Department of Software Engineering
Building 70-1690
Phone: (585) 475-5461

Senior Project Proposal

UNDERGRADUATE

Project Title:	Software Submission and Assessment System
Organization:	RIT Department of Computer Science
Contact:	James Heliotis
Email:	jeh@cs.rit.edu
Phone:	585-475-6133

Background Information

The Computer Science (CS) department at RIT has used various automated systems that enable our students to submit their programming assignments electronically. The advantages to this approach is the ease of dealing with electronic originals that are well-organized, logged, and automatically backed up, and in some cases the value added by automated testing of the students' code.

Currently the most popular such tool is '**try**' developed by Professor Kenneth Reek with the last major release coming in the early 1990's. The students are required to run **try** from a computer in the department that is running Solaris™ and shares the same NFS file system with the computer system from which the grader will be working. Each submitter's work is archived and saved in a grader account. It is also usually tested with customized scripts; compiled code and the results of any tests are also archived for use by the grader.

Project Description

It is the desire of the department to develop a replacement for **try** that expands its applicability, removes some of its shortcomings, and enables its integratability with other front-end tools being developed. What follows is a strawman statement of needs for the project.

There are three user roles, or actors, in this system.

- An *administrator* owns a target computer system where submissions are saved. The administrator establishes parameters for a particular submission.
- A *student* submits files to be assessed.
- A *grader* looks at submitted materials to assess the student's performance. Graders have at least limited access to the administrator's system.

A software tool is needed that will:

- allow students to send electronic files of all varieties to a separate area on another computer system accessible to the administrator and graders, but not to students;
- process the submitted files in a manner determined by the administrator;
- store the results of the processing along with the original student submission.

Some individual requirements are listed below in no particular order of priority.

A multi-platform solution is desirable. Students should be able install a client module on whatever computer system they prefer. That client should then use the Internet to enable the transmission of the student's files to the administrator system. In the near term, the service software that runs on the administrator's system needs to run on Solaris, but again a multi-platform solution is preferable here as well.

Logs of submissions should be kept, along with time stamps, to allow resolution of disagreements between graders and students, and to help with restoring lost files from the appropriate backup volume.

Scripting and structural languages should be employed to allow processing ("testing") of students' files during or after submission. Post-submission testing may be preferable since it would allow changes by the administrator even after submissions have taken place.

There is no foreseeable need for test results to be in any form but in plain text. If structured data is required, XML could be employed. If graphical interface testing is required, it is currently assumed that this will be done manually by a grader afterwards, or automatically by an administrator-developed program acts as a surrogate graphical module that can run its own tests based on textual instructions and report its results in textual form as well.

A secure network communication system is required. It should include authentication and student file encryption.

All reasonable precautions should be taken in the design of the system to eliminate the possibility of a student using the system for reasons other than that for which it was intended. A prime example of this would be submission of a "Trojan horse" file that could (a) damage the administrator's area or (b) report information about the tests back to the student.

Currently course laboratory and project descriptions are stored in a single XML file. All needed documents are generated from this file. These documents currently include student instructions, grader instructions, and information about the items to be submitted and their point values in the eventual assignment grade. The system described in this document should be designed to integrate with this data. More information about the XML file and its derivatives are available. Adjustments are possible to accommodate this project's design.

The grader will require an application that extracts the submitted materials and reports the results of any processing done on them. We do not currently see a need to have the application automatically assign a grade. However, it should be aware of the maximum grade and store a grade of the grader's choosing in some form of grade book. The grade book should be easily translated into popular forms like HTML, Microsoft Excel™, and others.

"Wizard" tools should be available that facilitate the setting up a new assignment by an administrator. Historically, the difficulty of creating a new assignment is perhaps the most notorious of the shortcomings of the current version of **try**.

Technical Constraints & Assumptions

The system should run on the hardware and software currently available in the CS Department. New software acquisitions would be considered only if there is no cost involved and the effort of maintaining them is low.

Project Scope

The time periods stated in this section are not binding.

1-2 months: A team of 2-4 members of the CS faculty will regularly meet with the team to work on clarifying the requirements. The faculty members will provide user stories or use case scenarios as needed to facilitate this process.

1-2 months: The team shall produce a core architecture and a prioritized list of features for the faculty to approve.

2-3 weeks: A schedule of deliverables, subject to revision, shall be negotiated with the faculty. If any of the deliverables expect a corresponding deliverable from the front-end tools by a member of the CS department as well, this will have to taken into consideration.

After at most 5 months, a preliminary working system will be delivered for testing. Final versions of all the deliverables will, of course, be delivered towards the end of the spring quarter 20043.

We want to emphasize that we are not expecting that everything mention here will get done during the 20042 and 20043 quarters. What we are looking for is a solid architecture with reasonable suggestions on how missing features could later be added.

This project is by no means mission-critical. In the long run, we will have to replace our current system in order to adapt to changes in computing platforms and in order to address faculty productivity problems. For now, however, our current system is meeting our needs.

Expected Deliverables

A well-documented and diagrammed architecture for the project

UML-based descriptions of the salient parts of the design

Well-commented code that conforms to any locally accepted standards

Diagrams and descriptions of the file contents and directory organization required by the system

A user manual for each of the 3 user roles (Detail required depends on the level of the computer-human interfaces in the system.)

A presentation to the faculty team

A seminar open to all members of the CS community for the purpose of familiarizing them with the system.