Evaluation of the BOSS Online Submission and Assessment System

PeyShan Heng, Mike Joy, Russell Boyatt and Nathan Griffiths
Department of Computer Science
University of Warwick
United Kingdom

September 5, 2005

Abstract

Computer programming lends itself to automated assessment. With appropriate software tools program correctness can be measured, along with an indication of quality according to a set of metrics. Furthermore, the regularity of program code allows plagiarism detection to be an integral part of the tools that support assessment. In this paper, we consider a submission and assessment system, called BOSS, that supports coursework assessment through collecting submissions, performing automatic tests for correctness and quality, checking for plagiarism, and providing an interface for marking and delivering feedback. We present the results of evaluating the tool from three perspectives — technical, usability, and pedagogy.

1 Introduction

The "BOSS" Online Submission System has been developed over a number of years, as a tool to facilitate the online submission and subsequent processing of programming assignments [1, 5]. In the academic year 2004–5 an opportunity was taken to evaluate BOSS with a view to identifying what changes could usefully be made to improve the software.

Our evaluation is restricted to its use at the University of Warwick, since — BOSS being an open source product — we do not have accurate information as to which other institutions have deployed BOSS, and what local changes they have made to the software.

1.1 Methodology

We employed a combination of techniques in order to gain information to assist us. These included:

- an analysis of end of module questionnaires;
- interviews with staff who have been involved in the BOSS marking or management process;

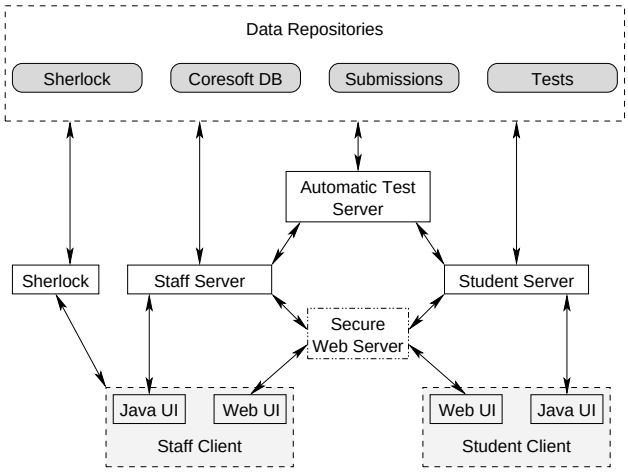

Figure 1: Overview of the BOSS architecture

- interviews with a representative group of students; and
- an analysis of the contents of the database used by BOSS.

We divide this report into three main sections. First of all, we present a technical evaluation of the software, and provide evidence which indicates that the software is now stable, and that remaining issues relate principally to the lesser-used dialogues within the staff clients.

Secondly, we discuss the usability of the software, and present a new user interface which is grounded in usability "best practice", and informed by structured interviews with staff and students [3].

Finally, we consider the pedagogy of BOSS, and argue that its original objective of being a "pedagogically neutral" tool has been achieved.

2 Technical

BOSS is designed to be a reliable, secure and user-friendly electronic coursework submission system. The current incarnation of BOSS began development in 1999 and has steadily evolved into its current form. We summarise the development and the improvements made to the software.

As evidence of the stable nature of BOSS, we describe the major design decisions that have influenced its development and how they have led to improved reliability and security. We also describe how BOSS was used in the academic year 2004–05. We outline the outstanding issues and explain why they do not pose major obstacles to general use.

2.1 History of Development

The original version of BOSS, created in the early 1990s, was a text-based utility running under the Sun Solaris operating system. It had gradually grown from text-based operation to support a graphical interface and connections to

the central University student record databases. Due to the nature of its development this system became difficult to manage and expand, and an updated version of the system was required. Development of the new BOSS began in 1999 and it was deployed in October 2000. The updated BOSS was designed for modern programming languages and assessment practice.

This new version, open-sourced in 2001, was coded with Java allowing for greater cross-platform compatibility. The design splits the student, staff and testing functionality into three separate servers with two client interfaces, one for student and another for staff (see Figure 1). The development initially focused on the student submission functionality to replicate the original system. Staff marking and automatic testing functionality followed later. The three servers are:

- the student server — accepting coursework submissions from students;
- the staff server — only accessible by staff for marking purposes; and
- the testing server — used by student and staff servers to run automatic tests.

Dividing functionality between separate servers has provided technical and organisational benefits. It has helped ensure that problems with the marking or testing systems do not directly affect the submission of coursework. Also, the student server and client have restricted access and limited functionality to help reduce the risk of a security breach. After deployment, BOSS was soon being used by hundreds of undergraduate computer science students to electronically submit their coursework, and this large user base has allowed us to quickly identify and resolve any problems with the software.

A plagiarism detection component called Sherlock [4] was added to BOSS in order to aid in the analysis of coursework. The plagiarism detection can work in two modes, one for program source code and another for natural language assignments.

Initially, we had significant problems with the stability of the testing servers. We found that students used the automatic testing interface at much earlier stages in their programming than we anticipated. Typically, this led to incorrect and incomplete programs being executed on the test server, and programs consequently consuming large amounts of memory and resources. After noting this behaviour, we added more aggressive routines to the testing server to cope under heavy test loads and better means to deal with errant processes. Our recent experience has been positive, showing much improved reliability and stability of the testing server in periods of heavy load, particularly when we are close to a deadline for coursework submission.

The design of BOSS with clearly defined boundaries between clients and servers has allowed new client interfaces to be introduced without requiring changes to the back-end systems. As web-based interfaces have become more prominent, these are have been developed to use the existing back-end servers and run alongside the Java GUI client. Servers were written accepting the possibility that a rogue client may connect. The interface between server and client is designed to help prevent malicious manipulation of submissions through stringent security measures and by exposing only the necessary information to the client. In recent years, students have come to prefer and expect a web

interface, accessible from anywhere on the Internet, to submit their work. A web interface was developed and has quickly become dominant over the Java GUI interface. We armour all connections between clients and servers with SSL to ensure data privacy.

2.2 Evaluation of BOSS in 2004–05 at Warwick

As evidence of the effectiveness of BOSS we present evidence of the use of BOSS in the academic year 2004–05. BOSS is designed to develop a closer link between staff and students. The technology in BOSS helps staff communicate more effectively with students so that their time can be spent on the educational aspects rather than organisational tasks.

With a paper based submission system, considerable time is spent organising the submission process. The technology in BOSS aims to remove this burden from staff members allowing their time to be spent more effectively. In 2004–05, the performance of BOSS can be summarised:

- over 5500 coursework submissions received electronically;
- no major outages, or downtimes impacting submission deadlines; and,
- no (known) security breaches.

BOSS also allows students to resubmit their work multiple times. The administrative burden of doing this with paper submissions would be substantial but with BOSS this is easy. We can examine the coursework submission for CS120 Programming Laboratory where 42% of students resubmitted their work, yet it required no further effort from the course manager. BOSS automatically selected the most recent submission and presented this at the marking stage. The number of resubmissions per student is presented in the graph in Figure 2.

There are some outstanding issues with BOSS, specifically with the staff client. Several staff have requested improvements to the interface relating to marking assignments, and some of these are addressed in later sections. Although these problems are important they do not impinge upon upon the ability to accept student submissions and store them accurately and securely, nor do they affect the accuracy and stability of the test server.

3 Usability

Ease of use is important for a tool such as BOSS, which receives heavy usage, and for which mistakes during the processing of data can have very serious consequences.

In this section, we report on our evaluation of the usability of the system, from the perspectives of its different users, as evidenced from a series of semi-structured interviews, from which all the quotes in this section are taken. The interface is also heuristically evaluated against a list of guidelines and design principles [2, 6]. This work has been used to design a new web interface for BOSS which addresses the issues highlighted by the evaluation [3].

BOSS users are divided into 2 main groups:

- student users, and

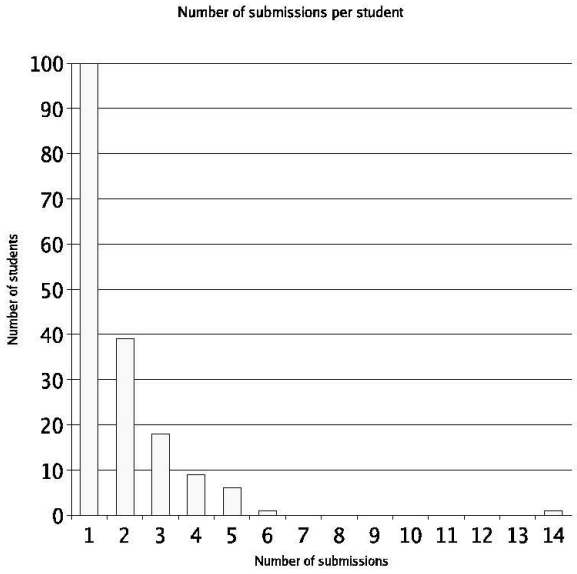

Figure 2: Number of resubmissions for the CS120 Programming Laboratory course

- staff users.

The staff roles are further subdivided into:

- administrator,
- module manager,
- marker, and
- moderator.

The BOSS Online interface is structured such that users' access to the system services and functionalities vary according to their user roles. For example, as shown in Figure 3, a fictional student user called Stephen Lee has limited access to the system, whereas the staff user shown in Figure 4 has access to all the features of BOSS.

The style used, which is reminiscent of an Apple Mac, is unusual. Both the student and the staff client are aesthetically pleasing, with minimal use of colours and lots of white space, which reduces the visual noise that may distract users from executing their tasks. Most of the pages have a clear and intuitive layout by putting most of the page content within the display area of the screen. At the default browser setting, the font style is appropriate and the use of images is kept to minimum. Page content is easily distinguished from the background and does not take long time to load.

3.1 Student client

All student users interviewed mentioned that they use BOSS only when they need to submit their assignment electronically, or to run automatic tests on

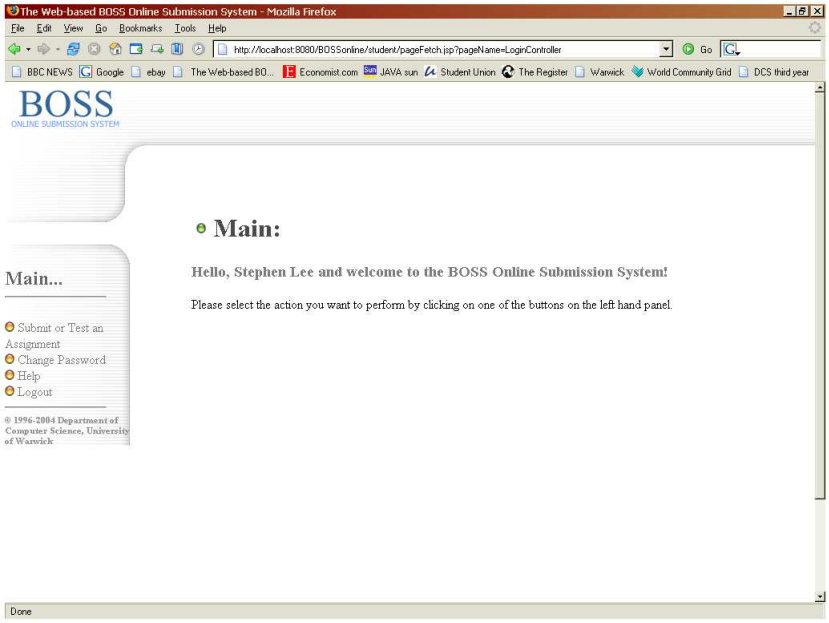

Figure 3: Student user

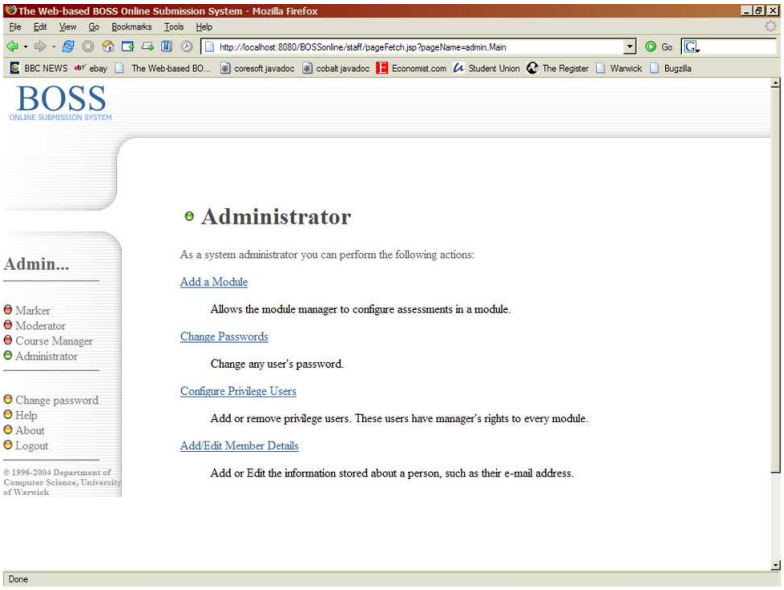

Figure 4: Staff user

their assignment prior the submission. All interviewed student users indicated that they did not have a problem learning to use the student client. Some students explained that a reason for this is their prior experience using other web applications.

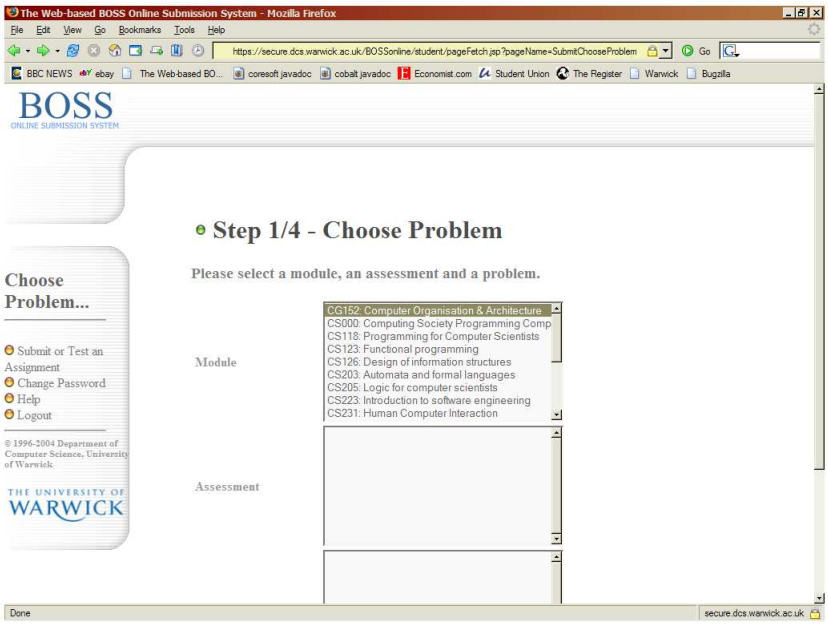

Figure 5: Scrolling on low resolution screen (i)

Students felt that "it looks nice", and compared to the previous interface (2002) that the new interface "looks nicer and more friendly". One commented that "I prefer clean design, the less graphics the better", and noted that the web interface is "light-weight and quick".

However, the web-pages of the client interfaces are often long, and one of the markers noted that "it looks OK on a large screen, not on a laptop". As shown in Figures 5 and 6, users with screen resolution lower than 1280 x 1024 will have at least a quarter of the page hidden below the fold of the screen. This increases excise for the users as they often need to scroll down the page to read all of the page content.

A specific issue raised was the lack of immediate feedback when students submit an assignment. Instead of generating a confirmation page *on the browser* indicating the receipt of a student's assignment immediately after a submission is made, an email is sent to the student's registered (University) email address. The receipt of emails by the students is often delayed. Some students mentioned that they usually make 2 or more attempts to submit the same assignment "just in case" the BOSS server did not receive the first one.

3.2 Staff client

The staff client received more focused feedback from the staff users. Almost all staff users demonstrated their need to look at the information presented on the interface before deciding on the sequence of actions that need to be performed by the system. They considered that the interface is more geared towards executing actions rather than informing them about the state of the modules and the submissions they are managing. For instance, module managers feel that they

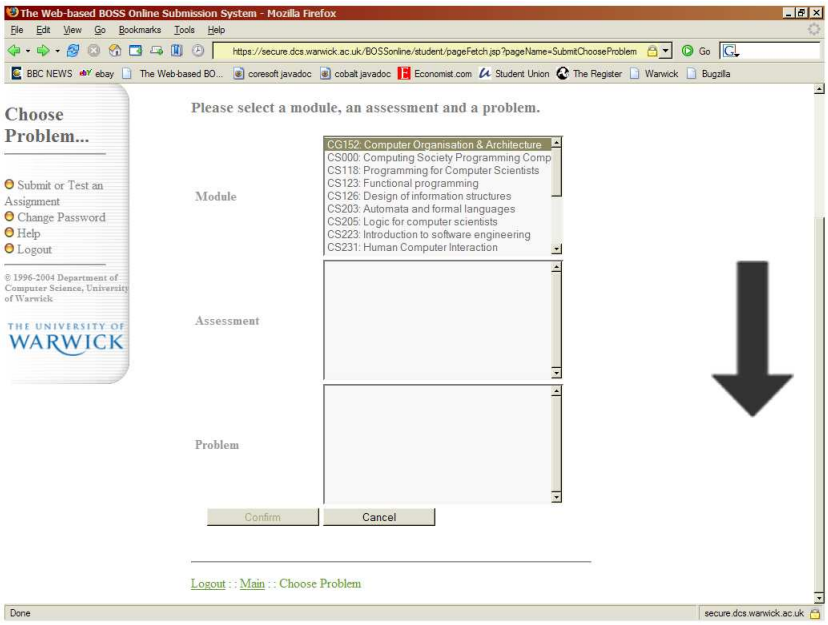

Figure 6: Scrolling on low resolution screen (ii)

have problems knowing how many students have submitted their assignment for an assessment and the status of an individual submission. One marker complained that it is difficult to correct a mistake made in a marked submission as he often needs to go through a lengthy process of finding that particular piece of submission before correcting it. The marking process sometimes becomes more time consuming as he has to click through many pages. Consequently, some of the staff users consider that BOSS is complicated to use.

The result of heuristic evaluation relates the poor information presentation of the staff client interface to the web site structure. The existing client uses a depth-emphasizing site structure rather than a balanced combination of depth-emphasizing and breadth-emphasizing. The advantage of using a depth-emphasizing site structure is that it expands the size of the site indefinitely by storing information in many levels of the site. Information is revealed gradually and other task buttons can be found in different levels when users click through the pages. However, this approach may fail if it is adopted by an application like BOSS which has incorporated many features and functionalities targeted at different groups of users. Since a depth-emphasizing site structure lacks linear navigation, many facilities of BOSS are hidden away in different levels of the site hierarchy, and users often do not have immediate access to them.

3.3 Consistency issues

Heuristic evaluation and interviews also suggested that although BOSS Online has achieved an overall consistency in look and feel by the use of stylesheets, some details have been neglected in the design. For example, the naming and positioning of buttons which have the same meaning in the interface are not

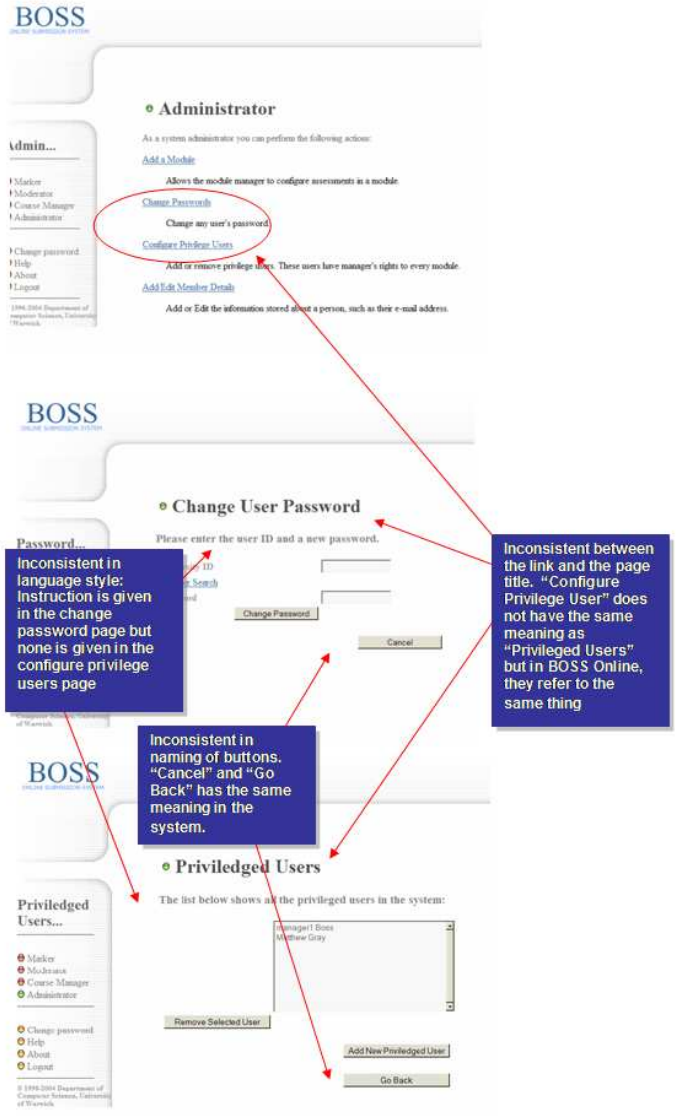

Figure 7: Inconsistency of staff interface

consistent. As shown in Figure 7 the buttons for stepping back an action is labelled as "cancel" or "go back" in different pages even though they all perform the same function. Figure 7 also highlights some other consistency problems, such as language style.

In a set of multiple data entry screens, inconsistencies exist when buttons are not placed at the same place in every page. As shown in Figure 8, buttons

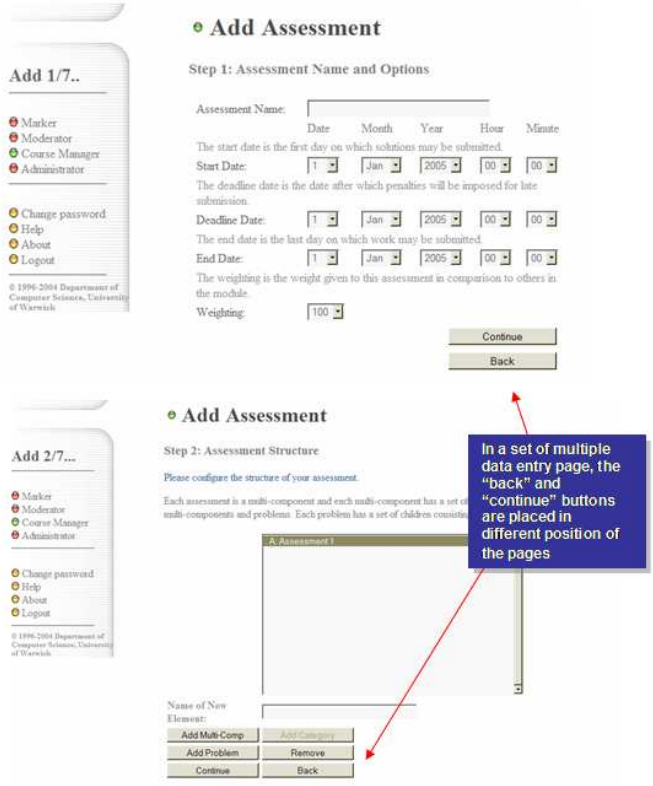

Figure 8: Inconsistency of buttons positioning

are sometimes placed in the lower left and sometimes in the lower right of the form.

3.4 Navigation issues

Navigation problems are also observed in both the staff client and student client interface. Access to other pages is highly sequential, and there are few handy shortcuts in place for users to access other useful functions of the system. They often have to spend time clicking through many pages. As a result, users are sometimes "lost" in the system. Although both interfaces provide breadcrumbs at the end of every page as a navigation aid, the pages are often long and users with low screen resolution will not discover the existence of these breadcrumbs unless they make the effort to scroll down the page. As a result, users are not informed about their location within the site.

3.5 Historical and pragmatic considerations

Information gathered from the developers and the system designers show that both the student and the staff interfaces have been implemented pragmatically.

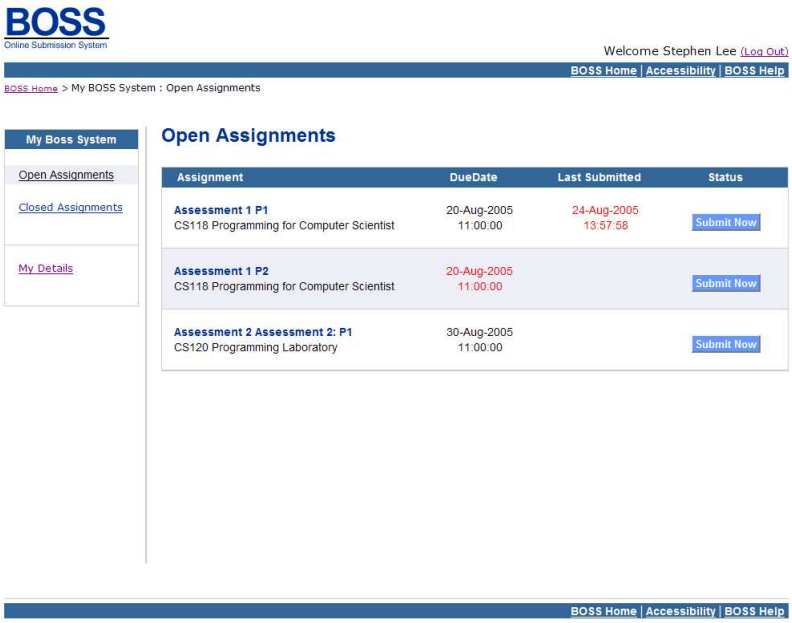

Figure 9: New student interface

BOSS was designed in order to satisfy the need of the Department of Computer Science to facilitate the submission and the marking process of student assignments. Development over the years has been incremental, and new features have been added to BOSS based on users' requirements while retaining the existing functionalities. The system is currently voluntarily supported and maintained by a group of staff users.

3.6 New Design

Following the usability evaluation, we have proposed a redesign of both student and staff client interfaces. The new student interface has been implemented over summer 2005, and the implementation of a new staff interface is in progress.

The new student and staff clients have been designed separately as each user group has different task requirements. Although the new interfaces retain much of the function sets already defined in the existing clients, both use a combination of depth- and breath-emphasizing design so that they are more geared towards information presentation. The advantages of this structure are that it shows many levels of the site hierarchically (depth-emphasizing design) and at the same time lists all the alternative options for each level linearly (breath-emphasizing design). Figures 9 and 10 show the first page of the student client and the staff client after the user has logged in to the system.

As shown in Figures 9 and 10, each page of the new clients uses minimum page elements to aesthetically improve the interfaces and to reduce visual noise. The colour scheme is chosen carefully. Both interfaces use a monotone colour scheme with blue as its single hue. Other colours used in the page are derived from the blue using different levels of saturation, tint and shade. Following con-

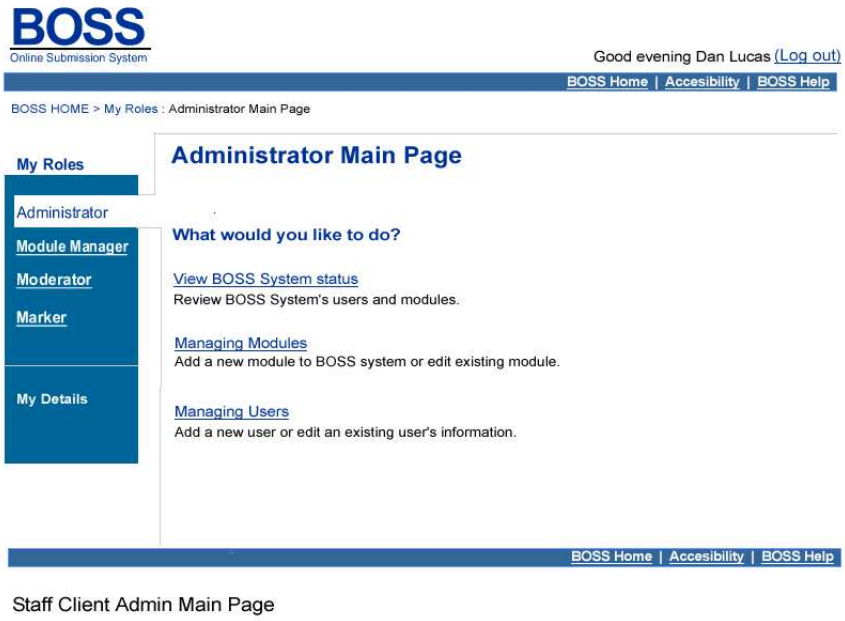

Figure 10: New staff client

ventional design rules, red is used as the colour for error messages or important information. To make sure that the pages are readable, both interfaces use a combination of dark blue on white, black on white, and white on blue to provide sufficient text contrast.

A set of persistent navigation aids signpost the location of the user within the site. The navigation aids include the BOSS logo on the top left corner of the screen, a side navigation bar on the left of the page, and 2 blue stripes that run across the screen at the top and the bottom of the page. Apart from serving as a primary navigation mechanism, the BOSS logo on the top left corner of the page is a convenient escape route for the users. Users will be logged out and redirected to the BOSS Online homepage if they click on the logo. The side navigation bar is also a consistent page component, which contains links to the functions that are most likely to be required by the users. Shortcut links to access the help page and the accessibility page are positioned at the right end of the top and bottom blue strips. Apart from the above mentioned navigation aids, "breadcrumbs" provided as a secondary navigational tool, since they do not take up much space on a page and are useful in displaying hierarchical information.

Most of the right of the page has a white background, and all content text is left justified and uses only one type of font family. To further utilize the usable screen estate, the most relevant information is placed at the top of the page so that when a page is loaded, users will be able to determine immediately which page they are on.

Style sheets are used to achieve consistency of the formatting and style of the page elements such as text, buttons, links and images. To further enhance the uniformity of the web application, many page elements are used repeatedly.

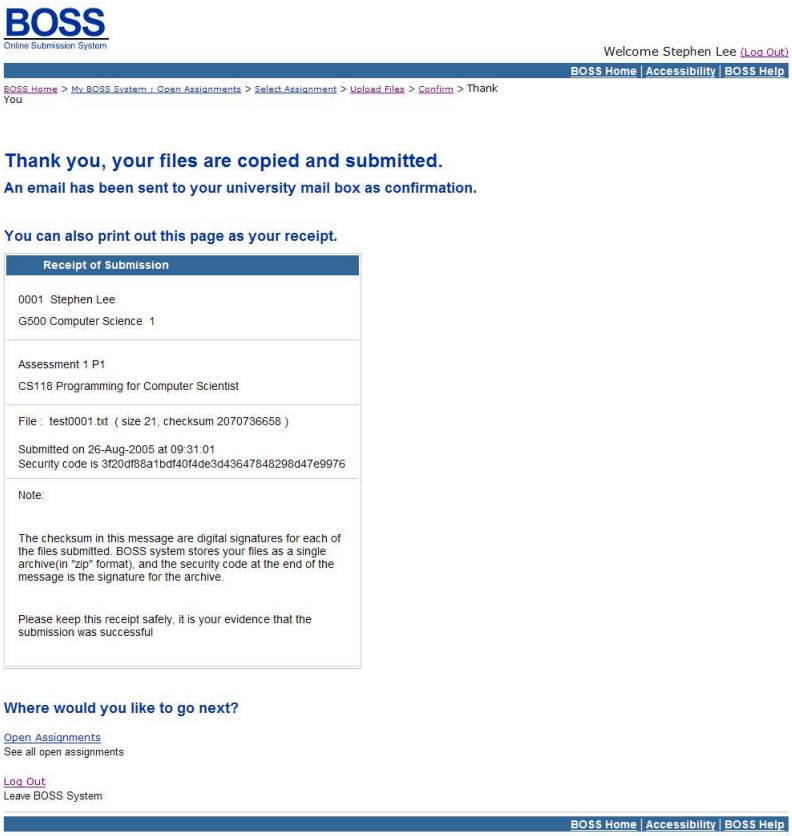

Figure 11: Submission confirmation page

Repetition of page elements throughout the interface reinforces the site structure and at the same time, provides a "corporate identity".

Figure 9 can be compared with Figures 5 and 6 shown in the previous section. Instead of clicking through the selection boxes on the page and then selecting the required module, assignment and problem before submitting an assignment, students are given a list of problems along with their corresponding module, assignment, deadline (etc.), arranged clearly in a table. This not only increases the efficiency of students using the system, it also reduces the chance of students making a mistake. Students are less likely to choose the wrong combination of module, assessment and problem if the number of selections they can make is reduced.

In order to provide immediate feedback to the students, a confirmation page is generated after a student has submitted an assignment, indicating that a submission is successful and that files have been uploaded are copied to the server. An example is presented in Figure 11. Students can either save the page electronically or print it out as an official receipt of the submission.

3.7 Summary

The BOSS client interfaces reflect the functionalities and the features of the server, serving as an effective front-end for the users. The student interface is generally regarded as simple and straightforward to use, but improvements have been identified. The staff interfaces are much more complex, and for historical reasons are depth-emphasising, with some inconsistent data presentation and navigation. Since the staff client is relatively infrequently used, this is not unexpected.

4 Pedagogy

BOSS is intended to be "pedagogically neutral". We provide an environment in which a secure online submission platform can be used with existing tools, or alternatively the extra assessment utilities in BOSS can be used in conjunction with them to offer increased functionality [5].

In order to evaluate BOSS against this objective, we identify two sources of evidence. First of all, we examine its patterns of use to test the hypothesis that the tool is indeed used with a *variety* of existing tools. Secondly, we consider the comments made by interviewed staff who have used the system which relate to its educational value.

4.1 Patterns of Use

The Department of Computer Science does not prescribe how an individual academic should manage his/her own modules, and each academic is free to use BOSS as much or as little as they desire. An analysis of the patterns of use over the fifteen modules which used BOSS highlights the different individual approaches taken to the adoption of the software.

Seven of the modules used the system as a collection mechanism only, allowing the assessment process to be supported by other tools, such as spreadsheets. The reasons for not using the assessment features include:

- assignments are essay-based (2 modules);
- assignments relate to formal methods, and don't require coding (1 module);
- assignments take the form of group projects, which contain added administrative complexity not appropriate for BOSS (1 module);
- assignments require students to code, but the nature of the programming paradigm (for example, declarative) requires tests to be performed on students' code which is not easily modelled as "expected output for given input" (3 modules).

One further essay-based module used BOSS for collection, and the plagiarism detection software only.

Of the remaining seven modules which did use the majority of the software features, all involved students programming using procedural or object-oriented languages, as illustrated in the following table, which identifies:

Module	Auto tests?	Plagiarism detection?	Online marking?	Code type
CG152	no	yes	no	simple C++
CS118	no	yes	no	simple Java
MA117	yes	no	yes	simple Java
CS120	yes	yes	yes	simple UNIX Shell and Perl
CS126	yes	yes	no	intermediate Java
CS203	no	yes	no	Prolog
CS237	yes	yes	yes	advanced Java

Table 1: Programming modules using BOSS

- whether the automatic tests were used,
- whether the plagiarism detection software was used,
- whether the marking process was conducted *online* within BOSS, and
- the type of code which the module was delivering.

4.2 Plagiarism detection

The assessment process involves ensuring that students are marked on their own work, and the prevention and detection of plagiarism is therefore an important part of the process. All staff who used the Sherlock plagiarism detection software commented that it was effective, and in each module disciplinary action was taken on a number of students as a result. Use of Sherlock as a regular part of the assessment process, and its associated visibility to students, has resulted in effective deterrence, and although plagiarism has not been eliminated, instances of it have been reduced on large programming modules to typically less than 5% [4].

The one module which did not use the plagiarism detection component, offered introductory Java programming for science students (principally mathematicians and physicists). The teaching style adopted involved the students being presented with "templates" which shielded them from the complexities of the object-oriented paradigm, and enabled them to concentrate on writing and editing relatively small amounts of procedural code appropriate to their disciplines. The use of automatic plagiarism detection software is not effective on data where the amount of individually contributed code is small.

4.3 Automatic testing

For four of the seven module assessments it was appropriate to use the automatic test harness together with the on-line marking dialogue (these were first and second year modules using Java or UNIX Shell code).

The final three modules were supported by alternative assessment and management regimes. The staff involved in each of these modules were skilled in

alternative software products, and wished to use facilities which would be inappropriate to include in BOSS. For example, one academic remarked:

> "A spreadsheet is very flexible, you can sort it in many ways, do lots of things on it, colour it, and so on. You can highlight it, give private comments, comments for yourself,... do layout, create graphs...".

The pedagogic effectiveness of BOSS is that of the educational paradigm it is used to support. The simplest non-trivial use of BOSS — and one of the original motivations for its creation — is the incorporation of black-box automatic tests into the assessment process for programming modules. Conversations with the module leaders suggest that the time necessary to devise and deploy a set of automatic tests is typically one or two hours, and that the time taken to mark a single student's submission may be as low as a couple of minutes, and this is strong evidence that the approach is administratively effective.

All the staff who used the software for its automatic tests and the online marking agreed that the black box paradigm worked successfully, although setting up those tests was regarded as complex. This is in part a usability issue, but is also a comment on the inherent difficulty of writing a correct test harness together with a clear and comprehensible specification which will enable students to understand what they are required to code.

Incorporation of program metrics into the software is recent, and has not yet been fully incorporated into any academic's marking scheme. Similarly, although JUnit unit tests have been included in the system functionality, the technology has not yet been taken up in any module at Warwick.

The student view has been generally very positive, and comments in end of module feedback sheets indicate that it is regarded as an efficient and convenient system to use.

A specific comment often made by students is that the automatic testing is "too fussy". In an environment where we encourage students to take a formal engineering-based approach to software development, it seems appropriate that a tool such as BOSS is precise, and this student view might be interpreted positively. However, there are drawbacks to such an engineering approach, and the following students' views about the use of automatic tests are not unusual:

> "Automatic tests are unfair against those who have tried and only just failed to reach the required outcome."

> "Didn't like the auto tests — too picky with spaces etc."

Furthermore, writing specifications which are clear to the student audience, and associated test cases which match them, may be unexpectedly difficult, as this student has identified:

> "... the specifications were very unclear, thus leading to the failure of certain automatic tests and unfair marking."

The number of tests, and their place in the marking scheme, is another focus of discussion. It is neither feasible nor desirable to provide a complete testing suite for use by students whilst they are learning the basics of programming, since the number of tests would be prohibitive, and communicating the results

to students correspondingly problematic. However, it may be desirable to allow students access to some automatic tests to assist them in their program development. It may also be desirable to reserve some tests for the marking process in order to encourage students to think through the specification thoroughly. This is a strategy which is difficult to justify to students:

> "I don't understand why there are more tests on the assignment. Can you give us all tests when you give us assignments. Then we can know what you want us to do."

More detailed staff feedback (via interviews) suggests that the functionality of such a system supports colleagues' requirements, but identifies the client dialogues as being over complex.

Our claim of "pedagogic neutrality" is supported by the use of BOSS as a vehicle for innovative pedagogic approaches. For example, the software has successfully been extended to provide a package to support peer assessment [7].

5 Conclusion

The use of BOSS over a period of years has demonstrated the effectiveness of a focused tool which addresses the requirements of assessing students' programming skills. The inclusion of a generic database schema and plagiarism detection software, together with a platform-independent client-server architecture, provide a foundation which is adaptable to changes both in technologies and in pedagogic requirements.

Evaluating the system, as implemented during the 2004–5 academic year, sends us several messages.

At a technical level, the system is robust, and has supported many modules with few interventions needed. Weaknesses involve the import of data from IT Services, and the accuracy of the data as maintained centrally.

Pedagogically, BOSS fulfills its aim of "pedagogic neutrality", and has allowed staff either to use the BOSS assessment and course management features, or alternatively to use it as as a mechanism to support their own educational tools and style. There is a friction between users who consider it to be over complex, and those who would welcome the addition of further features and functionality, suggesting that its feature set is probably sufficient at this time.

There are weaknesses with the BOSS interfaces relating to consistency and to navigation issues. The usability of BOSS for students is evidenced as being good, although improvements have been identified relating to its efficient use. The structure of the current web interface for staff users is regarded as overly complex due to its depth-emphasising structure, a feature of the software which is due to the incremental nature of its development.

References

[1] BOSS. The boss online submission system. Online, 2004. Available: http://boss.org.uk/ (Accessed 19 December 2004).

[2] MIT Usability Group. Usability guidelines. Online, 2005. Available: http://www.mit.edu/ist/usability/usability-guidelines.html (Accessed: 30 March 2005).

[3] PeyShan Heng. BOSS online usability evaluation and interface redesign. BSc Project Dissertation, University of Warwick, 2005.

[4] Mike Joy and Michael Luck. Plagiarism in programming assignments. *IEEE Transactions on Education*, 42(2):129–133, 1999.

[5] Michael Luck and Mike Joy. A secure on-line submission system. *Software - Practice and Experience*, 29(8):721–740, 1999.

[6] Jakob Nielsen. useit.com. Online, 2005. Available: http://www.useit.com/ (Accessed: 30 March 2005).

[7] Jirarat Sitthiworachart and Mike Joy. Effective peer assessment for learning computer programming. In *Proceedings of the 9th Annual Conference on the Innovation and Technology in Computer Science Education (ITiCSE 2004)*, pages 122–126, 2004.