XXXXX

XXX X XXX
xxxxx
xxxx, xx, xxx
xxxxx@xxx.xxx

ABSTRACT

Many computing students will ultimately work in team environments, which could range from large enterprise systems, to small mobile applications. Working on distributed teams often have numerous challenges including how to coordinate project related decisions, ensuring proper integration of all created components, and how to properly deal with team related issues which are related to working on teams larger than a few developers. Unfortunately, students do not usually conduct team oriented exercises which will closely resemble this experience.

At the Department of Software Engineering at –hidden –, recent efforts have been focused on the project component of our Engineering of Enterprise Systems course as an area of innovation. We created a single, class wide project comprised of six teams, where all teams work towards a common project goal. The primary learning objects of the project include: Organizing team based decisions, component integration, quality assurance in distributed systems, project and process management among various teams, and risk analysis.

This project has achieved significant student praise; qualitative and quantitative feedback demonstrates both increased satisfaction and fulfilled curricular requirements. Students enjoy the real-world aspect of the project and the ability to work with relevant applications and technologies. This paper outlines the project details and educational goals.

Categories and Subject Descriptors

K.X.X [**Computers and Education**]: XXXXX—*XXXXXX*

Keywords

XXXX, XXXX

1. INTRODUCTION

A substantial portion of today's software is created in teams for a variety of reasons such as the inclusion of diverse skill sets, project magnitude, resource constraints, or scheduling. **[add more...]**

2. ABOUT THE COURSE

Although this project has been implemented in an Engineering of Enterprise Software Systems course, it can be used in variety of other computing courses including, but limited to: Distributed systems, Software Architecture, Project Management, or general Introduction to Software Engineering courses..... *[Dan: Sam: add more]*.

Enterprise systems are defined as a business software system which provides an organization an integrated solution to effectively and efficiently manage their resources. Some of the more important qualities of an Enterprise system include the ability to share common data across the system in a real-time fashion, and the automation of an organization's business processes [2]. Common examples of enterprise software include billing systems, customer relationship management, call center and customer support systems, and payment processing.

Primarily comprised of upper division Software Engineering students, the Engineering of Enterprise Software Systems course is also offered to other programs including Computer Science, Computer Engineering, Electrical Engineering, and Game Design. The only prerequisite for our enterprise course is an earlier class which introduces basic concepts of software engineering including teamwork, development processes, and proper documentation. In this prerequisite Introduction to Software Engineering course, students have already been introduced to basic software engineering principles such as software design, teamwork, testing and design patterns. In many cases, other non-required courses and a required one-year cooperative internship (co-op) have exposed students to various aspects of enterprise systems as well. The Software Engineering department considers the 3-credit Engineering of Enterprise Software Systems course to be the primary means of fully exploring the design, development, maintenance and support of Enterprise software.

The course has several primary learning outcomes including the proper design and maintenance of Enterprise Systems, the introduction to the necessary tools and processes to support these activities, and how to ensure that a reliable and secure solution which meets the customer's requirements have been developed in an on time, and on budget solution using proper software engineering metholdogloies.

Students are graded on several criteria. Each term there are three exams, several homework assignments, and a project based component. Class size is typically 25-40 students. The

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

course textbook is currently "Patterns of Enterprise Application Architecture" by Martin Folwer [1]. The course website is publicly available at:–hidden –

3. ABOUT THE PROJECT

3.1 Project Description

The goal of the project was to mimic an existing organization as closely as possible. In order to keep the project relevant to the student's interests and the real world, the basic premise of the project was a mobile phone manufacturing, and sales organization.

Many large, or enterprise applications are often comprised of components that work together to form the larger system[cite]. Each of these components may have been created using different technologies, organizations, and timeframes. Working together in large teams to create, and maintain these frequently diverse components can be challenged, even for the most experienced software developers. In order to mimic these environments in the real world, our project was comprised of several distinct groups which are described below. Instructors are encouraged to tailor the specific project requirements to their needs and the course which they are teaching. *[Dan: Sam: Modify these to sound good]*

1. **Human Resources** XXXXX
   (a) **Personnel Management** Manage the information for the company's employees. This includes personal information, salary, title etc....
   (b) **Salary Management** The salary of each employee will come out of the primary financial management system.
2. **Inventory management** This includes both parts needed to manufacture the phones, along with completed devices. The parts of the phone include the case, antenna, battery, HD, Ram, CPU/motherboard, and screen.
3. **Sales** When a device is completed, we intend to sell it to make a profit. When a device is sold, this change should be reflected in our inventory, and financial management systems. There are two primary customers groups including:
   (a) **Business Customers** These are large organizations like Best Buy, Staples etc... That sell our phones. Orders will be in units >= 10,000. Purchases will be made with credit, and the actual payment may take 5 days to be submitted. The price of these phone units will be different (and configurable) for each customer.
   (b) **Individual customers** Customers will go through a 'typical' online order process to buy an individual phone.
4. **Manufacturing** After getting the outside inventory, it will need to be manufactured into an actual phone. The team will need to account for:
   (a) **Handing rejects** Parts and phones may all be rejected during the manufacturing process.
   (b) **Assembly** Parts should be assembled
   (c) **OS Installation** OS should be installed
   (d) **Different models** The company makes and sells several models of phones which we manufacture.
5. **Accounting** Track all financial data for the company. Should also perform real-time tax calculations. Will have especially close tie-ins with sales and manufacturing systems.
6. **Customer Support** Ability to perform technical support to customers via a call center. Information should be recorded about customers who call in asking for support including: Name, phone number, address, order number, and reasons for calling. Customer support representatives should be able to mark that an item should be 'returned' and the device should be added back into a re-manufacturing queue and their money should be refunded.

3.2 Team Organization

At the beginning of the term, students are asked to provide feedback about which project group they would like to be a part of using an online questionnaire. Team size is targeted to 4-6 students, as this is often the size of groups in industry and has been found to be conducive to student learning in previous research [3,5]. Once students are placed into teams, they are then asked to assign each of their members one of the following coordinator roles:

1. **Team Coordinator:** Responsible for organizing each of the groups. Some activities included organizing team meetings, enforcing deadlines, communication coordinator, and conflict resolution.
2. **Data storage:** All groups were expected to work on a single, unified database and the data storage coordinator was responsible for ensuring that each group was properly interacting this database. Some of their other responsibilities include ensuring data and structural integrity, data storage design, and coordination with other teams regarding any data decisions which could affect other teams.
3. **Quality Assurance:** This role was primarily responsible for ensuring that each group followed a robust test plan. As an aggregate, all QA coordinators were responsible for creating a unified test plan for the entire system which could include, but not be limited to unit, accessibility, usability, and acceptance testing.
4. **Requirements/Design:** Although the general project outline and requirements are provided to each team, they were still expected to collect requirements throughout the project, along with implementing these requirements into a proper system design. All group requirements/design coordinators were expected to work together to ensure that all components would properly interact with one another.
5. **Integration/Compliance:** A paramount task for each group was to ensure that their component properly interacted with the other teams. Although many interacts would take place over the generated APIs, numerous other interactions would occur in other areas, such as the client side.

6. **Development/Version Control:** All groups were expected to use a single, unified version control system (vcs). We chose to use git for our offerings, although the instructor should feel free to use whatever vcs is most appropriate for the course. For each of their respective groups, the coordinator is responsible for ensuring that all members are properly using version control. As a class, these coordinators are responsible for ensuring that the code is being merged properly, and in the case of git, branches are being correctly used and the appropriate code is being pushed to the proper environments.

Although each group had several assigned roles, there was expected to be substantial "bleeding over" of each of these roles among team members. For example, although there is a defined quality assurance coordinator, all team members would be expected to participate in activities which would ensure the quality of the system. The coordinator would merely be the organizer, and point person of these activities. Depending on the size of the class, and the desired size of the teams, the instructor may decide to combine or eliminate roles as their situation dictates. Many of the roles also bleed together. For example, the version control coordinators and integration coordinators both deal with integration challenges at both the version control, or implementation side. One of the learning objectives of the project is for students to decide where to draw the line and assign responsibility for these related activities.

3.3 Deliverables

[Dan: Sam: Feel free to alter these deliverables as you see fit.]

The project is comprised of four minor, and three major deliverables which are shown in Table 1.

Table 1: Project Deliverables

Week	Deliverable	% Points
1	Team Formation	0
3	Strategic Team Plan & Requirements	5
5	Integration & Test Plan	5
7	Integration Tests	10
8	Release 1	20
11	Release 2	25
15	Release 3	35

1. **Team Formation:** In this simple deliverable, students are to select the teams that they wish to be a part of. In our offerings, students were instructed to complete an online survey with their team preferences.

2. **Strategic Team Plan & Requirements:** This release entails each group creating both a Strategic team plan and a requirements document for the project. The strategic team plan should be a short 2-3 page statement of purpose, high level plan, or abstract of what each team plans to accomplish. The goal of this strategic team plan is to serve as a high level overview of how each group is going to conduct business throughought the project. Some topics that each group may with to consider include:
   (a) What are your team's expectations?
   (b) What challenges will you need to overcome?
   (c) What is your initial plan?
   (d) What other groups will you primarily need to interact with and how will you interact with them?

   All project related artifacts are intended to change throughout the lifecycle of the project, but students should be expected to create an initial requirements document in this initial release. Depending on the course and the instructor's goals, they may want to provide this document to the student groups. Some components that may be included in this requirements document include the functional and non-functional project requirements, all internal and external team deliverables, assumptions, constraints, risks, and dependencies. Depending on the course this project is being utilized in, the instructor may desire to place more or less emphasis on this initial requirements document.

3. **Integration & Test Plan:** A primary learning objective of the project is how teams may design, create, and maintain components across several teams. The goal of this deliverable is for each team to create a plan for how they intend to integrate their component, with the project as a whole. Some aspects of this integration plan include the process they intend to use for communicating and agreeing upon changes with other teams, what technical risks and challenges they must overcome, how they plan on integrating their components from a technical perspective, and general team communication and agreement perspective. Teams are also expected to create an architectural diagram using proper UML notation about not only their component, but the system as a whole. The integration plan should also include the type of API the teams will use to communicate, such as SOAP or REST, along with a rationale of why they chose what they did.

   The test plan should include aspects such as how students plan on verifying the integrity and functionality of not only their component, but on any of the exposed api endpoints between other groups. This test plan is to largely serve as a foundation for their next deliverable, when the teams actually create their tests. We require each team to create a comprehensive set of unit tests for all api endpoints across the different teams and components. These unit tests will not only served as a method of ensuring program integrity, but will act as a way to evaluate and grade the performance of each team.

4. **Integration Tests:** In this release, teams should implement all appropriate tests as defined in their test plan. Although no functional project deliverables are required in this release, teams are expected to use *stubs* and *mock objects* to mimic functionally which they may test against. Using these stubs and mocks, teams should be able to demonstrate that they will be able to validate not only the functionality of their component, but in all interactions with other components before they even begin to develop any real functionality. This deliverable should create a shell-like piece of functionality for each of the groups.

5. **Release 1:**

   In the first release, teams should.....[finish this....]

   We required three main project deliverables in our course, although instructors may choose to alter the number of these deliverables based on the length of their project terms.

6. **Release 2** XXX

7. **Release 3** XXX

4. PROJECT RESULTS

4.1 Project Observations

We would like to share several observations and recommendations for instructors to use at their own institutions.

Coordinating tasks can be difficult: Students effectively and efficiently coordinating the activities of teams of 4-6 students can be a very difficult task, so several teams of this size coordinating activities is an even more challenging task. Early on, students will likely see how demanding, but imperative multi-team coordination is to project success. However, without proper instructor oversight, this could be an issue which becomes overly problematic for the course. Policy and access enforcement can also be difficult in projects such as this. When working on a single code and infrastructure platform with several different teams and several dozen students, restrictions must be made to enforce system access to only necessary, and appropriate parties.Some recommendations for overcoming this challenge include educating and enforcing a strong communication process among teams, often through the encouragement of a documentation sharing system or frequent, required in class meetings.

One example of a lack of coordination and policy enforcement was when a single student decided to make alterations to the course-wide project virtual machines. Unfortunately, these updates created conflicts with several necessary components which had a severe negative impact on the project. Fortunately, this issue arose early in the term and proved to be a good example of why coordination and policy enforcement is necessary for projects such as this.

Students realized that they needed to use coordinators and allow them to make decisions for the project: Early in the project, project decisions were made only after large, in class discussions among the 35 students taking the course. A discussion topic among instructors should be how this is not a sustainable process and that other communication and decision making plans need to be made, just like in a real organization. By using group coordinators as proxy decision makers, this enabled decisions to be made faster, while allowing other members of the group to focus on other tasks. One recommendation for instructors is to allow students to make the mistake of large, classwide discussions early in the course to allow them to witness the need for coordinators to act on the behalf of their teammates.

Planning and design is important: Creating necessary documentation and plans is of paramount importance for many software projects, but is often a less popular activity among students who would rather jump right into development[cite]. We have witnessed that students feel no differently during this project. One recommendation for instructors is to enforce the requirement of documentation and planning for the project, but then include post mortems and discussions throughout the project that reinforce these ideas.

Cross Cutting Concerns are challenging to deal with: Cross cutting concerns are defined as functionality that does not align well with any single module, and whose functionality may be scattered across a system [6]. Examples of cross cutting concerns for this course project include user authorization, data access, error handling and data integrity. Since Cross Cutting Concerns span multiple components, there was no single team that was responsible for addressing these issues. With team coordinators leading the discussion, the class needs to properly and fairly break up these responsibilities among the teams, and ensure that they are properly developed. A recommendation for instructors is to regularly discuss possible cross cutting concerns with the students and lecture on proper methods of dealing with cross cutting concerns. Instructors must also be proactive to ensure that cross cutting concerns remain at the forefront of the project, and that they are actively being properly addressed.

4.2 Student Feedback

Student feedback to this project has been generally very positive, where students have enjoyed the project for a variety of reasons. Some of these include the project's real world nature, their ability to make architectural and design decisions, the experiences gained simultaneously working with other teams on a common project, and how it is different from any other course project that they have worked on.

Students have expressed a significant amount of satisfaction in this project and it has contributed to their overall satisfaction with the course. At the conclusion of the term, students are asked to submit an anonymous survey rating several aspects of the course, instructor, and project. Several of these questions and student responses are shown in Table 2.

Table 2: Student Project Responses[Best way to show likert average]

Question
I enjoyed the project
I learned a lot from the project
The project prepared you for the real world
I would recommend this project to a friend
I feel like this project prepared me for working with Enterprise applicati

[Further discuss these results]

The end of term feedback also allowed students to write some of their thoughts regarding the course project. The following are samples of written feedback that have been received:

> "I liked the idea of incorporating with other groups that you weren't communicating with all the time. It also helped me become more familiar with setting up applications on servers and installing databases."

> "Learning a new technology and interfacing with other groups' applications. When it finally all came together, it was a really good feeling."

> "Coming from a co-op where I worked on using and creating API endpoints and working on an Enterprise system, I wasn't expecting to learn too much new material. Outside of the course materials where I learned some stuff about enterprise architecture, the project gave more insight on how to communicate and design API across multiple teams without having them previously defined. Unlike on my co-op, where documentation and guidelines were in place, working with a lot of unspecified architecture, led to thoughtful collaboration and communication both inside the teams and across the class."

Although the student feedback about the project has been generally very positive, there were some displeasures that some students have voiced. The first is that far too frequently, a few members of class made important decisions for the class as a whole. For example, during a class wide discussion on the project, a few students mentioned that they would like to use a certain technology for the project. Many other students who disagreed with this idea did not offer their challenges to this idea, so the technology was selected. Although this proved to be troublesome for the project, it did serve as a valuable learning experience for the students in that they should make their feelings known more often and not let a few speak for the majority.

While many students enjoyed much of the freedom that the project offered them, other students did not. They felt like this freedom did not provide enough constraints on the project. They also did not enjoy having to interact so heavily with other teams. One aspect of the course that did alleviate these displeasures was that each group was required to create mock objects, stubs and unit tests for their project, so that they could demonstrate their component's functionality without having to rely upon another group's functionality.

5. RELATED WORK

[do this section]

6. LIMITATIONS & FUTURE WORK

Although we have conducted this project in our Engineering of Enterprise Systems course, we have not used this actvivity in another courses. We believe that this project, or one which is very similar, could be quite beneficial in a variety of courses such as Distributed systems, Software Architecture, Project Management, or general Introduction to Software Engineering courses. Future work may be done to analyze how well a project such as our does in any of these offerings.

We have created teams using a simple, online survey system which has worked reasonably well. Future offerings of the project could include a more robust team creation strategy, such as the web based system described by Henry*et al* [4].

One goal of this project is to teach students to work on distributed projects, in a real world setting. Conducting this activity with another team, at another university would serve to fulfill this function. We are currently in discussion with several other institutions, including one in Europe, about simultaneously having several project teams work together on this single projects. Although this would likely be an invaluable learning opportunity, some challenges which would need to be overcome include how to fairly grade each groups deliverables, finding common times for all the groups to communicate, and uneven term schedules.

As with many team oriented projects, one encountered challenge which we were unable to fully overcome were situations were some team members did not contribute as much as they could to the project. Having taught numerous other team oriented software engineering courses, we have not noticed this problem to be more profound in this course in comparison with any team based projects in software engineering courses.

7. CONCLUSION

Software developers need to know how to overcome the challenges posed by the creation and maintenance of large, distributed systems. Some of these challenges include communication, technical, and resource constraints. Unfortunately, students are often ill prepared to deal with these challenges in academia and are often forced to work on small, team based assignments and do not achieve the necessary experiences needed to work with these larger systems.

In order to address these issues, we created a single, class wide project that allows students to experience many of the challenges in creating a single, component based system across multiple groups. Students have generally found the project to be very educational and feedback has indicated that it been a significant asset to them in industry. At least two other universities are planning to incorporate a similar project in their ciriclum and we encourage others to consider this approach in their courses as well.

8. REFERENCES

[1] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[2] F. Fui-Hoon Nah, J. Lee-Shang Lau, and J. Kuang. Critical factors for successful implementation of enterprise systems. *Business process management journal*, 7(3):285–296, 2001.

[3] J. Guo. Group projects in software engineering education. *J. Comput. Sci. Coll.*, 24(4):196–202, Apr. 2009.

[4] T. R. Henry. Creating effective student groups: An introduction to groupformation.org. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 645–650, New York, NY, USA, 2013. ACM.

[5] D. Petkovic, G. Thompson, and R. Todtenhoefer. Teaching practical software engineering and global software engineering: Evaluation and comparison. In *Proceedings of the 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, ITICSE '06, pages 294–298, New York, NY, USA, 2006. ACM.

[6] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton, Jr. N degrees of separation: Multi-dimensional separation of concerns. In *Proceedings of the 21st International*

Conference on Software Engineering, ICSE '99, pages 107–119, New York, NY, USA, 1999. ACM.