Challenges and the Elements of Success in Undergraduate Research

Elaine Wenderholm
Computer Science Department
SUNY Oswego
Oswego New York 13126 USA
wender@cs.oswego.edu

Abstract
Faculty are often hesitant to undertake undergraduate research projects since they often ask themselves whether undergraduates have sufficient training and knowledge to conduct research. Additionally faculty researchers may not have experience in project planning and management. This paper suggests effective strategies for success, based upon my experience directing a four-year research project.

Keywords: Undergraduate research, project planning, project management, eXtreme Programming

1. Introduction

Undergraduate research is widely recognized as having a positive impact on not only the educational experiences of undergraduates, but also in providing research opportunities for faculty at undergraduate institutions [2,4]. In practice, it is often difficult to achieve success with undergraduate researchers.

This paper presents experience gained on a multi-year research project, "A Toolbox for Assembling Spatially-Explicit Multimedia Ecological Models from Reusable Components" which was funded by the U.S. Environmental Protection Agency STAR (Science to Achieve Results) program, Assistance Agreement No. R–82795801. This was a joint effort between research ecologists at the Boyce Thompson Institute for Plant Research at Cornell University, and my Computer Science undergraduate research group at SUNY Oswego.

I led a total of eleven undergraduate students over a period of four years through the process of taking the stated needs of our research group of ecologists and developing a Java(TM)-based framework [3,5] that accomplished every goal we initially established. This framework consists of a nontrivial (about 34K LOC) suite of integrated GUI editors, APIs, and a (source-to-source) compiler. Ecologists use these editors to specify spatial ecological models. The framework's compiler to generate simulation code, either for sequential or shared-memory parallel execution uses these specifications.

2. Key Challenges and Solutions for Success

2.1 Challenge 1

Identifying the "ideal" students for a multi-year research project.[1]

Faculty recommendations are invaluable, but it is my experience that motivated students are not usually identified until the end of their junior year. Since most student work is accomplished during the summer, this short time frame is not effective because of the learning curve that is often required, especially in the later stages of the project. Sophomore undergraduates can work for two summers but they are harder to identify and they also have less training.

As a strategy for identifying lower-division students as early as possible, I suggest the following:

- Teach as many sophomores in required CS courses as you can to get first-hand knowledge of their strengths and weaknesses.
- Look for initiative beyond course material. The CS curriculum at Oswego requires students to maintain a web page, largely of their own design, which includes links for course work. This can give additional insight into student motivation. For example, one student, during his second semester sophomore year (while taking CS2), wrote and provided Javadoc(TM) documentation for an applet for Tetris. Not only did he

[1] I define "ideal" as a lower-division student who has academic potential, is highly motivated, creative, self-starting, and works well with others.

have to learn Swing on his own, but also it was apparent that he had a natural aptitude for software design.

- Ask these potential students who they might like to work with and why.

2.2 Challenge 2

Successfully planning and managing an undergraduate research project.

Project planning and management typically falls into the following phases:

a. Proposal-writing phase

Design a budget so that you can hire a minimum of two students. Try to pay an hourly wage that you would expect for entry-level programmers in your geographic area, even if exceeds the guidelines for typical work-study students on your campus.

Make reasonable project milestones that do not interfere with the responsibilities that students have during the academic year.

b. Interview phase

I told each student that I had basically one job requirement: *"You have to be totally committed to the project. Don't feel obligated to put in X hours per week, especially when classes are in session. But just like working in the real world, you have to meet deadlines, and so you have to put in the time when necessary."*

c. Research phase

- Have the students practice Extreme Programming [1] techniques. You will find that productivity and morale thrive.
- If you are providing a product for a client, have your students sit in on design meetings. Introduce them to the clients; tell the clients (in their presence) that they are great students. Let them see how someone (you) who has experience in these aspects of defining and refining requirements, evaluating prototypes, etc., actually works with clients. Afterwards guide them in the steps they need to take.
- As espoused by Extreme Programming, start by having students prototype just the front-ends to any GUIs for client approval. Always remember that good software engineering practices should prevail.

Schedule a time for once-weekly group meetings to discuss the project. Most importantly *let them have fun*! I gave my students freedom and lots of creative license. They not only came up with great ideas but also implemented them, much to the delight of the research ecologists.

2.3 Challenge 3

Overcoming lack of knowledge

Personnel problems aside, this is the most difficult and critical aspect of undergraduate research. A research project of nontrivial size that spans several years initially should be staffed with more lower-division than upper-division students, with the hope that the lower-division students will continue to work on the project though their senior year. Extreme Programming prototyping is invaluable especially in the beginning of the research project. Students are inexperienced, and throwing away and starting again with increased knowledge not only leads to a much better product but also serves as an excellent training ground.

My research project required a skill set in several areas. Implementing GUIs and developing competence in working with different operating systems (Solaris, Linux, Mac OS X, Windows) is challenging but is learned relatively easily by lower-division undergraduates. The challenging aspects of the research relied on subject areas that typically are not covered until the junior and senior years. These areas included Java threads, Java reflection, shared-memory parallel programs, fork-join techniques, networking, compiler analysis, code generation and optimization, continuous-time simulation, finite difference methods for solving PDEs. It is impracticable for the project director to replace course content learning with individualized instruction.

Incorporate upper-division courses into the project planning and milestones *and* have upper-division courses support undergraduate research. The structure of upper-division courses in the CS curriculum at Oswego makes this successful for two reasons: First, the majority of our upper-division CS courses require term-projects. Second, the instructors of these courses allow research students to fulfill their term-project requirements by working on the relevant projects of their research. One could suppose that the other students in these courses might resent this. After all, we pay research students to do their projects! However, this is not the case. Students recognize that research projects typically have greater challenges than non-research projects. More importantly, it invariably leads to greater student interest and excitement for the course material and higher academic goals. In addition, plan with your research students which relevant courses to take at which point in their undergraduate schedules, and have them take these course together so they work together on the course's term project.

Acknowledgments

The following Computer Science undergraduate students, listed in no particular order, worked on this research project for as little as a few months to several years: Anthony Vito, Kennedy Roberts, Edward Jones, Timothy Martin, Thomas Smart, Erick Smith, Phillip Meyer, Julie Strempel, Kevin Kronnenbitter, Jon Markoff, and Brian Trowbridge. The success of this project is due to their efforts. Thanks to my colleague Doug Lea for supporting student research in his courses.

References

[1] Beck, K. Extreme Programming Explained: Embrace Change, Second Edition. Addison-Wesley:MA, 2004.
[2] Council on Undergraduate Research homepage. http://www.cur.org/ (August 2004).
[3] Eclpss homepage. http://www.cs.oswego.edu/~wender
[4] National Science Foundation Research Experiences for Undergraduates homepage. http://www.nsf.gov/home/crssprgm/reu/
[5] Wenderholm, E. Eclpss: a Java-based framework for parallel ecosystem simulation and modelling. Environmental Modelling and Software, 2004 (to appear).