SEEWeb: Making Experimental Artifacts Available

Jeff Offutt
ISE Department, MS 4A4
George Mason University
Fairfax, VA 22030-4444 USA
ofut@ise.gmu.edu

Yuan Yuan
Computer Science
George Mason University
Fairfax, VA 22030-4444 USA
yyang1@gmu.edu

Jane Huffman Hayes
Computer Science
University of Kentucky
Lexington, KY 40506-0495
USA
hayes@cs.uky.edu

ABSTRACT

This position paper suggests that some of the technical and methodological challenges facing software testing researchers can be addressed by establishing a repository of experimental software artifacts, in particular, artifacts that are related to software testing empirical research. We introduce the Software Engineering Experiments on the Web (SEEWeb) project, a Web site that is created to be a convenient and usable infrastructure for gathering, organizing, and distributing experimental software artifacts. A common problem in designing software engineering and software testing experiments is finding experimental artifacts that are appropriate for the experiment, convenient to gather and use, and will fit with other experimental artifacts. SEEWeb was initially funded by the NSF and is offered as a service to the community and provides access to experimental artifacts through an interface that allows browsing, searching and downloading. SEEWeb loosely follows the open-source philosophy; experimental artifacts are provided by researchers on an as-is basis with the only payment being citations and acknowledgments to the contributing researchers. SEEWeb can be accessed online through the URL `http://www.ise.gmu.edu/seeweb/`.

1. INTRODUCTION

A continual theme of the software engineering research field over the past two decades has been an increasing sophistication of experimentation, both in terms of the design of the experiments and analysis of data. Software engineering experimentation requires access to artifacts that are appropriate for the experiment, that are convenient to gather and use, and that will match other artifacts used in the same experiment. Many kinds of software engineering experimental artifacts exist, including program source, program executables, design documents, requirements, specifications, documentation, and of course, tests.

The reuse of software engineering artifacts contributes to replication of experimental results as well as consistency of reported results in the literature. But reuse is not easy. While it may be easy to obtain select artifacts for reuse, it is difficult to obtain entire projects in a form that is convenient for use. A researcher attempting to obtain project information for reuse encounters many obstacles:

- They are difficult to find
- Once found:
  - they are difficult to obtain
  - other researchers do not respond to requests
  - other researchers respond negatively to requests
  - researchers provide partial data that is not useful
  - companies may put restrictions on publications
  - agencies or companies will not allow use of data by non-citizens

Creating new artifacts is time consuming and difficult and runs the risk of containing unexpected errors (until used a few times).

Some further, unique challenges are faced by testing researchers. The WERST 2004 call for papers lists a number of relevant questions. For example, seeding of faults is frequently required in order to measure effectiveness. How do we identify representative faults and fault distributions? What represents good, reusable testing research hypotheses? What are representative subject programs? How can we access subject programs, faults, and tests used by other researchers?

Open-source software can certainly help. A recent editorial in Empirical Software Engineering stated [1]: "As empirical software engineers, we should embrace this development [open-source software]. Suddenly one of the greatest obstacles in the way of empirical software engineering has been cleared! Not only is source code available, but also defect reports, update logs, etc. For a change, we can now focus on the analysis rather than the data collection." However, this is only a partial solution. Open-source software represents part of the industry, but not all. Also, Chen et al. found problems with open-source projects, particularly with ChangeLog files [3].

These problems suggest a novel solution: The software engineering and software testing community needs access to a collection of experimental software artifacts that is **convenient** and **usable**. Our position is that this need can be satisfied by a Web-based method to share experimental artifacts that support experimental reuse and collaboration. Thus, in the summer of 2004, with support from a

grant from the NSF's Research Experience for Undergraduates program, we established the goal of making non-Open Source software artifacts available on the Web in a format that is as convenient to access as open-source software artifacts. This paper describes a Web application that is intended to be a convenient and usable infrastructure for gathering, organizing, and distributing experimental software artifacts.

The artifacts include software implementations, multiple versions of the software implementations, requirements, trace answer sets, formal specifications, design representations, faulty versions of implementations, change logs, test case sets created to satisfy various criteria, experimental data, and shell scripts that serve as experimental infrastructure. Our first step was to establish strong requirements for the application.

2. PROJECT REQUIREMENTS

One of our initial decisions for this project was that the User Interface (UI) to SEEWeb is crucial to its success. The whole point of creating the service is to make it more convenient to distribute experimental artifacts. A Web site that is difficult to use will not be used, and therefore will be a failure. Specifically, a Web site to distribute experimental software artifacts must provide very **flexible** access to artifacts, and also be very **simple** to use. The most obvious way to make a UI flexible is to add more and more features to access and search for artifacts. Unfortunately, adding features tends to make UIs more complicated to learn and use. Conversely, the easiest way to make a UI simple is to reduce the number of features. That is, flexibility and simplicity are conflicting requirements.

To resolve this conflict, our solution relies on two methods that take advantage of technologies for Web applications. First, users are shown the **least** amount of data necessary to browse the contents, organized in a flexible way. Second, a flexible and quick method for searching the Web site is available as the primary mechanism for interfacing with the system. To be precise, the searching technique is not exactly searching per se, but actually uses *filtering*[1]. Thus, the user is initially shown all projects in the SEEWeb system, then uses the filtering mechanism to reduce the number of projects of interest. The value of this filter will increase as the number of projects grows.

3. SEEWEB CONCEPTUAL DESCRIPTION

SEEWeb is still a work in progress and is not yet stabilized. Its database is also not yet well populated with projects. On the other hand, it already shows promise as a useful way to share experimental artifacts.

Conceptually, SEEWeb contains a collection of *projects*. Each project has a unique *name* and is associated with *information* about artifacts that are located in files. Information about projects are stored on the SEEWeb server, which is currently located on a server maintained by the Information and Software Engineering Department at George Mason University. This information is stored in a sequential database. The actual artifact information is contained in files. These fies may be stored on the SEEWeb server, or contributors may keep the files on their own servers and let SEEWeb manage links to those files.

Each project includes the name of an author and a list of publications that used or referenced the project. This list, of course, will need to be regularly updated.

The artifacts include, but are not limited, to the following:

- Implementation
- Test cases
- Faulty versions
- Specifications
- Design
- Documentation
- Requirements
- Change log information
- Experimental data
- Shell scripts
- General comments about the project

A key to the usefulness of SEEWeb is the ability to search for artifacts that match the needs of a particular experiment. Our approach is to define *categories of information* that experimenters will care about. The system provides the ability to filter on each category. For example, if a program must be implemented in a specific language, an experimenter can filter for projects in that language. Categories that are currently available include:

- Language: The implementation language of the program.
- Size: Approximate size of an implementation in terms of number of modules (classes) and lines of code.
- Date Added: The date the project was added to the system.
- Last Updated: The date the project was last modified.

SEEWeb also defines three different types of users, each with a different level of access to the projects.

- **Public User**: Public users shall be able to search, read and download projects. No login is necessary.
- **Trusted User**: Trusted users can contribute projects. They can add new projects into the system, and add new artifacts to existing projects, as well as search, read and download projects. Trusted users need to apply for a login to the administrator through the UI.
- **Administrator**: The administrator can delete, add and edit projects, and types of artifact and criteria. The administrator can also delete, add and edit new users, give permissions to users to add new projects, list the users who have downloaded certain projects and is able to search, read and download projects.

3.1 Software Design and Implementation

SEEWeb is designed as a straightforward Web application on the J2EE platform. Most user screens, including formatting, data entry and browsing capabilities, are implemented with Java Server Pages. The JSPs interact with Java beans, which manage the data. The Java beans, in turn, use a database to store the data (currently MySql). A significant advantage of the database is that the browsing and filtering requests from users are translated into SQL queries instead of being implemented as program methods.

[1]This filtering is modeled loosely on the filtering in the mail tool of the Mozilla Web browser.

One ramification of using MySql is that it only supports a subset of SQL. In particular, MySql does not support UNION and EXISTS clauses, nor does it support subqueries. This complicated searching and sorting and meant that some user requests have to be handled by several queries at the programming level, and then code had to be written to combine the results. We also found that OR statements could not be used because the execution time is to great.

3.2 User Services

SEEWeb currently contains 36 JSPs and 20 Java classes, most of which are simple beans to interface with the database. A list of the components currently implemented follows:

- **Registration**: Users can register for an account in SEEWeb. SEEWeb will automatically create public user accounts.
- **Logging in**: Any user can log on to SEEWeb and browse projects.
- **User accounts**: Public users can browse, view and download projects. Trusted users can add and edit projects and perform any of the functions of the public users. Administrators can delete, add and edit projects, and can delete, edit, and add new users. They can also change the status of users and perform all of the functions of the trusted users.
- **Viewing/Browsing projects**: Public, trusted and administrator users can view all projects.
- **Viewing/Browsing users**: Administrators can view all users. They can view users names, first name, last name, status, email address and the names of the projects they have added to SEEWeb.
- **Displaying projects**: Public, trusted and administrator users can view all projects in more detail.
- **Adding/Editing/Deleting projects**: Trusted and administrator users can add and edit projects. Administrators can delete projects.
- **Adding/Editing/Deleting users**: Administrators can add, edit and delete users.
- **Sorting projects**: Users can sort projects during browsing.
- **Searching projects**: Users can search for a project using the searching component. They can search by artifact, project name, requirements, specifications, implementation, faults, tests, design, documents, logs, data and scripts.

  Users can also use a filter during their searches. The search filters include *contains*, *does not contain*, *begins with*, *ends with*, *is*, and *is not*.
- **Sorting and searching projects**: Users can further customize their browsing through searching and sorting.
- **Sorting users**: Administrators can sort on the types of accounts to browse.
- **Searching users**: Administrators can search for users under user names, first name, last name, status, email address and project name.
- **Sorting and searching users**: Administrators can further customize their browsing through searching and sorting.

4. CONCLUSIONS

The SEEWeb project is in its infancy, but is ready to be populated and used. Recently, the NASA Independent Verification and Validation Facility placed several artifacts on SEEWeb. These include a high level requirement document, a design specification, and the verified trace between the two (trace answer set) [2]. As a Trusted User, they provided valuable feedback on the UI of SEEWeb. Use by other researchers will serve to further improve it. Preliminary feedback on SEEWeb lends support to our position: some of the technical and methodological challenges facing software testing researchers can be addressed by establishing a repository of experimental software artifacts, particularly those relating to software testing empirical research. SEEWeb can be accessed online through the URL `http://www.ise.gmu.edu/seeweb/`, with login and password (demo, demo).

5. ACKNOWLEDGMENTS

This work is supported in part by the U.S. National Science Foundation under grant CCR-00-97056 and CCR-00-97056 supplemental (NSF Research Experience for Undergraduates). This work was also partially supported by NASA under grant NAG5-11732.

6. REFERENCES

[1] R. Harrison, S. Counsell, and R. Nithi. Editoral: Open source and empirical software engineering. *Experimental Software Engineering*, 6:193–194, 2001.

[2] J. H. Hayes, A. Dekhtyar, S. Sundaram, and S. Howard. Helping analysts trace requirements: An objective look. In *International Conference on Requirements Engineering (RE'2004)*, 2004.

[3] L. Y. Kai Chen, Stephen R. Schach and J. Offutt. Open-source change logs. *Empirical Software Engineering Journal*, To appear, 2004.