



# **Converting PC SCORM to Pocket SCORM Objects (A Comparative Study between Compression Tools)**

تحويل محتويات وكائنات سكورم للاجهزة الشخصية الى سكورم يدوي شخصي (مقارنة بين  
ادوات ضغط البيانات)

**Prepared by**  
Mosa Theeb Omar Abu Lawi

**Supervisor**  
Dr. Omar al Bashier

**Co supervisor**  
Dr. Ezz Hattab


**This Thesis Submitted in Partial Fulfillment of the  
Requirements for the Degree of Master in  
Computer Science**

**Graduate College of Computing studies  
Amman Arab University for Graduated Studies  
2008  
Amman – Jordan**

**DELEGATION**

I, the undersigned "Mosa Theeb Omar Abu Lawi" authorize hereby Amman Arab University for Graduated Studies to provide copies of this thesis to libraries, institutions, agencies, organizations, individuals and any other parties upon their request.

Name: Mosa Theeb Omar Abu Lawi

Signature: 

Date: 12/4/2009

© Copyright by Amman Arab University for Graduated Studies

## APPROVAL

This Thesis titled "Converting PC SCORM to Pocket SCORM objects (A comparative study between compression tools) ", has been successfully defended and approved by the examining committee on 20, 1, 2009.

Prof. Mohammed Ahmed M. Al - Fayoumi

Dr., Chair.

Dr. Ezz Hattab, Member and Co supervisor

Dr. Mezher Shaban Al Anie

Member

## Acknowledgment

My praises and thanks to wisdom donator ALLAH, who provide me the willpower to complete this thesis. Then I would like to thank so much my supervisor Dr Omar Al Bashier and co supervisor Dr Ezz Hattab for their constant encouragement and technical advices to produce this work.

I also thank the examining committee, all my colleagues and relatives for support me throughout always to give my best. Therefore, I present the continuously thanks to all lecturers, administration and staff of Amman Arab University for their help.

Much thanks to my patient father for his praying and providing supports and special thanks to my beloved wife who kept close to me all the time. My brothers and sisters thank you very much.

There are many friends who have made this work possible that they can not be listed in one page, my thanks to them.

## DEDICATION

I dedicate this work to ... The soul of my mother, patient father, beloved wife, dearest children, impacted brothers and sisters, and supported friends.

**Thesis title: Converting PC SCORM to Pocket SCORM  
Objects**

**(A Comparative Study between Compression Tools)**

**Student :** Mosa Theeb Omar Abu Lawi

**Supervisor :** Dr. Omar al Bashier

**Co supervisor:** Dr. Ezz Hattab

**ABSTRACT**

In World Wide Web, there are millions of developed learning objects that are available in different platforms and stored in different Learning Content Management Systems (LCMS). However, a very limited interoperability is offered by such learning objects since they have been developed with no standards.

Therefore, the worldwide industry and governments have collaborated for collecting and organizing learning content into uniform digital packages to enhance the degree of interoperability among Learning Content Management Systems (LCMS). Sharable Content Object Reference Model (SCORM) is a collection of standards and specifications for web-based Elearning. It defines how content may be packaged into a transferable ZIP file.

This thesis discusses how content may be packaged into a transferable ZIP file among mobile appliances (i.e. PDA SCORM). Timothy K. Shih [8], proposed an algorithm to pack

SCORM standards that have been used for desktop applications (PC SCORM) into a SCORM learning objects that are used for mobile applications (PDA SCORM). The proposed algorithm is implemented in this thesis using C language. A number of learning objects have been packed from (PC SCORM) into (PDA SCORM) using the developed tool. The accuracy of conversion was 95%.

This thesis compared four different techniques for compression: WinZip, WinRar, Power Archiver, and WinAce. Based on the experimental results of compressing different mobile objects, it is found that WinAce is the best technique for compressing the shared content that are used in PDA SCORM.

تحويل محتويات وكائنات سكورم للاجهزة الشخصية الى سكورم يدوي شخصي (مقارنة بين ادوات ضغط البيانات)

اعداد الطالب : موسى ذيب عمر ابو لوي

المشرف : الدكتور عمر البشير

المشرف المساعد : الدكتور عز الدين حطاب

## Arabic Summary

### الملخص

في عالم الانترنت يوجد ملايين من كائنات التعليم (Learning Objects) التي تعمل في بيئات مختلفة وتخزن في مختلف انظمة ومحتويات التعليم المختلفة (LCMS) ولكن هذه الكائنات لا تستطيع الاستفادة من بعضها البعض (interoperability) وذلك لعدم وجود معايير قياسية لتكوين هذه الكائنات.

استطاع القطاع الحكومي وبالتعاون مع القطاع الخاص انشاء معايير قياسية بحيث تستطيع الكائنات التعليمية الاستفادة من بعضها البعض وتم انشاء معيار سكورم والذي هو عبارة عن مجموعة من المعايير والاصاف القياسية في التعليم الالكتروني بحيث نستطيع تجميع المحتوى بملف (ZIP file)

في هذه الرسالة سوف يتم توضيح الية تجميع المحتوى في ملف (ZIP file) من خلال Mobile appliances استطاع البروفيسور تومثي من انشاء خوارزمية لتجميع بيئة سكورم على الاجهزة الشخصية الى بيئة سكورم على اجهزة الموبايل وهذه الخوارزمية تم تطبيقها في هذه الرسالة باستخدام لغة البرمجة سي حيث تم تجميع مجموعة من كائنات التعليم المستخدمة في اجهزة الكمبيوتر الشخصية الى نظام الموبايل وكانت نسبة النجاح 95%

سوف يتم مقارنة مجموعة من ادوات ضغط البيانات في هذه الرسالة ( WinZip, WinRar, Power Archiver, WinAce ) ومن خلال التجربة بضغط مجموعة من كائنات التعليم الخاصة بالموبايل تم اختيار اداة WinAce كافضل اداة لضغط الكائنات الخاصة بنظام سكورم الخاص بالموبايل كافضل اداة لضغط الكائنات الخاصة بنظام سكورم الخاص بالموبايل



## Table of Contents

ACKNOWLEDGMENT.....	IV
ABSTRACT.....	VI
ARABIC SUMMARY .....	VIII
TABLE OF CONTENTS.....	IX
ACRONYMS AND ABBREVIATIONS.....	XII
LIST OF TABLES.....	XIV
CHAPTER ONE INTRODUCTION.....	1
1.1. PREFACE.....	1
1.2. PROBLEM STATEMENT .....	3
1.3. AIMS OF THESIS.....	3
1.4. SUGGESTED SOLUTION.....	4
1.5. RESEARCH METHODOLOGY.....	4
1.6. CONTRIBUTIONS .....	4
1.7. THESIS ORGANIZATION .....	5
<b>CHAPTER TWO PREVIOUS RELATED WORK .....</b>	<b>7</b>
2.1. INTRODUCTION .....	7
2.2. RELATED SCORM STUDIES AND WORKS .....	7
2.3. RELATED MOBILE ELEARNING STUDIES AND WORKS.....	9
2.4. RELATED COMPRESSION AND PACKING DATA STUDIES AND WORKS.....	11
<b>CHAPTER THREE APPLIED TECHNOLOGY.....</b>	<b>13</b>
3.1. INTRODUCTION TO SHARABLE CONTENT OBJECT REFERENCE MODEL .....	13
3.2. HIGH LEVEL REQUIREMENTS.....	13
3.3. SCORM COMPLIANT .....	16
3.3.1. A LEARNING MANAGEMENT SYSTEM (LMS) .....	16
3.3.2. SHAREABLE CONTENT OBJECT (SCO) .....	17
3.3.3. CONTENT AGGREGATION .....	18
3.4. SCORM CONTENT AGGREGATION MODEL (CAM).....	19
3.4.1. CONTENT MODEL: .....	19

3.4.2. CONTENT PACKAGING:.....	22
3.5. SCORM RUN-TIME ENVIRONMENT (RTE).....	24
3.5.1. RUN-TIME ENVIRONMENT TEMPORAL MODEL .....	25
3.6. APPLICATION PROGRAMMING INTERFACE .....	29
3.6.1. API OVERVIEW .....	29
3.6.2. API METHODS AND SYNTAX .....	31
3.7. THE SEQUENCING DEFINITION MODEL .....	31
3.8. SEQUENCING CONTROL MODES.....	32
3.9. PERSONAL DIGITAL ASSISTANT (PDA).....	32
3.9.1. INTRODUCTION .....	33
3.9.2 TYPES OF PDAS.....	33
3.9.2.1. TRADITIONAL PDAS.....	33
3.9.2.2.. PALM PDAS .....	34
.3.9.2.3. POCKET PC'S.....	34
3.9.2.4. SMARTPHONES.....	34
3.9.3. <i>Advantages of PDA</i> .....	35
3.9.4. <i>PDA Components</i> .....	38
<b>CHAPTER FOUR SCORM-COMPRESSION TOOLS: A COMPARATIVE STUDY .....</b>	<b>43</b>
4.1. INTRODUCTION .....	43
4.2. POCKET SCORM .....	43
4.3. THESIS SOFTWARE .....	49
4.4. PACKING DATA.....	51
4.1.1 <i>Comparison between tools</i> .....	57
4.5 COMPARISON RESULT .....	81
<b>CHAPTER FIVE IMPLEMENTING THE DEFLATE ALGORITHMS USING VC#.....</b>	<b>84</b>
5.1. INTRODUCTION .....	84
5.2. THE HUFFMAN'S CODING ALGORITHM.....	84
5.3. LZ77 COMPRESSION .....	86
5.4. DEFLATE ALGORITHMS PROGRAMS .....	94
<b>CHAPTER SIX CONCLUSION AND FUTURE WORK ....</b>	<b>102</b>
6.1. INTRODUCTION .....	102

6.2. RESEARCH CONCLUSIONS .....	102
6.2.1. IN GENERAL .....	103
6.3. RECOMMENDATIONS FOR FUTURE WORK .....	104
6.3.1. <i>Future work for the research</i> .....	104
<b>BIBLIOGRAPHY .....</b>	<b>105</b>

## Acronyms and Abbreviations

ADL	Advanced Distributed Learning
API	Application Programming Interface
CAM	Content Aggregation Model
CPU	Central Processing Unit
GPS	Global Positioning System
HTML	Hyper Text Markup Language
IR	Infrared
JPEG	Joint Photographic Experts Group
LCD	Liquid-Crystal Display
LMS	Learning Management System
MPEG	Moving Pictures Expert Group
PC	Personnel Computer
PDA	Personal Digital Assistant
PIF	Package Interchange File
PIM	Personal Information Management
RAM	Random Access Memory
RLE	Run Length Encoding
ROM	Read Only Memory
RTE	Run Time Environment

RTE	Runtime Environment
RTS	Run Time Services
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SOAP	Simple Object Access Protocol
TFT	Thin-Film Transistor
URI	Universal Resource Indicator
WAN	Wide Area Networks
WAV	Waveform Audio Format
WMA	Windows Media Audio

## List of Tables

Table 4.1 – Support of other archive formats.....	59
Table 4.2 – Support of solid archives and others.....	59
Table 4.3 – Support of operating system .....	60
Table 4.4 - Different files to be packed using WinRar tools	61
Table 4.5 - JPG files to be packed using WinRar tools.....	62
Table 4.6 - HTML files to be packed using WinRar tools ....	64
Table 4.7 - HTML files to be packed using WinZip tools.....	66
Table 4.8 - JPG files to be packed using WinRar tools.....	67
Table 4.9 - HTML files to be packed using WinZip tools.....	69
Table 4.10 - Different files to be packed using WinAce tools .....	71
Table 4.11 - JPG files used to packed using WinAce tools	72
Table 4.12 - HTML files to be packed using WinAce tools .	74
Table 4.13 - Different files to be packed using Power Archiver tools .....	76
Table 4.14 - JPG files to be packed using Power Archiver tools.....	78
Table 4.15 - HTML files to be packed using Power Archiver tools.....	79
Table 4.16 – Packing data result .....	82
Table 5.1– Huffman coding .....	85

Table 5.2– Encoding string using LZ77 .....	88
Table 5.3 – Decoding a stream of triples using LZ77 algorithms.....	90
Table 5.4 - Different files to be packed using deflate algorithms.....	96
Table 5.5 - JPG files to be packed using deflate algorithms .....	98
Table 5.6 - HTML files used to compress by deflate algorithms.....	99
Table 5.7 - Comparison between WinAce and the deflate algorithms.....	101

## List of Figures

Figure 3.1 - Example of Interoperability .....	14
Figure 3.2 - Example of accessibility.....	14
Figure 3.3 - Example of reusability.....	14
Figure 3.4 - Example of durability .....	15
Figure 3.5 - Example of maintainability .....	15
Figure 3.6 - Example of adaptability.....	16
Figure 3.7 - Sample assets .....	17
Figure 3.8 - Sample of SCO.....	18
Figure 3.9 - Sample of content aggregation.....	18
Figure 3.10– Example of Assets .....	19
Figure 3.11 – Conceptual Representation of Activities .....	21
Figure 3.12– Conceptual Illustration of a Content Organization.....	21
Figure 3.13– Conceptual Illustration of a Content Aggregation.....	22
Figure 3.14– Components of a Manifest.....	23
Figure 3.15– Single Learner Attempt with one Learner Session .....	27
Figure 3.16- learner attempt spread over several learner session .....	28



Figure 3.17- successive learner attempts ,spread over several learner session .....	28
Figure 3 -18.Illustration of API, API Instance and API Implementation.....	30
Figure 3.19- Component of PDA.....	39
Figure 4.1- Relationship of Components within Pocket SCORM Architecture.....	44
Figure 4.2 - Save as SCORM .....	50
Figure 4.3 - Specify the property of SCORM .....	50
Figure 4.4 - Set the appearance option.....	50
Figure 4.5 - Lossy compression example. A) High quality JPG image. B) Low quality JPG image. ....	52
Figure 4.6 - Sample of SCORM ELearning objects .....	58
Figure 4.7 –Packing Different Files Using WinRar Tools ....	62
Figure 4.8 – Packing JPG Files Using WinRar Tools.....	63
Figure 4.9 – Packing HTML Files Using WinRar Tools.....	65
Figure 4.10 – Packing Different Files Using WinZip Tools..	67
Figure 4.11 – Packing JPG Files Using WinZip Tools .....	68
Figure 4.12 – Packing HTML Files Using WinZip Tools .....	70
Figure 4.13 – Packing Different Files Using WinAce Tools	72
Figure 4.14 – Packing JPG Files Using WinAce.....	73

Figure 4.15 – Packing HTML Files Using WinAce .....	75
Figure 4.16 – Packing Different Files Using Power Archiver .....	77
Figure 4.17 - Packing JPG Files Using Power Archiver.....	79
Figure 4.18- Packing HTML Files Using Power Archiver ....	81
Figure 5.1- Huffman tree .....	85
Figure 5.2- create XIP files .....	94
Figure 5.3- Write the name of the file you want to create ...	95
Figure 5.4– Packing Different Files Using Deflate Algorithms .....	97
Figure 5.5– Packing JPG Using Deflate Algorithms .....	99
Figure 5.6- Packing HTML Using Deflate Algorithms .....	100

# Chapter One

## Introduction

### 1.1. Preface

Elearning is becoming the most effective paradigm for spreading knowledge among distributed heterogeneous audiences [23,24]. In order to facilitate communication and knowledge sharing among Elearning platforms, standards have been proposed for describing content metadata, course navigation, and online interactions. Advanced Distributed Learning Initiative (ADL) proposed SCORM (Sharable Content Object Reference Model) to being an international standard of Elearning system. And it had become the most popular standard among various learning management systems. SCORM enables learning content reusability and durability in the lifecycle of the Learning Management System (LMS).

Today computer devices have become smaller and powerful. As a result, many people enjoy ubiquitous learning using mobile devices such as Pocket PCs [8]. Pocket PC has good features, being small in size,

easy to use and used for sharing information with PC; also working as an electronic organizer or day player, and a good player in Elearning.

One of the aspects to increase the performance of a network environment, uses the data packing agent. This agent will pack the data before sending it to Communication Agent. For security reason, the data packing agent needs to add security protection to pack Elearners learning records and some of his privacy information [8].

Different types of files are generated in SCORM learning objects (HTML, JPG, XSD, etc...); transferring this file by using the original size will have a negative effect on the performance of the network, cost, and time .

In this thesis a comparative study between different compressions tools (WinZip, WinRar, Power Archiver, and WinAce) is used in data packing agent to select the best one for packing the SCORM learning objects.

## **1.2. Problem Statement**

The purpose of this thesis is to select the best tool for packing the SCORM learning objects by comparing different criteria (support for other archiving format, support for solid archives, support for volumes....etc.) for each tools and selecting the best; which has a strong effect on minimizing the time, space limitations and cost.

## **1.3. Aims of Thesis**

The main aim of this thesis is reducing the network traffic by selecting the best tools for packing the SCORM learning objects. Different tools will be used to pack the SCORM learning objects; however, a comparison between the results will lead to selecting the more suitable tool, which will eventually result in reducing the network traffic, and increasing the performance of packing data in the Data Packing Agent.

## **1.4. Suggested Solution**

The suggested solution is to compare between the tools used in packing the SCORM learning object; to evaluate the compression ratio of each tool, select the more suitable one, and then compare the result with the deflate algorithms results

## **1.5. Research Methodology**

This research will use qualitative researching methods. This type of methods will act by using analysis tools, comparing the results and then selecting the more suitable tool.

## **1.6. Contributions**

This thesis is meant to reduce the network traffic, and increase the performance of the data packing by selecting the more efficient tool, which will positively affect the cost and speed of data transfer

## 1.7. Thesis Organization

**The thesis organization for the remainder is as follows:**

- Chapter One: this chapter contains a problem statement, aims of this thesis and research methodology; and ends by the contributions of this work.
- Chapter Two: presents the literature review of the related studies and works for this research such as Sharable Content Object Reference Model (SCORM) , Pocket SCORM.
- Chapter Three: introduces a thorough background of SCORM, focuses on high level requirements, SCORM Compliant and content organization, and also SCORM content model, run time environment; application programming interface; a personal digital assistance (PDA), types of PDA and advantage of PDA.
- Chapter Four: introduces a suggested model (in details) from Professor Timothy K. Shih; about converting the PC SCORM to PDA SCORM,

- thesis software and compression algorithms; and ends by a comparative study between different compression tools.
- Chapter Five: thoroughly introduces deflate algorithms; implements the deflate algorithms, and compares between the algorithms and tools results.
- Chapter Six: discusses the final conclusion of this thesis and provides guidelines for possible areas of the suggested future works.



## **Chapter Two Previous Related work**

### **2.1. Introduction**

A few studies and works available are represented in the core of this research, because the PDA SCORM is a new research and has a few white papers or related technical works that benefits can be derived from. But for the compression algorithms and tools a lot of research, white papers and algorithms are used to pack different type of files. So, this research depends on the other several studies papers and related trusted works which are published and which have a strong link with this research fields such as PDA SCORM and Packing data.

### **2.2. Related SCORM Studies and Works**

Oliver Bohl [16] lists different benefits from applying the SCORM. The main advantages are in the field of portability of learning content, the standardized communication interface between the LMS and WBTs which support the reusability of learning objects. There are a critical problems concerning the market value of SCOs, the process of producing WBTs on the basis of different SCO providers, the maintenance of SCOs and WBTs (consisting of several SCOs), and the

quality of WBTs based on SCOs of different providers. In this thesis using the SCORM standard to improve the PDA training in the field of portability.

Polyxeni Arapi [17] states that the proliferation of interoperability Elearning specifications raises the need of extending existing Elearning platforms so that they can be used efficiently in a distributed environment where material producers, service providers and users exchange information using standard models.

SCORM 1.2 does not contain yet a schema for the description of assessments. IMS Global Learning Consortium has developed the Question and Test Interoperability specification, which defines an XML format for the coding of questions and exams and it is very possible in the future to be a part of SCORM. In this thesis a schema for the description of assessments and protect all the privacy of trainers.

Shueh-Cheng Hu [24] says that the SCORM is a promising standard for delivering Web-based Elearning contents. The UML, a standardized and extensible notational language, is able to be applied in modeling the SCORM-conformant

contents, from different views and during different phases. A UML profile for the SCORM-conformant contents is worthy to develop.

Khan [10] believes that designing open, flexible, and distributed Elearning systems for globally diverse learners are challenging. An understanding of the capabilities of Elearning components and features can facilitate the design of meaningful Elearning environments. A well-designed Elearning program, therefore, has the ability to provide learner-centered, engaging, interactive, affordable, efficient, easily accessible, flexible, meaningful, distributed, and facilitated learning.

Khan suggested continuing examining the factors bearing on the success and failure of various Elearning features used in online courses.

### **2.3. Related Mobile Elearning Studies and Works**

Hui-huang [8] states that people benefit from the flexibility of the distance learning since it has broken the time and space limitation, to extend this flexibility and to make Elearners able to learn at any location, a suggested model for the Pocket SCORM architecture was shown in this research.

In the Proposed architecture, two types of connection are used to connect Pocket PC with LMS Server. One type is a Pocket PC which is directly connected to the server through wired or wireless network to the internet while the other is a Pocket PC which connects to server via PC to the internet while Pocket PC is synchronizing with the PC. In this thesis using the same architecture for implement the PDA SCORM architecture and increasing the performance of data transfer by selecting the best tools in data compression

Robert Yu-Liang Ting [19] says that due to the thrive of mobile network and portable device; distance learning is evolved from desktop computer to mobile device. There are different challenges of mobile learning such as the convergence of wireless infrastructure with handheld devices, the limited text display in supports the learning and the instant communication in mobile network.

Antonella Grasso [1] declares that it is important for the research to focus on the design and development of a solution for advancing the new distance learning frontiers, and on the design of mobile devices compatible with mobile learning technology goals using tools like macromedia Flash to make the Developer's program easier.

When proceeding from design to application, the need to reduce the operations necessary to carry out a given task, to facilitate interaction by each user category with the application in the various possible scenarios emerges.

Kiyoshi Nakabayashi [11] states that a learner adaptive self-learning environment in which both mobile phones and personal computers can be used as client terminals has been developed. A content browser has been developed to implement a common content format on mobile phone from different carriers.

Several issues must be resolved such as The usability of the content browser that must be improved, to engage in multiple offline learning activities and automatically generate client-side content.

#### **2.4. Related Compression and packing data Studies and Works**

Przemysław Skibiński [18] The primary objective of this research was to design an efficient way of compressing HTML documents, which will reduce Internet traffic or reduce storage requirements of HTML data. This research present the Lossless HTML Transform (LHT) aiming to improve lossless HTML compression in combination with existing general

purpose compressors. The main components of algorithm are: a static dictionary or a semi-static dictionary of frequent alphanumeric phrases, and binary encoding of popular patterns, like numbers, dates or IP addresses.

Tomasz Bartczak [25] states that over the last seven years the number of global Internet users have doubled. At the same time, network information resources have grown to such a degree that a problem has appeared as how to manage them. We can also observe an increased activity of Internet users, which is a result of huge information demand. The greatest Internet search engines compete in developing solutions to improve filtering and selecting the most sought for Internet resources.

In this research the performed tests have proven that the developed "HTTP Optimizer" system improves transmission band usage. The optimization level varies depending on the structure of the tested website: from 10 % in the case of a website containing large graphics files in the JPEG format and using only the packaging process, to 85% in the case of a website with very small graphics files.

Images in the PNG format (compressed packaging). The last test demonstrates that the amount of incoming packets can be reduced by 50% on average for typical websites.

## Chapter Three Applied Technology

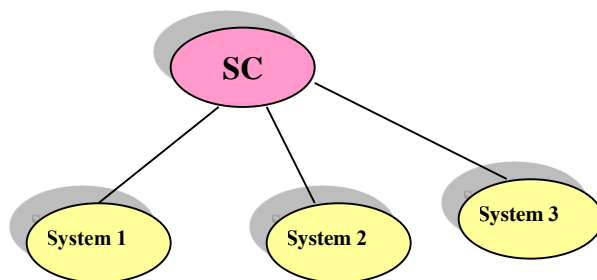
### 3.1. Introduction to Sharable Content Object Reference Model

Sharable content object reference model (SCORM) is a reference model, which shows types of services needed to solve a particular problem; how such services can be joined together, how the relevant standards can be applied and the way they can be used. SCORM helps defining the technical foundations of a Web-based learning environment [23].

### 3.2. High Level Requirements

ADL has identified six essential high-level attributes for all distributed learning environments [8].

**Interoperability:** the ability to take instructional component from one system, and use it in another



learning system as shown in Figure (3.1).

### Figure 3.1 - Example of Interoperability

**Accessibility:** the ability to locate and access instructional components from multiple locations, and deliver them to other locations as shown in Figure (3.2).

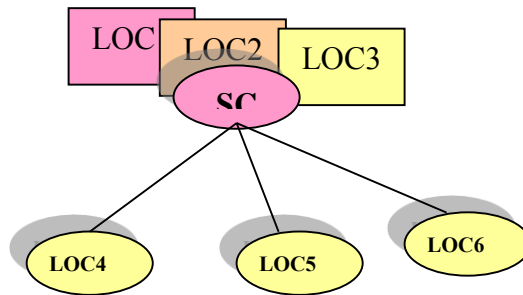
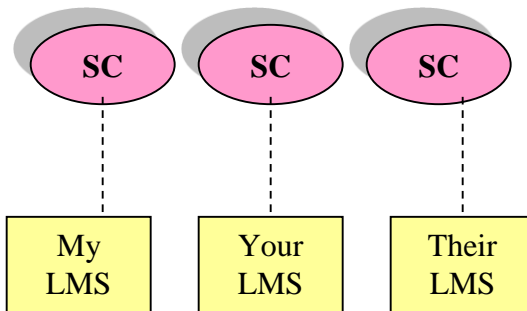


Figure 3.2 - Example of accessibility

**Reusability:** the ability to use instructional components in multiple applications, courses and contexts as shown in Figure (3.3).

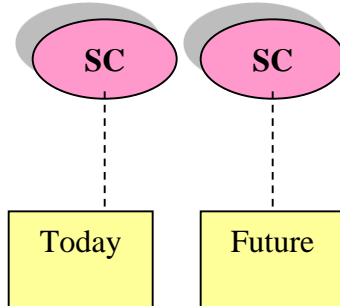


in Figure (3.3).

Figure 3.3 - Example of reusability



**Durability**: the ability to resist the technology changes over time without cost, reconfiguration or development

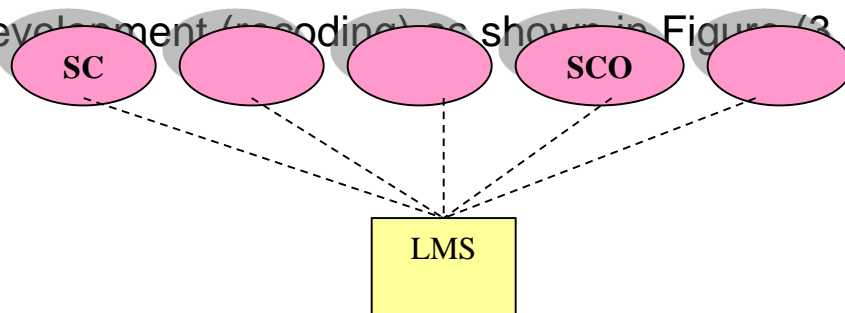


(recoding) as shown in Figure (3.4).

**Figure 3.4 - Example of durability**

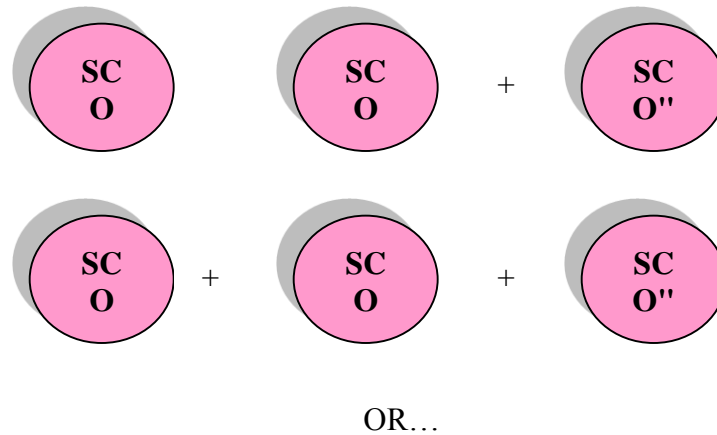
**Maintainability**: the ability to withstand content evolution and changes, without any cost, reconfiguration or

development (recoding) as shown in Figure (3.5).



**Figure 3.5 - Example of maintainability**

**Adaptability**: the ability to change/modify and satisfy different user needs as shown in Figure (3.6).



**Figure 3.6 - Example of adaptability**

### 3.3. SCORM Compliant

A SCORM compliant learning system consists of the following components:

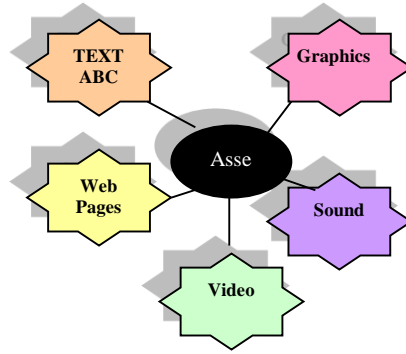
#### 3.3.1. A Learning Management System (LMS)

Software that automates training event administration, through a set of services that:

- launches learning content
- keeps track of learner progress

- sequences learning content

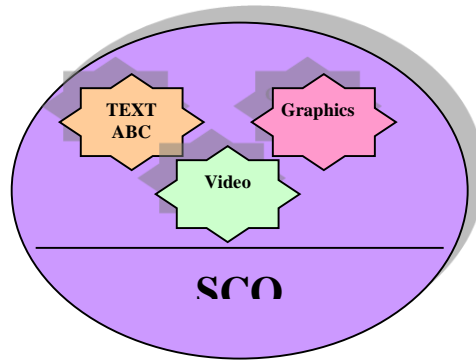
**Assets specific web media such as HTML files, images, or video as shown in Figure (3.7).**



**Figure 3.7 - Sample assets**

### **3.3.2. Shareable Content Object (SCO)**

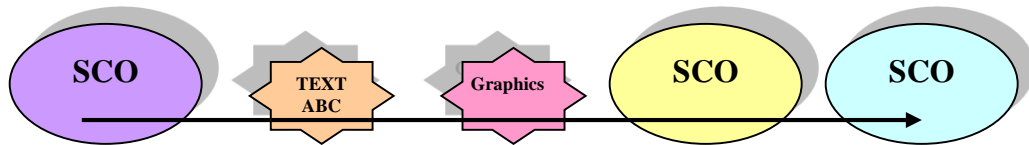
It is the collection of the assets (text, graphics, video, etc.) that form the learning activity as shown in Figure (3.8).



**Figure 3.8 - Sample of SCO**

### 3.3.3. Content Aggregation

An overall library of related content and sharable content objects as shown in Figure (3.9)



**Figure 3.9 - Sample of content aggregation**

### 3.4. SCORM Content Aggregation Model (CAM)

It represents learning taxonomy neutral means for designers and implementers of instructions to aggregate learning resources for the purpose of delivering a desired learning experience.

The SCORM Content Aggregation Model is made up of the following [20]:

#### 3.4.1. Content Model:

Nomenclature defines the content components of a learning experience; the SCORM Content Model is made up of the following.

- **Assets**

It is the basic structure of the learning resource as shown in Figure (3.10).

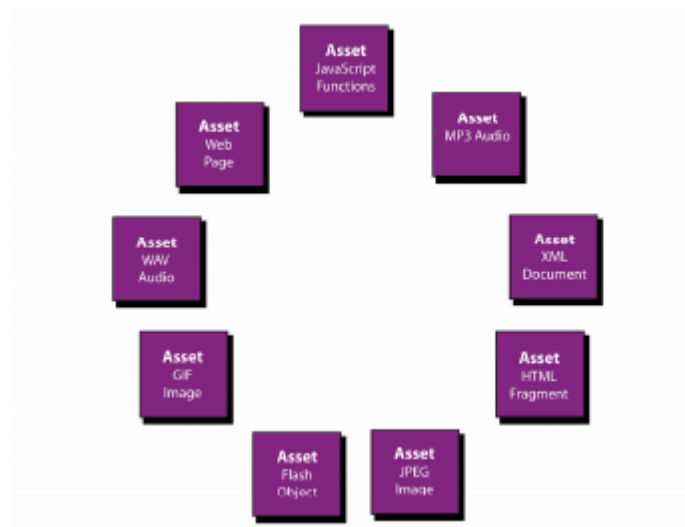


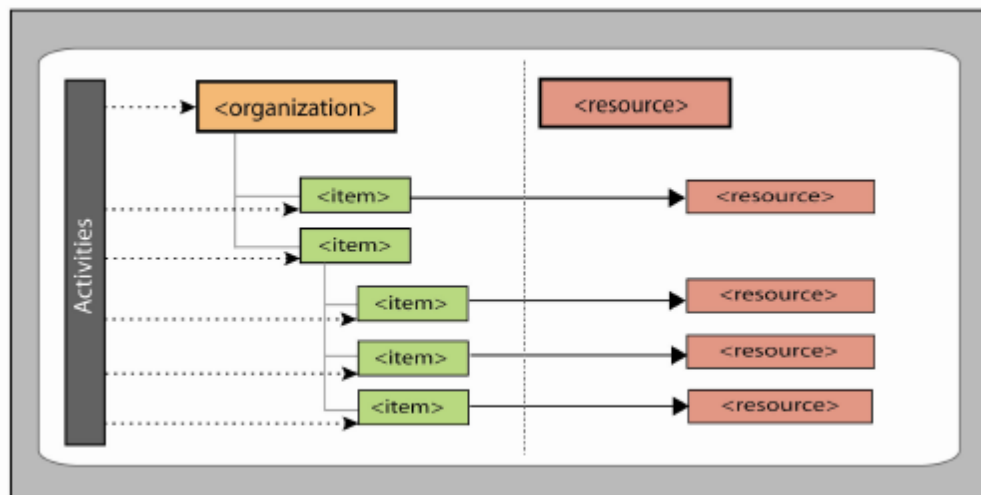
Figure 3.10– Example of Assets [21]

- **Sharable Content Object (SCO)**

A SCO is a group of one or more Assets; The difference between the assets and SCO is the communication with the LMS; using the Institute for Electrical and Electronics Engineers (IEEE) ECMAScript Application Programming Interface for Content, to Runtime Services Communication standard [9].

- **Activities**

It is the unit of instructions which is given to the learner, while progressing through instruction. A learning activity may provide learning resource (SCO or Assets) to the learner, or it may be composed of more than sub activity. There is no set limit to the number of levels of nesting Activities [20] as shown in Figure (3.11).



## Figure 3.11 – Conceptual Representation of Activities [21]

### ▪ Content Organization

A Content Organization is a representation or a map that defines the intended use of the content, through structured units of instruction (Activities). The map shows how Activities are interrelated as shown in Figure (3.12).

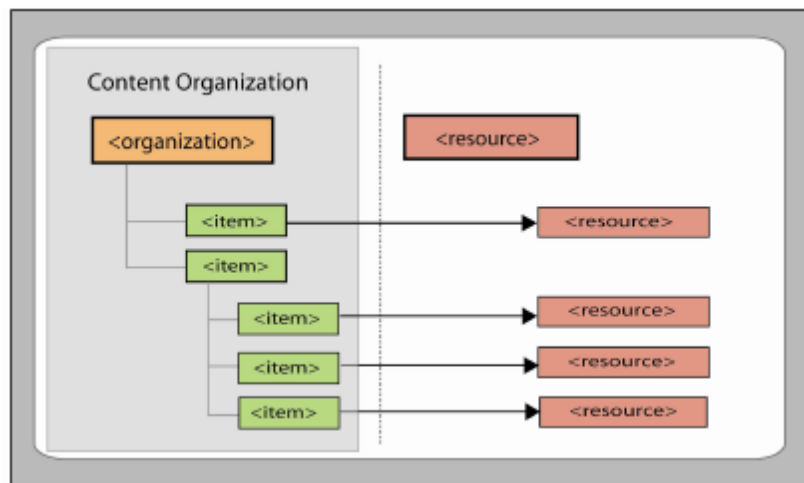
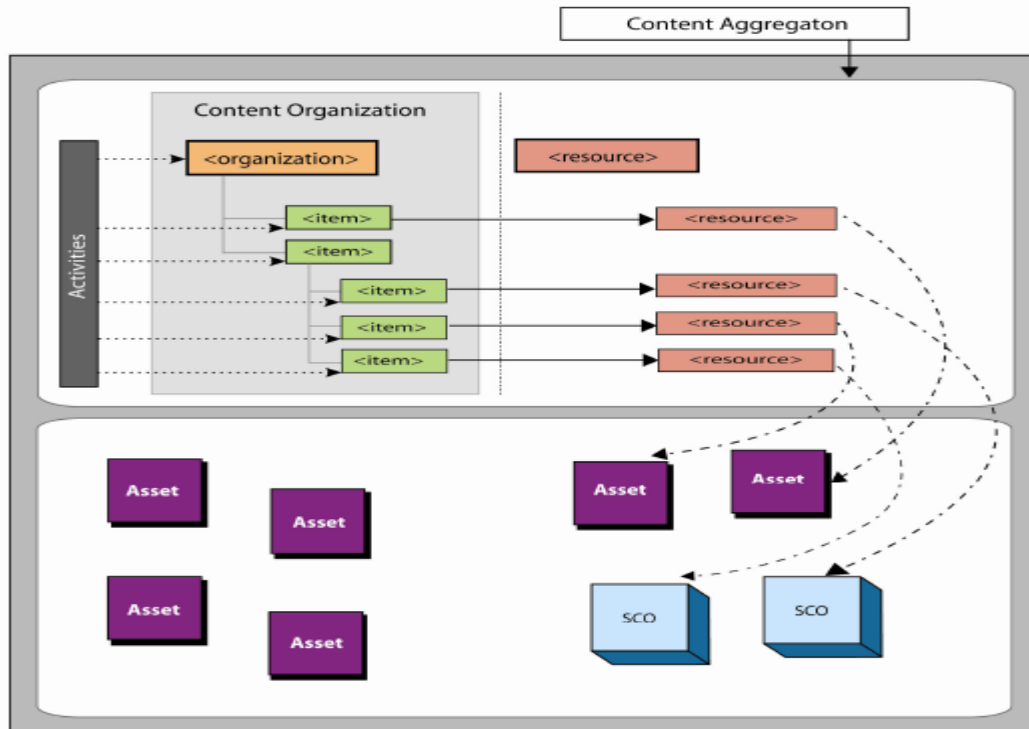


Figure 3.12– Conceptual Illustration of a Content Organization [21]

### ▪ Content Aggregation

Content aggregation can be used to describe the action or process of composing a set of functionally related content objects. A Content aggregation is also used to describe the entity created as a part of this process or action as shown in Figure (3.13).



**Figure 3.13–** Conceptual Illustration of a Content Aggregation [21]

### 3.4.2. Content Packaging:

SCORM Content Packaging is used to provide a standardized way to exchange learning content between different systems or tools [20]. A Content Package contains two major components:

#### Manifest File:

The manifest is an XML document that contains a structured inventory of the content of a package; It is composed of four major sections [20]:

- **Metadata:** Data describing the content package as a whole.



- **Organizations:** Containing the content structure or organization of the learning resources, making up a stand-alone unit or units of instruction.
- **Resources:** Defining the learning resources bundled in the content package.
- **(sub)Manifest(s):** Describing any logically nested units of instruction as shown in Figure (3.14) (which can be treated as stand-alone units).



**Figure 3.14–** Components of a Manifest [21]

## **Package**

The package represents the learning unit. The learning unit may be part of a course that has instructional relevance outside of a course organization, and can be delivered independently; as a portion of a course. A package must be able to stand alone; more specifically, it must contain all the information needed to use the packaged contents for learning when it has been unpacked.

## **Package Interchange File (PIF)**

Package interchange file (PIF) is a binding of the content package components; in the form of a compressed archive file. The PIF contains the `imsmanifest.xml`, and all control files and the resources referenced in the content package. SCORM recommends that content packages be created as PIFs.

### **3.5. SCORM Run-Time Environment (RTE)**

The SCORM RTE defines a model that picks up at the point when a specific content object has been identified for launch [21].

The API is the communication mechanism for informing the LMS of the conceptual communication state, between a content object and an LMS, and is used for data retrieving and storage.

A Data Model is a standard set of data model elements that is used to identify the information being tracked for a SCO.

### **3.5.1. Run-Time Environment Temporal Model**

A learner becomes engaged with the content object once an activity with the associated content object has been identified for delivery; and the content object has been launched in the learner's browser-environment. Several key aspects need to be defined; to aid in the tracking of the learner during the learning experience [21].

**Learner Attempt** – Is a tracked effort by a learner to satisfy the requirements of learning activity that uses a content object. An attempt may span one or more learner sessions, and may be suspended between learner sessions.

**Learner Session** – An uninterrupted period of time, during which a learner is accessing a content object.

**Communication Session** – An active connection between a content object (SCO) and an application programming interface.

**Login Session** – A period of time during which a learner begins a session with a system (logged on), until the time the learner terminates the session with the system (logged out).

These four terms are relevant when it comes to managing the RTE for a SCO.

### **3.1.1.1 Managing Learner Attempts and Learner Sessions**

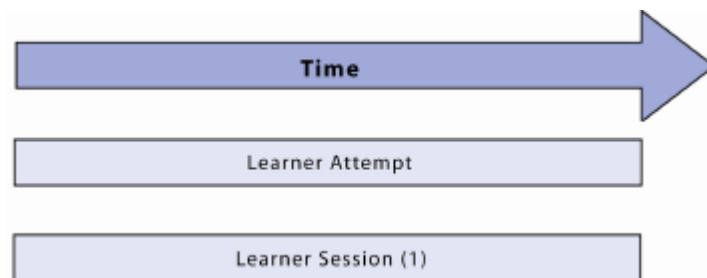
A learner attempt is associated with an important LMS requirement; defining the management of the set of run-time data model elements that the SCO may communicate to the LMS.

When a new learner's attempt begins for a SCO, the LMS is required to create and initialize a new set of run-time data for the SCO; to be able to access and use.

SCORM does not specify exactly when the new set of run-time data is created; but all data model accesses must show up (to the SCO) as if they are being performed on a new set of run-time data.

The LMS may elect to store the data for historical purposes, or for other purposes, such as reporting, auditing, diagnosis or statistical; when the previous attempt's data is outside the scope of SCORM. The LMS may elect to discard any run-time data collected during the previous attempt.

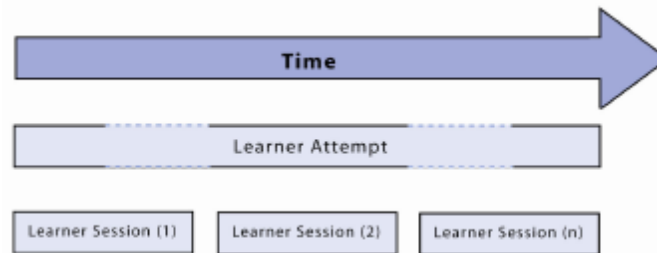
The following Figure (3.15) illustrates several different relationships, between a learner's attempt and session (s) [21].



**Figure 3.14–** Single Learner Attempt with one Learner Session [20]

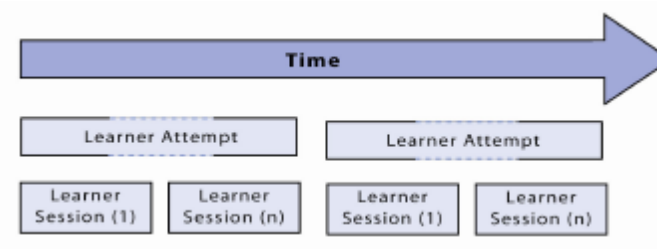
The following Figure (3.16) illustrates a single learner's attempt, proliferated among several learner's sessions. These

sessions have been suspended and the learner's attempt has been subsequently resumed; until a learning session ends normally [21].



**Figure 3.16- learner attempt spread over several learner session [20]**

The following Figure (3.17) illustrates successive learner's attempts. Within each of these learner's attempts, any number of learner's sessions may take place.



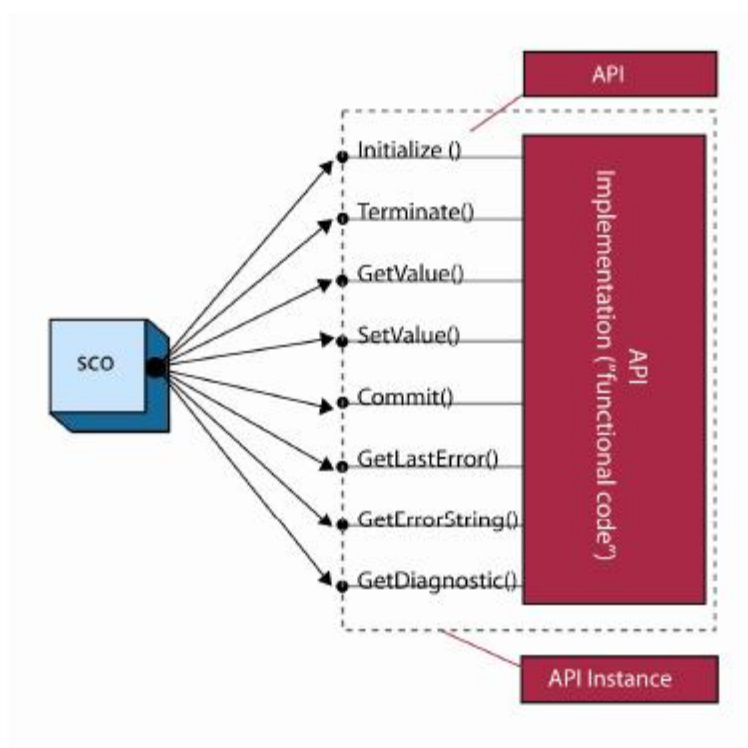
**Figure 3.17- successive learner attempts, spread over several learner session [20]**

## 3.6. Application Programming Interface

### 3.6.1. API Overview

API enables the communication of data between content and run time service (RTS); the use of a common API is to fulfill and reuse many of SCORM high-level requirements for interoperability. It provides a standardized way for SCOs, to communicate with LMSs; and also shields the particular communication implementation from the SCO developer.

There are several terms that are used throughout SCORM: API, API Implementation and API Instance as shown in Figure (3.18)



**Figure 3.18** -Illustration of API, API Instance and API Implementation [21]

- An API Implementation is a segment of functional software that implements and exposes the functions of the API.
- An API Instance is an individual execution context, and a state of an API Implementation.

A key aspect of the API is to provide a communication mechanism that allows the SCO to communicate with the LMS.

The methods exposed by the API Implementation are divided into three categories:

- **Session Methods:** Session methods are used to mark the beginning and end of a communication session, between a SCO and an LMS through the API Instance.
- **Data-transfer Methods:** Data-transfer methods are used to exchange data model values, between a SCO and an LMS through the API Instance.



- **Support Methods:** Support methods are used for auxiliary communications (e.g., error handling), between a SCO and an LMS through the API Instance.

### 3.6.2. API Methods and Syntax

There are some general requirements dealing with the API:

- All function names are case sensitive, and shall be expressed exactly as shown.
- All function parameters or arguments are case sensitive.
- Each call to an API function -other than the Support Methods- sets the error code.
- All parameters and return values are sanctioned as character strings, and shall be compatible with the data types and formats; described by the data models using the API for communication.

### 3.7. The Sequencing Definition Model

The SCORM Sequencing Definition Model defines a set of elements that may be used by content developers; to define intended

sequencing behavior. SCORM does not place any requirements, on when or how SCORM Sequencing definition model elements are applied to learning activities.

### **3.8. Sequencing Control Modes**

The Sequencing Control Modes allow the content developer to affect how navigation requests are applied to a cluster, and how the cluster's activities are considered while processing sequencing requests [22]. The control modes are used in the following ways:

- During processing of a navigation request; to determine whether or not the request will translate into a valid request.
- Sequencing request during various sequencing request minor processes; to affect how activities will be considered for delivery.
- During various sequencing processes; to affect how tracking status information is managed

### **3.9. Personal Digital Assistant (PDA)**

### **3.9.1. Introduction**

The main purpose of the Personal Digital Assistant (PDA) is to act as an electronic organizer or day planner that is portable, easy to use and capable of sharing information with a PC [4].

PDA's can do a lot of things such as contacts, appointments and connecting with the internet; and also acting as a global positioning system (GPS) device, and running multimedia software .

### **3.9.2 Types of PDA's**

#### **3.9.2.1. Traditional PDA's**

Traditional PDA's are descendents of the original Palm Pilot and Microsoft Handheld PC devices. Palm devices run the Palm OS (operating system), while Microsoft Pocket PC's run Windows Mobile. The differences between the two systems are fewer than they were in the past [4].

### **3.9.2.2.. Palm PDAs**

Most Palm devices are made by palmOne, which offers the Zire and Tungsten product lines. The company was incorporated in 2003; when Palm Computing acquired Handspring Inc.

### **.3.9.2.3. Pocket PC's**

Pocket PC is the generic name of Windows Mobile PDAs. Their standard features include:

Pocket versions of Microsoft applications, such as Microsoft Word, Excel, and Outlook

Synchronization with Microsoft Outlook on a Windows PC (synchronization with e-mail systems other than Outlook or with Macintosh computers requires additional software)

A virtual writing area, which maximizes the display size

Windows Media Player for multimedia content

### **3.9.2.4. Smartphones**

A smartphone is either a cell phone with PDA capabilities or a traditional PDA with added cell phone capabilities; depending on the form factor (style) and manufacturer. Such devices have many characteristics; like:

- A cellular service provider to handle phone service (As with cell phones, you typically purchase a cellular plan and smart phone from the service provider).
- Internet access through cellular data networks.
- Various combinations of cell phone and PDA features (depending on the device).
- A number of different operating systems, including Windows Mobile, Pocket PC Phone Edition and the Palm OS; and also the Blackberry OS for Blackberry smart phones.

### **3.9.3. Advantages of PDA**

Different advantages of a PDA:

#### **. Handle Standard PIM Functions**

All PDAs come with some kind of personal information management (PIM) software that typically handles the following tasks:

- Storing contact information (names, addresses, phone numbers and e-mail addresses)
- Taking notes

- Tracking appointments (date book, calendar)
- Reminding you of appointments (clock, alarm functions)
- Performing different calculations

### **3.9.1.1. Handle Run Application Software**

PDA's can run specialized software applications:

- Windows Mobile devices come with Pocket versions of Word, Excel, Internet Explorer and Outlook (including e-mail and PIM functions); along with Windows Media Player and voice memo recording.
- Most Palm OS devices include applications such as DataViz Documents; to go compatible with (Microsoft Word, Excel, and PowerPoint), palmOne Media (for photos and video), VersaMail e-mail software and web-browsing software.
- All types of devices can run other kinds of software including games, multimedia, diet and exercise; as well as travel, medicine, time and billing and reference.

### 3.9.1.2. Synchronize With PC's

- Because PDAs are designed to complement the PC, they need to work with the same information in both places. If making an appointment on the desktop computer, it needs to transfer it to the PDA.
- Synchronization software on the PDA works with companion software that is installed on the PC. Microsoft Pocket PC devices use ActiveSync, while Palm OS devices use HotSync synchronization software. This type of software also needs an application like Microsoft Outlook or the Palm Desktop that holds PIM information on the PC side.
- The synchronization always has a backup copy of the data, which can be a lifesaver if the PDA is broken, stolen or completely out of power.

### 3.9.1.3. Common Functions

Most PDAs incorporate various types of functions [4], such as:

- Short-range wireless connectivity using Infrared (IR) or Bluetooth technology. IR is found in most PDAs, and requires a clear line of sight. It's

commonly used to sync. with a notebook computer that has an IR port. Bluetooth (which is a radio frequency technology that doesn't require a clear line of sight) wirelessly connects to other Bluetooth-enabled devices, such as a headset or a printer.

- Internet and corporate network connectivity, and wireless access points.
- Support of Wireless WAN (Wide Area Networks).
- A memory card slot that accepts flash media, such as Compact Flash, Multimedia Card and Secure Digital cards (Media cards act as additional storage for files and applications).
- Audio support for MP3 files and a microphone.

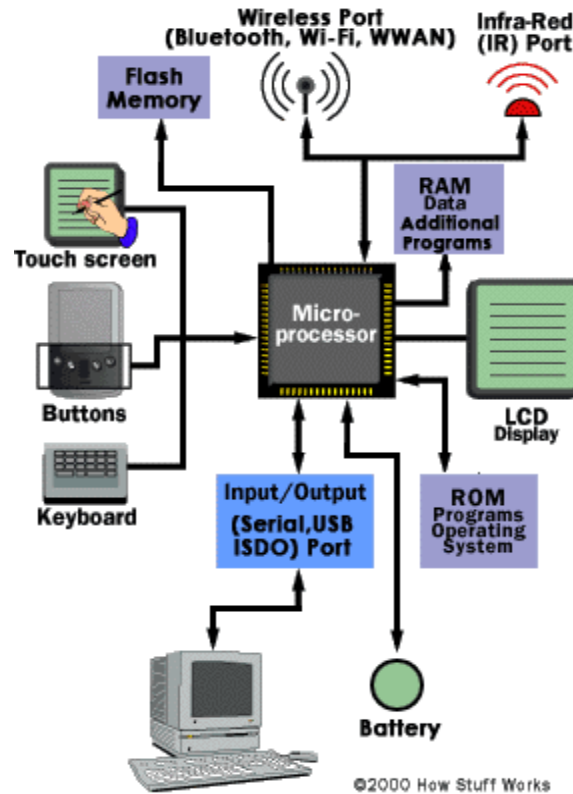
### **3.9.4. PDA Components**

#### **3.9.4.1 Microprocessors and Memory**

The microprocessor is the brain of PDAs as it coordinates all of the functions according to previously programmed instructions. PDAs use smaller and cheaper microprocessors. Although these microprocessors tend to be slower than their PC counterparts, they are still adequate for the tasks that



PDA's usually perform as in Figure (3.19). The benefits of small size and price outweigh the cost of slow speeds [4].



**Figure 3.19-** Component of PDA

Basic programs (address book, calendar, memo pad and operating system) are stored in a read-only memory (ROM) chip, which remains intact even when the machine shuts down. The data and any programs added later are stored in the device's random-access memory (RAM). Information in

RAM is only available when the device is on. Due to their design, PDAs keep data in RAM safe, because they continue to draw a small amount of power from the batteries; even when you turn the device off.

Less powerful PDAs have lower amounts of RAM; however, many application programs take up significant memory space, so most models have more memory. Pocket PC devices generally require more resources, and have even more RAM. To provide additional memory, many PDAs accept removable flash media add-on cards. These are handy for storing large files or multimedia content; such as digital photos.

Some newer PDAs, such as the Palm Tungsten E2, use flash memory instead of RAM. Flash memory is non-volatile, which means it preserves stored data and applications.

### **3.9.4.2 PDA Display and Input**

#### **LCD Display**

PDA's use an LCD (liquid-crystal display) screen. PDA's use their screens for output and input. The LCD screens of PDA's are smaller than laptop screens, but vary in size. Almost all PDA's now offer colored displays.

PDA displays have the following features:

- Transflective TFT (thin-film transistor) LCD for indoor and outdoor use
- Different pixel resolutions with higher resolutions for better quality
- Color screen
- Backlighting for reading in low light

### **Input Methods**

PDA's vary in how you input data and commands. Some devices use a stylus and touch screen, exclusively in combination with a handwriting recognition program. Using a plastic stylus, you draw characters on the device's display or dedicated writing area. Software inside the PDA converts the characters to letters and numbers. On Palm devices, the software that recognizes these letters is called Graffiti. Graffiti requires that each letter be recorded in a certain way,

and you must use a specialized alphabet. To help Graffiti make more accurate guesses, you must draw letters in one part of the screen and numbers in another part.

Pocket PC offer three handwriting-recognition applications: Transcriber, Letter Recognizer and Block Recognizer. Letter Recognizer and Block Recognizer are similar to Graffiti, and require specialized alphabets. By contrast, Transcriber recognizes your "regular" handwriting, as long as you write legibly. It is similar to the handwriting recognition capabilities found on Tablet PC's. If you can't get the hang of PDA handwriting, you can use a miniature on screen keyboard. It looks just like a regular keyboard, except you tap on the letters with the stylus. In addition, many devices now include a small (and usually cramped) QWERTY keyboard. Some of these require you to use your thumbs to type. You can also use a full-size keyboard, by connecting it to the PDA via Bluetooth or a USB port. Each model also has a few buttons and navigation dials, to bring up applications and scroll through files.

## Chapter Four

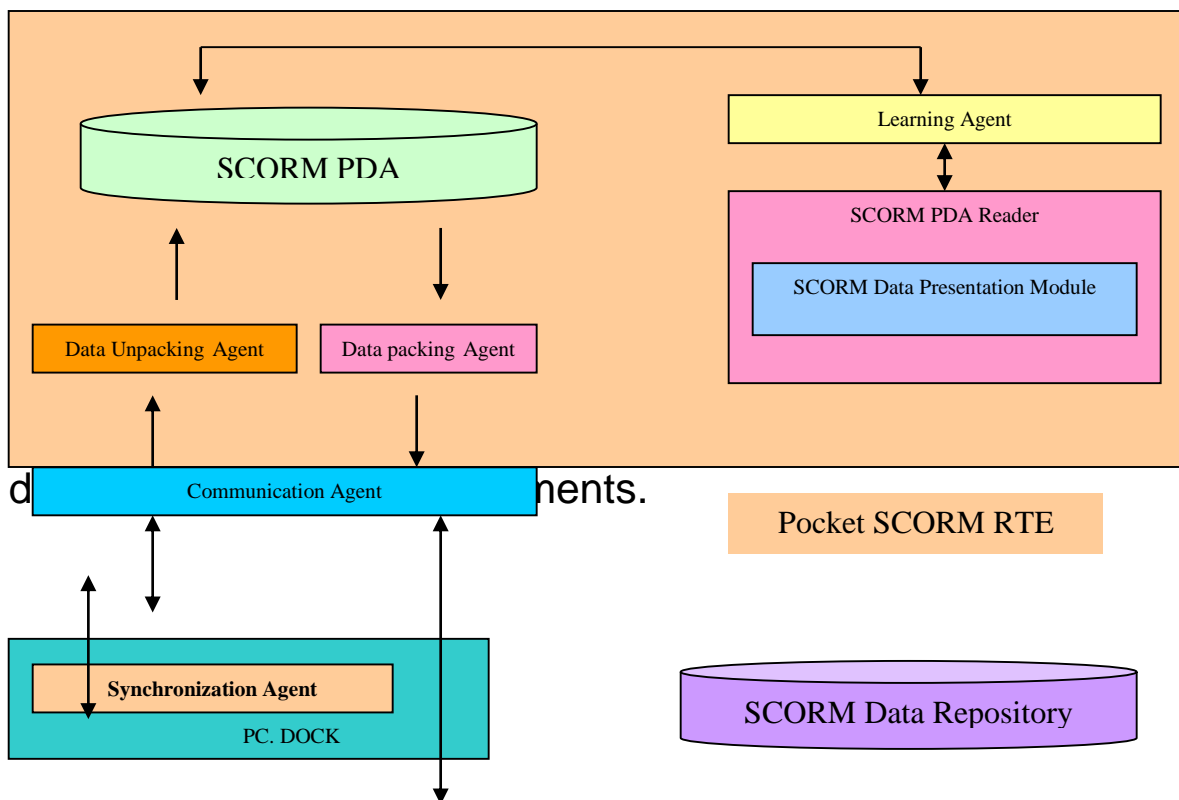
### SCORM-Compression Tools: A Comparative Study

#### 4.1. Introduction

A suggested model from Professor Timothy K. Shih, Pocket SCORM, in which he gave a comprehensive view of the pocket SCORM [8].

#### 4.2. Pocket SCORM

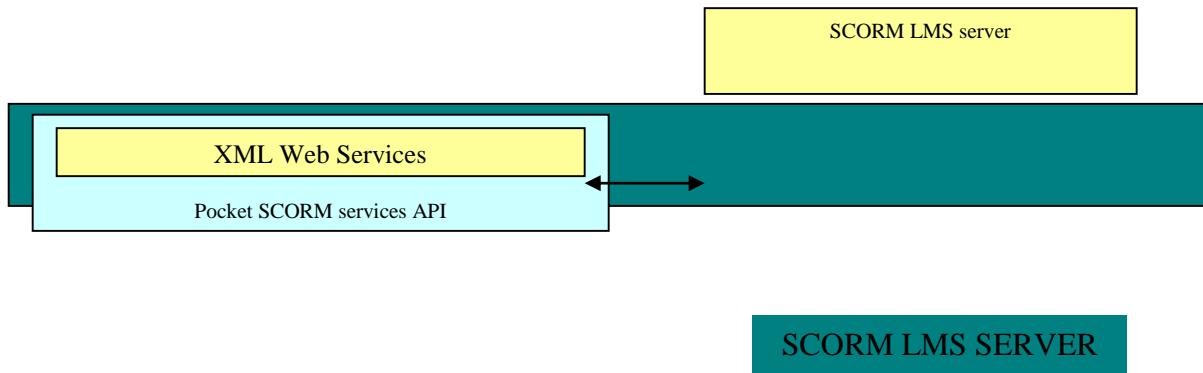
Pocket SCORM run time is consists of six components as shown in Figure (4.1)



Components.

Pocket SCORM RTE

SCORM Data Repository



**Figure 4.1-** Relationship of Components within Pocket SCORM Architecture [8]

### **Communication agent**

The communication agent uses a kind of ACL (Agent Communication Language) to connect PDAs with LMS server. All learning objects are stored in the server. The communication agent will be used to transfer data from/to the server and PDA.

The same process is used when we need to return back the data to learning Management System (LMS); especially for the student tracking and other information.

### **Data Packing agent :**

The data packing agent is used to reduce network traffic. This agent will pack the data before sending it to Communication Agent. For security reason, a security protection function must be added to this agent because normally it will pack Elearners learning records and some of his privacy information.

## **Data Unpacking Agent**

The courseware will be packed as a Package Interchange File (PIF) before it is downloaded by Communication Agent from LMS Server. Therefore, we need Data Unpacking Agent to unpack the PIF file after it has been downloaded from the LMS server and then restore the original courseware from the PIF.

## **Learning agent**

After downloading the learning objects, the Elearner starts to view the information; and the learning agent will track the learners learning record, in order to keep track of all the information related to the elearner, and will also keep history of this information; however, there must be synchronization between the PDA and the database.

## **SCORM PDA reader**

Because of the hardware restrictions in the PDA, the normal web based content is incompatible with the PDA size; for that reason, and to achieve compatibility with the PDA size, the SCORM PDA Reader is needed, to make a reflow that adjust the content, in order to make it fit in the width of the display width of other readers.

## **SCORM PDA Database**

After unpacking the data, it will be stored in a temporary data store, which is called SCORM PDA Database. When learners study the courseware, the SCORM PDA Reader will load the course content from the SCORM PDA Database. So, learners don't have to be on-line for learning courseware, because it has been previously stored in their pocket PC. During the learning period, the Learning Agent will track and store the learning records into this data store. After the learning records have been transferred to the LMS Server, those records will also be removed from this database as unused course content; to save the precious memory space of a Pocket PC.

PC Dock

PC Dock is a layer between Pocket SCORM RTE, and SCORM LMS Server. If the Pocket PC is incapable of being online, it will require a PC Dock; to be able to connect to the LMS server. There is a Synchronization Agent inside the PC Dock. This Synchronization Agent will perform the data transmission process, between Communication Agent on the Pocket SCORM RTE, and XML Web Service on the SCORM LMS Server. The transmission protocol will focus on SOAP,



while the PC Dock will aim at providing the internet communication ability for those Pocket PC's without network ability.

### **SCORM LMS Server**

There are two major components in the SCORM LMS Server. One is the SCORM Data Repository, and the other is Pocket SCORM Service API. SCORM LMS Server provides distance education courseware, which follows the definition of SCORM Data Model.

The learner's information is also saved in the SCORM LMS Server. When a learner connects to the LMS Server, he or she needs to first logon the system before being able to access any course materials.

### **SCORM Data Repository**

The SCORM Data Repository stores all the course materials, which follow SCORM Data Object Model. These course materials can not only be provided to pocket device users, but also to desktop or laptop computer users; using browser based application to access the courseware through LMS Server. This data repository should also support any SCORM based API. Learner's learning records are

also stored in this data repository. These SCORM based learning records should also be able to interact, by using either Pocket SCORM Service API or any SCORM based API.

### **Pocket SCORM Service API**

Due to some limitations of Pocket PCs, such as computing power and memory storage, the Pocket SCORM Server APIs needs to be built; by adopting XML Web Service technology. XML Web Service takes SOAP as its transmission protocol. One of the advantages of using XML Web Service as our APIs is the accessibility. Since SOAP is loosely coupled protocol by using XML wrapped envelope to invoke APIs, this advantage makes XML Service APIs accessible by any platform, which follows SOAP protocol to acquire the service. Another idea regarding the storage limitation, is not allowing Pocket PC users to download course materials of a course all at once. Our APIs must provide some functions, to get some information from user's device, and detect the storage limitation; then inform the learner to download a certain courseware portion of a course.

### 4.3.Thesis software

This software is created by HunterStone Inc., which allows Elearning content authors and subject matter experts to create and deploy standards-based multi-media learning content; to be used with any SCORM conformant learning management system. The following can be processed by this software:

Create online training sessions with well known applications (Microsoft Office Application).

Create SCORM conformant content, and ensure that each object will pass the ADL requirements for conformance.

#### **Steps to create SCORM Content with THESIS Software for PowerPoint**

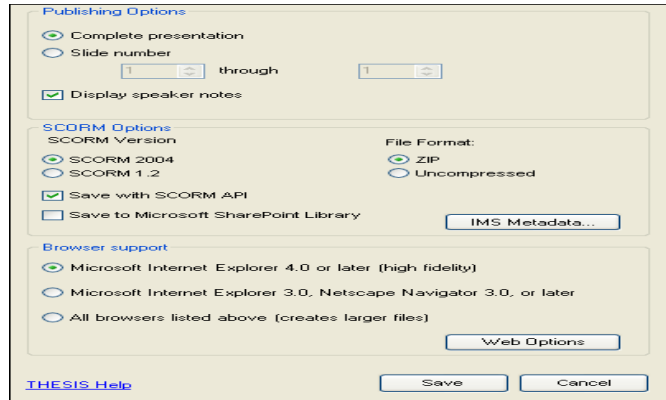
1. Launch PowerPoint
2. Create the learning content and save the file as usual
3. Save as SCORM as shown in Figure (4.2) and Figure



(4.3)

## Figure 4.2 - Save as SCORM

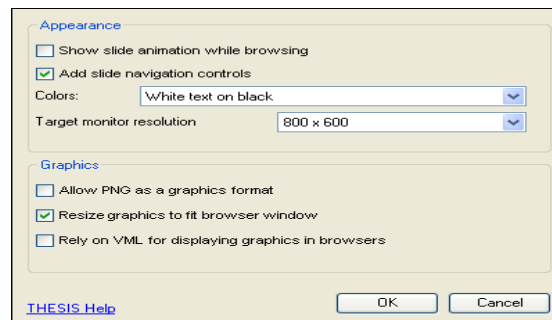
### 4. Save as SCORM



## Figure 4.3 - Specify the property of SCORM

### 5. Web Options

Set appearance options to change the appearance of the final publication. In the Graphics section, set options for how pictures will be formatted, and the target monitor screen size see Figure (4.4).



## Figure 4.4 - Set the appearance option

### 6. after generating the SCORM

#### **4.4. Packing data**

To reduce the Network Traffic and the size of SCORM [8], the data is packed by using packing data agent. Different algorithms and tools are used to Pack/Unpack the data; each one has its own features and properties [3]. This thesis is meant to compare between different packing/unpacking tools (WinZip, WinRar, WinAce, Power archiver), to select the more suitable one for the packing process.

##### **. Compression algorithms**

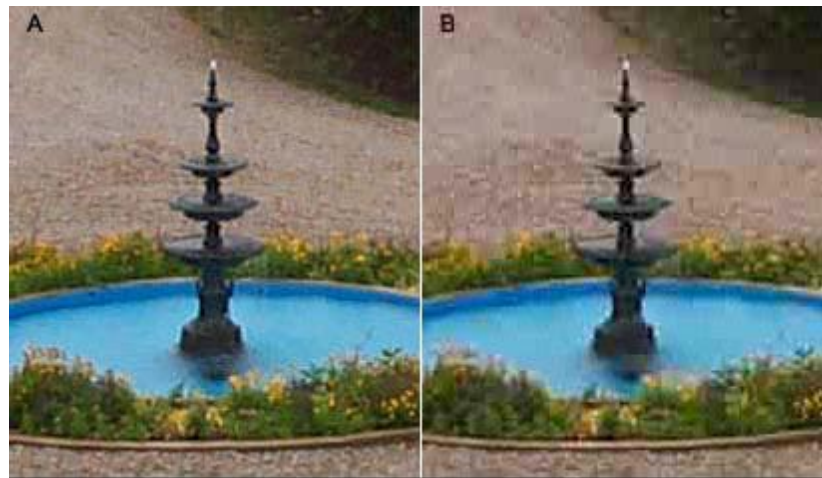
Compression algorithms can be divided into two groups: generic and specialized. Generic algorithms are able to compress any type of information, with good but not perfect results. Specialized algorithms are very good at compressing some types of information, like images or sound; but have poor results regarding other types of data.

##### **. Lossy and Lossless algorithms**

Data can be compressed with or without loss of information. Lossy compression is mostly used for drastically reducing size of images and sound files; because the human senses are not very sensitive

towards small changes in quality. The JPG format is able to reduce an image size by examining adjacent pixels, and making similar ones appear as the same; thus reducing the entropy and increasing the compression. Images A and B as shown in Figure (4.5) are saved using the highest and lowest JPG quality. Note that while image A is twice as large as B, the quality of picture B is not drastically different.

Lossless compression is used for all other types of data, where accuracy is mandatory.



**Figure 4.5 - Lossy compression example. A) High quality JPG image. B) Low quality JPG image.**

### **Specialized Algorithms**

Some well-known specialized algorithms are:

## **Run length encoding (RLE)**

Run length encoding algorithm [12] is mostly used with bitmap images (e.g., black and white images); where symbols (pixels) with the same value are often found in contiguous streams. The stream can be then replaced with <Count, Value> pair; thus decreasing the image size. This method must be cleverly implemented, to avoid data expansion; which can happen if the streams are short or the symbols alternate.

## **JPEG and MPEG algorithms**

The Joint Photographic Experts Group (JPEG) [12], and the Moving Pictures Expert Group (MPEG) committees proposed two Lossy algorithms; for image and moving picture compression respectively. These algorithms make two passes over the data. First pass converts the data into the frequency domain, using FFT-like algorithms. Once transformed, the data is smoothed by rounding off the peaks; resulting in a loss of information. In the second pass, the data is compressed using one of the lossless algorithms. Compression ratios can be very high, with acceptable quality degradation.

## **MP3 and WMA algorithms**

MP3 and Windows Media Audio (WMA) [15] are Lossy algorithms, used for audio signal compression. Compression ratios are high with a reduction in size, by a factor of 10 or more with low distortion of the audio signal. These algorithms use the fact that the human ear cannot hear some frequencies, if they are masked by other frequencies. They eliminate those hidden frequencies; hence reducing the size of the signal. The encoding procedure is highly CPU intensive, while decoding is not. This property is one of the reasons, why the MP3 format is widely accepted for storing audio files.

## **Generic Algorithms**

There are few generic algorithms currently in use, like (1) Dictionary, (2) Sliding Window, (3) Huffman, and (4) arithmetic coding algorithms. They are all implemented as lossless and adaptive.

## **Dictionary based Algorithms**

This family of algorithms encodes variable-length strings into fixed length codes, which are called tokens. The most popular algorithm in this group is the



Lempel-Ziv-12-Welch (LZW) algorithm, which is used in almost every commercial compression utility or library. It parses the input stream, and for each new phrase that encounters, it adds a <Token, Phrase> pair to the dictionary. The algorithm is fairly simple, but its implementation is usually complex; because of the dictionary maintenance tradeoffs. If the dictionary becomes full, one of three actions occurs: (1) the dictionary could be flushed, which affects compression, (2) the dictionary could be frozen, which affects compression if data changes over time, or (3) the dictionary could be expanded, which could lower the compression ratio; because the token size increases.

### **Statistical model based Algorithms**

This family of algorithms encodes symbols into bit-strings of variable length, using a statistical model. These algorithms are based on a Greedy Approach, where symbols with the high probabilities are given the shortest codes (bit-strings). The model has to accurately predict the probabilities of symbols, to increase the compression.

The Huffman coding algorithm achieves the minimum amount of redundancy possible, if the bit-strings are limited to an integer size. As a result, it is not an optimal method, just a good approximation. It is a very simple and fast algorithm, suitable for devices with slow CPUs. Memory consumption depends on the order of the model used.

The arithmetic coding algorithm does not produce a single code for each symbol; it produces a code for the whole message. Each symbol added to the message incrementally modifies the output code. This approach allows a symbol to be encoded with a fractional number of bits, instead of an integer number; thus exactly representing entropy of the symbol. Theoretically, arithmetic coding is optimal for a given model, but a real-world implementation has to make some tradeoffs related to floating point or integer arithmetic; thus decreasing the compression ratio. The arithmetic coding algorithm needs a powerful CPU for both the encoding and decoding process, which makes it less desirable on PDA devices than the Huffman coding algorithm.

### 4.1.1 Comparison between tools

Different data packing/unpacking tools are available in the market; offering various features, advantages and disadvantage. Also different criteria and questions can be utilized to select the more suitable tools for packing the SCORM learning objects.

### Operating System and Devices used

- Lab top LS50a with the following specification
  - Intel Pentium processor 1.8MHz
  - 512 MB RAM
  - 40 GB HDD
- Windows XP
- Type of files used
  - XML
  - XSD
  - JPG
  - HTML

### Compression Tools

The following tools are used in the comparison

- WinRar Tool
- WinZip Tool

- WinAce tool
- Power Archiver Tool

### Elearning object used

The SCORM learning object consisted of the following files as shown in Figure (4.6)

Name	Size	Type	Date Modified
common		File Folder	1/7/2008 10:10 AM
extend		File Folder	1/7/2008 10:10 AM
proposal_scorm		File Folder	1/7/2008 10:10 AM
unique		File Folder	1/7/2008 10:10 AM
vocab		File Folder	1/7/2008 10:10 AM
adlcp_v1p3.xsd	3 KB	XSD File	8/10/2006 10:56 AM
imscp_v1p1.xsd	17 KB	XSD File	8/10/2006 10:56 AM
imsmanifest	7 KB	XML Document	1/7/2008 10:10 AM
imsss_v1p0.xsd	3 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0auxresource.xsd	1 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0control.xsd	2 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0delivery.xsd	1 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0limit.xsd	3 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0objective.xsd	4 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0random.xsd	1 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0rollup.xsd	3 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0seqrule.xsd	5 KB	XSD File	8/10/2006 10:56 AM
imsss_v1p0util.xsd	4 KB	XSD File	8/10/2006 10:56 AM
lomStrict.xsd	5 KB	XSD File	8/10/2006 10:56 AM

**Figure 4.6 - Sample of SCORM ELearning objects**

The used SCORM learning object is consisted of the below file types

- XML
- XSD
- JPG
- HTML

## Comparison - Question and Criteria

The comparison for different Criteria and questions

- Support of other archive formats

**Table 4.1 – Support of other archive formats**

Tools	Zi p	7 Z	A C E	A R J	B Z 2	C A B	G Z	IS O	LZ H	TAR
WinRar	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
WinZip	Y	Y	N	P	Y	Y	Y	Y	Y	Y
WinAce	Y	N	Y	Y	N	Y	Y	Y	Y	Y
Power Archiver	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

- Solid archives, volumes, Recovery Volumes, Self – extracting archives and Encryption.

**Table 4.2 – Support of solid archives and others**

Tools	Solid Archive s	Volum es	Reco very Volu mes	Self- extractin g archives	Encryp tion
WinRar	Y	Y	Y	Y	Y
WinZip	Y	Y	Y	Y	Y
WinAce	Y	Y	Y	Y	Y
Power Archiver	N	Y	N	Y	Y

- Operating System Support

**Table 4.3 – Support of operating system**

Tools	Windows	Dos	Mac OS	Linux	Unix	Palm OS	Amiga OS
WinRar	Y	Y	Y	Y	Y	Y	N
WinZip	Y	Y	N	N	N	Y	N
WinAce	Y	Y	Y	Y	N	Y	N
Power Archiver	Y	N	Y	N	N	N	N

### **WinRar Tools**

WinRar is a 32-bit Windows version of the RAR archiver - a powerful tool, which allows you to create, manage and control archived files.

### **WinRar Version used**

WinRar 3.62

### **Compression of different file types**

Packing data using WinRar tools, including different types of files (XML, XSD)

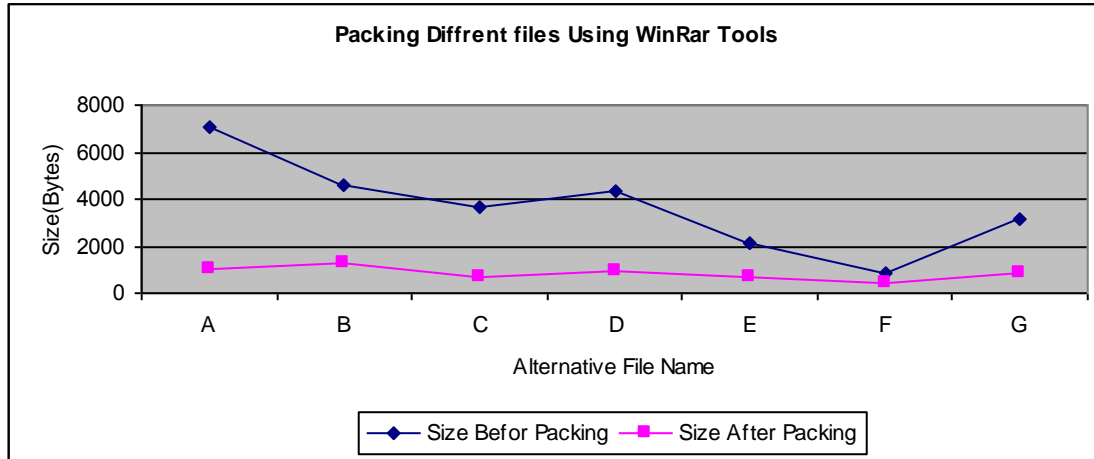
**Table 4.4 - Different files to be packed using WinRAR tools**

File name	Alternative Name	Type	Source data in (bytes)	Compressed data (Bytes)	Compression Ratio
imsmanifest.xml	A	Xml	7036	1051	14.9
lomStrict.xsd	B	Xsd	4618	1294	28
lmsss_v1p0util.xsd	C	Xsd	3691	723	19.5
lmsss_v1p0seqrule.xsd	D	Xsd	4341	938	21.6
lmsss_v1p0rollup.xsd	E	Xsd	2108	662	31.4
lmsss_v1p0random.xsd	F	Xsd	844	400	47.3
lmsss_v1p0objective.xsd	G	Xsd	3129	859	27.45

The average compression ratio	25767	5927	23.00
-------------------------------	-------	------	-------

As shown in the table 4.4, the total size before packing was 25767 byte, while the total size after packing is 5927; i.e. the compression ratio is 23.00.

When drawing the result using stacked line graph Figure 4.7, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.7 –Packing Different Files Using WinRAR Tools Compression for JPG files**

**Table 4.5 - JPG files to be packed using WinRAR tools**

File name	Alt er. Na me	Ty pe	Sou rce data in (byt es)	Co mp data (Byt es)	Compre ssion Ratio
Proposal_SCORM_slide0017_image020	H	JP G	6521	6258	95.9
Proposal_SCORM_slide0016_image018	I	JP G	6241	5609	89.8
Proposal_SCORM_slide0015_image016	J	JP G	6646	6291	94.6

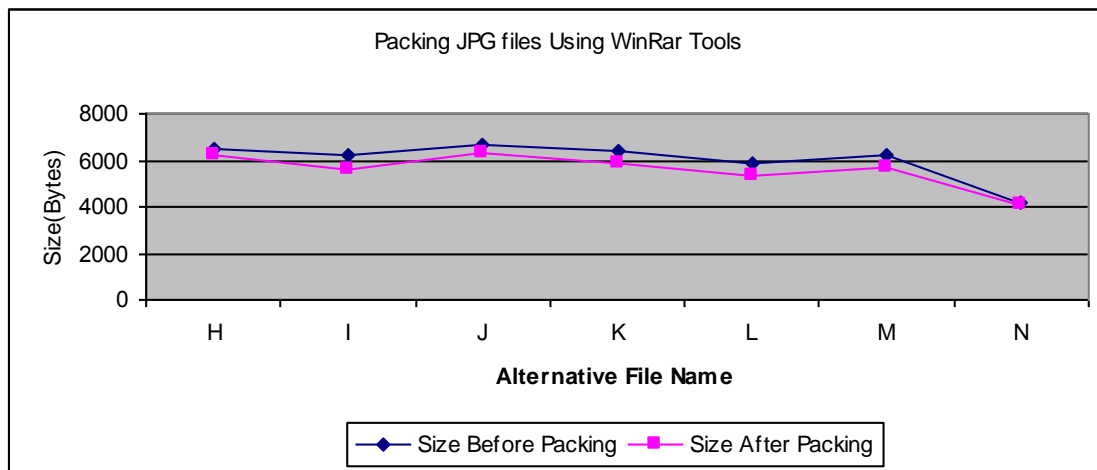


Proposal_SCORM_slide0014_image014	K	JP G	637 5	588 1	92. 2
Proposal_SCORM_slide0013_image012	L	JP G	588 8	534 1	90. 7
Proposal_SCORM_slide0012_image010	M	JP G	621 5	567 3	91. 2
Proposal_SCORM_slide0001_image002	N	JP G	419 2	405 4	96. 7

The average compression ratio	42078	39107	92.9
-------------------------------	-------	-------	------

As shown in the table 4.5, the total size before packing was 42078 byte, while the total size after packing is 39107; i.e. the compression ratio is 92.9.

When drawing the result using stacked line graph Figure (4.8), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.8 – Packing JPG Files Using WinRAR Tools**

## Compression for HTML files

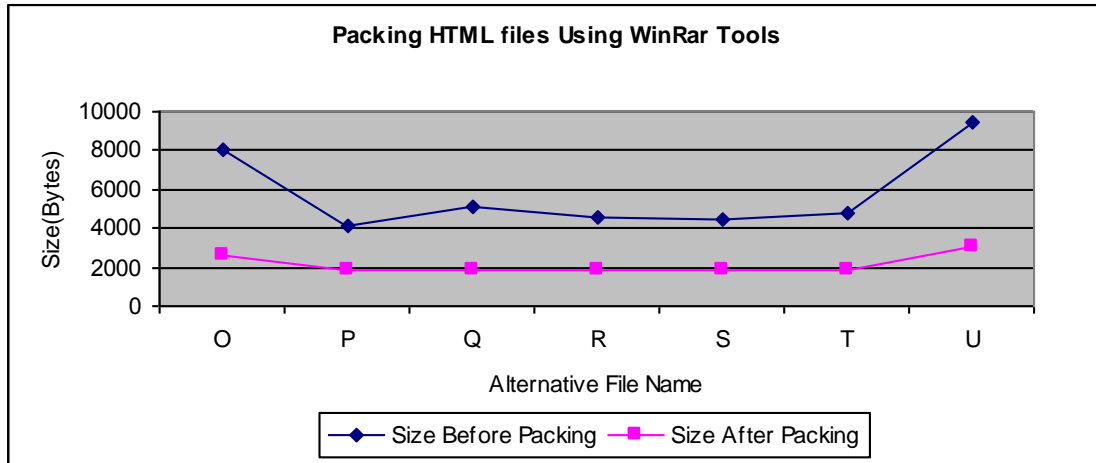
**Table 4.6 - HTML files to be packed using WinRar tools**

File name	Alter. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
Proposal_SCORM_slide0028	O	Htm	8069	2598	32.1
Proposal_SCORM_slide0027	P	Htm	4142	1806	43.6
Proposal_SCORM_slide0026	Q	Htm	5117	1828	35.7
Proposal_SCORM_slide0025	R	Htm	4584	1866	40.7
Proposal_SCORM_slide0024	S	Htm	4481	1817	40.5
Proposal_SCORM_slide0023	T	Htm	4741	1901	40
Proposal_SCORM_slide0022	U	Htm	9419	3096	32.8

The average compression ratio	40553	14912	36.7
-------------------------------	-------	-------	------

As shown in the table 4.6, the total size before packing was 40553 byte, while the total size after packing is 14912; i.e. the compression ratio is 36..

When drawing the result using stacked line graph Figure (4.9), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.9 – Packing HTML Files Using WinRAR Tools**

### Results for WinRAR Tools

Using WinRAR tool for packing XSD, XML, HTML files yield a good result; as the

Average compression ratio =29.85

When using WinRAR tool for packing JPG files, the compression ratio =92.9

Total Bytes before packing =108398 bytes

Total Bytes after packing = 59946 bytes

Average compression ratio = 55.3

## WinZip Tools

### WinZip Version used

WinZip 11.1

### Compression for different types of files

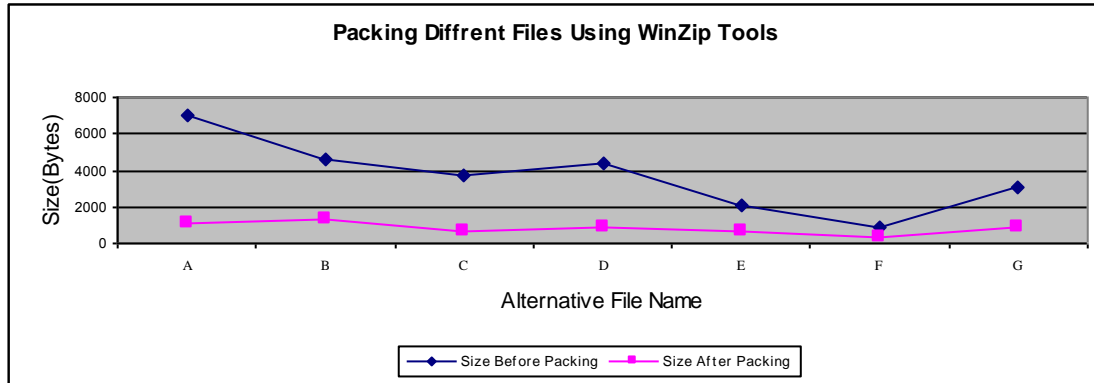
**Table 4.7 - HTML files to be packed using WinZip tools**

File name	Alternative Name	Type	Source data in (bytes)	Compressed data (Bytes)	Compression Ratio
imsmanifest.xml	A	Xml	7036	1064	15.12
lomStrict.xsd	B	Xsd	4618	1263	27.35
imsss_v1p0util.xsd	C	Xsd	3691	688	18.64
imsss_v1p0seqrule.xsd	D	Xsd	4341	911	20.99
imsss_v1p0rollup.xsd	E	Xsd	2108	634	30.08
imsss_v1p0random.xsd	F	Xsd	844	376	44.55
imsss_v1p0objective.xsd	G	Xsd	3120	832	26.67

The average compression ratio	25758	5768	22.39
-------------------------------	-------	------	-------

As shown in the table 4.7, the total size before packing was 25758 byte, while the total size after packing is 5768; i.e. the compression ratio is 22.39.

When drawing the result using stacked line graph Figure (4.10), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.10 – Packing Different Files Using WinZip Tools Compression for JPG files**

**Table 4.8 - JPG files to be packed using WinRar tools**

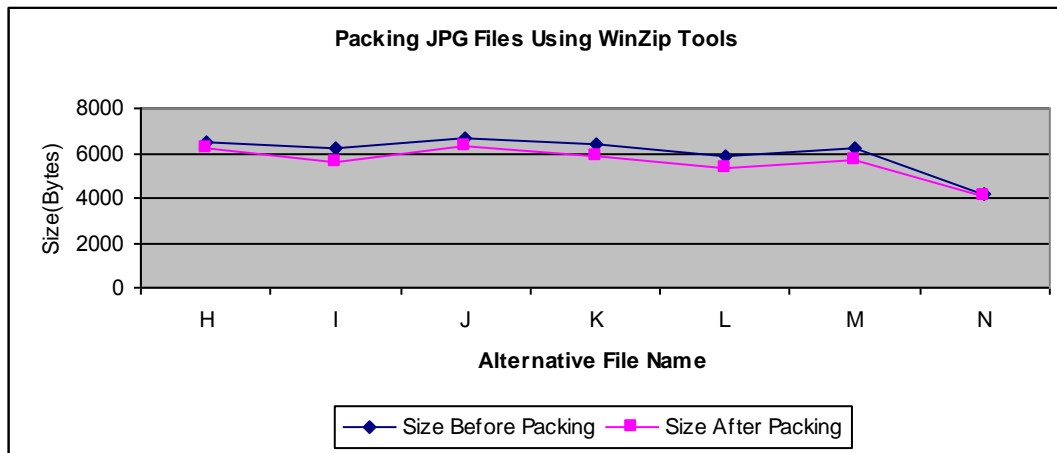
File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0017_image020	H	JPG	6521	6233	95.58
Proposal_SCORM_slide0016_image018	I	JPG	6241	5585	89.49
Proposal_SCORM_slide0015_image016	J	JPG	6646	6269	94.33

Proposal_SCORM_slide0014_image014	K	JP G	637 5	585 8	91.8 9
Proposal_SCORM_slide0013_image012	L	JP G	588 8	532 7	90.4 7
Proposal_SCORM_slide0012_image010	M	JP G	621 5	564 4	90.8 1
Proposal_SCORM_slide0001_image002	N	JP G	419 2	403 9	96.3 5

The average compression ratio      42078    38955    92.58

As shown in the table 4.8, the total size before packing was 42078 byte, while the total size after packing is 38955; i.e. the compression ratio is 92.58.

When drawing the result using stacked line graph Figure (4.11), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.11 – Packing JPG Files Using WinZip Tools**

## Compression for HTML files

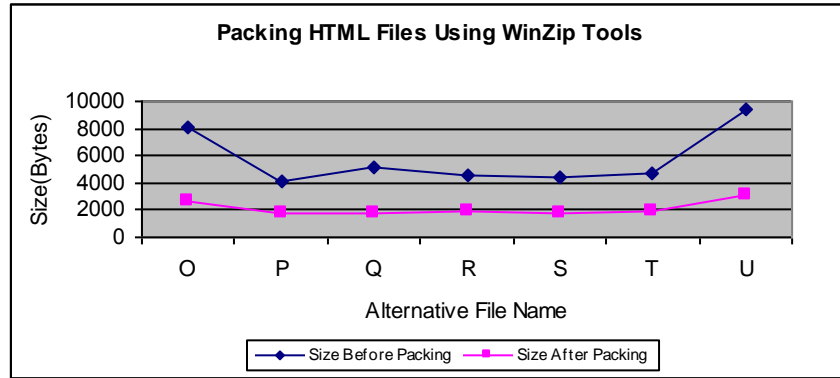
**Table 4.9 - HTML files to be packed using WinZip tools**

File name	Alter. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0028	O	Htm	8069	2600	32.22
proposal_SCORM_slide0027	P	Htm	4142	1787	43.14
proposal_SCORM_slide0026	Q	Htm	5117	1816	35.49
proposal_SCORM_slide0025	R	Htm	4584	1854	40.45
proposal_SCORM_slide0024	S	Htm	4481	1799	40.15
proposal_SCORM_slide0023	T	Htm	4741	1880	39.65
proposal_SCORM_slide0022	U	Htm	9419	3095	32.86

The average compression ratio	40553	14831	36.57
-------------------------------	-------	-------	-------

As shown in the table 4.9, the total size before packing was 40553 byte, while the total size after packing is 14831; i.e. the compression ratio is 36.57.

When drawing the result using stacked line graph Figure (4.12), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.12 – Packing HTML Files Using WinZip Tools**

### Result for WinZip Tools

Using WinZip tools for packing XSD,XML,HTML files yield a good result; as the

Average compression ratio =29.48

Using WinZip tools for packing JPG files the compression ratio =92.85

Total Bytes before packing: 108398 bytes

Total Bytes after packing: 59554 bytes

Average compression ratio = 50.51



## WinAce Tools

### WinAce Version used

WinAce 2.69

### Compression for different types of files

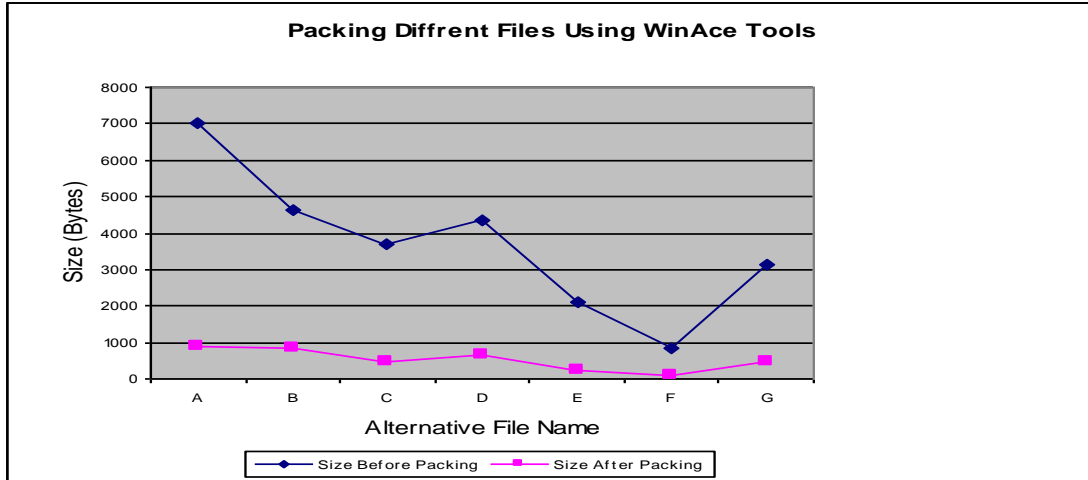
**Table 4.10 - Different files to be packed using WinAce tools**

File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
Imsmanifest.xml	A	Xml	7036	900	12.79
lomStrict.xsd	B	Xsd	4618	824	17.84
imsss_v1p0util.xsd	C	Xsd	3691	456	12.35
imsss_v1p0seqrule.xsd	D	Xsd	4341	632	14.56
imsss_v1p0rollup.xsd	E	Xsd	2108	232	11.01
imsss_v1p0random.xsd	F	Xsd	844	100	11.85
imsss_v1p0objective.xsd	G	Xsd	3120	460	14.74

The average compression ratio	25758	3604	13.99
-------------------------------	-------	------	-------

As shown in the table 4.10, the total size before packing was 25758 byte, while the total size after packing is 3604; i.e. the compression ratio is 13.99.

When drawing the result using stacked line graph Figure (4.13), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.13 – Packing Different Files Using WinAce Tools**

### Compression for JPG files

**Table 4.11 - JPG files used to packed using WinAce tools**

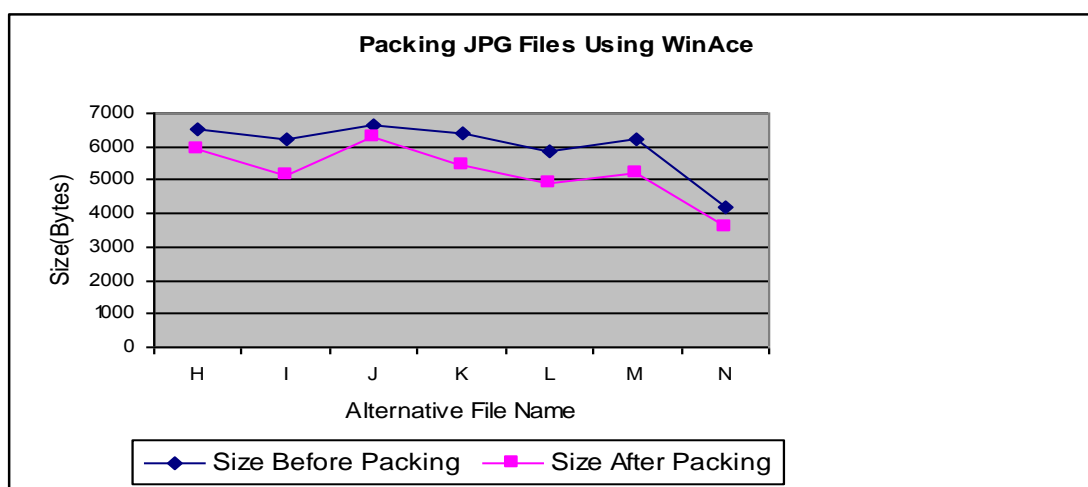
File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0017_image020	H	JPG	6521	5896	90
Proposal_SCORM_slide0016_image018	I	JPG	6241	5160	83

Proposal_SCORM_slide00 15_image016	J	JP G	6646	6260	94
Proposal_SCORM_slide00 14_image014	K	JP G	6375	5444	85
Proposal_SCORM_slide00 13_image012	L	JP G	5888	4884	83
Proposal_SCORM_slide00 12_image010	M	JP G	6215	5224	84
Proposal_SCORM_slide00 01_image002	N	JP G	4192	3588	86

The average compression ratio      42078    36456    87

As shown in the table 4.11, the total size before packing was 42078 byte, while the total size after packing is 36456; i.e. the compression ratio is 87.

When drawing the result using stacked line graph Figure 4.14, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.14 – Packing JPG Files Using WinAce**

## Compression for HTML files

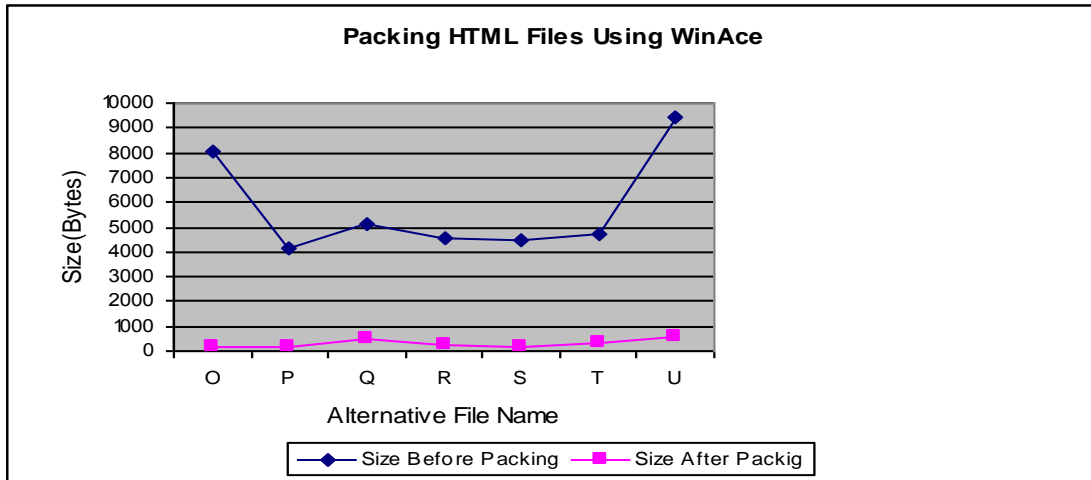
**Table 4.12 - HTML files to be packed using WinAce tools**

File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0028	O	HTML	8069	176	2.18
Proposal_SCORM_slide0027	P	HTML	4142	196	4.73
Proposal_SCORM_slide0026	Q	HTML	5117	472	9.22
Proposal_SCORM_slide0025	R	HTML	4584	228	4.97
Proposal_SCORM_slide0024	S	HTML	4481	180	4.02
Proposal_SCORM_slide0023	T	HTML	4741	296	6.24
Proposal_SCORM_slide0022	U	HTML	9419	556	5.9

The average compression ratio	40553	2104	5.19
-------------------------------	-------	------	------

As shown in the table 4.12, the total size before packing was 40553 byte, while the total size after packing is 2104; i.e. the compression ratio is 5.19.

When drawing the result using stacked line graph Figure 4.15, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.15 – Packing HTML Files Using WinAce**

### Results for WinAce Tools

Using WinAce tools for packing XSD,XML,HTML files yield a good result; as the

Average compression ratio =9.945

Using WinAce tools for packing JPG files the compression ratio =87

Total Bytes before backing = 108398 bytes

Total Bytes after backing = 59554 bytes

Average compression ratio = 35.63

## Power Archiver Tools

### Power Archiver Version

Power Archiver 4.56

### Compression for different types of files

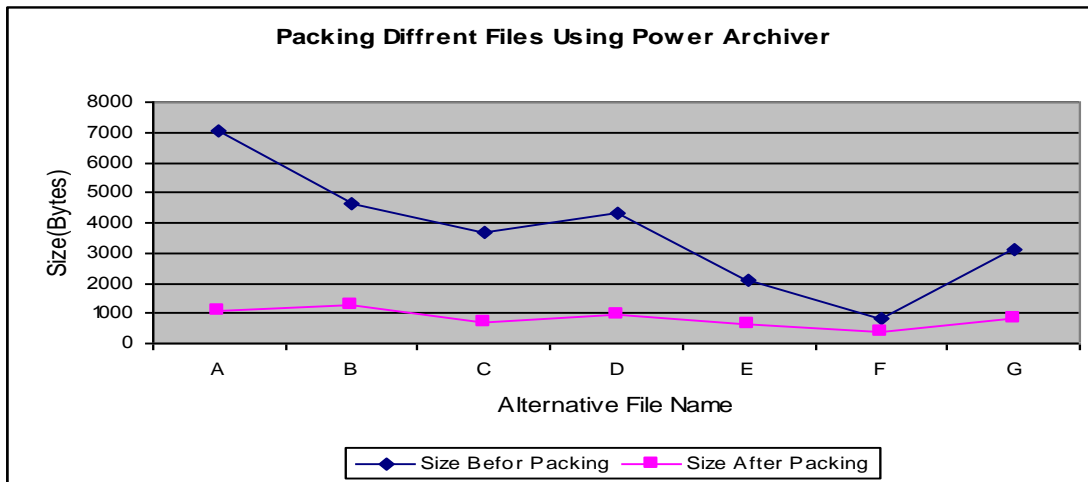
**Table 4.13 - Different files to be packed using Power Archiver tools**

File name	Alternative Name	Type	Source data in (bytes)	Compressed data (Bytes)	Compression Ratio
imsmanifest.xml	A	Xml	7036	1060	15.07
lomStrict.xsd	B	Xsd	4618	1267	27.44
imsss_v1p0util.xsd	C	Xsd	3691	691	18.72
imsss_v1p0seqrule.xsd	D	Xsd	4341	924	21.29
imsss_v1p0rollup.xsd	E	Xsd	2108	638	30.27
imsss_v1p0random.xsd	F	Xsd	844	377	44.67
imsss_v1p0objective.xsd	G	Xsd	3120	843	27.02

The average compression ratio	25758	5800	22.52
-------------------------------	-------	------	-------

As shown in the table 4.13, the total size before packing was 25758 byte, while the total size after packing is 5800; i.e. the compression ratio is 22.52.

When drawing the result using stacked line graph Figure 4.16, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 5.16 – Packing Different Files Using Power Archiver**

## Compression for JPG files

**Table 4.14 - JPG files to be packed using Power Archiver tools**

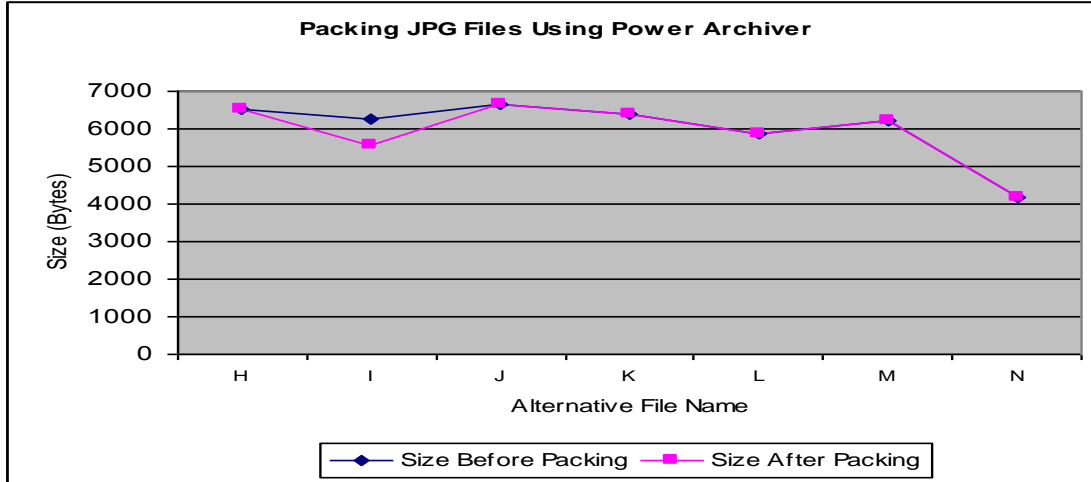
File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0017_image020	H	JPG	6521	6521	100
Proposal_SCORM_slide0016_image018	I	JPG	6241	5580	89.41
Proposal_SCORM_slide0015_image016	J	JPG	6646	6646	100
Proposal_SCORM_slide0014_image014	K	JPG	6375	6375	100
Proposal_SCORM_slide0013_image012	L	JPG	5888	5888	100
Proposal_SCORM_slide0012_image010	M	JPG	6215	6215	100
Proposal_SCORM_slide0001_image002	N	JPG	4192	4192	100

The average compression ratio      42078    41417    98.43

As shown in the table 4.14, the total size before packing was 42078 byte, while the total size after packing is 41417; i.e. the compression ratio is 98.43.



When drawing the result using stacked line graph Figure 4.17, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is slightly reduced after packing.



**Figure 4.17 - Packing JPG Files Using Power Archiver**

### Compression for HTML files

**Table 4.15 - HTML files to be packed using Power Archiver tools**

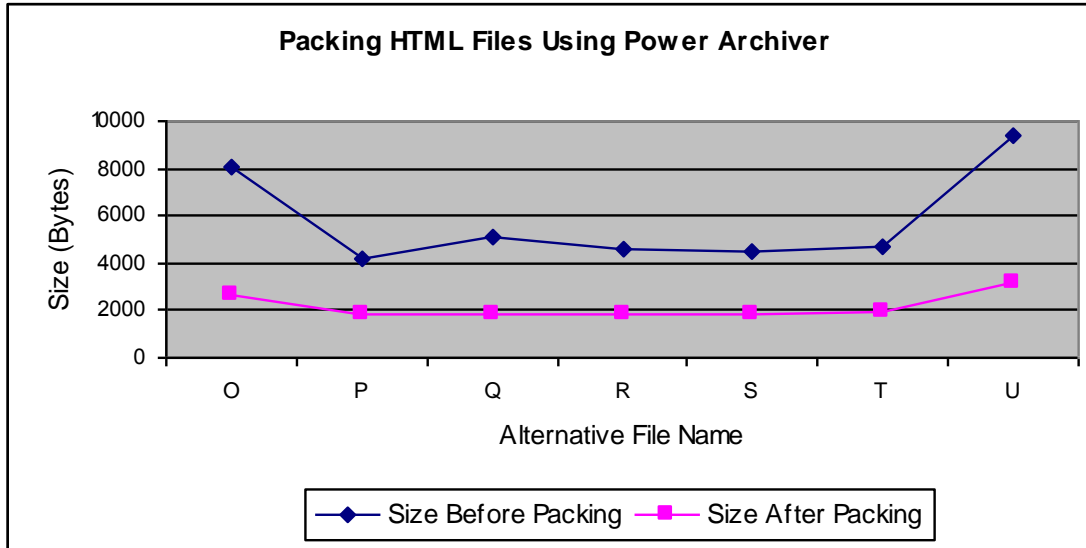
File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0028	O	HTML	8069	2626	32.54

Proposal_SCORM_slide002 7	P	HTM L	414 2	179 3	43.2 9
Proposal_SCORM_slide002 6	Q	HTM L	511 7	182 9	35.7 4
Proposal_SCORM_slide002 5	R	HTM L	458 4	186 1	40.6
Proposal_SCORM_slide002 4	S	HTM L	448 1	181 0	40.3 9
Proposal_SCORM_slide002 3	T	HTM L	474 1	189 1	39.8 9
Proposal_SCORM_slide002 2	U	HTM L	941 9	312 3	33.1 6

The average compression ratio	405 53	14933	36.82
-------------------------------	-----------	-------	-------

As shown in the table 4.15, the total size before packing was 40553 byte, while the total size after packing is 14933; i.e. the compression ratio is 36.82.

When drawing the result using stacked line graph Figure 4.18, two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 4.18- Packing HTML Files Using Power Archiver Results for power archiver Tools**

Using power archiver tools for packing XSD, XML, HTML files yields a good result; as the Average compression ratio =29.67

Using power archiver tools for packing JPG files the compression ratio =98.43

Total Bytes before backing: 108398 bytes

Total Bytes after backing: 62150 bytes

Average compression ratio = 52.59

#### 4.5 Comparison Result

This section presents the comparison result between the packing tools.

##### Compression ratio result

As shown in the table 4.16 (sorting the result in ascending order), the best average compression ratio in all tools was as follows: 35.63 for WinAce tools, then WinZip tools 50.51, then power archiver tools 52.59 and the last WinRar tools 55.3.

Using WinAce for packing SCORM learning objects proved to be the more suitable tool.

**Table 4.16 – Packing data result**

Tools	Total Bytes before backing	Total after Backing	Avg. Compression Ratio
WinAce	108389	42164	35.63
WinZip	108389	59554	50.51
power archiver	108389	62150	52.59
WinRar	108398	59946	55.3

### **Different advantages of using WinAce tools**

- Supports different types of archiving format
- Supports solid archiving
- Supports Multi Volumes
- Supports Self Extracting
- It can encrypt a file, and generate a password for that archived file
- Supports Recovery Record

In order to increase the performance, and make the compression file more secured, a password can be

created for the archived file; so that it can be only opened by authorized users. Another option to increase the compression rate is to choose a larger dictionary size (Compression Memory); which will require more memory during both compression and extraction processes.

## Chapter Five

### Implementing the deflate algorithms using VC#

#### 5.1. Introduction

Different types of algorithms are used to compress data. One of the algorithms used for compression is the deflate algorithm; to understand this algorithm, it is essential to understand the other two compression algorithms that make it up.

#### 5.2. The Huffman's Coding Algorithm

For each letter create a tree with a single root node and order all trees according to the probability of letter occurrence

While more than one tree is left

Take the two trees  $t_1, t_2$  with the lowest probabilities  $p_1, p_2$  and create a tree with probability in its root equal  $p_1 + p_2$  and with  $t_1$  and  $t_2$  as its subtrees;

Associate 0 with each left branch and 1 with each right branch

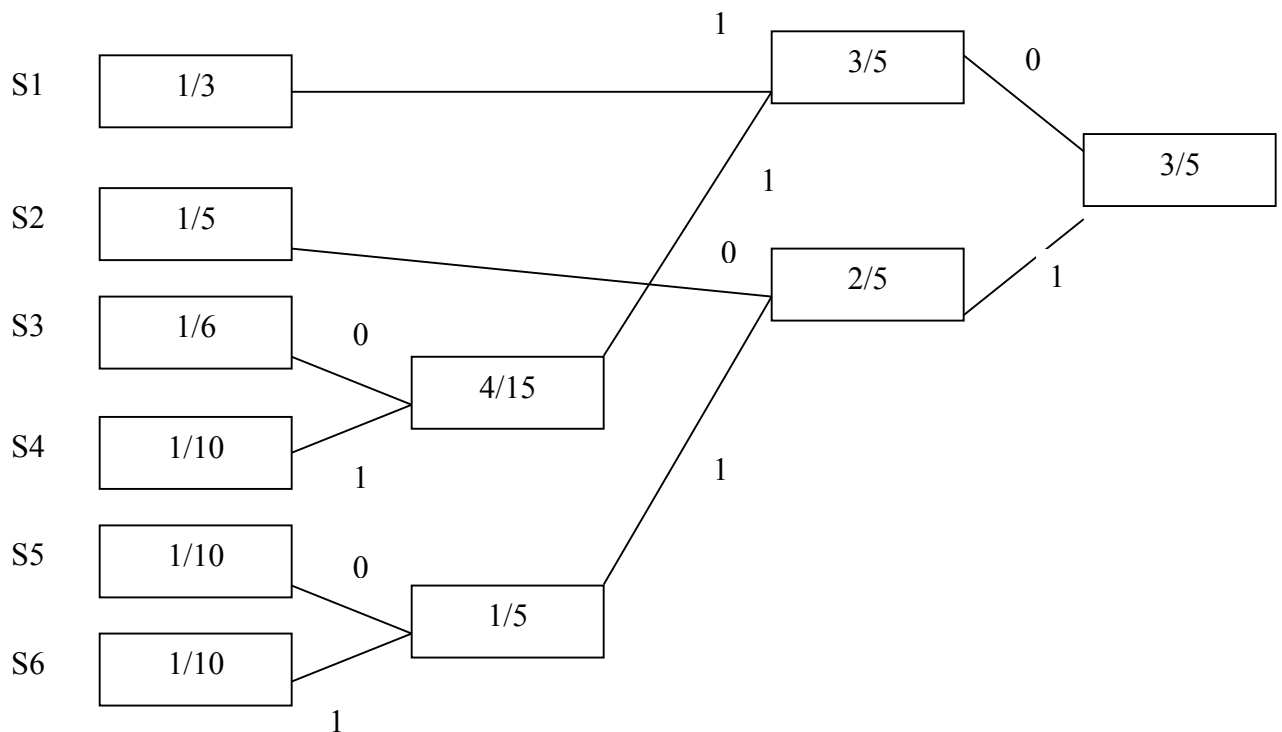
Create a unique codeword for each letter by

traversing the tree from the root to the leaf containing the probability corresponding to this letter and putting all encounters 0s and 1s together

The resulting tree has a probability of 1 in its root as shown in (Figure 5.1).

**Table 5.1– Huffman coding**

Symbol	S1	S2	S3	S4	S5	S6
Frequency	1/3	1/5	1/6	1/10	1/10	1/10
Code	00	10	010	011	110	111



**Figure 5.1- Huffman tree**

**The Compression Procedure**

The compression algorithm is consisted of four steps:

1. Scan the input stream and obtain the frequencies for each symbol. Save that information.
2. Build Huffman's tree and extract the code words.
3. Input a symbol from the stream and output its code.
4. Repeat step 3 until the end of stream.

### **The Decompression Procedure**

The decompression algorithm is consisted of five steps:

1. Build Huffman's tree from the saved information.
2. Move the pointer to the root node.
3. For each input bit follow the corresponding label down the tree; until a leaf node is reached.
4. Output the symbol at that leaf node.
5. Repeat the algorithm from step 2; until the end of the stream.

### **5.3. LZ77 Compression**

In this method, the buffer is divided into two parts. The first part, called a dictionary buffer, is a buffer of L1 positions, which holds the L1 most recently encoded symbols from the input; the second part called a lookahead buffer, is an L2 position buffer containing the L2 symbols about to be encoded. To initiate the process, the dictionary buffer is filled with with L1 copies of the first symbol of the input. In each



iteration, the dictionary buffer is searched for a substring matching a prefix of a string located in the lookahead buffer. If such a match is found, a code word is transmitted, which is a triple  $\langle p, l, s \rangle$  composed of the position  $p$  in which the match was found, the length  $L$  of the match, and the first mismatching symbols following the prefix. Then, the entire content of the buffer (dictionary buffer and lookahead buffer) is shifted to the left by the length of match plus one; some symbols are shifted out, and some new symbols from the input are shifted in.

Consider the case when  $L_1=L_2=4$  and inputs is a string

Positions in the buffer are indexed with  $0 \dots 7$ .

1. The initial situation is shown at the top of Figure 5.1.
2. The first symbol of the input is a, and the position 0 through 3 are filled up with a's. In the remaining positions, the first four symbols of the input are placed, aaba. The longest prefix matching any substring which begins in any position between 0 and 3 is aa. Therefore, the triple  $\langle 2, 2, b \rangle$  is generated, or simply 22b: The match starts in position 2, the match is two symbols long, and the symbol following the prefix is b. In Figure 5.1, the

3. match in the dictionary buffer is underlined with a dashed line, and the prefix equal to the match is underlined with a solid line. Next, a left shift occurs, three a's are shifted out, and string bac is shifted in.

**Table 5.2– Encoding string using LZ77**

Input	buffer	Output
aababacbaacbaadaaa....	aaaa	a
aababacbaacbaadaaa...	aaaa <u>aba</u>	22b
Abacbaacbaadaaa	aa <u>ab</u> <u>ab</u> ac	23c
Baacbaadaaa	<u>a</u> <u>ba</u> <u>ca</u> baac	12a
Cbaadaaa	<u>c</u> <u>ba</u> <u>ac</u> baa	03a
daaa....	cbaadaaa	30d
aaa...	.....	

4. The longest match for a prefix also starts in position 2 and is three symbols long, aba, with c following the prefix. The issued triple is 23c. Note that the prefix aba and the match overlap; that is, the match can cross the boundary between the two parts of the buffer. Afterwards, aaab is shifted out and baac is shifted in.
5. the triple 12a is output and the content of the buffer is updated accordingly.

6. The longest match of a prefix is four symbols long, but because the third element of each triple should be a letter following the prefix, the match is considered three symbols long, and the last letter of the match, a, becomes the third element of the triple 03a.
7. The prefix d does not have any match in the dictionary buffer; therefore. The length l of the match in the generated triple is 0. The starting position is 3, which is unimportant. It could be any number.

### **The Decoding Example**

Consider the sequence of triples output by the encoder :  
a22b23c12a03a30d.....

1. The first symbol, a, is used to fill the dictionary buffer.
2. After receiving the triple 22b, the decoder copies two symbols starting from position 2 to the lookahead buffer and attaches b after it finishes copying. The string aab is also output as part of the message being decoded. Then, the buffer is shifted by  $1+1=3$  positions to the left to make room for up to four symbols in the lookahead buffer.
3. To decode the triple 23c, three symbols are copied starting from the position 2. Note that there is an overlap here between the string copied from the dictionary buffer

4. and the string generated in the lookahead buffer. The third symbol of the buffer, a, is copied to the beginning of the lookahead buffer from which it is again copied to its third position.

**Table 5.2 – Decoding a stream of triples using LZ77 algorithms**

input	Output	buffer	Buffer after shifting
a		aaaa	
22b	Aab	aaaaaab	aaab
23c	Abac	aaababac	abac
12a	Baa	abacbaa	cbaa
03a	Cbaa	<u>cbaacbaa</u>	cbaa
30d	D	cbaad	baad

Putting it all together

The deflate compressor is given a great deal of flexibility, as to how to compress the data. There are three modes of compression:

- 1) Not compressed at all: This is an intelligent choice of data that have been already compressed. Data stored in this mode will expand slightly, but not as much as it would if it were already compressed; and one of the other compression methods was tried upon it.

Compression first with LZ77 and then with Huffman coding. The trees that are used to compress in this mode are defined by the Deflate specification itself; and so no extra space needs to be taken to store those trees.

Compression, first with LZ77 and then with Huffman coding with trees that the compressor creates and stores along with the data.

The data is broken up in blocks, and each block uses a single mode of compression. If the compressor wants to switch from non-compressed storage to compression with the trees defined by the specification, to compression with specified Huffman trees, or to compression with a different pair of Huffman trees, the current block must be ended and a new one begins.

Once the raw data has been turned into a string of characters, special length and distance pairs, these elements must be represented with Huffman codes. Call the point where we start reading in bits a "dial tone".

At that dial tone, one of three things could follow: a character, a length-distance pair, or the end of the block. Since we must be able to tell which it is, all the possible characters, elements that indicate ranges of possible lengths and a special end-of-block indicator are all merged into a single alphabet. That alphabet becomes the basis of a Huffman tree.

Distances don't need to be included in this alphabet; since they can only appear directly after lengths. If we got the end-of-block symbol, we're either at the beginning of another block, or at the end of the compressed data.

DEFLATE specification is the way trees are encoded to go along with the data; when that data is compressed with specialized trees. The trees are transmitted by their code lengths. The code lengths are put all together into a sequence of numbers between 0 and 15 (the Huffman trees that are created must be kept to code lengths of no more than 15).

Not all the elements have to be given code lengths; if the last elements of an alphabet are of 0 code lengths, they can and probably should be left out.

The number of elements in each of the two alphabets will be transmitted; so the trimmed alphabets go together into a single sequence.

Once this sequence of code lengths is assembled, it is compressed with a form of what is called run-length compression. When several elements in a row have the same code length (often 0) special symbols may be used to indicate the number of elements with this code length. Our sequence is now a sequence of numbers between 0 and 18 (possibly with extra bits forming integers to modify base values; as was the case with the length and distance codes).

This Huffman tree needs to be included with the data. Like the other Huffman trees, it will be included by recording the code lengths of the elements. They are not recorded in the order 0, 1, 2, 3, 4 ... 16, 17, 18. They are recorded in a special order: 16, 17, 18, 0, 8, 7, 9, 6, 10, 5, 11, 4, 12, 3, 13, 2, 14, 1, 15. The logic behind this is that the elements near the end of sequence are most likely to have 0 code lengths

(i.e. not be in the tree at all). If they are indeed 0, they can be trimmed from the end of the sequence; the number of elements that are being directly recorded is also included in the data.

#### 5.4. Deflate algorithms programs

In the thesis we implement the deflate algorithms; using Visual C# to evaluate the compression showing the effect of this algorithms in compressing the SCORM learning object.

To compress the SCORM learning objects in the deflate program, the following steps are executed:

- Create a file (XIP file) as shown in Figure (5.2)

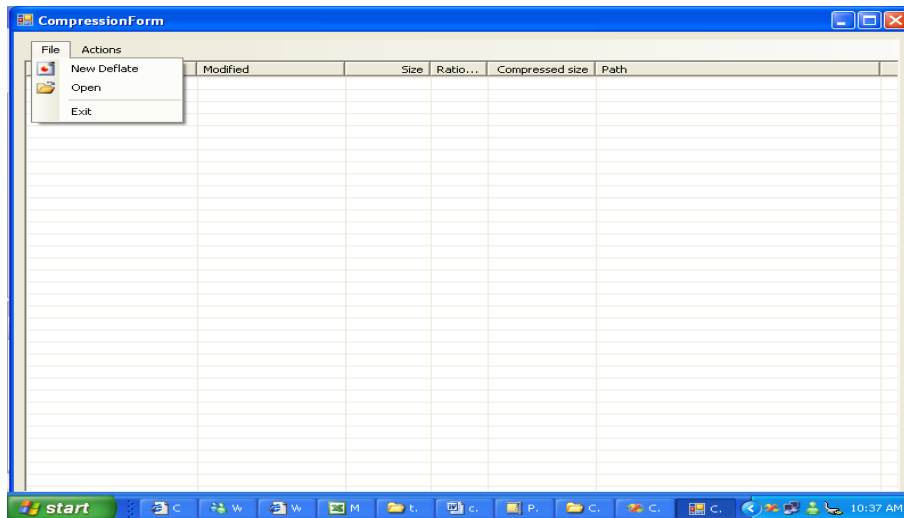
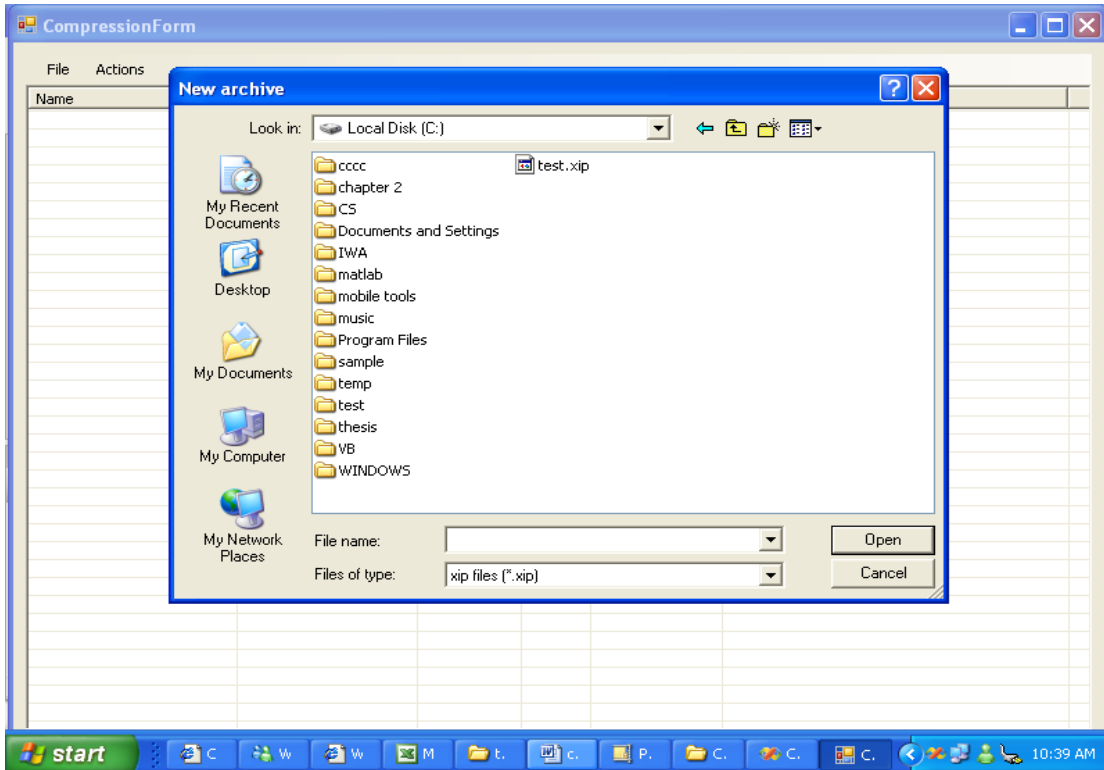


Figure 5.2 - create XIP files



- Select a new deflate

The following screen is displayed, as shown in Figure (5.3).



**Figure 5.3 - Write the name of the file you want to create**

- In the "Look in" field, select the drive you want to create the GZIP file in; as shown in Figure (5.3) above:
- In the "File Name", write the name of the file (compression file)
- Now select the files to be compressed

## . Compression for different type of files

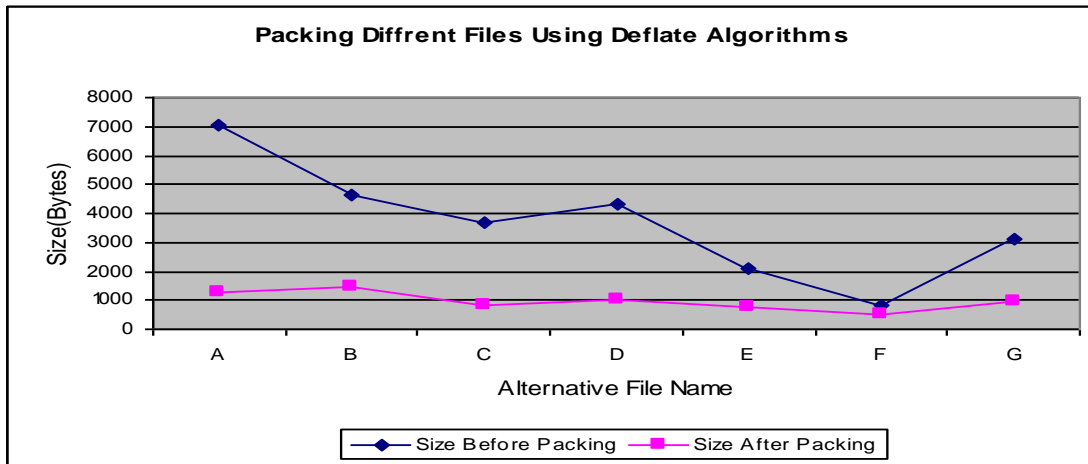
**Table 5.3 - Different files to be packed using deflate algorithms**

File name	Alternative Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
imsmanifest.xml	A	Xml	7036	1246	17.71
lomStrict.xsd	B	Xsd	4618	1450	31.4
lmsss_v1p0util.xsd	C	Xsd	3691	844	22.87
lmsss_v1p0seqrule.xsd	D	Xsd	4341	1041	23.98
lmsss_v1p0rollup.xsd	E	Xsd	2108	755	35.82
lmsss_v1p0random.xsd	F	Xsd	844	490	58.06
lmsss_v1p0objective.xsd	G	Xsd	3120	955	30.61

The average compression ratio	25758	6781	26.33
-------------------------------	-------	------	-------

As shown in the table 5.4, the total size before packing was 25767 byte, while the total size after packing is 6781; i.e. the compression ratio is 26.33.

When drawing the result using stacked line graph Figure (5.4), two lines will appear; the blue (data before packing) and the pink (data after packing), showing that the size of files is reduced after packing.



**Figure 5.4 – Packing Different Files Using Deflate Algorithms**

### Compression for JPG files

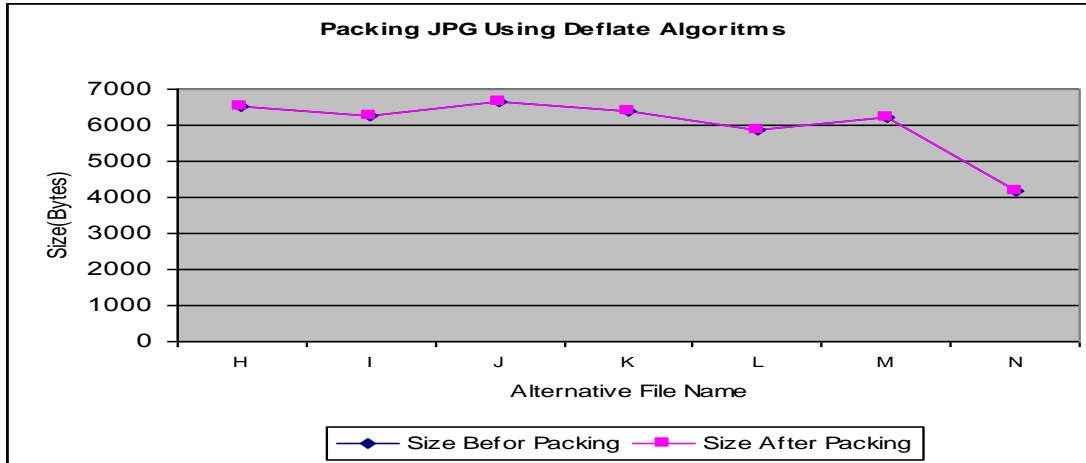
**Table 5.4 - JPG files to be packed using deflate algorithms**

File name	Alt. Name	Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio		
proposal_SCORM_slide0017_image020				H JP G	652 1	652 1	10 0
proposal_SCORM_slide0016_image018				I JP G	624 1	624 1	10 0
proposal_SCORM_slide0015_image016				J JP G	664 6	664 6	10 0
proposal_SCORM_slide0014_image014				K JP G	637 5	637 5	10 0
proposal_SCORM_slide0013_image012				L JP G	588 8	588 8	10 0
proposal_SCORM_slide0012_image010				M JP G	621 5	621 5	10 0
proposal_SCORM_slide0001_image002				N JP G	419 2	419 2	10 0

The average compression ratio	42078	2078	100
-------------------------------	-------	------	-----

As shown in the table 5.5 the total size before packing is 42078 byte, while the total size after packing is 2078 the compression ratio is 100.

When drawing the result using stacked line with marker graph, as shown in Figure (5.5), two lines appear the blue (data before packing) and the pink (data after packing) remains the same before and after packing



**Figure 5.5 – Packing JPG Using Deflate Algorithms**

### Compression for HTML files

**Table 5.6 - HTML files used to compress by deflate algorithms**

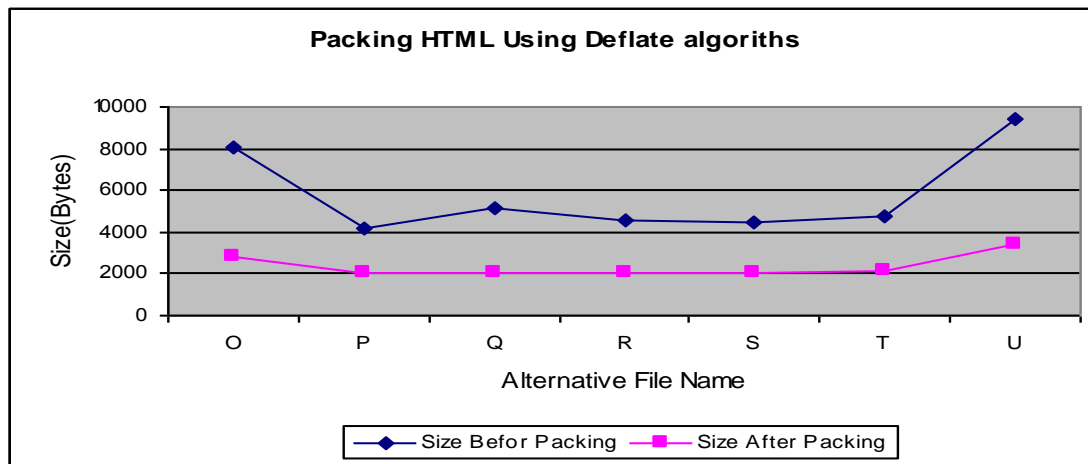
File name		Type	Source data in (bytes)	Comp data (Bytes)	Compression Ratio
proposal_SCORM_slide0028	O	HTML	8069	2834	32.22
proposal_SCORM_slide0027	P	HTML	4142	2009	43.14
proposal_SCORM_slide0026	Q	HTML	5117	2041	35.49
proposal_SCORM_slide0025	R	HTML	4584	2081	40.45

proposal_SCORM_slide002 4	S	HTM L	448 1	202 2	40.1 5
proposal_SCORM_slide002 3	T	HTM L	474 1	210 1	39.6 5
proposal_SCORM_slide002 2	U	HTM L	941 9	335 0	32.8 6

The average compression ratio	40553	16438	40.53
----------------------------------	-------	-------	-------

As shown in the table 5.6 the total size before packing is 40553 byte the total size after packing is 16438 the compression ratio is 40.53

Drawing the result using stacked line with marker graph Figure (5.6) two lines appear the blue (data before packing) and the pink (data after packing) the size of files going down after packing



**Figure 5.6 - Packing HTML Using Deflate Algorithms**  
Using Deflate algorithms for packing XSD, XML, HTML files getting a good result the

Average compression ratio =33.43

Using Deflate algorithms for packing JPG files the compression ratio =100

Total Bytes before backing = 108398 bytes

Total Bytes after backing = 65297 bytes

Average compression ratio = 55.62

Comparing the results between WinAce and the deflate algorithms table 5.7, the WinAce tools still has the best compression ratio

**Table 5.7 - Comparison between WinAce and the deflate algorithms**

Algorithms and tools	Total Byte Before Zipping	Total Byte after Zipping	Average Compression Ratio
Deflate algorithms	108389	65297	55.63
WinAce tools	108389	42164	38.9

## **Chapter Six**

### **Conclusion and Future Work**

#### **6.1. Introduction**

The aim of this thesis was to convert the PC SCORM to PDA SCORM, which encounters a lot of problems in the area of converting it to PDA SCORM.

The proposed solution for packing the data has been tested; where different type of tools and algorithms were used; in order to select the more suitable tool for packing the SCORM learning objects.

The conclusion of each part of this thesis is summarized, and recommendations for future work and suggested studies are presented; by means of this thesis.

#### **6.2. Research Conclusions**

This section will go through all chapters of the thesis; including introduction, backgrounds, related studies and works. Furthermore, it will conclude with the development of Deflate algorithm, and a comparative study between the proposed solutions.



### 6.2.1. In General

Converting the PC SCORM to Pocket SCORM has an important role; especially in Elearning software. In this thesis, the model suggested by Professor Timothy K. Shih, Pocket SCORM is concentrating at different parts of the pocket SCORM; then different questions and criteria were suggested to explain in details the more suitable tools for packing the SCORM learning objects; each question was answered; concentrating on the compression ratio which has a strong effect on reducing the size of the SCORM learning objects in a way to reduce the network traffic. By selecting the more suitable tool, reducing the size of the SCORM learning object shall have a strong effect on the network traffic, speed and cost.

The comparative study between different tools used for packing the data and algorithms was positive, and resulted in selecting the more suitable one for packing the SCORM learning object.

### **6.3. Recommendations for Future Work**

This section includes the suggestions that can be applied to future works. In addition, these recommendations will encourage the researchers to present further studies in this research fields.

#### **6.3.1.Future work for the research**

For future work, the proposed solution can be further improved by carrying out the following:

- 1) Completing the PC Dock and SCORM LMS Server that supports Pocket SCORM Service API.
- 2) Finding a solution to integrate the LMS server, with other learning management servers

## Bibliography

- [1] Antonella Grasso, Teresa Roselli, "Guidelines for Designing and Developing Contents for Mobile Learning ".IEEE Computer Society Press,2005.
- [2] B. Eckel, thinking in C++, 2nd Edition, Volume 1, Prentice Hall, Upper Saddle River, NJ, 2000.
- [3] Brent Smith, William Beki."Pocket PC Technology and advanced distributed learning environment". U.S Army Simulation.
- [4] Carmack, Carmen, and Craig Freudenrich, Ph.D. "How PDAs Work." 12 June 2003. HowStuffWorks.com. <http://communication.howstuffworks.com/pda.htm>
- [5] David Parsons, Hokyoung Ryu, Mark Cranshaw, "A Study of Design Requirements for Mobile Learning Environments". IEEE Computer Society Press,2006.
- [6] David Wirth, Jennifer Brooks. "An Introduction to the advanced Distributed Learning Initiative ". ADL CO-LAB, 2004.
- [7] G. K. Wallace, The JPEG still picture compression standard, Communications of the ACM, 34(4):32-44, April 1991.
- [8] Hui-huang , Timothy K. Shih, and others. "Pocket SCORM". IEEE Computer Society Press, 2004.
- [9] IEEE 1484.11.2 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication. November 10, 2003 Available at: <http://www.ieee.org/>

[10] Khan, B.H. (2005). Learning features in an open, flexible, and distributed environment . AACE Journal, 13(2), 137-153.

[11] Kiyoshi Nakabayashi, Takahide Hoshide, Masanobu Hosokawa, Taichi Kawakami, Kazuo Sato, "Design and Implementation of a Mobile Learning Environment as an Extension of SCORM 2004 Specifications".IEEE Computer Society Press,2007.

[12] M. Nelson, J. L. Gailly, The Data Compression Book, M&T Books, New York, 1996.

[13] M. Nunes, R.Pasley, M. McPherson, H.Thomas,T.Ishaya, "Learning Objects: Are they Serving Practicioners Working with VLES?"; Proc. of IADIS WWW2003, pp.728-735, Portugal, October 2003.

[14] Manuel Castro et al," Examples of Distance Learning Projects in the European Community"; IEEE Transactions on Education, Vol. 44, No. 4, pp. 406-410, 2000.

[15] Microsoft Corp., Support Page for Windowstm Media Formats,  
<http://support.microsoft.com/default.aspx?scid=/support/mediaplayer/wmptest/wmptest.asp#Windows%20Media>,  
Accessed: 02/10/2003.

[16] Oliver Bohl, Dr. Jörg Schellhase, Ruth Sengler, Prof. Dr. Udo Winand , "The Sharable Content Object Reference Model (SCORM) – A Critical Review ".IEEE Computer Society Press,2002.

- [17] Polyxeni Arapi, Nektarios Moutzidis, Stavros Christodoulakis, " Supporting Interoperability in an Existing e-Learning Platform Using SCORM ".IEEE Computer Society Press,2003.
- [18] Przemysław Skibiński , "Improving HTML compression".IEEE Computer Society Press,2008.
- [19] Robert Yu-Liang Ting. "Mobile Learning: Current Trend and Future Challenges". IEEE Computer Society Press, 2005.
- [20] SCORM 2004 3rd EDITION Sharable Content Object Reference Model Content Aggregation Model, advanced Distributed Learning.
- [21] SCORM 2004 3rd EDITION Sharable Content Object Reference Model Run-Time Environment, advanced Distributed Learning.
- [22] SCORM 2004 3rd EDITION Sharable Content Object Reference Model Sequencing and Navigation, advanced Distributed Learning.
- [23] SCORM 2004 3rd EDITION Sharable Content Object Reference Model, 2006 advanced distributed learning.
- [24] Shueh-Cheng Hu, "Application of the UML in modeling SCORM-conformant Contents".IEEE Computer Society Press,2005.
- [25] Tomasz Bartczak, Patryk Ziobron, Stanisław Paszczyński , "Improvement of Computer Network Transmission by Packing Agent Technology".University of Information Technology and Management, Rzeszów, Poland,2007.

