Rapid Peer Education of a Computational Nuclear Engineering Skill Suite

Kathryn D. Huff, Anthony M. Scopatz, Nicholas D. Preston, Paul P.H. Wilson
1500 Engineering Drive, University of Wisconsin, Madison, WI, 53706
khuff@cae.wisc.edu

INTRODUCTION

Detailed reactor models, massively parallelized calculations, and enormously collaborative simulation projects are increasingly integral to nuclear engineering. However, the quality and caliber of this work is limited by a workforce lacking formal training in a software development skill suite that is becoming increasingly essential. To address this unmet need, The Hacker Within (THW), a student organization at the University of Wisconsin, has developed a series of short courses addressing best practices such as version control and test driven code development, as well as basic skills such as UNIX mobility. These 'Boot Camps' seek to provide time efficient introductions to essential programming languages and tools without turning "biochemists and mechanical engineers into computer scientists."[1][2]

MOTIVATION

If sophisticated computational efforts in nuclear engineering are to succeed, the workforce must be versed in code development methods. While the straightforward coding projects of the past were handily written by self-taught scientists, the body of knowledge necessary to competantly produce state-of-the-art software today is no longer within the wheelhouse of the scientist.[3] Worse, the fundamental tenants of scientic endeavor (such as data control, reproduciblity, comprehensive documentation, and peer review) suffer in projects that fail to make use of current development tools such as unit testing, version control, automated documentation and others.

Analgous to experimental instrument testing and calibration, industry standard software testing methods which systematically verify the accuracy of each unit of code will be prerequisite to the success of ambitious projects underway in nuclear engineering today. Soberingly however, a recent review indicates that fewer than 47% percent of scientists conducting computational research reported a good understanding of software testing.[4] Another practice that will be essential to enormously collaborative efforts is the use of version control. Version control tools which monitor file changes during development enable the reproduction of output from any previous code revision. This tool in particular will empower nuclear engineers to confidently release their code and review the code of their collaborators; however, it is sorely underused in scientific computation today.[5][4]

CURRICULUM DEVELOPMENT

Four Hacker Within Boot Camps have emphasized an open, time efficient, exercise driven curriculum model. Content was contributed by THW student members who led interactive lessons during the multi-day Boot Camps. Curriculum development took place collaboratively on a public wiki such that a single lesson was often the collective work of many contributors. Lecture videos, notes, and example exercises were freely available online.[2] Feedback taken at each of the Boot Camps led to focused adjustments of the format where necessary (see Fig. 1).

Topic	Year	Days	$\frac{hr}{day}$	Attendees
Unix	2009	4	2	18
C++	2009	4	2	30
Python	2010	3	4	82
Software Carpentry	2011	3	4	79

TABLE 1: Previous Bootcamps have been conducted in two condensed formats. Feedback surveys from early bootcamps led to a three rather than four day structure.

While course content was primarily driven by what the graduate students leading the course found useful in their own work, content in first Boot Camp on UNIX mobility and the most recent on Software Carpentry were influenced by an open curriculum Software Carpentry course by Greg Wilson of Toronto.[1] This content included lessons on the shell environment, version control, text editors, debugging, testing, automated documentation, databases, and web programming.

OBSERVATIONS

Registration, feedback, and website statistics have informed observations and recommendations concerning the target audience, the appropriate level of instruction, and specific content for which an unmet need exists. Registration statistics for the Python and Software Carpentry bootcamps indicated an audience dominated by technical graduate students with intermediate programming skills. However, an appreciable number of undergraduates, professors, staff and members of the public also attended (Fig. 1) and the students spanned many levels of computational backgrounds (Fig. 2).

Skill variation of the audience presents a challenge to curriculum design. The THW model has sought to answer this by emphasizing basic skills in early lessons and presenting increasingly advanced tools in later lessons with decreasing levels of detail. Similarly, the diversity of disciplines demands that examples and explanations be presented as free of jargon

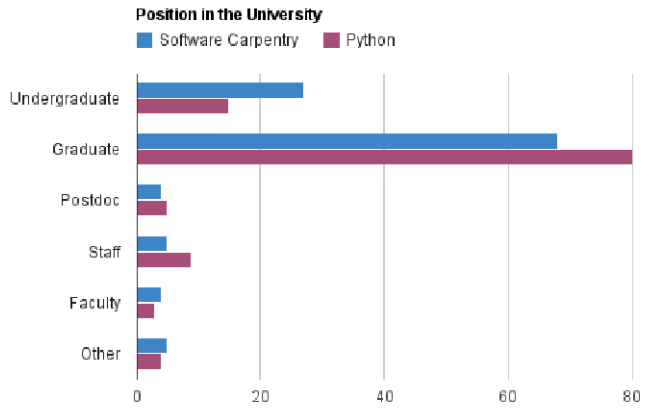

Fig. 1: Registration and attendance was diverse, but dominated by graduate students.

as possible. Though these solutions potentially sacrifice detail for clarity, expediency and accessibility are well served by this model.

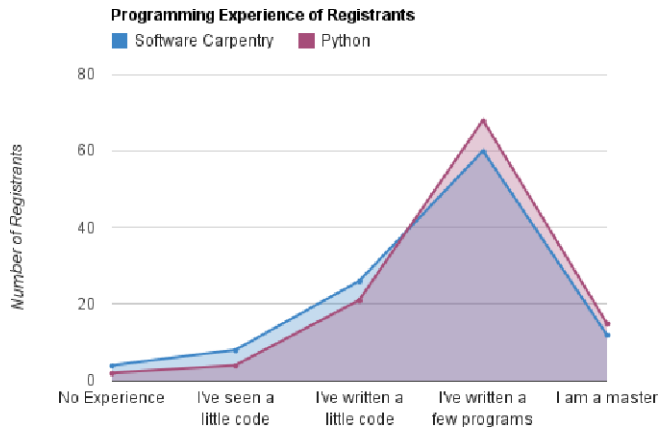

Fig. 2: Self reported programming experience demanded a curriculum that appealed to varied skillsets.

Finally, feedback indicated that some course content suggested greater future potential than other content. Specifically, students self reported the likelihood that in the future they will use various topics covered in the four bootcamps. Favorite topics included the shell, object orientation, and text editors as well as scientific, numerical, and statistical python libraries, automated documentation, version control, and build systems.

RECOMMENDATIONS

Based on the successes of the THW boocamp model for both student attendtees and teachers, the authors strongly recommend that similar peer-education groups be implemented at other institutions. Evidence for the breadth of impact of these bootcamps has been demonstrated above. Additionally, anecdotal knowledge indicates that all students involved become more effective and productive when they return to nuclear engineering. In addition to further bootcamps, future work will attempt to develop metrics by which the productivity increases may be measured.

ACKNOWLEDGEMENTS

This work is the result of the unparalleled dedication and enthusiasm of these authors as well as fellow founders Milad Fatenejad, Kyle Oliver, and Matthew Terry. Other noteworthy contributors to these efforts include Matthew McCormick, Rachel Slaybaugh, Matthew J. Gidden, Aronne Merrelli, Kurt Smith, Jim Porter and Greg Wilson. Finally, our activities have been sponsored by the Associated Students of Madison, the Wisconsin Experience Grant, and generous benefaction from the office of the CIO of the University of Wisconsin.

REFERENCES

1. G. WILSON, "Software Carpentry: Essential Software Skills for Research Scientists," (2006).
2. "the.hacker.within," http://hackerwithin.org/ (2010).
3. Z. MERALI, "Computational science: ...Error," *Nature*, **467**, 7317, 775–777 (2010).
4. J. E. HANNAY, C. MACLEOD, J. SINGER, H. P. LANGTANGEN, D. PFAHL, and G. WILSON, "How do scientists develop and use scientific software?" in "Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering," (2009), pp. 1–8.
5. G. V. WILSON, "Where's the real bottleneck in scientific computing?" *American Scientist* (2010).