

ROSE: A Repository of Education-Friendly Open-Source Projects

Andrew Meneely, Laurie Williams, Edward F. Gehringer
North Carolina State University
Raleigh, NC, USA
{apmeneel, lawilli3, efg}@ncsu.edu

ABSTRACT

Open-source project artifacts can be used to inject realism into software engineering courses or lessons on open-source software development. However, the use of open-source projects presents challenges for both educators and for students. Educators must search for projects that meet the constraints of their classes, and often must negotiate the scope and terms of the project with project managers. For students, many available open-source projects have a steep learning curve that inhibits them from making significant contributions to the project and benefiting from a “realistic” experience. To alleviate these problems and to encourage cross-institution collaboration, we have created the Repository for Open Software Education (ROSE) and have contributed three open-source projects intended for an undergraduate computer science or software engineering course. The projects in ROSE are education-friendly in terms of a manageable size and scope, and are intended to be evolved over many semesters. All projects have a set of artifacts covering all aspects of the development process, from requirements, design, code, and test. We invite other educators to contribute to ROSE and to use projects found on ROSE in their own courses.

Categories and Subject Descriptors

K.3.2. [Computing Milieux]: Computing and Information Sciences Education – *computer science education*

General Terms

Management, Documentation, Design, Verification

Keywords

Open-source software repository, software engineering curriculum

1. INTRODUCTION

In 2006, ACM President Dave Patterson made four suggestions for reinvigorating the computer science curriculum to draw more students to the discipline [11]. One of those suggestions was called “Courses I Would Love to Take #1: Join the Open-Source Movement.” Patterson’s reasoning behind his suggestions is

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ITiCSE’08

Copyright ACM

familiar. First, educators seek exciting and realistic projects, as can be provided by open source software projects. Students today are looking for meaningful assignments that will train them for their future profession [8]. Second, employers, whether they use/develop open source software or not, benefit from hiring students who can evolve and maintain existing software.

Patterson states, however, that the greatest challenge in using open-source software in a course is that students can be overwhelmed by a large code base [11], interfering with learning the conceptual material of the course. In our experience, other challenges exist, such as lack of documentation, automated tests, and other artifacts along with time-consuming coordination issues with the project owners.

Computer science and software engineering courses need non-trivial, meaningful projects that are also usable within the constraints of a class setting. *To help address the problem of providing realistic computer science course projects, we have deployed an open-source software engineering course project repository, the Repository for Open Software Education (ROSE). We contribute to ROSE three education-friendly open-source projects intended for an undergraduate software engineering course.* The three open-source projects we are contributing began as projects at our institution and have evolved into mature projects intended to be as realistic as possible while still being valuable in the classroom. In contrast to other project repositories, such as the Nifty Assignments¹ repository, the intent is for educators to have classes critique, test, and enhance the projects as opposed to replicating them. By providing realistic course projects, we believe we can better train students for professional software development.

In Section 2 of this paper, we discuss the trials and tribulations of using open-source projects for undergraduate courses. We describe ROSE in Section 3. In Section 4, we describe the first three education-friendly open-source software contributions. We describe course integration and lessons learned in Section 5.

2. USING OPEN SOURCE PROJECTS: TRIALS AND TRIBULATIONS

We teach an undergraduate software engineering course that centers on a semester-long project for students. The course is a core/required class typically taken in the third or fourth year of the curriculum. On average, approximately 60 students take the course each semester. In 2006, in an attempt to inject a sense of realism into the entire course, we searched Sourceforge² for an

¹ <http://nifty.stanford.edu/>

² <http://sourceforge.net>

open-source project through which we could demonstrate the course learning objectives. Our criteria for a project were:

- Have social relevance [8];
- Be implemented in Java;
- Have a comprehensive set of automated test cases, preferably JUnit³;
- Have a requirements document;
- Have other documents, such as class diagrams, sequence diagrams, or black box test plan; and
- Be manageable in code size (less than 10 thousand lines of code).

Intuitively, open-source projects appear to satisfy many of our criteria for a course project. After all, open-source projects tend to vary in size and are often used in a production environment with a well-defined user base. Literally thousands of these projects exist, so finding one that fits our class should be easy, right?

After scouring through many projects on Sourceforge and seeking the advice of an expert at Red Hat, we were unable to find a single project that satisfied our criteria. Project artifacts, such as requirements documents and source documentation, were not up-to-date or simply did not exist. Projects with verification and validation artifacts such as an automated test suite or a system test plan were usually the most difficult to find.

The code bases of most open-source projects were not designed for automated testing, such as JUnit. For the few projects that did have an automated test suite, the coverage was minimal. Integrating new tests into a project that did not already have tests is nearly impossible. Writing automated tests after code is complete often requires significant reimplementing or redesign. We emphasize automated unit testing in our course. Students have responded well to unit testing in an educational setting [2, 10], and unit testing is an important part of the software engineering curriculum [13]. The students are required to become familiar with a larger code base than they are accustomed to, and automated unit tests are valuable for regression testing when changes are made.

We suspect that using an open-source project would have been difficult anyway, as other educators have encountered that the “barrier to entry” for most open-source projects is large and quite daunting [1, 12]. Open-source projects tend to have a steep learning curve in terms of deployment, design, and use of technology. We as educators can forget that students are still becoming skilled at software development and are easily overwhelmed by the many complexities surrounding large projects.

Although we were unable to find an open-source project that fit our criteria, other educators have succeeded in adapting open-source projects to computer science curricula [3-6, 9]. Of particular note is the success had by Ellis et al. in their use of an open-source, humanitarian project in a series of independent studies and summer research [5, 6]. Students and faculty worked on a large web-based project used to coordinate various social services in a time of crisis. The project centered on building a new

component and integrating it with the rest of the system. Among the challenges associated with their project is the need for collaboration with project members in what they called a “chaotic” and “fluid” environment where “requirements changed on the fly” [5]. Constant communication and coordination with project managers, developers, and users was needed. Students came away from the experience with a sense of appreciation for both the needs and the unique challenges that humanitarian projects bring. While the project fit well for graduate students, we would not be able to replicate their experience with 60-70, relatively-inexperienced undergraduates.

Other faculty members in our university have, however, had success with using open-source software in the classroom. The third author had 62 masters-level students in an object-oriented design course participate in three open-source projects [7]. The product owners proposed a total of ten assignments for the students, nine of which were ultimately implemented by the students. Students found them comparable to other academic programming assignments in degree of difficulty and in educational value. Two out of three of the managers were pleased with the results, one of them very much so. The only reservation was that one manager concluded that the submissions did not satisfy the project requirements. An important lesson learned is to work with the project managers far enough in advance to clarify the requirements and to give students a significantly long period of time to work on the project. Two project managers have stated that they believe the students would need at least six weeks to complete a successful project.

For our large undergraduate class, we could not manage an “industrial strength” open-source project as did Ellis and Gehringer. We believe that educators should not have to “reinvent the wheel” for their course projects, which is why we have created a repository to suit a common need.

3. AN EDUCATIONAL OPEN-SOURCE REPOSITORY

We have created an education-friendly, open-source repository for computer science course projects, particularly for software engineering courses. The repository not only serves as a resource for artifacts of useful projects, but supports collaboration across institutions to improve projects. Educators can share requirements, ideas for features, and experiences with a particular project or technology [8]. Like the open-source movement itself, they would collaborate not only on ideas, but on the actual artifacts themselves. The result would be a diverse set of projects ready for educators and fit for training students in maintaining and contributing to large projects (a highly marketable skill).

Our repository, called the Repository for Open Software Education (ROSE), is a website⁴ containing links to various education-friendly, open-source projects. The site provides details on the projects such as the criteria we have for our course projects, as shown in Figure 1. ROSE also contains a wiki for all of the projects, so that collaborators can share ideas and experiences, shown in Figure 2. Content of the wiki includes experiences, reviews of the current materials, ideas for future assignments, and designs for possible extensions of each product. Reviews, such as those in Figure 2, are from those who adapted the project in their own context.

³ <http://junit.org>. JUnit is a framework for automating unit testing of Java code.

⁴ <http://agile.csc.ncsu.edu/rose>

ROSE - Repository for Open Software Education

Welcome to the Repository for Open Software Education! The goal of this repository is to keep track of "education-friendly" Open Source projects.

Project Name	Languages	Graduate or Undergraduate?	Unit testing?	Acceptance testing?	Test plan?	Requirements document?
iTrust	Java, JSP, SQL	Both	Yes, JUnit	Yes, FIT	Yes	Yes
CoffeeMaker	Java	Undergraduate	Yes, JUnit	Yes, FIT	Yes	Yes
RealEstate	Java	Both	Yes, JUnit	Yes, FIT	Yes	Yes

iTrust
Version: v4.0
Lines of Code: ~7,700
OS License: GPL

This project involves the development of an application through which doctors can obtain and share essential patient information and can view aggregate patient data. Currently, access to a patient's history regarding previous medical problems, previous surgery, medications, allergies and other factors is often difficult or obtainable only from a patient's recollection. Now as more hospitals and doctor's offices are automated, this information is available electronically, but is not accessible by other doctors and is often only viewed through some proprietary software so it can not be shared.

The final product is a site where health care workers can access important patient information, the non-emergency access can be controlled, and all access would be tracked. Security and privacy of such a system is of paramount importance. HIPAA rules protect patient's information and also allow a patient to dictate who can access this information.

[Top]

CoffeeMaker
Version: v1.0
Lines of Code: < 1,000
OS License: GPL

The Computer Science department at NCSU is in the process of building a new CSC building on Centennial Campus. We all know that computer scientists love caffene, so the CSC department is planning on installing a CoffeeMaker in a lounge across the hall from the 24-hour computer lab. Our job is to test and model the functionality of the CoffeeMaker. We are only working with the logic code behind the hardware, so only a command line interface is used.

[Top]

RealEstate
Version: v1.0
Lines of Code: < 1,000
OS License: GPL

The RealEstate example was created by two graduate students at North Carolina State University to be used as a teaching example in the undergraduate software engineering class. This example was formerly know as the Monopoly Example, but was changed to avoid infringing the Monopoly trademark.

The premis of RealEstate is to provide a teaching example of a common board game that most students will be familiar with. Some of the rules have been simplified for ease of programming. Adding these rules in could be an excellent teaching example (e.g. three tries to roll doubles to get out of jail, free parking wins the kitty, etc.)

[Top]

Created and maintained by [Andy Meneely](#) and [Ben Smith](#).

Figure 1: Screen capture of the ROSE website, found at <http://agile.csc.ncsu.edu/rose>

[[realestate:userinput]]

THE ROSE WIKI

Edit this page | Old revisions

Trace: » iTrust » start » realestate » userinput

Recent changes | Search

RealEstate

[Back to RealEstate Home](#)
[Back to ROSE Home](#)

Table of Contents

- RealEstate
- Reviews
- Comments

[Edit](#)

Reviews

At the University of Virginia, we've used RealEstate for two terms in our CS2 course, CS201. This course is a 2nd programming course in Java with an emphasis on SW Engineering ideas. Since this is a very early exposure to SW engineering, we focus on the RealEstate code and not the other artifacts. We use it as follows:

- We describe the homework using RealEstate as a *software maintenance* activity. Students have studied the SW lifecycle early in the course. The homework is the first time they have looked at a large program and tried to make sense of many classes that they did not write.
- Inheritance: Students have just learned about inheritance, and we ask them to add new classes for certain kinds of Cells that are found in the real Monopoly program. We also ask them to extend the Dice class to support Die that have different numbers of sides (rather than just six-sided die). They see their new subclasses being used with polymorphism in other parts of the program, thus seeing a realistic example of how inheritance extends a system by adding new objects that are compatible with other parts of an the application.
- Additional features: We asked gave them a requirement statement and a partially-completed implementation for mortgaging and un-mortgaging for property cells and asked them to complete this feature.
- Testing: They had studied unit testing early in the course, so we asked them to create JUnit tests for some of the features they added. They patterned this on existing JUnit classes that come with RealEstate. We think this eases them in to JUnit without overwhelming them but letting them see an extensive test-suite.

Again, as noted, this was as second programming course, and the amount of software engineering they had learned was lighter than would be expected of students in a 3rd or 4th year SE course. But having a large working program like RealEstate that was an application they could use and understand allowed us to make some 2nd-course programming activities have a software engineering flavor that is important in our course. RealEstate meets our needs for CS201, and in general the code is a high-quality artifact that can be used for a variety of uses.

I have also had students in a 3rd course on OO design study the code to evaluate its package structure and explore design patterns that are used in the code.

Tom Horton (Dept. of Computer Science, Univ. of Virginia)

Figure 2: Screen capture of the ROSE website, found at <http://agile.csc.ncsu.edu/rose/wiki>

Since the repository comprises of open-source software, most of the actual project artifacts will remain on separate sites such as Sourceforge. For contributors of new projects to ROSE, projects will be added by hand and a space on the wiki will be set up for each new project. For those who wish to evolve a current product of ROSE, we encourage coordinating with the project owners, however, this is not required. All are welcome to download project artifacts and use them for educational purposes.

The concept of an educational course materials repository is certainly not new. Other similar projects, like the Nifty Assignments project or MIT's Open Courseware project⁵, provide course materials and assignment ideas to other educators in an open-source fashion. While our repository contains ideas, experiences, and course materials, the projects in our repository are selected to be evolved and maintained over the course of many semesters. The goal of ROSE is not only to provide educators with education-friendly artifacts, but to coordinate a community around each project with the common goal of enhancing and evolving open-source projects while educating students.

One advantage of having a repository like ROSE is that, even if the actual project code is not evolved by another class, the ideas and requirements can be shared. Alternatively, a class can use the ROSE artifacts for a variety of tasks related to the learning objectives of the class beyond just evolving and enhancing the code base. Some example activities could include the following:

- test a project and report failures and/or security vulnerabilities to Bugzilla on Sourceforge;
- inspect the requirements, and/or
- evaluate the design.

If a project grows large, it can be forked into another project while maintaining its education-friendly characteristic. In the case of ROSE, a history and motivation for forking will be provided to the community. The open-source movement itself has proven that, given the proper coordination, projects grow and mature out of a need from the community.

4. THE FIRST THREE ROSE PROJECTS

As the first contribution to ROSE, we present three open-source projects, discuss their integration into the course, and describe the benefits to students.

4.1 iTrust

iTrust is a healthcare system written as a J2EE web application. Originally developed by graduate students in a Software Testing and Reliability course, iTrust has been maintained over four semesters as the course project for our undergraduate software engineering course. At the beginning of the semester, students are given the full project including all available artifacts: use case-based requirements, the code, a system test plan, and an automated test suite. The requirements for iTrust have come from consulting with members of the online medical records industry and have been designed to be compliant with the Health Insurance Portability and Accountability Act of 1996 (HIPAA⁶). The code is

implemented in Java Server Pages (JSP) and Java, using a MySQL database. The automated test suite consists of unit tests in JUnit and functional tests in FIT. The development environment is Eclipse.

4.2 Coffee Maker

Coffee Maker is a Java command-line program that students can use in learning automated testing with FIT and JUnit. The Coffee Maker program simulates interacting with a coffee machine to create various kinds of drinks. Inventory on ingredients is kept and recipes can be added or changed. For the first few labs of the course, students work with a fault-injected version of Coffee Maker to become familiar with the Eclipse environment and to write effective test cases. Requirements are provided with Coffee Maker so that students can find discrepancies between the specifications and the actual implementation. Tutorials on introducing JUnit and FIT using Coffee Maker as the running example are also included in the repository. Students have typically reacted well to starting with Coffee Maker as the command-line program is the kind of software they are most familiar with coming into our course.

4.3 Real Estate

Real Estate is a game based on a popular board game where properties are bought and sold. Real Estate is a Java application driven by a graphical user interface (GUI). Also included are requirements, automated test suite in JUnit and FIT, a black-box test plan, and UML diagrams. The Real Estate software is intended to introduce non-trivial software that emphasizes testing and brings educational depth to tutorials. One of the strengths of Real Estate is its basis on a well-known board game which nearly all students are familiar with, leading to swift adoption of the projects requirements and features.

5. COURSE INTEGRATION AND LESSONS LEARNED

We have used the iTrust project as the semester project in our undergraduate software engineering course for the past two years. The course is a required course for our institution's Computer Science curriculum and is intended to prepare students for a career in professional software development. Each semester, the course has approximately 60 students who are in their third or fourth year. The course consists of two weekly 50-minute lecture sessions and one weekly 2-hour lab session. The lab sessions each week provide students with hands-on experience through tutorials and other teaching activities.

The course has four major assignments over the semester, each directly evolving and maintaining the course project. For the first two assignments, students work in pairs. For the third assignment, student work alone. The fourth assignment is the team project for which the students work in teams of four or five for a six-week project. Each assignment requires students to write automated tests for new features or for maintenance and to ensure all previously-written test cases continue to pass. The iterative nature of the assignments and projects greatly helps students maintain control over their code and improves the overall reliability of their work.

⁵ <http://ocw.mit.edu/>

⁶ www.hipaa.org/

Prior to our class, most students had only experienced software development where they developed code from scratch. Unfortunately, however, students find that when they obtain software development jobs, they seldom get to start from scratch and instead are involved in software maintenance. By providing a code base up front, students learn the valuable skills of learning new architectures and traversing code.

In particular, using iTrust has had some unique highlights in our course. iTrust contains several socially relevant aspects to it, namely, the purpose of the project and the technology it employs. Studies have shown that students of the Millennial generation engage more with programming projects that have social relevance than those that do not [8]. We find that many students do become engaged with the project because they feel they are making a meaningful contribution. Through working with the iTrust project, students learn about privacy, privacy policies, and compliance to legal standards; iTrust is intended to be HIPAA compliant. Using medical standards in the project, such as the ICD9CM⁷ diagnosis or CPT⁸ procedure codes, makes the whole experience feel less simulated than their previous class projects and challenges to the students to look into how this software could be developed in the most concrete, real way possible.

iTrust is a web application that deals with very sensitive data. Through working with the iTrust project, students learn about the importance of security and incorporating good security practices into the implementation code. As a result, part of the students' workload is to fix particular security vulnerabilities in the code. In keeping with the realistic aspect of the project, all vulnerabilities assigned were originally introduced by former students' work on the code base (as opposed to being injected by the instructors).

The iTrust requirements are written by the instructors using suggestions and feature ideas from industry experts. As such, the requirements are sometimes unintentionally ambiguous. We release an initial requirements statement to the students prior to actually assigning it. As a lab activity, students are asked to inspect the requirements for feasibility and clarity. Students are required to provide constructive corrections to the requirements that would communicate the purpose of the feature better. The requirements are then revised based on student feedback and subsequently released for the assignment. While students are not actually *writing* requirements, they learn about the challenge in writing feasible, clear requirements by conducting an inspection.

6. CONCLUSION

In our experience, building a software engineering course around a realistic, education-friendly, and socially relevant project has proven to be a valuable experience to both students and educators alike. Going to open-source solutions, however, is often too difficult for educators and for students. To help address the problem of providing realistic computer science course projects, we have deployed an open-source software engineering course project repository, the Repository for Open Software Education (ROSE). We contribute to ROSE three education-friendly open-source projects intended for an undergraduate software engineering course. We believe that using projects like iTrust for a computer science course would alleviate the difficulties of using open-source projects in a classroom setting while maintaining the

realistic quality of the course. We hope that others will find our repository to be a good starting point for benefiting from and contributing to a better collection of education-friendly open-source projects. We invite other educators to contribute to ROSE and to use ROSE projects to the benefit of their students.

7. ACKNOWLEDGMENTS

We would like to thank Ben Smith for helping in the deployment of ROSE and for Sarah Smith and Chih-wei Ho for their development of the Coffee Maker and Real Estate programs. We would also like to thank Lucas Layman for his helpful comments. This work is supported by the National Science Foundation under CAREER Grant No. 0346903. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

8. REFERENCES

- [1] Allen, E., Cartwright, R., and Reis, C., "Production Programming in the Classroom," in 34th SIGCSE technical symposium on Computer science education, Reno, Nevada, USA, 2003, pp. 89-93.
- [2] Bowyer, J. and Hughes, J., "Assessing Undergraduate Experience of Continuous Integration and Test-Driven Development," in 28th international conference on Software engineering, Shanghai, China, 2006, pp. 691-694.
- [3] Carrington, D. and Kim, S. K., "Teaching Software Design with Open Source Software," in Frontiers in Education, 2003. FIE 2003. 33rd Annual, 2003, pp. S1C-9-14 vol.3.
- [4] Dionisio, J. D. N., Dickson, C. L., August, S. E., Dorin, P. M., and Toal, R., "An Open Source Software Culture in the Undergraduate Computer Science Curriculum," SIGCSE Bull., vol. 39, no.2, pp. 70-74, 2007.
- [5] Ellis, H. J. C., Morelli, R. A., Lanerolle, T. R. d., Damon, J., and Raye, J., "Can Humanitarian Open-Source Software Development Draw New Students to Cs?," in 38th SIGCSE Technical Symposium on Computer Science Education, Covington, Kentucky, USA, 2007, pp. 551-555.
- [6] Ellis, H. J. C., Morelli, R. A., Lanerolle, T. R. d., and Hislop, G. W., "Holistic Software Engineering Education Based on a Humanitarian Open Source Project," in 20th Conference on Software Engineering Education & Training, 2007, pp. 327-335.
- [7] Gehringer, E. F., "Open Source for Homework: Real Projects, Peer Reviewed," in SIGCSE '07: 38th Technical Symposium on Computer Science Education, 2007, <http://research.csc.ncsu.edu/efg/oo/presentations/sigcse07-OSS-poster.ppt>.
- [8] Layman, L., Williams, L., and Slaten, K., "Note to Self: Make Assignments Meaningful," in 38th SIGCSE Technical Symposium on Computer Science Education, Covington, Kentucky, USA, 2007, pp. 459-463.
- [9] O'Hara, K. J. and Kay, J. S., "Open Source Software and Computer Science Education," J. Comput. Small Coll., vol. 18, no.3, pp. 1-7, 2003.
- [10] Olan, M., "Unit Testing: Test Early, Test Often," J. Comput. Small Coll., vol. 19, no.2, pp. 319-328, 2003.
- [11] Patterson, D. A., "Computer Science Education in the 21st Century," Communications of the ACM, vol. 29, no.3, pp. 27-30, March 2006.
- [12] Pedroni, M., Bay, T., Oriol, M., and Pedroni, A., "Open Source Projects in Programming Courses," in 38th SIGCSE Technical Symposium on Computer Science Education, Covington, Kentucky, USA, 2007, pp. 454-458.
- [13] Wick, M., Stevenson, D., and Wagner, P., "Using Testing and Junit across the Curriculum," in 36th SIGCSE Technical Symposium on Computer Science Education, St. Louis, Missouri, USA, 2005, pp. 236-240.

⁷ <http://www.cdc.gov/nchs/icd9.htm>

⁸ <http://www.ama-assn.org/ama/pub/category/3113.html>