CS 3240: Advanced Software Development Techniques: An Evaluation and Proposal for Improvement

CS4991 Capstone Report, 2023

Adam Khan
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
azk7ad@virginia.edu

ABSTRACT

CS 3240: Advanced Software Development Techniques is offered within the UVA Computer Science curriculum and is characterized by a semester-long group software development project. As a student and former Teaching Assistant of the course, I have identified some of its strengths and weaknesses. The course provides a realistic simulation of real-world software development roles and fosters real-world experience. However, in-class instruction appears to be insufficient for ensuring success in the group project. Potential improvements to enhance CS 3240 include supplementing the assigned tutorial with in-class instruction and incorporating a structure that allows for gradual project progression where smaller components are developed over time under guidance. Future work would involve testing and evaluating the proposed improvements.

1. INTRODUCTION

“Any sufficiently advanced technology is indistinguishable from magic,” remarked Arthur C. Clarke, an eminent science fiction writer. This assertion underpins the essence of CS 3240: Advanced Software Development Techniques, a critical course within the UVA Computer Science curriculum designed to equip students with the skill set to maneuver the sophisticated world of software development. The course, primarily characterized by its simulation of real-world software development roles, has ignited curiosity among students toward the intricate functionalities of software systems. The students, under the guidance of Teaching Assistant Scrum Masters, undertake a semester-long group project, fostering invaluable real-world experience that proves valuable when applying for jobs and internships.

However, every pedagogical endeavor, irrespective of its merits, harbors in its framework room for improvement. This course, as seen from the perspective of a former student and teaching assistant, shares a similar story. The noticeable inadequacy in the in-class instruction has been discerned as a vulnerability in the course's framework. The abridged in-class instruction relating to the project assignment limits the students’ full potential, rendering the intricate tasks demanded by the group project considerably demanding and stressful. This vital shortcoming, therefore, poses an essential question: How can the structural foundation of CS 3240 be improved?

My proposal critically evaluates potential opportunities for enhancement. The ultimate objective is not just to critique, but to propose substantial improvements, thereby contributing to the transformation of CS 3240 into an even more insightful and impactful learning experience for future students. Laying a part of the path to make the “magic” of software development decipherable and successful forms the core of this capstone endeavor.

2. RELATED WORKS

The profound sentiment of a former student of CS 3240 highlights the pressing issue of insufficient class instruction (Anonymous, 2020). According to the student, they felt “thrown to the wolves,” a statement indicative of the struggle faced amidst limited guided learning and instructions where students are expected to navigate through the complexities of the course material independently. This portrayal evidenced a jarring gap between the in-class instruction provided by the faculty and the support needed to realize the complex projects included in the course.

A significant point of concern raised by the student involved their lack of familiarity with the primary languages used in the course, as evidenced in their statement: “I knew nothing of Django coming into the semester, and I had never even taken a class that used Python before.” This comment underscores either a potential shortcoming in the course prerequisites, or a missing bridge in the curriculum that failed to adequately prepare students for the substantial tasks involved in this course.

These viewpoints informed my perception of the course's strengths and weaknesses and directed my focus toward creating a proposal that would address these prevailing concerns. In the effort to bridge the gap described by the student review, I analyzed the Django tutorial assessment used in CS 3240 (“Writing your first Django app,” 2023). The completion of this tutorial is an assessment in CS 3240. Instead of fostering a thorough understanding of Django's inner workings, the tutorial concentrates more on creating a specific application—a web app where users can vote on polls. The result is a superficial understanding of Django's functionalities and principles, with learners merely acquiring the skills to recreate a standard, template-based project. This approach underprepares students to tackle unique problems and projects, such as those encountered in CS 3240. Consequently, there is a compelling need to reconsider the efficacy of this training resource within the context of this course and examine alternatives or improvements.

3. PROPOSED DESIGN

To enhance the effectiveness of CS 3240 and ensure students are well-equipped with essential software development skills, this section presents a detailed examination of the current syllabus, identifies key areas for improvement, and proposes a redesigned course structure.

3.1 Review of Existing Syllabus

The CS 3240 syllabus outlines a course structured around agile software development principles and centered on a semester-long team project. After introductory topics, students form project teams and undertake six sprint cycles to incrementally develop a complex web application. Sprints have specific deliverables such as planning documents, implemented features, and testing reports. Lectures cover software engineering concepts like requirements, architecture, design, quality assurance, security, and ethics. Assessments include quizzes, a final exam, graded assignments, and project milestones. Students are expected to gain hands-on experience in technologies like Django through self-directed learning. Lectures provide supporting conceptual knowledge.

The main issues identified in the current syllabus are insufficient in-class instruction on Django, over-reliance on a basic Django tutorial that does not provide enough depth, and lack of guided practice applying Django before starting the group project. These issues significantly impact students' ability to quickly gain proficiency in Django fundamentals beyond basic templating and apply Django to build non-standard web applications. Solving these issues would lead to increased student confidence with core course technologies, a deeper understanding of Django architecture and capabilities, and an enhanced ability to take on unique, complex projects.

3.2 New Syllabus Requirements

While the team project approach provides invaluable real-world experience, the current syllabus relies heavily on self-directed learning of core technologies like Django. Students have struggled to quickly gain proficiency in these unfamiliar skills, hampering their ability to successfully contribute to the group project in the early stages when foundational knowledge is still developing. Additionally, the limited in-class instruction on Django itself appears insufficient to instill the deep conceptual understanding required to properly apply it to a complex project.

To address these evident gaps in the current curriculum, the new syllabus should incorporate expanded Django training materials and graded assignments during the initial weeks to provide robust initial exposure. In-class code reviews, demos, and hands-on exercises using Django should be scheduled to reinforce concepts with guided practice. Especially in early sprint cycles, increased professor scaffolding and feedback can help propel student progress. Finally, breaking the singular large project into smaller milestone deliverables can provide measurable accomplishments that motivate students as they apply new skills. With enhanced support and supplemental instruction on key technologies, students can more smoothly onboard and gain the proficiency needed in unfamiliar skills that are foundational to effectively tackling the group project. Additional touchpoints throughout ensure continued comprehension while offering encouragement through incremental measurable progress.

3.3 New Syllabus Overview

The revised 15-week project timeline could be structured as:

- Weeks 1-3: Robust Django training culminating in a coding project
- Weeks 4-6: Short scope sprint cycles with professor code reviews
- Weeks 7-9: Core feature development (Project Milestone 1)
- Weeks 10-12: Advanced feature development (Project Milestone 2)
- Weeks 13-15: Bug fixing, testing, and polish (Beta Version)

Ongoing Django instruction would continue through demos and technical labs during the project phases to bridge theory with practice. With training upfront and initial practice on smaller projects, students can gain confidence before tackling larger milestone objectives. The professor can provide close guidance in the beginning, transitioning teams to greater independence in later sprints. Individual assignments also hold students accountable for foundational knowledge and skills.

This proposed syllabus overview aims to address identified gaps in the current curriculum through incorporating added support, segmented project deliverables, and supplemental hands-on instruction in the introductory stages. The course still centers on experiential learning and team collaboration. However, the inclusion of more robust upfront training, structured milestones, and touchpoints is designed to better equip students with the requisite skills for success in this capstone project course.

4. ANTICIPATED RESULTS

Students are likely to demonstrate improved proficiency and confidence with Django since the new syllabus structure provides expanded hands-on training upfront before requiring students to apply these skills to the large group project. Additionally, students are anticipated to gain a deeper understanding of software engineering concepts, as the in-class instruction on theory can be reinforced through practical application in guided coding activities and hands-on exercises.

The syllabus redesign should also lead to higher quality team project deliverables, as students will have gained sufficient knowledge and practice to contribute meaningfully to the group work, instead of floundering with unfamiliar skills. Finally, increased course satisfaction and perceived value are expected thanks to the incorporated support and scaffolding leading to fulfilled learning outcomes. Students are likely to appreciate the course more with transparency and purposeful design.

5. CONCLUSION

The proposed enhancements to CS 3240 represent a stride towards academic excellence in software engineering education. By addressing the critical gap in practical instruction and application, the redesigned syllabus is poised to foster an environment where students can not only engage but also master the complexities of software development. With a syllabus engineered to provide appropriate scaffolding and touchpoints, students can fully leverage the experiential team project model to develop professional skills. The proposed redesign aims to set students up for success in this capstone course.

The value of these improvements extends beyond academic performance, potentially elevating the employability of graduates by equipping them with a more profound and applied understanding of software engineering principles. This capstone project, through its focus on pragmatic teaching methods, underscores the importance of adaptability and hands-on experience in the rapidly-evolving field of technology.

On a personal note, this project has significantly contributed to my development as an educator and computer scientist, deepening my appreciation for the pedagogical challenges and rewards of fostering technical acumen within the constraints of an academic curriculum.

6. FUTURE WORK

The proposed curriculum redesign for CS 3240 is the starting point for a continuous improvement process. The next phase would involve the implementation of the new syllabus, careful monitoring of student outcomes, and iterative refinement based on feedback and results. Follow-up studies are essential to quantify the impact of these changes on student proficiency and job readiness. Collaboration with industry professionals to incorporate real-world case studies and guest lectures could further enrich the curriculum.

There is also scope for adapting the proposed model to other software development courses. Researchers might explore the pedagogical effectiveness of this approach, providing the potential to take this work in new directions, such as developing an open-source repository of educational materials, or creating a framework for peer-led supplemental instruction.

REFERENCES

Anonymous (2020, April 30). *CS 3240 Advanced Software Development Techniques Sherriff, Mark* [Online forum post]. theCourseForum. https://thecourseforum.com /course/7610/599/

Django (Version 4.2) [Computer Software]. (2013). https://www.djangoproject.com/.