POSSE BASICS

Cultural Immersion Weekend

CC-BY flickr user Angell Williams
POSSE
PROFESSOR'S OPEN SOURCE SUMMER EXPERIENCE

A Red Hat community service

SATURDAY

Game Plan

09:00: Workshop start
10:30: Reflections/Break
12:00: Lunch at Red Hat
14:30: Reflections/Break
16:00: Feedback/Break
18:00: POSSE Dinner

Opensource.com interview times (approximately 10 min. each)
11:00 – 12:00: Red Team
13:00 – 14:00: Blue Team
14:00 – 15:00: Green Team
15:00 – 16:00: Yellow Team

Today, we will be doing...

Hi REMOTE INTRODUCTIONS

Go THE FIRST 15 MINUTES

POSSE Hotel
Four Points Cary
500 Caitboo Avenue
Cary, NC 27518
6 miles / 15 minutes from Red Hat

POSSE Workshop
Red Hat Headquarters
1801 Varsity Drive
Raleigh, NC 27606

POSSE Dinner
Tobacco Road
222 Glenwood Avenue
Raleigh, NC 27603
tobaccoroadsportscafe.com
3 miles / 11 minutes from Red Hat
8 miles / 16 minutes from hotel

SUNDAY

Game Plan

09:00: Start
10:30: Reflections/Break
12:00: Lunch at Red Hat
14:30: Feedback/Planning
15:00: Workshop End

Today, we will be doing:

ROADMAP MERGE

NINJAHOOD PLAN

Airport
Raleigh-Durham International Airport (RDU)
1600 Terminal Blvd
Morrisville, NC 27623
13 miles / 20 minutes from Red Hat
12 miles / 16 minutes from hotel

Dining recommendations
Ole Time BBQ (hole-in-the-wall BBQ)
    oletimebarbecue.com
The Pit (upscale BBQ)
    thepit-raleigh.com
Rue Cler (French)
    ruecler-durham.com
Sitti (Lebanese)
    sitti-raleigh.com
Cameron Village (various)
    shopcameronvillage.com

COHORT MAP

And Major Events

A Red Hat community service

RAJEEV AGRAWAL

North Carolina A&T State University, Greensboro NC
ITT 700 Project Management for IT Professionals (Fall)

THE CLASS NOW

This is a graduate course, 15-20 students, taught online. This course is one of the four foundation courses taken by students in the first year and preferably in the first semester of their graduate degree program in Information Technology. Ours is an urban school and most of our students are African-American students. We have very diverse body of students. Most of them work full time, few part time and there are few international students. Most of them know programming, databases and have exposure to Unix/Linux at different levels.

This course covers project life cycle, planning templates, project deliverable, project work breakdown structure, estimating resources and task costs, Gantt charts, PERT techniques, project team duties and responsibilities, project team management techniques, and software tools for large projects. Currently, it is mostly theoretical course based on PMBOK guide of Project Management Institute (PMI). Students do learn concepts, but do not really apply in a real project. The students do pretty good in this course and value its importance. I would like to modify this course by including projects, where students can apply the project management concepts learned.

BY SUMMER 2012

I believe using open source project management tools provided my students a great opportunity to apply their knowledge of project management into real projects. Few students, who already work, have told me that they use the experience learned in this course at their work place and feel prepared to manage larger projects.

WHAT HAPPENS NEXT

I am aware of open source idea, but have never been part of any project.

- I need to create a list of open source articles, possible projects, and the tools to use in this course.
- Students will read the textbook by Karl Fogel about how to produce open source software.
- Fedora 16 release is in the pipeline and its feature freeze will happen around July, which is just before my course starts. I will create a survey to find my students' technical background and based on their expertise, they will be assigned on different fedora modules. They can contribute in meeting minutes analysis, mailing list analysis, commit analysis, ticket analysis. They can create burn-down charts. They can monitor whether feature development/porting is on track. Are there enough people working on a module? Are bugs description clear? They will also write user manual/one page release notes/marketing write up/ release feature profile/QA documentation. They can also work as a part of testing team.
- The Marketing team needs plenty of help translating the engineering work that's already being done out to multiple markets, and if they'll be lurking on various subgroups/projects, blogging and publishing weekly write-ups on their observations would help Fedora contributors keep track of other activities going on, since it's such a massive project and nobody can stay atop all of it. Students can work as ambassadors during Fedora road shows.
- I definitely want my students to learn about version control.
- NCAT will provide access to Linux to all the students of this course. I will provide a Linux tutorial to all the students, which will include basic commands, editor, version control and other important tools, student must know.
- Being closer to RTP, I am sure I can find some speakers, who will be willing to visit Greensboro.
- I have to find a list of possible conferences/ workshops, where my students can represent their work.
- Discussion on how to manage student teams will be useful.

EVELYN BRANNOCK

Georgia Gwinnett College, Lawrenceville GA
ITEC 3860 Software Development I (Fall/Spring)

THE CLASS NOW

GGC is the 35th institution in the University System of Georgia. Founded in 2006, it is the nation's first four-year public institution established in the 21st century. Another interesting point: GGC does not offer tenure. For over 20 years I worked in industry as a consultant, curriculum developer, and architect for software vendors that provide Application Development/Business Process Management/Enterprise Integration Application workbenches. My current students vary from sophomores to graduating seniors (the course has a prerequisite of CS2), with a few of them taking it as an elective. Next fall 2011, there is one section of Software Development I that I will teach, and it has already reached its enrollment cap of 15. Currently I am evaluating the text "Software Development: An Open Source Approach" by Tucker, Morelli, and de Silva, as a text for fall 2011.

Some of the highlights of the current course:

- Use of an open source CASE tool: Students (in a group) were assigned a Course Management Scenario. They were to choose a CASE tool, implement the project and produce a written report. The report included the technical aspects they experience with the tool, and also the project management and communication successes they experienced. They were also required to include the least successful aspects of the project they encountered. A demo of the toolset was also a deliverable.
- Article summary: Independent reading and presentation on an article from a scholarly magazine such as IEEE Software or IT Professional.
- Final Project – Code Format Standards Validation Project: Working as a team, students will create a design (use cases, class diagrams) for a case study.

The course wiki at can be viewed at http://wiki.ggc.edu/mediawiki/index.php/ITEC3860.

BY SUMMER 2012

My goal as a teacher is straightforward. I see every student I educate in my Software Development I class on the path to a career that fulfills them, that they enjoy, and that allows them to lead the lifestyle (that includes family, friends and community) they envision.

WHAT HAPPENS NEXT

- Advice on the best tools that are available to my students that do not have a budgetary requirement.
- Help with maximizing my productivity. GGC is a new school, as opposed to more established colleges, and I often have many obligations outside of course development. What works that I can re-use?
- Advice on the best real world projects that will give my students the skills they need to succeed in future courses, attain future internships, enhance their resume and attain meaningful, fulfilling employment.
- My students to contribute to open source projects; not for a grade, not for extra credit, not because I require it.
- Advice on getting my undergraduate students published, especially for those interested in graduate school.
- Ways to obtain financial help for my students to attend conferences, contests and other opportunities for them to network outside of the GGC community.

WEI KIAN CHEN

Champlain College, Burlington VT
CSI 280 Innovation II: Open Source Software Development (Spring)

THE CLASS NOW

Champlain College is a non-tenure granting teaching institution, and it is situated in Burlington, VT. Over the past ten years, Champlain had a dramatic change, but we are still a professional-oriented institution. Champlain College started Computer Science and Innovation (CSI) program last year housed in the Division of Information Technologies and Sciences. I am the first and only faculty member in the CSI program. The goal of this program is to add innovation into the CS program, whose enrollment had declined over the years.

One component we wanted to add is Open Source. We want to expose our students to this area, which we believe will be very beneficial to them. This course is offered in the Spring semester of the second year and the target audience are sophomores. This is a completely new course.

I want to create a course that better introduces the students to Open Source as a contributor rather than user. The course description is as follows: “Students will explore the history, philosophy, mechanics, and practices of the open source movement. Students will research some of the most prevalent and cutting- edge open source technologies, and gain experience working with the tools and techniques used in open source software development.”

This course is designed as a mini-capstone project, where the students are participating in a real-world project. The students will have to go through writing proposal, progress report, as a final presentation on the "completed" project. Along the project, students are expected to learn about change request, ticket trackers, etc. The language of choice will be C++ since this is what they would have up till this point. The overall aim of this course is project management and technical writing. Of course, this is what I have currently planned, and things might change after this workshop.

When the students take this course, they have had two semesters of C++ (Introduction to Programming and Advanced Programming) and also a semester in Data Structure. They are mainly majors with a few exception from Game Programming, and most are first-generation college students and are motivated in programming related career. They might not be very theoretical oriented and would prefer a more hands-on approach. These students typically have a job from local companies that need programming help when they are in school, and they will join the industry as soon as they graduate. Since they are already working, they usually like to learn about the cutting edge developments that can help them get a job easier when they graduate. Most of them have use open source tools, but none of them have participated in development.

BY SUMMER 2012

In the past, I have always been the user for Open Source and I have used many of the applications out there. However, I have never once contributed into this community. This course had given my students and I the opportunity to know this community better and contributed to it. I believe this is a very valuable learning experience for both my students and I. Since we are the only group in the College that knows Open Source better, we can advocate for it. We will help push the administration toward adopting more applications from the Open Source market.

WHAT HAPPENS NEXT

Since I have never contributed into the Open Source, I will have to do a lot of research to figure out how the community works. If I am able to participate in your program, it will help me tremendously. Not only your program can help me with the material, but it will also introduce me to the members in the community that I can seek help if needed. Currently, there is a lot of unknown for me, and I am hoping to be able to connect all dots with your help.

HEIDI ELLIS

Western New England University, Springfield MA
CS 490 Software Engineering (Fall)
CS 491 Advanced Software Engineering (Spring)

THE CLASS NOW

Western New England University has approximately 2500 undergrads and 1200 grad students. WNE has a very strong teaching emphasis and very much cares about education. WNE students are a work-oriented group. Many are first generation college attendees and most want to see the direct application of the material that they learn in class to work. My impression is that when they enter CS 490, they don't see the utility of the course clearly. By the end of the course, some students "get it", but others are still not convinced. Ad hoc feedback from folks that have graduated and been out indicate that the course is very useful. But I'd like students to get an understanding of the importance of software engineering before they graduate.

CS 490 Software Engineering is a required senior-level CS course (http://mars.wnec.edu/~hellis/CS490/). I taught the course with 11 students this past fall using the GNOME Caribou on-screen keyboard as a project base. In the past, this has been a traditional software engineering course that starts with requirements and goes through maintenance. This past fall, I've used a FOSS project as the base with mixed success. Definitely a learning experience. I have a follow-on course, CS 491 Advanced Software Engineering. (http://www.xcitegroup.org/softhum/doku.php?id=workingpapers)I taught it for the first time this year. In this course, I allow students to apply what they've learned in CS 490 to a FOSS project of their choice. This course had five CS students.

BY SUMMER 2012

The best class I ever had was when I had several teams of students involved in different humanitarian FOSS communities and contributing to the projects. Students now have an understanding of the software engineering aspects of software development and why these aspects are important. They understand the main communication mechanisms of open source and can now communicate effectively in a professional environment. They know how to identify project needs, how to scope a piece of work to be accomplished, how to plan to accomplish the work and how to get the work actually accepted into the project. One of the most cool aspects of the course was how the community provided professional affirmation to my students. Not only did students gain understanding of the mechanics of developing software, the interactions with the community also convinced them that they had the skills to make a valued contribution. They left the course with an understanding that they could do it! I've learned a lot from this course as well. I now know how to help students identify likely projects and how to help them gain entry into those projects, while not inserting myself in the middle. An added bonus was that the contributions were of sufficient visibility that I was able to point out these achievements to my dean. And I'm more confident in helping students learn within projects that I may not know the technologies involved.

WHAT HAPPENS NEXT

In order to achieve my dream, I have few needs, but many wants. My need is for mentors for students within FOSS projects. I also need to have a better idea of the FOSS community attitude towards intermittent contributions by students and how to handle these. On the technical side, I need a better understanding of Git.

Under the "want" category, I'd definitely like a FOSS community member to work with my students to help provide insight into technologies used in FOSS as well as FOSS culture. This person would have hands-on experience in the project, but they are also likely to be closer to my students' age and the near-peer experience has huge impact. I had the opportunity to take a few students to the GNOME Summit and Hackfest and/or to host a hackfest at WNE. I'd also like to be able to fund students to present work at other venues. For instance, to take students into local high schools or to conferences.

And I'd definitely like help in getting my students' work and the course advertised. This is an area in which I have little experience. I think that my students are doing cool things and I'd very much like to raise the visibility of our work.

ANDREA HICKERSON

Rochester Institute of Technology, Rochester NY
Newswriting I (Fall/Spring, quarter system)

THE CLASS NOW

The journalism program at RIT is one of the newest degrees on campus. Launched in 2009, we have over 30 majors and over 100 minors. These numbers surprise some who believe journalism is in decline. Indeed journalism is changing, but it is hardly in decline. Specifically, the programmer/journalist (or the journalist who knows just enough programming to be dangerous) is a hot commodity in newsrooms. RIT is looking to make a mark in producing these types of graduates because our school already excels in computing sciences. To this end, I would like to incorporate open source in my Newswriting I class, the introductory writing course taken by majors and nonmajors across the university. The class is capped at 24, and meets in a computer lab. Typically students who believe they are good writers take this class. They soon learn that Newswriting requires a crisp, balanced, technical style – usually the antithesis of what they learned in high school. I believe that having the students write about/learn about open source is an excellent project on three accounts.

First, it challenges them to write about something technical in a user-friendly way. Truly some of my students are in journalism to avoid math and science, which makes this assignment all the more challenging and beneficial. In Summer 2009 I experimented with open source and had my Newswriting students visit RIT's POSSE and write stories. This turned out great. This year, I would like to repeat the assignment but also to require students to post their stories in an open source community. This ups the ante by making their work public. Also, it allows them to add a real-published piece to their portfolio – win #2.

Another new layer I would like to add is to invite public relations professors to have an open source assignment which would require their students to write about POSSE. The press releases they write could be handed over to my students. This would give my students direct experience working with press releases, something they have to know how to do.

Finally, working on an open source project exposes my students to the open source community. I hope they will be encouraged to pursue other courses in CS, and maybe even CS minors. Right now I do not believe there is as much integration between liberal arts and CS as there could and probably should be. Different programs have different sensibilities that together can foster innovation and creativity. I wonder too if there isn't a model for producing journalism that already exists in a different form in the open source community.

BY SUMMER 2012

The 2011-2012 academic year was a banner year. All of my students had at least one piece of their work published. All of them had something published in the open source community and some of them continue to build their portfolios by writing about open source. Based on what they've learned, some of the students are taking computer science electives. The computer science types in my class are pursuing journalism minors. This makes me closer to my dream of creating a "News Application Laboratory," where CS and journalism students collaborate on interactive storytelling. My own "expertise" in CS is deepening such that I've gained the confidence I need to teach a cross-listed course between CS and journalism. Through my course's collaboration with open source, RIT is gaining recognition as a place to recruit and train computer-science-minded journalists. Furthermore, I am concluding some academic papers on the intersection of journalism and open source that I will present in 2012 at a few academic conferences.

WHAT HAPPENS NEXT

- I need to figure out how my students can have access to the people they need to to write their stories.
- I need to have a better grasp of open source publishing and where students can publish work.
- I need to beef up my CS basics. Can you recommend some good intro textbooks for me?
- I would be interested in possibly bringing in a big-name journalist with a CS background or who writes about CS stuff to campus to talk to my students. Perhaps you could give me some suggestions?

GREG HISLOP

Drexel University, Philadelphia PA
INFO 154 Software System Construction
INFO 324 – Team Process and Product

THE CLASS NOW

I'd like to focus on two of our classes that are relatively early in the degree programs. The two courses and descriptions of them are:

INFO 154 - Software System Construction - Introduces considerations that make large software systems challenging to design, build, and maintain. Topics include coding standards and documentation, program architecture, verification, software evolution, and managing large software systems. Includes software modification and development using pair and team programming.

INFO 324 - Team Process and Product - Provides hands-on experience with working in small teams to apply processes and produce products typical of current best practices in computing and information technology organizations. Allows students to develop an integrated understanding of project life cycle phases. Examines issues of team organization and operation, problem solving, and communication.

I see these courses as the most likely candidates to introduce all our students to FOSS. At present, the first course starts to introduce students to software beyond the very small system level. The second course focuses on team-based development. Neither course is specifically FOSS based, but both could clearly be focused around "the open source way".

The courses will be taken by students in Sophomore to Junior year. The students are typically majors in IS or IT. Both courses are fairly hands-on and mix individual and small team activities. Both courses are quarter term courses, so they meet for 10 class weeks plus an exam week.

BY SUMMER 2012

Across this pair of classes, students develop an understanding of FOSS and how to be FOSS contributors.

In INFO 154, they learn about FOSS tools and technique. They gain first experiences with larger code bases and how to navigate them. They also learn about how FOSS projects operate. At the same time, they continue to develop skills in writing requirements, designing parts of systems, and dealing with code.

In INFO 324, students have a chance to apply what they learned about FOSS in small team projects. They work on selected tasks for FOSS projects with the full class and in small teams.

WHAT HAPPENS NEXT

Time is always the scarcest resource. My biggest need is to find the time to implement the ideas, and to continue my own learning and participation in FOSS. I would also need to bring other faculty members up to speed on some aspects of this effort. In general, the notion of students participating in FOSS is well-supported here. But, as is typical, other faculty have not had the time to get themselves up to speed. I would like to build and share teaching materials both for faculty and for use in classes taught by those faculty. This on-going effort is reflected in sites like those of the SoftHum and HumIT projects (see http://xcitegroup.org/projects.html)

STEVEN HUSS-LEDERMAN

Beloit College, Beloit WI

Undergraduate CS Capstone Course (Fall/Spring)

THE CLASS NOW

Beloit College is an undergraduate, residential, liberal arts college with 1275 students. We have students from over 40 states and about 10% are international. We have a strong tradition of doing service work at the local, national and international levels. As such, I think a project in HFOSS might be of particular interest.

I am targeting our capstone course to be integrated into FOSS. We expect 10-12 students for both semesters. The catalog description is:In this developmental course, students learn from one another as well as from the instructor. Students work in teams to enhance an ongoing software project through design, implementation, testing, and documentation; teams regularly present ideas, progress reports, and designs. Programming is done in pairs, pairing a more experienced student with a less experienced student. Students learn current design and programming tools and give presentations on topics of current professional interest, including ethical considerations. Computer Science majors are expected to take this course each semester in which they are in residence during their junior and senior years.

A few CS students also did a research project involving FOSS this year. I had a student who worked on accessibility issues in GNOME and one who is trying to start a new OSS project to allow high schools to do sports team management with a web presence. The students are driving this work and I have not been deeply involved in the details.

BY SUMMER 2012

Looking back, I am struck by several things. The students are now confident in their abilities as contributors to the OSS community. They are justifiably proud of the work they did and how it made a difference. They came to Beloit College to make a difference in their lives and the lives of others and this class was one of the highlights in making that happen. They also laugh when we they think back about how scared they were to send notes to FOSS project members that they now chat with easily. It is great to see them excited about all the skills they gained: new tools, new programming languages, working with a team outside the students they know, understanding project dynamics, and working with many different types of people who run FOSS projects. Best of all, several students are going to work on FOSS projects this summer because it was so great for them.

I'm also loving the change in the culture of our students. They are bonded together and are proud to be in CS. They hang out at night in the lab to accomplish tasks together.. They brag to their friends about what they did. They even put together a kiosk in our hallway so everyone can see the work they did on a FOSS project. I even overhead a first year student saying they were thinking of taking a CS course because they heard what some others had been able to do with their CS skills.

WHAT HAPPENS NEXT

During the workshop I hope to continue my re-entry into Linux. I also want to understand the FOSS techniques so I can help my students be successful in their projects in FOSS. While I have set up a test LAMP server with subversion this year, we need to get this on reasonable hardware and include everything needed for FOSS development (for example, we probably want Git, etc.) The POSSE workshop will help with this along with getting into the FOSS development culture.

Another important aspect of the POSSE experience will be connecting to FOSS people and projects. I suspect it would be best to enter a project where others have worked and possibly have a mentor of someone who did POSSE or extensive FOSS work. This would give me someone to get help from on the educational aspects of using FOSS the first time.

In the longer term I would be thrilled if the students do something that they can to take to a meeting to discuss. This is another potential use of the funds from Red Hat.

RICHARD ILSON

UNC Charlotte, Charlotte NC
ITCS 3155 Software Engineering (Fall)

THE CLASS NOW

The class is the standard undergraduate Software Engineering class. After reviewing various textbooks and curricula for Software Engineering, I realized that such courses normally focus on the less technical aspects of the process --requirements, documentation, design review, testing, maintenance – in the context of large development projects. The first time I taught the class I used the Sommerville text. The class was dry, somewhat abstract, and not engaging to the students. The class size was 20-25, and I had them design, document, and implement a small software project of their choosing in groups of 4-5. The results were not particularly impressive.

The next two times I taught the course I used readings The Mythical Man-Month, and other web articles I found. I had the students write a short summary/critique of each reading; we discussed the topic in class; and I also had them do a group development project. The latest wrinkle is that I am assigned to teach one or two sections, one or both of which have an enrollment of fifty students. Last fall I chose a new textbook by Schach. I felt I could not pull off the readings and class interaction with that many students in the class. Overseeing so many free-form group projects was also something I felt would be too difficult (i.e., too much work) in classes of this size. I also felt that I was neglecting to introduce the students to the Object-Oriented Design techniques (e.g., the UML diagrams) that most textbooks introduced. Using the Schach text, I assign chapters to read from the text, assign (mainly straightforward) questions from the end of the chapter, and also assign about six small, relatively straightforward, group projects, also from the text.

The pre-requisite for this class is only our Introduction to Computer Science II course, which means they'll have had one introductory course in C++, only covering the basic control structures and arrays, and a second course introducing them to Java and some basic object-oriented concepts. Many, but not all, will have taken our Data Structures course, and about half will have taken Design and Analysis of Algorithms. Too many students, whether they've taken the Data Structures and Algorithms courses or not, are weak programmers. This course used to be a requirement for a Computer Science degree. A few years ago it was an optional elective, but often taken because students were required to take several of these optional elective courses. We have since reinstated it as a requirement.

BY SUMMER 2012

What excites me most about the POSSE opportunity is that for several years I've had the idea that the closest we might come to introducing students to a realistic software development experience is to have them work to extend or enhance an existing product. And the numerous available open-source software bases provide an ideal platform in which to work. The students would have to understand an existing design, by reading documentation and code, and provide their own design, implementation, documentation, and testing for their enhancement.

WHAT HAPPENS NEXT

Many of the issues that come up on a real, larger project are quite difficult to simulate in a classroom environment. The projects aren't big enough, don't last long enough, don't have real users, and don't have a maintenance period. Also, most of the groups will have one or two participants that aren't participating, and the students find it hard to find chunks of time for them all to get together. (A lot of students commute, and many have significant outside work obligations.) I'd like for the students to encounter and get a sense of some of the aspects of a real software development project.

Even though I've worked in a variety of development environments and situations, I haven't done such work in the last fifteen years, so my familiarity with the current generation of tools is limited. Having assistance with these aspects would be of great benefit, both to me, and in consequence, to my students. In any event, it would be of great benefit for me to interact with others facing the same issues in the classroom, and learn from those who have tried teaching this subject matter.

MATT JADUD

Allegheny College, Meadville PA
FS 101/102 Freshman Seminar (Fall/Spring)
CMPSC 220 Programming Languages (Fall)

THE CLASS NOW

My second Freshman Seminar, FS102: Technology and Activism, paired with *Art and Activism*, a seminar taught in parallel by Prof. Darren Miller in the Department of Art, introduced forty first-year students to the process of collaborating in open communities. Students conducted interviews, developed logos, and wrote text contributing to the Fedora 13 release during the spring semester of 2010. It was overwhelming and hectic. However, what we loved about the experience was that the course was extremely authentic: students were interacting on IRC and mailing lists, using bug trackers, reading documentation, installing FOSS software... they were fully engaging in as *process* FOSS participants. Based on this experience, my second dive with students was in the context of CMPSC 303: Human Centered Design (HCD). Students first engaged in lightweight usability testing of the Fedora website as part of the Fedora 14 release, and then dove deep in doing usability work on a selection of FOSS projects.

Working with a FOSS community means that I can find natural ways to collaborate with faculty in art, environmental science, biology, political science, economics... really, any department at the college. It simply becomes a matter of finding a community in which we can engage our students, and the process, communication skills, and reflective process of learning take center stage, while technical skills become something to be picked up and discarded as needed. From my point-of-view, this makes FOSS participation a natural fit for me as a member of a computer science department at a small, liberal arts college. In addition, it is likely that two senior capstone students will pursue open projects.

BY SUMMER 2012

My offering of *Programming Languages* was different from any course I've run previously. During the Spring of 2011, I took part in a workshop at SIGCSE regarding lab-based instruction that inspired the design and offering of a "by interpretation" offering of the study of languages in Scheme. In transitioning to patterns for parallel software construction in occam, I brought out the CC-licensed text and FOSS tools that I contribute to. The students were charged first with diving into a new language, and then set a simple challenge: given an interesting component from our lab, develop a new parallel-safe interface to this hardware that we can add to our project's library of code. In addition, they then had to contribute documentation to the recently webified version of the book.

The second success has a much greater impact on the larger FOSS/EDU community. During the spring of 2012, my colleague Darren ran a second FS102. We created a concise set of assignments that introduce first-year students from a broad background to the communication and collaboration involved in FOSS participation. We can share them out as modules that can be reused at other institutions.

During the 2011-2012 academic year, I began establishing a *Community Collaboratory* at Allegheny College; this is a "maker space," if you will -- a place where students, faculty, and members of the community come together to explore art, science, and technology through the act of making. We will follow up during the 2012-2013 academic year with a learning-living community dedicated to exploring the intersection of serving globally by collaborating and making from a fundamentally FOSS perspective. This LLC will draw on the experiences of our FS102 students, and explicitly bring them back as "alumni," so they can serve as "peer guides" as a new generation of students explore into the realm of FOSS.

WHAT HAPPENS NEXT

A critical resource we think we're going to need is support for either (1) bringing outsiders in to our seminar, or (better) (2) taking the students out into the world. A critical part of the Freshman Seminar success will be making the FOSS participation "real" for our students. The most important thing, however, is the community of educators doing this kind of work. Continuing to work with and develop infrastructure for collaborating with educators around the US and (for that matter) around the globe on how we can successfully integrate FOSS into our classrooms is essential.

MICHAEL JONAS

University of New Hampshire, Manchester NH
CIS 790 Capstone – Undergraduate Research Group in Speech (Fall/Spring)

THE CLASS NOW

Course URL: http://foss.unh.edu/mediawiki/index.php/Speech:Home

The University of New Hampshire at Manchester (UNHM) is the urban, commuter college for the University of New Hampshire. Our student body is diverse and many students have significant job responsibilities, some full-time, outside of their classroom commitments. The makeup of the class was split evenly between juniors and seniors from two majors: Computer Information Systems (CIS) and Engineering Technology with a concentration on Computer Technology (ET-CT), that latter being similar to Computer Engineering program at other institutions. CIS majors focus on standard IT topics with courses in database systems, web authoring, networking and operating systems. All ET-CT students matriculate from a community college system and spend their junior and senior years focusing on electrical engineering and computer topics.

Even though students from both majors have had extensive experience in project focused courses built into the curriculum, most of these courses have a specific topic and a specific goal. A research project gave students an opportunity to experience a unique environment where goals were more loosely defined. Similar to many Capstone courses, this course is designed as a self guided project with a faculty advisor overseeing student progress. Students are required to formulate a proposal where they plan their semester work and write a final report detailing the project and results attained. The project focused on applying the CMU Sphinx Speech Recognition Toolkit to generate a set of baseline speech models. Students were tasked with learning about Sphinx through online resources and followed by installing and configuring the latest release on a queue of machines running Linux.

The ultimate goal is to develop infrastructure for doing robust speech research and ease the knowledge transfer between student groups across different semesters. Documentation and developing tools to simplify processes would be crucial outcomes in the success of the project.

BY SUMMER 2012

Students are excited about being a part of UNHM's Capstone class in CIS. Some may have already done some independent study in it during the summer and now get to immerse themselves with other students to do some cool, fun, real world research. The type of research may surprise them because it opens their eyes that research isn't necessarily pure mathematical, highly obscure, but can be very hands on and very engaging.

The first significant contribution they made was to document how parallel model training worked on Sphinx. Students had such a hard time figuring this part out as there was little documentation they could discover online, but once the group figured out how to do it, they spend a good deal of time writing up a concise description on our wiki page and making a contribution to the community in general. From this, students realized that just writing about a process, clarifying and documenting something that had not been well documented before, can make a significant contribution to the overall speech community as a whole, as other researchers could benefit from that knowledge. It was also nice to see that as an IT program, this type of research fit neatly within the IT scope of their degree, yet was cutting edge. It demonstrated specifically to the group of CIS students that they were more than individuals that helped fix someone's printer.

WHAT HAPPENS NEXT

We're working with Sphinx and a collection of other open source tools and so there is a great deal of opportunity to immerse students not only in a research experience but also in the open source community to give them invaluable mentoring opportunities outside of the class room and of the university. This is what I want to gain from these workshops.

ELINOR MADIGAN

Penn State University, Schuylkill Haven, PA
IST-402 Emerging Issues and Technology

THE CLASS NOW

Penn State University, Schuylkill campus is located in rural east-central Pennsylvania. Our enrollment is about 1000 students and is a mix of commuter and residential. We have one course, IST-402 Emerging Issues and Technology, which allows an instructor to have the flexibility of offering courses of interest that may not be a regular offering. In the Fall 2009 term I offered Python as my IST 402 section. I had planned to work through a book with the students, as I knew that most were not strong programmers. I also decided set the course objectives to be high level and not too overly ambitious.

1. To create and implement Python programs with appropriate use of selection, repetition, functions, and data structures.
2. To analyze and explain fundamental programming concepts such as string manipulation, loops and recursion
3. To compare and contrast JAVA and Python
4. To understand how Python is used in embedded systems

Well little did I know that the students would fall in love with this language. On the first night we did a 'tour' of the language, the editor, and wrote 3 programs together. They were simple math programs and ones that the students had seen in other courses. I sent them home with the assignment of writing a program of their choice. I actually did not expect them to come back with much more than some problem that was similar to what we had done. My past experience with students in programming classes is that in the beginning they just do what is assigned. Oh, I was so, so wrong!

To a student they returned the next week having written not only a more complicated math problem (like solving the quadratic equation, a small statistics package and a grocery store change maker), they also included several functions and even started looking at ways to work with simple data structures. The students were thrilled with being able to successfully write a program on their first attempt and really understand what they were writing. As the course progressed their enthusiasm increased and we finished the book in six weeks.

Of course I was now looking for things to do so we started working on some hardware, gaming and database work. The students who took this course have been talking about it for the past year, so I have decided to offer it again and I am expecting about 15 students (sophomores through seniors) to take the course. I plan to change my objectives to be more specific and do more with hardware and I would like to investigate how Python and PHP work together. I am currently teaching a course that uses Blender and it would be great to find a way to write some module in Python for Blender.

BY SUMMER 2012

We had a blast in the Python class! We breezed through the material; the students really took to this language and kept writing extra programs. I found that they were writing things for other classes, even Spanish. PyGames was a hit. We started small with a word game and a simple shooter game. They ended up working together to make me a special birthday game -shoot down chocolate chip cookies! The student told me that they finally got programming - how to attack the problem first and not dive into the code and also how to document it so they could share code. We also wrote programs for the arduinos, starting with simple stop lights and moving up to simple toys. We also were able to write some programs that we gave back to the Open Source community.

WHAT HAPPENS NEXT

In order to actually do this I need to learn more about how Python works with Operating Systems and some project ideas that could be accomplished in one semester. I need to know how we could share with the open source community. I would like to have some really great programmer to come work with the students.

NANNETTE NAPIER

Georgia Gwinnett College, Lawrenceville GA
ITEC 3860 Software Development I (Spring)

THE CLASS NOW

Georgia Gwinnett College is a 4-year public institution founded in 2005. There are 24 full-time IT faculty from diverse academic and professional backgrounds. Many (like myself) have extensive industry experience as well as a Ph.D. GGC also hosted GiveCamp Atlanta in January 2011 and sponsored Tech Talk – a technical enrichment series for the entire GGC community (http://wiki.ggc.edu/mediawiki/index.php/Tech_Talk).

Software Development I is a Junior-level course required for students in our IT-Software Development concentration and an elective for students in our other concentrations. This course is offered every Fall and Spring semester. The number of students taking the course has gradually increased as our college as grown, and we have begun advertising that this course could be a good elective for other IT concentrations. In Spring 2011, we had our largest class to date with 12 students taking the course. http://wiki.ggc.edu/mediawiki/index.php/ITEC3860

Students come to the course after taking 2 semesters of Java programming, so they know the basics of how to develop relatively small programs as individuals when given a specific set of requirements. What they lack, though, is a perspective on the challenges involved when creating and maintaining larger applications for a client in a team environment. I see software engineering as a project-based team activity designed to solve a problem for the client. I try to impress upon them that programming skills alone are not enough to be a successful engineer; therefore, I create course activities that help students grow their technical, presentation, writing, and personal networking skills. I stress ideas like maintainability, reusability, and design. I always try to incorporate interaction with industry.

BY SUMMER 2012

My Software Development I class has been transformed! Students have always liked the real-world project part of the course. But through the collaboration with Red Hat, we were able to work with an external client and have an industry mentor. They were able to see first-hand the value of the concepts that before seemed just “academic”. I continue to get rave reviews on the Tech Talk presented by guests from Red Hat based on POSSE modules. More importantly, students continued to apply the skills from the course after the class ended. Over 50% of the students participated in Give Camp as developers, volunteering their time to enhance or create web sites for community organizations. Students presented their work at the annual GGC Student Research Expo as well as a local undergraduate research conference. Working on this project was instrumental in many of the students obtaining summer internships working at local companies. The professor for Software Development II has mentioned that they are much better prepared to tackle the projects and assignments in that class as a result.

WHAT HAPPENS NEXT

I need to know tools for getting started that don’t require the school to buy or configure anything. I need an idea of what technologies I should learn and introduce to my students before they can truly be productive contributors in an open source project. Most of them just know the Java they’ve been taught in school. Some may have a database background – but it’s not a required pre-requisite. What are the best tutorials or teaching resources that I can point to?

I would like advice on outlets for students to present their work at conferences, how students can develop a portfolio of their work to help them market themselves for internships and future jobs, and for GGC students going through this experience to network with other POSSE students going through a similar experience (maybe at a conference or other open source event). I would also like assistance coordinating a “field trip” or job shadowing experience for students during the academic year.

MIHAELA SABIN

University of New Hampshire, Manchester NH
CIS 505 Advanced Web Authoring (Fall/Spring)

THE CLASS NOW

The course is a required core course in the Computer Information Systems (CIS) program at University of New Hampshire (UNH). The large majority of UNH Manchester students attend school full-time while working, on average, more than 20 hours per week. On average, UNH Manchester students are three or more years older than traditional-age students. The UNHM CIS program curriculum is aligned with the ACM Information Technology Model Curriculum.

I taught Advanced Web Authoring (http://unhmwebapps.wordpress.com) for the first time in Fall 2010. The course enrolled 22 students: two sophomores, eleven juniors, and nine seniors. Students learn to:

- Apply dynamic web programming concepts and techniques
- Create and experiment with web applications and systems
- Review, document, share, test, and deploy web applications
- Use open source collaborative software and content management tools to develop web applications
- Communicate timely and work in teams effectively
- Argue for the use of open source software tools and adoption of open source collaboration practices.

BY SUMMER 2012

Where are my students now? They use version control for any project, whether it's an individual or team project in any of their computing courses. Project documentation, both software development artifacts and project management outcomes (meeting agendas and minutes, and other planning notes) are inevitably chronicled in wikis. Outside class time buzzes with posts on the class mailing list. At night, many of the students show up in the IRC that has been set up for the computing programs. More students get seriously involved in maintaining the computing infrastructure that supports the program's open source projects. They help each other more as explicitly as possible, leaving traces in wikis and mailing lists. The client-oriented projects for local nonprofits have active user and developer communities that are led and energized by the participation of students and alums. Students seek summer research opportunities and use open source for their research and capstone projects. There is an ongoing peer mentoring that focuses on open source.

In a long-term future, students and faculty across academic programs at the University use open source collaboration in their classes to improve teaching and learning, with external partners to solve problems that benefit communities, with fellow researchers, whether undergraduate and graduate students, other faculty, or industry professionals, who are intrigued by interdisciplinary big questions, and with K-12 students and teachers to improve STEM education.

WHAT HAPPENS NEXT

Faculty in my department must know the tools and practices. They need training, coaching, and assistance with learning and using adequate tools, services, environments, frameworks, platforms. They need teaching materials, including lab modules, that scaffold student learning, networking with other faculty to share and critique teaching resources, and support to set up and troubleshoot in-house infrastructure. We need assistance on how support to open source development and solutions created by students and faculty can be commercialized, and advice with determining a business model that sustains the school's curricular commitment to open source and the school's partnerships with community members (such as nonprofits or government agencies) that are the direct beneficiaries of open source projects.

How do I help my colleagues to join a department-wide FOSS effort? How do I help the students who work in the department to provide lab and system admin support to CIS courses? I will also be interested in finding and connecting with local FOSS developers and have them join some of the classes.

KRISTINA STRIEGNITZ

Union College, Schenectady NY
CSC 206 Natural Language Processing (Spring)

THE CLASS NOW

This is a lower level elective; it will have 10-20 participants, all undergraduates. The only pre-requisite is a CS intro course (CS1). It will be taken by students who have really taken nothing but the intro course and would like to take some more CS without having to go into data structures straight away as well as by more advanced CS students who are simply interested in the topic. My goal is to give the students an understanding of what NLP is, why it is hard, and some fundamental algorithms used in NLP.Software: I am using a software toolkit called NLTK (natural language toolkit; http://www.nltk.org) in this class. It is the most widely used software package in NLP education and it is open source. The package is a wonderful teaching aid and flexible enough to let the students explore quite sophisticated research questions, but it is not equally well developed in all areas of NLP. Specifically, it is lacking in natural language generation and dialog systems, which are my research areas.

In this class, I would like to teach the students about NLP and introduce them to NLTK using the areas that are well developed in NLTK. I would like to then introduce them to state-of-the-art models of natural language generation and dialog systems and have them contribute implementations of these models to NLTK and teaching material to go with the implementations.

Challenges:

1. As I said above, the only pre-requisite for this course is a CS intro course. So, not all students will be able to contribute code. I will have to make sure that no students feel left out and that all feel that they were able to contribute meaningfully, no matter what their CS background is. Right now my idea is that the students with more programming experience may focus more on the implementation, while the others focus more on the accompanying documentation and teaching material.
2. This is not a software engineering course. If the students learn some open source development practices along the way, that is great, but it is not the focus of the course. I will have to make sure that the main objectives of the course, i.e. learning about natural language processing, are reached.

BY SUMMER 2012

I have taught the NLP course, and all students have an understanding of the kinds of questions that are asked in NLP research and the kinds of things that make NLP hard. They also have some hands on experience with approaches to answering these questions and tackling the issues that make NLP hard because they have all worked with the NLTK toolkit. They know that NLTK is a FOSS project and they have an understanding of what that means. All students have contributed to the project in some way. Some students have contributed code addressing natural language generation and dialog systems. A particularly enthusiastic group of students, will continue to work on this as an independent study during the fall term. In the meantime, I am promoting their contributions to colleagues who have been looking for good teaching material for natural language generation and dialog systems. They promise to use the material in their next NLP courses and to give us feedback.

WHAT HAPPENS NEXT

I have to do some more thinking on the questions that I have listed above. Discussions with other educators and people who experience contributing to FOSS as part of a class is also going to be helpful. I also have to have a closer look at nltk.org. I have used it for teaching, but I have not looked at it from the point of view of a contributor. I will have to figure out what the processes are that they use, and what their policies are for including new contributions.

ALLEN WHITE

Bacone College, Muskogee OK
Problems in Information Systems (Fall/Spring)

THE CLASS NOW

Before I describe the course, let me describe the environment. Bacone College is the oldest continuous center of higher education in the state of Oklahoma. We were founded in 1880 with the mission to provide education services to American Indians. The majority of our students are first generation college students. Most of these students, particularly those in the CIS program, view their college career as a real opportunity. Some may not be as well prepared academically as might be desired, but no one should question their desire. They will and do work and work hard to make the most of their opportunities. Our institution is actively moving to support cultural preservation projects with various tribes. I would like to find a way to make this project fit into the overall direction of the college at this time.

I have selected our capstone class, Problems in Information Systems. It is taken during their last year in our program. The course focuses on three primary issues: the development of an understanding of the individuals place and responsibility to their discipline, a demonstration that the student has developed an understanding of Information Technology such that they are prepared to contribute, and a demonstration that the student can complete a project that is deemed appropriate and useful to its user community. Among the projects that these classes have completed in the past include development and demonstration of a issue tracking system for the college’s housing and maintenance teams and reconfiguring and deployment of a server deemed obsolete by the institutions IT staff to serve as a file server for students in CIS courses.

Bacone College currently uses a significant amount of Open Source Software in its operation. I was the prime mover in the implementation of Moodle as our Course Management System. Our issue tracking system, which grew from the work of a Capstone section, uses osTicket. The college’s website is implemented using Joomla. Every computer on campus has an installation of OpenOffice on it. Recently, I have added a Mahara implementation to support portfolios for my courses and for my students. Similarly, I attempt to use Open Source software in my courses where appropriate. My background is a bit unusual. I spent many years working in Oil and Gas Exploration and Production Research at three of the larger energy companies in the United States. I have worked as a development programmer, project leader and development director. Now I have returned to academia and have been a member of the faculty at Bacone for nine years. When I came to this institution, I was, and remain, the only faculty member dedicated to Computing.

BY SUMMER 2012

The biggest challenge in teaching my students is often confidence. Many of them come from a background where they have not felt real success academically. They may have succeeded on an athletic field but not felt that ‘aha moment’ that drives creativity in problem solving. This course allowed these students to feel that moment. They took a real problem; not one from a book. They found a solution to that problem and they made it available to others. During the capstone course, the professor starts to become more of a project advisor; one who provides encouragement, guidance and suggestion but who allows the student to become the colleague. There is a real sense of peace that comes when, as an instructor, you realize that your students are capable.

WHAT HAPPENS NEXT

I have a background as a working software developer and have led development teams of significant size. But I can provide only one person's perspective. In an institution of our size, students get to know one CIS instructor very well. In larger schools, there are more faculty, so more perspectives are available. Access to a working professional who is willing to mentor my students would provide significant help in resolving that shortcoming of our program. As we are a small institution with very limited support resources, access to materials is always appreciated. I have worked diligently to amass various teaching materials over the years but, frankly, access to Open Source material is often lacking. As an individual basically working with little infrastructure support, access to turnkey systems sounds wonderful.

KARL WURST

Worcester State University, Worcester MA
CS 193 Seminar: It's An Open Source World! (Fall)
CS 401 Software Development (Spring)

THE CLASS NOW

Worcester State University is a medium-sized (~5000 student), urban university. Many of our students are first-generation college students, most are commuters, and most work while attending school. We have a high teaching load - 4 courses per semester, and a small department - only 4 faculty in the CS Department.

CS 193 First Year Seminar: It's an Open Source World! has never been taught before. It is a First Year Seminar for 20 incoming freshmen of all majors to introduce students to college-level learning. I intend for this course to fit the Worcester State University Fall 2011 Theme Semester theme: “Worcester in the World.”

CS 401 Software Development is a capstone Experience for up to 24 junior/senior CS Majors. When this course was last taught in Fall 2010 the class consisted of 5 groups of 4 students each. Each group looked for their own open source project and tried to identify a bug or feature to implement. I met with each group individually, to get a sense of where they were, what was currently blocking them, and what they needed to do next. These meetings were probably the most useful portion of the class, both for me and for the students. The groups did not document their status on the wiki very well. Many times, the only documentation of their progress was my notes from my weekly meeting with them. The groups all made some accomplishments, but how much varied significantly from group to group, likely dependent on the project they had chosen. Not all groups contributed their accomplishment back to the community.

BY SUMMER 2012

We spent an exciting Fall semester exploring the open source movement/phenomenon. We read about and discussed people coming together to create and organize in ways that were not possible before the existence of the Internet, creating, sharing, and remixing content. We read about and discussed the open source movement, its motivations, policy issues, manifestos, licensing issues, and the idea of the commons. We discussed the positives and negatives of the open source movement. The students participated in an open source project - some wrote documentation, some worked on marketing, and some students with programming experience from high school even wrote some code. We had a few guest speakers, some from the open source community, and others from our own Sociology Department (discussing social networks) and our own Business Department (discussion organizational dynamics.) The students and I have gotten to know each other, they have found that they are capable of doing more than they thought they could, and some of them are going to continue to contribute to the community.

We spent a productive Spring semester learning the tools and processes of the open source community. The students got to know members of the project we worked on, and made some significant contributions. The students now have a feel for working in a large codebase, with a distributed and diverse team, and what benefits and problems those can bring. They are competent in using version control, bug tracking, debugging, and documentation tools. Many of the students plan to continue to contribute to the open source community, perhaps in other projects that are more to their own personal tastes. We have set up a blog planet so that the students can showcase their work, and show off the department. Some of the students have decided that this is something that students should get involved with earlier, and are starting an open source club, so that they can help underclassmen get started.

WHAT HAPPENS NEXT

We need lists of readings about the open source movement, speakers from the open source community, and suggestions for contacts in projects. Suggestions for tools to teach them. Tutorials on using tools like version control, bug trackers, wikis, debuggers, generating patches, IRC. Speakers from open source community. Maybe some of our alumni who have contributed to open source. Ideas on deliverables, grading and assessment. How to assign students to groups. Blog planet; I think I would rather install that on a server on our campus.

REMOTE INTRODUCTIONS

Online at http://teachingopensource.org/index.php/IRC_and_wiki_introduction_exercise

BACKGROUND

IRC is a text chat program. It is used by the vast majority of open source projects as their forum for real-time communication. This is where meetings are held, but also a social place where people idle and hang out while they are working, similar to the casual banter that goes on in an office or a lab or cafeteria. One of the primary differences between IRC and messaging programs most people new to open source are familiar with is that IRC conversation is in *group chatrooms by default*, whereas AIM, ICQ, Jabber, etc. tend to do *individual conversations by default.*

A wiki is a publicly editable webpage. You may be familiar with wiki-based sites such as http://wikipedia.org. What most people don't know is that wikis are extremely simple to set up, so individuals, open source projects, and other sorts of groups use them for notes, project webpages, and general collaboration/documentation infrastructure.

A wiki userpage is like a profile page for an individual user of a wiki. These can be extremely elaborate (http://wiki.laptop.org/go/User:Mchua) or highly minimalist (http://teachingopensource.org/index.php/User:Ctzenben) ; most are somewhere in between (https://fedoraproject.org/wiki/User:Sdz) (if you have more time, explore some others here). People decide what they want to show and share on their userpages; it typically acts as a "homepage" for them for their work and introducing themselves to potential collaborators who may not yet know them.

INSTRUCTIONS

Your goal for this exercise is to have a wiki user page of your own. Use the wiki specified by your instructors. Your page must be completed by the end of class, and it can say anything (tasteful) you like; think of this as a professional homepage for your open source activities. There are two additional constraints that you are under for this. The first constraint is that you may not edit your own wiki user page. This implies that someone else needs to edit it for you. That person can be anybody, in the classroom or remote.

The second constraint is that you are only allowed to communicate electronically during this exercise. You may not speak in person, write on pieces of paper, show somebody else your screen, use hand gestures, etc. Essentially, pretend that you are in a room by yourself with your computer, and communicate and collaborate with your partner that way. This constraint extends to finding your partner - you need to find a way to match up with someone else for this (we suggest IRC) and coordinate between yourselves online. The only exception to the second constraint is that you may communicate with your instructors in-person, particularly to ask about tool usage and communication strategies. Please do not use this exception to secretly communicate with your partner. The point of this exercise is to give you an experience in distributed collaboration; we are particularly interested in what sorts of coping strategies you find, evolve, and negotiate on your own while working on this. We expect this exercise to be confusing and occasionally frustrating, so your instructors will be walking around throughout the exercise to help you turn that confusion into insight, because exploring new ways of doing things should be slightly frightening, but ultimately a fantastic adventure where you'll end up being surprised at how much you learn.

DELIVERABLES

- An IRC log of the conversation
- Individual wiki user pages

EXAMPLES OF USAGE

For an IRC log of attendees of an actual POSSE completing this exercise, see POSSE Doha transcript (http://meetbot.fedoraproject.org/teachingopensource /2011-01-10/teachingopensource.2011-01-10-07.10.log.html)

ROADMAP MERGE

Based on http://mchua.fedorapeople.org/talks/2011-diving-in/roadmap-merge-overview.pdf

INTRODUCTION

You've already got too much to do. There are exams to deliver, homeworks to grade, standards you need to hit, multiple projects to manage. We get that. The point of this workshop isn't to add to the work you already have; it's to explore how to leverage that work by doing it within an open source community – or leverage an open source community to do your work; either way is great.

In order to do that, we've got to understand the calendar you already have for your semester and how they map into the timelines and rhythms of the open source community you've chosen. Maybe you'd like to use a feature coming in the next release; when will that be? Maybe your project could use a Test Day; when are those typically scheduled, and how far in advance are they planned? Maybe you're curious how the Infrastructure team prepares for a massive server-hitting launch or how the Translation team coordinates sprints before string freeze; where do you look to find out the dates you should be listening in on the conversation?

We'll show you a quick overview of what an open source project's release cycle looks like, then set you loose to hunt down the major milestones of your chosen project's current cycle so you can find out what's happening *right now*. Once you've got those dates, you'll draw a roadmap to lay out your school's timelines right next to your chosen project's time-line, then see if there are any natural opportunities for connecting or piggybacking in either direction.

LEARNING OBJECTIVES

At the end of this unit, you should be able to:

- Find and list the major milestones of an open source project's current release cycle.
- Draw a timeline of how the work you're already doing fits into a project's release and feature schedule.

BACKGROUND

TIME-BASED VS FEATURE-BASED

Fedora uses a time-based release cycle, which means that feature planning is done by picking a deadline, then asking the question “which features can be finished by this deadline?” Features may be dropped in order to complete the release by the deadline. In contrast, feature-based release cycles (for example, Debian) pick features they wish to see completed in a given release, then ask the question “when can these features be completed?” Deadlines may be extended in order to complete all the features on the list. (This is by no means a binary; time-based cycles may “slip” to extend deadlines, and feature-based cycles may cut a feature in midcycle if it's apparent that it lags far behind the others on the list.)

Both types of release cycles are common in the open source world. Fedora's release cycle is 6 months long, which is an average release cycle length shared by major upstreams such as the GNOME Desktop. Other time-based projects may have release cycles as short as a few weeks or as long as a year or more.

ROADMAP MERGE

RELEASE CYCLES AS A MULTI-ACT PLAY

One useful analogy for a release cycle is that of a multi-act play. For instance, in the Fedora world, Act I is feature selection, Act II the Alpha, Act III the Beta, and Act IV the final release. Each act has multiple scenes, sometimes playing at the same time in different parts of the stage. Each scene is staged by certain players, and each scene may be scripted, improvised, or a combination of the two. For instance, the Act I scene “Deciding on a Feature Set” might feature several Engineers, a Marketer or two, and an Infrastructure person who's onstage but mostly a silent watcher of the largely improvised dialogue. Act IV's “Getting The Mirror Up” would involve multiple Infrastructure players in a complex, heavily prescripted routine.

FINDING THE ACTS AND SCENES

You'll first need to find your project's release cycle, or piece it together from multiple resources. Different projects call their schedules by different names. Here are some to try:

- release cycle
- release schedule
- calendar
- roadmap
- deadlines
- feature process

If your project doesn't have a roadmap/schedule, call your instructor over. Think about why that might be the case – perhaps it's a new project, or a small one, or there is a schedule but it's undocumented – how could you find out?

FINDING THE PLAYERS

Every project groups its people into different sorts of teams. Here are some of the more common ones.

- Infrastructure (sometimes called Sysadmin)
- Websites (sometimes called Webmaster; may be part of Infrastructure or Marketing, or a separate team of its own)
- Marketing (sometimes called PR, occasionally merged with Ambassadors and/or News)
- Ambassadors (sometimes called Events or Outreach, occasionally merged with Marketing; may be further split geographically by country or region, or into individual local clubs/chapters.)

ROADMAP MERGE

- News (not all projects have this; sometimes merged with Marketing)
- Design (sometimes called Art)
- Translation/Internationalization/Localization (Also known as i18n, t9n, and l10n - count the number of characters between the first and last letters of the word, and consider the difference between British English and American English spelling, and the subsequent debates of how, exactly, to name these teams. These may be one team, or split into several teams.)
- QA (sometimes called Testing, Bug Squad, or Bugzappers)
- Documentation (sometimes called Docs)
- Packaging and distribution (may not exist for all projects; primarily relevant for distribution projects)
- Developers (sometimes called Coders; may not exist for all projects, particularly distributions)

Don't forget to look for SIGs (Special Interest Groups) – there may be regional or discipline-specific groups you're interested in. For instance, would you imagine the following to exist in Fedora?

- Multiple desktop groups – not just GNOME, but also KDE, XFCE, and others
- Musicians' SIG
- Education SIG (and OLPC/Sugar SIG)
- Finance SIG
- Campus Ambassadors (university-level)
- Summer Coding (Google Summer of Code and similar mentoring projects)
- Ham Radio
- Robotics

They all do. See if you can find them – and other groups like them.

CONSTRUCTING YOUR ROADMAP

You'll be making your roadmaps on a large sheet of paper. A roadmap consists of a double timeline (the top is for the timeline of your current work, the bottom is for your chosen open source community) followed by a "mergepoint" section. A mergepoint is an opportunity you spot for one event happening at one time and one (online or real-world) location to "double-count" and score an Epic Win for both your work and open source projects.

Here's how we suggest building your roadmap.

- Write in the months on the timeline, so we can see what dates we're looking at.

ROADMAP MERGE

- Plot in the major milestones, deadlines, and time-periods for your work project(s) on the top half of the timeline. Mark major holidays/vacations as well.
- Now look at the “resources” page you filled out previously. What milestones there are relevant to your work project? Draw them in on the bottom half of the timeline.
- Now look at individual teams within your project. Are they doing activities you are interested in? Draw in when they happen on the bottom half of the timeline. For instance, “infrastructure freeze” may not be part of the feature process on the release schedule, but if you're thinking about asking the project's sysadmins for advice on trying out a new tool, the frantic week before the freeze may not be the best time to ask.
- Look for a mergepoint. (If you find more than one, that's great!) Remember, a mergepoint is anything that could double-count for both your work and your open source project. For example, do you need to write a technical specification a week before feature proposals are due? Why not submit the same text for both? Is there an upcoming conference or hackathon? Maybe community members for your project would be interested in hearing students present their work. If you're stuck, ask your instructor for help.

If you finish early, see if you can do the following:

- Describe at least 3 teams/contributor-roles within an open source project that your work will affect and/or be affected by, and where each team fits into the RACI model for your project.
- Adjust your work timeline in response to schedule slips in an upstream open source project. What sorts of dates and deadlines do your mergepoints depend on? What happens if they happen a week early? A week late? Two weeks late? How do you need to prepare and react (if at all) – and how can you make sure you'll know if schedule changes happen? Alternatively, is there a way you can modify your work to remove its dependency on that deadline, and what sorts of tradeoffs does that entail?
- Chase down more info on your “what needs to happen” -sections wave down someone in the classroom (an instructor or a fellow learner) and ask them to help you drill down on more details; pop on IRC and see if you can find a community member to answer your questions, search online and see if you can find documentation that answers your questions.
- If you're really feeling bold, start taking action on the “what needs to happen” list for your mergepoints.

NINJAHOOD PLAN

OBJECTIVE

We talk a lot about making our students self-directed, reflective learners who have “learned how to learn.” It's time to walk the walk and do the same ourselves for open source community participation. At this stage, we're exploring options, not making commitments – you and your coach will be in touch throughout the year and can readjust as needed. You'll end up with the start of a coaching plan for the school year.

SELF-EVALUATION ON CULTURAL UNDERSTANDING

This is a tough one – how do you and your students become reflective about something that's inherently fuzzy? In this case, thinking of open source communities in similar ways to a language immersion or study abroad experience might help. Here are some other questions that might be interesting to ask.

- Question-timing: do you ask questions too soon, or wait too long to ask for help?
- Tact filter: are you “too nice” or overly worried that you'll waste someone else's time? How do you treat newcomers – do they get a “RTFM” reception? What assumptions do you make about who they are and what they know? What assumptions might others make about who you are and what you know?
- Asynchronous/multitasking: Are you too asynchronous – hard to get hold of, doing so many things at once it's difficult to tell (remotely) where your attention lies or when you'll get back? What's your typical latency for various types of communication? What are your expectations for the latency of other people getting back to your communications, and what sets those expectations for you? How do you react when your expectations are not met? How do you communicate your status to the people you are working with?
- Bloodhound: Can you follow conversational threads that hop media (IRC to wiki to microblogging back to 3 different IRC channels) and location?
- Breadcrumbs: Do you leave a trail of thoughts and work that is easy for other people to trace?

COACHING

As a POSSE cohort member, you'll get the following for every month you're teaching a TOS class this school year:

- Two, 1-hour coaching sessions (on IRC, phone, videochat, in-person, or a format you and your coach agree on based on schedule, needs, and location). You may pool or share your coaching sessions with other cohort members, colleagues, students, and/or your class.
- Necessary phone/IRC/email support between sessions, including backup support on public lists.
- POSSE modules. Note that you can ask for special exceptions if there's a module you need taught on or by a specific date, or a topic that needs to be covered that isn't in a module already.
- Up to two check-ins with a community mentor from your project to incorporate feedback, based on availability.