Restructuring 207 (placeholder title)

Spencer Liberto

Grinnell College  
Grinnell, Iowa, 50112, U.S.A  
+1 641 888 0384

libertos@grinnell.edu

Lea Marolt Sonnenschein

Grinnell College  
Grinnell, Iowa, 50112, U.S.A  
+1 641 275 0574

maroltso@grinnell.edu

Daniel Torres

Grinnell College  
Grinnell, Iowa, 50112, U.S.A  
+1 310 592 9565

torresda@grinnell.edu

**ABSTRACT**

In Grinnell College students get introduced to CS through a structured three course multi-paradigm sequence. Seeing the success CS1 and CS2, that use media computation and robots, respectively, to teach functional and imperative programming, have had with enrollment and student satisfaction at the end of the course, it was due time to incorporate a more motivating technology, and not just the Eclipse IDE, into CS3, that teaches object-oriented programming in Java. Recent studies [2] have shown that students are more likely to enroll in courses pertaining to socially relevant issues, which is why we have decided to restructure the course. Out hope is that with the introduction of a relatively fresh technology – mobile application development on Android, and an overarching theme of computing for social good – with the help of the Ushahidi platform, students will learn how to solve complex and relevant problems with computing with an exciting technology that will ultimately enhance their learning experience and convince them to pursue computing more in depth.

**Categories and Subject Descriptors**

K.3.2 [**Computer and Education**]: Computer and Information Science Education Features – *Computer science education, Curriculum*

**General Terms**

Design, Experimentation, Languages

**Keywords**

Computer science education. CS3. Android. Ushahidi platform. Object-oriented programming. Java. Course materials. Lab-based course. Interdisciplinary approaches.

# INTRODUCTION

This paper describes a reconstruction of the last course in a three course multi-paradigm CS introductory sequence taught at Grinnell College. Our goal is to incorporate tangible software development techniques, under an overarching theme of computing for social good, in a course that introduces students to object-oriented programming.

WRITE MORE STUFF

# CONTEXT

WRITE STUFF!

## Motivating Technologies

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

*Conference’10*, Month 1–2, 2010, City, State, Country.

Copyright 2010 ACM 1-58113-000-0/00/0010 …$15.00.

WRITE STUFF!

## Multiparadigm Approach

WRITE STUFF!

## Computing for Social Good

Unfortunately, incoming students who have no previous computer science experience are often under the misconception about what a computer science course or major actually entails. Studies have shown that freshmen tend to avoid taking computer science classes because they think they’re boring, tedious, and irrelevant [3, 12]. Not only that, but students, especially females, tend to choose majors that are easily relatable to social good [2], which is something they don’t associate with computer science. This is why educators have recently tried to incorporate the theme of computing for social good in introductory computer science classes. As such they can provide students with meaningful and tangible projects that have an impact on society, and give them an incentive to continue their computer science education.

## Introducing Version Control Systems

Even though version control is essential to software development, very few CS courses actively emphasize and require their use [4]. Another misconception about CS, is that it’s a very solitary field. By introducing students to version control systems early on, we hope to both teaches them good software development practice, and show them how many CS disciplines require a collaborative team effort. By doing so we hope to disprove the aforementioned misconception and promote the idea of teamwork.

## Redesigning CS3

WRITE STUFF!

# MOTIVATING TECHNOLGOIES

In the early stages of our project we decided to incorporate a new technology and theme to our CS3 course, in order to provide students with a tangible technology for which they can program, and a theme that motivates them, and gives them the opportunity to impact relevant issues. This would provide a natural transition from CS1 and CS2, whose workshop style teaching has proven to be very successful. In May 2013 we sent out surveys to students and faculty, to give them a chance to be a part of our decision to restructure the course. More than 80% of the survey takers were most interested in incorporating mobile development with an overarching theme of computing for social good. In order to teach them how “real-world” software development works, we decided to give them a fairly open-ended collaborative final project where they work closely with a client, and apply what they’ve learned about using a version control system throughout the course.

## Android

In the past few years, the revolution of mobile computing has made software development much more accessible. It is our hope that students, having used mobile phones for most of their life, will find programming for the platform stimulating and exciting [10]. Because Android is natively based in Java, and as a mobile framework, very object based, it was a clear choice. The GUI design gives students tangible objects they can work and play around with, which reinforces the aspect of object-oriented programming, the ultimate pedagogy goal of this course.

## Ushahidi Platform

The Ushahidi platform is an open-source software that was released in 2007. Its goal is to collect relevant information about a certain topic through crowdsourcing to visualize the information on an interactive map. It was first used in the Kenya’s 2007 presidential election to collect reports of violence and voter fraud via email and text message, to place them on a map, thus informing and educating the public of corruption that was happening around them. Even though it can be used to gather any kind of information, Ushahidi deployments tend to address pressing social issues. Each Ushahidi deployment provides a large data set of reports. Each report has many different attributes, so it essentially alludes to an object. Because data structures and efficient data processing is a very large part of a computer science education, Ushahidi provides a great collection of motivating data that students can manipulate.[[1]](#footnote-1)

## Final Project

Another thing our surveys showed was that professors and students have a desire to work on a project with a client earlier on in the CS curriculum. Working with the department of Service Learning and Engagement, we were able to find a number of interesting projects that students could be able to initiate on campus (wellness/self-care and accessibility), in the town (the Chamber of Commerce, Drake Public Library, the United Way…) and with faculty (Sociology, Prairie Studies, Leisure…). The final project is going to include:

* Students working in a team of 3 or 4 people
* Decide what issues they’re going to tackle and find a client
* Create an Ushahidi deployment pertaining to that issue
* Work with their client to understand what best suits their needs
* Educate the client on how to maintain their deployment

A few example final projects might include reporting on:

* Racial incidents
* Transportation in Grinnell
* Failures of “Self Gov” 1
* Client mapping

## Git

We chose git as our version control and source code management system because of its flexibility, relative ease of you, and speed. We wanted to use an SVN that has all the functionality that we need, yet doesn’t add too much overhead to the already packed course. There are many pedagogical and student benefits to using git [5]. Educators can keep better track of student work via git than via email, they can easily identify potentially bad work habits, and better collaborate with their students. At the same time, students learn to work better in teams [9], they are more motivated to submit assignments on time, and they ultimately build essential skills for their future careers.

# COURSE PLANNING

## Course Goals

Before developing a curriculum using the selected motivating technologies and themes, we compiled a list of goals for the course:

* Introduce object oriented programming.
* Introduce algorithms which maintain and manipulate data structures.
* Involve students in a computing for social good.
* How to set up a basic Ushahidi deployment, and how to collect usable data from it.
* How to design and develop a basic Android application.

## Semester Outline

|  |  |  |
| --- | --- | --- |
| Topic | Course Hours | Topics |
| Developing in Java | 4 | Using an IDE; Git verison-control; Javadoc, Unit Testing, Debugging; Java Syntax |
| Object-Oriented Programming | 4 | Classes vs Objects; OOD principles; OOP in Java; Basic I/O |
| Android Development | 4 | Android development; Making a simple app; |
| Arrays | 4 | Arrays; Loops and iteration; Catching exceptions; Linear vs Binary search; Big-oh analysis |
| Linked Lists | 8 | Interfaces, abstract classes, generics; linked lists; Collections API |
| Linear Structures | 8 | Stacks, queues, priority queues; Sorting algorithms |
| Trees | 4 | Trees; Binary search trees; tree traversal; |
| Heaps | 4 | Heaps; Binary heaps; Heapsort |
| Other Data Structures | 4 | Hash tables; Sets; Dictionary |
| Final Client Project | 4 | In class final project development time. |

# PROJECT DEVELOPMENT

PICTURES?!?!?!

## Technology Experimentation

In planning the course, we have selected new technologies to introduce: Git, Ushahidi, and Android. After some initial experimentation, we found ways in which to incorporate these technologies into the curriculum to their full potential, to enhance the CS3 experience, and to not distract students from the topics of object-oriented design and data structure algorithms.

## Introducing Git

Because we wanted students to familiarize themselves with git, and develop a habit of using it regularly, we decided to introduce it in the first week of the course. We wrote several comprehensive readings and accompanying laboratory exercises that thoroughly explain:

* What version control systems are
* How to use GitHub
* Git specifics through examples (via terminal and Egit)

Students will be responsible to fork weekly projects posted on the class’s GitHub account, and submit them by sending a pull request. The instructor will then look at their code, and evaluate it accordingly. When the students are working in teams, they will have to create a joint public repository.

## Establishing an Ushahidi Server

Our vision for involving Ushahidi in CS3 is to have students pull information from a server running the Ushahidi platform, and to teach students how to set up their own deployment. We wanted to establish our own server as a reliable source of data that is relevant to our institution, and as an exemplar of an Ushahidi deployment.

First, we established an Ushahidi server within a virtual machine, using Oracle’s VirtualBox. VirtualBox was run on an 2011 Apple iMac. The virtual machine runs the Xubuntu Linux distribution. We documented the instructions on how to set-up a virtual Ushahidi server.

Afterwards, we set up a dedicated machine to run a server for us. We install the Xubuntu operating system onto the machine, and set up another server. The server was established on an HP Compaq 6200. This server will be used as an exemplar Ushahidi deployment, and as a source of motivating data. It will also be used throughout the course to organize data.

An alternative method for establishing and Ushahidi deployment is to host one on a website called CrowdMap. CrowdMap was developed and maintained by the same team who develops Ushahidi. CrowdMap allows a user to set up a basic deployment of Ushahidi, running on CrowdMap’s servers. CrowdMap has abstracted Ushahidi servers to be simple and easily maintained. However, through investigation, we found that CrowdMap did not support many features that our team found vital to the course we were designing. Ultimately, we chose not to use CrowdMap in our redesign of CS3.

## Developing an Ushahidi Java API

WRITE STUFF!

## Designing Android Course Material

An important challenge we faced was, how to incorporate a fairly complicated realm of mobile application development for Android, without adding too much overhead to an already dense course. After some failed experimentation with MIT’s AppInventor, and writing wrappers, we decided it was best to provide the students with readings and accompanying laboratory exercises in a weeklong introduction to:

* The Android SDK
* Creating an Android Project from scratch and what each generated file is used for
* A holistic overview of using XML for the GUI

We immerse them in Android development for a week, where they explore and create apps with our guidance. By the end of the week students should be familiar with basic Android app development practices, and know how to manipulate both the Java code and the XML code to achieve simple functionality. At the end of the week, we present them with a larger-scale skeleton app, which sets up the structure for their projects in the following weeks.

They start out with a basic “Annoying Kitty” app, where they put an image of a cat on the screen, and a button, that when pressed, produces a meowing noise. The following projects deal with data manipulation like sorting and searching with different data structures to verify their efficiency, and figure out what types of data processing they’re best used for.

## Final Client Project

WRITE STUFF!

## Reorganizing Course Material

WRITE STUFF!

# INTERACTING WITH GITHUB

WRITE STUFF!

# CONCLUSIONS AND NEXT STEPS

WRITE STUFF!

# ACKNOWLEDGEMENTS

We would like to thank Dr. John Stone, and Jordan Yuan, who provided technical assistance with installing the Android SDK on our MathLAN computing network. We would like to thank the CS faculty at Grinnell College, who provided meaningful pedagogic insights; especially our mentor Sam Rebelsky, whose enthusiasm and guidance inspired us throughout the summer. We would like to thank Susan Sanning for giving us ideas and resources for student projects. Finally we would like to thank Grinnell College for providing funding over the summer, without which, none of this would have been possible.

# REFERENCES

1. David Cowden, April O'Neill, Erik Opavsky, Dilan Ustek, and Henry M. Walker. 2012. A C-based introductory course using robots. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (SIGCSE '12). ACM, New York, NY, USA, 27-32. DOI=10.1145/2157136.2157150 <http://doi.acm.org/10.1145/2157136.2157150>
2. Michael Goldweber, John Barr, Tony Clear, Renzo Davoli, Samuel Mann, Elizabeth Patitsas, and Scott Portnoff. 2013. A framework for enhancing the social good in computing education: a values approach. ACM Inroads 4, 1 (March 2013), 58-79. DOI=10.1145/2432596.2432616 <http://doi.acm.org/10.1145/2432596.2432616>
3. Michael Goldweber, John Barr and Elizabeth Patitsas 2013. Computing for the Social Good. THIS IS DEFINITELY NOT CITED CORRECTLY! http://db.grinnell.edu/sigcse/sigcse2013/Program/viewAcceptedProposal.pdf?sessionType=specialSession&sessionNumber=19
4. Mark Guzdial. 2009. Education: Teaching computing to everyone. Commun. ACM 52, 5 (May 2009), 31-33. DOI=10.1145/1506409.1506420 <http://doi.acm.org/10.1145/1506409.1506420>
5. Joseph Lawrance, Seikyung Jung, and Charles Wiseman. 2013. Git on the cloud in the classroom. In *Proceeding of the 44th ACM technical symposium on Computer science education* (SIGCSE '13). ACM, New York, NY, USA, 639-644. DOI=10.1145/2445196.2445386 <http://doi.acm.org/10.1145/2445196.2445386>
6. John Lewis, Anthony Allevato, and Stephen Edwards. 2012. Motivating CS1/2 students with the android platform (abstract only). In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 682-682. DOI=10.1145/2157136.2157439 http://doi.acm.org/10.1145/2157136.2157439
7. Ananya Misra, Douglas Blank, and Deepak Kumar. 2009. A music context for teaching introductory computing. In Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITiCSE '09). ACM, New York, NY, USA, 248-252. DOI=10.1145/1562877.1562955 http://doi.acm.org/10.1145/1562877.1562955
8. Samuel A. Rebelsky, Janet Davis, and Jerod Weinman. 2013. Building knowledge and confidence with mediascripting: a successful interdisciplinary approach to CS1. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). ACM, New York, NY, USA, 483-488. DOI=10.1145/2445196.2445342 <http://doi.acm.org/10.1145/2445196.2445342>
9. Karen L. Reid and Gregory V. Wilson. 2005. Learning by doing: introducing version control as a way to manage student assignments. In *Proceedings of the 36th SIGCSE technical symposium on Computer science education* (SIGCSE '05). ACM, New York, NY, USA, 272-276. DOI=10.1145/1047344.1047441 http://doi.acm.org/10.1145/1047344.1047441
10. Derek Riley. 2012. Using mobile phone programming to teach Java and advanced programming to computer scientists. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 541-546. DOI=10.1145/2157136.2157292 http://doi.acm.org/10.1145/2157136.2157292
11. Jay Summet, Deepak Kumar, Keith O'Hara, Daniel Walker, Lijun Ni, Doug Blank, and Tucker Balch. 2009. Personalizing CS1 with robots. In Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09). ACM, New York, NY, USA, 433-437. DOI=10.1145/1508865.1509018 <http://doi.acm.org/10.1145/1508865.1509018>
12. Sarita Yardi and Amy Bruckman. 2007. What is computing?: bridging the gap between teenagers' perceptions and graduate students' experiences. In Proceedings of the third international workshop on Computing education research (ICER '07). ACM, New York, NY, USA, 39-50. DOI=10.1145/1288580.1288586 http://doi.acm.org/10.1145/1288580.1288586
13. Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender, Ed. ACM Press Frontier Series. ACM, New York, NY, 19-33. DOI= <http://doi.acm.org/10.1145/90417.90738>.

Columns on Last Page Should Be Made As Close As Possible to Equal Length

1. Elaborate on what Self-Gov is. [↑](#footnote-ref-1)