

國立交通大學

資訊學院 資訊學程

碩士論文

以 CMMI 為基礎的多媒體學習內容

開發流程品質管制方法

A CMMI-Based Quality Control Method

for Multimedia Learning Contents Developing Process

研究生：連瑞斌

指導教授：陳登吉 教授

中華民國九十六年七月

以 CMMI 為基礎的多媒體學習內容
開發流程品質管制方法

A CMM-Based Quality Control Method
for Multimedia Learning Contents Developing Process

研 究 生：連瑞斌

Student：Rui-Bin Lien

指 導 教 授：陳登吉

Advisor：Deng-Jyi Chen

國 立 交 通 大 學



Submitted to College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

in

Computer Science

July 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年七月

以 CMMI 為基礎的多媒體學習內容 開發流程品質管制方法

學生：連瑞斌

指導教授：陳登吉 博士

國立交通大學資訊學院 資訊學程碩士班

摘 要

由於網際網路以及通訊技術的成熟，使得多媒體內容可在一般的電腦自由流通及呈現，同時使得多媒體教材市場的需求量大為增加。如何運用低成本及高效率的方式來設計出高品質的多媒體教材，是當前學界及業界所需面對的重要議題。為了提昇多媒體教材的品質水準，首先必須了解造成品質問題的相關因素，並針對該因素建立一套有效的品質管制機制。在本研究中發現，造成品質問題的原因有二：(1)來自於多媒體教材本身內容腳本設計是否達到學習者的學習標的以及內容呈現效果的吸引力；(2)來自於多媒體教材開發過程當中，往往會因為內部或外部顧客的要求，而不斷的變更原始設計所造成的。其中在教材開發過程中由於缺少有效的控管措施而造成的品質不良。

由於目前多媒體教材設計皆遵循國際性通用標準如SCORM規範來開發，並無法保證可製作出高品質的多媒體教材。因此本研究以卡內基美隆大學的軟體工程研究所提出，在國際上具有學界及業界公信力的軟體品質標準能力成熟度整合模式CMMI為基礎，針對多媒體教材開發流程，提出“多媒體教材品質管制方法”。此品質管制方法可提供多媒體教材開發過程“不一致性偵測”機制，以協助多媒體教材在協同設計環境下，避免因為變更設計而產生不一致性的品質問題。最後藉由Client-Server架構，設計一個多媒體教材品質管制系統，來顯示本研究提出的概念具有可行性及適用性。

關鍵字：能力成熟度整合模式、軟體工程、品質管制、數位學習

A CMMI-Based Quality Control Method for Multimedia Learning Contents Developing Process

Student : Rui-Bin Lien Advisor : Dr. Deng-Jyi Chen

Degree Program of College of Computer Science
National Chiao Tung University

Abstract

The third industrial revolution in the twenty-first-century, led by the internet and information telecommunication technology, has made the multimedia learning contents (MLCs) creators and learners using multimedia for presentation in a major trend in education and for both the academic community and industrial sectors, researchers are seeking for better methodologies to design high-quality MLCs. To improve the quality level of MLCs, we need to understand the related factors and then design an efficient quality control mechanism. This thesis study addresses two quality issues: (1) whether the script design of the MLCs fulfills the learning objective, and whether it is attractive enough; (2) in the MLCs developing process, various changes of original requirements or designs may be continuously requested from either inner or exterior customers; if there is a lack of effective control and management mechanisms, there may be causing quality problems.

Currently, although the design of MLCs follows international standards such as the SCORM, but this still does not guarantee the high-quality of MLCs. Therefore, this research employs the CMMI developed by the software engineering institute (SEI) of Carnegie Mellon University, to develop a quality control method for MLCs creators. This MLCs quality control (MQC) method includes an inconsistent detection mechanism in the MLCs developing process, so as to avoid quality problems due to continuous changes in design under a collaboration environment. The feasibility and compatibility of the proposed methodology will be demonstrated through a client-server system implementation that integrated into MQC system.

Keywords: Capability Maturity Model Integrated (CMMI), Software Engineering, Quality Control, e-Learning.

誌 謝

在交通大學兩年的研究所求學生涯中，承蒙恩師陳登吉教授於悉心指導及與諄諄教誨，使本論文研究得以順利完成。陳教授是一位認真負責的傑出教授，在學術上的專業涵養及平日的待人處事都讓學生獲益良多，在平日繁忙的時間裡，陳教授不辭辛勞的在研究方法與作學問的態度給予指導，在生活應對與待人處世的態度上亦諄諄教誨，在此獻上最誠摯的感謝。

此外感謝本論文的口試委員游寶達教授、洪茂盛教授、孔崇旭教授在論文審查期間，對本論文所提出的寶貴建議及修正意見，使整體論文更加充實，在此亦表達最誠摯的感謝。在求學的這段期間，特別感謝的蔡明志學長所給予的建議與協助，也恭喜學長獲得博士學位。此外感謝鍾貴榮學長、同窗林賢忠、學弟翁浚恩、謝佳成以及實驗室學弟妹們，無論在生活上以及課業上所提供的許多寶貴意見，讓我能順利完成學業。

最後特別感謝我的父母與親愛的家人，因為有你們在背後的支持、栽培與鼓勵，使我得以專心在研究的路上繼續鑽研。在此謹以此篇論文獻給所有關心我的長官、老師、家人以及所有曾經幫助我的人，願與他們一同分享這份喜悅與榮耀。



Contents

ABSTRACT (CHINESE)	i
ABSTRACT	ii
ACKNOWLEDGEMENT (CHINESE)	iii
CONTENTS	iv
LIST OF FIGURES	v
LIST OF TABLES	vii
CHAPTER 1 INTRODUCTION	1
1.1 GENERAL REVIEW	1
1.2 BACKGROUND	2
1.3 PROPOSED APPROACH	4
1.4 MOTIVATION	6
1.5 OBJECTIVE	6
1.6 THESIS ORGANIZATION	7
CHAPTER 2 RELATED WORK	7
2.1 SHARABLE CONTENT OBJECT REFERENCE MODEL	8
2.2 MULTIMEDIA LEARNING OBJECT	13
2.3 CAPABILITY MATURITY MODEL INTEGRATION	15
2.4 CAUSES OF INCONSISTENCIES	21
2.5 DETECTING AND IDENTIFYING INCONSISTENCIES	23
2.6 HANDLING INCONSISTENCIES	24
CHAPTER 3 DESIGN METHODOLOGY	25
3.1 MLCs PROCESS LIFE CYCLE	25
3.2 MLCs BIDIRECTIONAL TRACEABILITY MATRIX	27
3.3 MLCs DIRECTED ACYCLIC GRAPH	32
3.4 INCONSISTENCY DETECTION CAPABILITY DESIGN	34
CHAPTER 4 SYSTEM IMPLEMENTATION	37
4.1 SYSTEM DEVELOPMENT TOOLS	37
4.2 SOFTWARE ARCHITECTURE	38
4.3 MLCs DEPENDENCY EDITOR	40
4.4 MLCs CHANGE DETECTOR	41
4.5 MLCs INCONSISTENT DETECTOR	42
CHAPTER 5 AN USAGE EXAMPLE FOR MQC SYSTEM	46
CHAPTER 6 CONCLUSIONS AND FUTURE WORK	59
6.1 CONCLUSIONS	59
6.2 FUTURE WORK	60
REFERENCE	61

List of Figures

Figure 1.1 Activities by percentage of total development staff effort.	4
Figure 1.2 Requirements management (REQM) context diagram	5
Figure 2.1 File types of Assets	10
Figure 2.2 Example of Content Aggregation (CA)	11
Figure 2.3 Content Package (CP) conceptual diagram	12
Figure 2.5 The frameworks quagmire	15
Figure 2.6 CMMI History	16
Figure 2.7 CMMI derivation	17
Figure 2.8 CMMI maturity framework	18
Figure 2.9 CMMI stage representation	20
Figure 3.1 MLCs process flow chart	26
Figure 3.2 MLCs directed acyclic graphic (MDAG)	32
Figure 4.1 System architecture	38
Figure 4.2 Functional model of the MQC system	39
Figure 4.3 The screen shot of MLCs dependency editor (MDE)	41
Figure 4.4 The state diagram of MLCs change detector (MCD)	42
Figure 4.5 A MDAG of MLCs inconsistency detector (MID)	44
Figure 4.6 The change impact patterns of MID	44
Figure 4.7 The screen shot of MLCs inconsistency detector (MID)	45
Figure 5.1 Example of original MLCs material	47
Figure 5.2 Selecting a section of original teaching material	47
Figure 5.3 Analyzing a section of original teaching material	48
Figure 5.4 Example of the storyboard in primary draft design phase. (Part 1)	49
Figure 5.5 Example of the storyboard in primary draft design phase. (Part 2)	49
Figure 5.6 Example of the storyboard in primary draft design phase. (Part 3)	50
Figure 5.7 Example of the storyboard in primary draft design phase. (Part 4)	50
Figure 5.8 Example of the storyboard in secondly detail design phase. (Part 1)	51
Figure 5.9 Example of the storyboard in secondly detail design phase. (Part 2)	51
Figure 5.10 Example of the storyboard in secondly detail design phase. (Part 3)	52
Figure 5.11 Example of the UI layout in secondly detail design phase. (Part 1)	52
Figure 5.12: Example of the multimedia actor list in secondly detail design phase	53
Figure 5.13 The screen shot of a multimedia learning object “What is on the mat?”	53
Figure 5.14 The screen shot of MQC system “Login” module	54
Figure 5.15 The screen shot of MQC system “Create” module	54

Figure 5.16 The screen shot of MQC system “Edit” module	56
Figure 5.17 The screen shot of MQC system “Modify” module	57
Figure 5.18 The MLCs directed acyclic graph (MDAG) of a multimedia learning object “What is on the mat?”	57
Figure 5.19 The screen shot of multimedia actors	58
Figure 5.20 The screen shot of a multimedia authoring tool	58
Figure 5.21 The screen shot of multimedia actor list report	59
Figure 5.22 The screen shot of inconsistency report	59



List of Tables

Table 1.1 The cost of correcting software errors	3
Table 3.1 Phases and activities in MLCs development process	27
Table 3.2 An example of storyboard	30
Table 3.3 MLCs forward dependency matrix (MFDM)	31
Table 3.4 MLCs backward dependency matrix (MBDM)	32



Chapter 1

Introduction

1.1 General Review

The third industrial revolution in the twenty-first-century, led by the internet and information telecommunication technology, has made the multimedia learning contents (MLCs) creators and learners using multimedia for presentation in a major trend in education and for both the academic community and industrial sectors, researchers are seeking for better methodologies to design high-quality MLCs. To improve the quality level of MLCs, we need to understand the related factors and then design an efficient quality control mechanism. This thesis study addresses two quality issues: 1) whether the script design of the MLCs fulfills the learning objective, and whether it is attractive enough; 2) in the MLCs developing process, various changes of original requirements or designs may be continuously requested from either inner or exterior customers; if there is a lack of effective control and management mechanisms, there may be causing quality problems. [24]

Currently, although the design of MLCs follows international standards such as the SCORM, but this still does not guarantee the high-quality of MLCs.[20][21][22][23] To develop high quality multimedia learning contents, two quality issues are considered

1. The quality of the learning contents itself: The quality of the domain knowledge of learning contents itself relies mainly upon the creativity of content creators, such as interesting materials (attractive stories and scripts) and vivid presentation (art and music design). This aspect depends on the subjective judgments from readers to readers, instead of technological developments.

2. The quality in the learning contents developing process: This relies upon proper quality control mechanisms. This issue can therefore be evaluated using some objective metrics, and it is the main issue in this research.

In order to interact with users and integrate learning contents and multimedia elements (text, pictures, animations, videos, and music), a digital software format is most fitting for the container of modern multimedia learning contents. This has the multiple advantages of interactivity, exchangeability, portable ability and reusability, but also retains the traditional weaknesses of software design. The digital software format requires one or many digital editing software tools during the developing process, so traditional quality problems exist. These quality problems are the main object of "system engineering" [16].

Multimedia learning contents have the characteristics of dual professional realms: "education" and "software." As already mentioned, the SCORM standard does not guarantee a high quality product. The reason is that the SCORM is purely a multimedia learning contents aggregation design standard, lacking the developing process and quality requirements of developing digital learning contents. Therefore, to develop high quality multimedia learning contents, international digital learning contents standards such as the SCORM are not sufficient. The theoretical basis and process standards of "system engineering" also need to be fulfilled.

1.2 Background

At present, in the software engineering field, the CMMI developed by the Software Engineering Institute (SEI) of Carnegie-Mellon University is widely adopted. According to the CMMI, it is known that under a collaboration process of software development, lack of a stable developing process and good developing standards will cause a problem

in the final integration phase of the developing process, when different software modules designed by different team members are found to be in conflict. This will result in delay of the due date, rise of developing costs, complaints of customer, and low morale of the developing team [7][8]. These problems in traditional software developing processes are also prone to occur in the developing process of multimedia learning contents.

Table 1.1: The cost of correcting software errors
(Source: Hughes department of defense software error history)

Software Development Phase	% of Development Funds	Errors Introduced (%)	Errors Found (%)	Relative Cost to Correct
Requirements Analysis	5	55	18	1.0
Design	25	30	10	1.0-5.0
Code & Test	10			
Integration Test	50	10	50	1.0-5.0
Validation and Documentation	10			
Operational Maintenance		5	22	10-100

According to a research based on 13,000 IT case studies in 2003, 15% are found to be complete failures, 51% face serious challenges, and only 34% projects are successful [4]. This explains why multimedia learning contents developed according to a software format face so many complaints from customers relating to quality problems. As the research of R.C. Lee and W.M. Tepfenhart shows (Table 1.1) [4], 55% of mistakes come from the requirement analysis phase in the developing process. Therefore, if mistakes can be avoided in this phase, then the lowest cost for revision is only 20% of the design phase, or 1% of the maintenance phase. Hence, it is established that using proper software engineering technologies during the requirements analysis phase can efficiently decrease mistakes in the multimedia learning contents and lower developing costs.

Furthermore, a software developing methodology that is developed by the Software Engineering Laboratory (SEL) of National Aeronautics and Space Administration (NASA) argues that the requirements analysis phase exists in any phase of the software

developing life cycle in a gradually decreasing state [6], as Figure 1.1 shows. This is different from traditional software developing models, such as the Water-fall model [14]. This explains why, in the multimedia learning contents developing process, changes made to meet the needs of inner or exterior customers will result in inconsistency quality problem of multimedia learning contents.

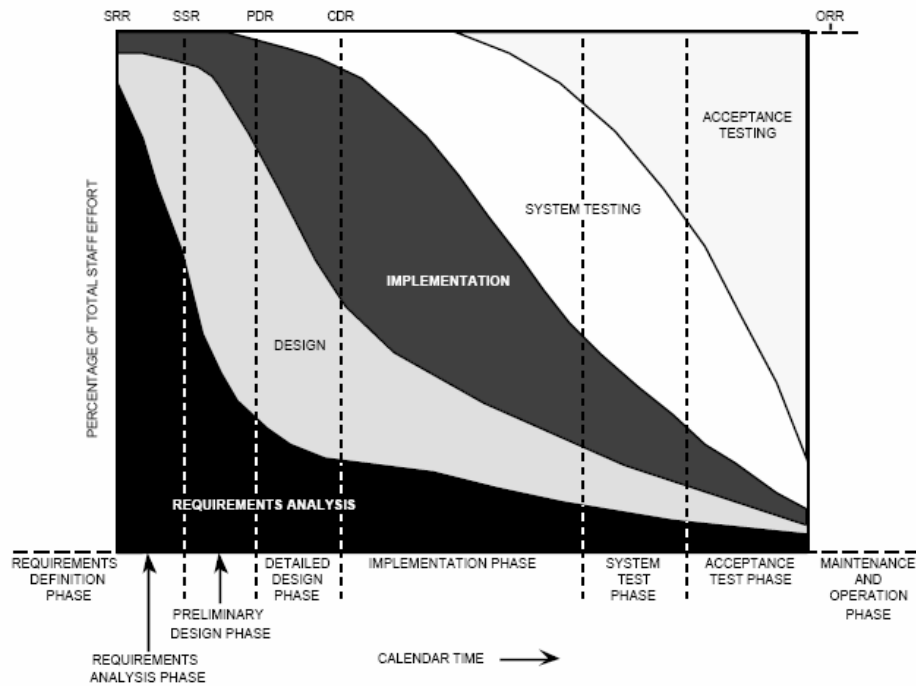


Figure 1.1: Activities by percentage of total development staff effort (Source: NASA, recommended approach to software development)

1.3 Proposed Approach

To resolve quality problems due to dynamic changes in requirements during the developing process, this research uses the Process Area (PA) of Requirement Management (REQM) related practices in the CMMI, as follows [7]:

SG 1 Manage Requirements

SP 1.1 Obtain an Understanding of Requirements

SP 1.2 Obtain Commitment to Requirements

SP 1.3 Manage Requirements Changes

SP 1.4 Maintain Bidirectional Traceability of Requirements

SP 1.5 Identify Inconsistencies between Project Work and Requirements

The SEI of Carnegie Mellon University also offers a context diagram for Requirement Management (REQM), as Figure 1.2 shows [8], and presents the relations among various standards. This research takes these five practices and the context diagram as foundation, in order to develop a multimedia learning contents (MLCs) developing process called the MLCs Quality Control (MQC) model. This MQC model provides an inconsistent detection mechanism for the developing process of MLCs under a collaboration environment, to avoid quality problems due to inconsistency.

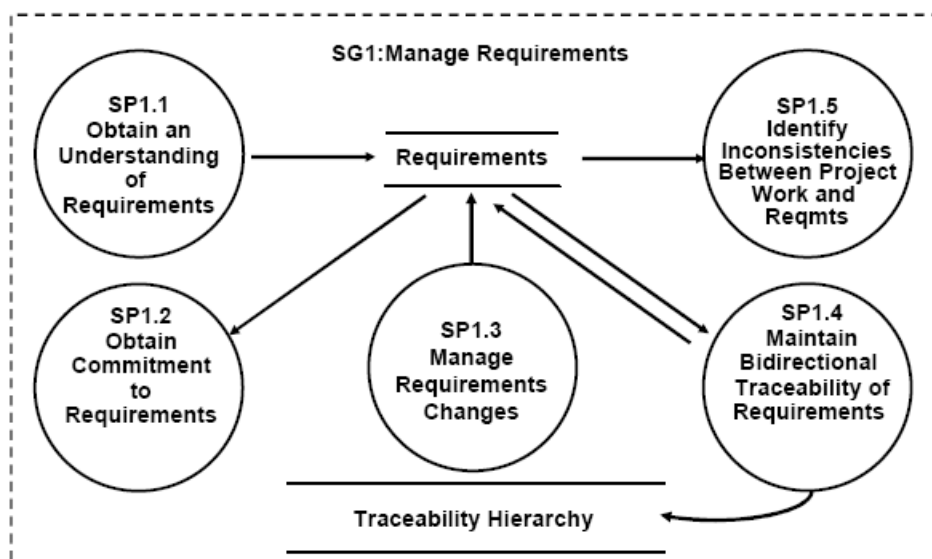
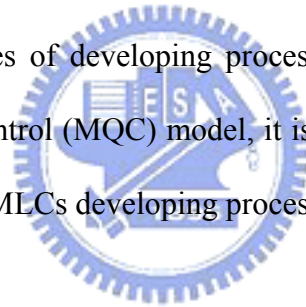


Figure 1.2: Requirements Management (REQM) context diagram
(Source: Introduction to CMMI V 1.1 by Carnegie Mellon University)

1.4 Motivation

At any phases in the multimedia learning contents (MLCs) developing process, whether in the modification of the original designs or multimedia objects, the inconsistency phenomenon can easily happen and cause many latent unstable factors in the multimedia final products. This kind of quality problem may influence the end user of the MLCs, including the satisfaction and desire to use the product of the teacher or student. That also results in the embarrassing situation for the design staff of endless required debugs, causes the delay of due-date, and increases development costs and the loss of market competitiveness for the company.

Because of the need of preventing protecting inconsistency phenomenon occurs on MLCs elements at any phases of developing process life cycle and the lack of entire solution for MLCs quality control (MQC) model, it is an urgent demand of providing an integrated MQC solution for MLCs developing process in this digital world.



1.5 Objective

This thesis will focus on developing a CMMI-based, integrated and server-client system for multimedia learning contents (MLCs) designers to reveal inconsistency phenomenon on MLCs elements at any phases of developing process life cycle. This tool includes extra reports which are relative to MLCs information such as inconsistency components and MLCs components process status, and then may be used to assist the matching requirements of CMMI relative practices. A tool helping MLCs designers or MLCs quality review stakeholder easily execute collaboration operations on web-based software environment.

1.6 Thesis Organization

The remains of this thesis are structured as follows: the background and relative literatures about this research is described in Chapter 2. Then, Chapter 3 discusses the details of MLCs quality control (MQC) mechanism and inconsistent detection capability we proposed. Chapter 4 introduces the architectural definition of a MQC system with client-server structure. Chapter 5 presents an example to explain the appliance of this MQC model to demonstrate the feasibility and the applicability of this system. Finally, we summarize the contribution and conclusion of this research and present recommendations for further research in Chapter 6.



Chapter 2

Related Work

In this chapter, we will review learning object and the foundation of Sharable Content Object Reference Model (SCORM), Multimedia Learning Objects, Capability Maturity Model Integration (CMMI), causes of inconsistencies, detecting and identifying inconsistencies of software for handling inconsistencies.

2.1 Sharable Content Object Reference Model

Since 1997, Advanced Distributed Learning (ADL) made efforts on Sharable Content Object Reference Model (SCORM) as the standard specification for the LMS and sharable content, it declared to save learner time in terms of output performance because it not only made learner browses course content more easily on the web, but also gave a more high interoperable learning content. In the long term, verifying content quality and taking an accounting management for content's reusing and sharing can be assigned to an independent third party or an institute of official government.

The SCORM aims to coordinate emerging-technologies and commercial and/or public implementations, including the Meta-data Dictionary from IEEE Learning Technology Standards Committee (LTSC), Content Packaging from the IMS Global Learning Consortium, Inc., and Data Model and Communication from the Aviation Industry CBT (Computer-Based Training) Committee (AICC). The SCORM makes course content sharable by way of two steps. Firstly, standardizing courseware content as package with structured metadata file must be made if content is developed. Secondly, a content package can be delivered in SCORM-compliant learning management system.

SCORM consists of two parts. One is the Content Aggregation Model (CAM), which is used to define course components and then to pack them with organization, manifest and java script files together. The output of this packing format makes exchanging course content possible. The other one is the Run-Time Environment, which defines a common way to start course content, and to communicate with a learning management system during its execution. The definition for the four areas of SCORM follows [20][21][22][23]:

- **Reusable:** Content is independent of learning context. It can be used in numerous training situations or for many different learners with any number of development tools or delivery platforms.
- **Accessible:** Content can be identified and located when it is needed and as it is needed to meet training and education requirements. As the definition, the goal of SCORM is to create flexible learning materials by ensuring content that is reusable, interoperable, durable, and accessible, regardless of the content delivery and management system.
- **Durable:** Content does not require modification to operate as software systems and platforms are changed or upgraded.
- **Interoperable:** Content will function in multiple applications, environments, and hardware and software configurations regardless of the tools used to create it and the platforms on which it is delivered.

SCORM achieves its goal with the use of Sharable Content Object (SCOs), which are composed of assets. In order to identify SCOs, assets, or any other types of learning materials, ADL proposed “metadata”, which means “data about data”, to identify and locate learning materials by managers, learners, designers, programmers and others who are interested in education. Units in SCORM are introduced from the lowest level (asset)

to the highest level (course) in the following sections.

2.1.1 Assets

Assets are the smallest physical units in SCORM. They are electronic representations of media, such as texts, audios, web pages, assessment objects, and other pieces of data, that can be delivered to a Web client (Figure 2-1)[20]. In order to be reused, assets must be described with metadata. Assets may be reusable in many contexts and applications by searching in online instructional repositories.

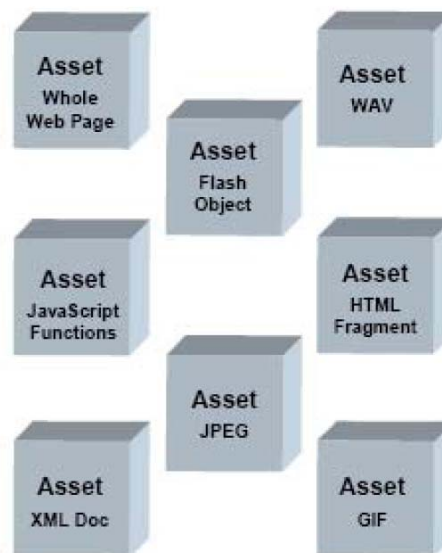


Figure 2.1: File types of Assets

2.1.2 Sharable Content Object

A Sharable content object (SCO) is a collection of assets that becomes an independent, defined piece of learning material. SCOs are the smallest logical unit of instruction you can deliver and track via a learning management system [20]. It is important that SCOs cannot directly access other SCOs. In addition, each SCO should be able to stand alone. A SCO could be a learning object, segment, lesson, module, chapter, or unit. Instructors can design the “role”, what a SCO play in their own way. The term of

SCO may have different meaning for different users. For an instructor, SCO means instructional content. For a programmer, SCO is a pointer in the source code when they create manifest or content package. With the usage of metadata described for SCO, authors can search, discover, reuse, and aggregate SCO within content repositories.

2.1.3 Content Aggregation

A Content Aggregation (CA) is defined as a parent and its children in a tree structures. They are used to group related content so that it can be delivered to the learner in the manner the designer describes [21]. CA is the process of aggregating resources into a defined structure to build a particular learning experience (Figure 2-2). In other words, it is composed of one or many SCOs or another aggregation. It is also a course material structure without sequencing rule in teacher or student's views. In Figure 2-2, there are two types of resources—SCOs and Assets. Resources are integrated into the hierarchical content structure.

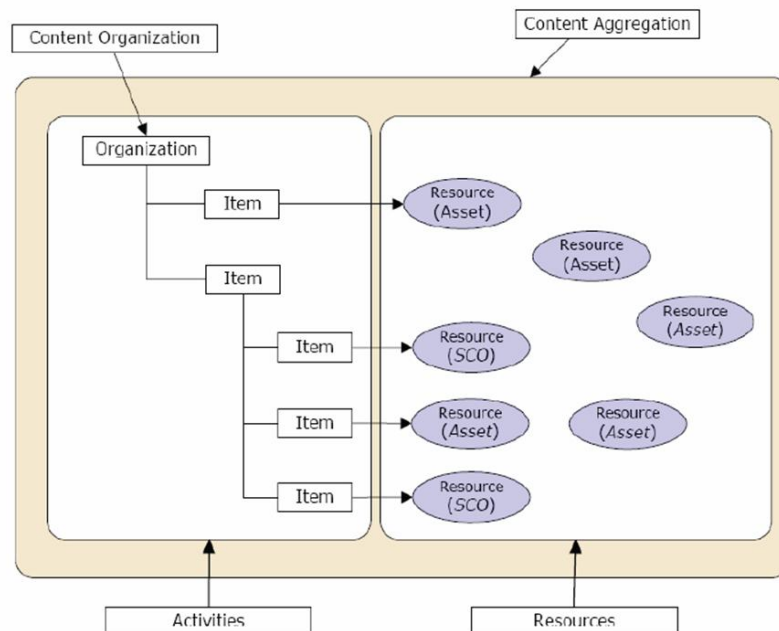


Figure 2.2: Example of Content Aggregation (CA)

2.1.4 Content Package

Content Package (CP) is based on IMS Content Packaging Specification which describes data structures used to provide interoperability of Internet-based content with authoring tools, learning management systems, and run-time environments. The purpose of content packaging is to provide a standardized way to exchange digital learning resources between different systems or tools [21]. A content package contains two components: manifest and physical file. Manifest will be introduced in the next section. (Figure 2.3) The physical files may be local files that are actually contained within the content package or be external files that are referenced by a Universal Resource Locator (URL). A content package may be part of a course, a unit of learning object, or an entire course. It should be able to stand alone. When the content package arrive its destination, it allows itself to be disaggregated or aggregated.

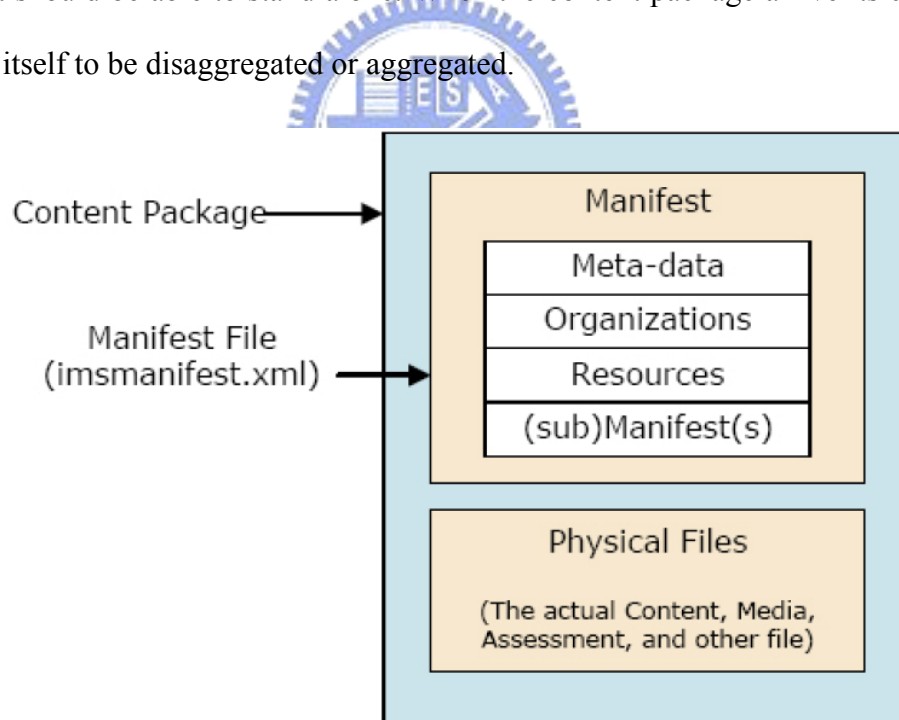


Figure 2.3: Content Package (CP) conceptual diagram

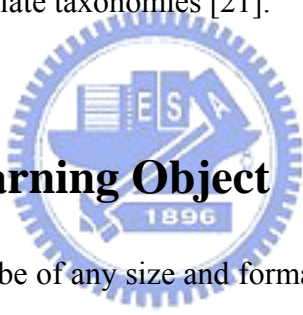
2.1.5 Manifest

A manifest is a document that contains a structured inventory of the content of a package (Figure 2.3) [20]. A manifest describes how the content is organized. In addition, a manifest translates source code from content structure to Web files by transfer content structure and behavior, and the list of the references to the source in the package.

2.1.6 Content Structure

Content structure is a basic tool to define the course structure by courseware designers. Content structure defines a mechanism that can be used to aggregate learning resources into a cohesive unit of instruction (e.g., course, chapter, module, etc.), application structure and associate taxonomies [21].

2.2 Multimedia Learning Object



Educational content can be of any size and format and sometimes in mixed formats. Moreover, educational content can be presented as text, graphics, animated graphics, audio, video, logic, or a combination of any type of multimedia. The associated metadata represents a standardized set of rules and methods that allow collecting and processing this content in many perspectives such as learners, content providers. Therefore, a learning object can be defined as a structured electronic resource that encapsulates its metadata based on SCORM specifications that consists of learning object description, keyword, ownership, and so on. For school teachers, one obstacle to the spread of online course is the mismatch between what they really need—customized courses that are tailored to the course object and its unique schooling culture, and what they can afford. Creating even one e-learning course from scratch can take several months and involve

many domain experts. Thus, most school teachers are stuck with classroom teaching. To resolve this status, we must define “Learning object” and its metadata (Learning Object Metadata). The associated metadata represents a standardized set of rules and methods that allow collecting and processing this content [18].

As stated in the specification of IEEE’s Learning Objects Metadata (LOM) [18], “a learning object is defined as any entity, digital or non-digital, which can be used, re-used or referenced during technology-supported learning”. Examples of Learning Objects include multimedia content, instructional content, instructional software and software tools, referenced during technology supported learning. In a wider sense, learning objects could even include learning objectives, persons, organizations, or events. A learning object is not necessarily a digital object; however, the remainder of this paper will focus on learning objects that are stored in a digital format. The learning object (LO) model is characterized by the belief that independent chunks of educational content can be created that provide an educational experience for some pedagogical purpose. With regard to object-oriented programming (OOP), this approach asserts that these chunks are self-contained, though they may contain references to other objects, and they may be combined or sequenced to form longer (larger, complex, other) educational units. These chunks of educational content may be of any type, interactive (e.g. simulation) or passive (e.g. simple animation), and they may be of any format or media type. Another requirement for learning objects is related to tagging and metadata. To be able to use such objects in an intelligent fashion, they must be labeled as to what they contain, what they communicate, and what requirements with regard to their use exist. A reliable and valid scheme for tagging learning objects is hence necessary.

The LO model provides a framework for the exchange of learning objects between systems. If LOs are represented in an independent way, conforming instructional systems

can deliver and manage them. The learning object initiatives, such as IEEE's LOM or Educom's IMS are a subset of efforts to creating learning technology standards for such interoperable instructional systems.

2.3 Capability Maturity Model Integration

Capability Maturity Model Integration (CMMI) is the new de-facto standard process improvement model for determining the organizational maturity in product or software development (Figure 2.5). Since many organizations would like to focus their improvement efforts across the disciplines within their organizations. In October of 1997, U.S. Department of Defense requested SEI to include the development of a common framework for supporting the future integration of other discipline-specific CMMI models (Figure 2.6). CMMI is an important model for product development industry. Many procurers require a specific level of maturity from their suppliers. Also many companies set internal process improvement objectives driven by the maturity levels of CMMI.

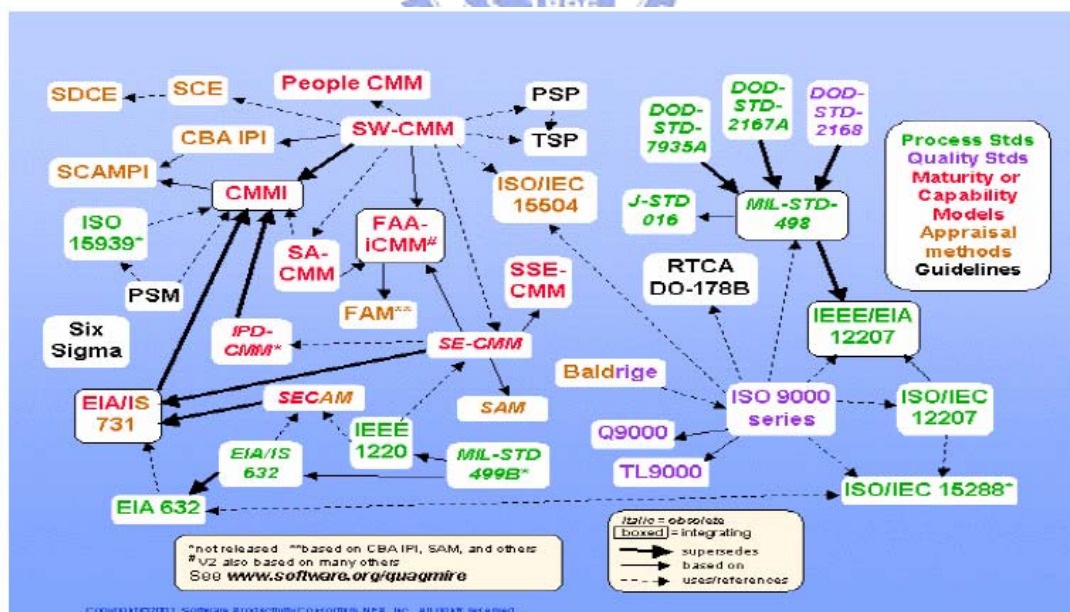


Figure 2.5: The frameworks quagmire
(Source: Sarah A. Sheard, Software Productivity Consortium 1997)

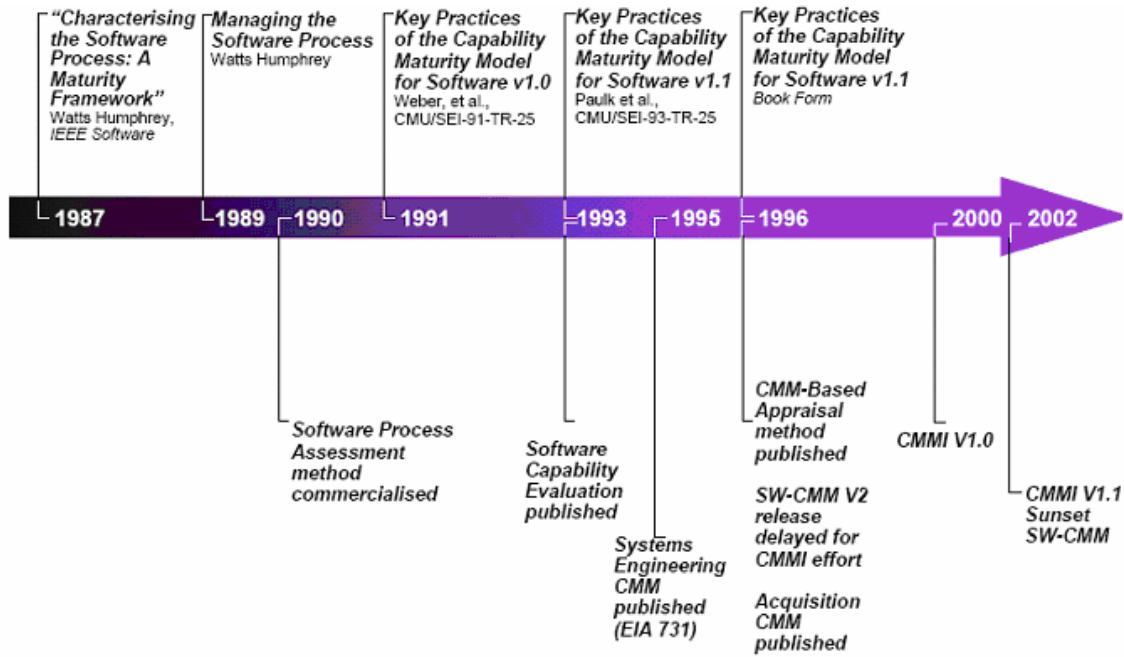


Figure 2.6: CMMI History

(Source: *Interpreting the CMMI: A Process Improvement Approach*, April 2003)

A model is a simplified representation of the world. Capability Maturity Models (CMMs) contain the essential elements of effective processes for one or more bodies of knowledge. These elements are based on the concepts developed by Crosby, Deming, Juran, and Humphrey [8]. CMMI is an integrated model of many CMMs intended to achieve process improvement (Figure 2.7). CMM is a model that contains the essential elements of effective processes for one or more disciplines and describes an evolutionary improvement path from ad hoc, immature processes to disciplined, mature processes with improved quality and effectiveness [4]. CMMI models are not processes or process descriptions. The actual processes used in an organization depend on many factors, including application domain(s) and organization structure and size. In particular, the process areas of a CMMI model typically do not map one to one with the processes used in organization.

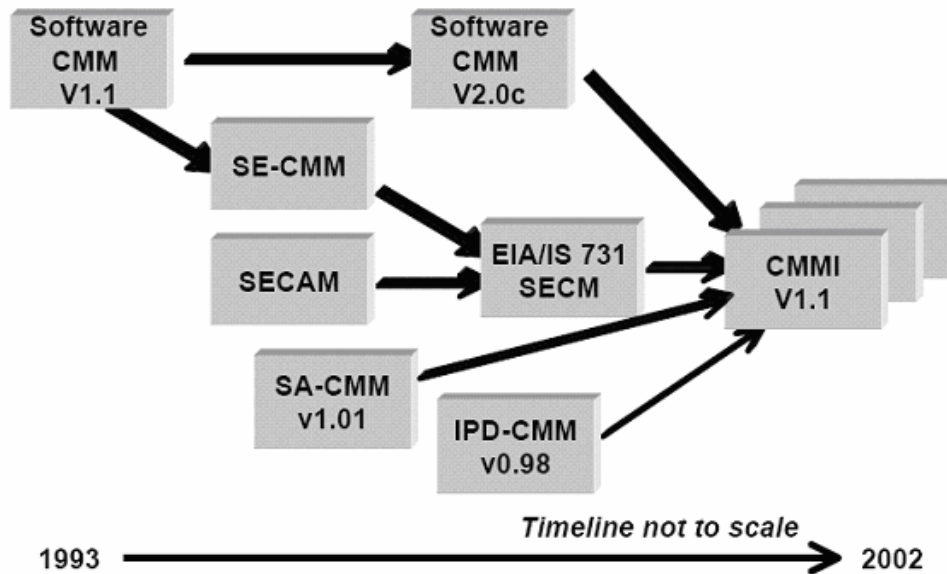


Figure 2.7: CMMI derivation

(Source: Interpreting the CMMI: A Process Improvement Approach, April 2003)

CMMI defines five levels of organizational maturity in product development (Figure 2.8). Level 1 (initial) represents the lowest, and the level 5 (optimizing) the highest maturity. CMMI defines the maturity levels through process areas. By default, every organization is at maturity level 1. To reach level 2, an organization should satisfy the goals of seven process areas – such as Requirements Management and Project Planning. To achieve the level 3, an organization should perform all the process areas of the level 2 plus the process areas defined for the level 3 – such as Requirements Development and Technical Solution. Analogically, maturity levels 4 and 5 require the implementation of new process areas as well as those of the lower level process areas. We describe as follows:

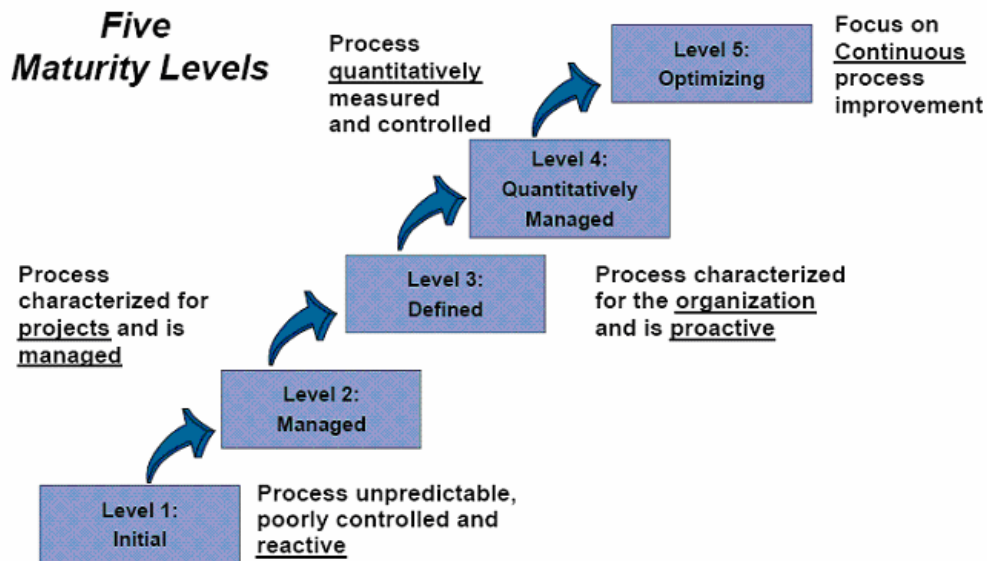


Figure 2.8: CMMI maturity framework

(Source: *Interpreting the CMMI: A Process Improvement Approach*, April 2003)

- **Maturity 1: Initial**

At maturity level 1, processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment. Success in these organizations depends on the competence and heroics of the people in the organization and not on the use of proven processes. In spite of this ad hoc, chaotic environment, maturity level 1 organizations often produce products and services that work; however, they frequently exceed the budget and schedule of their projects.

- **Maturity 2: Managed**

At maturity level 2, an organization has achieved all the specific and generic goals of the maturity level 2 process areas. In other words, the projects of the organization have ensured that requirements are managed and that processes are planned, performed, measured, and controlled.

- **Maturity 3: Defined**

At maturity level 3, an organization has achieved all the specific and generic goals of the process areas assigned to maturity levels 2 and 3. At maturity level

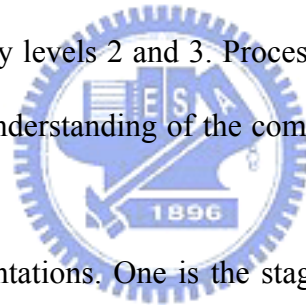
3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods.

- **Maturity 4: Quantitatively Managed**

At maturity level 4, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, and 4 and the generic goals assigned to maturity levels 2 and 3. Sub-processes are selected that significantly contribute to overall process performance. These selected sub-processes are controlled using statistical and other quantitative techniques.

- **Maturity 5: Optimizing**

At maturity level 5, an organization has achieved all the specific goals of the process areas assigned to maturity levels 2, 3, 4, and 5 and the generic goals assigned to maturity levels 2 and 3. Processes are continually improved based on a quantitative understanding of the common causes of variation inherent in processes [3].



CMMI has two representations. One is the staged representation. The other is the continuous representation. In the staged representation, maturity level of an organization ranges from level 1 to 5. In the continuous representation each process capability level ranges from 0 to 5. The staged representation is most suitable for an organization that does not know which processes need to be improved first because the staged representation offers process areas applicable to each maturity level (Figure 2.9). The continuous representation provides flexibility for selecting processes fit for achieving business goal of the organization [5]. CMMI provides 25 process areas (Process area means a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making significant improvement in that area [4]. Goals are classified as generic goals and specific goals. A generic goal describes

the characteristics that must be present to institutionalize the processes that implement a process area. A specific goal describes the unique characteristics that must be present to satisfy the process area [4].

Practices are expected components for satisfying goals. Practices are classified as generic practices and specific practices. A generic practice is the description of an activity that is considered important in achieving the associated generic goal. A specific practice is the description of an activity that is considered important in achieving the associated specific goal [4].



Figure 2.9.: CMMI stage representation

(Source: Interpreting the CMMI: A Process Improvement Approach, April 2003)

2.4 Causes of Inconsistencies

Inconsistency is an inevitable part of a complex, incremental software development process. Even in an idealized process, system requirements are often uncertain or contradictory, alternative design solutions exist, and errors in implementation arise.

The requirements engineering stage of development is particularly illustrative of such inconsistencies. During requirements acquisition, customer requirements are often sketchy and uncertain. For large projects in particular, a number of “client authorities” may exist who have conflicting, even contradictory requirements. In many instances customers may not even be certain of their own needs, and a requirements engineer’s job is partly to elicit and clarify these needs. The requirements specification produced as a result of such a specification and analysis process however is not static: it continues to evolve as new requirements are added and conflicts identified are resolved. In fact, even with strict project management practices in place, requirements specifications - and subsequent design specifications - continue to evolve.

Thus, there is a wide range of possible causes of inconsistencies and conflicts in software development. Many of these are due to the heterogeneity of the products being developed (e.g., systems deploying different technologies) and the multiplicity of stakeholders and/or development participants involved in the development process. Inconsistencies arise between multiple development participants because of [3]:

- the different views they hold,
- the different languages they speak,
- the different development strategies (methods) they deploy,
- the different stages of development they address,
- the partially, totally or non-overlapping areas of concern they have, and

- the different technical, economic and/or political objectives they want to achieve.

While inconsistencies can occur in software development processes and products for a variety of reasons, we adopt a simple definition of what actually constitutes an inconsistency [3]:

An inconsistency occurs if and only if a (consistency) rule has been broken.

Such a rule explicitly describes some form of relationship or fact that is required to hold. In previous work, we have examined three uses of such consistency rules. They may describe syntactic relationships between development artifacts prescribed by a development method, which is also a way of describing semantic relationships between artifacts produced by that method [2]. They may also be used to prescribe relationships between the sub processes in an overall development process, which is also a way of coordinating the activities of developers deploying different development strategies. Finally, consistency rules can be used to describe user-defined relationships that emerge as development of a software specification proceeds. This is useful for capturing ontological relationships between the products of a development process (for example, two developers specifying a library system may use the term “user” and “borrower” to refer to the same person).

Reducing an inconsistency to the breaking of a rule facilitates the identification of inconsistencies in specifications, and is a useful tool for managing other “problems” that arise during software development. For example, if we treat conflict as the interference of the goals of one party caused by the actions of another party, then we can use inconsistency as a tool for detecting many conflicts. Similarly, if we define a mistake as an action that would be acknowledged as an error by its perpetrator (e.g., a typo), then we can detect mistakes that manifest themselves as inconsistencies.

2.5 Detecting and Identifying Inconsistencies

Detecting an inconsistency that breaks an explicit rule is relatively straight forward. For example, a type checker can check whether or not an instance or variable conforms to its type definition. Similarly, a parser can check whether or not a sentence conforms to the syntactic rules specified by its grammar. Simple inferences in classical logic can also be used to detect logical inconsistencies resulting from too much or too little information. For example, a contradiction (where a rule of the form $X \rightarrow \neg X$ has been broken) may be detected in this way.

Other kinds of inconsistency are more difficult to detect. A conflict between two development participants may not manifest itself as an inconsistency until further development has taken place (making the original source of the inconsistency difficult to identify). Furthermore, what actually constitutes an inconsistency from one participant's perspective may not be the case from another perspective. An example of this is an "inconsistency" in a person's tax return. Such an inconsistency may actually be a "desirable" piece of information from a tax inspector's point of view!

One of the difficulties in handling inconsistencies effectively, even after they have been successfully detected, is that the kind of inconsistency detected also has to be identified. The CONMAN project for example, attempts to classify consistency in programs into one of several kinds in order to facilitate inconsistency handling later on:

- Full consistency - where a system satisfies the rules that a programming language specifies for legal programs (insofar as they can be checked prior to execution).
- Type consistency - where a system satisfies the static type checking rules of the programming language.

- Link consistency - where each compilation unit is free of static type errors, and each symbolic reference between compilation units is type safe according to the rules of the programming language.
- Reachable consistency - where all code and data that could be accessed or executed by invoking the system through one of its entry points are safe.

The CONMAN system checks for all kinds of consistency automatically, and then reacts differently depending on the kind of inconsistency detected. It does however appear appropriate for configuration management applications only, and it is therefore desirable to identify a more general set of inconsistencies that arise during software development in-the-large system.

2.6 Handling Inconsistencies

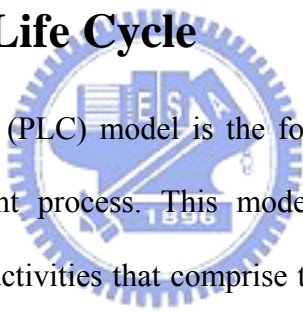
Many approaches to handling inconsistency attempt to maintain and enforce consistency, usually by adopting simple procedures for inconsistent detection followed by immediate resolution. We now examine alternative approaches to inconsistency handling that “tolerate inconsistency” [2] in a variety of ways. We believe that these approaches represent more realistic attempts at supporting multimedia learning objects development, and we therefore discuss some general techniques for acting and reasoning in the presence of inconsistency.

Chapter 3

Design MLCs Quality Control Methodology

Whatever the cause or kind of inconsistency that exists in a multimedia learning contents (MLCs) developing process, there is a demand for semi-automated mechanism that detect, identify, record and handle such inconsistencies in this setting. In this chapter, we will briefly explain on the design methodology of proposed of MLCs quality control (MQC) method including MLCs bidirectional traceability matrix, MLCs directed acyclic graph (MDAG), and the inconsistent detection capability method that supports such MLCs inconsistent detection mechanism and used in the remainder of the paper.

3.1 MLCs Process Life Cycle



The Process Life Cycle (PLC) model is the foundation of a multimedia learning contents (MLCs) development process. This model is a directed acyclic graphical representation of the various activities that comprise the MLCs development process. Its purpose is to show explicitly what is to be done and in what sequence. Although general MLCs process life cycle model was decided to follow a variation of the waterfall life cycle model (Figure 2.4) because of its emphasis on planning, a variation of a spiral model [16] was finally adopted. The main reason was that the MLCs developing process is highly iterative process which can be well illustrated by a spiral model while the waterfall model follows a sequential developing approach.

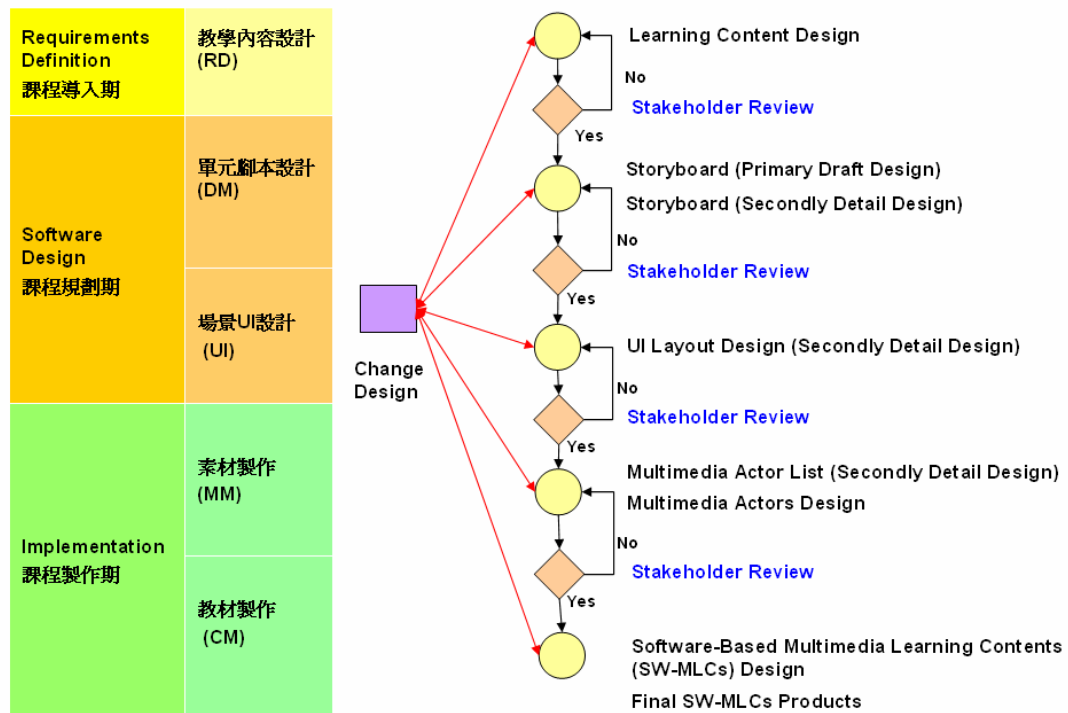


Figure 3.1: MLCs process flow chart

The development methods describe how the various phases which comprise the MLCs developing process are implemented. Each phase contains a number of activities shown in Table 3.1. As illustrated in Table 3.1, the MLCs developing process was divided into five phases [24].

Table 3.1: Phases and activities in MLCs developing process

Phases	Activities
Courseware Initialization (RD)	<ul style="list-style-type: none"> – definition of target audience – definition of aims and objectives – definition of subject matter – specification of pedagogical methods – specification of assessment methods
Courseware Planning Unit Scripts Design (DM)	<ul style="list-style-type: none"> – allocation of the content to courseware parts – allocation of learning activities to courseware parts – for each courseware component design of: structure, access, layout, navigation, etc. – storyboarding

Courseware Planning Scenario UI Design (UI)	– UI and background designing
Courseware Development Multimedia Material Design (MM)	– design of text – design of graphics – design of sound – design of animation – design of video
Courseware Development Courseware Integration (CM)	– integration of the various elements into a whole

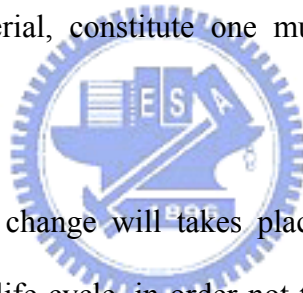
The division of the courseware development process into 5 phases: Courseware Initialization , Courseware Planning- Unit Scripts Design, Courseware Planning- Scenario UI Design, Courseware Development- Multimedia Material Design, Courseware Development-Courseware Integration and their division into activities was done based on the methodology of the BESTWISE international company method [24]. The reason is that this method from the previous generations by focusing on the activities of each phase offering flexibility and employment of instructional principles and theory.



3.2 MLCs Bidirectional Traceability Matrix

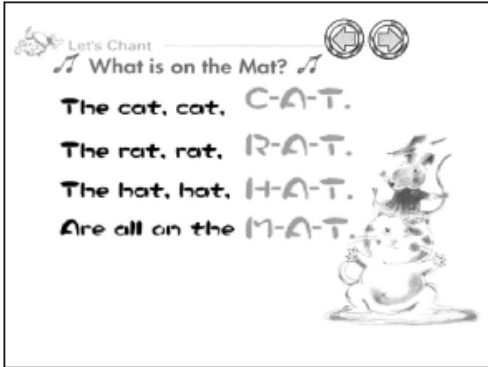
For resolving multimedia learning contents of the collaboration environment of develop process, due to change the script or multimedia material of the originality design specification will cause the quality problem of the inconsistency. Have to aim at according to the specific practice (SP) concerning the manage requirements in the CMMI develop any stage of the process life cycle (PLC) the variations need (the Requirements Changes) in the multimedia teaching material cause of Inconsistency quality problem, create a set of valid mechanism of Quality Control. First in regard to the developing life cycle of the multimedia learning contents, its basic step is as follow:

- The teaching material design of the multimedia learning contents
- The contents design storyboard of the basic teaching plan
- Design the field of the multimedia learning contents view and User Interface depend on the Storyboard plot demanded
- According to the field of the Storyboard plot, multimedia learning contents view with make the User Interface of need, the multimedia material actor needed by the design learning contents as (text, picture, animation, video, music)
- The multimedia of the usage adequacy edits tool, according to the storyboard plot contents, the field of the multimedia learning contents view and the User Interface and material, constitute one multimedia learning contents of the integrity



Since the requirements change will takes place at any stage of the multimedia learning contents developing life cycle, in order not to the creation of the Inconsistency quality problem, have to have the observation in the learning contents the developing the process, because of modifying the view or multimedia material of plot, field for end the capability of the range dimension of influence that finished product cause. First want to understand at what the Storyboard contents describes is each plot, to the field that should use the view or multimedia material is why. Can create the dependency matrix here, recording it each other of dependency relationship. For example, a script's contents that suppose the multimedia learning contents designed is shown as Table 3.2, from here the script contents can knows to have a scene all together with three natural language script, each a plot use a different multimedia actors, but use the same user interface design.

Table 3.2: An example of storyboard

User Interface: UI1	
	
Natural Language Script	Multimedia Actor
NLS1: 播放鈴聲，Let's Chant What is on the Mat? 前後方向按鈕的圖片，隨開場場景出現。	AR1 [Aduio1] AR2 [Pic1]
NLS2: 播放鈴聲，女生說「Page sixteen. Let's Chant. What is on the Mat?」	AR2 [Aduio1] AR3 [Aduio2]
NLS3: 畫面出現對話 男生說「The cat, cat, C-A-T.」女生說「The rat, rat, R-A-T.」 男生說「The hat, hat, H-A-T.」 男生和女生同時說「Are all on the M-A-T.」	AR1 [Aduio1] AR2 [Pic1] AR4 [Aduio2]
Final Product: CM1	

According to Table 3.2 of the scope script contents, can know the contents that is the modification plot NLS1, may influence the design of a view UI1 and multimedia material actor AR1, modify the contents of the plot NLS2, may influence the design of a view UI1 and multimedia material actor ATR2, modify the contents of the plot NLS3, may influence the design of a view UI1 and multimedia material actor ATR1, ATR2, modify the design of a view UI1, may influence the design of multimedia material actor ATR1, ATR2, ATR3, ATR4 and the end multimedia learning contents finished product CM1;When modify the design of the multimedia material actor ATR1, ATR2, ATR3, ATR4, may influence the design of the end multimedia learning contents finished product CM1. We can know what of the plot, field view or multimedia material actor exists by the above analysis mutually according to the dimension of the dependency

relationship and the modification influence caused in design range, here can define a forward direction mutually according to the Forward Dependency Matrix, such as Table 3.3 to recording mutually according to the relationship. When each work stage exists mutually according to the relationship, then in forward direction mutually according to matrix to should have the related field setting value is 1, if the nonentity then sets to 0 according to relationship mutually. The so-called forward direction is mutually its main meaning is according to the matrix can by Table 3.3 medium Dependency-Out-Degrees (DOD).The DOD value means all the representatives are the multimedia learning content process life cycle of when the need design of the any Phase change, influence the design of the first order segment work of range dimension.

Table 3.3: MLCs forward dependency matrix (MFDM)

Name	Description
SCO	Story
SE	Scene
NLS	Natural Language Script
UI	User Interface
AR	Actor Type : Text \ Image / Picture \ Audio / Voice \ Video
CM	Code Module
PF	Physical Files

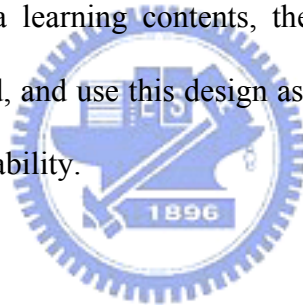
Source	Target									
D.Value	UI1	NLS1	NLS2	NLS3	AR1	AR2	AR3	AR4	CM1	Out-Degree
SE1	1	1	1	1	0	0	0	0	0	4
UI1	0	0	0	0	1	1	1	1	0	4
NLS1	0	0	0	0	1	1	0	0	0	2
NLS2	0	0	0	0	0	1	1	0	0	2
NLS3	0	0	0	0	1	1	0	1	0	3
AR1	0	0	0	0	0	0	0	0	1	1
AR2	0	0	0	0	0	0	0	0	1	1
AR3	0	0	0	0	0	0	0	0	1	1
AR4	0	0	0	0	0	0	0	0	1	1

We can makes to transpose according to the Forward Dependency Matrix mutually the forward direction in addition, can get another reverse direction mutually according to the Backward Dependency Matrix, such as Table3.4 shows. So-called reverse direction mutually according to matrix its main meaning can by Table 5 medium of the meaning of the Dependency-In-Degree (DID) to explain its characteristic, the DID value all the representatives are the design need of any phase of the multimedia learning contents developing life cycle, being subjected to an arrival from last the design work of the first order segment influence of range dimension.

Table 3.4: Multimedia backward dependency matrix (MBDM)

Name	Description	Target	Source									
SCO	Story	D.Value	SE1	UI1	NLS1	NLS2	NLS3	AR1	AR2	AR3	AR4	In-Degree
SE	Scene	UI1	1	0	0	0	0	0	0	0	0	1
NLS	Natural Language Script	NLS1	1	0	0	0	0	0	0	0	0	1
UI	User Interface	NLS2	1	0	0	0	0	0	0	0	0	1
AR	Actor Type : Text \ Image / Picture \ Audio / Voice \ Video	NLS3	1	0	0	0	0	0	0	0	0	1
		AR1	0	1	1	0	1	0	0	0	0	3
		AR2	0	1	1	1	1	0	0	0	0	4
		AR3	0	1	0	1	0	0	0	0	0	2
CM	Code Module	AR4	0	1	0	0	1	0	0	0	0	2
PF	Physical Files	CM1	0	0	0	0	0	1	1	1	1	4

Synthesize forward direction mutually according to the Forward Dependency Matrix and the reverse direction mutually according to the Backward Dependency Matrix inside recording mutually according to the record of the dependency relationship, can know that the requirements change occurrence develops any stage of the Process Life Cycle in the multimedia learning contents, the related information of the range dimension of influence caused, and use this design as the foundation of CMMI to design Bidirectional Traceability capability.



3.3 MLCs Directed Acyclic Graph

In previous chapter, we defined forward direction mutually according to the Forward Dependency Matrix and the reverse direction mutually according to the Backward Dependency Matrix, can according to Table 2 and Table 3 of example, create according to the dependency relationship mutually the multimedia learning contents a new mutually according to model, be called a multimedia to lead to accord to mutually not circularly the MLCs Directed Acyclic Graph (MDAG) $G=(V, E)$ as Figure 3.2 show. In each node $s \in V$ in the MDAG, representative multimedia learning contents the finished work needed exists mutually according to the dependency relationship in each first order phase developing life cycle. In each edge in the MDAG $(u, v) \in E u \rightarrow v$, of the work existence that represents each first order segment of the multimedia learning contents developing life cycle mutually according to the dependency relationship with complete order of sequence.

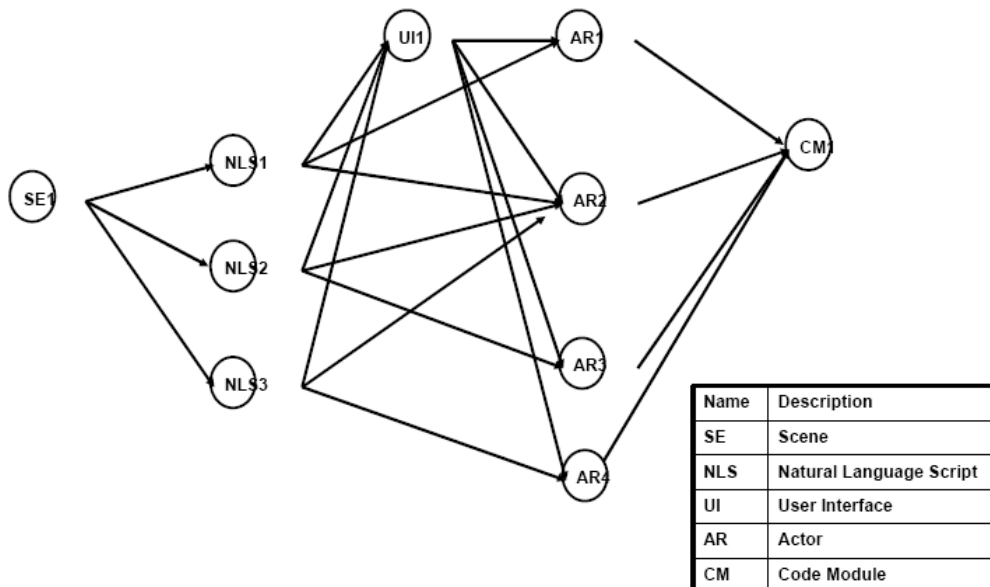


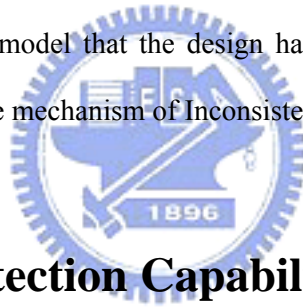
Figure 3.2: MLCs directed acyclic graphic (MDAG)

In the MDAG graphics inside the type of the nodal point used is as follow:

- The Scene nodes: This nodal point representative is in the storyboard, the independence scope menu homework unit in the multimedia learning contents. Have to complete a relation of recording the related plot, multimedia material and the usage field view in this nodal point
- The Natural Language Script nodes: This nodal point representative is in the storyboard, in the independence scope in the multimedia learning contents, multimedia material plot that needs to be performed. Have to explain the multimedia material is according to the time of the plot and the relation of the space carries on performing behavior by natural language in this nodal point. The attribute of the plot nodal point can be divided into: The opening scenario, the interactive scenario and the finale scenario.
- The User Interface nodes: This nodal point representative is in the storyboard, in the independence scope in the multimedia learning contents, multimedia material for performing a related plot, have to use of the stage and the background. Have to complete the design work of a view and the user interface in this nodal point
- The Actor nodes: This nodal point representative is in the storyboard, in the independence scope in the multimedia learning contents, according to the details demand of the plot, the multimedia material actor used. Have to complete the design work of the multimedia material in this nodal point
- The Code module nodes: This nodal point representative is in the storyboard, in the independence scope in the multimedia learning contents, multimedia material according to this the related plot of the scope, Be in the particular field the view carries on performing of present a result. In all aspects this

nodal point has to edit the software tool by multimedia learning contents, completing contents of course to carry.

We cannot know the explanation Table by MDAG the nodes and edges 2 and Table the Dependency Relationship record recording by 3 se two matrixes, and can know the need design that is any stage of the multimedia learning contents developing life cycle a change, influence the design of the first order segment work of range dimension, and the design need of any stage, be subjected to up the first order segment of the design work influence of range dimension. In addition the Directed Acyclic Graphs have directive and will not forming cycle of characteristic, can explain multimedia learning contents in the process of developing life cycle, need the characteristic that designs and develops order of sequence according to each workflow phase, meanwhile again can express each time a work stage of the Dependency Relationship. So the MDAG is fit is the foundation model that the design has multimedia learning contents of the Bidirectional Traceability and the mechanism of Inconsistent detection.



3.4 Inconsistent detection Capability Design

There are three broad areas of inconsistent detection that benefit from computer-based automated mechanism support, and what follows identifies their scope.

- **Detecting Inconsistency**

This includes a wide range of tools that check consistency rules, such as type checkers and parsers. Detecting inconsistency can be automated if the appropriate consistency rules can be defined precisely. Conflicts or mistakes that do not manifest themselves as inconsistencies (because no pre-defined rule was prescribed), cannot be detected automatically and normally require human involvement.

- **Identifying Inconsistency**

Once an inconsistency has been detected, the next step is to identify the kind of inconsistency it is (perhaps by comparing it against some pre-defined classification of inconsistencies). Identifying inconsistency automatically can be difficult, particularly if there are multiple sources/causes of the inconsistency. However, once an inconsistency is identified, then removing it is often also simplified. Tools that detect inconsistency usually also attempt to identify or suggest its possible cause.

- **Handling Inconsistency**

Reacting to inconsistencies in a system is a particularly challenging area for the provision of tool support. Many tools allow the inconsistency to be ignored or require actions to resolve it. Some of the tools described in section 4.1 also allow controlled development to continue in the presence of inconsistency. More tools are needed however for tracking inconsistencies in software systems, as well as tools that use this monitoring information to remove inconsistencies, or to ameliorate inconsistent information.

For providing the inconsistent detection mechanism in the multimedia learning contents developing life cycle, have to define Consistency condition first. Have to design while taking the multimedia learning contents developing process in the collaboration environment with teamwork, each segment been responsible by different member if having already modified an original need and designing a detect a mechanism to judge these needs and design of whether variations act against Consistency cognition condition or not. In the control policy of this standard consistency as follow:

- There are some work Si without reviewed, have to the ex- first order segment Si-1 the design work of the dependency relationship has already completed to

examine, just carrying on work also can allow.

- The idea of modification originality design, but can't carry on the first order segment S_{i+1} of design work
- Already the review work S_i can't modify the originality design, but can carry on the first order segment S_{i+1} design work of Dependency Relationship
- If there some already review the design work S_i , are requested to modify again the originality design, then need the execution Inconsistent detection, and pause a next stage S_{i+1} design work of Dependency Relationship, until the completion Review
- When we carry on the mechanism of inconsistent detection, have to check multi-media teaching material in the condition of the consistency in the process life cycle process as follows:

- Complete design time consistency: The design completion time that has the last rank design work S_i of the Dependency Relationship has to be small in descend the design of the rank design work S_i completion time

$$T(C_i) \leq T(C_{i+1}) \text{ for } i \geq 1, T(C_i) \geq 0$$

$T(C_i)$: Timestamp of MLCs component node i was finished design

- Complete review time consistency: Last rank design the work S_i that has the Dependency Relationship's examining time has to be small in descend the rank design work S_{i+1} of examine the completion time

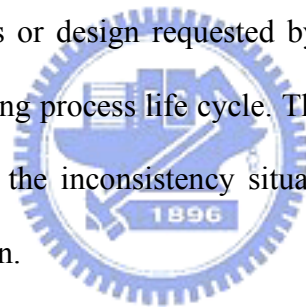
$$T(R_i) \leq T(R_{i+1}) \text{ for } i \geq 1, T(R_i) \geq 0$$

$T(R_i)$: Timestamp of MLCs component node i was confirmed

Chapter 4

System Implementation

In this chapter, we use the MLCs Quality Control (MQC) model discussed in Chapter 3 as the basis to implement the prototype of MQC System and design MLCs inconsistent detection mechanism. For standardization of multimedia learning objects we adopt the SCORM standard. Then, the final multimedia work products of MOQ system will follow the SCORM standard. There are two major purposes of quality control mechanism in this chapter. First is to detect the changes of the multimedia learning object's original requirements or design requested by MLCs designers or customers in any phases of MLCs developing process life cycle. Then the next design consideration is that MQC system can detect the inconsistency situation caused by changing the MLCs original requirements or design.



4.1 System Development Tools

During system development, this study uses Web-based application technology and Client-Server architecture to implement inconsistent detection mechanism in the MLCs Quality Control (MQC) System. The research applies PHP and Java program technology to compose MQC system and use ODBC technology as the basic of database connection and translation. In database, this research adopts MySQL Server to design database as the Back-end platform of the system. Figure 4.1 shows the system architecture physical view of MQC system.

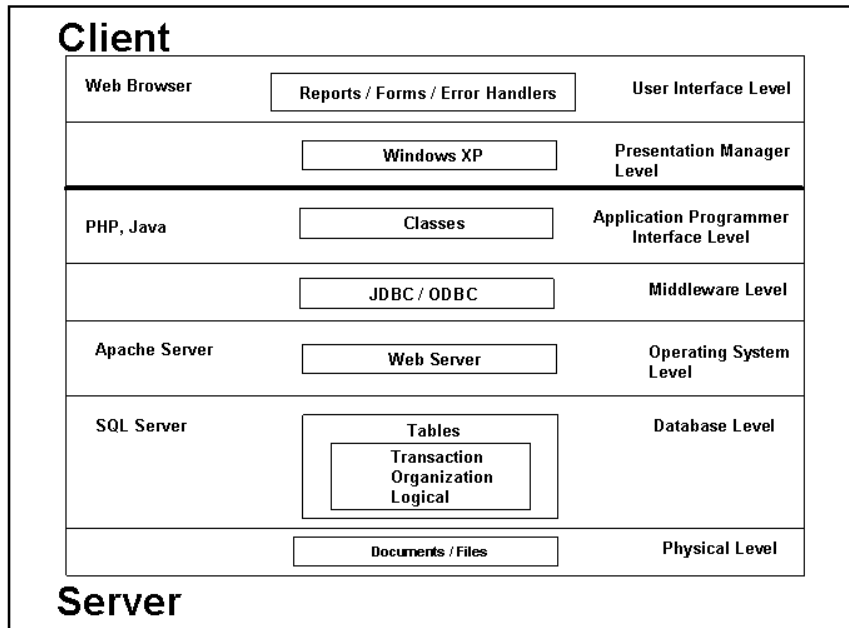


Figure 4.1: System Architecture

4.2 Software Architecture

In this section, we will discuss how to integrate the dependency requirements specifications to an automated inconsistent detection mechanism called MLCs Quality Control (MQC) System. The MQC system functional model architecture is described in Figure 4.2 below. It depicts the relationship between a MQC server and its client applications. Applications interact with the MQC server through the application interface to store and retrieve objects or perform MLCs developing operations. The MQC system requests from applications to multimedia objects and MLCs structure management subsystems which, in turn, interact with the physical files/documents management module to perform the desired operations.

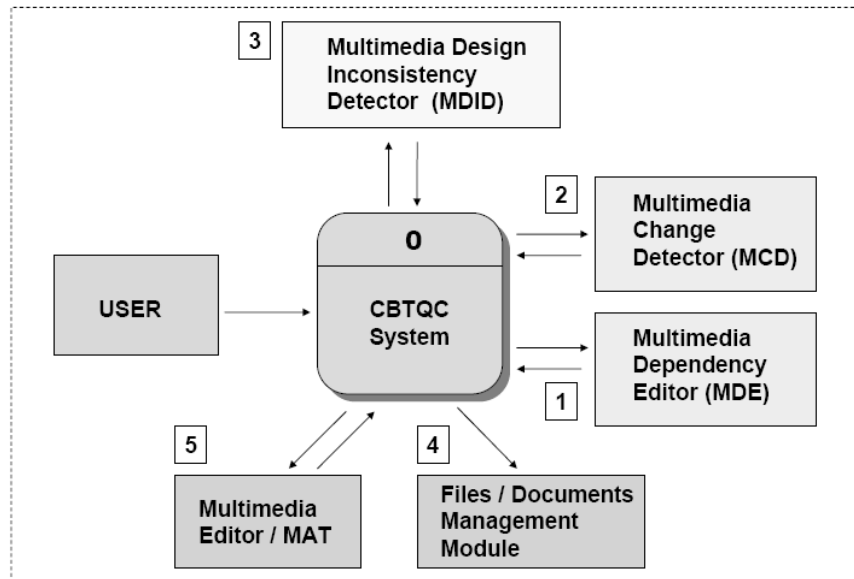


Figure 4.2: Functional model of the MQC system

The MQC system software architecture can be viewed as consisting of three major components:

- A MLCs dependency editor (MDE) for natural language scripts (NLS), scenarios (SNE) and multimedia actors (ATR) specifications respectively. (Note that this formal representation is likely to be an abstraction of the original specification).
- A MLCS change detector (MCD) for inspecting the currently operative NLS, the SNE, the currently applicable set of ATR (including text, image, audio and video-specific ATR), the user interfaces (UI), and the dependency relationship specification. The dependency relationship specification is represented both in the original notation it was written in NLS and in a formal representation that is obtained via multimedia directed acyclic graph (MDAG) markup and translation
- A MLCs inconsistency detector (MID) that serves to both detect and resolve inconsistencies in a dependency relationship specification.

Two alternative configurations of the system architecture exist: (1) each quality review stakeholder incorporates a role of the MQC system (2) only the dependency

relationship requirements repository incorporates the machinery of the MQC system.

4.3 MLCs Dependency Editor

In most multimedia dependency editor (MDE), it is possible to construct links between specific regions within nodes. Two levels of specification are required to completely identify the endpoints of this type of link – the inter-scene level and the intra-scene level. The inter-scene level resolves a link endpoint to the node level of granularity. At the intra-scene level, a location or region within the target node is designated as the precise endpoint for the link. Together these two levels completely define the link endpoint.

Dependency relationship specification complicates link specification in multimedia learning objects (MLO) by introducing complexities at both the inter-node and intra-scene levels of specification. In a MLO developing environment, inter-scene specification consists of identifying a particular object in the MLO that contains the target of a link. Since this involves MLO-level operations, this issue should be addressed at the MLO level. Intra-scene references specify locations or regions within MLO objects that serve as link endpoints. This level of specification requires application-specific knowledge (such as the format in which data objects are stored). The issues associated with intra-scene referencing are more appropriately addressed at the MLO level. Inter-scene references must identify a specific object in a MLO as the target of a link. In a MLO with dependency relationship specify facilities, several dependency relationship of an MLO might exist. The MLO identification process must be augmented to include the specification of dependency relationship specify information in order to uniquely identify each object. In our prototype implementation, we use multimedia dependency editor (MDE) as the multimedia learning objects (MLO) dependency relationship specification

editor (Figure 4.3).

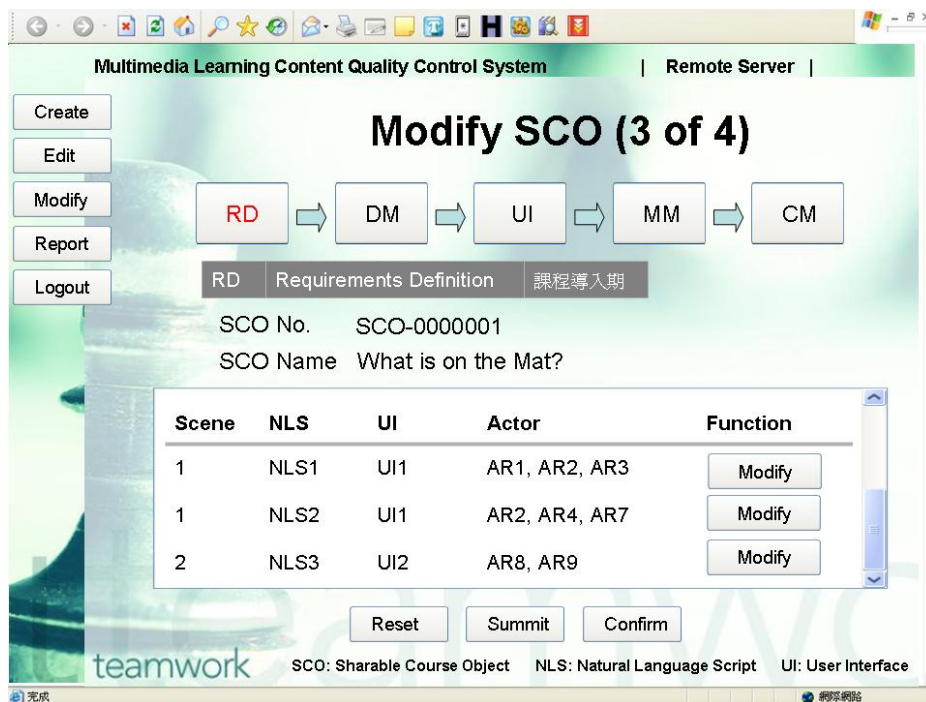


Figure 4.3: The screen shot of MLCs dependency editor (MDE)

As discussed in chapter 3, such a MLCs dependency editor (MDE) can operate as module to MQC System). The MDE would load the relevant elements (including natural language scripts, scenarios, user interfaces, actors and code modules) and display dependency relationships and properties on a screen. The annotator would interact with the users to prompt for links to appropriate concepts and properties every time a new element of the MLO is introduced in the editor. When the user finishes editing the specification, the annotator would generate MLCs dependency directed acyclic graphic (MDAG) automatically.

4.4 MLCs Change Detector

MLCs change detector (MCD) defines the set of activities that need to be performed when there are some new requirements or changes to existing requirements or original design (we will call both of these as changes in the requirements or original

design). Requirement changes can occur at any phases of the MLCs development process. The basic goal of MCD module is to detect requirement changes and minimize the impact of changes on the multimedia project. This involves understanding the full impact of a requirement change request, as well as the cumulative impact of changes, on the project. It also requires making the customer fully aware of the impact of the changes on the multimedia project so that changes in the negotiated terms can be done amicably. The MCD module, in a sense, tries to ensure that a multimedia project succeeds despite requirement changes (Figure 4.4).

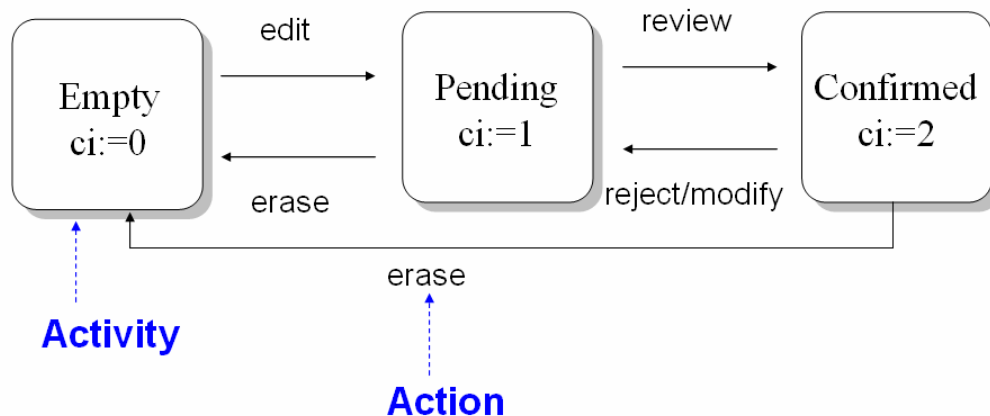


Figure 4.4: The state diagram of MLCs change detector (MCD)

4.5 MLCs Inconsistency Detector

The key functions of the MLCs Inconsistency Detector (MID) consist of managing: (1) data contained in multimedia learning object (MLO) nodes and (2) inconsistent detection for the structural connections among MLO nodes. Fundamental to these functions is some form of machinery that is able to determine whether a given specification violates a given set of consistency rules. This process directly involves basic MLO nodes and operations. (Figures 4.5) In a multimedia developing environment, MID services are provided at the consistency level. Aspects of the inconsistent detection

process that involves basic MLO should also be implemented at specific level. These basic inconsistent detection services may also be used by multimedia developers to provide inconsistent detection in changing original requirements or design of the MLO. A desirable and important characteristic for MLO-level inconsistent detection is that it should not reflect content aggregation (CA) level inconsistent detection policies.

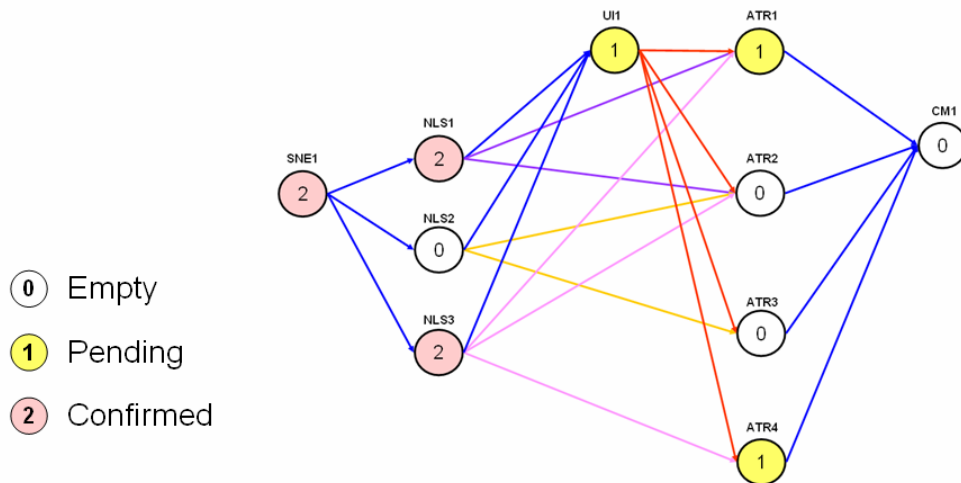


Figure 4.5: A MDAG of MLCs inconsistency detector (MID)

The broad definition of inconsistency as the breaking of a consistency rule means that there is an equally broad range of tools that support inconsistent detection. Process-centered environments on the other hand, check not only the artifacts of development, but also the process by which these artifacts are developed. In general however, most of these tools have limited inconsistency handling capabilities, concentrating instead on inconsistent detection and identification, and leaving inconsistency handling to be performed by the user of these tools. We have further developed the MLCs inconsistency detector (MID) to support the MQC system, and have used it as a vehicle for demonstrating the feasibility of our methodology (Figures 4.6). In the inconsistency detecting area, The MID provides a range of complementary mechanism.

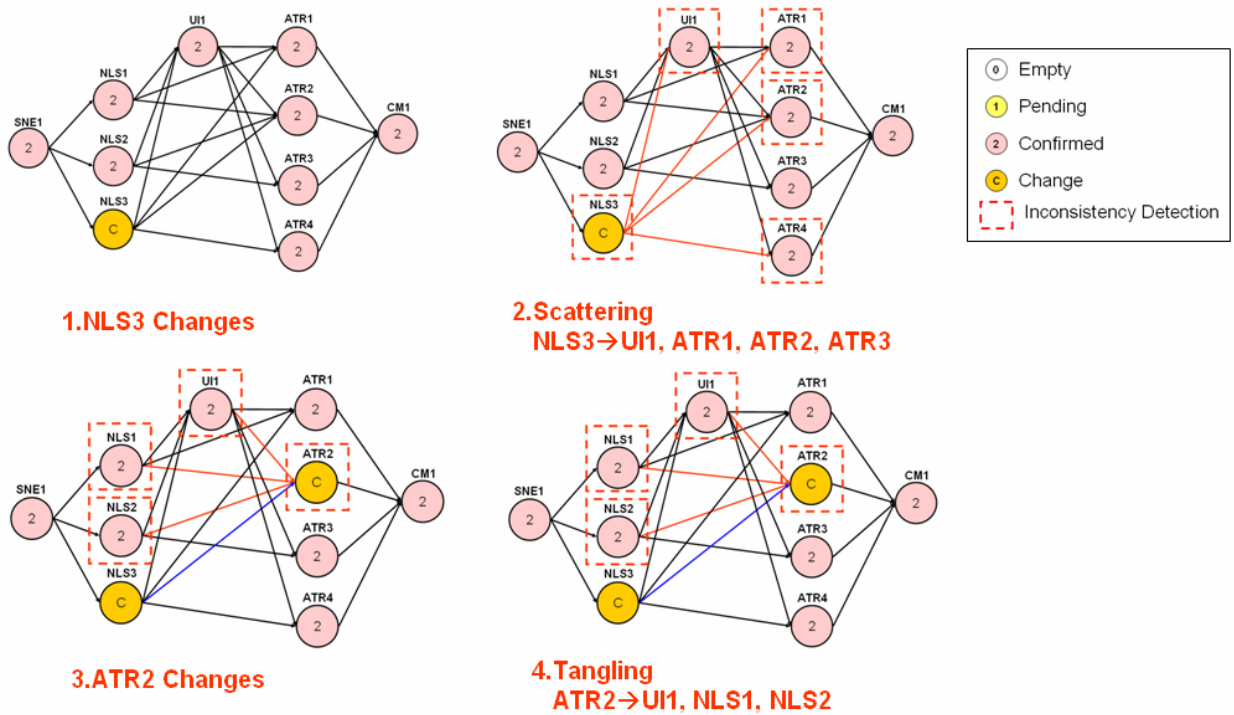


Figure 4.6: The change impact patterns of MID

From a MLCs designer or quality reviewer's point of view, The MDID facilitates the expression of inconsistency rules. In Figure 4.6 describes a sample multimedia inconsistency detector (MDID) in the multimedia development environment. This tool can be used to check inconsistency of partial specifications for internally (inter-scenes) and against other (intra-scenes). The particular rules that the developer wishes to check may be selected, and executed by clicking on the "Apply Inconsistency Checks" button. If one or more inconsistencies are detected, then clicking on the "Consistency Check" button invokes the appropriate inconsistent detection system (Figures 4.7).

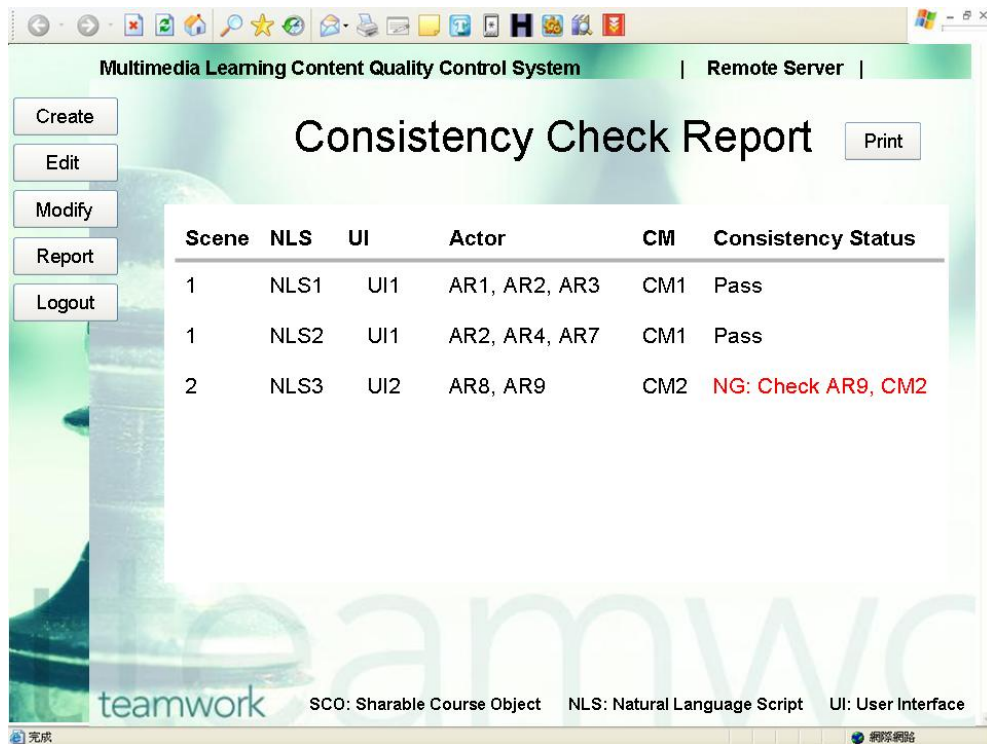


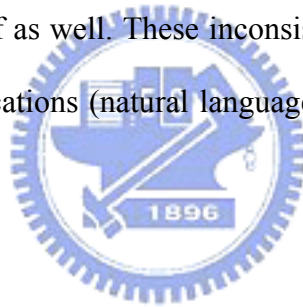
Figure 4.7: The screen shot of MLCs inconsistency detector (MID)



Chapter 5

An Usage Example for MQC System

We have gone through our methodology with an example in this chapter. In order to verify our methodology further, we apply the MLCs Quality Control (MQC) system to a multimedia learning contents (MLCs) case study “What is on the mat?” First of all, we will describe the requirements specifications dependency relationship from different stakeholders (designer, reviewer and supervisor). Then we will list all inconsistencies of these requirements specifications that were detected by MQC system based on the pre-defined rules. These inconsistencies exist between each two stakeholders and they exist in each stakeholder itself as well. These inconsistencies also occur among different types of requirements specifications (natural language scripts, scenarios, user interfaces, actors and code modules).



Step 1. Prepare the MLCs material

Designer can upload one's own MLCs material in web directly and store it in the database, if there is content of teaching material needs to be altered and which can be taken out and written again from the data base (Figure 5.1, Figure 5.2).

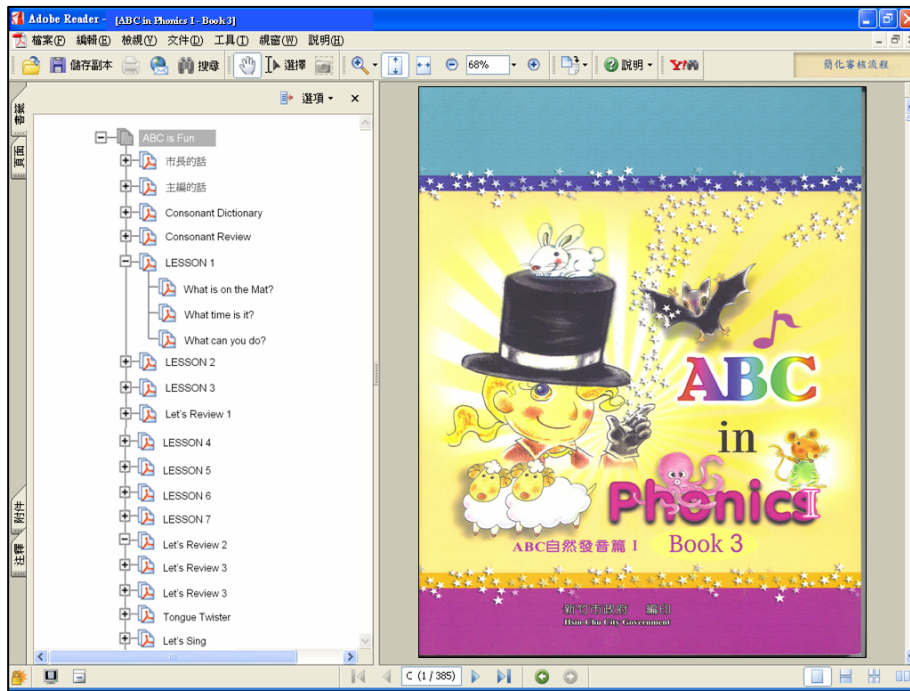


Figure 5.1: Example of original MLCs material

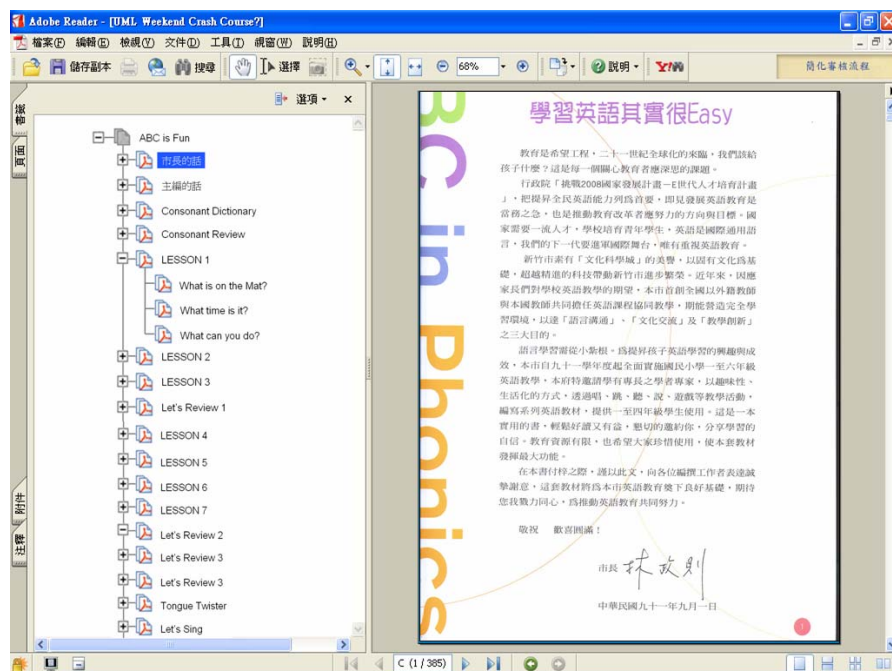


Figure 5.2: Selecting a section of original teaching material

Step 2. Analyze the MLCs story

Designer can select a story of MLCs material and analyze the structure of it by using SCORM standard (Figure 5.3).

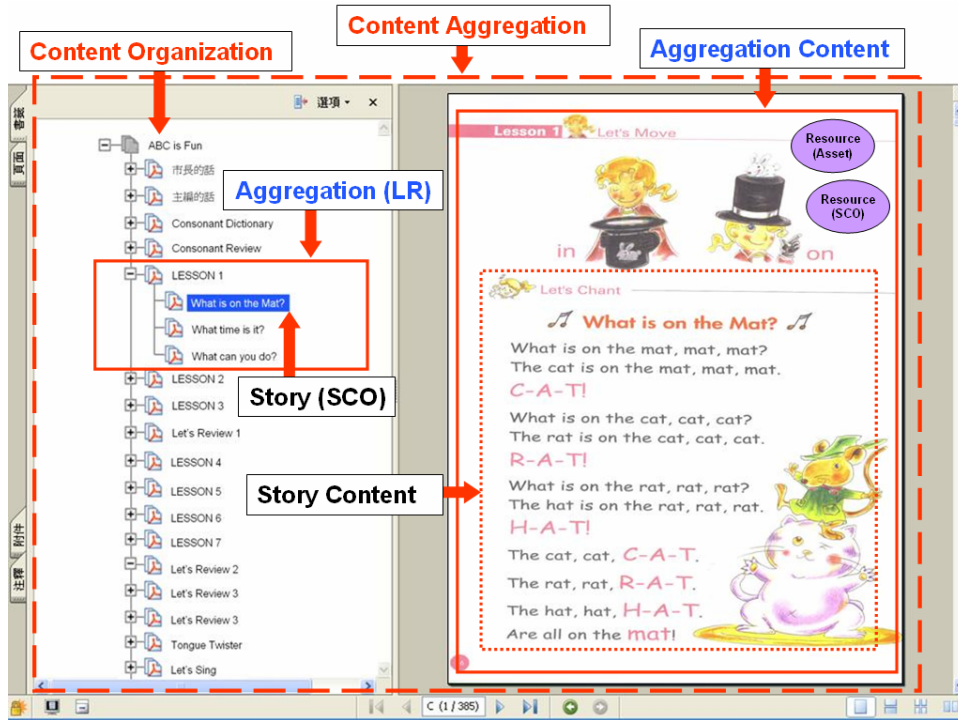


Figure 5.3: Analyzing a section of original teaching material

Step 3. Prepare the storyboard of MLCs story

Designer can edit storyboard of MLCs story by using general word editors, and store it in the web system database if there is content of storyboard needs to be altered and which can be taken out and written again from the data base (Figure 5.4, Figure 5.5, Figure 5.6, Figure 5.7).

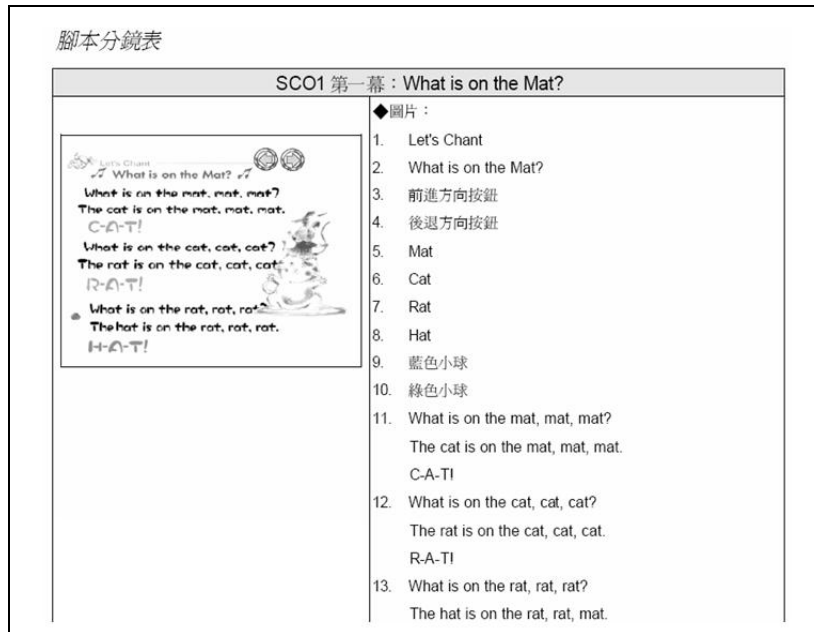


Figure 5.4: Example of the storyboard in primary draft design phase. (Part 1)

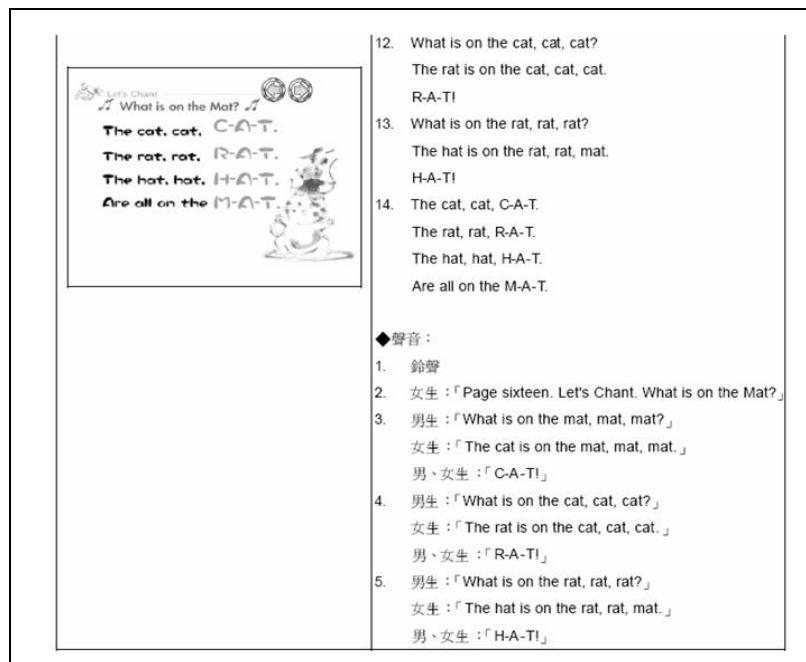


Figure 5.5: Example of the storyboard in primary draft design phase. (Part 2)



Figure 5.6: Example of the storyboard in primary draft design phase. (Part 3)



Figure 5.7: Example of the storyboard in primary draft design phase. (Part 4)

Step 4. Prepare the detail design of MLCs storyboard

Designer can edit detail storyboard of MLCs story by using general word editors, and store it in the web system database if there is detail content of storyboard needs to be altered and which can be taken out and written again from the data base (Figure 5.8, Figure 5.9, Figure 5.10).

腳本分鏡表 劇情場景設計				
SCO1 第一幕 主題：What is on the Mat?				
劇情	劇情所使用的演員 (備註一)			
	文字演員	聲音演員	圖片演員	影片演員
<p>開場劇情</p> <p>劇情一： 播放鈴聲[聲音 1]、Let's Chant[圖片 1]、What is on the Mat? [圖片 2]、前後方向按鈕[圖片 3+圖片 4]與 Mat[圖片 5]等四張圖片，隨開場場景出現。 女生說「Page sixteen. Let's Chant. What is on the Mat?」[聲音 2]</p>		SCO-1-S-1-A001 SCO-1-S-1-A002	SCO-1-S-1-P001 SCO-1-S-1-P002 SCO-1-S-1-P003 SCO-1-S-1-P004 SCO-1-S-1-P005	
<p>劇情二： 男生先發問「What is on the mat, mat, mat?」[聲音 3-1] 同時畫面出現本段對話[圖片 11]，並出現 Cat [圖片 6]於 Mat[圖片 5]上面，以及閃爍的藍綠色小球[圖片 9+圖片 10]跳到此段對話[圖片 11]左邊 女生回答說「The cat is on the mat, mat, mat.」[聲音 3-2] 男生和女生同時說「C-A-T」[聲音 3-3]</p>		SCO-1-S-1-A003	SCO-1-S-1-P005 SCO-1-S-1-P006 SCO-1-S-1-P009 SCO-1-S-1-P0010 SCO-1-S-1-P0011	
<p>劇情三： 男生先發問「What is on the cat, cat, cat?」[聲音 4-1] 同時畫面出現本段對話[圖片 12]，並出現 Rat [圖片 7]於 Cat[圖片 6]上面，以及閃爍的藍綠色小球[圖片 9+圖片 10]跳到此段對</p>		SCO-1-S-1-A004	SCO-1-S-1-P006 SCO-1-S-1-P007 SCO-1-S-1-P009 SCO-1-S-1-P0010 SCO-1-S-1-P0012	

Figure 5.8: Example of the storyboard in secondly detail design phase. (Part 1)

<p>話[圖片 12]左邊 女生回答說「The rat is on the cat, cat, cat.」[聲音 4-2] 男生和女生同時說「R-A-T」[聲音 4-3]</p>				
<p>劇情四： 男生先發問「What is on the rat, rat, rat?」[聲音 5-1] 同時畫面出現本段對話[圖片 13]，並出現 Hat [圖片 8]於 Rat[圖片 7]上面，以及閃爍的藍綠色小球[圖片 9+圖片 10]跳到此段對話[圖片 13]左邊 女生回答說「The hat is on the rat, rat, rat.」[聲音 5-2] 男生和女生同時說「H-A-T」[聲音 5-3]</p>		SCO-1-S-1-A005	SCO-1-S-1-P007 SCO-1-S-1-P008 SCO-1-S-1-P009 SCO-1-S-1-P0010 SCO-1-S-1-P0013	
<p>劇情五： 清除畫面上[圖片 9][圖片 10][圖片 11][圖片 12][圖片 13]</p>			SCO-1-S-1-P009 SCO-1-S-1-P010 SCO-1-S-1-P011 SCO-1-S-1-P012 SCO-1-S-1-P013	
<p>劇情六： 畫面出現對話[圖片 14] 男生說「The cat, cat, C-A-T.」[聲音 6-1] 女生說「The rat, rat, R-A-T.」[聲音 6-2] 男生說「The hat, hat, H-A-T.」[聲音 6-3] 男生和女生同時說「Are all on the M-A-T.」[聲音 6-4]</p>		SCO-1-S-1-A006	SCO-1-S-1-P014	

Figure 5.9: Example of the storyboard in secondly detail design phase. (Part 2)

互動劇情：			SCO-1-S-1-P003	
劇情一： 按前進方向按鈕[圖片 3]進入主選單				
劇情二： 按後退方向按鈕[圖片 4]進入第二幕			SCO-1-S-1-P004	
結尾劇情：				

備註一：演員編碼方式使用下列表示方式：SCO-編號-S-編號-演員類別代碼+編號
 舉例：文字演員 SCO-14-S-1-T002 代表在第十四個 Sharable Content Object (SCO)的第一個 Scene 裡面的第二個文字的文字的演員 T002
 聲音演員 SCO-14-S-1-A003 代表在第十四個 Sharable Content Object (SCO)的第一個 Scene 裡面的第三個聲音的演員 A003
 圖片演員 SCO-14-S-1-P002 代表在第十四個 Sharable Content Object (SCO)的第一個 Scene 裡面的第二個圖片的演員 P002
 影片演員 SCO-14-S-1-V001 代表在第十四個 Sharable Content Object (SCO)的第一個 Scene 裡面的第一個影片的演員 V001

Figure 5.10: Example of the storyboard in secondly detail design phase. (Part 3)

Step 5. Prepare the UI design of MLCs

Designer can design UI of MLCs story by using general word and graphic editors, and store it in the web system database if there is content of UI design needs to be altered and which can be taken out and written again from the data base (Figure 5.8, Figure 5.9, Figure 5.10).

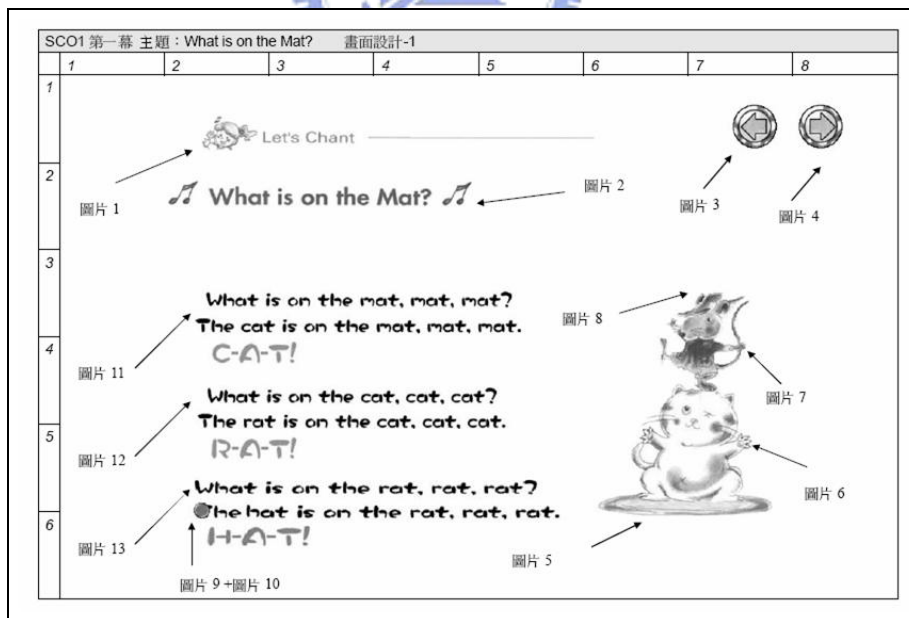


Figure 5.11: Example of the UI layout in secondly detail design phase. (Part 1)

Step 5. Prepare the multimedia actors list of MLCs

Designer can edit multimedia actors list of MLCs storyboard by using general

word editors, and store it in the web system database if there is content of multimedia actors list needs to be altered and which can be taken out and written again from the data base (Figure 5.12).

檔名	說明
聲音演員	
SCO-1-S-1-A001	: 鈴聲
SCO-1-S-1-A002	: 女生 : 「 Page sixteen. Let's Chant. What is on the Mat? 」
SCO-1-S-1-A003	: 男生 : 「 What is on the mat, mat, mat? 」, 女生 : 「 The cat is on the mat, mat, mat. 」, 男、女生 : 「 C-A-T! 」
SCO-1-S-1-A004	: 男生 : 「 What is on the cat, cat, cat? 」, 女生 : 「 The rat is on the cat, cat, cat. 」, 男、女生 : 「 R-A-T! 」
SCO-1-S-1-A005	: 男生 : 「 What is on the rat, rat, rat? 」, 女生 : 「 The hat is on the rat, rat, mat. 」, 男、女生 : 「 H-A-T! 」
SCO-1-S-1-A006	: 男生 : 「 The cat, cat, C-A-T. 」, 女生 : 「 The rat, rat, R-A-T. 」, 男生 : 「 The hat, hat, H-A-T. 」, 男、女生 : 「 Are all on the M-A-T. 」
圖片演員	
SCO-1-S-1-P001	: Let's Chant
SCO-1-S-1-P002	: What is on the Mat?
SCO-1-S-1-P003	: 前進方向按鈕
SCO-1-S-1-P004	: 後退方向按鈕
SCO-1-S-1-P005	: Mat
SCO-1-S-1-P006	: Cat
SCO-1-S-1-P007	: Rat
SCO-1-S-1-P008	: Hat
SCO-1-S-1-P009	: 藍色小球
SCO-1-S-1-P0010	: 綠色小球
SCO-1-S-1-P0011	: What is on the mat, mat, mat? The cat is on the mat, mat, mat. C-A-T!
SCO-1-S-1-P0012	: What is on the cat, cat, cat? The rat is on the cat, cat, cat. R-A-T!
SCO-1-S-1-P0013	: What is on the rat, rat, rat? The hat is on the rat, rat, mat. H-A-T!
SCO-1-S-1-P0014	: The cat, cat, C-A-T. The rat, rat, R-A-T. The hat, hat, H-A-T. Are all on the M-A-T.

Figure 5.12: Example of the multimedia actor list in secondly detail design phase

Step 6. Prepare the detail UI design of MLCs

Designer can edit detail UI design of MLCs story by using general word and graphic editors, and store it in the web system database if there is content of UI detail design needs to be altered and which can be taken out and written again from the data base (Figure 5.13).

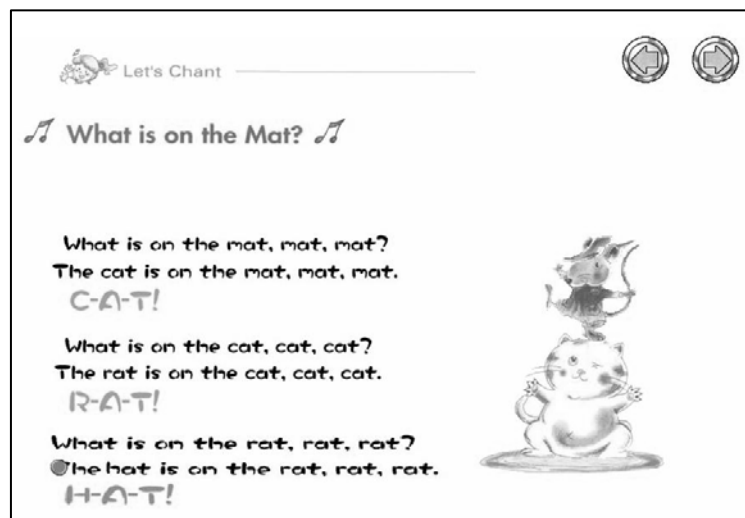


Figure 5.13: The screen shot of a multimedia learning object “What is on the mat?”

Step 7. Login Web-based MQC system

Designer can input user account and password to login MQC system, and edit documents and multimedia actors stored in the web system database (Figure 5.14).



Figure 5.14: The screen shot of MQC system “Login” module

Step 8. Create multimedia components of MLCs

Designer can create multimedia component of MLCs by using MQC “Create” module, and store it in the web system database if there is content of multimedia actors list needs to be altered and which can be taken out and written again from the data base (Figure 5.15).

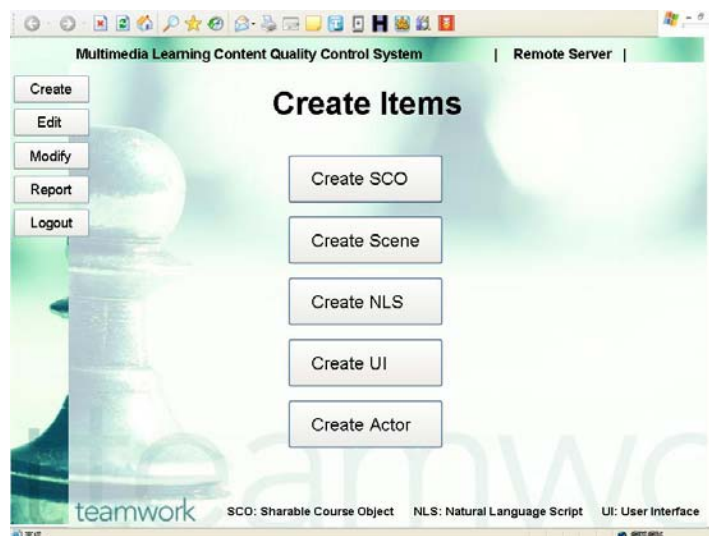


Figure 5.15: The screen shot of MQC system “Create” module

Step 9. Edit multimedia components of MLCs

Designer can edit multimedia component of MLCs by using MQC “Edit” module, and store it in the web system database if there is content of multimedia actors list needs to be altered and which can be taken out and written again from the data base (Figure 5.16).

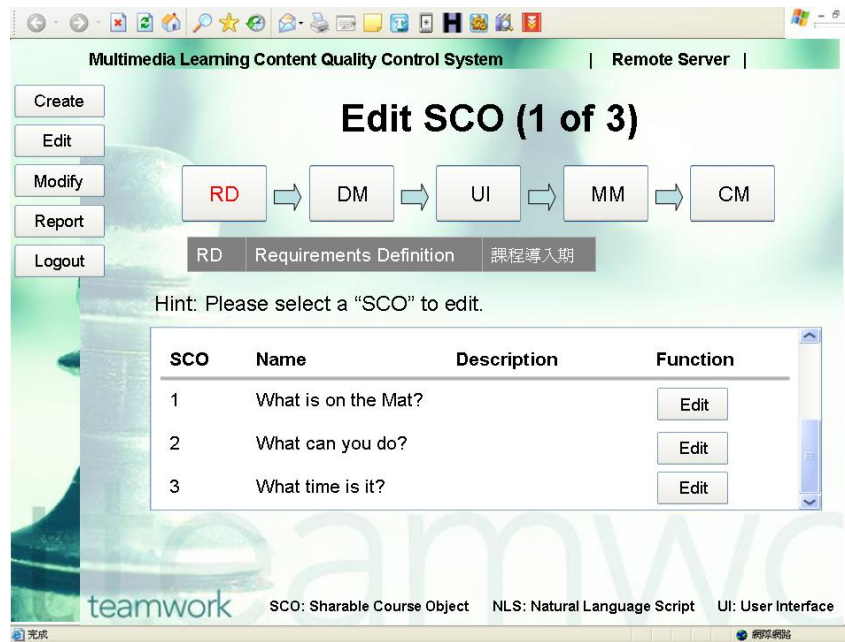


Figure 5.16: The screen shot of MQC system “Edit” module

Step 10. Modify multimedia components of MLCs

Designer can edit multimedia component of MLCs by using MQC “Modify” module, and store it in the web system database if there is content of multimedia actors list needs to be altered and which can be taken out and written again from the data base (Figure 5.17).

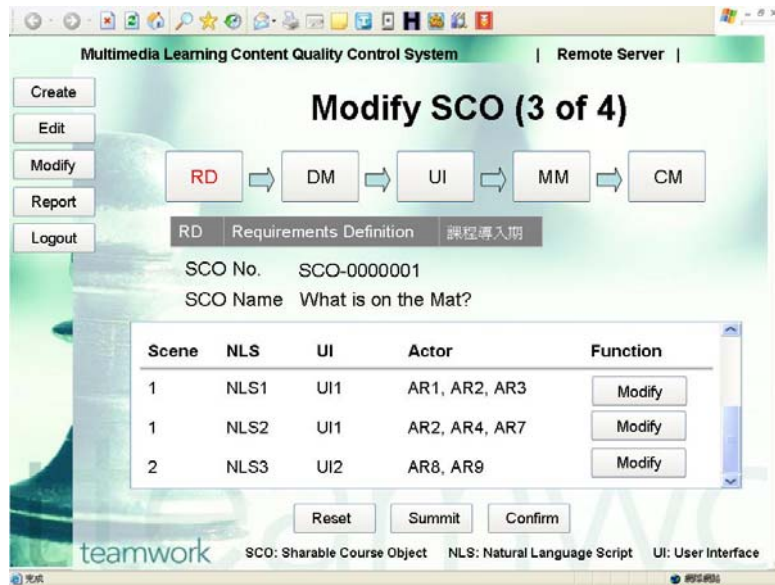


Figure 5.17: The screen shot of MQC system “Modify” module

Step 11. MQC system generate MDAG of MLCs

MQC system can generate a MDAG data structure of MLCs in web system database (Figure 5.18).

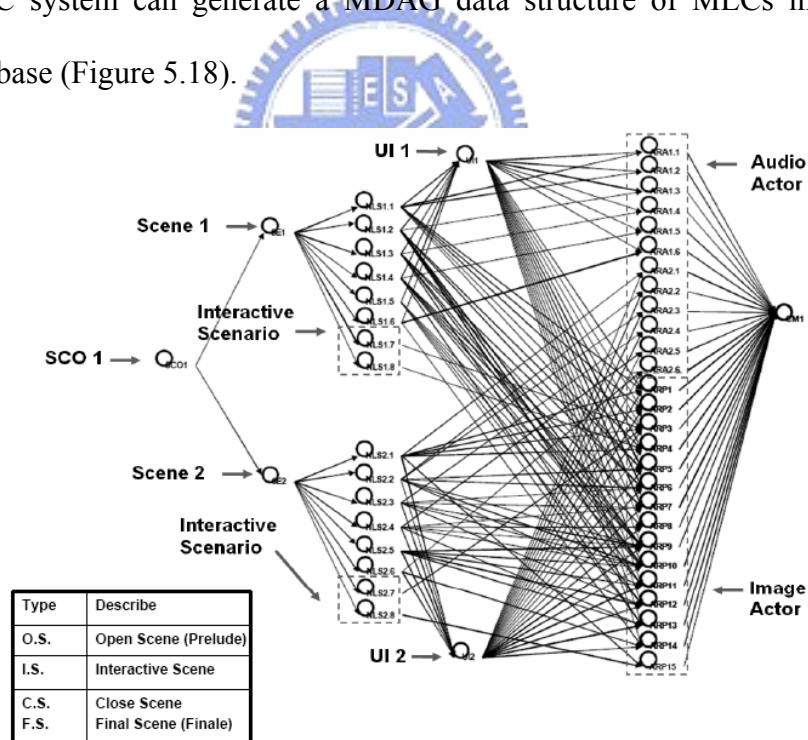


Figure 5.18: The MLCs directed acyclic graph (MDAG) of a multimedia learning object “What is on the mat?”

Step 12. Create the multimedia actors of MLCs

Designer can edit detail multimedia actors of MLCs by using general sound,

music, video, text and graphic editors, and store it in the web system database if there is content of multimedia actors needs to be altered and which can be taken out and written again from the data base (Figure 5.19).

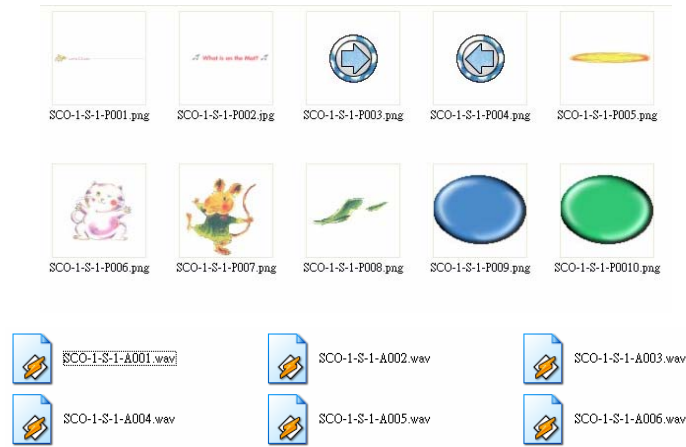


Figure 5.19: The screen shot of multimedia actors

Step 13. Create the SCO files of MLCs

Designer can edit SCO files of MLCs by using multimedia authoring tools (Flash, PowerPoint, Authorware, Virtual Basic, Director, 編輯手...), and store it in the web system database if there is content of multimedia actors needs to be altered and which can be taken out and written again from the data base (Figure 5.20).

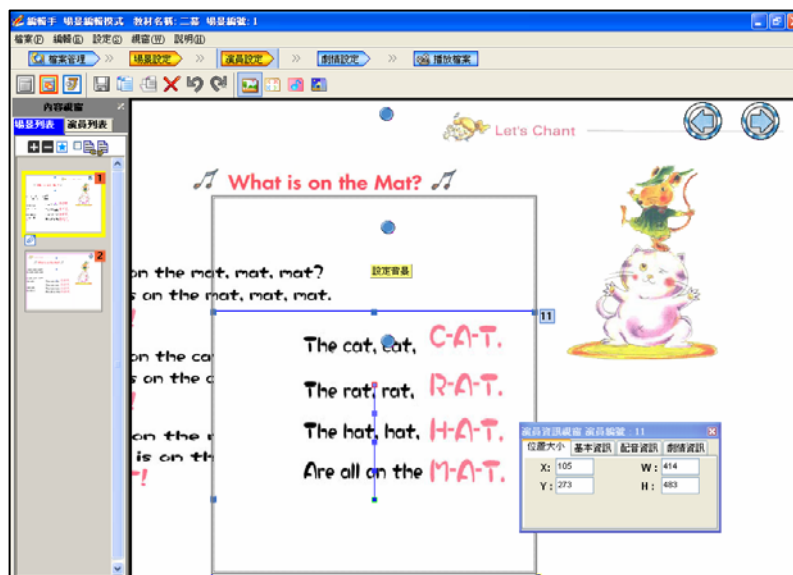
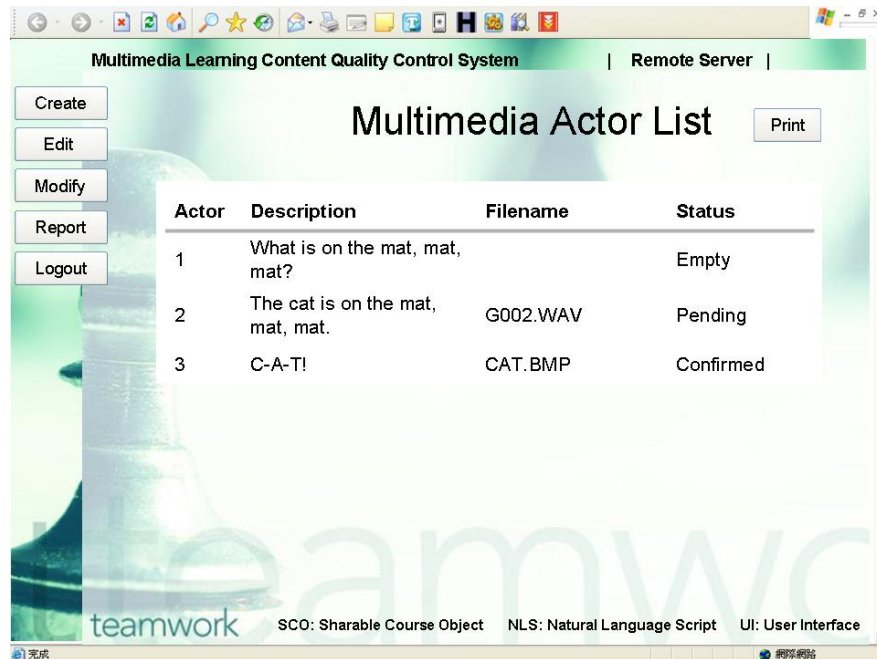


Figure 5.20: The screen shot of a multimedia authoring tool

Step 14. Create the multimedia actors list report of MLCs

Designer can create detail multimedia actors list report of MLCs by using MQC module (Figure 5.21).



The screenshot displays the 'Multimedia Actor List' report within the 'Multimedia Learning Content Quality Control System'. The interface includes a navigation menu on the left with buttons for 'Create', 'Edit', 'Modify', 'Report', and 'Logout'. The main content area features a table with the following data:

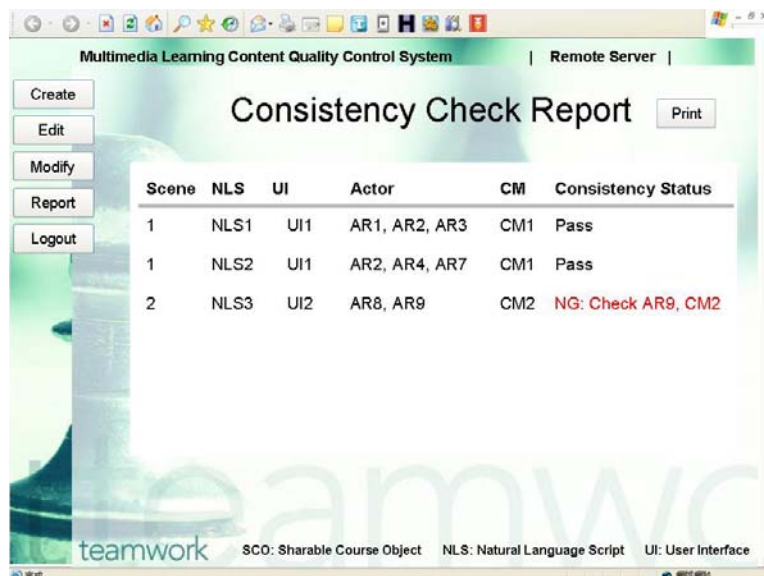
Actor	Description	Filename	Status
1	What is on the mat, mat, mat?		Empty
2	The cat is on the mat, mat, mat.	G002.WAV	Pending
3	C-A-T!	CAT.BMP	Confirmed

The footer of the application window includes the 'teamwork' logo and the following text: 'SCO: Sharable Course Object NLS: Natural Language Script UI: User Interface'.

Figure 5.21: The screen shot of multimedia actor list report

Step 14. Create the inconsistency report of MLCs

Designer can create inconsistency report of MLCs by using MQC module (Figure 5.21).



The screenshot displays the 'Consistency Check Report' within the 'Multimedia Learning Content Quality Control System'. The interface includes a navigation menu on the left with buttons for 'Create', 'Edit', 'Modify', 'Report', and 'Logout'. The main content area features a table with the following data:

Scene	NLS	UI	Actor	CM	Consistency Status
1	NLS1	UI1	AR1, AR2, AR3	CM1	Pass
1	NLS2	UI1	AR2, AR4, AR7	CM1	Pass
2	NLS3	UI2	AR8, AR9	CM2	NG: Check AR9, CM2

The footer of the application window includes the 'teamwork' logo and the following text: 'SCO: Sharable Course Object NLS: Natural Language Script UI: User Interface'.

Figure 5.22: The screen shot of inconsistency report

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, a MQC system which consists the file management of multimedia learning contents (MLCs) actors and the inconsistent detection mechanism of MLCs components are proposed and a server-client system is also provided for helping other system designers to extend their MQC functionalities. Our main contribution of this thesis is to construct a MLCs quality control mechanism on inconsistent detection for MLCs components and satisfy the CMMI-compliant specific practices on MLCs developing process as well as an Web-based platform for MLCs designers and quality review stakeholder with the MLCs work in process documents and media file. Without the proposed system, MLCs components are easily causing inconsistency problems. That results in the embarrassing situation for the design staff of endless required debugs, causes the delay of due-date, and increases development costs and the loss of market competitiveness for the company. The MQC system can keep this vicious tendency in prevent.

6.2 Future Work

There are some future works about quality controlling facilities of this MQC system listed and discussed as follows:

1. Adopt more CMMI practices relative to the MQC mechanism

New technology will bring the new interaction on the MLCs creators and learners

in the future. It may be a whole new multimedia learning object, or a new e-learning user interface with the appearance of new technological innovation. MLCs designer and quality review stakeholders will need more CMMI-compliant practices relative to the quality control mechanism in the MLCs developing process.

2. The enhance version control mechanism for managing components of MLCs in MQC process.

The version control mechanism is a hot issue in the software engineering research fields. People still need a good solution for managing the different versions of MLCs components in MQC process. The enhanced version control mechanism on MLCs components will pay attention to the companies of e-learning contents provider.

3. Support large amount nodes of the MDAG

In order to actualize the management of the curriculum-level MLCs, the inconsistency information used in large amount nodes of the MDAG should be precisely recorded. Unfortunately, the extant recording methods cannot guarantee the integrity of the using MLCs inconsistency information. Therefore, a more comprehensive and powerful method is needed to insure the MQC system in the curriculum-level MLCs developing process.

Reference

- [1] NASA SEL, Recommended Approach to Software Development: Revision 3, 1992.
- [2] Finkelstein et al., “Inconsistency handling in multi-perspective specifications.” IEEE Trans. on Software Engineering, 20:569–578, 1994.
- [3] S. Easterbrook and B. Nuseibeh, “Using viewpoints for inconsistency management.” Software Engineering Journal, 11:31–43, 1996.
- [4] R.C. Lee and W.M. Tepfenhart, UML and C++: A Practical Guide to Object-Oriented Development, 1st Edition, Prentice Hall.
- [5] Sarah A. Sheard, “Software Productivity Consortium”, 1997.
- [6] Cormen, Thomas H. Leiserson, Charles Eric. Rivest, Ronald L., Introduction to algorithms, Cambridge, Mass.: MIT Press, 2001.
- [7] Pankaj J., CMM in Practice: Processes for Executing Software Projects at Infosys, Addison-Wesley, Inc., June 2001.
- [8] CMMI Product Team, Introduction to CMMI-Continuous V 1.1, SEI, Carnegie Mellon University, August 2002.
- [9] CMMI Product Team, CMMI for Software Engineering (CMMI-SW, V1.1) Staged Representation, CMU/SEI-2002-TR-029, SEI, Carnegie Mellon University, August 2002.
- [10] Bohner, S.A., Gracanin, D., “Software impact analysis in a virtual environment”, Software Engineering Workshop, 2003. Proceedings. 28th Annual NASA Goddard, pp. 143- 151, 2003.
- [11] Margaret Kulpa, Kent A. Johnson, Interpreting the CMMI: A Process Improvement Approach, April 2003.
- [12] The Standish Group International Inc., “CHAOS Chronicles v3.0. West Yarmouth”, USA, 2003.

- [13] Mary Beth Chrissis, Mike Konrad, Sandy Shrum, CMMI: Guidelines for Process Integration and Product Improvement, Addison-Wesley, Inc., Feb 2003.
- [14] Dennis M. Ahern, Aaron Clouse, Richard Turner, CMMI Distilled: A Practical Introduction to Integrated Process Improvement, 2nd ed, Addison-Wesley, Inc., Sep 2003.
- [15] Goldenson, D.R., Gibson, D.L., “Demonstrating the Impact and Benefits of CMMI: An Update and Preliminary Results”; CMU/SEI-2003-SR-009; 2003.
- [16] Ian Sommerville, Software Engineering, 7th ed, Addison-Wesley, 2004.
- [17] Geoff Draper, Rickhefner Ph.D., “Applying CMMI Generic Practices with Good Judgment”, SEPG Conference Tutorial, March 2004.
- [18] IEEE Learning Technology Standards Committee (LTSC) IEEE P1484.12 Learning Objects Metadata Working Group, [On-line]. Available: <http://ltsc.ieee.org/wg12/>
- [19] CMMI Mappings and Comparisons, [On-line]. Available:<http://www.sei.cmu.edu/cmmi/adoption/comparisons.html>, 2006.
- [20] Advanced Distributed Learning Department of Defense (DoD), [On-line]. Available: <http://www.adlnet.org>
- [21] Sharable Content Object Reference Model: The SCORM Book 1: The SCORM Overview, [On-line]. Available: <http://www.adlnet.org>
- [22] Sharable Content Object Reference Model: The SCORM Book 2: The SCORM Content Aggregation Model, [On-line]. Available: <http://www.adlnet.org>
- [23] Sharable Content Object Reference Model: The SCORM Book 3: The SCORM Run-Time Environment, [On-line]. Available: <http://www.adlnet.org>
- [24] Kuei-Jung Chung, “The Design and Implementation of a Content Quality Control Process and Management System for SCORM-based Multimedia Curriculum” Master Thesis of N.C.T.U. Taiwan, 2006.