Using Computers to Facilitate Formative Assessment of Open-Ended Written Assignments

A thesis presented in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

at Massey University, Palmerston North, New Zealand

Jun ZHANG

2005

Abstract

This thesis presents an e-learning solution to facilitate formative assessment of electronically submitted open-ended written assignments.

It is widely accepted that formative assessment is highly beneficial to student leaning. A number of researchers are active in developing specialized approaches and software systems for assisting formative assessment of student work. However, no comprehensive e-learning solution exists for facilitating formative assessment of students' open-ended written work. The project presented in this thesis has developed a new approach for using computers to facilitate formative assessment of electronically submitted open-ended written assignments.

Based on the literature review of the education theories around formative assessment and current computer software technologies, this project has developed three principles for e-learning support for formative assessment of open-ended written assignments:

1. It needs to facilitate all the activities that are potentially required for formative assessment of student assignments (for example, the creation of assessment criteria, the submission of assignments, and the analysis of the assessment results), not only the marking activity to create feedback on assignments.
2. It needs to provide an onscreen marking tool which enables human markers to mark open-ended written assignments in an intuitive and efficient way by replicating their paper-based assessment approaches.
3. It needs to provide a generic solution for facilitating formative assessment of open-ended written assignments from all disciplines, not a limited solution restricted to some specific domains (for example, computers science or business courses).

Based on these principles, a specification of an e-learning system for facilitating formative assessment of open-ended written assignment was developed and a system was implemented accordingly. This system, called the Written Assignment

Assessment (WAA) system, has been already used in the assignment marking of several courses at Massey University.

Acknowledgements

With a deep sense of gratitude, I wish to express my sincere thanks to my supervisor, Dr Eva Heinrich, for all the help, encouragement, and guidance given to me during this research project.

A special thank goes to Alex Lawn for his valuable suggestions on the implementation of the MARK application.

I would like to thank the computer support staff in the Institute of Information Sciences and Technology at Massey University. Without their help, I could not have installed the MARK application on the computers of the institute laboratory to make the application available to students.

Thanks go to all my colleagues in the Institute of Information Sciences and Technology at Massey University for the encouragement and help given to me. In particular, I am grateful to Jiayi Lu for her advice on the user interface design of the WAA system.

A heartfelt thank you goes to my love, Yun Peng, for her constant encouragement and enduring patience throughout the duration of this research project.

Finally, I would like to thank all whose direct and indirect support helped me completing my research and thesis in time.

Publications

Publications related to this research are:

Zhang, J., Heinrich, E. (2005). Using Computers to Support Formative Assessment of Assignments. *Proceedings of Ed-Media2005 World Conference on Educational Multimedia, Hypermedia & Telecommunications*. P. Kommers, G. Richards (Eds.), Association for the Advancement of Computing in Education, Norfolk, USA, pp4510 - 4515.

Zhang, J., Heinrich, E. (2005). A System Designed to Support Formative Assessment of Open-Ended Written Assignments. *Proceedings of the 5th IEEE International Conference on Advanced Learning Technologies 2005 (ICALT 2005)*. Kaohsiung, Taiwan, pp88-92.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

Assessment is the process of gathering information about student performances and using the collected information in further teaching and learning activities. The importance of assessment to education has been iterated by many researchers. It is generally agreed that assessment is a critical and indispensable part of education (Biggs, 1999; Boud, 1995; Brown *et al*., 1997; Erwin and Knight, 1995; Freeman and Lewis, 1998; Rowntree, 1987). Erwin and Knight (1995:181) argue that assessment is the single most important influence on learning:

> "If all other elements of the course point in one direction and the assessment arrangements in another, then the assessment arrangements are likely to have the greatest influence on the understood curriculum."

One aspect of the importance of assessment to education is reflected by the various educational goals it can serve, for example, to select students for higher level study, to describe and certify student achievements, or to improve learning and teaching. Although a wide range of different purposes underlie assessment, it is widely accepted that the primary purpose of assessment in education should be to improve student learning (Brown and Knight, 1994; Forum, 1995; Freeman and Lewis, 1998; Rowntree, 1987). Evidence of a number of studies reveals that this goal can be achieved successfully by using formative assessment properly (Black and Wiliam, 1998a; Crooks, 1988).

The research project presented in this thesis aims to develop an efficient way for using computers to facilitate formative assessment of electronically submitted, open-ended written assignments. The remainder of this chapter provides an overview of the research project by presenting the motivating forces behind the project, the research objective, the steps involved in the project, and the structure of the thesis.

1.1 Motivating Forces

This research project is motivated by the fact that current e-learning software provides only limited support for formative assessment of students' open-ended written work.

1.1.1 What Is Formative Assessment

Assessment can be defined as "formative" or "summative" in reference to its purpose. Brown and Knight (1994:15) define these two types of assessment as:

> "**Formative assessment** is where the purpose is to get an estimate of achievement which is used to help in the future learning process. Formative assessment also includes course-work where students receive feedback which helps them to improve their next performance; discussion between a mentor and a student; and end-of-module examinations whose results are used to identify areas for attention in later modules."
>
> "**Summative assessment** includes end-of-course assessment and essentially means that this is assessment which produces a measure which sums up someone's achievement and which has no other real use except as a description of what has been achieved."

Summative assessment always leads to the final judgement of students' learning performance or achievement (in most cases, with the format of assigning alphabetic grades or numeric marks to students' work) and therefore is termed as assessment ***of*** learning. Formative assessment (assessment ***for*** learning) focuses on providing students with useful feedback (for example, well worded textual comments on students' written work) that they can interpret and act on to improve their future performances (Freeman and Lewis, 1998). The feedback in formative assessment needs to inform students of the learning goals (what they are expected to achieve), their current levels of knowledge (what they have and especially have not achieved), and the strategies how to improve their further learning. Self-assessment and peer-assessment are two special formats of formative assessment in which students, not teachers or markers, are in charge of providing formative feedback on their own or their fellow students' work.

It is the consensus among educational researchers that student learning can be improved significantly by using formative assessment, including self-assessment and peer-assessment (Black and Wiliam, 1998b; Brookhart 2001; Brown and Knight, 1994; Freeman and Lewis, 1998; James, 1998; Natriello, 1987). This is supported by the strong evidence of a number of studies (Black and Wiliam, 1998a; Crook, 1988;

Fontana and Fernandes, 1994). These researchers also emphasize that only high-quality and properly implemented formative assessment can be productive in promoting student learning, for example, providing students with detailed and relevant feedback, using explicit assessment criteria and associating feedback with these criteria, and encouraging assessment related communication between students and teachers. A more detailed description of the relationship between formative assessment and student learning as well as how to ensure the quality of formative assessment will be presented later in the literature review part, Chapter 2, of this thesis.

1.1.2 Using Computers in Assessment

Nowadays, computers technologies are widely used in various assessment related activities to help teachers to carry out assessment more efficiently. For example, a number of word processing tools, such as Microsoft Word and OpenOffice.org Writer, can be used by teachers to create the detailed requirements and marking schemes for various assessment tasks. Most teachers are familiar with using spreadsheet applications, like Microsoft Excel, to record the marks of their students. Email applications enable students to discuss various assessment related issues with teachers without having to go to see teachers in their offices. In addition to these general purpose software applications, a variety of Computer-Assisted Assessment (CAA) packages have been developed specifically to aid assessment, for example, Question Mark (2004), TRIADS (2004), and the online test module of WebCT (2004). CAA is a common term for the use of computers in the assessment of student learning (Bull and McKenna, 2004). It encompasses the use of computers 1) to deliver, mark and analyse students' work; 2) to record, analyse and report on achievement; and 3) to collate, analyse and transfer assessment information through networks. CAA software can be helpful to assessment in a wide range of areas (Brown *et al.*, 1997; Freeman and Lewis, 1998; Bull and McKenna, 2004). For example, more frequent assessment is made possible and assessment can be available "on demand"; students can receive feedback immediately after assessment; teachers' marking load can be reduced significantly by automated marking; it offers the opportunity to widen the range of assessment methods, like the inclusion of multimedia components in questions and the use of the computer adaptive test in which questions are selected according to the

candidate's response to the previous questions; it makes the administration and management of assessment data more efficient and less prone to human error.

1.1.3 Limitations of Software Support for Formative Assessment of Students' Open-Ended Written Work

Although the use of computers in assessment is growing rapidly, most available CAA applications are only capable of dealing with the assessment of objective tests (Bull and McKenna, 2004; Brown *et al.*, 1997). In objective test students are required to choose or provide a response to a question for which the correct answer is predetermined. Multiple-choice, true-false, text matching, graphical hotspot (which presents an image and students have to click the spot representing the correct answer), and fill-in-the-blank are typical examples of objective test questions. CAA applications can easily assess students' answers to objective test questions by comparing them with the stored predetermined correct answers and in turn automatically create feedback either for summative purpose (for example, a numeric mark identifying whether the answer is correct or not) or formative purpose (for example, a brief comment explaining why the provided answer is incorrect). The situation with the assessment of students' open-ended written work (like creative essays, answers to questions that lend themselves to a wide range of interpretations, and solutions to problems that offer several potential pathways) is different. None of the existing software applications provides a satisfactory solution to fully support the assessment, especially formative assessment, of students' open-ended written work. A few CAA packages, like e-rater (2004) used in the assessment of essay questions of the Graduate Management Administrations Test (GMAT, 2004), can automate the marking of essays. However, these automated-essay-marking applications are mainly used for the summative assessment purpose and their contribution to formative assessment is very limited. These applications are only able to provide numeric marks and some brief comments selected from a predefined comment bank according to the marks awarded. They cannot produce detailed formative feedback to reflect each individual student's particular approach taken in the essay. Yet the provision of such detailed and relevant feedback is one of the most important features of formative assessment. For the formative assessment of open-ended written work, computers are mainly used to help human markers to assess students' work manually. The

commenting functions provided by most word-processing tools enable teachers to add detailed textual comments to students' written work as formative feedback more efficiently than the traditional paper-based way. However, the disadvantages of using these tools in formative assessment are also prominent. For example, it is hard to associate a comment with the corresponding assessment criterion using the available word-processing tools; teachers have to use additional tools to receive and manage student submissions; and students need to complete and submit their work in the document types that can be opened by the tools used by their teachers. Several CAA applications provide some good features in supporting formative assessment of open-ended written work in addition to enabling teachers to add free text comments to students' written work, like the MarkTool application described in Heinrich and Lawn (2004) and the Markin (2004) application (these applications will be examined in detail in Chapter 2). However, these applications only offer fragmented and insufficient support for formative assessment of students' open-ended written work. For example, the support for assessment-related communication is ignored and no specific facilities are provided to help teachers to control the quality of feedback.

1.2 Objective of the Research

The objective of this research project is to develop an e-learning solution to facilitate formative assessment of electronically submitted open-ended written assignments. It aims to:

- Assist in transferring the educational theories around formative assessment established in the literature into practice;
- Provide tools that increase the efficiency of dealing with formative assessment of open-ended written assignments and contribute to a higher uptake of formative assessment;
- Provide approaches and tools that are universally applicable and easy to use.

The following guidelines were established in pursuit of the research objective:

- Throughout the project the educational theories on formative assessment need to be consulted to ensure that best practices are observed.
- The project needs to provide a generic solution for facilitating formative

assessment of electronically submitted open-ended assignments from all disciplines.

- The provided solution needs to cover all the formative assessment related issues, not only the marking activity (i.e. the activity to create feedback on assignments, for example, to add comments or assign marks), but also all the other activities involved (such as assignment submission, management of the submitted assignments, delivery of marked assignments, self-assessment and peer-assessment, quality and consistency control and feedback, and assessment result analysis).
- The project needs to develop an approach to help human markers to mark the electronically submitted written assignments in an intuitive and efficient way, i.e. using their well-established paper-based assessment skills, so that little extra training is required and students are provided with feedback in familiar formats.
- The provided solution needs to offer a high degree of usability, effectiveness, efficiency, and security.

The term "open-ended written assignment" refers to any types of student assignment that can be submitted electronically and are in printable formats, i.e. the assignment document only consists of printable symbols (letters, digits, graphics, and static images). It can be an essay written using Microsoft Word or any other word processing applications, a brochure created using Microsoft Publisher, or a UML diagram drawn by a computer science student with IBM Rational Rose or Microsoft Visio. Assignments containing hypertext and audio/video components are beyond the scope of this research project.

Although the use of some advanced statistic algorithms makes it possible to automate the summative assessment of essay-type work, into the foreseeable future it will be impossible to automate formative assessment of open-ended written assignments (Heinrich and Lawn, 2004). This means that human markers are still needed to understand students' patterns of thinking as reflected in the submitted assignments and construct formative feedback according to each individual student's particular solution towards the assignment problems. Therefore, this project takes the approach

of facilitating human markers to perform formative assessment and does not intend to automate formative assessment of open-ended written assignments in any way.

1.3 Research Steps

To fulfill the project aims identified in the previous section, the following research work has to be undertaken:

1. Literature review – the literature of assessment theories, especially formative assessment theories, needs to be reviewed thoroughly in the early stage of the project, as these theories are the fundamentals underlying this project. Available software applications that can be used to assist assessment, especially the applications that provide useful features in facilitating formative assessment of open-ended written assignments, also need to be reviewed at this stage. This investigation is important as to establish which facilities already exist and which can be built on by the project.
2. Conceptualization of solution – the second step of the project is to develop the concepts of facilitating formative assessment of open-ended written assignments with current e-learning technologies. The related work involves:
    2.1. Concept development – analysing the findings of the literature review; identifying the gaps between the facilities provided by current software applications and the requirement of formative assessment theories; developing concepts of how to fill these gaps;
    2.2. Specifying a software solution – specifying a software system according to the requirements developed in Step 2.1, integrating the useful features of already available applications that facilitate formative assessment, examined in Step 1, into the system conceptualization.
3. Software development – the third step of the project is to develop the software system specified in Step 2, including: 1) firstly, designing the system; and 2) secondly, implementing the designed system using suitable programming technologies.
4. Evaluation – the last step of the project is to evaluate the implemented system in real teaching contexts to verify its effectiveness in facilitating formative assessment of open-ended written assignments.

1.4 Structure of the Thesis

This thesis presents the results of the research project. This first chapter introduced the context of the research as well as the motivation and objectives of this project. The remainder of this thesis is structured following closely the research steps described above:

- Chapter 2 presents the review of assessment theories, especially formative assessment theories, and the examination of available software applications assisting assessment.
- Chapter 3 consists of three parts – first it introduces the principles of using computers to facilitate formative assessment of open-ended written assignments; then it describes the requirements developed in this project for using computers to facilitate formative assessment of open-ended written assignments; the last part of Chapter 3 presents the specification of the WAA (Written Assignment Assessment) system which is a software system designed and implemented in this project to facilitate formative assessment of open-ended written assignments.
- Chapter 4 discusses technical issues relevant to the design and implementation of the WAA system.
- Chapter 5 presents an informal evaluation of the WAA system.
- Chapter 6 concludes this thesis by reviewing the contributions of this project from both conceptual and technical perspectives. Future work and research directions are also suggested in Chapter 6.

Chapter 2 Literature Review

This literature review chapter consists of two parts. The first part is an overview of current educational assessment theories, especially the theories on the relationship between formative assessment and student learning. The educational theories reviewed in this chapter provide the foundations for this research project and will be referenced frequently in the following chapters of the thesis. The second part presents an examination of available software applications that can be used to support educational assessment, especially the applications that are helpful in facilitating formative assessment of written assignments.

2.1 Educational Assessment Theories

Looking up the term "assessment" in *Chambers 21st Century Dictionary* reveals that "assessment" is related to the following:

- To estimate, judge the importance, quality, value, extent of something;
- To fix the amount of something (tax or fine).

In the education field, only the first of these two meanings makes sense – to judge the extent of students' learning. Rowntree (1987:4) gives an informal definition of assessment in education as the process of gathering and interpreting information about students' "knowledge and understanding, or abilities and attitudes".

Educational assessment has existed for centuries and it is now widely recognized that assessment plays an indispensable and crucial role in all education activities (Brown and Knight 1994; Earl, 2003; Freeman and Lewis, 1998; Murphy and Torrance, 1988; Rowntree, 1987; Weeden *et al.*, 2002). The related educational assessment theories on which this project is based are reviewed in the following part of this section.

2.1.1 The Primary Purpose of Assessment

Assessment can serve a wide range of educational purposes, for example, to choose students for higher level study by the means of entrance examinations, to certify students' achievements in the form of formal qualifications, to help students' learning

by providing feedback, or to judge educational quality by the means of national tests. Black (1997) groups the purposes of assessment under three broad categories:

- To improve learning and teaching;
- To report achievement of individual student for selection, and certification;
- To evaluate how well teachers or institutions are performing.

In the early ages when education resources were scarce, assessment was mainly used for the purpose of selection. Now it is generally accepted that the primary purpose of assessment in education is to improve students' learning (Black 1993; Brown and Knight 1994; Forum, 1995; Freeman and Lewis, 1998; Murphy and Torrance, 1988; Rowntree, 1987; Stobart and Gipps, 1997; Torrance and Pryor, 1998; Weeden *et al.*, 2002; Wiggins, 1998). A number of research studies reveal that this primary goal of assessment can be achieved successfully by the proper use of formative assessment (Black and Wiliam, 1998a, b; Brookhart, 2001; Crooks, 1988; Natriello, 1987).

2.1.2 Concepts of Formative Assessment

As introduced briefly in Section 1.1.1, assessment can be defined as summative assessment or formative assessment in reference to its purpose. The former aims to provide a final judgment of student performances and is termed as assessment ***of*** learning, while the purpose of latter is to provide students with feedback which they can interpret and act on to improve their further learning and is called assessment ***for*** learning.

Some key concepts of formative assessment are reviewed in this section.

❖ **Feedback in Formative Assessment**

In order to help students to promote their learning, the feedback provided in formative assessment needs to consist of three elements (Black and Wiliam, 1998b):

- The identification of the desired learning goals – what students are expected to achieve.
- Evidences showing the students' current positions in respect to the learning goals – what students have achieved and especially what they have not.

- Some instructions for students to close the gap between the current position and the learning goal – what they need to do to overcome the difficulties occurred in their studies.

❖ Requiring Further Actions from Students and Teachers

Researchers argue that assessment is only summative and not formative if students do not read, understand, and act on the received feedback even though the feedback contains all the three elements mentioned above (Brown and Knight, 1994; Freeman and Lewis, 1998; Higgins *et al.*, 2002; Rowntree 1987). Brown and Knight (1994:17) emphasise this by saying "If that piece of coursework is marked, returned and just stored away, then its effective use has been summative, whatever the intension had been".

The concept of formative assessment also requires teachers to evaluate the effectiveness of their teaching by analysing the assessment data and adjusting their future teaching according to the results of assessment. Weeden *et al*. (2002) state that assessment cannot become formative if the collected assessment information is not used by teachers to improve their teaching.

❖ Self-Assessment and Peer-Assessment

There are two special forms of formative assessment – self-assessment and peer-assessment in which the persons who are in charge of providing feedback are not teachers or tutors but student themselves. During self-assessment and peer-assessment, students will give feedback on their own or their fellow students' performances based on their understanding of the learning goals (Brown and Knight, 1994; Earl, 2003; Freeman and Lewis, 1998; Rowntree 1987). The feedback produced in self-assessment and peer-assessment needs to provide all the three elements required for formative assessment – the learning goal, the present position, and the instructions to close the gap.

❖ Overlap of Formative and Summative in Practice

In educational practice, assessment is often both of formative and summative. Freeman and Lewis (1998) demonstrated this by using the example of the assessment of university on-course assignments. In the assessment of such assignments, on the

one hand, teachers or markers need to assess the assignments by awarding some numeric marks since these marks count towards the students' final credits of the course they are taking; on the other hand, teachers or markers also need to provide students with formative assessment in the format of detailed textual comments to tell students what they have missed and how they can improve so that students can learn from the feedback and take actions to improve their learning before the final examination.

2.1.3 Formative Assessment and Student Learning

The Assessment Reform Group (ARG, 1989) is an organization of assessment experts. In 1997 ARG commissioned Professors Paul Black and Dylan Wiliam to review the research literature on formative assessment. The evidence of this extensive literature review (Black and Wiliam, 1998a, b), during which more than 70 journal titles for relevant publications were scanned and about 250 publications were analysed, revealed that formative assessment, implemented properly, is a powerful tool to improve student learning. Based on this review, ARG (1999) identified a number of key factors that help to improve students' learning through formative assessment:

- Formative assessment requires the provision of effective feedback which:
  - Helps students to understand the standards (or criteria) they are aiming for and stimulates learning by identifying the learning goal;
  - Aids students in recognizing their strengths and weaknesses by specifying what they have achieved and what they have not;
  - Tells students what they need to improve and how to improve.
- Formative assessment involves self-assessment and peer-assessment.
- Formative assessment requires students to act on the feedback.
- Formative assessment requires teachers to adjust teaching their according to the assessment results.

Researchers argue that the role of formative assessment in promoting student learning can be reinforced by the application of self-assessment and peer-assessment (Black and Wiliam 1998a, b; Brookhart 2001; Brown and Knight 1994; Eral, 2003; Freeman and Lewis, 1998; Rowntree, 1987; Weeden *et al.*, 2002). Some benefits of self-assessment and peer-assessment as claimed by Freeman and Lewis (1998) are:

- Students will be more motivated in their learning since self-assessment and peer-assessment give students a greater ownership of the learning they are taking;
- Self-assessment and peer-assessment can help students understand clearly what the learning goals and the assessment criteria mean;
- Self-assessment can help students to develop the ability of self-awareness which is important to their learning and future careers;
- Students can learn from their fellow students during peer-assessment. Collaboration and other interpersonal skills can be developed as well by carrying out peer-assessment.

These arguments are, for example, supported by the findings of a study by Fontana and Fernandes (1994). The study looked at the influence of self-assessment on student learning. It involved 25 mathematics teachers and 354 students, and the results showed that after a 20-week part-time course the average gain of the students who were trained to understand and use self-assessment was about twice that of the students who had no knowledge about self-assessment. The substantial review by Black and Wiliam (1998a, b) reports similar achievement gains for students who can apply self-assessment and peer-assessment properly in their learning.

2.1.4 Quality of Formative Assessment

Black and Wiliam (1998a, b) believe that only high-quality and properly implemented formative assessment can be productive in improving student learning. The same notion is expressed by Brown and Knight (1994), FairTest (1999), Freeman and Lewis (1998), Harlen (1999), Torrance and Pryor (1998), and Weeden, *et al.* (2002).

As discussed earlier in this chapter, formative assessment improves student learning mainly through the provision of feedback that students can interpret and act on. This means that the quality of feedback given to students is of the highest importance to the success of formative assessment. A number of strategies have been proposed by researchers on how to create high quality feedback on students' work (Brown and Knight, 1994; Boud, 1995; Freeman and Lewis, 1998; Rowntree, 1987; Weeden *et al.*, 2003). Freeman and Lewis (1998) list the following as the particular important characteristics of "good" feedback:

- Good feedback is positive – it needs to motivate and encourage students to move forward in their learning.
- Good feedback is prompt – it needs to be given to students as soon as possible, otherwise, students may feel it is useless since they have already entered a new learning unit.
- Good feedback is clear – it needs to be comprehensible to students so that students can recognize what they need to do to improve their learning.
- Good feedback is relevant – first, it needs to be linked with the assessment criteria so that students can quickly grasp their strengths and weaknesses; secondly, it needs to be relevant to the individual students respecting the particular approach they took in their work.
- Good feedback is informative – it needs to provide students with detailed information about their weaknesses and the possible reasons, as well as suggestions for improvement.
- Good feedback encourages self-assessment – for example, it can invite students to be involved in assessment by asking them to provide their own assessment on some specific parts on their work.
- Good feedback encourages dialogue – it needs to stimulate response from the students and continuing dialogue between teachers and students, termed as "post-assessment discussion" by Murphy and Torrence (1988). This kind of dialogue is intended to serve two purposes. First, it can help students to understand and take actions on the feedback. Secondly, it can provide teachers the information they need to adjust their teaching.

Black and Wiliam (1998b) argue that if formative assessment is to be productive teachers need to help students to understand clearly the purpose of their learning, i.e. what they are supposed to achieve. As introduced earlier in Section 2.1.3, this can be achieved via self-assessment and peer-assessment. Researchers also claim that explicit and detailed assessment criteria, which are revealed to students, can help students understand clearly what they are expected to achieve and in turn improve their performances (Brown *et al.*, 1997; Brown and Knight, 1994; Freeman and Lewis, 1998). Typically, the assessment criteria for an assessment task need to include the overall requirements for the task and a set of criteria. Each criterion is associated with

a specific aspect of the task and describes what students are expected to achieve in regard to this aspect (Brown *et al.*, 1997; Brown and Knight, 1994; Freeman and Lewis, 1998). Each individual criterion should also be associated with a rating scale which is used to indicate the extent to which the criterion is met to serve the summative assessment goal since assessment is often both formative and summative (see Section 2.1.2). Figure 2-1 gives an example of a part of the assessment criteria for an English essay assignment in Athabasca University (Baig and Pierre, 2004).

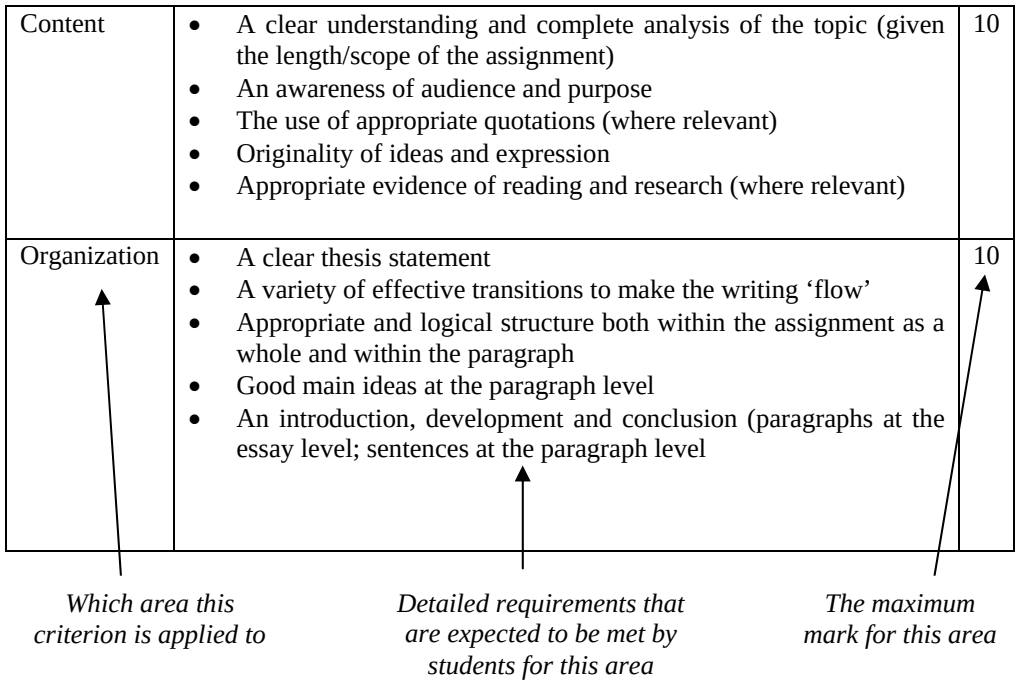


Figure 2-1 Part of the assessment criteria for an essay-type assignment

Unfortunately, studies show that teachers and students may have different perceptions of what the assessment criteria mean even when these criteria are made explicit (Freeman and Lewis, 1998). This could lead to the misinterpretation of the feedback provided by teachers. Therefore, students need help in understanding and acting on assessment criteria. There are a number of ways by which this help can be provided (Brown and Knight, 1994; Erwin, 1997; Freeman and Lewis 1998; James, 1998; Weeden *et al.*, 2002):

- The involvement of students in setting up the assessment criteria;
- The use of assessed samples;
- Self-assessment and peer-assessment;

- The use of a student feedback sheet. Each individual student should be provided with a feedback sheet. This feedback sheet consists of the assessment criteria and all the feedback for the student.

The other two important issues associated with the quality of formative assessment are the validity and reliability of assessment (Brown and Knight, 1994; Freeman and Lewis 1998; Harlen and James, 1997; Stobart and Gipps, 1997). The former is defined as the extent to which an assessment measures what it is designed to measure. The latter emphasizes the consistency of assessment results (James, 1998; Stobart and Gipps, 1997). This thesis focuses only on reviewing the literature of the reliability of formative assessment. Whether an assessment is "valid" to assess students' abilities mostly depends on how the assessment is designed (for example, whether the questions of an examination are too easy or difficult to reflect students' real knowledge levels), and therefore is beyond the scope of this project. Brown *et al.* (1997) argue that there are two measures of the consistency in assessment. One is the measure of the agreement among markers (if more than one marker is involved in the assessment), and the other is the measure of the agreement within one marker (the work of different students is assessed by one marker). A number of strategies are proposed by researchers to improve the reliability of formative assessment (Brown and Knight, 1994; Freeman and Lewis, 1998; Stobart and Gipps, 1997):

- To create and use clear assessment criteria;
- To increase the number of the marked samples, which are shared among markers;
- To conduct discussions among markers;
- To double-assess students' work by different markers;
- To review a sample of each marker's work.

As reviewed in Section 2.1.2, self-assessment and peer-assessment are two special formats of formative assessment. Therefore to maintain the quality of self-assessment and peer-assessment, all the formative assessment principles discussed earlier in this section (for example, the need for high-quality feedback and the use of explicit assessment criteria) should be followed as well. Furthermore teachers need to help students to develop the skills required for carrying out self-assessment and peer-

assessment properly. For example, teachers can start by working through assessment samples with students, and then ask students to assess their own or their peer students' work. To guarantee the quality of self-assessment and peer-assessment, teachers also need to provide students with feedback on the feedback created by students in self-assessment and peer-assessment (Brown *et al*,. 1997; Brown and Knight, 1994; Freeman and Lewis, 1998; Weeden *et al*., 2002).

2.1.5 Current Formative Assessment Practice

Unfortunately, although it is widely accepted that formative assessment is highly beneficial to student learning, research shows that high-quality formative assessment is relatively rare in education (Black and Wiliam, 1998a, b; Weeden *et al*., 2002; Yorke, 2003). This poverty of practice is mainly due to the conflict between the workload of conducting high-quality formative assessment and the increasing time pressures on teachers (Ashcroft and Palacio, 1996; Yorke, 2003). To provide each individual student with detailed, relevant formative feedback takes much more time than simply marking students' work summatively by assigning numeric marks or letter grades. Furthermore, to ensure the success of formative assessment, efforts have to be made to help students understand the received feedback and the assessment criteria against which their work was assessed. In the case of self-assessment and peer-assessment, teachers have to spend a certain amount of time to help students to develop the required skills and monitor the process of self-assessment and peer-assessment. All these are time-consuming activities and conflict with the pressures on teachers (such as the increasing student-teacher ratios and the research demands for teachers in higher education), and therefore hinder the wide-spread use of formative assessment. Although teachers' marking workloads can be reduced by having the assignments or examinations marked by other markers (for example, senior students), the time-saving will be counteracted in many cases by the extra work required for the quality and consistency control of the feedback.

2.1.6 Summary

Various formative assessment theories have been reviewed in the previous sections of this chapter and can be summarized as:

- Formative assessment is primarily concerned with promoting student learning. It focuses on providing students with feedback that they can act on to improve their future learning.
- Student learning can be improved significantly by using formative assessment (including self-assessment and peer-assessment) in a proper way.
- Formative assessment can be productive in promoting learning only when its quality is maintained in the following ways:
  - Providing students with high quality feedback – the feedback given to students needs to be detailed, clear, relevant, positive, and timely;
  - Maintaining consistency among different markers and within one marker;
  - Involving self-assessment and peer-assessment;
  - Using explicit assessment criteria that are revealed to students and associating feedback with these criteria;
  - Encouraging assessment related dialogue among teachers, markers, and students – these dialogues play an important role in helping to understand assessment criteria and feedback, maintaining consistency among different markers, and providing help for self-assessment and peer-assessment.

Although a number of strategies have been developed to ensure the quality of formative assessment, high-quality formative assessment is still not used widely enough in education. This is mainly because of the conflict between the workload of conducting high-quality formative assessment and the pressures on teachers.

2.2 Use of Computers in Assessment

The use of computers in assessment is now expanding rapidly – more and more teachers and students are using computers to design and develop assessment tasks, create feedback, record and deliver assessment results, submit their work electronically, and view assessment results. By using computers in assessment, the assessment workload on teachers can be reduced significantly and students can be provided with feedback on their learning more efficiently than is usually possible with traditional paper-based assessment (Brown *et al.*, 1999). This section will review the currently available computer software technologies that can be used to assist

assessment. This review will focus on the technologies for facilitating formative assessment of open-ended written assignments as the purpose of this research project is to develop a new way for using computers to facilitate formative assessment of such assignments as introduced in Section 1.1.

There are a number of ways in which computers can be used to assist assessment by taking advantage of the current software technologies' powerful capabilities in data processing, storage and transfer (Brown *et al*., 1997; Bull and McKenna, 2004; Freeman and Lewis, 1998). For example computers can be used:

- By teachers to prepare assessment materials;
- By students to create assessment answers;
- By teachers to provide feedback;
- By students to submit work and by teachers to deliver assessment tasks and feedback;
- By teachers to manage assessment and assessment results;
- By teachers and students to discuss assessment related issues with others;
- By students to carry out self-assessment and peer-assessment.

How the above assessment functions can be assisted by computers will be reviewed in the rest of this section.

2.2.1 Using Computers to Prepare Assessment Materials

Most teachers are familiar with using word processing tools, for example, Microsoft Word, to create, store, and modify various assessment materials, such as the requirements and marking scheme for an assignment or lab work, example solutions, and test questions. By providing various editing, formatting, and drawing functionalities, these word processing tools enable teachers to create assessment materials which may include texts, graphics, tables, figures, or even multimedia components in a more effective way than the traditional pen-and-paper-based means.

In addition to using generic word processing applications to create assessment materials, a number of CAA packages, for example, Question Mark (2004), TRIADS

(2004), and EQL Interactive Assessor (Brown *et al*. 1997), are available to help teachers to conveniently create objective test questions. Most Learning Management Systems (LMS) also provide their own CAA modules which enable teachers to efficiently create online objective tests, for example, like the Quizzes/Surveys and Self Test tools of WebCT (2004). These CAA applications provide teachers with user-friendly graphical interfaces via which they can easily input the questions of various objective tests. All the created questions are stored permanently by the CAA applications, for example, using database technology, and the stored questions can easily be modified by teachers and presented to students in the tests.

2.2.2 Using Computers to Create Assessment Answers

Many software applications can be used by students to create their assessment answers more efficiently than the traditional pen-and-paper-based means – for example, Microsoft Word and other word processing applications are widely used by students to write their essay assignments, Microsoft PowerPoint is the most frequently used tool by students to create slides when they prepare for presentations, and the use of CAA software by students is rapidly increasing as computer-based tests are adopted by more and more education institutions.

2.2.3 Using Computers to Provide Feedback

Like the preparation of assessment and creation of assessment answers, the provision of feedback on students' work can also be assisted by a variety of software applications. This assistance is achieved mainly through two different ways: the first is to automate the provision of feedback; and the second is onscreen marking of students' written work – presenting students' work electronically to teachers via computer monitors so that teachers can add feedback by using various computer input devices, like keyboard and mouse.

❖ **Automating the Provision of Feedback**

Most CAA packages, like the applications mentioned earlier in this section, are capable of automatically providing students with feedback on their answers to objective test questions. The CAA packages compare students' input with the

predetermined correct answers stored in a database and then automatically create numeric marks based on the comparison results. A few CAA packages are able to automate the provision of feedback (including a numeric score and a short text comment which is selected from a predefined comments bank) on students' free-text essays work (Bull and McKenna, 2004). The major methodological approaches used by these essay-automated-scoring CAA packages are (Callear *et al*., 2001; Palmer *et al*., 2002; Valenti *et al*., 2003, Williams, 2001):

- Project Essay Grade (PEG) developed by Page (1966) which primarily relies on the linguistic surface attributes of essays, such as the rate of occurrences of uncommon words, the use of prepositions and punctuations, and the count of words;
- Latent Semantic Analysis (LSA) which uses statistical modelling to compare semantic similarity between texts;
- Hybrids of statistical approaches and Natural Language Processing (NLP) techniques, like Electronic Essay Rater (e-rater) which uses a combination of NLP techniques to parse the syntactic structure of essays and a series of statistical approaches to analyse the rhetorical and topical structure of essays (Burstein *et al*., 1998).

The idea of automating essay assessment has been around for over thirty years, and a number of systems have been developed based on the approaches introduced above. However, none of the systems satisfactorily assesses students' essay work (Callear *et al*., 2001). PEG does not take into account the semantics of essays and only measures the quality of writing styles (Callear *et al*., 2001; Chung and O'Neill, 1997; Kakkonen and Sutinen, 2004). LSA ignores grammar and spelling and the order of words in a sentence, for example, LSA cannot distinguish the sentences "A is after B" and "B is after A" from each other (Callear *et al*., 2001; Chung and O'Neill, 1997). None of the systems is able to deal with tabular and graphical content which are used very often by students in their essays (Williams, 2001). Another serious drawback of these systems is their inability to create detailed, relevant feedback on essays (Bull and McKenna, 2004), although some of them, for example, e-rater, are able to provide several predefined textural comments as the summary of an essay (see Figure 2-2 for the score and comment generated by e-rater). Furthermore, all the systems need to be trained by a large amount of sample essays that have been correctly graded by human

markers. These samples need to cover all the knowledge and writing styles that could be contained in students' essay work which is difficult and in some cases, for example, for research essays, impossible to achieve (Palmer *et al*., 2002; Valenti et al., 2003; Williams, 2001).

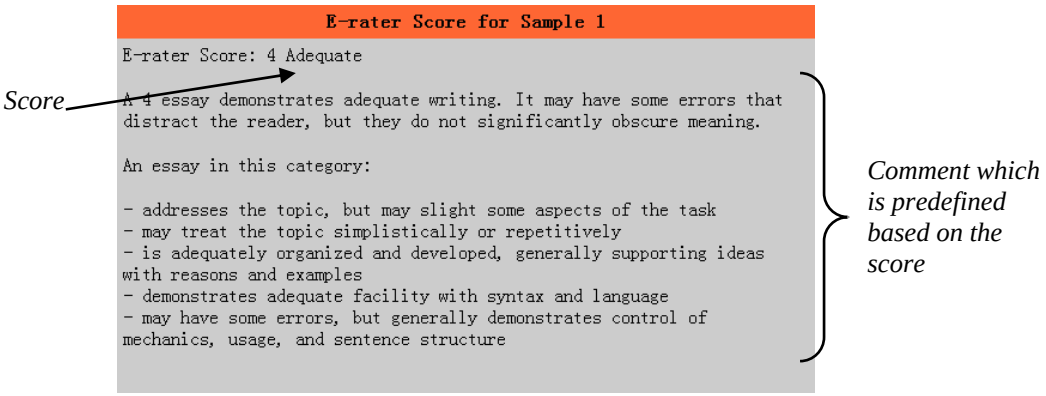


Figure 2-2 Score and comment automatically generated by e-rater

❖ **Onscreen Marking**

Due to the limitations of the current automated-essay-scoring technology, computers are mainly used to assist providing feedback for students' open-ended written work through onscreen marking (Atkinson and Davies, 2005; Heinrich and Lawn, 2004; Stephens *et al*., 2001). In onscreen marking, students' work is presented electronically to human markers via the computer monitor using software applications. The human markers need to create the feedback (for example by adding free-text comments) based on their understanding of students' work.

Most word processing tools provide users with a "Comment" functionality which can be used by teachers to easily add free-text comments to students' written work. The "Track change" functionality of Microsoft Word and similar functionalities provided by other word processing applications enable teachers to correct students' errors with special display effects so that the original content and the corrections can be distinguished easily.

There are several specialist onscreen marking applications that offer additional facilities to help teachers to assess students' open-ended written work. Three examples of such onscreen marking applications are MarkTool (Heinrich and Lawn, 2004), the Markin (2004) application, and the marking tool in the Classmate system (Baillie-de Byl, 2004).

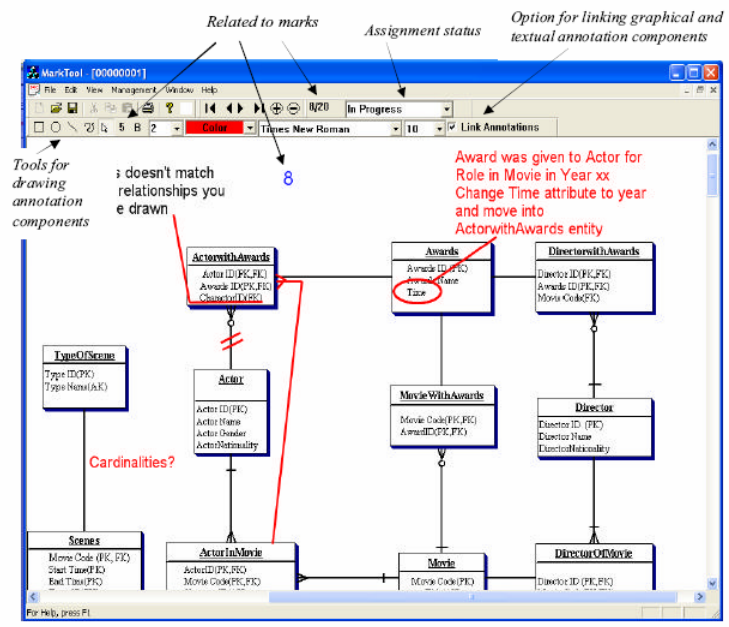


Figure 2-3 Screenshot of the MarkTool application with explanations (Heinrich and Lawn, 2004, p.1988)

MarkTool is an onscreen marking desktop application. It uses the Portable Document Format (PDF, 2003) to display documents. It allows human markers to add annotations to any parts of the displayed PDF pages leaving the original formatting of the PDF document intact. Each annotation consists of two components – one is a graphical component to indicate to which exact area of the displayed PDF page the annotation refers and the other is a textual comment that provides the main message of the annotation. These two components are linked together by a line, as illustrated by Figure 2-3 (Heinrich and Lawn, 2004, p.1988). Markers can choose different colors, shapes, or fonts for the graphical or textual components for categorizations of annotations. MarkTool saves the created annotations separately from the PDF documents in a Microsoft Access database.

The Markin application is another desktop onscreen marking tool. It is capable of presenting plain text and Rich Text Format (RTF) files. Users can insert some simple, predefined annotations into the text as superscripts by clicking the appropriate buttons (see Figure 2-4). For example, a superscripted "Spl" indicates a spelling error. Users can add/delete buttons and customize the definition of a specific button (see Figure 2-5). Users can also award a numeric mark to the displayed document and insert free-text comments at points within the displayed document as superscripted numbers which are linked to the actual comments (see Figure 2-4). For each plain text or RTF document, the Markin application creates an XML file and stores the annotations, comments, and definitions of buttons in this XML file (see Figure 2-6).

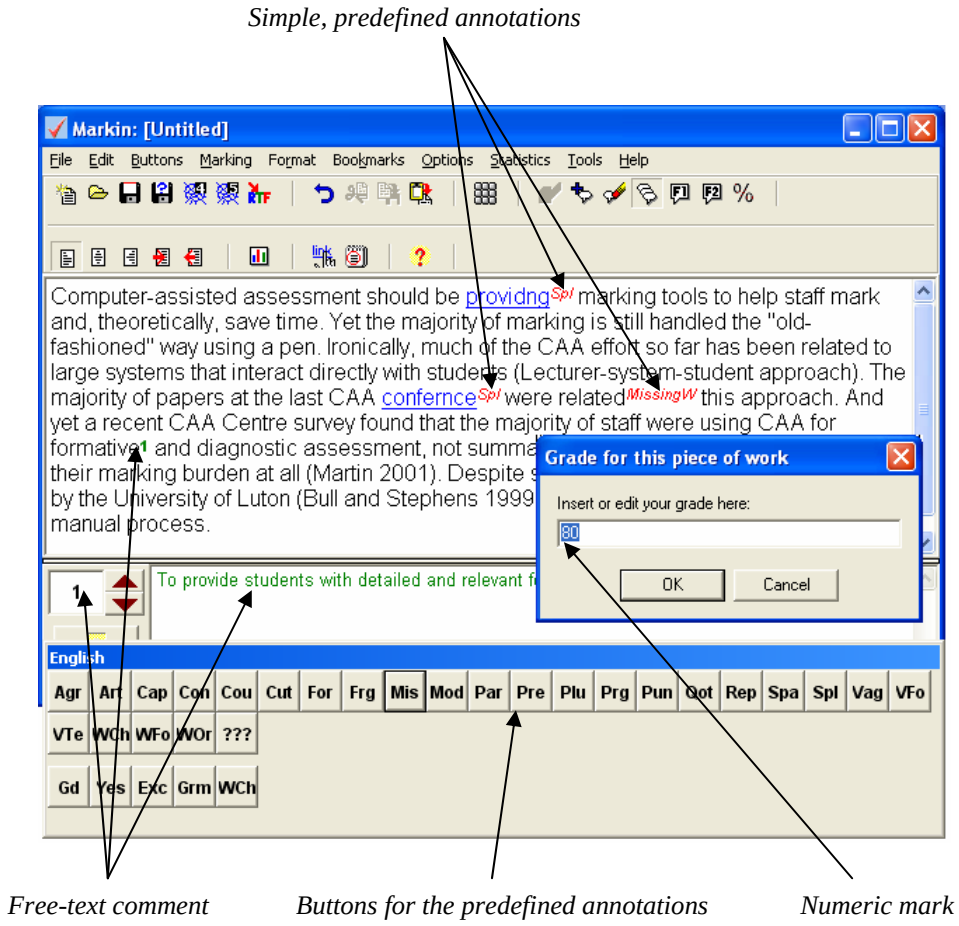


Figure 2-4 The Markin application

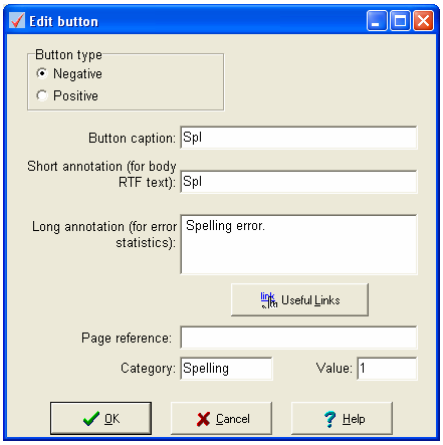


Figure 2-5 Customizing the definition of the "Spl" button in Markin

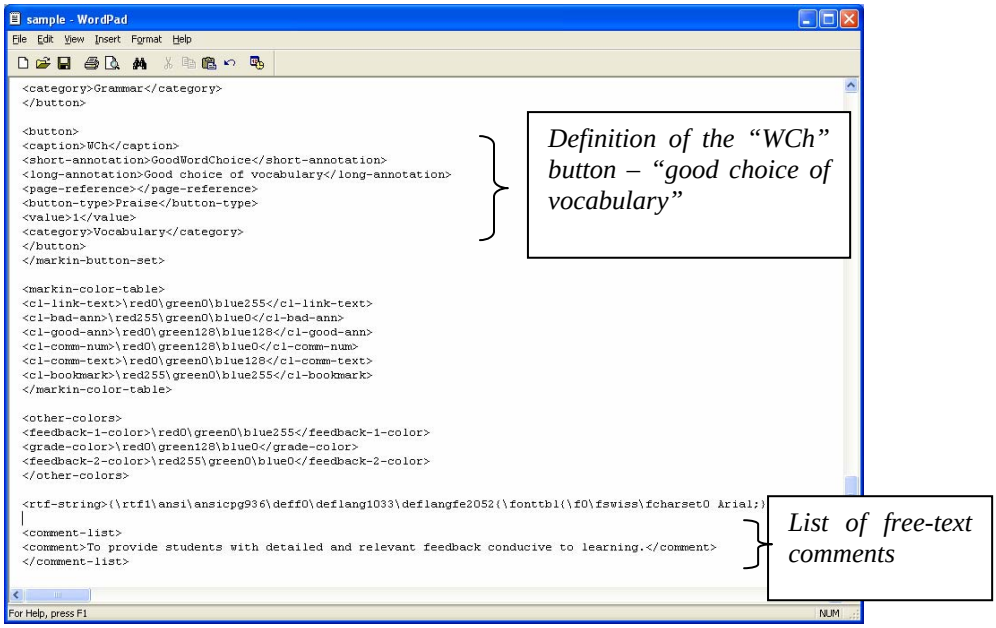


Figure 2-6 An XML document created by Markin

The Classmate system is an online assignment marking system used in University of Southern Queensland, Australia. It provides teachers with a marking tool which is, like the Markin application, able to display plain text and RTF documents and allows users to insert free-text comments into the displayed document. All the comments entered by teachers appear in red so that they can be easily distinguished from the original content of students' work as shown in Figure 2-7 (Baillie-de Byl, 2004, p.34). The tool also creates a marking sheet for each individual student where teachers can write an overall comment on the student's work and assign numeric marks according to some predefined marking criteria as shown in Figure 2-8 (Baillie-de Byl, 2004, p.35).

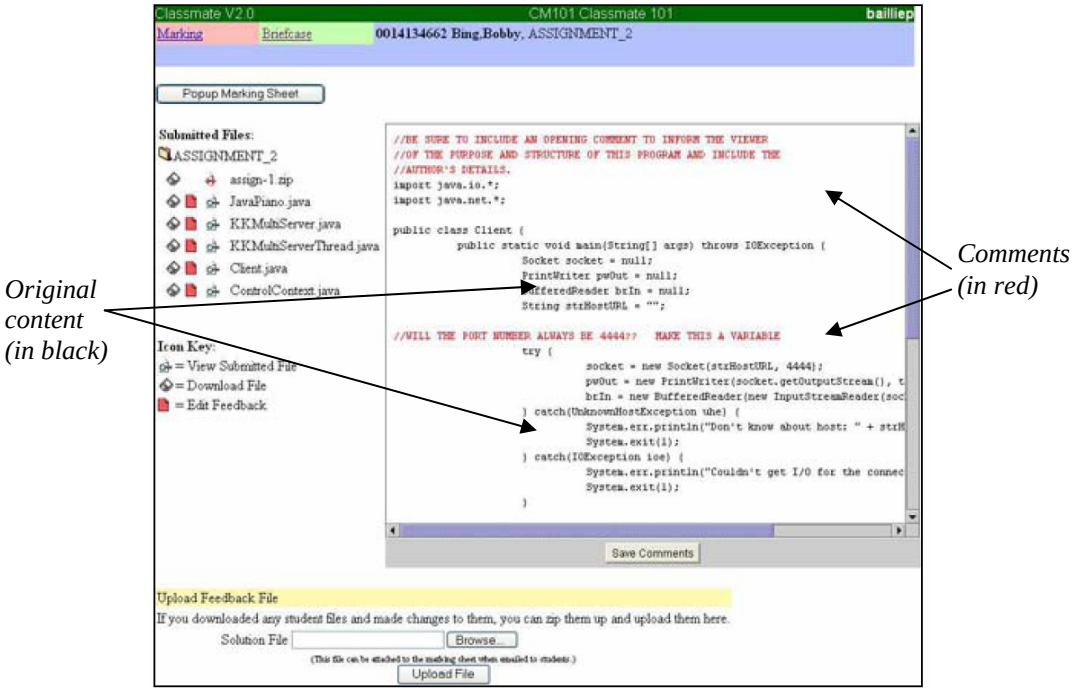


Figure 2-7 Free-text comment in the Classmate system (Baillie-de Byl, 2004, p.34)

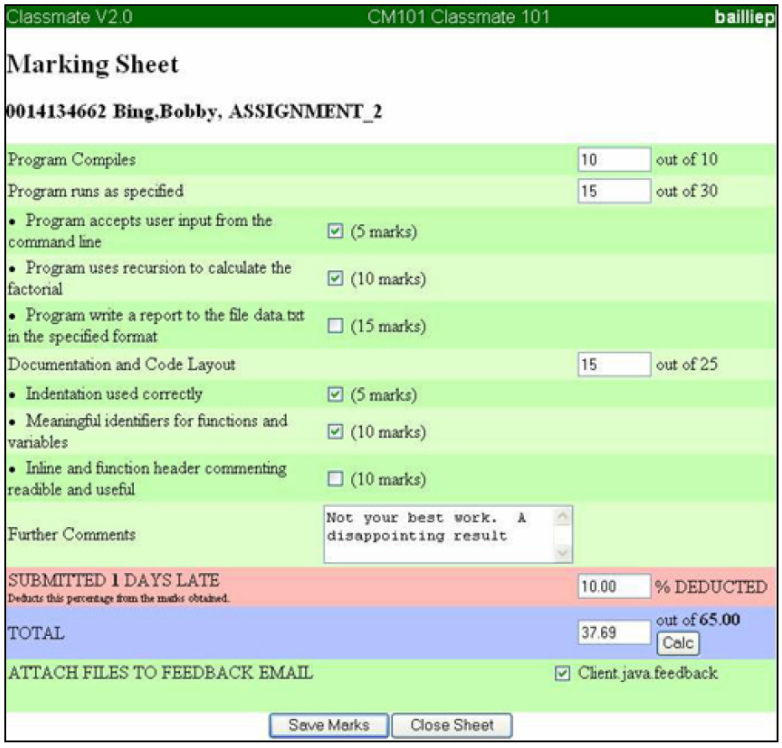


Figure 2-8 Marking sheet in the Classmate system (Baillie-de Byl, 2004, p.35)

2.2.4 Using Computers to Submit Student Work and Deliver Assessment Tasks and Results

A very straightforward use of computers in assessment is for the submission of student work and the delivery of assessment tasks and results. Two different methods are used by students and teachers. The first way is to attach documents to emails. However, using personal email for this purpose has the shortcoming that teachers need to manually organize and administer the email lists. The second way is to use the submission/delivery functionalities integrated in web-based CAA software and LMS, and in fact this is the major approach used in universities to deliver assessment tasks, submit/receive student work, and return assessment results. Web-based objective test CAA packages, like Question Mark (2004), can deliver and present objective test questions to students in the format of webpages, collect student answers from the webpage, and send back the results immediately to students through another webpage. The assignment module of WebCT enables teachers to set up a new open-ended assignment with some specific parameters, like the maximum grade and the submission due date. Teachers also can upload documents to WebCT for the assignment requirements or marking scheme. These assignment requirement or marking scheme documents can be downloaded by students from WebCT. This assignment module also provides students with an upload tool that they can use to submit their assignments electronically. Teachers can download the submitted assignments from WebCT, mark these assignments on their local computers, and upload the marked assignments and the final marks to WebCT (see Figure 2-9). The uploaded marked assignments and marks can be accessed by students through WebCT. Other LMS systems (for example Blackboard, 2004 and Moodle, 2004) and some online assignment marking systems (like the Classmate system introduced in the previous section) also offer similar submission/delivery functionalities as provided by the assignment module of WebCT.

2.2.5 Using Computers to Manage Assessment and Assessment Results

When using computers in various assessment-related tasks, the assessment data are created, stored and transferred in electronic format. This makes the administration and

management of these tasks and the assessment data more efficient, secure, and reliable than the traditional paper-based approach. For example, WebCT provides a number of assignment submission management functionalities, like for keeping record of the date when an individual student submitted the assignment and for notifying students by email immediately after their assignments have been received. More importantly, the assessment results can be quickly analysed in order to give teachers information on the performance of their students. For example, student marks can be analysed using a spreadsheet program (such as Microsoft Excel) by calculating the highest, lowest, and mean marks. WebCT and the Markin application introduced earlier in Section 2.2.3 provide such mark calculation facilities as well.

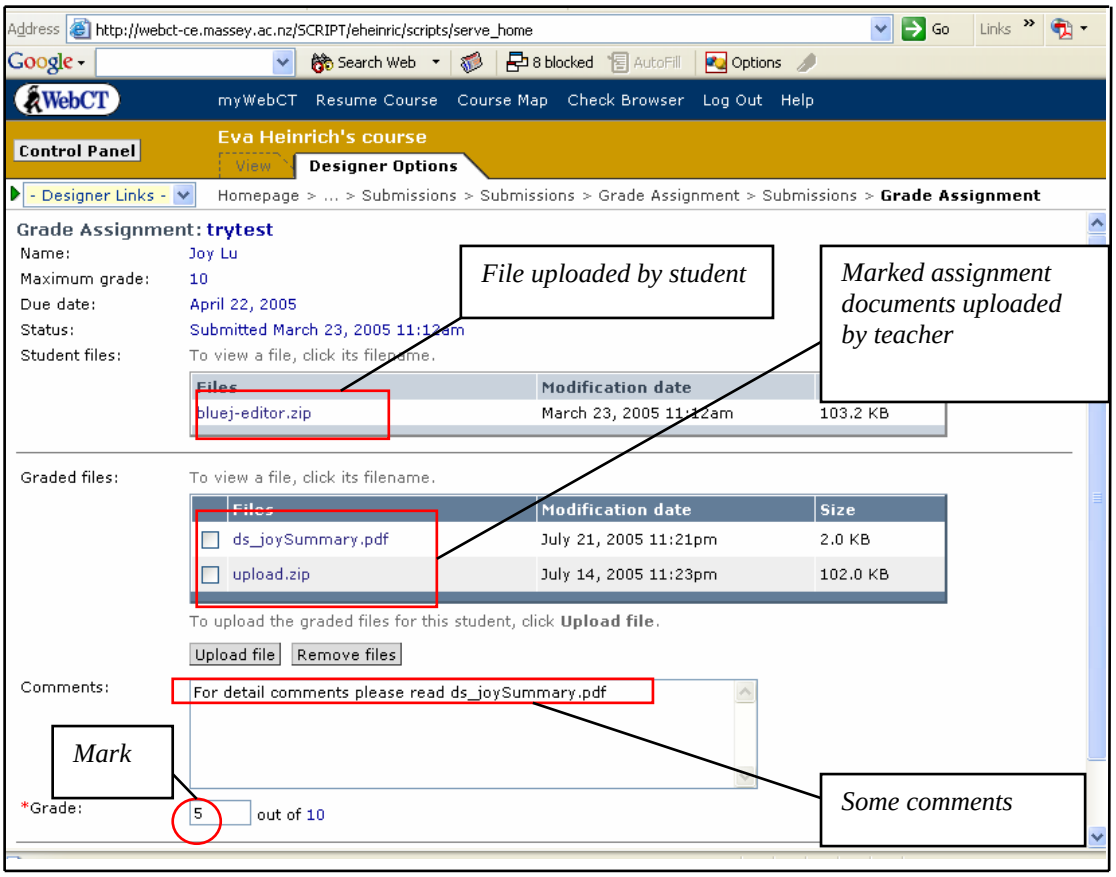


Figure 2-9 Part of a WebCT webpage

2.2.6 Using Computers in Discussions of Assessment-Related Issues

With the help of current web technology, one can easily communicate with others to discuss approaches to assessment or the assessment results using personal email or electronic discussion boards. Most LMS provide their own discussion board systems that offer students and teachers a good platform for discussions and for asking and answering questions. The internal email systems of LMS allow students and teachers to email others without knowing the recipient's personal email address (see Figure 2-10).

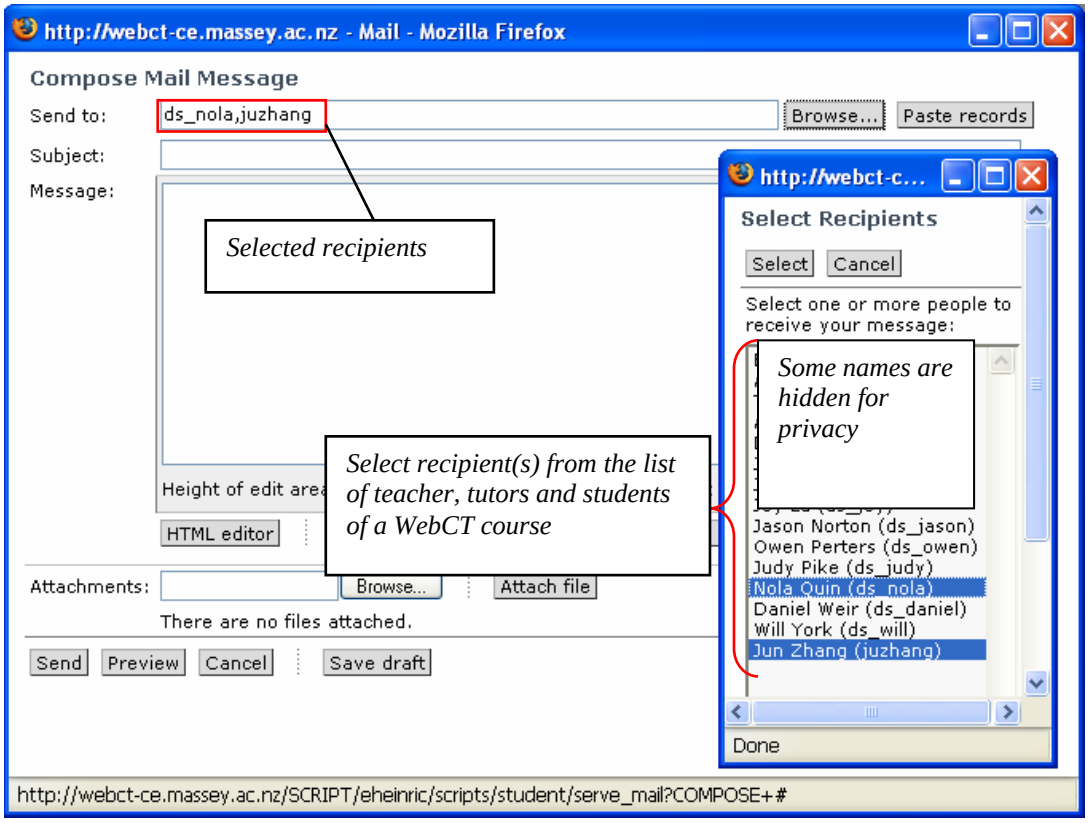


Figure 2-10 Compose email in WebCT

2.2.7 Using Computers for Self-Assessment and Peer-Assessment

The automated marking facilities of various objective test CAA packages make it easy for students to undertake self-assessment with this type of tests. The reports created

by these CAA packages allow teachers to monitor and analyse student performances and in turn provide students with feedback conducive to their learning (Bull and McKenna, 2004). A virtual community can be easily created using current network technologies among teachers and students. Students can share their work and carry out peer-assessment through this virtual community. Teachers also can monitor student peer-assessment activities and provide students with feedback on the feedback they gave to their peer students by taking advantage of this virtual community (Robinson, 1999).

2.2.8 Summary

As this literature review has shown, computers can be used to assist many different assessment functions. Yet there is a stark difference in the level of assistance for the assessment of objective test questions and open-ended student work. For objective test questions, a number of CAA systems, like Question Mark (2004), TRIADS (2004) and the online test module of WebCT (2004), have been developed that integrate many of these assessment functions and are widely used in education. Teachers can easily create various objective test questions via the graphic user interfaces provided by these CAA systems. The questions are delivered to students through networks so that students can input their answers using their web browsers, like Microsoft IE or Mozilla Firefox. Student answers are marked automatically by the CAA systems and the feedback is returned to students immediately. The assessment results are stored in a database and the systems are capable of producing detailed reports to help teachers analyse the performances of their students. The electronic discussion boards or internal email facilities of these CAA systems provide teachers and students the platform via which they can communicate with each other and discuss various assessment issues. However the assessment, especially formative assessment, of students' open-ended written work is not so well assisted by current CAA software. Current software technologies are not sophisticated enough to automate the provision of formative feedback on students' open-ended written work. Computers are mainly used to assist providing feedback for such work through onscreen marking. Although several onscreen marking applications provide some good features in supporting formative assessment of open-ended written work in addition to enabling teachers to add free text comments to students' written work, like MarkTool, Markin, and the

marking tool in Classmate. All these applications only offer fragmented and insufficient support for formative assessment of students' open-ended written work. For example, the support for assessment-related communication is ignored and no specific facilities are provided to help teachers to control the quality of feedback.

Chapter 3 Concept Development

As shown in Section 2.2, a number of researchers are active in developing specialized approaches and software systems for assisting formative assessment of students' open-ended written work. Yet no comprehensive solution exists for using computers to support formative assessment of such work. This project has developed a new approach for using computers to facilitate formative assessment of students' open-ended written assignments. This chapter discusses this approach in detail. It consists of three parts – first it lists the principles for using computers to facilitate formative assessment of open-ended written assignments; then it describes the results of requirement analysis undertaken in this project towards building a software system implementing these principles; the last part presents the specification of such a system – the Written Assignment Assessment (WAA) system.

For simplicity, in the remainder of this thesis, the term "assignment" will be used for "open-ended written assignment".

3.1 Principles for Using Computers to Facilitate Formative Assessment of Assignments

To implement the concept of using computers to facilitate formative assessment of assignments, the following principles should be followed:

1. It needs to facilitate all the activities that are potentially required for formative assessment of assignments.
2. It needs to provide an onscreen marking tool which enables human markers to mark assignments in an intuitive and efficient way by replicating their paper-based approaches.
3. It needs to provide a generic solution for facilitating formative assessment of assignments from all disciplines, not a limited solution restricted to some specific domains (for example, computer science or business courses).

The rest of this section discusses these three principles in detail.

3.1.1 Facilitating All Activities in Assignment Assessment Life Cycle

The first principle of the concept of using computers to facilitate formative assessment of assignments requires that computers should be used to support not only the marking activity, but also all the activities involved in the formative assessment of assignments – i.e. all the activities included in an assignment assessment life cycle. Assignment assessment life cycle is a new concept developed in this project based on the formative assessment theories reviewed in Chapter 2 and the "conversational framework" theory developed by Laurillard (1993) which claims that the dialogue between teachers and students is central to learning. An assignment assessment life cycle covers all the activities that are carried out either by teachers/markers or students and are necessary for formative assessment of assignments, as illustrated by Figure 3-1.

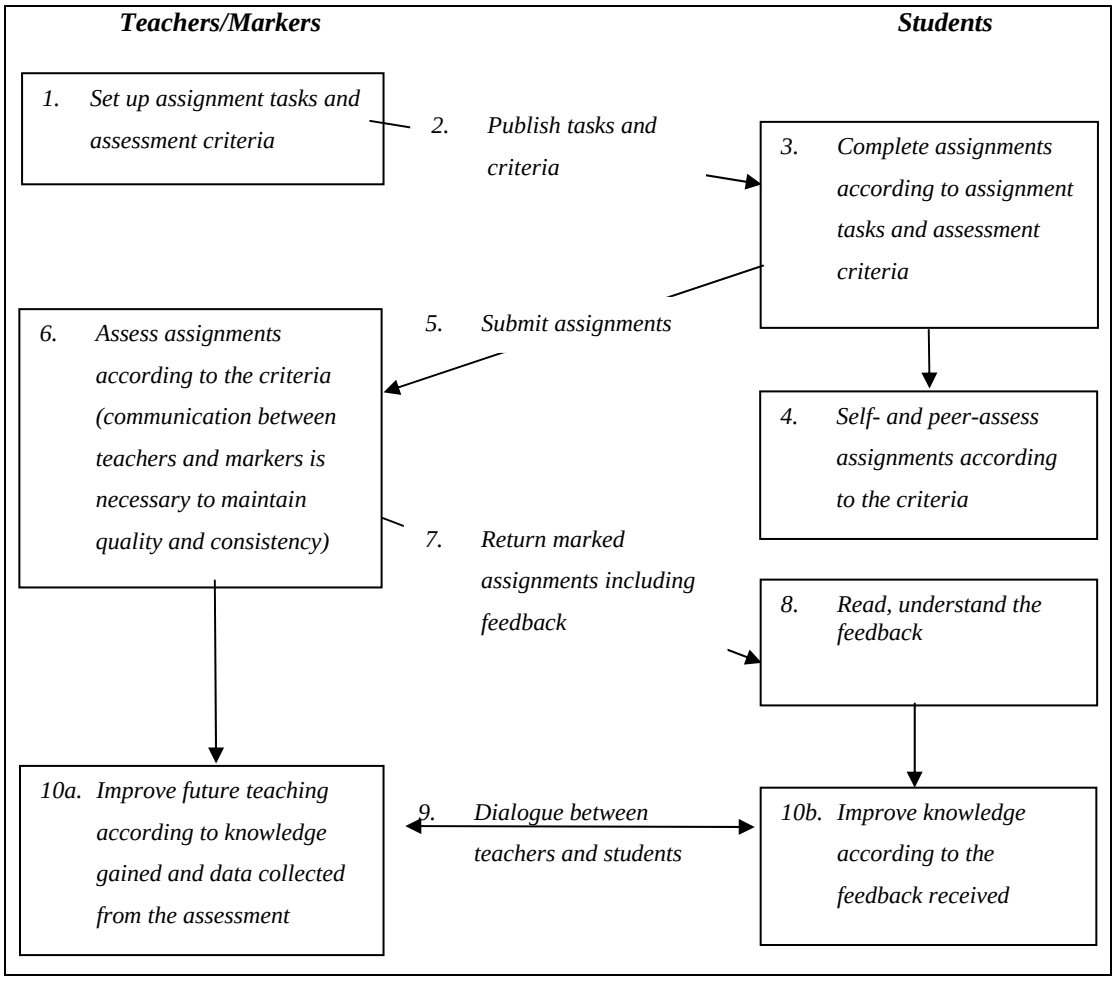


Figure 3-1 Assignment assessment life cycle

For a specific topic of assignment, teachers need to set up the assignment task and the assessment criteria first (Activity 1). The assessment criteria must be clear and each criterion should be associated with a specific aspect of the assignment task. Students can be involved in the setting of the assessment criteria so that they can understand more clearly what they are expected to achieve in the assignment. Not only the assignment task but also the assessment criteria must be revealed to students (Activity 2) before they can start their assignments (Activity 3). It must be ensured that the task and assessment criteria are fully understood by the students. This can be helped by a dialogue between teachers and students (Activity 9). When students have completed their assignments, they can carry out self-assessment and peer-assessment based the assessment criteria (Activity 4). Teachers need to monitor the process of self-assessment and peer-assessment and provide feedback on the feedback created by students during self-assessment and peer-assessment via interactions with students (Activity 9). After students have submitted their assignments (Activity 5), teachers/markers need to assess summatively (assigning grades or marks) and formatively (providing high-quality formative feedback, for example textual comments that are conducive to learning) the submitted assignments according to the predefined assessment criteria (Activity 5). For the formative purpose, the feedback provided should be associated with the corresponding assessment criteria. The quality and consistency of the feedback has to be maintained. This can be achieved by the sharing of assessment samples, discussions among teachers and markers, and double-marking by different markers. Marked assignments, including feedback, must be returned to students in a timely manner (Activity 7). After students have received the marked assignments, they need to read and understand the feedback provided by teachers/markers (Activity 8) and in turn take actions following the feedback to improve their learning (Activity 10b). Teachers need to help students to understand the feedback and remind them in dialogues to act on the feedback (Activity 9). The post assessment dialogues can also help teachers to identify whether the feedback was conducive to learning. Finally, teachers need to analyse the assessment results and adjust their future teaching accordingly (Activity 10a).

A different concept with a similar name – the assessment development cycle, is used for the policy for the assessment of student achievement in Auckland University of

Technology (AUT, 2003). The assessment development cycle describes the activities that need to be included to design appropriate assessments for courses.

3.1.2 Onscreen Marking

There are two different approaches in principle to assist the actual marking activity of assignments – automated marking and onscreen marking, as reviewed in Section 2.2.3. The former uses some advanced statistic and natural language processing algorithms to automate the provision of feedback on student assignments. The latter provides human markers with a specific tool which can present the assignment documents electronically to the markers via computer monitors; with the help of this specific tool, human markers add feedback (either formative, like textual comments, or summative, like numeric marks) to the displayed documents.

As discussed in Section 2.2.3, automated marking is not a suitable solution for formative assessment of student assignments. Into the foreseeable future human markers will still be needed to understand the thinking patterns reflected in student assignments and in turn provide students with formative feedback conducive to their leaning. This determines that a generic solution for using computers to facilitate formative assessment of assignments has to be based on human markers supported by onscreen marking technology.

However, onscreen marking requires functionalities beyond allowing human markers to add some comments to the assignment documents as made possible by the commenting functionality of most word processing tools. For example, the created comments should be associated with the corresponding assessment criteria for students to grasp their strengths and weaknesses quickly as suggested by formative assessment theories, and the manner in which the comments are added should replicate the paper-based assessment approaches so that teachers do not need too much extra training and students can be provided with feedback in familiar formats. How to implement onscreen marking will be discussed in detail in Section 3.2.3.

3.1.3 A Generic Solution

The principle of generic solution requires that the solution needs to be able to handle all assignments electronically submitted by students. From the technical view point, it means that the solution should be capable of dealing with all the document types that can be used by students to complete and electronically submit their assignments, provided that these document types are printable, since this project only targets written assignments. This is based on the facts that:

1. Assignments of different disciplines may require different document types. For instance, computer science students may complete and submit their UML data modeling assignments in Microsoft Visio format (for example documents with the *.vsd* extension) and the Microsoft Word document type is widely used by social science students to write their essay assignments.
2. For the same assignment topic, different students may use different software packages and submit their assignments in different document types because of the availability of software and student preferences. For example, for a computer science UML data modeling assignment, some students may use Microsoft Visio while others may use IBM Rational Rose to draw the UML diagrams.

3.2 Requirements Analysis

This section will discuss in detail what facilities need to be provided to implement the concept of using computers to support formative assessment of assignments introduced in Section 3.1. Basically, the discussion in this section follows the sequence of the activities included in the assignment life cycle – it begins with the discussion about the support for setting and publishing assignment task and assessment criteria (Activity 1 and Activity 2 in Figure 3-1) and ends with the description of how to support analysis of assessment results (Activity 10a in Figure 3-1). This section concludes with a summary which categorizes the required facilities into three different groups.

3.2.1 Setting Up and Publishing Assignment Tasks and Assessment Criteria

As discussed in Section 3.1.1, for any assignment topic, teachers have to set up the task and assessment criteria first (Activity 1 in Figure 3.1). This should be facilitated by the provision of a specific tool which enables teachers to create, modify and save the detailed description of the assignment task and the assessment criteria. Although most word processing applications can be used for this purpose, the document type in which to save the assignment tasks and the assessment criteria needs to be further investigated. The selected document type has to enable human markers (teachers/markers or students in self-assessment or peer-assessment) to associate feedback with the corresponding assessment criterion easily when they assess assignments against the created assessment criteria.

The electronical distribution of assignment tasks and assessment criteria via a computer network to students and other markers offers a number of advantages (for example, efficiency and the saving of resources) compared to handing out the hard copies (Freeman and Lewis, 1998). However, a publishing mechanism has to be set up to ensure the efficiency. For example, to put the assignment tasks and assessment criteria on a website which can be accessed by students (like what teachers are doing with WebCT online assignments) is more efficient than emailing students with the tasks and criteria attached. As suggested by researchers the availability of assessment samples which have been marked using the same assessment criteria can help students and other markers understand the assessment criteria more clearly (Brown and Knight, 1994; Freeman and Lewis, 1998). Therefore, this publishing mechanism needs also to enable teachers to deliver some assessment samples, which were marked against the same assessment criteria, to students and other markers. In addition, the publishing mechanism should be set up with the facilities for teachers to efficiently publish some other parameters of assignment, such as the submission due date, the submission cut-off date (after which no assignments are accepted) and whether group work is allowed.

Dialogues among teachers, markers and students (Activity 9 in Figure 3-1) can help markers and students to understand assignment tasks and assessment criteria. Computer communication technology makes carrying out such dialogues more

efficient than the traditional face-to-face communication. For example, students can email their questions to teachers or put the questions on an electronic discussion board. However, to send emails or put questions on a discussion board, students have to use tools additional to the applications that they use to view the tasks and criteria. Further, they need to quote the questioned task and criteria in their messages. More efficient mechanisms need to be investigated to carry out such communications. For example, teachers and students should be provided with a specific tool that does not only display the assignment task and assessment criteria, but also enables students to select a specific part of the task or criteria and input their questions on the selected content. The tool can automatically send a message, including the question and the selected content, to the teacher who created the task or criteria. When the teacher uses the same tool to read the received message, the corresponding assignment task or assessment criteria document can be opened automatically with the questioned content highlighted so that the question is immediately linked to the correct context and can be answered accordingly.

3.2.2 Submitting Assignments

When students have completed their assignments according to the assignment tasks and assessment criteria provided by their teachers (Activity 3 in Figure 3-1), they need to be provided with the means of submitting their assignments in an efficient, reliable, and secure way (Activity 5 in Figure 3-1). Electronic versions of assignments are necessary for using onscreen marking technology to mark assignments. To submit the assignments through a computer network is more efficient than to hand in floppy disks/CD-ROMs or to scan hard copies submitted into electronic documents. A mechanism for the management and administration of the submission process and the storage of submitted assignments needs to be set up to ensure the assignments are dealt with reliably and securely:

- It is important that the physical storage of the received assignments is safe – these assignments must be protected from being damaged or lost due to potential problems of the teachers' computers. Teachers should not be able to change the original content of the submitted assignments. This can be achieved with students submitting their assignments to a central server run and maintained by the university or department. Teachers access assignments via

suitable network applications or download these assignments to their own computers.

- The assignments must be submitted to the teachers who are responsible for the assignment. If students submit assignments to a central server, the submitted assignments can only be accessed or downloaded by the students who have submitted the assignments and the teachers responsible for the assignments.
- Students need to receive feedback immediately on whether the assignments have been submitted successfully and what has been submitted (for example, a list of the submitted filenames) so that students can decide whether they need to resubmit their assignments.
- The submission status of each student, i.e. whether and at what time the student has submitted the assignment, needs to be recorded.
- Submission of group assignments needs to be supported. For example, if one group member has submitted an assignment on behalf of the whole group, the submission status of all group members should be recorded as "submitted".

3.2.3 Marking of Assignments by Teachers or Markers

This section discusses the principles that need to be followed to implement onscreen marking. Besides the provision of an onscreen marking tool, some other issues that are related to the marking process of assignments are also discussed in this section, including allocating submitted assignments to different markers, the management of the marking process, and quality and consistency control of the feedback.

❖ **Presenting Assignments Electronically**

A specific onscreen marking tool is needed to present the submitted assignments electronically to human markers (teachers/markers or students in self-assessment and peer-assessment) with the original content and presentation effects (page setup, layout, font size, etc.) of the assignments intact. Human markers must not be allowed to modify the content and presentation effects of the original assignment documents via this onscreen marking tool.

As discussed earlier in Section 3.1.3, a generic solution should not restrict students to the use of some specific document types for completing and submitting their

assignments. This leads to the requirement that the proposed onscreen marking tool has to be able to handle all printable document types. It is hardly feasible and would be inefficient to develop an application which can display all printable document types directly. This means that all the submitted assignments should be converted to a specific document type at some stage (for example, after the teachers have received the submitted assignments) before using the proposed onscreen marking tool for marking. The proposed onscreen marking tool should use this specific document type to display the assignments. It is important that the conversion preserves the content and presentation effects of the original assignment documents. The Portable Document Format (PDF, 2003) is one possible document type that can be used for the proposed onscreen marking tool as all printable documents can be converted to PDF format leaving the original content and presentation effects intact.

❖ **Annotating Assignments Onscreen**

The proposed marking tool needs to enable human markers to assess the assignments onscreen in an intuitive way. The manner in which human markers use the onscreen marking tool to assess the assignments should be similar to what teachers are doing in pen-and-paper assessment, so that human markers do not need extra training to use the onscreen marking tool and students can be provided with results in familiar format. To mark paper-based assignments, most teachers like to "scrawl" on the assignments as illustrated by Figure 3-2. This behaviour should be replicated in the proposed marking tool:

1. Human markers need to be able to add textual annotations to the displayed assignment documents as formative feedback.
2. Each annotation needs to consist of two components – a graphic component (such as a line, a circle, a rectangle, etc.) that human markers can draw on and move to any area of the displayed assignment document to identify which part of the assignment the annotation refers to; a textual comment which provides the main message of the annotation and is entered by human markers via the keyboard.
3. The two components of the annotation need to be linked together, for example, using a line to link them together in the same way as illustrated in Figure 3-2.

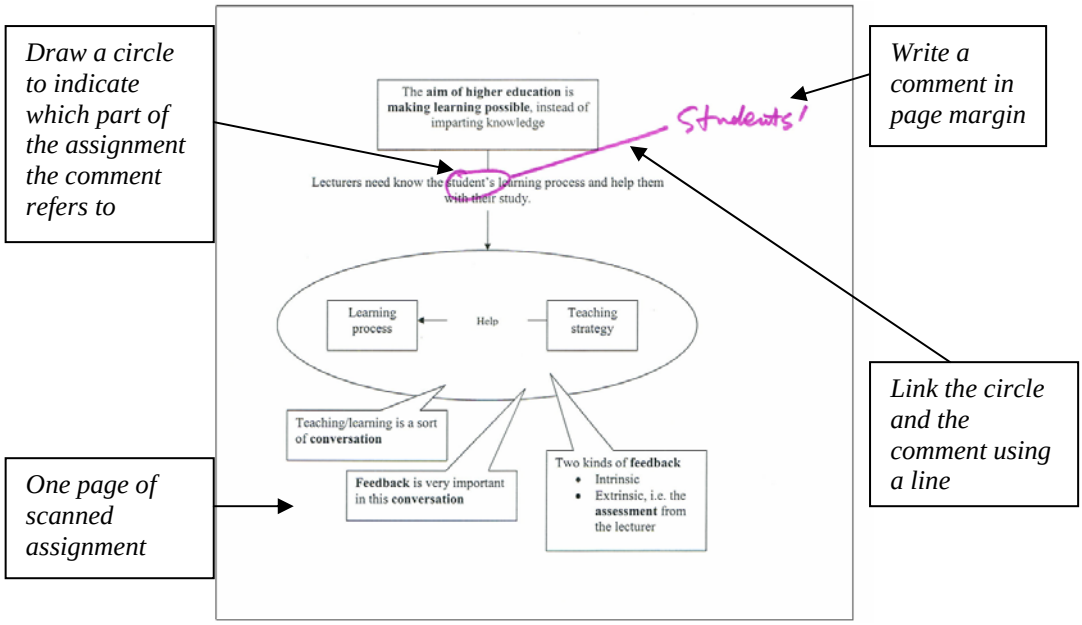


Figure 3-2 Paper-based assessment

❖ Linking Feedback with Assessment Criteria

Formative assessment theories demand that feedback needs to be associated with the relevant assessment criterion predefined by teachers (Freeman and Lewis, 1998). Therefore, the proposed onscreen marking tool needs to provide the facility to enable human markers to link the created annotations with the corresponding assessment criteria. The association between annotations and assessment criteria needs to be easily identifiable. For example, applying different colours to the graphical component or the textual comment of the annotation can be used to distinguish between different assessment criteria.

❖ Summary Sheet

A summary sheet which groups the feedback under the different assessment criteria can assist students to grasp their strengths and weaknesses quickly, as requested by assessment theories (Brown and Knight, 1994). Therefore, the proposed onscreen marking tool should be able to automatically generate such a summary sheet for each assignment based on the annotations created for this assignment.

❖ Awarding Marks/Grades to Assignments

Human markers need to be able to award numeric marks or letter grades to an assignment. This requirement is based on the fact that the assessment of assignments

is often both formative and summative – on the one hand, teachers can provide students with formative feedback in the format of textual comments on their assignments; on the other hand, teachers need to allocate numeric marks or letter grades to the assignments since these marks/grades contribute to the students' final grades of the courses they are taking. Black and Wiliam (1998) argue that summative assessment may negatively affect learning. To avoid this negative impact on learning, the numeric marks or letter grades should be separated from the textual annotations, for example, by restricting the appearance of these marks/grades to the summary sheets.

❖ **Storage of Feedback**

It is obvious that the proposed onscreen marking tool should be able to:

1. Save the created feedback, including the annotations (the graphic components, the textual comments, and the associated assessment criteria), the marks or grades, and the summary sheets, in suitable formats;
2. Retrieve feedback from storage and display the marked assignments with the feedback providing the same effects as when the feedback was added to the assignments (for example, the positions of the graphic components and the textual comments).

The way in which to save the feedback needs to facilitate various data processing and manipulation steps. For example, the storage format should make it easy to extract feedback from all marked assignments in order to analyse the assessment results. One possible solution is to save the feedback separately from the original assignment documents to enable flexible access to the feedback, as proposed by Heinrich and Wang (2003).

❖ **Allocating Submitted Assignments to Different Markers**

Teachers need to be able to allocate the submitted assignments to different markers if more than one marker is involved in the marking. An electronic document exchange mechanism needs to be developed to enable teachers to send the assignments to the allocated markers and collect the marked assignments with the feedback from these markers efficiently via a computer network.

❖ **Management of the Marking Process**

The following management functionalities need to be provided to ensure that a large number of assignments can be marked efficiently:

- When a specific assignment document is to be marked, it should be associated automatically with the corresponding assessment criteria so that the proposed marking tool can import the corresponding assessment criteria automatically.
- For each individual submitted assignment, the marking status (i.e. whether an assignment document has or has not been marked) needs to be recorded.
- For the marking of group assignments, it must be ensured that the assignments submitted by different members of a group cannot be allocated to different markers.

❖ **Quality/Consistency Control of Feedback**

As demanded by assessment theories, quality and consistency of feedback must be maintained, especially when more than one marker is involved in the marking (Brown and Knight, 1994; Freeman and Lewis, 1998). An important way to achieve this goal is to double-mark markers' work by teachers as suggested in the literature (Freeman and Lewis, 1998). To double-mark the assignments marked by other markers is time-consuming work. Therefore specific tools should be provided to facilitate teachers in carrying out the double-mark task in an efficient way. For example, listing the feedback extracted from all the marked assignments of a specific topic enables a teacher to go over the feedback quickly without necessarily looking at the marked assignments. The listed feedback needs to be linked with the corresponding marked assignments in some way so that teachers can find the context and make some modifications quickly if they suspect some comments or marks/grades may not be suitable. For example, double-clicking on a listed comment should cause the proposed onscreen marking tool to display the corresponding assignment documents with the selected comment highlighted. It should be possible to sort the comments and marks/grades by the order of the creators so that teachers can identify whether consistency is maintained among different markers. It is clear that all these proposed operations need the flexible access to the stored feedback. This kind of flexible access can be achieved by saving the feedback separately from the original assignment documents as suggested by Heinrich and Wang (2003).

3.2.4 Self-Assessment and Peer-Assessment

During self-assessment or peer-assessment, students not teachers or markers need to provide formative feedback on their own or their peers' work. From this point of view, self-assessment and peer-assessment are nothing other than two special formats of formative assessment and all the formative assessment principles (explicit assessment criteria, association between feedback and criteria, etc.) also apply to self-assessment and peer-assessment (Freeman and Lewis, 1998). This means that the same onscreen marking tool discussed in the previous section can also be used by students to add textual annotations or marks/grades to their own or their peer students' assignments (Activity 4 in Figure 3-1). For successful self-assessment and peer-assessment, teachers need to help students to develop the required skills (Brown et al., 1997; Brown and Knight, 1994; Weeden et al., 2002). This can be achieved by providing students with assessment samples. Required for this is an electronic document exchange mechanism to enable teachers to deliver the assessment samples to students via a computer network. Furthermore, teachers need to provide feedback on the feedback created by students during self-assessment or peer-assessment (Brown and Knight, 1994; Freeman and Lewis, 1998). This requires that the proposed onscreen marking tool provides a facility for enabling teachers to add textual comments to the annotations created by students during self-assessment and peer-assessment. One possible solution to implement this is to allow an individual annotation to have more than one textual comment (so that a teacher can add an additional comment to an existing annotation as the feedback on the feedback created by students) and to record the creator of each comment (so that students can identify which comments are from their teachers); this proposed solution requires that each individual textual comment and not the entire annotation needs to be linked with a specific assessment criterion since teachers and students may have different ideas on which criterion should be used for a specific annotation. Additional facilities are needed to enable teachers to collect the assignments marked by students in self-assessment and peer-assessment and return these assignments with their comments to students through a computer network.

3.2.5 Returning Marked Assignments to Students

As demanded by formative assessment theories, students need to be provided with feedback on their assignments as early as possible (Brown and Knight, 1994; Freeman and Lewis, 1998). To return feedback to students in a timely manner (Activity 6 in Figure 3-1), a mechanism for sending back the marked assignments including feedback electronically via a computer network needs to be created. This mechanism needs to ensure that teachers can return marked assignments to students in a managed, secure, and efficient way:

- If more than one marker is involved in the marking, teachers first need to collect all the marked assignments from the markers and then check these marked assignments to ensure that quality and consistency are maintained before the marked assignments can be returned to students, as discussed earlier under "Quality/Consistency Control of Feedback" in Section 3.2.3.
- Teachers must be able to fully control the process of returning marked assignments – teachers need to be able to determine which marked assignments can be returned (for example, only the assignments that have been checked for quality and consistency) and at which point-of-time these assignments are returned.
- It is important that the marked assignments are returned to students securely to preserve confidentiality – each individual marked assignment with the feedback must be returned to the student who submitted it.
- If an assignment is submitted as group work, each member of the group, not only the student who submitted the assignment on behalf of the group, should receive a copy of the marked assignment.
- The marked assignments should be returned to students in an efficient way. For example, teachers should be able to return the marked assignments to the whole class in one operation instead of having to email students the marked assignments one by one.

The same principles apply when teachers need to return the feedback on the feedback created by students during self-assessment and peer-assessment.

3.2.6 Reading of Feedback by Students

Assessment theories acknowledge that formative assessment can be effective only when students read, understand and act on the feedback provided by their teachers (Black and Wiliam, 1998; Higgins et al., 2002). This is illustrated by Activity 8 and Activity 10b in Figure 3-1. Researchers also argue that the dialogue about the feedback between teachers and students is necessary not only to help students understand the feedback on their assignments but also to remind students to take actions to improve learning (Freeman and Lewis, 1998; Murphy and Torrence, 1988). This means a specific tool is required to present marked assignments including feedback to students and this tool should to be able to facilitate the communication between teachers and students. The onscreen marking tool proposed in Section 3.2.3 forms a possible solution for such a tool, provided the required communication functionalities are added. One example for such communication functionalities could be that a student is able to send a question message to the teacher with respect to a specific comment on his/her assignment. For this question message, the student should only need to choose the comment and input why he/she does not understand this comment. The proposed tool will associate this message with the questioned comment and email the message to the teacher automatically.

3.2.7 Analyzing Assessment Results

Formative assessment is beneficial not only to learning but also to teaching. The analysis of assessment results can help teachers review the effectiveness of their teaching and make adjustment accordingly (Freeman and Lewis, 1998). To support this kind of analysis (Activity 10a in Figure 3-1), a tool which is capable of extracting feedback from marked assignments should be provided. For example, simply listing the comments grouped by the assessment criteria can help teachers identify quickly in which area remedial work is required. This can be easily fulfilled if the feedback is saved separately from the original assignment documents, as proposed in the “Storage of Feedback” part of Section 3.2.3.

3.2.8 Summary

The detailed discussions about the requirements for using computers to support formative assessment of assignments have been presented in the sections from 3.2.1 to 3.2.7. These requirements can be collated into three groups:

1. From the viewpoint of users which can be teachers, markers, or students, the following facilities should be provided:
    * The teachers viewpoint:
        - The facility to set up assignment tasks and assessment criteria and publish the created tasks and assessment criteria along with some assessment samples to markers and students through a computer network;
        - The facility to receive assignments submitted by students via a computer network and allocate these assignments to different markers (and students in the case of peer-assessment);
        - The onscreen marking tool that enables teachers to mark the assignments according to the assessment criteria onscreen and save the assessment results electronically;
        - The facility to collect marked assignments with feedback from markers (or students in the case of self-assessment and peer-assessment);
        - The facility for maintaining quality/consistency of the feedback and conducting assessment result analysis;
        - The facility to return the marked assignments with feedback to students;
        - The facility to have discussions with markers and students to help them understand the assessment criteria or the feedback;
        - The facility to help students carry out self-assessment and peer-assessment by the means of providing feedback on the feedback created by students.
    * The markers viewpoint:
        - The facility to access the assignment tasks, assessment criteria, and assessment samples published by teachers;
        - The facility to access the assignments allocated by teachers for

marking;

- The onscreen marking tool to mark the assignments;
- The facility to send the marked assignments to teachers;
- The facility to have assessment related communications with teachers, other markers, and students;

* The students viewpoint:
  - The facility to access the assignment tasks, assessment criteria, and assessment samples published by teachers;
  - The facility to submit assignments (including the assignments marked in self-assessment and peer-assessment) to teachers;
  - The facility to access marked assignments;
  - The facility to access the assignments allocated by teachers for marking in self-assessment and peer-assessment;
  - The onscreen marking tool to mark the assignments in self-assessment and peer-assessment and to view the marked assignments with the feedback provided by their teacher or markers;
  - The facility to have assessment related communications with teachers, markers, and other students;

2. The electronic document exchange and communication channels among teachers, markers, and students should be provided for:
   * Publishing assessment criteria and assessment samples;
   * Submitting assignments;
   * Distributing submitted assignments to markers and collecting marked assignments from markers;
   * Returning marked assignments;
   * Carrying out dialogues or discussions among teachers, markers, and students.
3. To provide an efficient, reliable, and secure solution, the following background management functionalities should be provided:
   * To manage the assignment submission process and the submitted assignments;
   * To manage the marking process;
   * To manage the document exchange and communications among teachers, markers, and students introduced in Item 2;

* To manage the process of returning marked assignments;
* To perform all the above functionalities efficiently, causing the least possible amount of non-task related overhead;
* To protect privacy and confidentiality of all parties involved.

3.3 Specification of the WAA System

To implement the principles of using computers to facilitate formative assessment of assignments, the Written Assignment Assessment (WAA) system has been developed by this research project. This section presents the specification of the WAA system. Technical issues for the implementation of the system will be discussed in Chapter 4.

In principle, the WAA system is specified as a networked system with the following features:

- The WAA system takes advantage of LMS (such as WebCT or Blackboard) installed at universities to manage assignments and student submissions. It also uses the LMS services as the bridge for the communication and document exchange among different users (teachers, markers and students).
- The WAA system provides users with the functionalities that are necessary for using computers to facilitate formative assessment from the user's view point, such as to create assignment tasks and assessment criteria, to allocate student submissions to markers, to mark assignments, as summarized in Section 3.2.8. These functionalities are provided through the applications installed on users' local computers.
- The WAA system is capable of converting documents of printable document types to PDF files and of rendering PDF documents through a graphical user interface.

3.3.1 Utilizing LMS Services

As discussed in Section 3.2, a comprehensive solution for using computers to facilitate formative assessment of assignments requires a communication and document exchange channel for transferring documents and assessment data (such as

assessment samples, assignments, and assessment related discussion messages) between teachers, markers and students. This means that the software system intended to provide such a comprehensive solution should be a distributed system – a system consisting of applications distributed to different computers which are connected by a computer network. There are two different ways to implement the communication and document exchange channel in a distributed system – the peer-to-peer method and the client/server method.

In the peer-to-peer method, applications sitting on different computers communicate with each other directly. For example, when a student submits an assignment, the application on the student's computer will send the assignment document directly to the application installed on the teacher's computer. The same applies for a teacher returning the marked assignment, as illustrated by Figure 3-3.

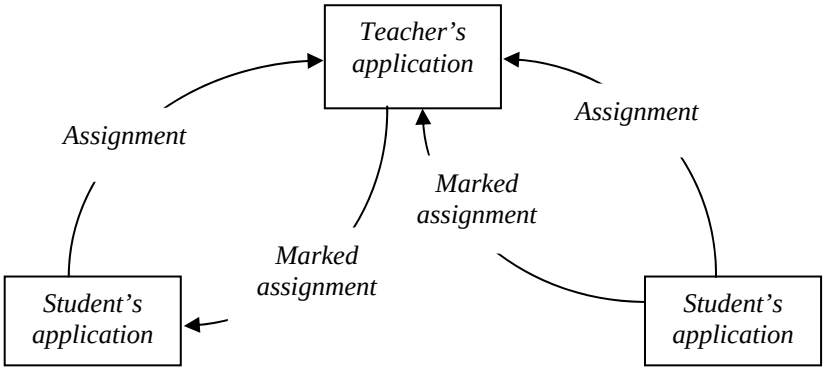


Figure 3-3 Peer-to-peer communication

In the client/server method, applications on different computers (i.e. the client applications) do not connect to each other directly. They are all connected to the same application (i.e. the server application) sitting on a central server computer. The typical way to implement client/server communication and document exchange is:

1. A user uses a client application to send some data with identification (for example, username) of the recipient(s) to the server application.
2. The server application stores the received data and the identification of the recipient(s) on the server computer.
3. Another user uses a client application connect to the server application, if the user is the recipient of some specific data (this is verified by the server

application through the comparison of the identifications), then the user is eligible to fetch a copy of the data from the server application. The original data is still stored on the central computer.

In this way, when students need to submit their assignments, they do not submit the assignments directly to their teacher but send the assignment documents and the information about which course this assignment is for to a central application, and the teacher of this course can get the copies of the submitted assignments from the server application. This is depicted by Figure 3-4.

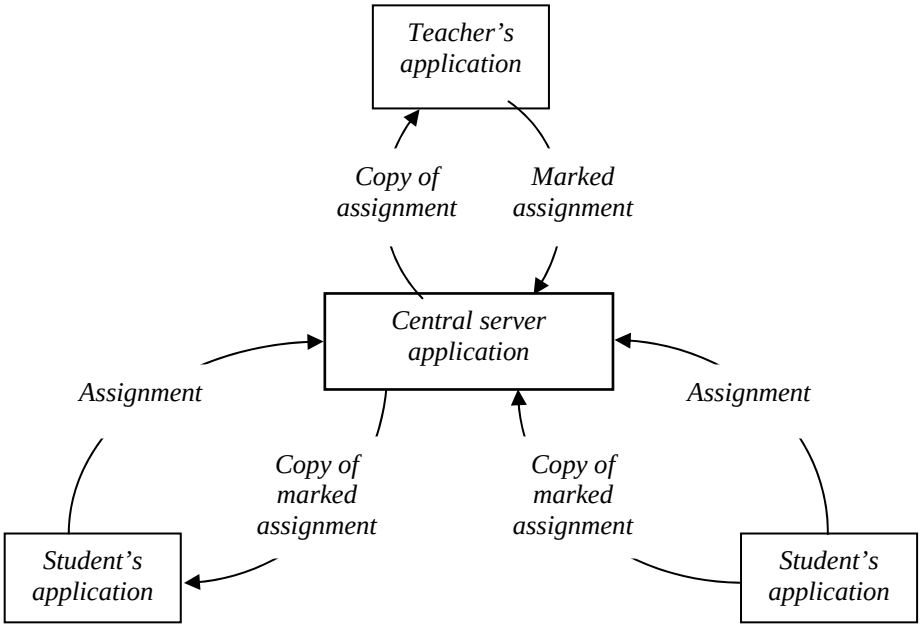


Figure 3-4 Client/Server communication

The client/server method has the following advantages over the peer-to-peer method when used for the communication and document exchange channel in facilitating formative assessment of assignments:

- In the client/server method, data can be sent no matter whether the connection to the application of the recipient is available, as long as the sending application is connected to the server application. However in the peer-to-peer methodology, it is obvious that the communication can succeed only if 1) the two computers, on which the sending application and the receiving application are installed, are physically connected, for example, connected to the same intranet or the Internet; 2) both of the sending and receiving applications are

running; and 3) the sending application knows the network address of the receiving application.

- The storage of data is more secure in the client/server method. The original data is stored on the central server machine which can be maintained by university or department technicians and the recipient only gets a copy of the data from the server application. Even if the recipients have lost or damaged the data on their local computers, they can always get another copy of the data from the server application. As the original data is stored on the central computer, the sender does not need to worry about the content being modified by the recipient. While in the peer-to-peer method, the received data (such as the submitted assignments) are unprotected from being lost or damaged due to the software or hardware failure of the recipient's computer and there is no way to prevent the recipients from modifying the original content of the received data.

Based on the comparison between the client/server and the peer-to-peer methodologies for implementing the communication and document exchange among different applications, this project decided to develop the WAA system as a client/server network system. The server application of the system sits on a central machine and works to manage assignments, student submissions, and the returning of marked assignments. It also acts as the bridge for the document exchange and communication among different client applications of the system. The client applications of the system are distributed on the end users' computers and connected to the server application through a computer network. These client applications provide end users with the functionalities that are necessary from the user's view point for using computers to facilitate formative assessment of assignments, such as publishing assignment task and assessment criteria, marking assignments, and analyzing assessment results.

As the major LMS systems, like WebCT and Blackboard, already provide most of the management functionalities required for using computers to facilitate formative assessment of assignments, this project decided to use LMS services as the server side of the WAA system. For example, the WAA system can use the WebCT online assignment facility to set assignment submission due date and record student

submission status. The online assignment module and the internal email system of WebCT can be utilized by the WAA system as channels for document exchange (for example submitting assignments by students and publishing assessment criteria by teachers) and assessment related communication (for example to send email to a teacher via the WebCT internal email system). All these can be achieved by the WAA system without having its own server application as long as it can interact with the WebCT server. From another view point, the WAA system can be thought of as a special client side application of a LMS system (see Figure 3-5). The benefits of taking advantage of LMS services over developing its own server side application can be listed as:

- User acceptance – teachers/students are familiar with managing courses and assignments, publishing/accessing course and assignment material, submitting assignments, and delivering grades in the environment of the LMS installed at their university. Teachers and students could be reluctant to use a new system to manage and submit assignments, as encountered by the system presented in Preston and Schackelford (1999).
- Technical support issues – using existing LMS services eliminates the need for additional server hardware and the maintenance work to ensure that the server software runs in a reliable and secure way. An individual user can easily install and run the WAA system on his/her own computer without any support from the university or department.
- Development effort – the workload of developing such server side software would be immense, especially for implementing the various management functionalities required for online assignments, such as the management of assignment settings and student submissions.

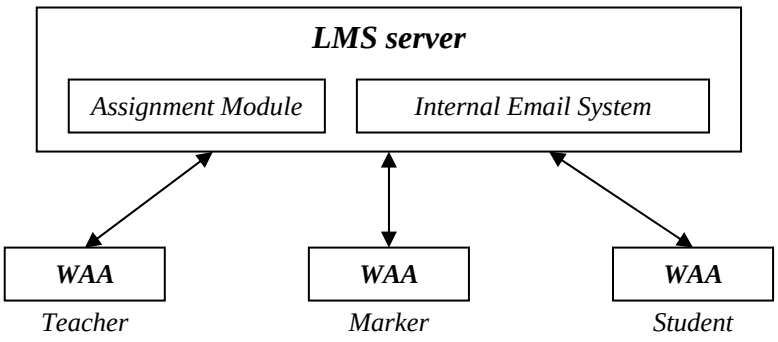


Figure 3-5 Using LMS services

As shown in Figure 3-5, the WAA system actually behaves as the client application of an LMS system. It is obvious that the WAA system must be able to exchange data with LMS through computer networks. Also the WAA system needs to provide the functionalities that is necessary from user's point of view for using computers to facilitate formative assessment of assignments but not supported by LMS, for example the onscreen marking tool, the management of marking process, and feedback quality control.

3.3.2 Components of the WAA System

As discussed in Section 3.3.1, the WAA system will take advantage of LMS for most of the management functionalities and LMS also acts as the bridge for document exchange and assessment related communication. The WAA system actually is a special client application of LMS. Therefore, in addition to the provision of the functionalities that are necessary from the user's view point for using computers to facilitate formative assessment of assignments, the WAA system must be able to interact with LMS. For example, it should be able to upload/download documents to/from the LMS server so that students can use the WAA system to submit assignments and teachers can use the system to publish assignment tasks and download the submitted assignments; it should be able to send/receive email message to/from the internal email system of LMS so that teachers, markers and students can carry out assessment related discussions. The WAA system also needs to provide the management functionalities on the marking process that are not offered by LMS, for example the recording of marking status of each individual assignment as discussed under "Management of Marking Process" in Section 3.2.3.

As discussed earlier in Section 3.2, an onscreen marking tool is necessary for marking of assignments. The document type used by this tool for rendering an assignment document should be a generic document type that all printable document types can be converted to. This project has chosen the PDF format as this generic document type since all printable documents can be converted to PDF files without modifying the original content and presentation effects. The WAA system should provide the functionality to convert documents of other printable types to PDF files.

Based on the functionalities specified for the WAA system, this project finally specifies the WAA system with three components – a LMS interaction module, an assignment assessment module, and a document conversion module (see Figure 3-6).

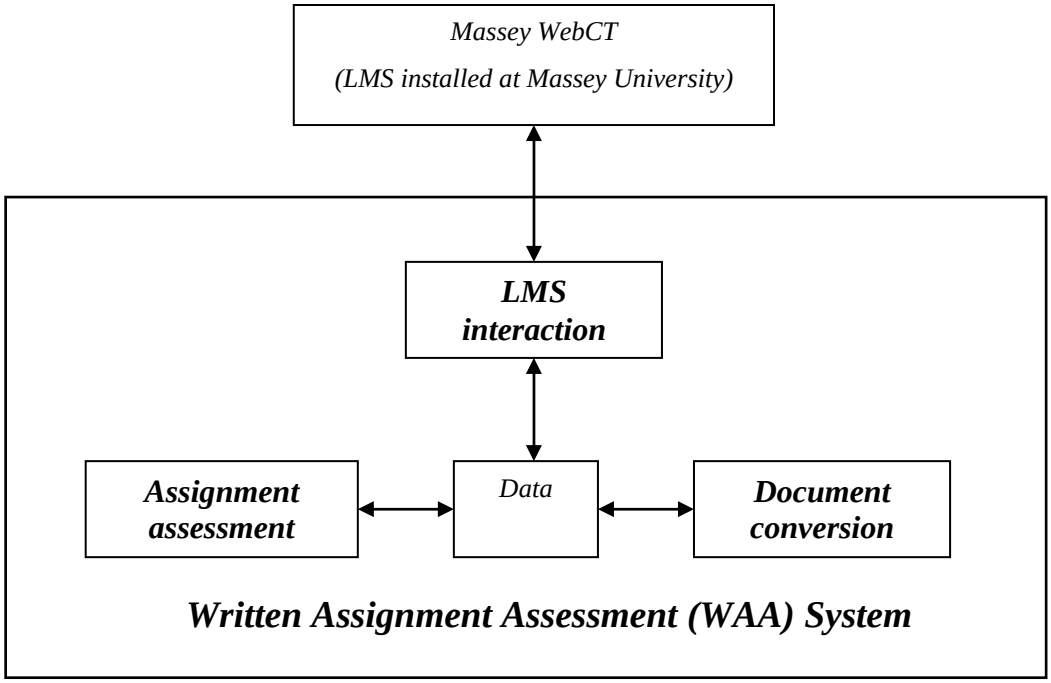


Figure 3-6 Components of the WAA system

The LMS interaction module allows users to send/fetch data to/from the LMS system. At Massey University this is WebCT Campus Edition 4.1 (WebCT, 2004).

The assignment assessment module provides users with all the functionalities that are necessary from a user's point of view for using computers to facilitate formative assessment of assignments. The core of such an assignment assessment module is an onscreen marking tool that can be used by users to mark the assignments (in PDF format) stored on their local computers.

The document conversion module needs to be able to convert all printable documents to PDF format which is the document type the onscreen marking tool works with. It needs to allow users to batch convert a set of non-PDF documents to PDF format. For example, after a teacher has downloaded the assignment documents for a whole class from WebCT, the teacher should be able to convert all these documents to PDF in one operation without having to access each individual document to convert it to PDF

format. Such automated support is required to fulfill the goal of efficiency set for this research project and increases user acceptance by automating mundane tasks.

3.3.3 Summary

As introduced in Section 3.3.1 and Section 3.3.2, to implement the principles of using computers to facilitate formative assessment of assignments, this research project has specified the WAA system. The main features of this WAA system are:

- It acts as a client application of LMS.
- It consists of three components:
  - A LMS interaction module which is able to communicate with LMS through a computer network;
  - An assignment assessment module which provides the functionalities which are necessary from the user's point of view for using computers to facilitate formative assessment. The core of this assessment assignment module is an onscreen marking tool which is able to display PDF documents.
  - A document conversion module which is capable of converting all printable documents to PDF format.

Chapter 4 Implementation of the WAA System

As discussed in Section 3.3.2, the WAA system is specified with three components – the assignment assessment module, the LMS interaction module and the document conversion module. This project has implemented these components of the WAA system via three separate applications – an onscreen marking tool, the MARK application, which is the core of the assignment assessment module; the WebCTConnect application which was developed as the LMS interaction module and is able to interact with the WebCT server installed at Massey University via the Hypertext Transfer Protocol (HTTP, 1999); and the ToPDF application for the document conversion module. This chapter will discuss the implementation issues of these three applications.

4.1 The MARK Application

MARK is an onscreen marking tool that uses the PDF format to display assignment documents and allows human markers to add annotations to PDF pages. It was developed based on the MarkTool application described in Heinrich and Lawn (2004) and reviewed in Section 2.2:

- It uses the same software API to render PDF documents as the MarkTool application.
- The annotations created by MARK have a similar format as the annotations created by the MarkTool application – the former consists of a graphic component and one or more textual comment, while the latter consists of a graphic component and only one textual comment.
- Like the MarkTool application, MARK stores the annotations separated from the content of the original PDF documents.

The MARK application also extends the MarkTool application:

- In the MARK application, the rendering of PDF documents is more stable and faster.
- Unlike the MarkTool application which stores the annotations of all PDF documents in a single Microsoft Access file, the MARK application creates a

marked assignment document (with the file extension of *.mrk*) for each PDF document and stores the annotations in the *.mrk* document using XML format.

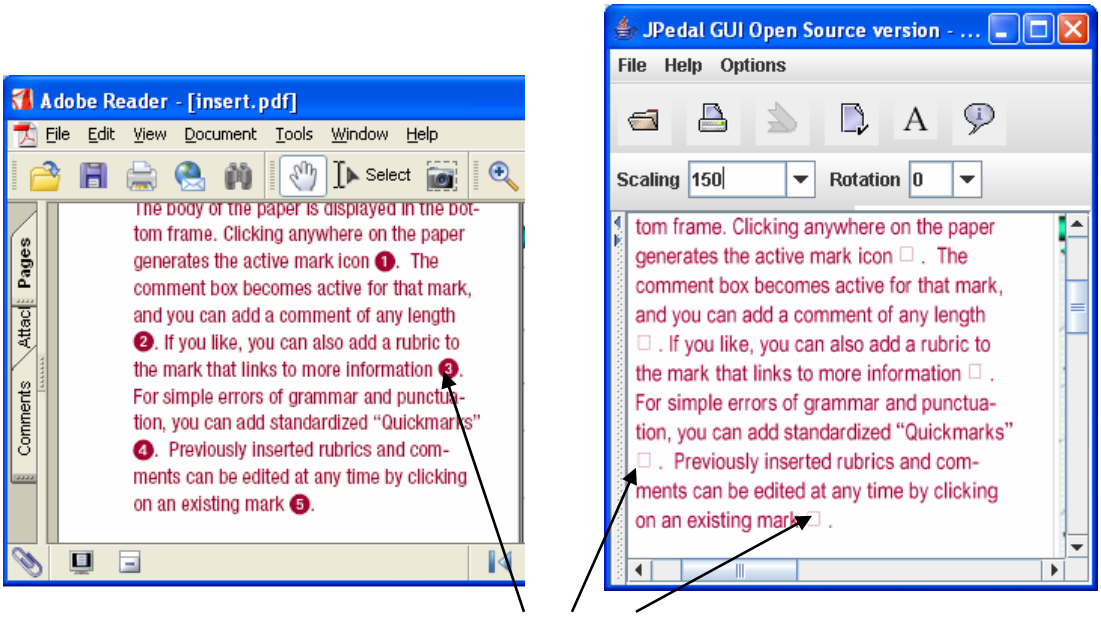
- The MARK application provides overall support for using assessment criteria in assessing assignments, while the MarkTool application only provides very limited support for this aspect.
- The MARK application is capable of generating a summary sheet for each marked PDF document, while the MarkTool application does not provide such a facility.

This rest of this section will discuss in detail the software technologies and some key APIs used to implement the above functionalities of the MARK application.

4.1.1 Selection of PDF Rendering API and Development Language

As discussed in Chapter 3, this project decided to select PDF as the format to be used by the onscreen marking tool of the WAA system to render assignment documents. Therefore, the question of how to render PDF documents has to be addressed at the design stage of the MARK application development. It is impracticable for this master's project to develop its own PDF-rendering software solution. Such software development would require a thorough mastery of the PDF specifications published in PDF (2003) which is an 1172-page book. The only choice left for this project is to find an off-the-shelf PDF-rendering API. There are a number of such APIs available. However, most of these APIs are commercial products and not free, for example the PDF-rendering API packaged in the Adobe Acrobat SDK (2004a) and some third-party APIs, such as the PDF rendering engine of ICEpdf developed by ICEsoft Technologies Inc. (2004) and 3-Heights PDF Viewer and Viewer Pro developed by PDF Tools AG (2004). There are a few free PDF-rendering APIs, for example, Adobe Acrobat Viewer for JavaBean used in the marking tool presented in Heinrich and Wang (2003), the open source version of the JPadel Java PDF-rendering API provided by IDRsolutions (2004), and the GhostScript Interpreter API used in the MarkTool application (Heinrich and Lawn, 2004). However, Adobe Acrobat Viewer for JavaBean is not a good choice as 1) it is a testing API which only had "undergone limited testing and may contain bugs, errors, and other problems that could cause system failures" as claimed in the README file distributed with the software (Adobe,

2004b); and 2) Adobe is not active in the development of Adobe Acrobat Viewer for JavaBean since Adobe has never updated the software since it was released in 1999. JPedal is a Java class library developed by IDRsolutions to ease the display, manipulation and content-extraction of PDF documents. The developer provides both a commercial and free open source version of the library and is active in enhancing the product. Unfortunately, the open source version does not support embedded fonts (i.e. the definition of the font is embedded in the document so that it can be rendered properly even the font is not installed in the computer) in PDF documents (see Figure 4-1). This disadvantage of the open source JPedal PDF-rendering API determined that it is not suitable to be used by MARK to display PDF documents.

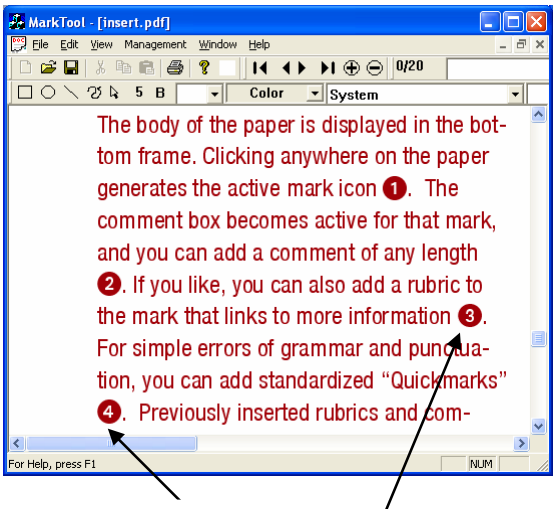


The characters in embedded fonts are rendered properly in Adobe Reader (left), but replaced by squares in open source JPedal (right)

Figure 4-1 No support for embedded fonts by open source JPedal

GhostScript is a set of software that is able to display PDF files (GhostScript, 2004b). It is also capable of converting PDF documents to many different image formats, for example, the BMP, JPEG and PNG formats. For the Windows platform, a GhostScript Interpreter API is available as a C-language dynamic linked library (DLL, 2004) file, *gsdll32.dll*, within the GhostScript software bundle (GhostScript, 2004a). Windows applications can call the functions exported by this DLL file to convert a PDF file to a specific image format or produce memory bitmap data for each individual page of the PDF file. A bitmap is a representation of a raster image which consists of a

rectangular grid of pixels. For a raster image, a bitmap defines the positions of pixels by row and column and the characteristics of individual pixel such as gray-scale and color. However, the GhostScript Interpreter API cannot work alone. The exported functions can only work properly when the whole set of the GhostScript software is installed. This GhostScript Interpreter API is used by the MarkTool application to render PDF documents – the application first creates bitmaps from the input PDF files and then renders these bitmaps as images on the screen. Embedded fonts are well supported by the GhostScript Interpreter API when rendering PDF documents (see Figure 4-2). The developer is active in improving the quality and performance of GhostScript. The software, including the GhostScript Interpreter API (i.e. the *gsdll32.dll* file) had evolved from version 1.0 released at August 1988 to version 8.51 which was released in April 2005. Therefore this project decided to use the GhostScript Interpreter API for PDF-rendering.



Characters in embedded fonts are retained in the MarkTool application which uses GhostScript Interpreter API to render PDF documents

Figure 4-2 Support for embedded fonts by GhostScript Interpreter API

Since the GhostScript Interpreter API is built as a C DLL file for the Windows platform, the easiest way for the MARK application to take advantage of the API is to develop it as a C/C++ Windows application which can simply import and call the functions exported by the DLL file. There are two major ways to develop such a Windows application – to develop it as a C-language Windows-based application using only the low-level Win32 API (Rector and Newcomer, 1997); or to develop it as an object oriented C++ Windows application utilizing some high-level C++ class

libraries, such as the Microsoft Foundation Class (MFC [Shepherd, 2003]) library or the Borland Object Windows Library (OWL [Herbert, 1997]). This project decided to develop the MARK application as a MFC-based C++ Windows application:

- Using object oriented C++ makes the development easier than using the procedural C language.
- The Windows-based C++ class library simplifies Windows programming by providing developers with reusable Windows components that encapsulate low-level Win32 API functions.
- MFC was used in the development of the MarkTool application, using the same class library makes it easier to learn from the implementation of the MarkTool application.

4.1.2 Improvements in Rendering PDF

Although the MARK application uses the same PDF-rendering API, programming language and class library as the MarkTool application, the MARK application and the MarkTool application use different ways in:

- Loading the GhostScript Interpreter API from the DLL file;
- Calling the functions exported by the DLL file to create bitmaps for PDF pages;
- And rendering the created bitmaps as images.

These differences make the PDF-rendering in the MARK application more stable and faster than in the MarkTool application.

❖ **Loading of GhostScript Interpreter API from *gsdll32.dll***

A DLL can be loaded into applications either by implicit linking or explicit linking (Shepherd, 2003). In implicit linking, a LIB file in addition to the DLL file is required (for *gsdll32.dll*, the corresponding LIB file is *gsdll32.lib*). The LIB file contains the names of the functions exported by the DLL file and the filename (but not its full pathname). Programmers first need to add this LIB file to the program's project. Then the exported functions can be called directly. For example, if *gsdll32.lib* has been added, the *gsapi_revision* function exported by *gsdll32.dll* can be called directly. In explicit linking, programmers need to load the DLL by calling the WIN32

LoadLibrary function with filename of full pathname of the DLL file as the parameter, like *LoadLibrary("gsdll32.dll")* or *LoadLibrary("c:/gs/gs8.11/bin/gsdll32.dll")*. The LoadLibrary function returns an *HMODULE* or *HINSTANCE* value which is the handle (pointer) to the loaded DLL. Programmers need to call the *GetProcAddress* function with the value of the *HMODULE* or *HINSTANCE* and the name of the function as the parameters to get the memory address of the exported function and assign this memory address to a function pointer. Then the calling to this function pointer will actually invoke the function loaded from the DLL. Figure 4-3 is a code fragment (extracted from the source code of the MarkTool application) showing how to call the *gsapi_revision* function exported by *gsdll32.dll* in explicit linking.

```
//define structure that carries the version information of GhostScript
//declare Revision_data as an instance of this structure
struct gsapi_revision_t {
    . . .
} Revision_data;

//define a function prototype that takes a pointer of gsapi_revision_t
//and an int as peremeters, returns an int
//declare a function point variable revision of the defined function prototype
int  (__stdcall* revision ) (gsapi_revision_t *pr, int len);

//load gsdll32.dll by explicit linking
//with the filename of gsdll32.dll as the parameter of LoadLibrary
HINSTANCE gsdll=LoadLibrary("gsdll32.dll");

//get the memory address of gsapi_revision loaded from gsdll32.dll
//assign this memory address to the declared revision function pointer
revision = (int (__stdcall*)(gsapi_revision_t *,int))
                    GetProcAddress(gsdll,"gsapi_revision");

int len=sizeof(gsapi_revision_t);

//invoke the gsapi_revision function loaded from gsdll32.dll by calling revision
revision(&Revision_data,len);
```

Figure 4-3 Explicit linking in loading *gsdll32.dll*

In implicit linking or explicit linking with the filename of the DLL file as the parameter of the *LoadLibrary* function, an application will follow this search sequence to locate the DLL file:

1. The directory in which the executable file (the *.exe* file) is stored;
2. The directory of the application;
3. The Windows system directory (for example, *C:/WINDOWS/system* and *C:/WINDOWS/system32* for Windows XP);
4. The Windows directory (for example, *C:/WINDOWS* for Windows XP);
5. The directories listed in the *Path* environment variable of the operating system.

As shown in Figure 4-3, the MarkTool application uses explicit linking with the filename of *gsdll32.dll* as the parameter of the *LoadLibrary* function. When distributed to users, the *gsdll32.dll* file is packaged in the same directory in the *MarkTool.exe* file. This explicit linking may cause the MarkTool application to fail rendering PDF documents. As introduced earlier in Section 4.1.1, the GhostScript Interpreter API can only work properly when the whole set of the GhostScript software is installed. Furthermore, the call to the functions exported by *gsdll32.dll* will succeed only if the GhostScript of the same version as the DLL file is installed. For example, if the MarkTool application is distributed with the *gsdll32.dll* of version 8.11 while GhostScript 8.11 is not installed on the user's computer, the application will crash when the user tries to open a PDF file. This is illustrated in Figure 4-4.

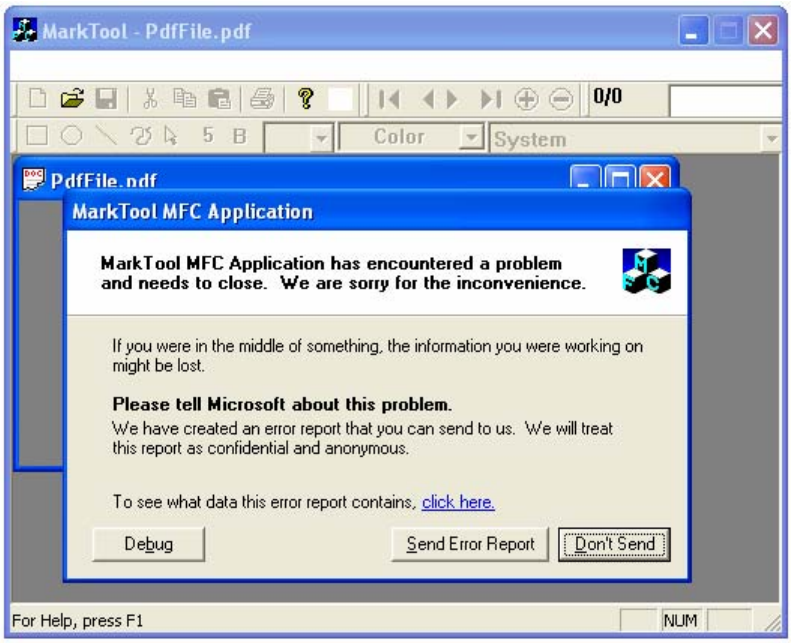


Figure 4-4 Crash of MarkTool due to different versions of *gsdll32.dll* and GhostScript

In the MARK application, the above problem is fixed using the following mechanism:

- When GhostScript is installed, the full paths of the *gsdll32.dll* and *gsdll32.lib* files are recorded in the Window registry (see Figure 4-5).
- The MARK application searches the Windows registry to find the full path of the *gsdll32.dll* file and loads it using explicit linking with the full pathname of *gsdll32.dll* as the parameter for the *LoadLibrary* function.

- If multiple sets of GhostScript of different versions are installed on one computer, the MARK application will choose the path of *gsdll32.dll* of the latest version.

Using this mechanism, it can be guaranteed that the MARK application will load the *gsdll32.dll* file with the same version as the installed GhostScript. The MARK application uses the WIN32 *RegOpenKeyEx* and *RegQueryValueEx* functions to open a specific key in the Windows registry and retrieve the value of the opened key respectively.

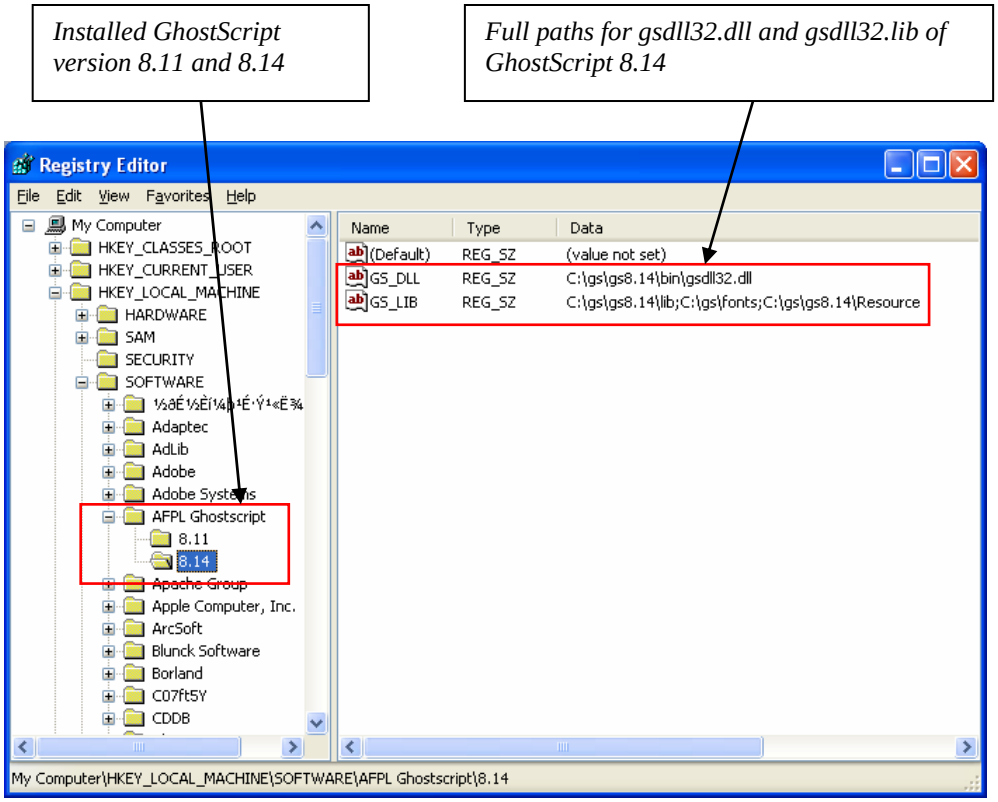


Figure 4-5 Windows registry entries for GhostScript

❖ Getting Bitmaps from PDF Documents

To get bitmaps (a bitmap for each individual page) for a PDF document, the MarkTool application calls the functions exported by *gsdll32.dll* in the following order:

1. Call the *gsapi_new_instance* function to create a new instance of the GhostScript engine. This instance will be used as the parameter for the following function calls.

2. Call the *gsapi_set_display_callback* function to provide the GhostScript engine with a set of callback functions. The GhostScript engine will call back these functions to pass results to the calling application when processing PDF documents. For example, a function which has the parameter list of *(void*, void*, int, int, int, int, char*)* should be provided. The GhostScript engine will pass the width and height (in pixels) of the PDF page, the stride of the bitmap (i.e. the count of bytes for each row in the bitmap), and the memory address of the raw data of the bitmap (i.e. an array of bytes) in the first three *int* parameters and the last pointer of *char* parameter respectively.
3. Call the *gsapi_init_with_args* function to initiate the GhostScript engine with some arguments, for example the resolution rate (in dots per inch [dpi]) and the name of the output device.
4. Call the *gsapi_run_string* function to invoke the GhostScript engine to create bitmaps. The *gsapi_run_string* function takes a string as the parameter. This string needs to specify some PostScript (2004) instructions that the GhostScript will follow to convert a PDF page to bitmap, including the path of the PDF document and which page(s) of the document need to be processed. PostScript is a programming language introduced by Adobe for printing graphics and text and is used internally by GhostScript to interpret PDF documents.
5. After a bitmap has been created, the GhostScript engine will call the callback functions provided as the parameter of the *gsapi_set_display_callback* function. The MarkTool application uses the values passed by these callback functions to create *CBitmap* objects that can be rendered as images to the user interface components of the application. *CBitmap* is an MFC class which provides various imaging functionalities, for example to create a bitmap from an array of bytes, to scale or rotate the bitmap, and to draw the bitmap to a specific user interface component.

There is one problem with this mechanism. A low-level PostScript command has to be specified as the string parameter of *gsapi_run_string* function, Figure 4-6 shows fragments of such a PostScript command (extracted from source code of the MarkTool application). This requires a good knowledge of the PostScript language and makes it difficult to debug the program. Furthermore, GhostScript of different versions may use different PostScript commands to interpret PDF. Therefore,

different string values need to be used as the parameter of the *gsapi_run_string* function for *gsdll32.dll* of different versions. For example, the MarkTool application based on *gsdll32.dll* version 8.11 does not work for *gsdll32.dll* versions 8.50 and 8.51.

```
. . .
. . . = ") (r) file pdfopen begin\n\
/FirstPage where { pop FirstPage } { 1 } ifelse\n\
/LastPage where { pop LastPage } { pdfpagecount } ifelse\n\
flush (%GSVIEW_PDF_PAGES: ) print exch =only ( ) print =only (\n) print flush\n\
currentglobal true setglobal\n";
. . .
. . . = "/.LockSafetyParams true\n\
>> setpagedevice\n\
setglobal\n";
. . .
. . . = " GSview_PDFpage\n\
Page pdfshowpage_init pdfshowpage_finish\n\
currentdict pdfclose\nend\nend\nend\n";
```

Figure 4-6 Part of the PostScript command used in MarkTool as the parameter of the *gsapi_run_string* function

This issue is addressed in the MARK application in the following way. When calling the *gsapi_init_with_args* function, MARK does not only specify some general arguments (like resolution rate and output device) that will be used by the GhostScript engine to interpret PDF, but also specifies the full pathname of the PDF document and the range of the pages that need to be interpreted. As the path of the PDF document and which pages need to be processed are provided to the GhostScript engine by the *gsapi_init_with_args* function, it is unnecessary to call the *gsapi_run_string* method with the low-level PostScript command as the parameter. Although using the GhostScript Interpreter API in this way is not included in the documentation of the API (GhostScript, 2004a), it should work for *gsdll32.dll* of all versions:

- In the Windows platform, users can invoke GhostScript (no matter which version it is) to display a PDF document by the console command of *gswin32 -dFirstPage=[page number] -dLastPage=[page number] [path of PDF document]* (*gswin32* is the filename of the executable file of GhostScript). For example, the command *gswin32 –dFirstPage=2 –LastPage=10 k:/book.pdf* will invoke GhostScript to display *k:/book.pdf* from page 2 to page 10 and *gswin32 k:/book.pdf* will display all pages of *k:/book.pdf*.
- When the *gsapi_init_with_args* function is called, *gsapi_init_with_args*

actually invokes the GhostScript executable file (i.e. *gswin32*) and passes the function parameters to the GhostScript executable file as command line arguments.

Figure 4-7 gives the code showing how to call the *gsapi_init_with_args* function in the MARK application. This code had been tested with *gsdll32.dll* from version 6.01 to version 8.51 and worked fine for all these versions.

```
char* arg_value[ 20 ];
int arg_count = 0;

//specify the values of some general arguments
arg_value[ arg_count++ ] = "not used";
arg_value[ arg_count++ ] = "-dTextAlphaBits=4";
. . .

//to render a specific page, the FirstPage and LastPage arguments
//should be assigned with the page number of this page
if( m_bReadOnePage ){
  char firstPage[ 64 ];
  sprintf( firstPage, "-dFirstPage=%d", m_iPageNumber );
  arg_value[ arg_count++ ] = firstPage;

  char lastPage[ 64 ];
  sprintf( lastPage, "-dLastPage=%d", m_iPageNumber );
  arg_value[ arg_count++ ] = lastPage;
}

//specify the filename
arg_value[ arg_count++ ] = filename;

//call the gsapi_init_with_args function exported by gsdll32.dll
//m_init_with_args is a function pointer pointing to gsapi_init_with_args
//m_instance is an instance created by gsapi_new_instance
m_init_with_args( m_instance, arg_count, arg_value );
```

Figure 4-7 Calling *gsapi_init_with_args* in MARK

❖ **Rendering Bitmaps**

As discussed earlier, both the MarkTool application and the MARK application use the GhostScript Interpreter API to get bitmaps (objects of MFC *CBitmap* class) from PDF documents (one bitmap for each individual PDF page). In principle, these two applications use the same mechanism to render these bitmaps as images to the user interface component. In both of these two applications, the user interface component that displays the bitmap image is an object of the MFC *CScrollView* class. To render a bitmap, they both first create an MFC *Graphics::Graphics* object from the *CScrollView* object, and then call the *DrawImage* function of the *Graphics::Graphics* object to draw the bitmap to the specific area of the *CScrollView* object. However, to

open a multiple-page PDF document, the MarkTool application retrieves the bitmaps for all pages and renders these bitmaps at once in the resolution rate selected by the users (for example 96dpi). Users can navigate through different pages by toolbar buttons and the vertical scrollbar (see Figure 4-8). The MARK application only retrieves and renders the bitmap of the first page at the resolution rate selected by the users. It retrieves the bitmaps of all the pages at a low resolution rate (12dpi) and renders these bitmaps as thumbnail images. Users can navigate through different pages by toolbar buttons and these thumbnail images. The MARK application will retrieve the bitmap of the selected page each time when the user switches to another page. This difference allows the MARK application to render a multiple-page PDF document faster than the MarkTool application since the creation of bitmaps with higher resolution rate takes longer time and consumes more memory resources. For example, on a 3GHz Pentium 4 CPU and 1GB RAM computer, it takes the MarkTool application 4-5 seconds to open a 13-page PDF document and the memory consumption increases by around 37 megabytes (from 12,648K to 49,876K as shown in Figure 4-9). It takes the MARK application only less than 2.5 seconds to open the same document and the memory consumption increases only by around 3 megabytes (from 11,620K to 14,448K). Although when the user switches from the current page to another page, MARK has to call the GhostScript Interpreter API to get the bitmap of the selected page, there is not a significant time delay during the swap between the two pages.

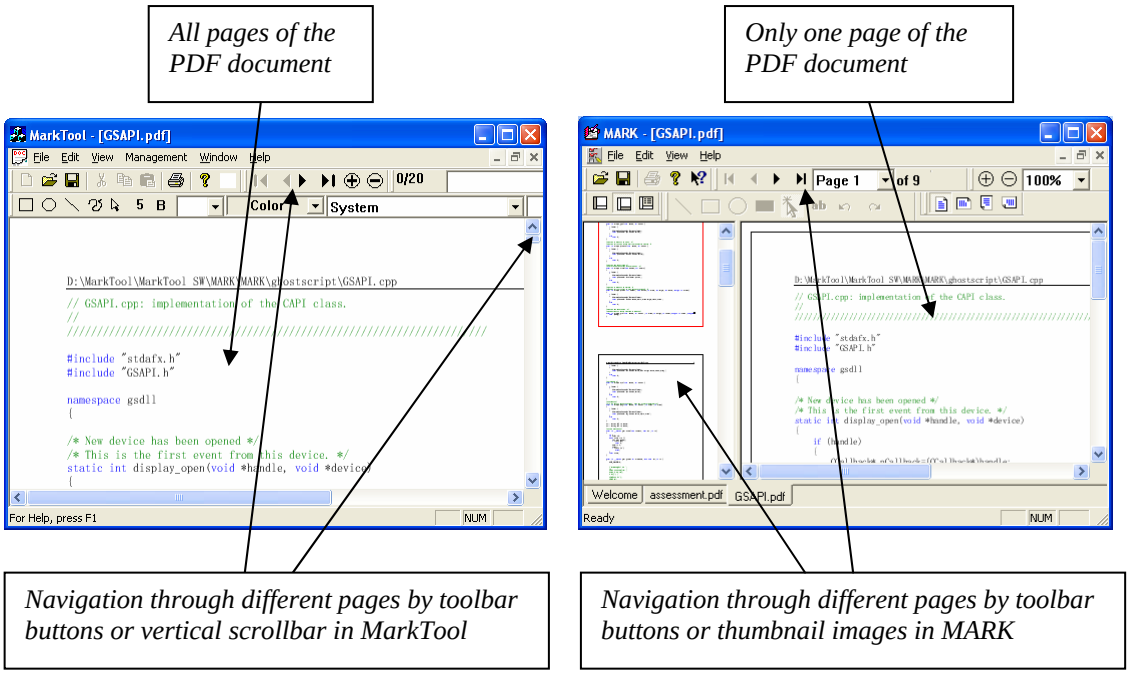


Figure 4-8 Navigation through pages in MarkTool and MARK

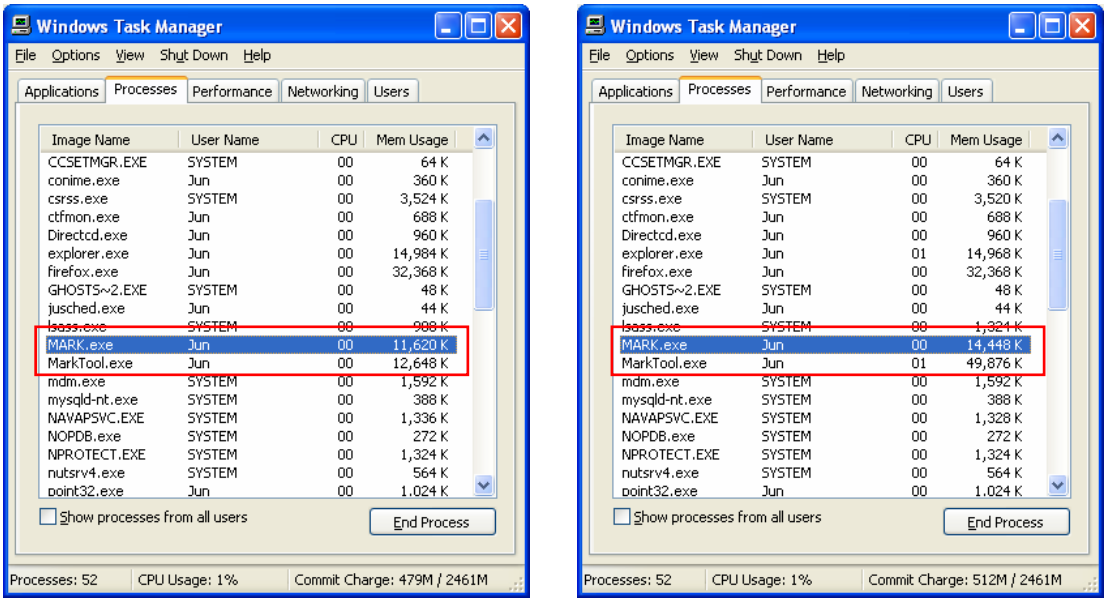


Figure 4-9 Memory consumption of MarkTool and MARK

4.1.3 Drawing on the Displayed PDF Page

To add annotations on the PDF document, users need to be able to draw graphics and texts on the surface of the displayed PDF page. The MarkTool application and the MARK application use the same mechanism to achieve this goal. First, they need to

get the position where the graphics or text should be drawn by capturing the Windows mouse messages (for example, *WM_LBUTTONDOWN*, *WM_MOUSEMOVE*, and *WM_LBUTTONUP* messages). Then they need to get the MFC *Graphics::Graphics* object of the *CScrollView* in which the PDF page is displayed. Finally, they draw the graphics and text to the specified position by calling the *DrawXXX* (for example, *DrawLine*, *DrawRectangle*, *DrawEllipse* and *DrawString*) functions of the *Graphics::Graphics* object.

In MarkTool, users can use different color for either the graphical or textual component of annotations and different font type and size for the textual component in order to categorize the annotations (see Figure 4-10). However, there is no way by which users can define what a specific color or font type stands for (for example, red means grammar error, green means conceptual error, and so on).

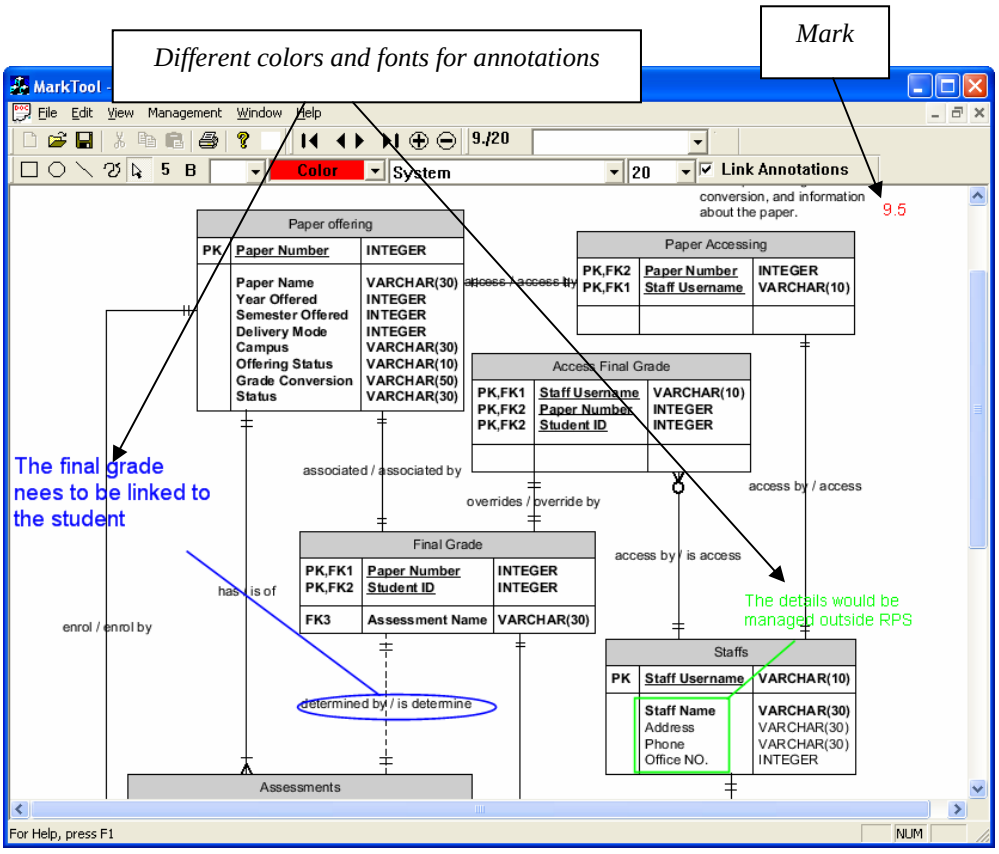


Figure 4-10 Using different colors and fonts in MarkTool

The MARK application offers a more comprehensive support for assessment criteria. For an assignment, users are able to create a marking scheme which includes the

maximum mark for the assignment and a set of assessment criteria. Each criterion is associated with a specific aspect of the assignment requirement and consists of 1) a short text describing which aspect the criterion is associated with, 2) a long text defining the detailed requirement for this aspect; and 3) a numeric value which is the maximum mark for this aspect. Users need to assign a unique color for each criterion. When inputting the textual comment, users also need to specify which criterion the comment is associated with. MARK will draw the textual comment with a title bar in the color that assigned to the selected criterion (see Figure 4-11). All the criteria are listed at the bottom part of user interface of MARK so that the meaning of a specific color can be recognized easily. The title bar is made half transparent by setting the *Alpha* field of the *Graphics::Color* object accordingly to prevent the content of the PDF from being hidden.

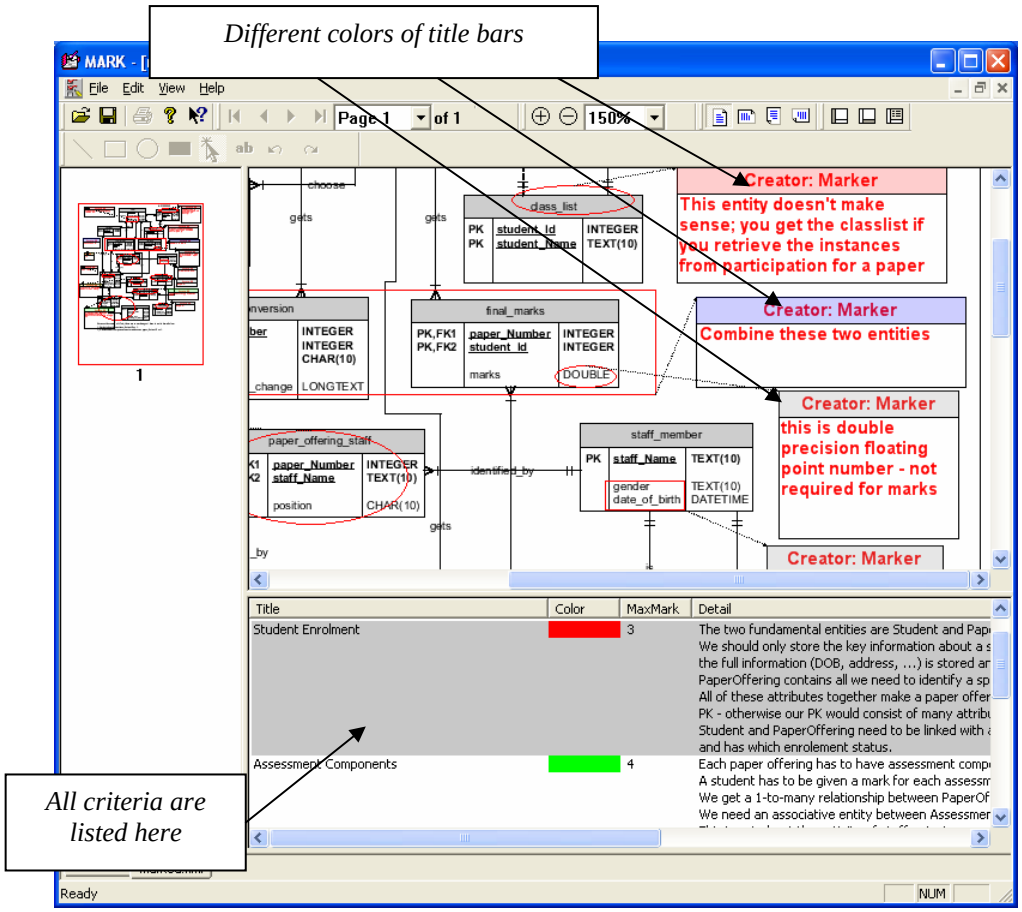


Figure 4-11 Assessment criteria in MARK

The MARK application creates a summary sheet for each PDF document and users can assignment numeric marks to the assignment in this summary sheet. This is different from the MarkTool application in which the numeric marks are drawn on the PDF page as shown in Figure 4-10. This summary sheet also lists all annotations added to the PDF document and allows users to create an overall comment on the assignment. The MARK application renders the summary sheet via an object of MFC *CFormView*. Users can switch the user interface of the MARK application between this *CFormView* object and the *CScrollView* object in which the PDF document is rendered by pressing one toolbar button. Figure 4-12 shows such a summary sheet.

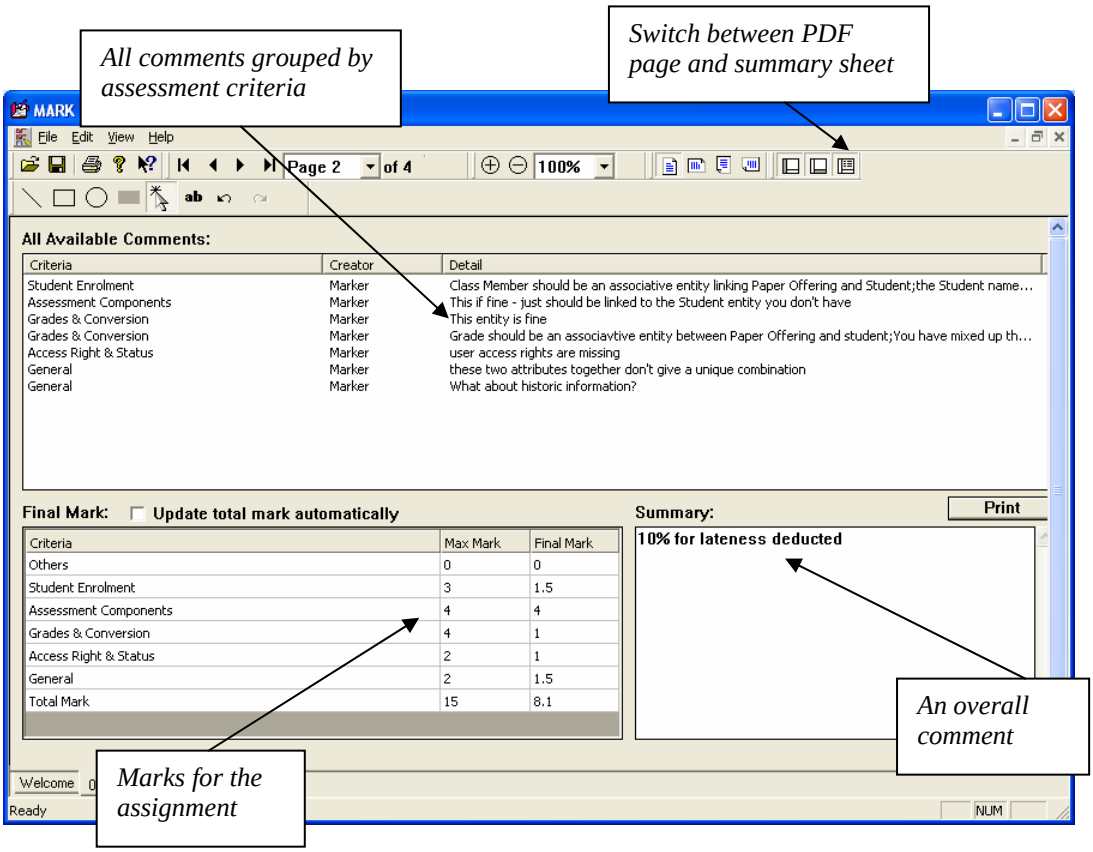


Figure 4-12 Summary sheet in MARK

4.1.4 Using XML for Data Storage

The MarkTool application stores data, including feedback for students (i.e. the annotations and numeric marks), in a single Microsoft Access database file. This data storage mechanism has two main drawbacks:

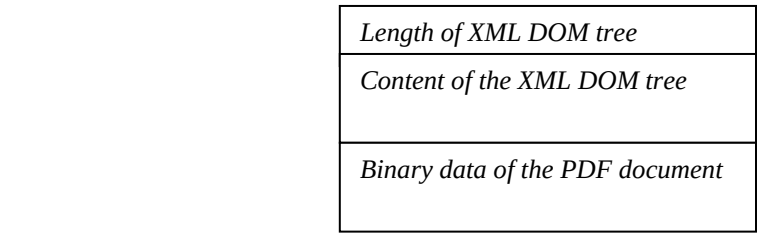
1. After a PDF document has been renamed or moved to another directory, the annotations added previously to it by MarkTool will be lost. This is because the association between annotations and PDF documents is maintained in a specific table within the Access file via the relationship between annotation IDs and the absolute pathnames of PDF documents.
2. The feedback for all students is stored in the same Access file. Therefore, to return a marked assignment to an individual student, the teacher has to use an additional tool to extract the feedback for the student from the Access file and create a new file to save the extracted feedback. The teacher then needs to send this file together with the PDF document to the student.

The MARK application implements the data storage in a different way. To save a marked assignment, it needs to go through the following steps:

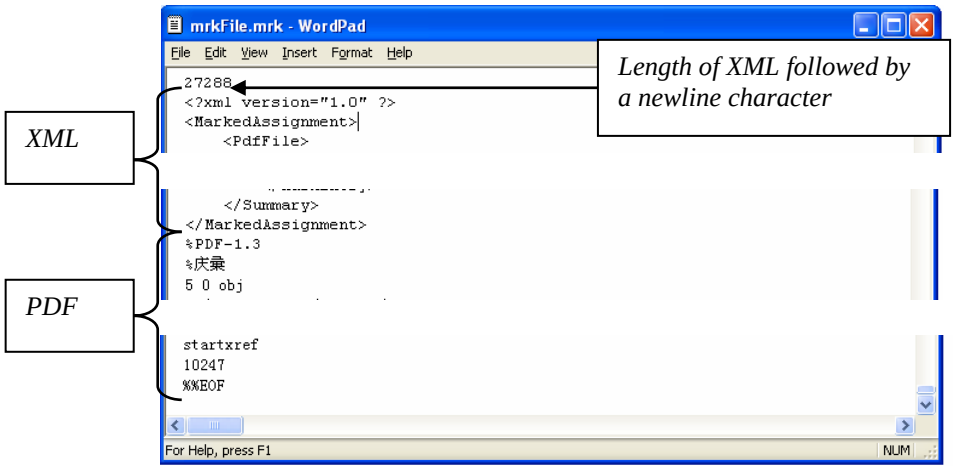
1. Create an in-memory XML Document Object Model (DOM, 2005) tree and store the data of assessment criteria used in the marking, the created annotations, the awarded marks and the overall comment in this DOM tree.
2. Calculate the length in bytes of the DOM tree.
3. Create a new file as the marked assignment document with file extension of *.mrk*.
4. Write the length of the DOM tree followed by a newline character (‘\n’), the content of the DOM tree followed by a newline character and the binary data of the PDF document into the created *.mrk* document (see Figure 4-13 for the structure of such an *.mrk* file).

When opening a marked assignment document (*.mrk* file), MARK first reads byte by byte until a newline character is reached and parses these bytes into an integer number. This number specifies the length of the following XML content as discussed previously. Then the MARK application reads the specified number of bytes into memory and parses these bytes into an XML DOM tree. Finally MARK creates an empty PDF document, copies the remaining part of the marked assignment document to this PDF document, and renders this PDF document along with the assessment criteria, annotations and marks stored in the XML DOM tree. The reason why MARK has to create a new PDF document is that the GhostScript Interpreter API does not accept memory stream data as input. Using this mechanism, the MARK application

solves the two data storage problems present in the MarkTool application. The annotations and marks for a PDF document will never be lost since they are stored together with the PDF document (as different components) in one *.mrk* file. When returning assignments, no extra effort is needed to extract feedback from one file and create another new file to save the extracted feedback. The use of XML to store annotations, marks and assessment criteria makes it possible for other applications to parse and use the data.



a) Structure of the marked assignment file



b) Fragments of a marked assignment file (opened with Microsoft WordPad)

Figure 4-13 Marked assignment (*.mrk*) file

The MARK application creates the in-memory DOM tree as a string and does not use any specific software API to produce such a string. It just adds the data of assessment criteria, annotations, and marks to this string according to the predefined Marked Assignment XML schema (see Appendix A for this XML schema). The XML API provided by Microsoft in its MSXML Software Development Kits (MSXML SDK, 2005) is used by the MARK application to load and parse the XML DOM tree. After MARK has read the XML part of an *.mrk* document into a string, it creates an

MSXML::IXMLDOMDocument object and calls the *LoadXML* function of the *IXMLDOMDocument* object to load the DOM tree from the string. The *LoadXML* function assigns the root element of the XML document to the *documentElement* property of the *IXMLDOMDocument* object. Then the MARK application uses the facilities provided by MSXML to iterate through the DOM tree and parse information of assessment criteria, annotations and so on from the DOM tree. For example, a pointer to the first child element of the root element can be retrieved from the *firstChild* property of the *IXMLDOMNode* object specified by *documentElement*.

4.2 The WebCTConnect Application

WebCTConnect is a desktop application that was developed as the LMS interaction module of the WAA system. It can interact with the WebCT server installed at Massey University using the HTTP protocol and mainly provides users with the following functionalities:

- Users can use the application to retrieve from the Massey WebCT server the list of all WebCT courses available to them using their Massey WebCT username and password.
- If the user's WebCT role for a specific course is teacher or teaching assistant, the application is capable of fetching from the WebCT server the lists of the students enrolled and the assignments for the course. For a teacher, WebCTConnect also can get the list of teaching assistants of the course from WebCT.
- For a specific assignment, users can use the application to download the assignments submitted by students from WebCT, provided their WebCT roles for the course are teacher or teaching assistant.
- Users can upload the marked assignments along with marks and comments for the whole class to WebCT in one operation instead of having to upload marked assignments one by one for each individual student.
- Teachers and teaching assistants can email each other and students via the WebCT internal email system.

A detailed introduction to the functionalities of the WebCTConnect application can be found in the application manual which is available from:

http://www-ist.massey.ac.nz/MarkTool/WebCTConnect1.1Jul20051.pdf
The content table of the WebCTConnect application manual is also attached with this thesis as Appendix B.

The following sections discuss the major software APIs used in the development of the WebCTConnect application.

4.2.1 Using HTTP to Interact with WebCT

The software API provided by WebCT in the WebCT PowerLink SDK can be used to develop applications that interact with WebCT (WebCT, 2004). However, by default this SDK is only delivered with WebCT Vista Edition. For the institutions that have only purchased WebCT Campus Edition, including Massey University, additional investment is required to get the WebCT PowerLink SDK. Therefore this project has to find a way by which the WebCTConnect application can interact with the WebCT server installed at Massey University without taking advantage of the WebCT PowerLink SDK.

❖ **Interaction between Web Browser and LMS**

As discussed in Section 3.3.1, the WAA system is actually the client side application of the LMS (WebCT installed at Massey University for this project) that it needs to interact with. Generally, the client applications of web-based LMS, such as WebCT and Blackboard, are standard web browsers, like Microsoft Internet Explorer (IE) or Mozilla Firefox. These web browsers interact with web-based LMS using the HTTP protocol. To retrieve data from the LMS, a web browser sends an HTTP request to the web server of the LMS. After the server has received the HTTP request, it parses the request to identify what information the web browser requires, packages the required information into an HTTP response, and returns this response to the web browser. When the web browser receives the HTTP response, it extracts the information from the response and renders it through the user interface or saves it on the local computer. A web browser also needs to include the user identification data in the HTTP request so that the LMS can verify whether the user has the access right to the required information. To upload data (for example, a file) to LMS, a web browser also needs to send an HTTP request to the LMS server with the data packaged in the request.

Therefore if WebCTConnect can send the same HTTP request to the Massey WebCT server as a web browser does, it can get the same HTTP response and extract data from the response. Also, to upload data to Massey WebCT, WebCTConnect only needs to include the data into the request.

To investigate what a web browser exactly sends to the Massey WebCT server, a specific tool, ieHTTPHeaders (2005), has been used in this project. ieHTTPHeaders is a free application that can be embedded into IE. It displays the headers and message body of every HTTP request sent from IE as well as the headers of the HTTP response received by IE. Figure 4-14 is the screen shot for using ieHTTPHeaders to analyze the HTTP requests and responses in editing the settings of a WebCT assignment.

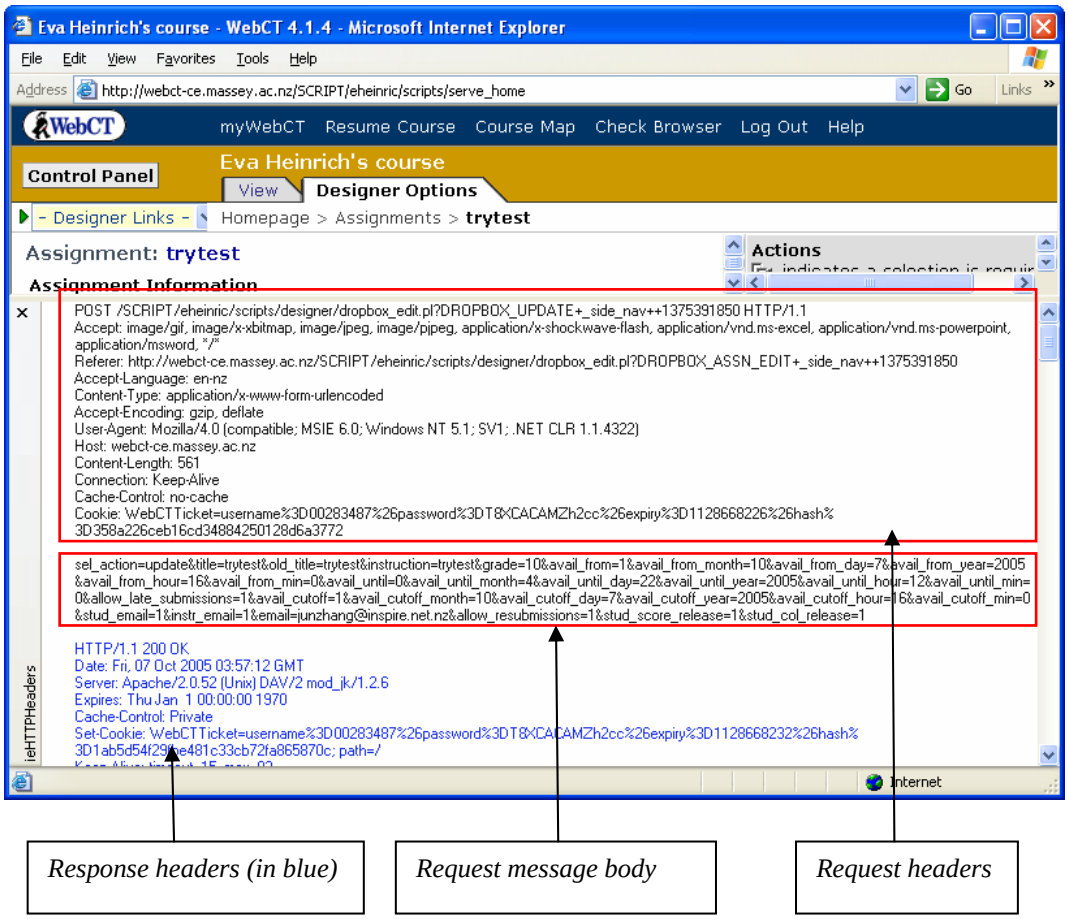


Figure 4-14 Using ieHTTPHeaders to analyse HTTP request and response

❖ **Sending HTTP Request to WebCT by WebCTConnect**

To make WebCTConnect to be able to send HTTP requests with the same content as revealed by ieHTTPHeaders, this project only needs to find a suitable software API that can be used by WebCTConnect to create and send such requests. There are a number of APIs that can be selected for this purpose, for example, the MFC WinInet (2005) classes, the .NET classes in the *System.Net* namespace (Troelsen, 2003), and the Java classes in the *java.net* package (Harold, 2004). Furthermore, all these APIs also provide developers the facilities for reading the headers and message body data in the HTTP responses returned from web servers. However, two special issues still need to be handled properly when creating such requests for WebCTConnect. One is how to include the user identification data in the request and the other is how to upload documents via an HTTP request.

To log into Massey WebCT using a web browser, first a user has to input his/her Massey WebCT username and password via the Massey WebCT login webpage. Then the web browser will send a *POST* request to the Massey WebCT server and include the username and password in the message body of the request. When the server receives this request, it parses the username and password from the request, validates the username and password against its database records, creates a *WebCTTicket* cookie, associates the value of this *WebCTTicket* cookie with the user's username in its database, and returns the *WebCTTicket* cookie in an HTTP response to the web browser. The web browser extracts the value of this *WebCTTicket* cookie from the response, stores it in memory, and adds it to the header of the next request for Massey WebCT. When the WebCT server detects a *WebCTTicket* cookie in an HTTP request, it maps this *WebCTTicket* cookie to the associated username and in turn creates the response accordingly. Therefore, WebCTConnect first needs to send a login *POST* request to the Massey WebCT server with user's Massey WebCT username and password in the message body. Then WebCTConnect needs to parse the *WebCTTicket* cookie from the response returned by the Massey WebCT server and include it in the following requests. However, the value of a *WebCTTicket* cookie will expire after a specific time span and each response returned from WebCT will carry a new value for the *WebCTTicket* cookie. This means that WebCTConnect has to extract the value of the *WebCTTicket* cookie from each HTTP response and use it in the subsequent request.

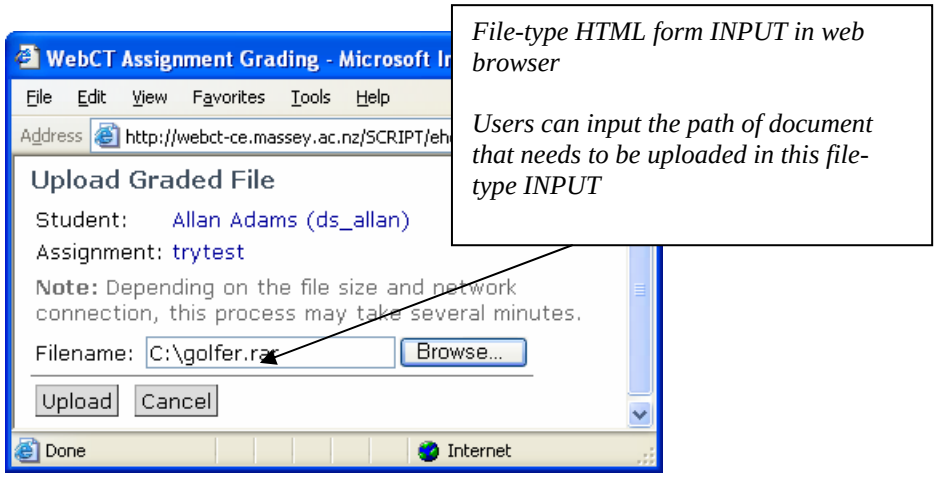
When using a web browser to upload a document to WebCT server, for example, to submit assignments, return marked assignments to students, or to send an email with an attachment via the WebCT internal email system, a user needs to input the local path of the document via an HTML file-type form *INPUT* (see Figure 4-15). The web browser will then send a multipart *POST* request to the WebCT server. The data of the document is included in the message body of the request in a special multipart format. As defined in RFC 2388 Returning Values from Forms: multipart/form-data (RFC 2388, 1998), the *Content-Type* header of a multipart *POST* request must be *multipart/form-data; boundary=[value of boundary]* and the message body contains a series of parts. These parts are separated from each other by a *boundary* line. A *boundary* line consists of two hyphens (--) followed by the value of *boundary* and is terminated by a pair of Carriage Return and Line Feed (CRLF) characters. The *boundary* can be defined as any value. However, it must be ensured that such a *boundary* does not occur in the data of any part. Each part represents an *INPUT* in the HTML form and consists of:

1. A *Content-Disposition* line with the format of *Content-Disposition: form-data; name="[the name of the HTML form INPUT]"*. For a file-type HTML form *INPUT*, this line also has a filename item.
2. An optional *Content-Type* line defining the media type of the part. For example, *Content-Type: image/gif* indicates that the content of the part is a gif image file. If this line is not provided, the type of the part is plaintext.
3. An optional *Content-Transfer-Encoding* line specifying the encoding type of the part. For example, *Content-Transfer-Encoding: 7bit* means the value of the part is encoded as 7bit ASCII characters.
4. The value of the part (the value of the form *INPUT* that needs to be posted). For a file-type HTML form *INPUT*, this value is the data of the document specified by the filename in the *INPUT*).

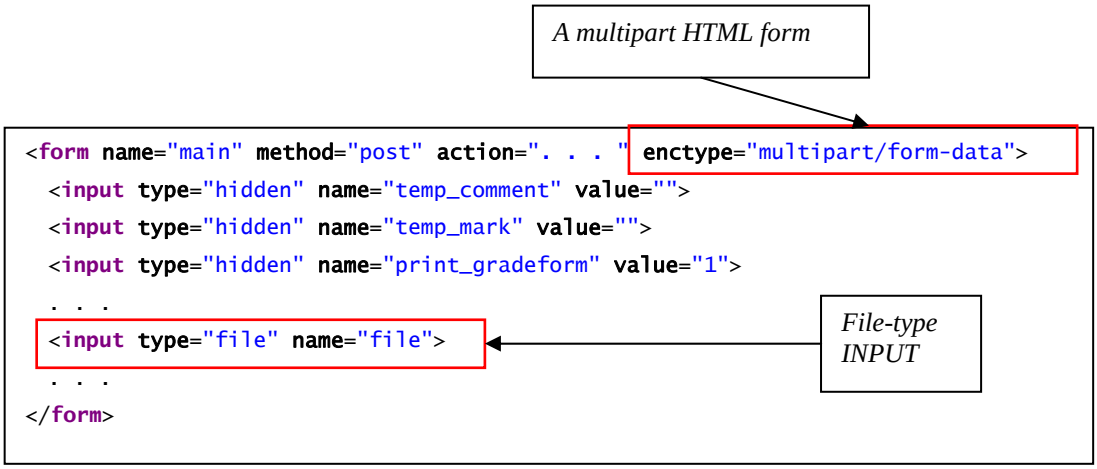
Figure 4-16 shows part of the content of the multipart *POST* request sent by IE after the user has pressed the Upload button in the WebCT webpage presented in Figure 4-15. This request contains four parts – three plaintext parts and an *application/pdf* part for uploading a PDF document.

Therefore to upload a document to Massey WebCT in a multipart *POST* request, WebCTConnect should be able to:

1. Generate the value of *boundary* used to separate parts and ensure that the value of *boundary* does not appear in any part.
2. Create the message body of the *POST* request following the format specified in RFC 2388 (1998).



a) A WebCT webpage from which users can upload a document as marked assignment



b) Part of the HTML source for the above WebCT webpage

Figure 4-15 Upload file using HTML file-type *INPUT*

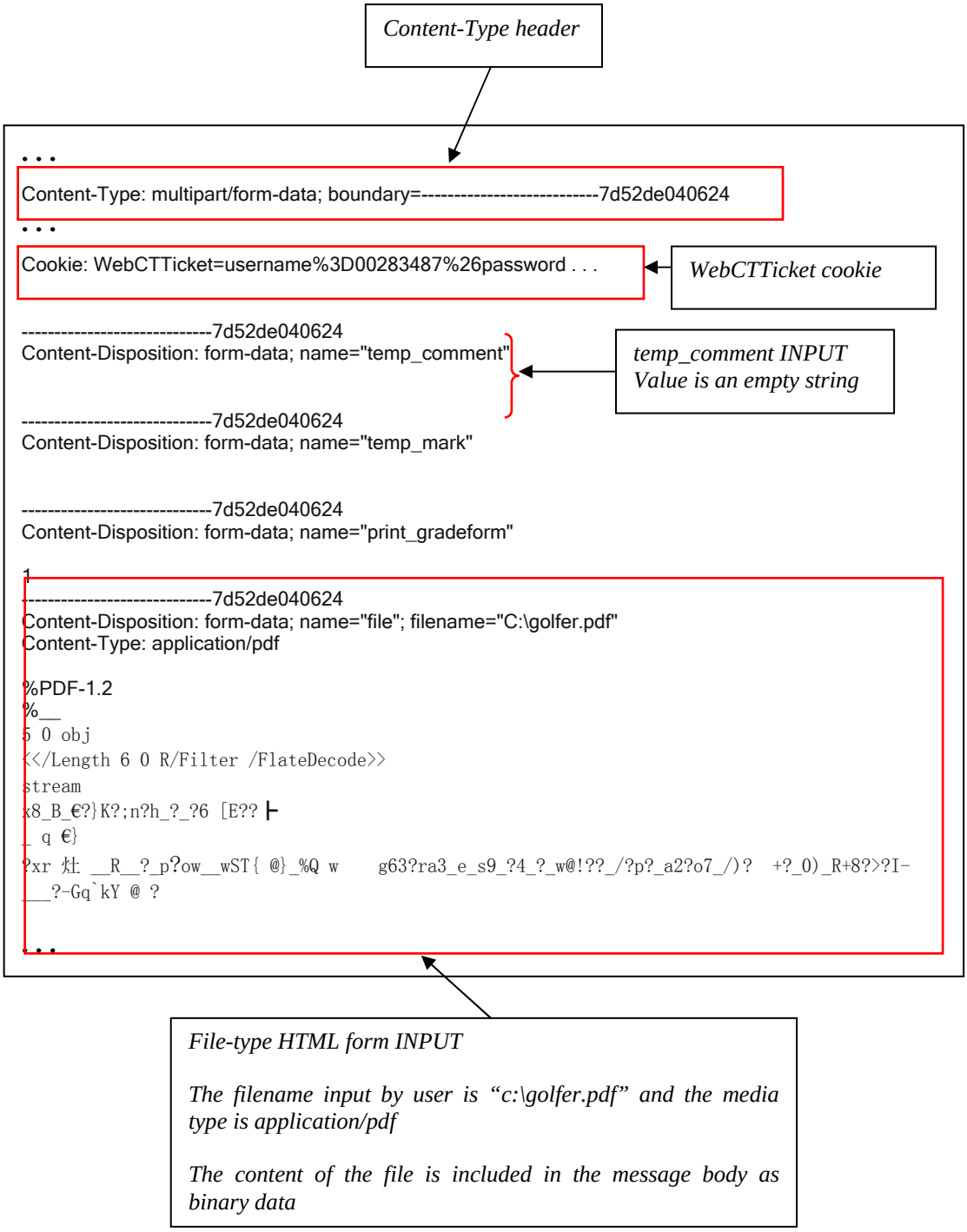


Figure 4-16 A multipart *POST* request

❖ Extracting Information from HTTP Response

After WebCTConnect has sent an HTTP request to Massey WebCT following the principles discussed earlier, it will receive an HTTP response from the Massey WebCT server. Then WebCTConnect needs to parse information from the response. It

has to extract the value of the *WebCTTicket* cookie from the header of the request since this value will be used in the consequent request. It also needs to parse the user-required information from the message body of the response. For example, if the user requires the list of students enrolled in a specific course, it needs to parse the student ID numbers and student names from the HTML document returned in the response message body. If the user downloads a document from WebCT, it should read the message body as an array of bytes and save this array of bytes into a local document.

4.2.2 API for HTTP Interaction with Web Server

A number of APIs can be used by WebCTConnect to create and send HTTP requests as well as to receive and parse data from HTTP responses, for example, the MFC WinInet (2005) classes, the .NET classes in the *System.Net* namespace (Troelsen, 2003), and the Java classes in the *java.net* package (Harold, 2004). However when using these language-built-in APIs to develop WebCTConnect, the *WebCTTicket* cookie has to be extracted from each HTTP request returned from the Massey WebCT server, stored in memory, and added into the next request. For example, in MFC WinInet, an HTTP response is encapsulated in an object of *CHttpFile*. To get the value of the *WebCTTicket* cookie has to take the following actions:

1. Call the *QueryInfo* function of the *CHttpFile* object. This function will return the entire response header as a multi-line string.
2. Search the multi-line string returned by the *QueryInfo* function until a line starting with "*Set-Cookie:*" is reached.
3. Extract the value of the *WebCTTicket* cookie from the string found in step 2.

After the value of the *WebCTTicket* cookie has been extracted, the static *SetCookie* function of the *CInternetSession* class must be called. Otherwise the value of the *WebCTTicket* cookie cannot be used by the next request. In addition to the requirement of the continued handling the *WebCTTicket* cookie, another disadvantage of using these APIs in the development of WebCTConnect is that none of them provides direct support for multipart *POST* HTTP requests. Developers have to write code by themselves to:

- Generate the value of *boundary* and ensure that this value does not appear in the data that will be posted.

- Create the message body following the format specified in RFC 2388 as introduced earlier in this section.

Compared with the above language-built-in HTTP APIs, the Java HttpClient library developed by Apache (HttpClient, 2005) makes the development of WebCTConnect much easier. This HttpClient library not only can be used to send HTTP request and receive HTTP response but also provides automatic cookie handling facilities and the direct support for multipart *POST* request.

The core of the Apache HttpClient library is an *HttpClient* class. An HTTP request is sent to the web server by calling the *executeMethod* method of an *HttpClient* object. When an HTTP response is returned from the web server, the *HttpClient* object will automatically extract cookies from the response, save the cookies in memory, and send the cookies back to the web server in the next execution of the *executeMethod* method. An object of *GetMethod* or *PostMethod* can be used as the argument of the *executeMethod* method for a *GET* or *POST* request. The request headers can be set by calling the *addRequestHeaders* method of the *GetMethod* or *PostMethod* object. However, most times it is not necessary to call this *addRequestHeaders* method. When a *GetMethod* or *PostMethod* object is constructed some default header values are also created. For a non-multipart *POST* request, the message body can be created by calling the *setParameter* method of the *PostMethod* object. Such a *setParameter* method takes two string arguments which should be the name and value of an HTML form *INPUT* if the request is to post HTML form data. A multipart *POST* request can be created by the following steps without worrying about the value of *boundary* and the detailed format of the message body:

1. Create a *StringPart* object for each non-file part by calling the constructor of *StringPart*.
2. Create a *FilePart* object for each file part (i.e. the document that needs to be uploaded) by calling the constructor of *FilePart*.
3. Create a *MultipartRequestEntity* object from the *StringPart* and *FilePart* objects by calling the constructor of *MultipartRequestEntity*.
4. Calling the *setRequestEntity* method of the *PostMethod* object using the *MultipartRequestEntity* object as argument.

Figure 4-17 shows the full code for using the HttpClient library to upload a marked assignment to Massey WebCT (the corresponding WebCT webpage is presented in Figure 4-15).

```
//upload a file as marked assignment document
//url is the URL address for uploading marked assignment
//file is the document that needs to be uploaded
bool uploadMarkedAssignment( string url, File file )
{
   //create an HttpClient object
  HttpClient client = new HttpClient();

  //create a POST request
  PostMethod post = new PostMethod(url);

  //create parts
  Part[] parts = new Part[ 4 ];
  parts[ 0 ] = new StringPart( "temp_comment", "" );
  parts[ 1 ] = new StringPart( "temp_mark", "" );
  parts[ 2 ] = new StringPart( "print_gradeform", "1" );

  //FilePart for uploading file
  parts[3] = new FilePart("file", file);

  //set message body of multipart POST request
  post.setRequestEntity( new MultipartRequestEntity(parts, post.getParams()) );

  try{
    //send request
    client.executeMethod( post );

    post.releaseConnection();
    return true;
  }catch( Exception e ){
    return false;
  }
}
```

Figure 4-17 Using Apache HttpClient to upload marked assignment

Using HttpClient to read data from HTTP response is also easy. Calling the *getResponseBodyAsString* method of the *GetMethod* or *PostMethod* object will get a string representative of the response message body. This can be used by WebCTConnect when an HTML document is returned from Massey WebCT. The *getResponseBodyAsStream* method will return a file IO stream from which the binary data of the response message body can be read. This should be used by WebCTConnect when downloading documents from Massey WebCT.

In addition to the automatic cookie handling and direct support for multipart *POST* requests, Apache HttpClient also encapsulates the implementation for supporting

other HTTP communication functionalities, for example, the automatic handling of redirection of HTTP responses and connection management for using HTTP connections concurrently by multi-threads. Therefore this project decided to use Apache HttpClient in the development of WebCTConnect.

4.3 The ToPDF Application

ToPDF is a document conversion application for the Windows platform. ToPDF can convert a printable document (the document that can be printed out from a Windows application) to PDF format. For example, ToPDF can convert either any Microsoft Excel Spreadsheet documents (*.xls* files) or Comma Separated Value documents (*.csv* files) to PDF format as all these documents can be printed from Microsoft Excel. However, ToPDF is not able to convert a Microsoft Access document (*.mdb* file) to PDF format since an *.mdb* file cannot be printed by any Windows application.

The feature that distinguishes ToPDF from other PDF conversion applications (such as Adobe Acrobat Professional [Adobe, 2005], PDFCreator [2004], or PrimoPDF[2005]) is that ToPDF allows folder-based conversion. After setting some simple output rules in ToPDF (see Figure 4-18) and specifying the path of a folder, ToPDF will automatically convert all the printable documents in the folder to PDF format without any user interaction. As the MARK application requires student assignments to be converted to PDF format, this feature will save users a great amount of time in such document conversion. Otherwise, users have to convert the assignments to PDF one document by one document and take a series of actions to convert each individual document.

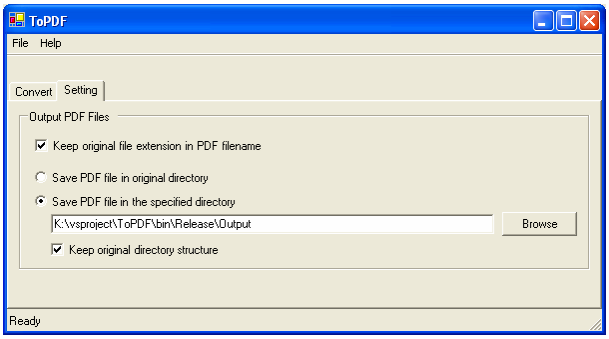


Figure 4-18 Setting output rules in ToPDF

The following sections discuss the mechanisms used in ToPDF to convert printable documents to PDF format and how these mechanisms are implemented.

4.3.1 Converting a Printable Document to PDF Format

Any printable document can be converted to PDF format in the following two steps:

1. Print the document to a PostScript file using a PostScript compatible printer.
2. Convert the PostScript document to a PDF document.

The Microsoft Windows Operating System (OS) is released with many PostScript compatible printer drivers, for example, IBM InfoPrint 32 PS driver and HP Color LaserJet 8500 PS driver. Adobe also provides a free-of-charge universal PostScript printer driver for Windows which is available from:
http://adobe.com/support/downloads/product.jsp?product=44&platform=Windows
Any of these printer drivers can be used to add a PostScript printer to Windows OS. To print a document to a PostScript file instead of paper, the *Port* variable of the printer has to be set as a file port specified by a filename. Figure 4-19 shows how to set such a *Port* when using *Add Printer Wizard* to add a printer to Windows OS.

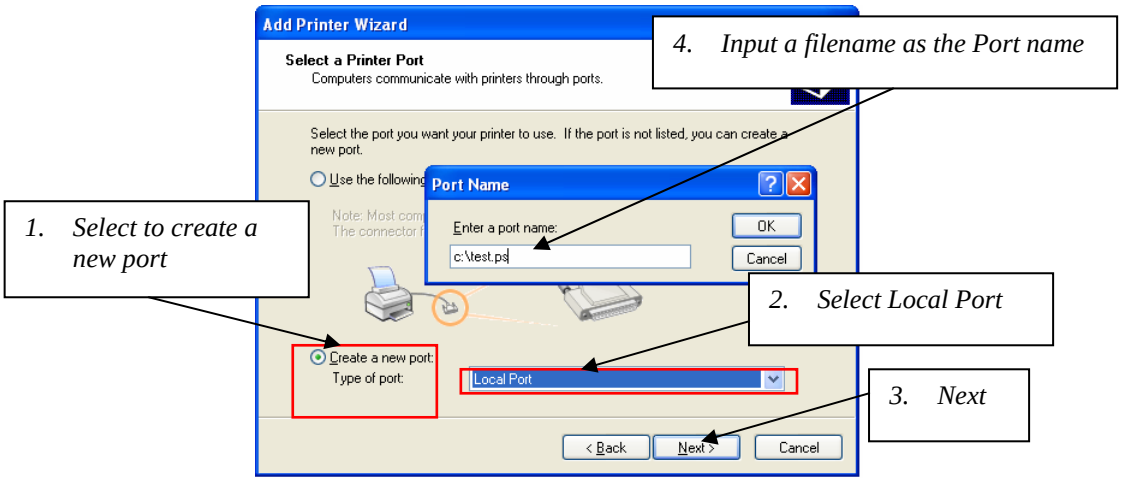


Figure 4-19 *Add Printer Wizard* of Windows

4.3.2 Implementation of ToPDF

To be able to convert a printable document to PDF format, ToPDF only needs to programmatically execute the two steps specified in the previous section. For multiple documents, for example, the documents within a folder, ToPDF just needs to apply these two steps to each individual document. An output rule is necessary which defines how to name the created PDF document so that ToPDF can convert the PostScript file to a PDF document without users having to input the PDF filename.

Therefore ToPDF needs to be able to:

- Create a local printer file port.
- Programmatically add a printer and set the *Port* variable of the printer without invoking the *Add Printer Wizard*.
- Automatically print a document using a specific printer.
- Invoke GhostScript or call the GhostScript Interpreter API to convert a PostScript file to PDF format.

The Win32 *AddPortExA* or *AddPortExW* function in the *winspool.drv* file (which is a Windows system library file) can be used to add a local printer port to Windows and point the port to a specific filename.

The Win32 *PrintUIEntry* function in the *printui.dll* file (which is a Windows system library file) can install a printer silently without any user interactions. The driver, port, and name of the printer need to be specified by setting the arguments of the *PrintUIEntry* function.

The Win32 *ShellExecute* function can programmatically print a document when the filename and print action are provided as arguments. The .NET *system.diagnostics.Process* class also provides the same functionality. To print to a specific printer, one possible solution is using the *SetDefaultPrinter* Win32 function to set the system default printer to this printer.

The *ShellExecute* function and the .NET *system.diagnostics.Process* class also can be used to invoke GhostScript to convert a PostScript document to PDF format. However,

such PostScript to PDF conversion can be achieved by calling the functions exported by GhostScript Interpreter API as well (see Section 4.1.2 for how to call these GhostScript Interpreter API functions).

As a number of Win32 functions need to be called to convert a printable document to PDF format, this project selected C# (one of the .NET languages) as the development language for the ToPDF application. A C# application can easily import Win32 functions by using the .NET Platform Invoke (*pInvoke*) services and call these functions as normal C# methods (Jones, 2003). MFC C++ would be another good choice. However, the explicit memory management of C++, for example, to manually allocate and free memory resources for any C++ objects, is a big disadvantage compared with the automatic garbage collection featured by the .NET languages.

4.4 Summary

This chapter has discussed the key software APIs used in the development of the MARK application, the WebCTConnect application and the ToPDF application of the WAA system.

The MARK application is an onscreen marking tool developed using the C/C++ language. The key software used in MARK is the GhostScript Interpreter API to convert a PDF document to bitmap images.

WebCTConnect is a Java application capable of interacting with the WebCT server installed at Massey University using the HTTP protocol. The core software API used in the development of WebCTConnect is the Apache HttpClient library.

ToPDF is document conversion application which can convert printable documents to PDF format. It was developed using C#. ToPDF uses several Win32 printing-related functions to add a PostScript compatible printer and print documents to PostScript files using this printer. ToPDF also uses GhostScript to convert the printed PostScript files to PDF format.

Chapter 5 Evaluation of the WAA System

As discussed in Chapter 3, this project has developed three principles for using computers to facilitate formative assessment of assignments:

1. It needs to facilitate all the activities that are potentially required for formative assessment of assignments.
2. It needs to provide an onscreen marking tool which enables human markers to mark assignments in an intuitive and efficient way by replicating their paper-based approaches.
3. It needs to provide a generic solution for facilitating formative assessment of assignments from all disciplines, not a limited solution restricted to some specific domains.

This project has implemented these principles in the WAA system as three separate applications – the MARK application, the WebCTConnect application, and the ToPDF application.

The WAA system has been already used in the assignment marking of several courses at Massey University, for example, the computer science courses of 159202 Declarative Programming (Semester II 2005), 159254 Software Engineering A (Semester II 2004 and Semester II 2005), and 159351 Software Engineering B (Semester I 2005).

This chapter presents some informal evaluations of the WAA system.

5.1 Implementing the Principles for Using Computers to Facilitate Formative Assessment of Assignment via the WAA System

This section evaluates whether the principles for using computers to facilitate formative assessment of assignments have been implemented in the WAA system.

5.1.1 Support for Activities in Assignment Assessment Life Cycle

As discussed in Section 3.1.1, computers should be used to support not only the marking activity, but also all the other activities involved in the formative assessment of assignments – i.e. all the activities included in an assignment assessment life cycle as illustrated in Figure 3-1, for example, the setting up of assignment tasks and assessment criteria, the marking of assignments, and the analysis of assessment results.

This requirement has been successfully fulfilled in the WAA system:

- Teachers can use the MARK application to create assignment task and assessment criteria. Then teachers can publish the tasks and criteria to students and tutors by uploading them to the Massey WebCT server using WebCTConnect.
- The WAA system takes advantage of WebCT to manage assignment submissions – students use their web browsers to submit assignments to Massey WebCT.
- Teachers and tutors can use WebCTConnect to download student submissions from the Massey WebCT server.
- Teachers, tutors, and students can use the MARK application to mark assignments (they need to convert the non-PDF documents to the PDF format).
- Teachers and tutors can returned the marked assignments to students by uploading the marked assignments to the Massey WebCT server using WebCTConnect.
- Students can use the MARK application to read the feedback given by their teachers or tutors.
- Teachers, tutors, and students can use WebCTConnect to communicate with each other through the WebCT internal email system.

The WAA system does not only simply use WebCT to manage assignment submissions and return marked assignments, but also provides a more efficient way to use the WebCT services. For example, to return the marked assignments for the whole class, teachers only need to specify the path of the folder in which all the marked assignments are stored. WebCTConnect automatically returns these marked assignments to the proper students by uploading them to Massey WebCT. If the

assignments have been marked using the MARK application, WebCTConnect will also extract the marks from the marked assignment documents (*.mrk* files) and upload the marks to Massey WebCT. Without the help of the WAA system, returning marked assignments and marks to students via WebCT is time consuming and prone to human error – teachers have to 1) upload the marked assignments document by document and student by student and 2) manually input the mark for each individual student via a WebCT webpage.

5.1.2 Onscreen Marking Tool

The MARK application of the WAA system is an onscreen marking tool that enables human markers to mark assignments in an intuitive and efficient way by replicating their paper-based approaches. Users can easily add free-text annotations as formative feedback to assignment documents (in the PDF format) leaving the original formatting of the documents intact. Annotations are added in similar ways as in the paper-based assessment – drawing a shape to indicate which part of the assignment the annotation refers to and writing a textual comment which carries the main message of the annotation. MARK also allows users to award numeric marks to assignments as summative feedback. Furthermore, users can associate both the textual annotations and the numeric marks with the corresponding assessment criteria.

5.1.3 A Generic Solution

As discussed in Section 3.1.3, a generic solution for using computers to facilitate formative assessment of assignment requires that the WAA system needs to be able to handle all assignments submitted electronically by students. From the technical perspective, it means that the WAA system must be capable of dealing with all printable document types that can be used by students to complete and submit their assignments electronically. This principle has been successfully implemented in the WAA system – the system enables users to assess assignments submitted electronically by students covering all printable document types:

- Users can use the ToPDF application to convert all printable documents to the PDF format.

- Then users can use the MARK application to assess the assignments (in the PDF format) by adding free-text annotations and awarding numeric marks to the PDF documents.

5.2 Overview of Uses of the WAA System at Massey University

The WAA system has been used in the assignment marking of several courses at Massey University. Table 5-1 lists the computer science assignments marked using the WAA system. The WAA system is available from the website of this project:
http://www-ist.massey.ac.nz/MarkTool
Some lecturers outside the computer science department have downloaded the system and used it in their assignment marking. However the detailed data of their marking has not been collected.

Table 5-1 Computer science assignments marked using the WAA system

Assignment	Document Types	Number of Submissions	Group Assignment	Used Applications
Assignment 1 159202 Declarative Programming Semester II, 2005	Haskell programming language files	37	No	WebCTConnect
Assignment 2 159202 Declarative Programming Semester II, 2005	Prolog programming language files	35	No	WebCTConnect
Assignment 1 159254 Software Engineering A Semester II, 2004	Microsoft Word or Visio documents	49	Yes	WebCTConnect ToPDF MARK
Assignment 1 159254 Software Engineering A Semester II, 2005	Microsoft Word or Visio documents	46	Yes	WebCTConnect ToPDF MARK
Assignment 1 159351 Software Engineering B Semester I, 2005	Microsoft Word or Project documents	43	Yes	WebCTConnect ToPDF MARK
Assignment 2 159772 Multimedia Systems and Computer Based Learning Semester II, 2005	Microsoft Word document	10	No	WebCTConnect ToPDF MARK

For the assignments listed in Table 5-1, students needed to submit their assignments to Massey WebCT through WebCT webpages using web browsers. Teachers or tutors

used WebCTConnect to download the submitted assignment from Massey WebCT. In the assignments of 159254, 159351 and 159772, teachers and tutors also used the ToPDF application and the MARK application to convert the downloaded assignment documents to the PDF format and mark the PDF assignment documents respectively. In the assignments of 159202, as students submitted programming language files, teachers and tutors needed to mark the assignments by running the submitted files using the corresponding applications. For all these assignments, teachers and tutors returned the marked assignments and marks to students using WebCTConnect by uploading the marked assignment documents and marks to Massey WebCT.

This project has organized several informal interviews with the teachers and tutors who were involved in using the WAA system to mark the assignments listed in Table 5-1. Most of the collected feedback is positive, for example:

- The WAA system provides a more effective way in using the university's current LMS in assignment marking.
- The WAA system saves teachers and tutors a significant amount of time in marking assignments.
- The marking of a large amount of assignments is well managed.
- Using the WAA system in marking assignments is more convenient than the traditional paper-and-pen method.

Some advice for the improvement of the system was also suggested by teachers and tutors:

- Students have to use the MARK application to view the marked assignment documents (*.mrk* files). It would be better if the marked assignment can be stored in a normal PDF document. Students could then use any PDF reader application to view the marked assignments.
- A text-searching functionality should be added into the MARK application so that a specific part of the assignment can be found quickly by searching for some keywords.

Chapter 6 Conclusion and Future Work

This chapter concludes this thesis and consists of two parts. Section 6.1 summaries the contributions of the research project presented in this thesis and Section 6.2 identifies the future work for the continuation of this research project.

6.1 Contributions

Based on the literature review of educational assessment theories and current computer software technologies, this project suggests three principles for using computers to facilitate formative assessment of assignments:

- It needs to provide a generic solution for facilitating formative assessment of assignments from all disciplines, not a limited solution restricted to some specific domains.
- It needs to facilitate all the activities that are included in the assignment assessment life cycle. Assignment assessment life cycle is a new concept developed by this project. It covers all the activities that are potentially necessary for the assessment of an assignment.
- It needs to provide an onscreen marking tool which enables human markers to mark assignments in an intuitive and efficient way by replicating their paper-based approaches.

According to these three principles, this project has developed a set of requirements for the software system that intends to facilitate formative assessment of assignments. These requirements can be collated into three groups:

- The provision of the functionalities which are necessary from the view point of users (including teachers, markers, and students) for facilitating formative assessment of assignments, for example, the functionalities to set up assessment criteria, to submit assignments, to mark assignments, and to return the marked assignments.
- The provision of the electronic document exchange and communication channels among teachers, markers, and students.

- The provision of management functionalities to ensure the efficiency, reliability, and security of the system.

Based on these requirements, this project has designed the WAA system to implement the principles for using computers to facilitate formative assessment of assignments:

- The WAA system takes advantage of LMS (such as WebCT or Blackboard) installed at universities to manage assignments and student submissions. It also uses the LMS services as the bridge for the communication and document exchange among different users (teachers, markers and students).
- The WAA system provides users, via applications installed on users' local computers, with the functionalities that are necessary for using computers to facilitate formative assessment from the user's view point. The core of these applications is an onscreen marking tool. This onscreen marking tool enables users to mark assignments in an intuitive and effective way by replicating the paper-based assessment approaches. These applications are also capable of interacting with LMS via computer networks.
- To provide a generic solution for facilitating formative assessment of assignments, the onscreen marking tool of the WAA system uses the PDF format to render assignment documents and the WAA system is able to convert all printable documents to the PDF format.

This project has partially implemented the specified WAA system by three separate applications – the WebCTConnect application which can interacts with Massey WebCT via the HTTP protocol, the MARK application which is an onscreen marking tool, and the ToPDF application which is able to convert all printable documents to the PDF format. These applications are available for downloading from this project website (*http://www-ist.massey.ac.nz/MarkTool*) and have been used in the assignment marking of several courses at Massey University.

6.2 Future Work

A number of topics for future research have arisen from the work on this thesis. These topics mainly fall into two categories. First is the further implementation of the WAA system and second is the formal evaluation of the WAA system.

Currently, the WAA system is only partially implemented. Some functionalities that are necessary from user's view point for facilitate formative assessment of assignments are not implemented in the system. Future work is needed to add these functionalities to the WAA system, for example:

- The MARK application needs to provide users with a text-searching functionality to help human markers to search student assignments with some particular keywords or phrases.
- Currently, the MARK application only provides a very limited "analyzing" function for quality-control of feedback and analysis of assessment results by simply extracting text comments from the marked assignment documents and listing these comments in the order of assessment criteria. More analyzing functionalities need to be added. For example, using the Microsoft Word Object Library (2005) in the MARK application would help users to check for spelling and grammar errors. Statistical data based on the numeric marks extracted from the marked assignment documents would help teachers to quickly grasp the strengths and weaknesses of the whole class.

Furthermore, the MARK application is only able to save the marked assignments as *.mrk* documents. To view such *.mrk* documents, students have to use the MARK application. The MARK application needs to be further implemented with the ability to store the marked assignments in normal PDF documents so that students can view the marked assignment using standard PDF reader applications.

The WAA system has been used in the assignment marking of several courses at Massey University. However this project has not carried out a formal evaluation on the effectiveness of the system. Further work is needed in this area. More teachers and tutors from different departments need to be involved in using the WAA system in marking their assignments. A formal survey needs to be designed and undertaken to collect the feedback on the effectiveness of the WAA system.

References

Adobe (2004a). PDF Library Software Development Kit (SDK). Retrieved 10/10/2004, from http://partners.adobe.com/public/developer/pdf/library/index.html.

Adobe (2004b). Adobe Acrobat Viewer – Download. Retrieved 01/10/2004, from http://www.adobe.com/products/acrviewer/acrvdnld.html.

Adobe (2005). Adobe Acrobat 7.0 Professional. Retrieved 01/07/2005, from http://www.adobe.com/products/acrobatpro/main.html.

Atkinson, T., & Davies, G. (2005). Computer Aided Assessment (CAA) and Language Learning. Retrieved 02/06/2005, from http://www.ict4lt.org/en/en_mod4-1.htm.

Anastasi, A. (1988). *Psychological Testing*. New York: MacMillan.

ARG (1989). Assessment Reform Group. Retrieved 11/05/2005, from http://arg.educ.cam.ac.uk/.

ARG (1999). *Assessment for Learning: Beyond the Black Box*. Cambridge, UK: University of Cambridge School of Education.

Ashcroft, K., & Palacio, D. (1996). *Researching into Assessment and Evaluation in Colleges and Universities*. London: Kogan Page.

AUT (2003). Guidelines on the Policy for the Assessment of Student Achievement. Retrieved 25/10/2005, from http://www.aut.ac.nz/staff/academic_quality_office/policies_guidelines_notes.htm.

Baig, V., & Pierre, W. (2004). English 255 Introductory Composition – Marking Scheme. Retrieved 11/11/2004, from http://www.athabascau.ca/courses/engl/255/marking.html.

Baillie-de Byl, P. (2004). An Online Assistant for Remote, Distributed Critiquing of Electronically Submitted Assessment. *Educational Technology & Society*, 7(1), 29-41.
Bennett, S., Skelton, J. (2001). *UML*. London: McGraw-Hill.

Biggs, J. (1999). *Teaching for quality learning at university*. Philadelphia: SRHE & Open University Press.

Black, P. (1993). Formative and Summative Assessment by Teachers. *Studies in Science*, 21, 49-97.

Black, P. (1997). *Testing: Friend or Foe? Theory and Practice of Assessment and Testing*. London: Falmer.

Black, P., & Wiliam, D. (1998a). Assessment and Classroom Learning. *Assessment in Education: Principles, Policies & Practice*, 5(1), 7-74.

Black, P., & Wiliam, D. (1998b) Inside the Black Box: Raising Standards through Classroom Assessment. *Phi Delta Kappan*, 80(2), 139-149.

Blackboard (2004). blackboard.com. Retrieved 28/10/2004, from http://www.balckboard.com.

Bond, L. A. (1996). Norm- and Criterion-Referenced Testing. *Practical Assessment, Research & Evaluation*, Vol. 5. Retrieved 11/11/2004, from http://pareonline.net/getvn.asp?v=5&n=2.

Boud, D. (1995). *Enhancing Learning through Self Assessment*. London: Kogan Page.

Brookhart, S.M. (2001). Successful Students' Formative and Summative Uses of Assessment Information. *Assessment in Education*, 8(2), 153-169.

Brown, G., Bull, J., & Pendlebury, M (1997). *Assessing Student Learning in Higher Education*. London: RoutledgeFalmer.

Brown, S., & Knight, P. (1994). *Assessing Learners in Higher Education*. London: Kogan Page.

Brown, S. Race, P., & Bull, J. (1999). *Computer-Assisted Assessment in Higher Education*. London: Kogan Page.

Bull, J., & McKenna, C. (2004). *Blueprint for Computer-Assisted Assessment*. London: RoutledgeFalmer.

Bull, J., & Stephens, D. (1999). The use of Question Mark software for formative and summative assessment in two universities. *Innovations in Education and Training International*, 36(2), 128-136

Burstein, J., Kukich, K., Wolff, S., Lu, C., Chodorow, M., Braden-Harder, L., & Harris, M. D. (1998). Automated scoring using a hybrid feature identification technique. *Proceedings of the 17th International Conference on Computational Linguistics*. Montreal, Canada, Aug1998, pp206-210.

Callear, D., Jerrams-Smith, J., & Victor, S. (2001). Bridging Gaps in Computerised Assessment of Texts. *Proceedings of IEEE International Conference on Advanced Learning Technologies 2001*. Madison, WI, Aug2001, pp139-140.

Charman, D. (1999). Issues and impacts of using computer-based assessments (CBAs) for formative assessment. In Brown, S., Race, P., Bull, J (Eds.) *Computer-Assisted Assessment in Higher Education*. London: Kogan Page.

Chung, G. K. W. K., & O'Neill, H. F., Jr. (1997). Methodological Approaches to On-Line Scoring of Essays. Retrieved 11/09/2004, from http://www.cse.ucla.edu/CRESST/Reports/TECH461.pdf.

Crooks, T. J. (1988). The Impact of Classroom Evaluation Practices on Students. *Review of Educational research*, 58(4), 438-481.

DLL (2004). About Dynamic-Linked Libraries. Retrieved 05/05/2005, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dllproc/base/about_dynamic_link_libraries.asp.

DOM (2005). W3C Document Object Model (DOM). Retrieved 01/05/2005, from http://www.w3.org/DOM/.

Earl, L. M. (2003). *Assessment as Learning: Using Classroom Assessment to Maximize Student Learning*. Thousand Oaks, CA: Corwin Press.

e-rater (2004). The e-rater Overview. Retrieved 30/10/2004, from http://www.ets.org/portal/site/ets/menuitem.

Erwin, T. D. (1997). *Developing Strategies and Policies for Changing Universities.* London: Kogan Page.

Erwin, T. D., & Knight, P. (1995). A Transatlantic View of Assessment and Quality in Higher Education. *Quality in Higher Education*, 1(2), 179-188.

ETS (2004). ETS Home. Retrieved 29/10/2004, from http://www.ets.org.

FairTest (1999). The Value of Formative Assessment. Retrieved 11/10/2004, from http://www.fairtest.org/examarts/winter99/k-forma3.html.

Foltz, P. W. (1996). Latent Semantic Analysis for text-based research. *Behavior Research Methods, Instruments and Computers*, 28(2), 197-202.

Fontana, D., & Fernandes, M. (1994). Improvements in Mathematics Performance as a Consequence of Self-assessment in Portuguese Primary School Children. *British Journal of Educational Psychology*, 64, 407-417.

Forum (1995). Principles and Indicators for Student Assessment System. Retrieved 08/10/2004, from http://www.fairtest.org/princind.htm.

Frase, L. T., Almond, R. G., Burstein, J., Kukich, K., Sheeham, K. M., Steinberg, L. S., Mislevy, R. J., Singley, K., & Chodorow, M. (2003). Technology and Assessment. In O'Neil, Jr., H. F., & Perez, R. S. (Eds.) *Technology Applications in Education: A Learning View*. Mahwah, NJ: L. Erlbaum.

Freeman, R., & Lewis, R. (1998). *Planning and Implementing ASSESSMENT*. London: Kogan Page.

GDI+ (2005). Documentation of Microsoft Graphic Device Interface Plus API. Retrieved 12/02/2005, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdicpp/GDIPlus/GDIPlus.asp.

GhostScript (2004a). The Ghostscript Interpreter Application Programming Interface (API). Retrieved 12/10/2004, from http://www.cs.wisc.edu/~ghost/doc/cvs/API.htm.

GhostScript (2004b). How to Use GhostScript – Using GhostScript with PDF Files. Retrieved 12/10/2004, from http://www.cs.wisc.edu/~ghost/doc/cvs/Use.htm#PDF.

GMAT (2004). Take the GMAT. Retrieved 30/10/2004, from http://www.mba.com/mba/TaketheGMAT.

Harlen, W., & James, M. (1997). Assessment and Learning: Differences and Relationships between Formative and Summative Assessment. *Assessment in Education: Principles, Policies & Practice*, 4(3), 365-380.

Harlen, W. (1999). *Effective Teaching of Science: A Review of Research*. Glasgow, UK: Scottish Council for Research in Education.

Harold, E. R. (2004). *Java Network Programming*, Third Edition. Cambridge MA: O'Reilly.

Heinrich, E., & Lawn, A. (2004). Onscreen Marking Support for Formative Assessment. *Proceedings of Ed-Media2004 World Conference on Educational Multimedia, Hypermedia & Telecommunications*. L. Cantoni, C. McLoughlin (Eds.), Association for the Advancement of Computing in Education, Norfolk, US, pp1985 – 1992.

Heinrich, E., & Wang, Y. (2003). Online Marking of Essay-type Assignments. *Proceedings of Ed-Media2003 World Conference on Educational Multimedia, Hypermedia & Telecommunications*. D. Lasser, C. McNaught (Eds.), Association for the Advancement of Computing in Education, Norfolk, US, pp.768-772.

Herbert, S. (1997). *Borland C++: the Complete Reference*. Berkeley: McGraw-Hill.

Higgins, R., Hartley, P., & Skelton, A. (2002). The Conscientious Consumer: reconsidering the role of assessment feedback in student learning. *Studies in Higher Education*, 27(1), 53-64.

HTML (1999). *HTML 4.01 Specification*. Retrieved 11/11/2004, from http://www.w3.org/TR/html401/.

HTTP (1999). HyperText Transfer Protocol – HTTP/1.1. Retrieved 11/11/2004, from http://www.w3.org/Protocols/rfc2616/rfc2616.html.

HttpClient (2005). The Apache Jakarta Commons HTTPClient. Retrieved 11/11/2004, from http://jakarta.apache.org/commons/httpclient/features.html.

ICEsoft Technologies Inc. (2004). Java PDF Viewer PDF Java Applet Bean. Retrieved 11/11/2004, from http://www.icesoft.com/products/icepdf.html.

ieHTTPHeaders (2005). What is ieHTTPHeaders? Retrieved 01/03/2005, from http://www.blunck.info/iehttpheaders.html.

James, M. (1998). *Using Assessment for School Improvement*. Oxford, UK: Heinemann Educational.

Jones, A. (2003). *C# Programmer's Cookbook*. Redmond, WA: Microsoft Press.

Kakkonen, T., & Sutinen, E. (2004). Automatic assessment of the content of essays based on course materials. *Proceedings of the 2nd International Conference on Information Technology: Research and Education 2004*, July 2004, pp126-130.

Laurillard, D. (1993). *Rethinking University Teaching: a framework for the effective use of educational technology*. London: Routledge.

LSA (2004). Latent Semantic Analysis @ CU Boulder. Retrieved 30/10/2004, from http://lsa.colorado.edu.

Markin (2004). Creative Technology – Software for Teaching – Markin. Retrieved 29/10/2004, from http://www.cict.co.uk/software/markin/index.htm.

Microsoft Word Object Library (2005). Office Primary Interop Assemblies. Retrieved 01/10/2005, from http://msdn2.microsoft.com/en-us/library/15s06t57.

Miller, T. (2003). Essay Assessment with Latent Semantic Analysis. *Journal of Educational Computing Research*, 29(4), 495-512.

Moodle (2004). Moodle – A Free, Open Source Course Management System for Online Learning. Retrieved 18/11/2004, from http://www.moodle.org.

MSXML SDK (2005). Documentation of the Microsoft XML SDK 4.0. Retrieved 05/02/2005, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/xmmscXML.asp.

Murphy, R., & Torrance, H. (1988). *The Changing Face of Educational Assessment*. Philadelphia: Open University Press.

Natriello, G. (1987). The impact of evaluation processes on students. *Educational Psychologist*, 22, 155-175.

Page, E. B. (1966). The imminence of grading essays by computer. *Phi Delta Kappan*, Jan66, 238-243.

Palmer, J., Williams, R., & Dreher, H. (2002). Automated Essay Grading System Applied to a First Year University Subject – How Can We Do it Better? *Proceedings of the Informing Science IT Education Conference 2002*. Jun2002, Cork, Ireland, pp1221-1229.

Parshall, C. G., Spray, J. A., Kalohn, J., C., & Davey, T. (2002). *Practical Considerations in Computer-Based Testing*. New York: Springer-Verlag.

PDF (2003). PDF Reference, Version 1.5. Retrieved 30/10/2004, from http://partners.adobe.com/asn/acrobat/sdk/public/docs/PDFReference15_v6.pdf.

PDFCreator (2004). PDFCreator create pdf files for free. Retrieved 01/08/2005, from http://www.mrbass.org/freeware/pdfcreator/.

PDF Tools AG (2004). 3-Heights PDF Viewer Pro API. Retrieved 12/10/2004, from http://www.pdf-tools.com/asp/products.asp?name=VWPA.

PostScript (2004). Adobe PostScript 3. Retrieved 12/10/2004, from http://www.adobe.com/products/postscript/main.html.

Preston, J. A., & Schackelford, R. (1999). Improving On-line Assessment: an Investigation of Existing Marking Methodologies. *Proceedings of the 4th annual SIGCSE/SIGCUE ITiCSE conference on innovation and technology in computer science education*, Cracow, Poland, pp. 29-32.

PrimoPDF (2005). Free PDF Converter – create high-quality PDF from any printable file type. Retrieved 01/08/2005, from http://www.primopdf.com/.

Question Mark (2004). Software for quizzes, surveys, exams, assessments and tests online. Retrieved 11/11/2004, from http://www.questionmark.com/us/home.htm.

Rector, B. E. & Newcomer, J. M. (1997) *Win32 Programming*. London: Addison-Wesley.

RFC 1867 (1995). Form-based File Upload in HTML. Retrieved 01/10/2005, from http://rfc.net/rfc1867.

RFC 2388 (1998). Returning Values from Forms: multipart/form-data. Retrieved 01/10/2005, from http://rfc.net/rfc2388.html.

Robinson, J. M. (1999). Computer-Assisted Peer Review. In Brown, S., Race, P. and Bull, J. (Eds.) *Computer-Assisted Assessment in Higher Education*, London: Kogan Page.

Rowntree, D. (1987). Assessing Students: How shall we know them? Revised Edition. London: Kogan Page.

RTF (1999). Rich Text Format (RTF) Specification, Version 1.6. Retrieved 11/11/2004, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnrtfspec/html/rtfspec.asp.

Schleimer, S., Wilkerson, D., & Aiken A. (2003). Winnowing: Local Algorithms for Document Fingerprinting. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Jun2003, pages 76-85.

Shepherd, G. (2003). *Programming with Microsoft Visual C++ .Net*, Sixth Edith (Core Reference). Redmond, WA: Microsoft Press.

Sims Williams, J. H., Maher, J., Spencer, D., Barry, M. D. J., & Board, E. (1999). Automatic test generation from a database. In Brown, S., Race, P., Bull, J. (Eds.) *Computer-Assisted Assessment in Higher Education*. London: Kogan Page.

Stephens, D., Sargent, G., & Brew, I. (2001). Comparison of assessed work marking software for the ideal Integrated Marking Tool (IMT). In Danson, M., & Eabry, C. (Eds.) *Proceedings of the 5th International Computer-Assisted Assessment Conference*. Loughborough University, UK.

Stobart, G., & Gipps, C (1997). *Assessment A teacher's guide to the issues*. London: Hodder & Stoughton.

Torrence, H., & Pryor, J. (1998). Investigating Formative Assessment: Teaching, Learning and Assessment in Classroom. Philadelphia: Open University Press.

TRIADS (2004). Tripartite Interactive Assessment Delivery System. Retrieved 12/11/2004, from http://www.derby.ac.uk/assess/newdemo/mainmenu.html.

Troelsen, A. (2003). *C# and the .NET Platform*, Second Edition. Berkeley, CA: Apress.

UML (2004). Object Management Group – UML. Retrieved 11/11/2004, from http://www.uml.org/.

URI (1998). Uniform Resource Identifier (URI): Generic Syntax. Retrieved 01/10/2005, from http://rfc.net/rfc2396.html.

Valenti, S., Neri, F., & Cucchiarelli, A. (2003). An Overview of Current Research on Essay Grading. *Journal of Information Technology Education*, 2, 319-330.

WebCT (2004). WebCT Homepage. Retrieved 29/10/2004, from http://www.webct.com.

Weeden, P., Winter, J., & Broadfoot, P. (2002). Assessment. In Myers, K. and MacBeath, J. (Eds.) *What's in it for school?* London: RoutledgeFalmer.

Wiggins, G. (1998). *Educative Assessment: Designing Assessments to Inform and Improve Student Performance*. San Francisco, CA: Jossey-Bass.

Williams, R. (2001). Automated essay grading: An evaluation of four conceptual models. Retrieved 01/11/2004, from http://www.fit.cbs.curtin.edu.au/AEG/Documents/Automated%20essay%20grading.pdf.

WinInet (2005). Internet Programming with WinInet. Retrieved 01/05/2005, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore98/HTML/_core_internet_programming_with_wininet.asp.

Yorke, M. (2003). Formative assessment in higher education: Moves towards theory and the enhancement of pedagogic practice. *Higher Education*, 45(4), 477-501.

Appendix A Marked Assignment XML Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="MarkedAssignment">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="PdfFile" type="xsd:string"/>
        <xsd:element name="MaxMark" type="xsd:decimal" minOccurs="0"/>
        <xsd:element name="Criteria" type="CCriteria" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="DRAWELEMENT" type="CDrawElement" minOccurs="0"
                     maxOccurs="unbounded"/>
        <xsd:element name="Summary" type="CSummary" minOccurs="0"
                     maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="MarkingSchema">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Criteria" type="CCriteria" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="CCriteria">
    <xsd:sequence>
      <xsd:element name="MaxMark" type="xsd:decimal"/>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Detail" type="xsd:string"/>
      <xsd:element name="Color" type="CColor"/>
      <xsd:element name="FrequentComment" type="xsd:string" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CDrawElement">
    <xsd:sequence>
      <xsd:element name="ShapeID" type="SHAPE_ID"/>
      <xsd:element name="ContextINFO" type="CONTEXT_INFO"/>
      <xsd:element name="ElementType" type="CDrawElementType"/>
      <xsd:element name="FirstPoint" type="CPoint"/>
      <xsd:element name="SecondPoint" type="CPoint"/>
      <xsd:element name="PenWidth" type="xsd:unsignedByte"/>
      <xsd:element name="Color" type="CColor"/>
      <xsd:element name="Annotation" type="CAnnotation" minOccurs="0"
                   maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="SHAPE_ID">
    <xsd:sequence>
      <xsd:element name="PageNumber" type="xsd:positiveInteger"/>
      <xsd:element name="Creator" type="xsd:string"/>
      <xsd:element name="CreatorNickname" type="xsd:string"/>
      <xsd:element name="CreateTime" type="xsd:string"/>
      <xsd:element name="email" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CONTEXT_INFO">
    <xsd:sequence>
```

```
      <xsd:element name="PageOffsetX" type="xsd:positiveInteger"/>
      <xsd:element name="PageOffsetY" type="xsd:positiveInteger"/>
      <xsd:element name="PageWidth" type="xsd:positiveInteger"/>
      <xsd:element name="PageHeight" type="xsd:positiveInteger"/>
      <xsd:element name="Rotation" type="CRotationType"/>
      <xsd:element name="Resolution" type="xsd:positiveInteger"/>
      <xsd:element name="PageNumber" type="xsd:positiveInteger"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CPoint">
    <xsd:sequence>
      <xsd:element name="X" type="xsd:integer"/>
      <xsd:element name="Y" type="xsd:integer"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CColor">
    <xsd:sequence>
      <xsd:element name="R" type="xsd:unsignedByte"/>
      <xsd:element name="G" type="xsd:unsignedByte"/>
      <xsd:element name="B" type="xsd:unsignedByte"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="CDrawElementType">
    <xsd:restriction base="xsd:unsignedByte">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="4"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="CRotationType">
    <xsd:restriction base="xsd:unsignedByte">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="3"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:complexType name="CAnnotation">
    <xsd:sequence>
      <xsd:element name="ShapeID" type="SHAPE_ID"/>
      <xsd:element name="FirstPoint" type="CPoint"/>
      <xsd:element name="SecondPoint" type="CPoint"/>
      <xsd:element name="Content" type="xsd:string"/>
      <xsd:element name="Color" type="CColor"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CSummary">
    <xsd:sequence>
      <xsd:element name="MaxMark" type="xsd:decimal"/>
      <xsd:element name="FinalMark" type="xsd:decimal"/>
      <xsd:element name="Creator" type="xsd:string"/>
      <xsd:element name="CreatorNickname" type="xsd:string"/>
      <xsd:element name="CreateTime" type="xsd:string"/>
      <xsd:element name="Content" type="xsd:string"/>
      <xsd:sequence>
        <xsd:element name="MarkEntry" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="MaxMark" type="xsd:decimal"/>
              <xsd:element name="FinalMark" type="xsd:decimal"/>
              <xsd:element name="Color">
                <xsd:complexType>
                  <xsd:sequence>
```

```
                        <xsd:element name="R" type="xsd:unsignedByte"/>
                        <xsd:element name="G" type="xsd:unsignedByte"/>
                        <xsd:element name="B" type="xsd:unsignedByte"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Appendix B Table of Contents of the WebCTConnect 1.1 Manual

The full content of the manual is available from:
http://www-ist.massey.ac.nz/MarkTool/WebCTConnect1.1Jul20051.pdf