On-line Assessment for Web-Based Programming Portfolios

John K. Estell
Bluffton College

Abstract

A methodology for Web-based programming portfolios that focuses on utilizing the interactive nature of the medium is presented. The concept of a portfolio and its value for assessment is reviewed, leading into a discussion on the benefits of electronic portfolios and rubrics for enhancing student learning outcomes. The development of the Interactive Programming Portfolio at our institution is used as a case study to examine how on-line assessment can be implemented.

1. The Portfolio

A portfolio consists of a collection of materials assembled over a period of time that is used to both demonstrate and document one's ability in a particular subject. Portfolios are commonly used in the artistic professions. For example, photographers who specialize in weddings will present to the inquiring engaged couple an assembled collection of their work. By constructing a portfolio photographers have the opportunity to reflect upon their work as they select the best results from their photographic sessions; similarly, the couple looking to hire someone for their wedding can use the portfolios to evaluate the ability of each photographer. So not only is the portfolio a means to demonstrate and document competence, it also allows for assessment by both the person assembling the portfolio and those who must pass judgement on that person's work. While not as widely used, portfolios can also be found in the computing sciences. The programming portfolio contains a selection of computer programs that a programmer has produced over a period of time.

The traditional format of the portfolio is a physical document, typically a ring binder that allows for the easy insertion or removal of items. For the programming portfolio, the document usually consists of a notebook containing pages of program source code listings, sometimes combined with text-based example runs or graphics-based snapshots showing particular moments of program execution. While useful, reviewing such material is about as exciting as watching paint dry as it fails to capture the essence of the programs in action. There is also the noticeable drawback that at most one person can use the portfolio at any given time as physical possession of the document is required. Additionally, the notebook format forces the organization of the portfolio to be sequential because of the nature of the print medium; however, the structure of ideas is not sequential. As early as 1945 Vannevar Bush noted that the human mind operated by the association of thoughts and proceeded to describe his "memex" system that is considerably similar to the Web environment of today[3]. The concept of hypertext, presented by Ted Nelson in 1965, brought forth the possibilities of using the computer to form documents of arbitrary structure[4]. With the advent of the personal computer and the development of hypertext systems in the mid-1980s, it was a simple jump for educators to promote the use of electronic portfolios. This eliminated the "at most one" problem as it is easy to provide multiple copies of an electronic document. Hypertext also allows a better structural representation of the thought process involved in the learning process, as it can better show how a portfolio entry was the product of the synthesis of a variety of ideas.

With the rise of the World Wide Web there has been increased interest in the development of electronic portfolios. The nature of the Web as an interactive multimedia facility that can provide information on demand opens up new possibilities for the use of portfolios in many disciplines. Added interest has been generated by the need to document and access student outcomes according to guidelines developed by accreditation agencies such as the Accreditation Board for Engineering and Technology. Probably the best known effort for the implementation and use of electronic portfolios is the RosE-Portfolio system developed at the Rose-Hulman Institute of Technology[6]. The electronic portfolio was adopted at their institution to reduce the accessibility problem encountered with the traditional portfolio. Deemed an efficient and cost-effective method of collecting and accessing student materials, the RosE-Portfolio has so far proven itself useful. Students have found it easy to enter and update documents using a familiar multimedia format. Faculty members that have used the system have found the system reliable and easy to use. There is optimism that the information provided overall by this assessment methodology will be useful in helping to analyze and improve the programs offered at their institution.

Applying the concepts of an electronic portfolio to the subject of computer programming is straightforward. It is simple to create a Web document containing listings and descriptions of one's computer programs, especially as the source code is already stored in an electronic format. The pages within this document may include graphical snapshots of programs "in action" and links to downloadable source code and executable files. The problem with this approach is the amount of effort required by the reviewer to evaluate the material. It is worthwhile to download the source code only if one possesses a suitable compiler or interpreter for the language in which the program is written. Furthermore, despite the claims of the portable nature of high-level programming languages, a sufficiently robust program can easily contain platform-dependent code that would prevent its compilation and execution on different systems without some degree of modification. The downloading of an executable file also suffers from platform dependencies; in addition, there is an element of risk associated with running this type of file on one's own computer as the code could maliciously contain a hidden virus.

Research using various search engines on the Web has failed to turn up many student programming portfolios at the present time. The majority of those found had the appearance of being student-initiated and not established as an instructional requirement. The lack of guidance is apparent at these sites, both in terms of portfolio design and of web site design. Almost every site examined simply presented a list of programs. There was neither elements of reflection upon the ideas and concepts found in each of the entries for a typical portfolio, considered vital for the learning experience[5], nor was there evidence of specific learning outcomes. Even more apparent was the near total lack of aesthetics in the design of the web sites. For some reason, many students seem enamored with the use of dark-colored text on a black background, the selection of which fails to provide sufficient contrast for proper readability. Other design flaws, too numerous to mention here, were also apparent on many of these web sites. If programming portfolios are to be effectively used in the computer science curriculum, then students must be provided with some form of structural guidance for the development of portfolio entries and for proper web page design. In addition, the accessibility issue concerning program entries needs to be addressed so that reviewers can efficiently examine executable code without having to go through various contortions for the downloading and execution of such code.

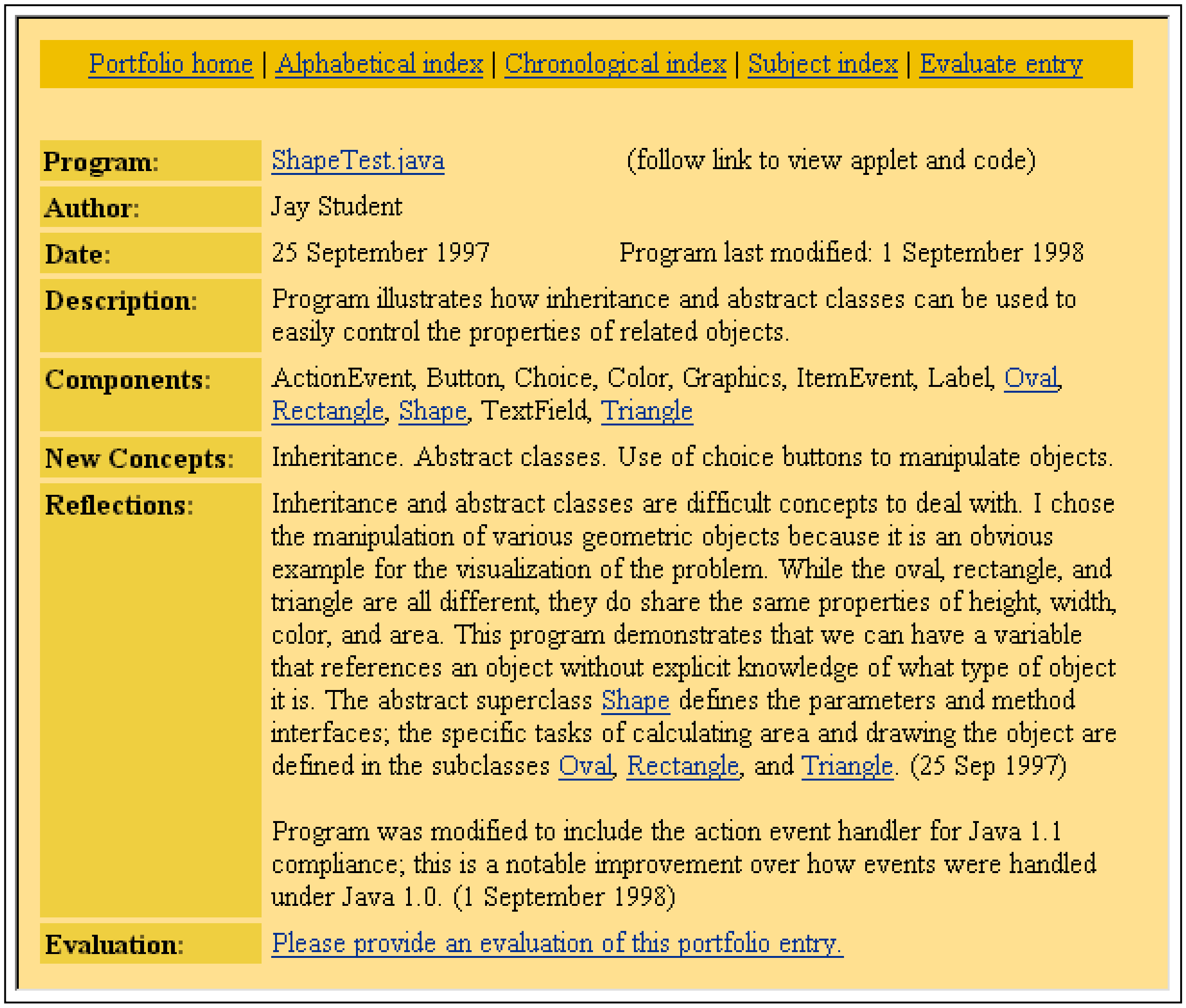
2. Java and Interactive Portfolios

The Java programming language provides a new design paradigm for programmers: write once, run anywhere. This object-oriented language was designed to maximize portability by specifically defining many of the details of the language for all implementations, from the source code all the way down to the byte code for the abstract machine language into which Java code is translated[2]. Java programs can be executed on any platform that has a Java Virtual Engine written for it; this engine is usually incorporated into all Web browsers. While Java programs can be written as stand-alone applications, they are best known in their applet format, which allow interactive programs to be incorporated into web page design.

At our institution Java is the language used in the object-oriented programming (OOP) courses that our students take in their second year of study. Many of the non-OOP constructs of the language are derived from C, which is learned by our students in their first year of study. Accordingly, the introductory OOP course focuses on the OOP aspects of Java. In order to explore these concepts through the writing of programs, applets are introduced early in the course curriculum. This requires that students obtain a fundamental understanding regarding event-driven graphical user interface (GUI) programs. A minimal, but sufficient, amount of coverage is presented at this time regarding the use of Symantec's Visual Café Java rapid application development tools, the button, label, and text field GUI components, a small number of methods associated with these components, and how one handles action events generated by components. As our students have previously been instructed as to web page development, once this material is covered they are able to write simple Java applets.

Now that the students have a basic understanding of event-driven GUI programming using applets, the interactive programming portfolio (IPP) is introduced. The IPP is presented through the examination of an example portfolio developed for this purpose. The home page for the IPP is very simple, containing text identifying that this is a student's programming portfolio and providing links to three different orderings of entry information: alphabetical, chronological, and subject. The pages for the alphabetical and chronological ordering of portfolio entries are straightforward, with each item consisting of a link to the portfolio entry followed by a brief description of what that particular entry is about. The alphabetical ordering allows for quick access to a known portfolio entry whereas the chronological ordering allows for easy analysis of the student's development over time. Subject ordering is used to focus the student's attention onto what key concepts are embodied by their programs. For this index the subjects are listed alphabetically and are coupled with one or more links to the relevant portfolio entries. One of the benefits of this ordering scheme is that a reviewer can easily determine whether a particular subject has been grasped by the student without having to go through all of the portfolio entries.

The actual portfolio entry can be in a variety of forms; however, it is in the student's best interest to adopt and stick with a particular design format for all entries in the portfolio. A sample portfolio entry from the example portfolio shown to our students is given in Figure 1. The navigation bar at the top of the page allows quick access to all indexes plus the home page. The column to the left shows the fields represented in this entry. Note that the applet and corresponding source code is not contained in the portfolio entry page. Instead, it is stored on a separate page and is accessed via a link in the "Program" field. This is done in order to allow the viewing of a portfolio entry without forcing the reviewer to also download the source and executable code. If the reviewer finds the program to be of interest, the executable applet is only one click away. The "Components" field is used to indicate what classes are included in the applet; if a user-defined class is present then a link is given to a page that, at a minimum, contains the source code for that class. The key fields, from a portfolio standpoint, are "New Concepts" and "Reflections." The "New Concepts" field requires the student to examine and identify what is being accomplished through the writing of the program. The "Reflections" field is used by the student to gain valuable insight into the learning process by incorporating such items as documentation of the design choices before writing the code and an analysis of the


Portfolio home | Alphabetical index | Chronological index | Subject index | Evaluate entry

Program:	ShapeTest.java (follow link to view applet and code)
Author:	Jay Student
Date:	25 September 1997 Program last modified: 1 September 1998
Description:	Program illustrates how inheritance and abstract classes can be used to easily control the properties of related objects.
Components:	ActionEvent, Button, Choice, Color, Graphics, ItemEvent, Label, Oval, Rectangle, Shape, TextField, Triangle
New Concepts:	Inheritance. Abstract classes. Use of choice buttons to manipulate objects.
Reflections:	Inheritance and abstract classes are difficult concepts to deal with. I chose the manipulation of various geometric objects because it is an obvious example for the visualization of the problem. While the oval, rectangle, and triangle are all different, they do share the same properties of height, width, color, and area. This program demonstrates that we can have a variable that references an object without explicit knowledge of what type of object it is. The abstract superclass Shape defines the parameters and method interfaces; the specific tasks of calculating area and drawing the object are defined in the subclasses Oval, Rectangle, and Triangle. (25 Sep 1997) Program was modified to include the action event handler for Java 1.1 compliance; this is a notable improvement over how events were handled under Java 1.0. (1 September 1998)
Evaluation:	Please provide an evaluation of this portfolio entry.

Figure 1. Example Interactive Programming Portfolio entry.

finished product. Reflection can be performed at any time; the entries in the example portfolio illustrate this by showing multiple entries in the "Reflections" field. In our courses students are asked near the end of the term to review and update their portfolio entries. This provides students the opportunity to perform an informal self-assessment of their abilities as well as to see how much their skills have developed over the past term.

Once placed onto the web site the portfolio entry is available for perusal by the reviewer, who has the luxury of sitting at the computer, surfing to the appropriate web page, and interacting with the executing applet. The reviewer can evaluate the program without having to deal with a bunch of floppy disks or with scripting programs designed for handling electronic submissions, and without needing to explicitly download and compile the source code.

The IPP has been used in our OOP courses for students to demonstrate competence in OOP through a combination of self-designed and instructor-assigned applets. Students best learn programming by writing programs; the IPP is used to encourage them to explore aspects of the Java language on their own by providing a vehicle to showcase their ability. Due to the multimedia features incorporated into the language, many students gravitate towards exploring applications involving graphics and sound. Some of the student-designed applets that have been written include a graphing window, a calculator, and a dancing John Travolta with "disco fever." Games are also popular; some that have been written include skeet shooting, Mad Libs, and poker. It is vital that students are provided with choice in the process as to what is included in the portfolio; otherwise, it degrades into an accountability folder for which the student feels no ownership. While the number of applets submitted by our students averages twelve per semester, it should be noted that students are not, and should not be, judged based on the quantity of applets generated. Students are periodically provided with a list of concepts covered in the course; they are to incorporate and exercise these concepts in their programs. As long as the student can demonstrate an understanding of the set of concepts provided, it matters not whether it took three or six or whatever number of applets to illustrate this understanding. Instructor-assigned applets are used, albeit sparingly at an average rate of three per semester, as a comparative assessment tool and as a way to ensure that students write programs incorporating certain language constructs at particular points in the course. Deadlines are strategically set for these instructor-assigned applets to ensure that students stay on pace with the current set of topics being covered in the course. Some of the assignments that have been given include implementing Yahtzee-like dice games complete with graphically-displayed dice, creating classes derived from an abstract class for a shape manipulation program, and a drag racing simulation. To further keep students on track with their studies, the traditional assessment methods of quizzes and exams are also used in conjunction with this portfolio method.

This assignment paradigm has shown itself to be an effective motivational tool for most students. Students work on programming topics that are of interest to them. Because of this interest the average student writes more code and places greater effort into ensuring its correctness than is normally encountered in the typical programming assignment. The IPP format has also been used by the instructor as a means to make example programs available to the students in the class. This provides students with both the source code, which is also distributed in class as a handout when the program is discussed, and the working applet so that they can interact with the program at their convenience.

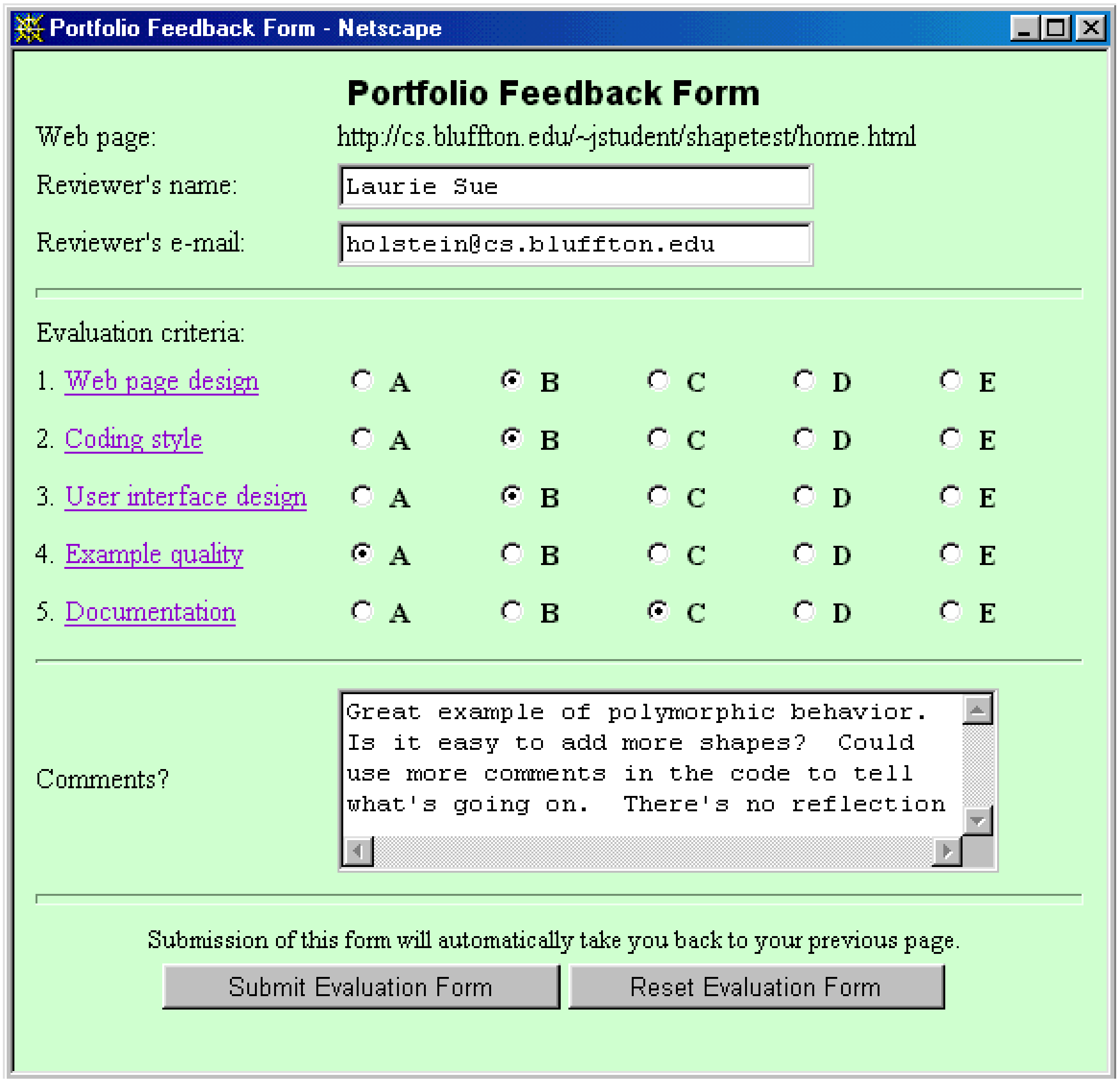
3. Rubrics and On-line Assessment

One of the difficulties with the use of portfolios is that of assessment. In the traditional model, a student provides a body of work to be graded by the instructor. The student in this model is motivated to tailor the presentation of material in the portfolio toward the one person who will assign a grade to the body of work. Unfortunately, the student is often unaware of the criteria that will be employed by the instructor. This failure to properly communicate goals and expectations takes away from the long-term learning experience desired from the use of a portfolio. Students need to know what is expected of them; instructors need to communicate this information to their students.

The development of the rubric as an assessment tool has helped to solve this problem. Rubrics are scoring guides that provide sets of standard descriptions for a group of outcome parameters that form the basis of the evaluation. For each outcome there are descriptions relating to the full range of quality levels of student work. Rubrics are commonly found in tabular form, with rows representing the outcomes being measured and the columns representing levels of achievement. This information can be made available to students, thereby allowing them to understand what is expected of them. The use of the rubric promotes objectivity in the assessment process; accordingly, the construction of a rubric must be developed through a carefully thought-out process.

The design of the IPP rubric evaluation procedure is motivated by the concept that successful portfolios require authentic audiences[5]. Accordingly, the evaluation forms for the IPP are made available on-line to anyone who wishes to provide feedback to the student. Simply clicking on a link that is provided with each portfolio entry allows access to the primary evaluation form, an example of which is shown in Figure 2. This form indicates the particular web page under review, presents the evaluation criteria categories and associated radio buttons for entering scores, and provides text boxes to allow reviewers to enter their name, return e-mail address, and comments. A reviewer needs only to fill in the required information then click on the submit button. A CGI script written in Perl processes the submitted form information and automatically mails it to the student's electronic mailbox. An example of the mail received by a student that was generated by this evaluation form is shown in Figure 3. In order to provide an uncluttered evaluation form, links are provided for each of the criteria categories that lead to detailed descriptions on their own web pages. An example rubric for example quality guidelines is presented in Figure 4. The value of this design is that anyone, be it a fellow classmate, student in the major, the instructor, or someone surfing the web from outside can provide the student with feedback. This openness requires the student to take greater ownership of the portfolio since it is freely available to all who have access to the Internet, thereby allowing for an increase in the number of learning experiences from both the added responsibility and hopefully greater amount of feedback.

Rubrics were developed for the IPP in five primary areas of evaluation: web page design, coding style, user interface design, example quality, and documentation. For producing a quality portfolio on the Internet, the design of the web pages is just as important as the information contained on the pages. The pages need to be created in such a way as to invite the visitor to examine its contents; poor design can easily drive people away from a site with quality content.



Portfolio Feedback Form

Web page:	http://cs.bluffton.edu/~jstudent/shapetest/home.html
Reviewer's name:	Laurie Sue
Reviewer's e-mail:	holstein@cs.bluffton.edu

Evaluation criteria:

1. Web page design	A	[✓ B]	C	D	E
2. Coding style	A	[✓ B]	C	D	E
3. User interface design	A	[✓ B]	C	D	E
4. Example quality	[✓ A]	B	C	D	E
5. Documentation	A	B	[✓ C]	D	E

Comments?

```
Great example of polymorphic behavior.
Is it easy to add more shapes?  Could
use more comments in the code to tell
what's going on.  There's no reflection
```

Submission of this form will automatically take you back to your previous page.

Figure 2. Portfolio evaluation form.

Portfolios are judged on four aspects of web page design: text appearance, graphics and layout, proper navigation, and reference links. The coding style employed in the writing of a computer program is very important as it provides valuable documentation within the source code. Simple things such as proper indentation, structure, use of comments, and meaningful variable names can be valuable later on when a program is modified, upgraded, or studied. Conversely, the use of poor coding style can render a great executing program into a worthless array of bytes when maintenance needs to be performed. For coding style, a portfolio entry is evaluated on the use of meaningful variable names, comments, coding structure, and implementation clarity. The design of the user interface for a program is very important; the program is of little value if one is unable to use it effectively. This category is affected by the underlying philosophy for the language and/or operating system being used for development. For example, Java programs usually deal with graphical user interfaces for handling events whereas filter programs for UNIX systems rely on using command-line entries for program behavior modification. The current rubric focuses on the proper use of the Java graphical user interface with regards to layout,

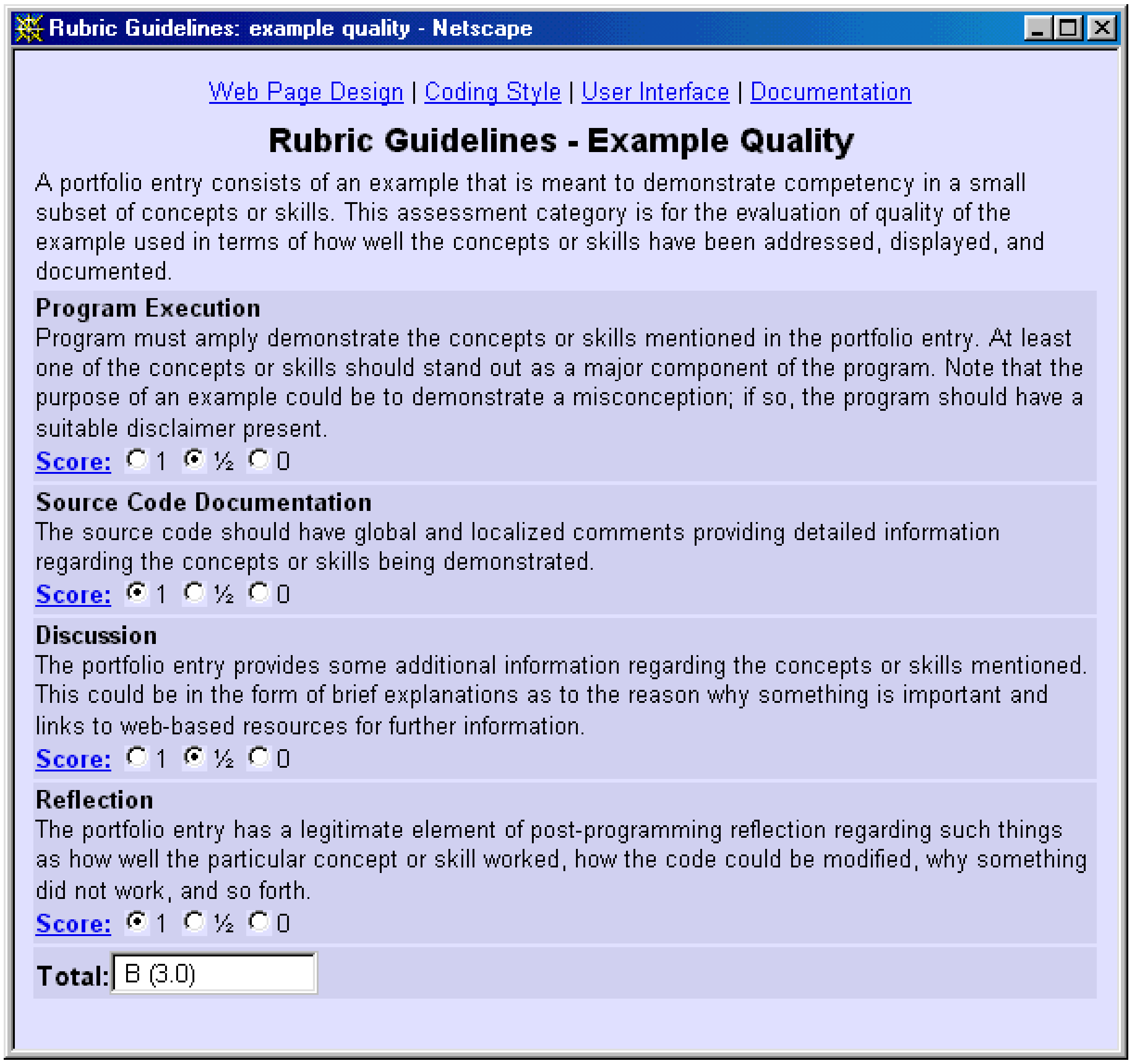
```
From: jstudent@cs.bluffton.edu (WebMonitor mail)
To: jstudent@cs.bluffton.edu
Subject: Portfolio Feedback
X-Comments: =============================================================
X-Comments: NOTE:  This message was sent through the WebMonitor mail form
X-Comments: =============================================================
X-Comments: HOST:      10.1.5.213 (10.1.5.213)
X-Comments: BROWSER:   Mozilla/4.7 [en] (Win98; U)
X-Comments: =============================================================
Content-Length: 653
Status: R

(webpage)  http://cs.bluffton.edu/~jstudent/shapetest/home.html
(next-url)  http://cs.bluffton.edu/~jstudent/shapetest/home.html
(subject)  Portfolio Feedback
(name)  Laurie Sue
(email)  holstein@cs.bluffton.edu
(webPageDesign)  B
(codingStyle)  B
(interfaceDesign)  B
(quality)  A
(documentation)  C
(comments)
----------------------------------------------------------------------------
Great example of polymorphic behavior.
Is it easy to add more shapes?  Could
use more comments in the code to tell
what’s going on.  There’s no reflection
given in your portfolio entry; however,
you might not yet have had time to look
back on what you did.
----------------------------------------------------------------------------
(Submit)  Submit Evaluation Form
```

Figure 3. Sample mail generated from posting of evaluation form.

responsiveness of event feedback, and intuitive use. The example quality category is for evaluating the quality of the example used in terms of how well the concepts or skills being presented have been addressed, displayed, and documented. The rubric guidelines for this category include evaluation of the student’s discussion of the examined concepts and post-coding reflections of what this particular entry has accomplished. Finally, documentation is presented as a separate category here, although other rubric areas contain some of these evaluation elements, to provide the student with a clear evaluation on this important issue. The lack of meaningful documentation within a program can prove catastrophic for a project at some point in time; throwing in some token comments after the code has already been written does not constitute proper code documentation skills. The rubric also emphasizes the evaluation of how well the portfolio entries document the concepts and skills being examined, and how well the portfolio documents the student’s learning experience.

The scoring guidelines are consistent for each question asked in the evaluation rubrics. There are three possible outcomes. A score of 1 indicates that the student has demonstrated mastery of the concept or skill being evaluated and that there is little to no doubt that the student understands


Rubric Guidelines: example quality - Netscape
Web Page Design | Coding Style | User Interface | Documentation

Rubric Guidelines - Example Quality

A portfolio entry consists of an example that is meant to demonstrate competency in a small subset of concepts or skills. This assessment category is for the evaluation of quality of the example used in terms of how well the concepts or skills have been addressed, displayed, and documented.

Program Execution
Program must amply demonstrate the concepts or skills mentioned in the portfolio entry. At least one of the concepts or skills should stand out as a major component of the program. Note that the purpose of an example could be to demonstrate a misconception; if so, the program should have a suitable disclaimer present.
Score: 1 [✓ ½] 0

Source Code Documentation
The source code should have global and localized comments providing detailed information regarding the concepts or skills being demonstrated.
Score: [✓ 1] ½ 0

Discussion
The portfolio entry provides some additional information regarding the concepts or skills mentioned. This could be in the form of brief explanations as to the reason why something is important and links to web-based resources for further information.
Score: 1 [✓ ½] 0

Reflection
The portfolio entry has a legitimate element of post-programming reflection regarding such things as how well the particular concept or skill worked, how the code could be modified, why something did not work, and so forth.
Score: [✓ 1] ½ 0

Total: B (3.0)

Figure 4. On-line rubric for portfolio entry example quality.

what is going on. The treatment does not have to be exhaustive, but should be appropriate to the task at hand. A score of ½ is used to indicate an incomplete understanding of the concept or skill being evaluated. Things may be performed inconsistently, not used properly, overcomplicated, stated vaguely, and so on. A score of 0 is used when the student has made little to no attempt to address the concept or skill being evaluated or when the work is of poor quality. Each rubric category contains four scored questions; these scores are added together and a letter "grade" is assigned using the traditional four-point scale. It is this letter grade that is mailed back to the student for the result of that particular rubric. Note that this is not a grade that will be marked into a grade book by the instructor; it is used because being evaluated on a letter grade scale is something to which a student can easily relate.

The evaluation by the instructor of the IPP needs to be performed carefully. One does not want to counteract the learning experiences obtained via the portfolio method by a display of authoritarianism. A better approach is to use instructor-student conferences as a means to demonstrate an interactive, collaborative process between student and instructor. The focus of the instructor should be on the big picture: Does the portfolio reveal thoughtful reflections? Is there insight regarding areas of strengths and weaknesses? Has the student demonstrated evidence of learning? Asking questions such as these as the portfolio is examined with the student can help point the way for the student's future goals and demonstrates the commitment that the instructor has for the student's educational fulfillment.

For the actual scoring of the IPP as part of the student's overall grade, another evaluation rubric is used. The rubric used for the IPP is based on a rubric developed in 1997 for traditional portfolio assessment[8]. This rubric, shown in Figure 5, is distributed to the students on the first day of class so that they are aware as to how their efforts will be judged. An important thing to note is that students are not scored on each individual entry; the grade assigned is for the composite body of work. The result of this evaluation is a value on the five-point scale; this value is converted as necessary and incorporated into the overall grading scheme. The criteria areas reflect the basic tasks involved with the construction of a portfolio. The portfolio should demonstrate a wide range of ability on the part of the student. The self-reflections contained in the entries should contain insight into what the student has learned and accomplished with writing the program associated with this entry. Thoughtful attention to the development and documentation process should be evident. Students should talk about the problems encountered while writing a program and what was done in order to solve those problems. The overall content, form, and mechanics of the portfolio should clearly indicate that the student has ownership of the portfolio, and treats it accordingly.

Criteria	Strong – 1 point	Average – ½ point	Weak – 0 points
Versatility	Portfolio shows student's wide range of interests and abilities.	Portfolio shows an adequate range of student's interests and abilities.	Portfolio shows little range of interests and abilities.
Reflections	Reflections are thoughtful. Student reveals strong insights about areas of strengths and improvement; has future goals.	Reflections are reasonable. Student shows some insights about areas of strength and improvement; indicates reasonable goals for future.	Reflections show little attention. Insights are lacking about areas of strength and improvement; lacking goals for the future.
Process	Entries show thoughtful attention to process. Indicates that student has gained some from the experience.	Entries show some attention to process. Indicates that student has gained some from the experience.	Entries lack examples of process. Indicates that student has learned little or nothing from the experience.
Problem Solving	Entries indicate that student recognizes most problems or responds to those pointed out through the review process. Shows resourcefulness in solving problems.	Entries indicate that student recognizes some problems or responds to some of those problems pointed out through the review process. Shows some resourcefulness in solving problems.	Entries indicate that student is unwilling or unable to deal with problems. Student does not identify own problems and ignores those pointed out during the review process.
Content, Form, and Mechanics	Student shows thoughtful attention to final product. Has strong command of content, form, and mechanics.	Student shows adequate attention to final product. Has growing command of contents, form, and mechanics.	Student shows little or no attention to final product. Significant improvement needed for content, form, and mechanics.

Figure 5. Instructor's rubric for portfolio evaluation.

4. Drawbacks and Potential Pitfalls

Using a Web-based programming portfolio does have its drawbacks. Currently, the greatest drawback for the IPP is that it can interactively showcase only those programs written in Java. If a student wants to include examples from other languages into the portfolio, the old static format of program listings and downloadable code must be used. Preliminary research is being conducted into the writing of compilers for languages other than Java that will produce bytecode for use with the Java Virtual Machine. These compilers, along with supplemental modules to provide appropriate interface support, will allow programs written in other languages to be displayed interactively.

Another potential problem is plagiarism. As noted in a recent article, there is an increasing number of "cybercheaters" who abuse electronic media by first using a Web search engine to easily sift through millions of on-line documents, then perform cut and paste operations from relevant sources to produce their "original" work[7]. There are many Java applet repositories already available on the Web that can be easily found using any Web search engine. Several of these sites provide the source code for the displayed applets. Additionally, there is the potential for visiting other student IPPs and copying the code presented there. Fortunately, the same tools that allow cybercheaters to operate can be used to verify suspicions of plagiarism. Fragments of suspicious code can be entered into a search engine, either a typical one such as www.hotbot.com or one specialized toward detecting fraudulent submissions such as www.plagiarism.org, in order to determine if it has been lifted from a listing available on the Web. For those foolish enough to copy and modify a classmate's submission, detection is no different than that practiced for the submissions of traditional programming assignments. Any competent, experienced instructor is capable of detecting plagiarized programs within the same submission set, as long as the set is not too large or the borrowed code is a small part of a large program. For those cases where there are a lot of submissions, there is a large amount of code to process, or comparisons need to be made with previous submissions, automated software plagiarism detection systems such as Moss[1] are very capable of finding and documenting examples of cheating. It is preferable, however, to curb plagiarism via preventative measures. The key method employed to prevent plagiarism is the simple fact that individual portfolio entries are not scored by the instructor. It is the entire body of work that is judged, and part of the evaluation process is performed with both student and instructor looking together at the portfolio. Another effective method for combating plagiarism is the use of the closed laboratory session, where the instructor is present in the lab and interacts with the students. It is rather difficult to perform an act of plagiarism while under the instructor's direct supervision. It is not necessary for all programming sessions to be conducted in this manner; an occasional closed laboratory session or two is sufficient to discourage this type of plagiarism. Having short conversations with one's students regarding their programs is also sufficient to expose a case of plagiarism. Regardless of how adept one is at modifying code to hide its true source, the student who plagiarizes code is unable to explain the workings of the program and the logic behind the code. This methodology is highly effective in a small class setting as the instructor quickly learns of student's abilities and so is better able to assist students in their learning efforts. Being able to actively engage and interact with one's students in the learning process and displaying a willingness to assist the student in their efforts renders moot the need for plagiarism on the part of the student.

One of the problems always encountered in the classroom is the lack of motivation among students. The IPP actually improves the performance of some of these students. There is usually a group of students whose lack of motivation is due to boredom; they find the traditional programming assignment approach to be trivial and uninspiring. The IPP allows these students to explore the areas that they have an interest in, in part because of the many bells and whistles that the Java programming language features. Through their explorations, they learn the concepts that are being covered in class, but they do this in a context that is acceptable to them. This behavior has been observed in some of our students. For weaker students, it is a different story. If left to their own ways, these students will write few, if any, programs on their own. It is difficult for them to initially buy into the concept of an open-ended portfolio; they will frequently try to coerce the instructor into specifying the quantity of programs that must be written to ensure their passing of the course. If one succumbs to this pressure the inevitable result is an IPP consisting of the minimum number of programs, most of which are of inferior quality. In an idealistic setting it would be nice to have students write their own programs to learn and explore the various aspects of programming as they progress through the course material. Unfortunately, the reality is that these students are often unsure as to what to do or are in a procrastination mode since there is usually no ominously approaching deadline staring them in the face. There are a couple of solutions to this problem. First is to incorporate periodic evaluations of student portfolios into the assessment scheme; a firm deadline is a good motivational tool and allows the instructor to systematically monitor student progress. This has been implemented in our OOP classes and has worked well. To assist students in determining what programs to write, a list of objective goals can be presented at the beginning of the course. A second list containing assignments that illustrate how some of the goals can be met can now be given. A simple example assignment would be to ask students to write a Java applet that will dynamically modify the text of a displayed component. This could involve changing the text, the font type, style, or size that the text appears in, or the color of the text. This would address such subjects as using text fields, font objects and color objects. By providing this information the struggling student now has some general guidance that is usually sufficient for getting started on a program.

One perceived difficulty with the use of the IPP is with the potential of a student receiving detrimentally negative feedback in an evaluation. This could be the result of a deliberate attempt to belittle one’s efforts or unconsciously done by a student who naively believes that he or she knows everything there is to know about Java programming. As part of this research the CGI script used to mail the results of a submitted evaluation to a student will also mail a copy of the results to a central repository for later analysis. A careful review of all submitted evaluations has shown that this concern has not been a problem. While there has been critical remarks made, they have always been of a constructive nature, encouraging the recipient to improve upon the current work. Part of this is due to our computer science program having small class sizes where the students all know and work with each other. Additionally, we have spent a couple of closed laboratory sessions where the students evaluated each other’s portfolio under the instructor’s supervision and guidance. Most students are sophisticated enough to know that there will be cases where they will receive a poor evaluation from a clueless reviewer. However, this negative review does not affect their score and will be more than offset by the several favorable reviews that they receive for that same portfolio entry.

5. Conclusion

The use of the interactive programming portfolio at our institution has shown itself to be an effective assessment and professional development tool for both the student and the instructor. However, it must be noted that the development of this methodology is still in its infancy; the full potential of this approach has yet to be realized. While at the present time only programs written in Java are truly interactive when featured on a web page, the production of Java Virtual Machine compliant compilers for other languages along with appropriate support modules will open up greater possibilities. Even in its current limited state the IPP offers benefits that traditional portfolio methods lack as its use of the Web embodies the visions of documents formed with structures of association proposed by Bush and Nelson decades ago.

Student use of the IPP has been favorable. The key difficulty for student acceptance of this new paradigm is getting them to understand and appreciate the long-term benefits. It has helped that the IPP has shown itself to be useful as part of a resume. One of our former students received a job partially due to his employer being impressed with his Java programming skills. On his resume the student included the URL of his IPP, which was visited by his soon-to-be employer during the hiring process and its contents commented favorably upon during his interview.

As the need for assessment tools greaten, methodologies must be developed to successfully address the concerns of the various interested communities. With the continuing rise in stature of Web-based documents, the use of the electronic portfolio, with its benefits of accessibility and association, is becoming increasingly more common. As it grows and matures, the interactive programming portfolio will play a greater role in documenting and demonstrating the competency and ability of computer programmers.

Bibliography

1. Aiken, A. URL: http://www.cs.berkeley.edu/~aiken/moss.html; “Moss: A System for Detecting Software Plagiarism.”
2. Arnold, K. and Gosling, J. *The Java Programming Language*. Addison-Wesley, Reading, Mass., 1996.
3. Bush, V. As We May Think. *The Atlantic Monthly 176, 1* (July 1945), 101-108.
4. Nelson, T. *Dream Machines*, Microsoft Press, Redmond, Wash., 1987.
5. Porter, C. and Cleland, J. *The Portfolio as a Learning Strategy*. Boynton/Cook Publishers, Portsmouth, N.H., 1995.
6. Rogers, G. M. and Williams, J. “Building a Better Portfolio,” *ASEE Prism 8, 5* (January 1999), 30-32.
7. Ryan, J. J. C. H. “Student Plagiarism in an Online World,” *ASEE Prism 8, 4* (December 1998), 20-24.
8. Supernault, P., Project Leader, *Portfolios for Teaching and Learning*, URL: http://www.monroe2boces.org/shared/instruct/portfolios/porteval.htm; “Rubric: Portfolio Evaluation.”

JOHN K. ESTELL
John K. Estell joined Bluffton College as an associate professor of computer science in 1996. He was previously an associate professor at The University of Toledo. He received a B.S. (1984) degree in computer science and engineering from Toledo and received both his M.S. (1987) and Ph.D. (1991) degrees in computer science from the University of Illinois at Urbana-Champaign. His areas of interest include computer science education, the social impact of computers, programming development tools, and interface design. Dr. Estell is a member of ACM, ASEE, IEEE, Tau Beta Pi, and Eta Kappa Nu.