Managing On-line Submission and Marking of Programming Assignments

by

Nuray Dindar

M.Sc., Koc University, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA
(Vancouver)
December 2012
© Nuray Dindar 2012
Abstract

This thesis describes an on-line system for the management and marking of course materials. The interface is tailored to provide administrative benefits, and time and resource savings while taking privacy, security and reliability into account. There are three user roles in the system. Instructors use the system to manage the course material, design the marking rubrics for assessment, assign teaching assistants to mark assignments and receive feedback from students regarding exceptions and special needs. Teaching assistants evaluate students’ works electronically by accessing students’ files on-line. Each student enrolled in a course has a secure personal “pickup directory” (web page) to access his/her submitted materials, the corresponding marking reports, specific deadlines for his/her lab section, and other personal course-related information including responses to polls and seating preferences for exams. We have used our system in an introductory-level computer science course offered at University of British Columbia with 190 students and eight teaching assistants.
This work has not been previously published. All of the contributions were made by me. The original version of the customized hand in script was written by Matt Hoffman. I modified the script as described in the thesis.
Table of Contents

Abstract ............................................................. ii
Preface ............................................................... iii
Table of Contents ............................................... iv
List of Figures .................................................... vi
Acknowledgements ................................................ v

1 Introduction .................................................... 1
  1.1 Motivation for the System .............................. 1
  1.2 Marking Programming Assignments vs. Other Types of Marking .............................. 2
  1.3 Organization of the Thesis ............................. 3

2 On-line Marking System ...................................... 4
  2.1 The Marking Process ..................................... 4
  2.2 Requirements of an On-line Marking System .......................... 5

3 Review of Existing On-line Marking Systems .............. 8
  3.1 Web-CAT ..................................................... 8
  3.2 Marmoset .................................................... 10
  3.3 Caesar ........................................................ 12
  3.4 WebCTConnect ............................................ 12
# Table of Contents

4 UBC CS Evaluation Systems: A Case Study ............................................. 16

4.1 Pre-Marking ................................................................. 16

4.1.1 The Existing System ............................................. 16

4.1.2 Problems with the Existing System ........................... 18

4.1.3 Our System .......................................................... 21

4.2 Marking ..................................................................... 21

4.2.1 The Existing System ............................................. 22

4.2.2 Problems with the Existing System ........................... 23

4.2.3 Our System .......................................................... 25

4.3 Post-Marking and Course Management ................................. 27

4.3.1 The Existing System ............................................. 28

4.3.2 Problems with the Existing System ........................... 28

4.3.3 Our System .......................................................... 28

5 Conclusions ................................................................. 30

5.1 Future Work ............................................................. 30

References ................................................................. 32

Appendices
# List of Figures

2.1 The process of managing and marking assignments .......................... 4

3.1 Screenshots of the Web-CAT interface: (a) Main status page, (b) Automated results of a submission ............................... 9

3.2 Screenshots of the Marmoset interface: (a) Main status page, (b) Automated results of a submission ............................... 11

3.3 Screenshots of the Caesar interface: (a) Dashboard interface with submission and task assignments, (b) Reviewing interface, (c) Interface for browsing submission code ............................... 13

3.4 Screenshots of the WebCTConnect interface: (a) Main status page, (b) Marking Schema with frequently used comment box  .......... 14

4.1 Snapshot of UBC handin interface ............................................. 17

4.2 Assignment names and due dates at (a) UBC handin, and (b) CPSC 260 webpage ......................................................... 19

4.3 Screenshot of a pickup directory webpage ................................. 20

4.4 Formatted marking report ......................................................... 22

4.5 (a) Setting the marking report, (b) Creating the small rubrics, and (c) Allocating markers and monitoring progress  .......... 24

4.6 Marking interface with four panes ............................................. 25
Acknowledgements

The work reported in this thesis was partially supported by funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and by the Networks of Centres of Excellence Program through the Graphics, Animation and New Media NCE (GRAND).

I would like to express my appreciation to my supervisor, Dr. Kellogg Booth, for his valuable time, abundant help, and patience throughout the entire course of this thesis. Working under his supervision has always been inspiring.

Special thanks to the second reader of my thesis, Dr. Ed Knorr for his time, advice and understanding.

I would like to acknowledge my friends, Angelika Gnieser, Marcos Gines-tra, Mikhail Bessmeltsev, Neto Torres, Simona Radu, Rahul Jiresal, and Roj Johal for being supportive and making Vancouver years memorable.

My sincere appreciation goes to Vasanth Kumar Rajendran for tirelessly listening to my problems and brainstorming solutions, coding with me and even editing this thesis.

I am indebted to my family for enduring me all my life and for their infinite trust in me and encouragement to further my education. I want to specifically thank to my niece, Ela and my nephew, Arda who make me laugh even under the most stressful conditions. Lastly, I thank my brothers, Cemil and Zafer, my sisters, Nihal and Beste, and my parents, Ayse and Fettah. I am really lucky to be part of this family.
Chapter 1

Introduction

1.1 Motivation for the System

Assessments of assignments are an important part of the educational process. They are essential for achieving pedagogical goals and increasing student motivation. However, assessment introduces administrative workload, particularly for large classes. On-line marking systems are becoming increasingly important and widely used. They offer time and resource savings as well as minimizing geographical dependency, which reduces the administrative workload, while retaining and in some cases increasing the pedagogical benefits of assessment.

In the education literature, assessments are considered as support and improvement tools for student learning. They are directly linked to the teaching and learning strategy of a course (Kendle, Northcote, et al., 2000; Macdonald, 2003). Hattie (2009) states that post-secondary institutions constantly work to enhance assessment quality and points out that appropriate feedback is a essential component. Nitko (1996) and Sadler (1989) note the importance of having explicit explanations in the feedback about what students need to do to meet the goals of the assignment, and what a good solution looks like.

Technology can be used effectively to improve the evaluation of student submissions while giving priority to support for student learning (Whitelock & Watt, 2007). A study conducted by Dermo (2009) about student perceptions of on-line learning tool usage indicates that electronic assessment helped student learning and the teaching process. Advantages of using an on-line marking tool include increased efficiency and effectiveness, quality of marker feedback due to the ability to re-mark or to perform parallel assignment marking, possible participation of students in the assessment process such as peer review, and strengthening of learning outcomes (Milne, Heinrich, Crooks, Granshaw, 2008).
1.2 Marking Programming Assignments vs. Other Types of Marking

& Moore, 2007; Whitelock, 2009; Burrows & Shortis, 2011). A study by Denton et al. (2008) on the effectiveness of electronic feedback showed that students rate electronic feedback as superior to traditional feedback.

After review of the educational literature on existing on-line marking systems, we designed and implemented an on-line system to manage and mark programming assignments in the Department of Computer Science at the University of British Columbia.

1.2 Marking Programming Assignments vs. Other Types of Marking

At a high level, the assessment process follows a somewhat standard pattern, which we review in Chapter 2. However, the low-level details vary a lot depending on the academic subject and the specific type of work that is being assessed. Our particular interest is in marking programming assignments, especially for introductory courses in computer science.

The assessment process always involves some mechanism whereby students submit material that will be assessed according to some set of guidelines or procedures. For programming assignments this usually takes the form of one or more files that contain the source code for the program(s) that are being assessed. The particular programming language may dictate the number and types of files that are required, as well as some aspects of how the assessment will be carried out. For our purposes, we will assume that one or more files are submitted that are being assessed through a combination of automated or semi-automated steps along with one or more manual steps where human markers review and comment on the source code.

One thing that is very different for programming assignments, and which presents a technical challenge that is beyond the scope of what we have done, is the problem of malicious code. Unlike essays or other text-based answers, the assessment of computer programs often requires the use of a compiler to translate the source code into executable code that is then run against one or more sets of test cases to detect whether the computation embodied in the program is correct. The difficulty that arises that is unique to programming assignments is that the code that is executed could in fact perform actions that
are extremely detrimental, such as deleting all of the files in the account under which the program is run, or copying or altering some or all of the files.

If, for example, the files that are modified by a “rogue” program contain the results of the marking process, a malicious (but talented) student could manage to achieve a high mark in the course while awarding lower marks to all other students. Obviously this is a situation that should be avoided. There are fundamental limitations on the degree to which automated tools can guarantee that rogue programs can be detected. We therefore do not address this problem in our work, although we do adopt a solution that can be extended in a variety of ways to use tools that are specifically designed to help solve this problem.

## 1.3 Organization of the Thesis

The organization of the remainder of this thesis is as follows: Chapter 2 presents the general characteristics and requirements of on-line assessment systems, while the review of existing marking systems is described in Chapter 3. Chapter 4 describes the evaluation systems currently used in the Department of Computer Science (CS) at the University of British Columbia (UBC), some of the problems that exist, and our approach to resolving those problems. We offer some conclusions in Chapter 5 and describe possible future work that would build on what we have done so far.
Chapter 2

On-line Marking System

In this chapter, we outline the basic phases of the marking process, and identify key requirements for an on-line assignment marking system. According to the educational literature, a 'good' marking system has to clearly specify marking rubrics, provide personalized feedback and be consistent. In addition to these basic requirements, it is also desirable that an assignment marking interface provide an on-line viewer for students’ work, secure access for distributing the work among markers and marking the submissions, and a system for automatically releasing the results to students (Heinrich, Milne, & Granshaw, 2012).

2.1 The Marking Process

![Diagram](image.png)

Figure 2.1: The process of managing and marking assignments

The process of managing and marking assignments (Heinrich et al., 2012) derived from interviews with academics (Milne et al., 2007) and educational theories is shown in Figure 2.1. Three conceptual categories can be expressed in detail as:

1. **Pre-Marking:** Pre-marking includes setting up the assignment drop box,
2.2 Requirements of an On-line Marking System

specifying the assignment’s specific parameters such as due dates and late submission rules, handling submissions of students and storing the submitted files in a secure place.

2. **Marking:** The marking process is initiated by the instructor when s/he creates the marking rubrics and allocates markers to students. Teaching assistants then assess the assignments based on those criteria. Often the marking process also includes a progress monitoring system to facilitate timely return of marking reports and to check consistency among the markers.

3. **Post-Marking:** The marking process is followed by post-marking when the results are returned to students and saved for subsequent grade calculations by the instructor.

2.2 Requirements of an On-line Marking System

The following are some of the important requirements of an on-line marking system. Many are derived from work by Heinrich et al. (2012), which is based on educational theories and general practices with electronic systems. A few are based on our observations and experience with previous marking processes.

**Marking Schema**

The marking schema forms the base of the assessment by creating criteria and guidelines which students follow in their work. An effective assignment assessment tool should let the instructor tailor the marking schema and related functionalities for each assignment. Each student normally receives feedback on their work, which is broken down into detail according to the assignment’s marking schema.

**Marker Allocation**

Allocation of markers to students has to be done by the instructor. A list of potential markers is identified and their access permissions for the required students’ works are set accordingly. Based on the course, this allocation can be done manually or by a predefined allocation algorithm.
2.2. Requirements of an On-line Marking System

Monitoring the Progress
A monitoring tool is needed for the instructor to obtain information about the marking progress and perform final quality checks on the marking. In case there is a need for re-marking or reallocation to a new marker, this tool should allow the instructor to take the desired action. Marked assignments that pass the quality review should be labeled as ‘done’ and become available to students at a time determined by the instructor.

Marking Results
Marking results have to be accessible without the need for any specialized software. Students should be able to export and download marking reports into a separate feedback sheet, in case they want to refer to them beyond the course duration.

Comment Bank
A comment bank is a list of comments that fits the marking criteria and points out possible common problems students might have in their assignments. It is used to promote consistency across the various markers’ feedback.

Analysis Tool
An analysis tool is needed for a detailed review of the students’ work and their marks. Both statistical as well as textual analysis should be possible. This tool needs to give information about students’ strengths and weaknesses in specific topics. It can also be used to perform a comparative analysis of the markers.

Re-usability
Access to the marking schemes for institutions has to be retained beyond the course duration, since these schemes can be used in future assignments. With the help of the analysis tool, schemes and comment banks can be modified to address problems with previous offerings of a course.

Customizability
The system should be able to cater to a wide range of course types and class sizes. The customizability should also allow for (semi-)automation of some of the tasks, and extension of its functionalities using add-ons or plug-ins.
2.2. Requirements of an On-line Marking System

Security
Access to the assessment tool and students’ work has to be secure and only reachable by authorized markers. The communication between the markers has to be secure as well and the evaluated work needs to be stored with adequate measures in place to prevent data loss.

Mobility
Markers should be able to use the marking tool from any location, with the ability to smoothly transition between different locations. They also need to be able to mark assignments even without an active network connection, or at least with efficient data usage so that they can mark assignments with a slow network connection.

Integration
The system has to be able to integrate with other marking tools (such as plagiarism detection tools, for example) to provide comprehensive assessment. Moreover, the architecture of the system should allow for future extensions.

Usability
The system has to be easy to learn and to use. The instructors and markers should focus on assessment, rather than being distracted with learning to use the system’s features and complexities.
Chapter 3

Review of Existing On-line Marking Systems

In this chapter, we review some of the existing marking tools for programming projects. We give a brief overview of these tools and follow with a discussion of their features and shortcomings with respect to the requirements outlined in Chapter 2.

3.1 Web-CAT

Web-CAT (Shah, 2003) is open source software developed at the Virginia Polytechnic Institute and State University. It is mainly used for automating the marking of coding assignments, and providing feedback to students through code testing. This allows students to test their code throughout its development (Edwards, 2003). Web-CAT also allows a manual grading interface for the assignments where TAs can login from a web browser, and provide feedback using comment boxes in an editor.

Web-CAT provides an open API, so that several authentication strategies can be chosen or it can be used with existing campus-wide authentication services through institution-specific adapters. The interface can be accessed from a web browser. Web-CAT’s plug-in based architecture makes it flexible, and open to other languages. For guarding the services, they provide two different controls: user-specific permission and role-based access control. They also provide a detection system for erroneous and malicious programs.

Web-CAT provides a strong interface for the pre-marking tasks (see Figure 3.1), but the marking and post-marking phases have limited support. The marking is mainly done automatically based on code style and test cases. The
3.1. Web-CAT

(a)

Figure 3.1: Screenshots of the Web-CAT interface: (a) Main status page, (b) Automated results of a submission
marking schema is not given importance, so it might not be suitable for many courses.

### 3.2 Marmoset

Marmoset (Spacco, Hovemeyer, & Pugh, 2004) is open source system designed at the University of Maryland. It is similar to Web-CAT in terms of providing automated code testing results to students. Management of programming projects starts when an instructor posts a project description with sample inputs and expected outputs as unit tests. Students can submit their work online and see the server-side code testing results on the web page. This can identify platform-specific discrepancies. For some languages, Marmoset also provides automated static analysis and code coverage results. Screen shots of the interface are shown in Figure 3.2.

Marmoset differs from Web-CAT by offering a token-based incentive system. The instructor has confidential test cases, called “release tests”. Students can test their code against these release tests by using up a token. Usually, the limit is three tokens for every 24 hours. The system reveals the names of the first two failed test cases, and the total count of failed tests. This incentive system encourages students to start working on their project early and to analyze code behavior critically.

Marmoset also supports various kinds of code reviews such as in-progress, instructional, and peer review. In-progress reviews are initiated by a student asking for a help. They are done by an instructor or a teaching assistant. Instructional reviews are a part of the usual marking process. Peer reviews allows students to review each other’s code.

Although Marmoset supports pre-marking and marking tasks adequately, post-marking is limited. Students cannot directly request a re-evaluation and reports cannot be downloaded. Marking TAs have to write comments inside text boxes without auto-completion or comment banks. This makes it hard for TAs to be consistent and introduces extra workload. The layout of the interface is less usable compared to the other marking systems.
3.3. Caesar

Figure 3.2: Screenshots of the Marmoset interface: (a) Main status page, (b) Automated results of a submission
3.3 Caesar

Caesar (Tang & Miller, 2011) is an open source distributed social code review tool developed at the Massachusetts Institute of Technology. It is designed to be used in a classroom setting, while being modeled after industrial code review tools. The submissions are first processed using the Java preprocessor, where the code is partitioned into small chunks, trivial and redundant code segments (such as source code provided by the instructor, or empty function definitions) are filtered out, automated comments are generated by running static analysis, and similar code chunks are clustered. The chunks, referred to as tasks, are routed to reviewers dynamically. Screenshots of the interface are shown in Figure 3.3.

Reviewing is done through a web interface and feedback is presented as annotations on the source code. The annotations can be rated and commented on, making it a social interface for reviewing. Course staff can monitor and evaluate the reviewing progress.

Caesar is primarily a code review system and is not built for marking. It does not have any marking rubrics, but we think the idea of “social” code comments and chunking of code can be useful additions for marking interfaces.

3.4 WebCTConnect

WebCTConnect (Massey University, 2006) is a desktop application that allows instructors and TAs to retrieve assignment submission from WebCT (Goldberg & Salari, 1996) and upload the marking reports back to it. It provides a table-like interface for managing assignments, marks and TA allocations. The table can be sorted by various criteria such as student ID, name, marker, etc. From this interface, the markers can download the student’s work directly.

The most notable feature of WebCTConnect is the comment bank, which allows TAs to add and reuse frequent comments. This saves time, increases consistency and allows a more cooperative marking process for TAs. The main page (see Figure 3.4) presents a quick overview of marking progress and marker allocation. This system also provides support for group assignments, which is very helpful in practice.
3.4. WebCTConnect

Figure 3.3: Screenshots of the Caesar interface: (a) Dashboard interface with submission and task assignments, (b) Reviewing interface, (c) Interface for browsing submission code
Figure 3.4: Screenshots of the WebCTConnect interface: (a) Main status page, (b) Marking Schema with frequently used comment box.
3.4. WebCTConnect

Being a desktop application, WebCTConnect ties TAs down to one computer, and requires them to download and install the software. It is only a marking management tool and does not provide a file viewer for submissions. The only method of synchronization among the markers is by importing and exporting XML files and sharing them via e-mail, which makes the system error prone and the marking process difficult to keep track of.
Chapter 4

UBC CS Evaluation Systems: A Case Study

We discuss the three phases of the marking process (Figure 2.1) with reference to an introductory-level computer science course offered in the Department of Computer Science at the University of British Columbia. We evaluate existing interfaces against the list of requirements discussed in Chapter 2. We designed our own online marking system for this course that addresses some of the shortcomings of existing systems. We discuss the details of this interface in this chapter.

4.1 Pre-Marking

The pre-marking process consists of three tasks: (i) creating an assignment, (ii) receiving submissions from students, and (iii) storing the assignment submissions in an appropriate format. The creation of an assignment depends on the course, and is not a relevant part of the process for our work. Hence, we discuss only tasks (ii) and (iii) in this section.

4.1.1 The Existing System

The Department of Computer Science at UBC provides a simple and secure interface, handin (University of British Columbia, 2012), for submitting assignments electronically. Most of the computer science courses use the handin web interface to receive students’ work. Some use the customizable script to process the files in some way particular to their course. The web interface is shown in Figure 4.1. It requires users to have an active undergraduate student account to
be able to submit an assignment to the system. The students input the course name, assignment name, a file path for a compressed archive of files to be uploaded, and an option for overwriting previous submissions.

Every course can have an optional handin configuration file to manage the assignments that can be handed in. This file has information about the assignment name, the due date and time, and possibly a blocking date and time. This configuration system carries a few quirks in its behaviour:

- If there is just an assignment name in the configuration file or the date information does not fit the specified format, students are allowed to submit their work only once.
- If both the name and due date with time is specified, students can submit their work as many times as they would like to until the due date for the assignment.
4.1. Pre-Marking

- If a block date is set, students are allowed to submit after the due date but they cannot submit any files after the block date.

- If the block date is omitted, students can submit their files only once after the due date.

- Finally, if there is no configuration file in the course directory, the system lets students hand in an assignment of any name at most one time per assignment.

The departmental handin saves the students’ work to separate directories named with each student’s four character undergraduate user ID under a directory with the assignment name. For example, if a student with undergraduate user ID a1b2 submits an assignment for lab01 of course CS120 offered in the Winter Term 1 of 2012, then the submitted files are saved into a directory called cs120/2012W1/lab01/a1b2.

After a student’s work is copied by handin, a course can have its own customized script to process the student’s work. For example, an automated marking script can assess students’ submissions or a filter can be built to reject specific files.

4.1.2 Problems with the Existing System

There are both security and design problems in the handin interface.

Students can upload their assignments to any of the offered courses in the Department of CS at UBC as long as they have a valid undergraduate account. The system does not check if the student is enrolled in the course they’re submitting files for. The absence of this check might result in extra work for the markers, in case there are files submitted by non-registered students.

The system does not check the contents of submitted files, which makes the task of receiving and storing them unreliable. In case a student’s account is compromised, it is possible for compromised or malicious files to be submitted as the assignment. If these submissions are executed on the same server as the other course files, this malicious source code might cause the loss of all submissions or alter the marking reports.

There are also some design problems with handin that make the interface
4.1. Pre-Marking

handin lists the assignment names and corresponding due dates directly the way they are written in the configuration file of the handin system (see Figure 4.2a). This violates a basic design heuristic by introducing a mismatch between the system and the real world. The handin interface is also not consistent with the standard course codes used at UBC. In the UBC registration system, the computer science courses have “CPSC” as the departmental code, but students needed to use “cs” instead of “CPSC” for the course name field in the interface.

The system is also error prone, because it relies on the student to enter the course name manually. Moreover, the system assumes that every student using the handin interface intends to submit the file for the current term. However, there are instances where a student submits their work for a previous term (in case of a medical absence, for example). In the existing handin system, it is not possible to accomplish this task, because term information is implicitly assumed.

While the handin interface lets the course instructors and TAs access the students’ work, there is no way for students to see the latest version of their submitted assignments. Lack of this feature makes students feel less confident about the system.
4.1. Pre-Marking

Figure 4.3: Screenshot of a pickup directory webpage
4.2. Marking

4.1.3 Our System

We augmented the existing handin web interface with a script that processes the submitted files and gives a warning message if necessary. If a student submits a file with a name other than one of the acceptable names specified in the configuration file of the specific assignment, a warning message is shown on the web interface. The warning message shows the missing or unrecognized files in the submitted archive. The script also extracts the submitted archive and copies its contents to the student’s “pickup directory”.

Pickup directories are personal directories for each student with a webpage where they can see personal course related content including the latest submitted documents for each lab or assignment, corresponding marking reports, personal deadlines (see Figure 4.2b), and responses to course polls. Screenshots of this webpage are shown in Figure 4.3. Each student needs to have a valid UBC CS username and password to be able to access their pickup directory. The usernames are checked against the course’s list of registered students. Because students are asked for credentials, the system is secure or at least as secure as all other aspects of the undergraduate computer environment. This unified webpage makes it easy for students to access relevant and personalized course-related content.

4.2 Marking

The marking process usually consists of five tasks, done either by the instructor or the markers (usually the teaching assistants).

- The instructor defines the marking scheme for the assignment
- The assignment submissions are allocated to markers, either automatically, or by the instructor
- Markers assess the students’ work
- The instructor monitors the marking progress
- The consistency among different markers is checked
4.2. Marking

4.2.1 The Existing System

There is no unified marking interface provided by the Department of CS at UBC. Therefore the marking process differs between various courses and even between different offerings of the same course across terms.

There is always a marking scheme provided for the markers, explicitly or implicitly. The most common approach is to provide a text file explaining various parts of the assignment, accepted solutions and marking scale.

The allocation of markers is usually done by the team of markers (TAs)
internally, such that every marker gets an equal number of assignments to assess. Some instructors choose to use an algorithm to divide the work among the markers. In such cases, usually the aim is to make sure each student is marked by a different marker for different assignments. This ensures that the effect of individual differences between markers is minimized.

Because the submitted files are stored in the course account and there is no interface for accessing the folders other than logging into the course account, the instructors usually hand over the course account’s password to the markers, even though the official departmental policy states otherwise. Then, the markers do the marking either right on the server or on their personal computers. Finally, they fill out the marking report for each student.

A sample marking report, which is a formatted text file, is shown in Figure 4.4. When markers are done with marking, they need to upload the reports back to the server. Updates on the marking progress are usually done through emails. Some courses use scripts to check the number of finished marking reports in the course account.

4.2.2 Problems with the Existing System

Sharing the course account credentials with the TAs raises many potential problems. When there is only one course account and multiple markers have access to the account, tracking down actions in the account is almost impossible. In case one of the students’ submissions is deleted by accident, it might be impossible to recover from that mistake.

Another possible problem is some of the TAs may also be students for other courses. By possessing one course’s account credentials, they have access to everything that course account can access, which might introduce a security problem. Normally, this is not a problem because courses have separate accounts, but the accounts are all in the same Unix file group, which creates a potential for shared information that might not be appropriate for every TA to have access to. In addition, since most of the time TAs change each term, the password for the course account should be changed at least every term, so that only the intended set of TAs have access to the account. This is not always followed, especially when an instructor for a previous or subsequent term uses the website to set up or retrieve information for a different term.
4.2. Marking

Figure 4.5: (a) Setting the marking report, (b) Creating the small rubrics, and (c) Allocating markers and monitoring progress
4.2. Marking

Using text files requires the markers to preserve the formatting (usually a fixed tab-delimited format). In case there are scripts which monitor or release these reports, this process can become extremely error-prone.

![Figure 4.6: Marking interface with four panes](image)

4.2.3 Our System

Our system provides a unified web interface for managing and marking the assignment submissions.

In the marking interface, the marking scheme is shown as a series of small collapsible schemes for various subsections of the assignment. The instructor initializes the report format by specifying the schemes needed for that assignment and the files associated with each of the schemes. File names substitute underscore ("_") for dot ("." ) to avoid complications in the scripts that process the files. An example of the setup process is shown in Figure 4.5a where the marking report has three sub-schemes with php extensions. The associated files are listed in the same line for each marking block. For example, in
4.2. Marking

the Figure[4.5a] author.php is associated with the files date.cpp, date.hpp and date-driver.cpp, while declare.php and implement.php are linked to date.hpp and date.cpp respectively.

Instructors can reuse previously defined marking blocks for an assignment or use the management interface to create a new one. The interface for creating a new php file is shown in Figure[4.5b]. The instructor is required to enter a file name and caption information to create the marking template. The instructor also asked for the items that need to be assessed by the markers and the associated marks (points). The instructor can see the scheme block in the “File view” pane at the bottom to make changes before finalizing the marking scheme. Access to the management system is granted only to the instructor and the course account.

Allocation of the markers is semi-automated. It needs information about the markers and the students who have submitted files for the corresponding assignment. These files are provided by the instructor. Our allocation strategy assigns an equal amount of work for each TA. Then, TAs can access the list of students with links of corresponding marking reports through the TA assignment interface shown in Figure[4.5c]. The interface also shows the marking progress as “done” or “not done”. The option to choose the allocated assignments for a specific TA makes the system more visible for TAs, because they can filter out the other TAs and make sure that they have not missed marking any of their allocated assignments. Access to the marker allocation list is granted to the marking team, the instructor and the course account.

For the actual marking task, we designed the marking interface with four panes as shown in Figure[4.6]. The top left pane has a clickable list of files, while the bottom left pane has a viewer. The top right pane has information about the student’s identity, the marker’s name, and the date of the assignment. The bottom right pane has a marking form that needs to be filled out by the markers.

The file list pane has three extra files in addition to the files submitted by the student. One of these files is “compilation.txt”, which has information about the compilation results (and possible errors and warnings from the compiler) of the submitted project. The other one is “confirmation.txt”, which has names of the submitted files and the submission date. The last one is “report.txt”, which is the saved marking report for that assignment. The file “report.txt” is
only visible to the markers. Students do not have access to that file.

When a marker clicks the name of a file listed in the top left pane, s/he can view the content of the file with syntax highlighting in the viewer pane. We used the javascript code prettifier for syntax highlighting. It supports all C-like, Bash-like and XML-like languages (Google code prettify, 2012). Simultaneously, the associated marking rubrics in the bottom right pane are shown. Every box on the right-hand side is expandable, so that markers can collapse the box as they mark the assignments.

Each section in the marking form is provided with a comment box for markers to give personal feedback to students. The identity information pane and in lab activities are only for information. Their content is loaded from the corresponding files on the course account.

Because our marking scheme does not have any partial marking, our marking form has only radio buttons and text areas. However, if there is a need for a more elaborate marking scheme, pull-down menu items can be used as well. The structured form makes the system more reliable by not relying on TAs to keep the formatting of the report intact. Access to the marking interface is only granted to the marking team, instructor, and course account. A marker has access to the marking reports created by another marker. This helps in ad-hoc normalization among markers. After all of the marking is done, the instructor can check the consistency of marking and release the marks to the students.

4.3 Post-Marking and Course Management

This section discusses the details of releasing the marks to the students, receiving their feedback regarding their marks, and managing any special requirements. As mentioned in Chapter 2, marking results have to be easily accessible without the need for any specific software. Students often make enquiries about their marks, but it is not clear whom they should approach—the instructor, or one of many TAs. In addition, it is also desirable to have an unified system that can keep track of requests from students and be accessible online.
4.3. Post-Marking and Course Management

4.3.1 The Existing System

The Department of CS at UBC does not have an interface that addresses the requirements of post-marking tasks. Usually, the marking results are returned to students via e-mail. If a marking scheme was used, the results are in a formatted text file (shown in Figure 4.5a). Sometimes the results are given on paper with the marker’s comments. If there is no need for any feedback, students can just see their grades through WebCT or some other mechanism. If students need clarification or re-evaluation, such requests are mainly handled via e-mail, through the course discussion board, in person or with a printed form. Management of special needs, such as left-handed seating requests for in-class exams, or disability assistance are also dealt with through similar means.

4.3.2 Problems with the Existing System

Students receive e-mail reports from different courses, and it makes it hard for them to keep track of their reports. For instructors it is difficult to ensure that reports are delivered and read.

Students requests might be initiated through various means and handled by any of the markers or instructors. This makes it hard to keep track of the requests, and ensure a uniform protocol for handling them. At the least it introduces additional administrative work. There is a possibility that requests might be lost, in particular when printed forms are used.

4.3.3 Our System

In our system the marking reports are released to students with same interface used by the markers. They see the same interface as the TAs do, but the buttons in the bottom right pane are disabled (see Figure 4.6). Hence, students can see their marked files in the viewer pane with the corresponding feedback (based on the marking scheme) on the right hand side.

If students request re-evaluation, they submit a “mark change request form”. This can be accessed from their pickup directory and submitted electronically. When they submit the form, a confirmation is sent, and the instructor receives a notification. In addition, a copy of the form is saved under the course account, and can be accessed by the student, the TAs and the instructor.
4.3. Post-Marking and Course Management

We also implemented an interface to conduct course-related polls to find out students’ preferences, for example about the date of a make-up class, or exam seating. This interface can be reached from the pickup directory. We have scripts that process and summarize the responses.
Chapter 5

Conclusions

In this thesis, we have described the common practices for assessing coding in the Department of Computer Science at UBC and reviewed the existing programming assignment marking systems. Based on our observations and the educational literature, we developed a list of requirements (discussed in Chapter 2). We have also identified appropriate design aspects, features, and potential design flows for modeling a secure, reliable, and fair on-line marking system (described in Chapter 4). This system was used over one term by 190 students and eight teaching assistants in a introductory-level computer science course offered at the University of British Columbia. Based on our experience using the system we present areas of improvements for our system and outline future work.

5.1 Future Work

We can add an intermediate step between the pre-marking and marking phases to run pre-processing tools on the submitted code. For example, we can check for plagiarism using a tool like MOSS [Aiken et al., 2005] to look for similar assignments and take necessary actions. Based on the results, we could create “clusters” of assignments that are allocated to the same marker, so that potential plagiarism can be detected more easily.

The allocation strategy we use in our system is a simple strategy. In the future, we would like to give the instructor the option of choosing different strategies or even manually allocating markers. Randomization is a commonly used strategy to ensure that the effects of individual differences among markers are minimized. Another strategy that is sometimes preferred is to retain the same markers across assignments, particularly if the assignments build on each other.
5.1. Future Work

In our marking interface, we have comment boxes for each sub-schema. But sometimes, it is necessary to refer to specific parts of the code from the comment. We could develop a associative array structure that will allow us to place (potentially multiple) anchors in our comments. For example, if a student has a memory allocation mistake for a pointer, we would like our comment to link to the declaration and problematic access point(s) in the code.

A comment bank should be added to our marking interface for ensuring consistency, reducing workload and qualitatively analyzing students’ work in the course. These comments could be initially added by the instructor or be automatically added after a sufficient number of instances are entered by the markers. Our system needs to be smart enough to group similar comments. The comment bank can be accessed through either a menu or an auto-completion feature. It might also be useful to have the ability to have thumbs-up-or-down comments so that the comment bank can be more collaborative.

Lastly, we could give more freedom to the user to choose the UI layout to suit their task and needs. For example, the code viewer pane could be made re-sizeable or be made to pop out into a separate window.
References


