FrontDesk: An Enterprise Class Web-based Software System for Programming Assignment Submission, Feedback Dissemination, and Grading Automation.

Mike Maxim
Carnegie Mellon University School of Computer Science
5000 Forbes Avenue, Pittsburgh, PA 15213
USA
mmaxim@cs.cmu.edu

Abstract

The problem of managing an effective relationship between course staff and students in large programming courses admits no trivial solution. Students often complain of lack of feedback, slow assignment grading times, and a gap in communication between them and the course staff responsible for evaluating their work. In addition, course staff feels powerless to help because of the complexity and sheer numbers of students involved in such courses. In this paper, we describe our Web-based distributed application, **FrontDesk** that attempts to bridge the feedback and communication gap that these courses suffer from. FrontDesk provides tools for students to submit their work through the Web and to receive rich, informative feedback. In addition, it provides course staff with the ability to give such effective subjective feedback for large courses and to automate objective programming assignment testing in a flexible, distributed, and efficient manner.

Key Words

submission, automatic, grading, feedback, enterprise, web

1. Introduction

There have been many proposals for online submission and grading systems for computer science courses. Such systems are created in order to simplify and improve the process of producing objective correctness tests and define grading criteria for subjective evaluation of code. Both the course staff and the students aim to benefit from this improvement. Course staff is able to concentrate more on the actual evaluation process and less on the bureaucratic nature of processing a large number of submissions. In addition, staff can formally define evaluation criteria in terms of both objective and subjective tests in order to increase grader consistency and efficiency. Objective tests can be automated through the use of automatic test suites written with tools like JUnit. Subjective grading is enhanced through the introduction of common criteria and comments to which all graders must adhere. Students benefit from the increased access to grading feedback. Thorough objective tests let students know where and why their program fails. Subjective analysis enables students to learn about how to create code in the correct manner. Grading systems aim to make the ideal interaction of course staff and students through the evaluation process described a practical goal. [1]

When the number of students enrolled in a course is high, stress is placed on the submission and grading systems. Many submission systems fail under the pressure of hundreds of submissions. Because of this, it is necessary for these systems to be defined using enterprise software design techniques, and to mobilize the correct amount of hardware support behind the solution.

In this paper, we describe our system, **FrontDesk**, which we believe accurately addresses the main issues involved in the creation of a software system to manage computer science courses by providing a submission and feedback portal. Due to the degree of complexity and potentially high load of submissions and grading jobs, FrontDesk is built on a Web-based multi-tier enterprise architecture that is more cognizant of its role within a larger, potentially university wide portal or course delivery system. We first discuss the exact specifics of the problem domain, and then proceed to explain both the solution of FrontDesk, and its software design.

2. Problem Domain

Course systems need to implement subsystems to handle evaluation of submissions, the submission of work for students, and the management of the course for course staff. We first define the main stakeholders in the system.

2.1 Stakeholders

- Student
- Course Staff
    - Teaching Assistant
    - Instructor

2.1.1 Teaching Assistant

We consider the teaching assistants to be the main administrators of the evaluation of student submissions. That is, teaching assistants are responsible for defining both the objective and subjective grading criteria for assignments. Each of the following below is a problem TAs face when presented with this challenge.

- Consistency – Consistency in grading across sections and across graders is important for subjective evaluation. TAs need tools, particularly the ability to formally define subjective grading criteria, to make this process cohesive and easy for graders to remain consistent.
- Efficiency – Making objective correctness tests is difficult in the *ad hoc* setting. As a result, TAs are slow in producing these tests and the grading process suffers. Because of this, it is incumbent upon the grading system to deliver toolkits to enable the fast creation of test cases. In addition to creation time of tests, execution time is also a major concern [2]. For instance, testing a program that performs a compression algorithm on 30 or 40 files can take minutes to finish. Multiplying this by the number of students in the course make the importance of a distributed, parallel testing platform obvious [3].
- Robustness – TAs should be able to deliver rich, informative feedback to a student that accurately reflects their feelings about the student submission. The system should provide mechanisms to make dissemination of large amounts of subjective and objective grading results practical.
- Groups – Many times assignments will be administered with students forming into groups. When groups span multiple sections and graders, then the grading process slows down to sort out who is going to grade what. The system should allow for students to form into groups on the system itself, and allow graders to process these submissions as group submissions, not as submissions that happen to be identical for more than one student.

2.1.2 Student

Students want three things from the system: easy submission, informative feedback, and submission verification.

- Submission – Conceptually, submission is not a difficult task to accomplish. However, it is necessary to have students submit their work in the same environment they receive feedback. The integrated aspect of submission and feedback reduces the amount of elements students need to learn to operate in the course.
- Feedback – With the increased ease of feedback creation for TAs, students are now able to access this feedback to improve the learning process. Students should be given an unmitigated view of the TA generated feedback, and be able to learn from this without going through much difficulty deciphering TA comments or complaining about grading inconsistencies.
- Verification – Students are very nervous about their grades and submissions, so it becomes necessary to provide them with a level of assurance about their submission. The submission system should pre-test student code so they have an idea their code performs like they think it should. Students should also be able to browse their submission files for verification.

2.1.3 Instructor

Instructors are concerned with defining course content and assignment material, not in the administration and evaluation of the assignments. For this reason, we feel any course system must make it possible for instructors to delegate these responsibilities to TAs.

- Permissions and Settings – Instructors should be able to create permissions and course settings to control the behaviour of subordinate TAs and customize the course to their liking.
- Statistics – Instructors are also interested in statistics about the grades and submissions. For instance, an instructor may want information about how many people have already submitted their work. They may also want information about grade data.

2.2 Volume

When dealing with multiple large courses, the volume of submissions and automatic grading jobs increases dramatically. In order for a course system to handle this increased load, enterprise design and implementation techniques must be used to produce the system. Design techniques used to produce state of the art information and transactional systems used in today's most successful firms must be applied to the course management and grading scenario if an effective deployment is to be achieved.

3. The FrontDesk Solution

FrontDesk is primarily a Web-based system built on ASP.NET. The Web interface serves as a portal for course staff and students to access the submission and grading systems. Along with the Web interface, there exists a **Testing Center** application designed to run on multiple platforms. The testing center runs separate from the Web server, and provides the functionality of actually executing test suites and harvesting results. FrontDesk also exposes most of its functionality through XML web services, allowing great potential for integration with existing systems.

3.1 FrontDesk Assignments

The main item of course content in FrontDesk is the assignment. Assignments consist of a grading schema composed of both subjective and objective tests, written content such as write-ups, student submission groups, and assignment specific settings. All grading and submission actions are performed with respect to an individual assignment. The next sections discuss each subsystem of FrontDesk, and how each works to support the students and course staff during the assignment's lifetime.

The scenarios below are what the subsystems described in the next sections attempt to implement.

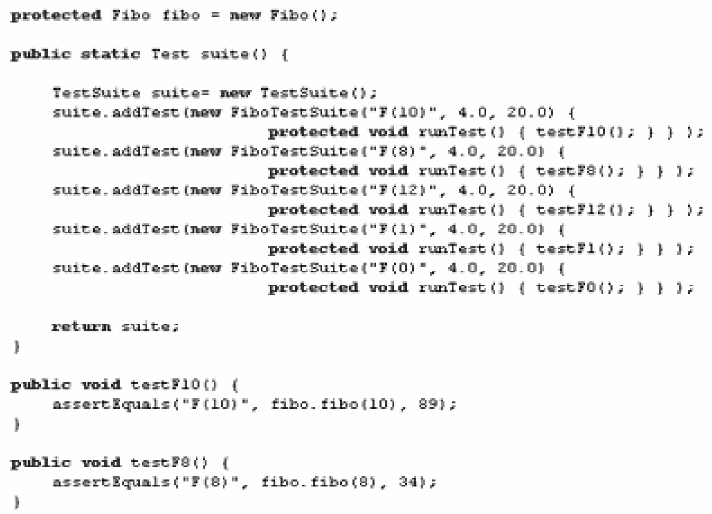
- Course Staff
  - Assignment Creation – Course staff create assignments by defining the grading schema in terms of objective and subjective tests. In addition, assignment write-up documents are made available.
  - Assignment Evaluation – After the submission process has ended, the course staff evaluates the student submissions. Staff accesses the objective testing system to obtain the objective correctness test results. Drawing upon these results, staff accesses the subjective grading system to evaluate the quality and style of student code. Once this has completed, the results of the submissions are made available to the students.
- Students
  - Submission – Once a student has completed their assignment, they access FrontDesk to submit their work. Depending on whether or not they worked in a group, the student forms into a submission group.
  - Accessing Feedback – After the course staff has completed the grading process, students access FrontDesk to receive grading information about their submission.

During the discussion of each subsystem, we maintain an example assignment based around the computation of Fibonacci numbers to make the features of FrontDesk more concrete.

3.2 FrontDesk Objective Testing System

Objective correctness testing is an important aspect of the grading of any computer science assignment, however, it is often difficult to implement. FrontDesk provides services for making the creation and execution of objective tests as easy as possible. Test suites are written to operate with the testing center application. In order to make this process flexible with respect to programming language and platform, test suites communicate with the testing center in a loosely coupled manner. Communication between test suites and the testing center is made possible by the testing center hooking into the standard output stream of the test suite process. Test suites are required to output an XML result description conforming to a FrontDesk defined XSD schema. The schema defines precisely how point deductions are reported. Test suites must report through the XML result a list of failures and errors generated by the submission being tested. A failure is defined to be an expected malfunctioning of the submission. In our Fibonacci example, this would correspond to testing if the submission correctly computes *F(10)* = 89. On the other hand, an error is defined to be an unexpected malfunction. In Java, this corresponds to something like a NullPointerException. Failures and errors are separated in order to allow staff to define different point deductions for each type of malfunction.

In order to make development of test suites a tractable task, FrontDesk provides toolkits for popular software unit testing APIs. For instance, FrontDesk provides extensions to the widely used JUnit unit testing library for Java programs. Course staff can take existing JUnit tests and convert them to FrontDesk test suites by changing very few lines of code. All of the XML output is taken care of by the extensions to the JUnit testing API. Similar toolkits exist for various other popular testing APIs as well as a general toolkit for custom tests. To better understand the process of creating test suites for the FrontDesk testing center, we will examine a sample test case for our Fibonacci assignment. Figure 1 shows a code sample from a JUnit test.



```
protected Fibo fibo = new Fibo();

public static Test suite() {

    TestSuite suite= new TestSuite();
    suite.addTest(new FiboTestSuite("F(10)", 4.0, 20.0) {
                          protected void runTest() { testF10(); } } );
    suite.addTest(new FiboTestSuite("F(8)", 4.0, 20.0) {
                          protected void runTest() { testF8(); } } );
    suite.addTest(new FiboTestSuite("F(12)", 4.0, 20.0) {
                          protected void runTest() { testF12(); } } );
    suite.addTest(new FiboTestSuite("F(1)", 4.0, 20.0) {
                          protected void runTest() { testF1(); } } );
    suite.addTest(new FiboTestSuite("F(0)", 4.0, 20.0) {
                          protected void runTest() { testF0(); } } );

    return suite;
}

public void testF10() {
    assertEquals("F(10)", fibo.fibo(10), 89);
}

public void testF8() {
    assertEquals("F(8)", fibo.fibo(8), 34);
}
```

Figure 1. JUnit code snippet from a Fibonacci test suite.

Using methods such as the **assertEquals()** method shown in the code above, failures and errors are generated for the submission being tested. Each failure and error is given a point value that indicates how many points are lost per failure or error. The JUnit extensions FrontDesk provides will transform JUnit exceptions thrown by the assert methods into the correct XML format for consumption by the testing center. JUnit, and toolkits similar to it, play a large role in making

FrontDesk an efficient objective testing solution. FrontDesk provides a "low-level" XML interface between test suites and the testing centers; however, it relies on JUnit to make creation of tests within this environment a simple task.

In addition, objective tests can be defined to contain dependencies amongst themselves. All dependencies of a test are executed before the main test. This is useful for making sure that a student submission is always built before correctness tests are to be run. Making a build test a dependency of the correctness test ensures the correctness test will execute properly (assuming it is correct of course).

Creation of test suites is only half of the story of objective testing. Execution of objective tests also provides a bottleneck on grading time. In order to address this problem, FrontDesk provides a distributed testing application called the testing center. Testing centers are intended to be installed on multiple workstations in order to distribute objective testing jobs. The testing centers are constantly polling the main database to check if any objective testing jobs have been requested. If a testing center finds a job that it can execute, it will proceed to download the student files and test files onto its local file system and proceed to execute the test. Once the test has completed, the testing center will log the results of the test back to the global database for consumption by both staff and students. Figure 2 shows a diagram of this process.

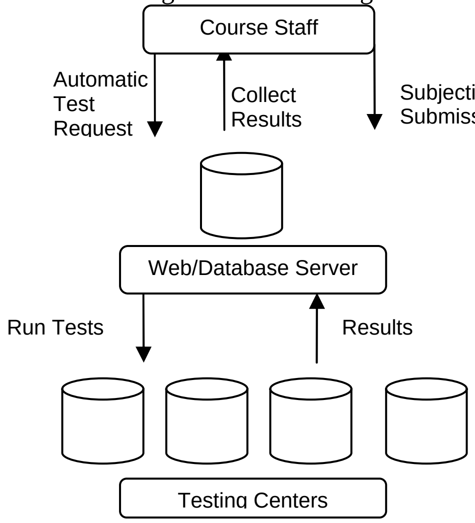


Figure 2. Flow chart for the process of starting and receiving results from the automatic testing system.

This system for objective testing addresses the efficiency concern noted in Section 2. The distributed testing center application allows for easy upgrades to test execution time. By adding an extra testing center, the time to run an entire job for the whole course is substantially reduced. In addition, the test creation process is kept simple. Course staffs can continue to use unit testing APIs such as JUnit and have these tests integrate fully with the FrontDesk testing centers through the toolkits provided. The loose coupling between test suites and the testing centers maximizes the flexibility in the types of assignments that can be run through FrontDesk.

3.3 FrontDesk Subjective Testing System

In addition to providing a rich environment for the creation and troubleshooting of objective tests, FrontDesk provides a system to give detailed and organized subjective feedback from course staff to students. The principle idea behind the FrontDesk subjective feedback system is the ability for course staff to formally define hierarchical subjective grading criteria. An example hierarchy for the Fibonacci assignment is given in Figure 3.

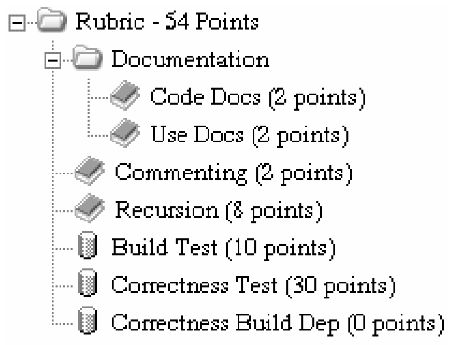
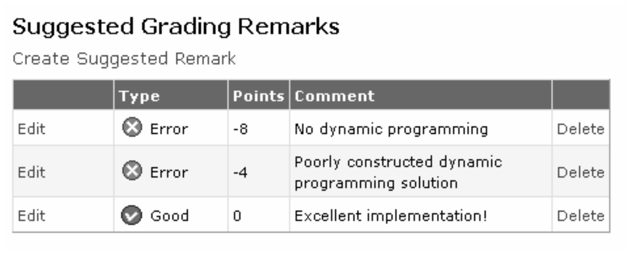

Rubric - 54 Points
- Documentation
  - Code Docs (2 points)
  - Use Docs (2 points)
- Commenting (2 points)
- Recursion (8 points)
- Build Test (10 points)
- Correctness Test (30 points)
- Correctness Build Dep (0 points)

Figure 3. A view of the hierarchical grading schema for the Fibonacci assignment.

The advantages of this approach are two fold. For the course staff, each grading member of the staff is now encouraged to fill in detailed information for each category of the subjective grading schema. The criteria are organized into a tree-view Web control which allows the staff to easily make remarks in each section of the grading schema about the current submission being graded. Each entry is allotted a point value, enabling course staff to enter in exactly how many points the student earned for that particular category. Students now have the opportunity to view the same tree-view Web control in order to determine exactly what the grader of their submission thought of the submission. They can browse each category and effectively determine issues that their submission had with regard to the particular category. This addresses the problem of providing students with subjective information under a common interface with course staff.

In order to further increase grader consistency, FrontDesk provides the option of creating pre-determined subjective remarks for each of the categories defined in the grading schema. The pre-determined comments are defined by a point value, a comment type, and a message for the student. Figure 4 shows a listing of such comments for a grading schema entry in the Fibonacci example. When a grader is assessing a submission, they can draw comments from either the pre-determined set, or make custom comments. The comments have the option of being attached to specific files and lines from the student submission files. This gives the grader the ability to link subjective grading content to specific places in the student submission. Figure 5 shows how the student can view the file-based grading comments for the Fibonacci assignment.

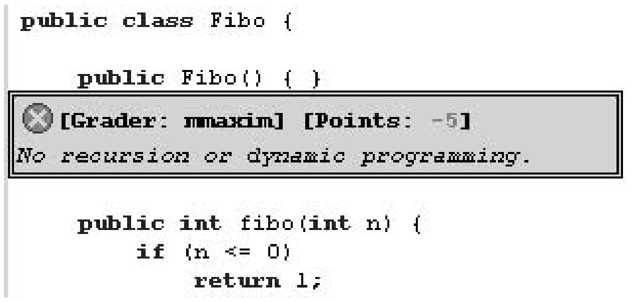


Suggested Grading Remarks

Create Suggested Remark

	Type	Points	Comment	
Edit	Error	-8	No dynamic programming	Delete
Edit	Error	-4	Poorly constructed dynamic programming solution	Delete
Edit	Good	0	Excellent implementation!	Delete

Figure 4. Suggested comments for a rubric entry.

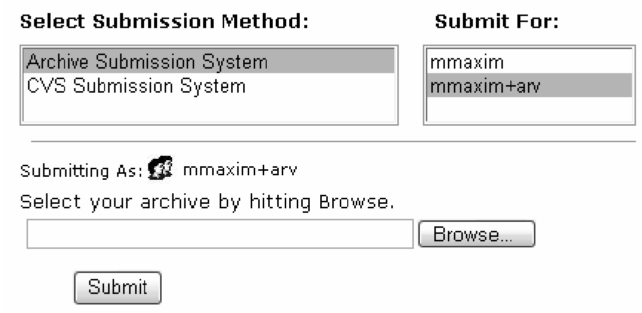


```
public class Fibo {

    public Fibo() { }
[Grader: mmaxim] [Points: -5]
No recursion or dynamic programming.

    public int fibo(int n) {
        if (n <= 0)
            return 1;
```

Figure 5. A sample in-file comment that a student can view after grading has completed.

3.3 FrontDesk Submission and Management

FrontDesk provides a Web interface for student solution submission. The Web interface allows for many different types of submission methods. Examples include archive based submission and CVS based submission allowing students to directly submit from a CVS repository. Upon submission, students are informed that they will receive results of tests that have been designated by course staff to be run when the student submission is received. Such tests are queued in the testing center system described above and the results of the tests are emailed to the students upon completion. Students are also presented with the FrontDesk Web interface file browser allowing them to browse their submission files to further validate that they submitted the correct files with the proper directory structure. Another important feature for students during the submission process is the ability to submit under a group identity. Student can impersonate the identity of the group during submission, implementing the process of submitting for their group. Figure 7 shows a portion of the Web interface for file browsing.



Select Submission Method: Archive Submission System / CVS Submission System (selected: Archive Submission System)

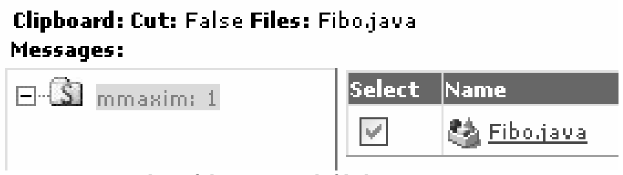
Submit For: mmaxim / mmaxim+arv (selected: mmaxim+arv)

Submitting As: mmaxim+arv

Select your archive by hitting Browse.

Figure 6. Screenshot of the FrontDesk submission options.

Students are provided with the assurance of receiving preliminary test results, and the ability to browse the server file system through the Web interface.



Clipboard: Cut: False Files: Fibo.java

Messages:

Figure 7. Screenshot of the FrontDesk file browser.

The course staff is given fine grained control over various aspects of course management, particularly administration of individual assignments. Sections and groups allow the staff to divide labor amongst the members of the staff in order to increase the efficiency of the subjective grading process. The group system employed by FrontDesk addresses the problem discussed in Section 2 regarding the confusion that sometimes accompanies the grading of group submissions. Since the submission is under one identity, one grader will be assigned to the submission. The staff is also given access to the Web interface file browser to correct minor mistakes in a student submission, as well as the main engine for the application of the subjective comments described in Section 3.3. In addition, instructors are able to define specific settings and permissions on both the course and assignment level. They can designate settings such as maximum student group size, maximum number of submissions, allow group submissions, etc. Instructors can also set permissions on the actions other members of the course staff can perform in the administrative mode of FrontDesk. FrontDesk provides detailed grade reports at both the assignment and course level. These grade reports contain the student performance across assignments. Reports can also be generated for entire sections in order to get section-wide data. Submission statistics are available to course staff during the student submission time period so that staff can monitor the patterns of the student submissions.

With the course management features described above, course staff is given an effective means to solve many of the problems discussed in Section 2. The Web interface to the underlying file system provides the ability for staff to make changes without needing special privileges to access the Web server through alternate means. Security is increased by making sure only senior course staff have the ability to do the most damage to critical student records.

4. The FrontDesk Design

As mentioned in Section 2, a large concern during the design and implementation of the FrontDesk solution was the ability to scale to meet heavy load requirements and be flexible enough to handle diverse course offerings. FrontDesk is written entirely in C# and uses the Microsoft .NET Framework extensively to implement many enterprise style features such as data source access, XML web services, and distributed transactions. The main data warehouse in the design is implemented using the data provider model. Providers can be written for many different types of data sources. They are subject to many implementation invariants and must conform to a FrontDesk specified Provider layer interface. Currently two data providers have been implemented: Microsoft SQL Server 2000 and Oracle 9i relational databases. Application specific rules (business logic) are implemented in a separate layer that sits on top of the main Provider layer. This layer, called the data access layer, implements the rules of the FrontDesk system and implements many of the features described in Section 3. Using .NET Remoting, this layer is able to be distributed across many different workstations, maximizing scalability under high load. For small loads however, performance can remain high by isolating the layer on a single workstation. The presentation layer, as described in Section 3, is a Web interface implemented using ASP.NET. The presentation layer never directly accesses the Provider layer, but must attempt all data operations through the data access layer. Testing centers are implemented upon the same architecture as the Web interface. Testing centers are able to use the same data access layer to perform operations such as requesting tests to run and reporting results. The enterprise class design makes FrontDesk a highly distributed, scalable application capable of handling multiple large programming courses. Figure 8 shows a schema of the architecture design.

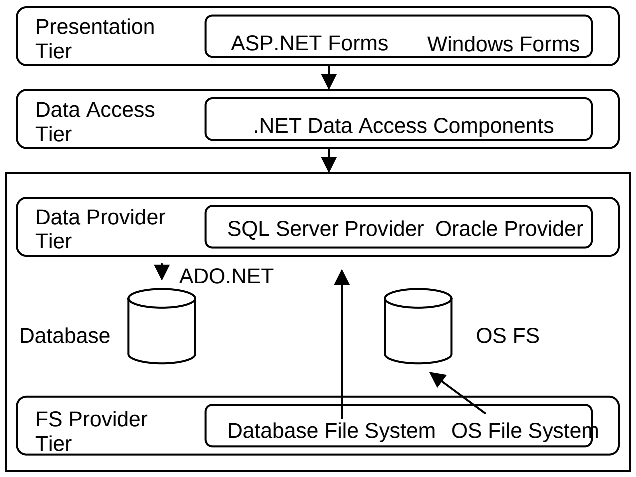


Figure 8. A pictorial high level topology of the FrontDesk architecture.

FrontDesk was designed with the realization that most courses and universities already have existing grade tracking systems that are hard to replace directly. To help alleviate the burdens of introducing FrontDesk into such a situation, we exposed much of FrontDesk's functionality through XML web services. The service oriented architecture allows for external grade management systems to easily connect to FrontDesk to retrieve grading information in a platform independent manner. Student submissions can be made in a non-Web interface by writing programs that invoke the submission system through the web service exposing the submission system. In short, FrontDesk exposes all major functionality through the use of web services allowing for flexible integration of the system with existing systems.

5. Experience and Future Work

FrontDesk has been deployed for the Spring 2004 semester offering of 15-211 Fundamental Data Structures and algorithms at Carnegie Mellon which has 300 students enrolled. It has successfully handled over 1000 submissions over the course of 3 assignments. Current course staff reports that students generally agree with course staff evaluation of submissions and find that students are better able to learn from mistakes their code suffers from. They also respond positively on the ability to manipulate files directly through the Web interface. Future work will involve the use of FrontDesk throughout the undergraduate curriculum at Carnegie Mellon. There is also an effort to collect more scientific data in order to determine the tangible effects FrontDesk has on a course.

6. Conclusion

FrontDesk provides functionality that directly addresses the problems associated with the administration of multiple large programming courses. Through its flexible objective correctness testing and rich subjective feedback model, the experiences of the course staff and students are made easier and, we believe, more enjoyable. We believe that the cost of a principled enterprise design has helped bridge the gap between course staff and students and additionally provides a flexible approach suitable for diverse programming intensive courses.

7. Acknowledgements

We would like to thank Professors Peter Lee and Bill Scherlis at Carnegie Mellon for their support and input on the project and permission to use the system in courses they taught.

References:

[1] Stephen Edwards, Teaching software testing, automatic grading meets test-first coding. *Addendum to the 2003 Proceedings of the Conference on Object-oriented Programming, Systems, Languages, and Applications*, San Francisco, CA, 2003, 500-506.

[2] J.B Hext, J.W. Winings, An automatic grading scheme for simple programming exercises, *Communications of the ACM, 12*(5), 1969, 272-275.

[3] Kent Beck, *Test Driven Development* (Addison-Wesley, 2002).