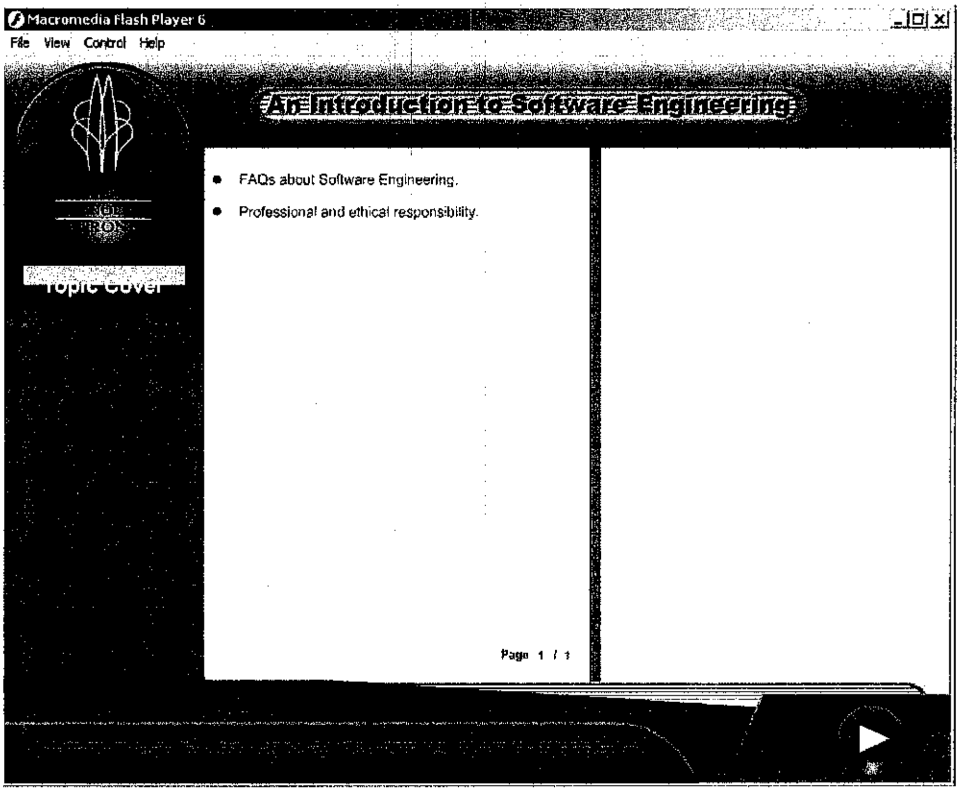
## Chapter 1 – Software Engineering



Macromedia Flash Player 6
File View Control Help
An Introduction to Software Engineering
Objective

- To introduce Software Engineering & to explain its importance.
- To answers some FAQs about Software Engineering.
- To introduce ethical & professional issues and to explain why they are of concern to Software Engineers.

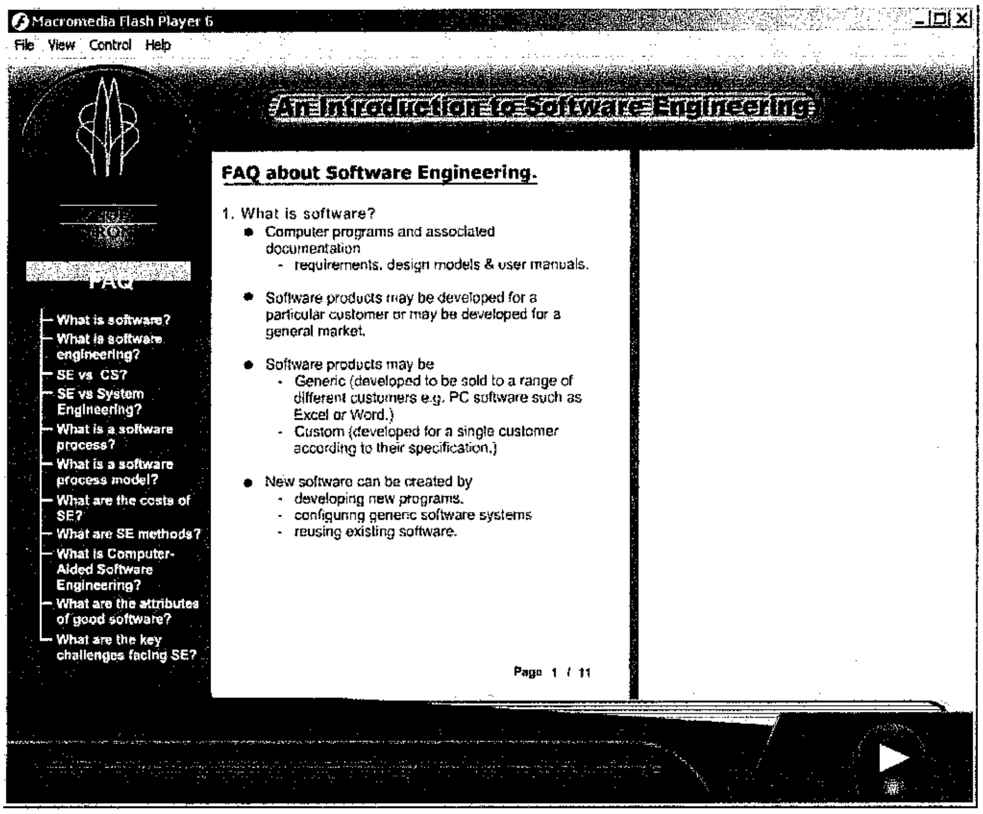
**Objective Page**



Macromedia Flash Player 6
File View Control Help
An Introduction to Software Engineering
Topic Cover

- FAQs about Software Engineering.
- Professional and ethical responsibility.

**Topic Cover**

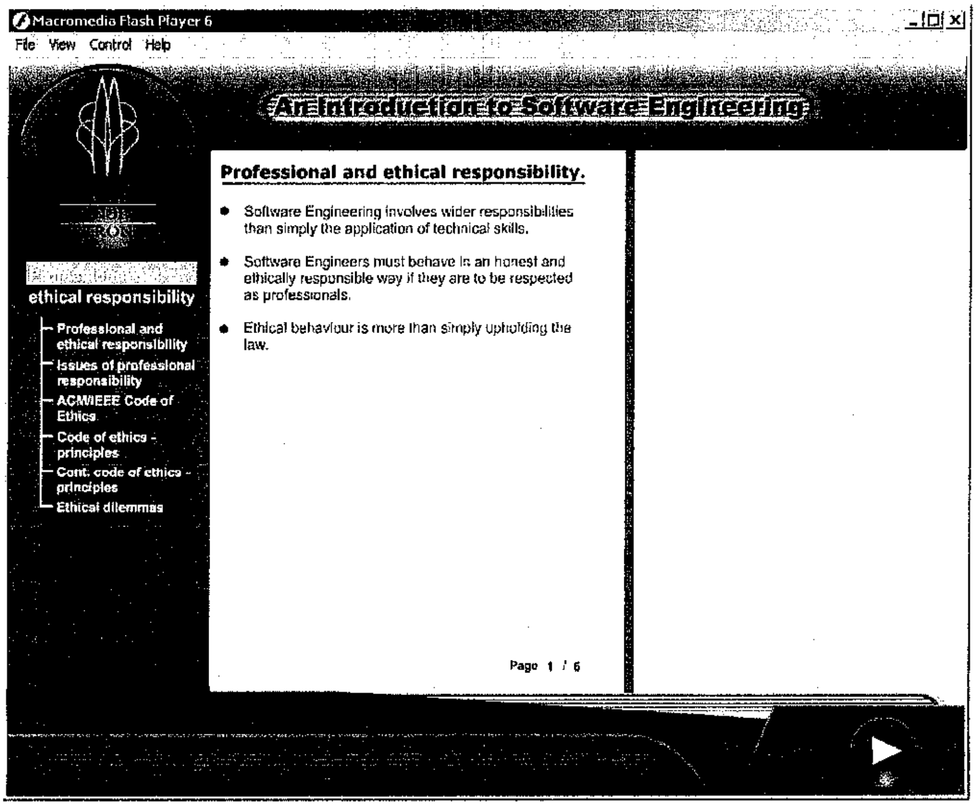

Macromedia Flash Player 6
File View Control Help
An Introduction to Software Engineering
FAQ
What is software?
What is software engineering?
SE vs CS?
SE vs System Engineering?
What is a software process?
What is a software process model?
What are the costs of SE?
What are SE methods?
What is Computer-Aided Software Engineering?
What are the attributes of good software?
What are the key challenges facing SE?

**FAQ about Software Engineering.**

1. What is software?
   - Computer programs and associated documentation
     - requirements, design models & user manuals.
   - Software products may be developed for a particular customer or may be developed for a general market.
   - Software products may be
     - Generic (developed to be sold to a range of different customers e.g. PC software such as Excel or Word.)
     - Custom (developed for a single customer according to their specification.)
   - New software can be created by
     - developing new programs.
     - configuring generic software systems
     - reusing existing software.

Page 1 / 11

## FAQ about Software Engineering

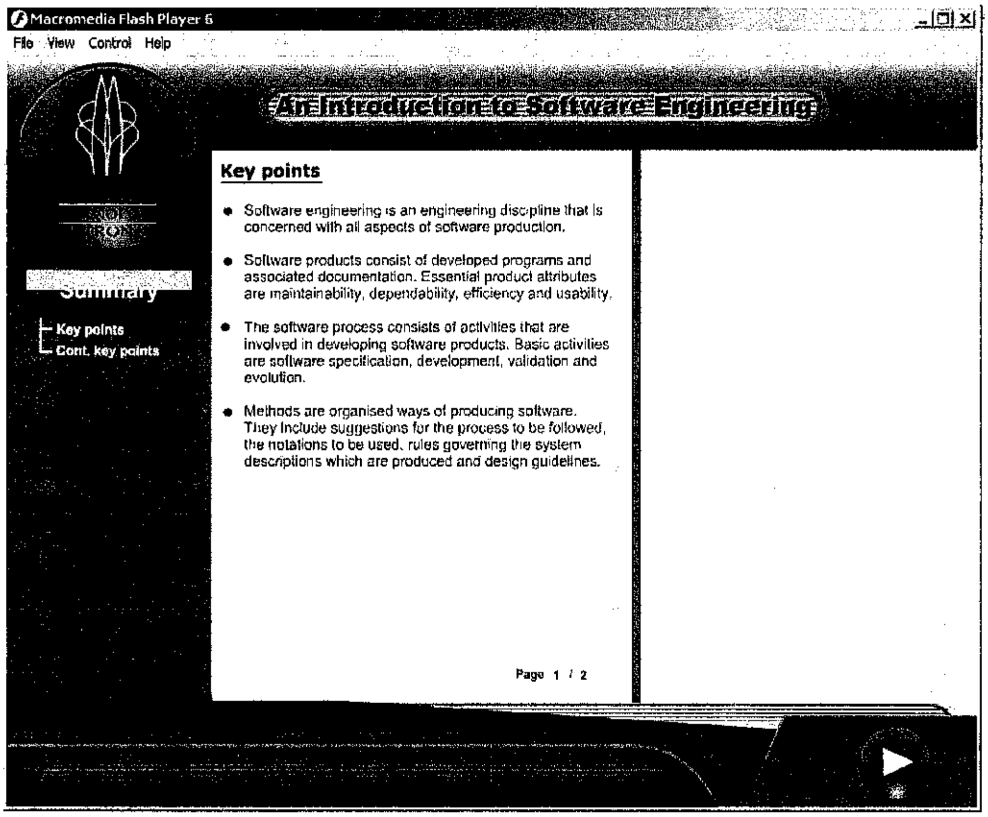

Macromedia Flash Player 6
File View Control Help
An Introduction to Software Engineering
ethical responsibility
Professional and ethical responsibility
Issues of professional responsibility
ACM/IEEE Code of Ethics
Code of ethics - principles
Cont. code of ethics - principles
Ethical dilemmas

**Professional and ethical responsibility.**

- Software Engineering involves wider responsibilities than simply the application of technical skills.
- Software Engineers must behave in an honest and ethically responsible way if they are to be respected as professionals.
- Ethical behaviour is more than simply upholding the law.

Page 1 / 6

## Professional and Ethical Responsibility



Macromedia Flash Player 6
File View Control Help
An Introduction to Software Engineering
Summary
Key points
Cont. key points

## Key points

- Software engineering is an engineering discipline that is concerned with all aspects of software production.
- Software products consist of developed programs and associated documentation. Essential product attributes are maintainability, dependability, efficiency and usability.
- The software process consists of activities that are involved in developing software products. Basic activities are software specification, development, validation and evolution.
- Methods are organised ways of producing software. They include suggestions for the process to be followed, the notations to be used, rules governing the system descriptions which are produced and design guidelines.

Page 1 / 2

**Summary**