Product, Process, and Professionalism: An Analysis of Industry Practices in Senior Design

An Undergraduate Honors Thesis
Submitted in Partial fulfillment of
University Honors Program Requirements
University of Nebraska-Lincoln

by
Samuel Anderson, BS
Computer Science
College of Arts and Sciences

May 6, 2021

Faculty Mentor:
Christopher Bohn, PhD, Computer Science and Engineering

Abstract

One of the biggest problems facing fresh Computer Science, Computer Engineering, and Software Engineering graduates intending to start their post-college careers is their lack of experiential learning and practical skills in real-world software development. From the UNL CSE Senior Design Mission Statement and Core Values, the goal of Senior Design is to "provide a project-based capstone course that gives students a design-centered educational experience," in order to "prepare students for the most challenging, innovative, and fastest-growing careers of the 21st Century." In short, the CSE Senior Design project is meant to be UNL students' first "real" design-centered project experience, with industry sponsors and UNL faculty supervision. Completion of CSE Senior Design allows for students to have some practical software development experience to reference on their respective résumé's for submitting to future employers. In order to examine the applicability of the CSE Senior Design curriculum to the state-of-the-art and the state-of-the-practice of real-world software development, this paper will outline where our Senior Design team was faithful to the prevalent practices in the industry, and where we might have differed in the areas of client management, version management, release management, requirements analysis, estimating, and design.

Key Words: computer science, computer engineering, industry, practical skills, career encedevelopment, software development, project management, requirements analysis, design.

Introduction

The Senior Design course is the most significant and culminating experience for undergraduate Computer Science, Computer Engineering, and Software Engineering students to apply everything that they have learned in their undergraduate coursework with respect to software development, product design, professional communication, and working in teams. With 88% of employers believing that it is important for colleges and universities to ensure that all students are prepared with the necessary skills to complete an applied learning project (Hart Research Associates 2015), the importance of the Senior Design experience from the employers' perspective cannot be understated. Furthermore, with there being 21 million college students nationwide and only 2 million internship positions (Lawrence-Fowler et al. 2015), Senior Design is the only opportunity for career-applicable service learning that is available to the vast majority of college students. Because the Senior Design experience is meant to prepare CSE students for the real-world software development experiences that they will encounter in their post-graduation careers, it is important to assess how closely the software development practices that are outlined and encouraged in CSE Senior Design adhere to the practices that are prevalent in the highly competitive software development industry.

This paper will examine the software development processes of client management, version management, release management, requirements analysis, estimating, and design, how these development processes were supported in this year's Senior Design experience, and relate the CSE Senior Design practices to those that are prevalent in the state-of-the-art and the state-of-the-practice of real-world software development. Conclusions are drawn from this

comparative analysis with the goal of determining whether or not the CSE Senior Design experience adequately prepares CSE students for real-world software development.

Client Management

Client management, in terms of software development and project management, involves maintaining the relationship between the software developers and the company that is contracting the developers to create said software. While client management does not encompass the actual programming or creation of software, it is still a crucial aspect of software development, as it facilitates the developer's understanding of what the client wants built, as well as the client's understanding of how that software can be built most efficiently with the least amount of confusion. This also ensures the sponsor's confidence in the development of the final software product by reinforcing that their vision of the final product is understood by the development team and can ultimately be realized.

In CSE Senior Design, the structure of the software development teams followed the Agile Scrum team structure, which includes a product manager, a development manager, a squad lead, and the developers. The product manager is the design role that is responsible for the project vision, and how the software solution ought to be built. The development manager is the lead developer that is responsible for directing the implementation of the software solution and prioritizing features based on design implications put forth by the product manager. The squad lead is responsible for ensuring that all of the Senior Design practices and Agile values are followed. With our small 5-person team, we assigned the roles of product manager and squad

lead to one developer, and the role of development manager to another developer. For the purposes of client management, our team also delegated the responsibility of leading the communication with the sponsor team to our product manager/squad lead.

With the circumstances surrounding the COVID-19 pandemic and its effects on our year in CSE Senior Design, our development team went to great lengths to keep everyone on our sponsor team in the loop by documenting all development progress and communication with the sponsor in the form of weekly status reports, email and Slack message correspondence, and both video recordings and written meeting minutes of the weekly meetings with the sponsor team over the Zoom teleconferencing platform. This communication was made available to both the members of the sponsor team and the University of Nebraska-Lincoln faculty who could not make it to a number of the weekly sponsor meetings due to scheduling conflicts, in order to keep them up to date on the current state of the project. The documenting of this communication helped greatly in ensuring that our team could still be successful in the absence of any face-to-face meetings with the client throughout the year.

While it is much easier to be personable and customer-focused in the small-scale software development experience that CSE Senior Design provides than in working under a multinational technology company such as Google or Microsoft, these values still have importance for larger-scale companies and development teams, too. Take Microsoft's client management practices, for example. Microsoft recognizes that, as their organization grows, it is easier to lose sight of the customers that they are serving, and so they go out of their way to form relationships and keep empathy between the product group and the customers by pairing up individual

developers from the engineering team with individual customers from the companies that they work for, so that they can better understand the requirements of these companies on a more personal level (Woodward 2018). This isn't something that our five-person development team had issues with in working for a six-person Senior Design sponsor team, but it is something to keep in mind in the future as we move on to working in larger development teams for larger companies.

Version Management

Version management, otherwise known as version control, is the process by which the development team tracks and documents changes to the code repository over the course of the software project's development. Version management is vital in software development because it allows for the members of the development team to work concurrently on the same files, backup the code repository, and recover from mistakes that have manifested in newer versions of the software by restoring older versions of the software for bug-fixing.

In CSE Senior Design, the project development teams were instructed to use Git as our version control system, and GitHub as our hosted source code management platform. Git and GitHub certainly both qualify as state-of-the-practice industry software, with many major tech companies such as Twitter, Netflix, Amazon, and Google using them in their tech stacks (Schildt 2020). Our development team learned many important software development lessons from working in a shared team repository, including how to review other developers' code for approval, how to create pull requests, and how to manage merge conflicts.

One of the industry standard practices for version management that Google makes use of is that all changes that are made to the main source code repository have to be reviewed by at least one other software engineer (Henderson 2020). Our team for Senior Design expanded on this practice, and required two other developers to manually review and approve any pull requests before they could be merged into the master branch. This ensured that at least three pairs of eyes were on every change to the source code, and we could catch mistakes that other developers might have missed in their original commits. In this way, our team expanded upon the industry code reviewing practice that Google has put into place with mandatory code reviewing.

Another version control practice that our Senior Design team made use of that can also be seen in development at Google was having the developers work on feature-specific development branches until any new changes have gone through a full round of testing. This approach attempts to solve the problem of the instability of the software product across different versions. Google has gradually moved away from development branch-based version control in recent years, due to the fact that there are significant scaling risks when relying on development branches (Winters et al. 2020). However, for our Senior Design team's small-scale purposes, development branch-based version control worked quite well.

Another prevalent industry practice in version control and code review, as can be seen at Microsoft, is to have developers make small, incremental changes that are easy to understand, as well as cluster related changes together in the same commits to the source code repository (MacLeod et al. 2018). Our Senior Design team adhered to this practice by only changing a few files of the source code with each commit to the repository, always writing substantial commit

messages that described all of the changes that were made, and clustering the changes to have each commit focus on delivering a specific feature or bug-fix, in order to make version control and code review as simple as possible for the other developers on the team.

Release Management

Release management in software development is the process by which the development and deployment of the software by the development team is aligned with the business priorities of the client. Proper release management practices involve showing the client what has been accomplished with the project during each release, as well as demonstrating the feasibility of the project being completed with the amount of time remaining in the project period.

Our Senior Design team outlined the goals and target dates for each release in the project roadmap, which was discussed during each weekly sponsor meeting as the scope of the project changed with each release. The main highlights of the features added to the software during each release are also documented on our project's GitHub wiki. During each of our release meetings, we demonstrated the value of our web application solution through extensive user acceptance testing with the sponsor team, carrying out all of the necessary operations that their proposed software system required.

Requirements Analysis

Requirements analysis is the elicitation of all of the functional and non-functional requirements of the software system that the client needs to be implemented in the final product. The implementation of these requirements in the final software solution is documented in the form of user stories, where the developers create a simplified description of a given software requirement from a user's point of view, outlining what that user wants to do with the software and why they want to do it. This area of software development is critical to the successful development of a product, ensuring that the final product delivers all of the client's required functionalities. Requirements analysis has a significant amount of overlap with client management in the area of constant, comprehensive communication with the client over all of the planned aspects of their proposed system.

In the case of our Senior Design team's project, the elicitation of the project's requirements from our sponsor was a back and forth process that took a few months to finally arrive upon the complete list of functional and non-functional requirements, which resulted in less development work on the project in the first half of the academic year. Our development team believed that this was a consequence of the sponsor team not having a firm grasp on all of the necessary functionalities and requirements of their proposed system at the start of the project. In the retrospective feedback from the sponsor team for our project's final release, they echoed this sentiment, saying, "This was not an easy concept to grasp, as even in this office not all [of us] understand the process." This goes to show that the requirements analysis process, in our development team's case, is an iterative process whereby the client is made to reflect and

deliberate upon the necessary product requirements week after week in order to arrive on the final list of requirements that the development team can work with in designing the final product.

Estimating

Estimating is the process by which the development team predicts the relative amount of time, in units of "story points," it would take to complete a given feature or story in the development storyboard. The process of estimation is crucial to software development because it allows for the development team to prioritize key features of the target software that have more story points allocated to them, and put features with fewer story points in the development backlog, or "icebox." In Agile software development, the estimation of story points is also used to track development progress in the form of a "burndown chart," which is a visual representation of how many story points have been completed during the course of a development sprint.

In CSE Senior Design, the project development teams were advised to use ZenHub, an Agile project management tool that natively integrates with GitHub, for our development storyboarding and estimating. At the beginning of each development sprint, every member of our development team estimated and assigned story point amounts to the user stories on the ZenHub development storyboard based on the relative amount of development time it would take to complete each user story. Then, we would assign developers to the user stories in such a way that each developer would be working on roughly the same amount of story points for the sprint. Our team's experience with estimating and assigning story points during each development sprint was that we got much better at our story point estimates during the later stages of the project when we

had a greater understanding of the project's requirements and how long it would take to implement a given feature in our software solution.

Design

Software design is the process that immediately follows the requirements analysis of the client's proposed software product. The design stage consists of conceptualizing the final software solution that will fulfill all of the client's proposed requirements, and making plans to deliver on that conceptualized design. The design stage of software development is important because it allows for the development team to formulate an idea of the final product that they need to be working towards throughout the course of the project.

Our Senior Design team's initial conceptualization of the final product solution involved creating a non-functional wireframe of the web application solution that was proposed by our sponsor. We then demonstrated this wireframe with the sponsor team during a release meeting to determine whether or not it was up to their standards, and if any design changes needed to be made before we moved to develop the working web application solution. In subsequent project releases, our design ended up evolving over time based on new requirements that were brought forth by the sponsor team. After the product design was outlined to the sponsor's final specifications, development work proceeded on the application with a test-code strategy, where our development team wrote the unit-test cases while writing the production code in order to direct immediate development work towards passing these unit-tests. Intel Shannon, an Intel research facility located in western Ireland, has also made use of test-code development to great effect,

with the developers obtaining a better understanding of their software from the client's point of view using this strategy (Fitzgerald 2005).

Conclusion

This paper examined the software development processes of client management, version management, release management, estimating, requirements analysis, and design in CSE Senior Design, and provided a comparative analysis of these processes with the state-of-the-art and the state-of-the-practice seen in prominent tech companies, such as Microsoft, Google, and Intel. With the CSE Senior Design curriculum placing a great emphasis on the Agile development framework that is favored by many major tech companies in the industry, there is much evidence to suggest that the CSE Senior Design experience greatly prepares graduating CSE students for real-world software development in their post-graduation careers. Furthermore, in the case of some Agile development practices that are not being widely utilized in the industry, such as sprint review meetings and burndown charts, there are still many practitioners that strongly agree with the benefits associated with the usage of these practices in software development (Rauf and AlGhafees 2015). Therefore, this paper concludes that the CSE Senior Design curriculum does provide CSE students with many opportunities to develop software development skills and improve their career readiness to facilitate their transition from the academic environment of Senior Design to the real-world software development industry. Moving forward, graduating CSE students could look to share what they have learned in their Senior Design experiences with the companies that they intern with, in order to bring good software development practices to these organizations and to other interns working at said organizations.

References

CSESeniorDesign. “UNL Computer Science & Engineering Senior Design 2019-2020 Year In Review.” Issuu, issuu.com/cseseniordesign/docs/2001.019_cse_20annualreport_vf_web.

Fitzgerald B., Hartnett G. (2005) A Study of the Use of Agile Methods within Intel. In: Baskerville R.L., Mathiassen L., Pries-Heje J., DeGross J.I. (eds) Business Agility and Information Technology Diffusion. TDIT 2005. IFIP International Federation for Information Processing, vol 180. Springer, Boston, MA. doi.org/10.1007/0-387-25590-7_12

Hart Research Associates, "Falling short? college learning and career success," Association of American Colleges and Universities, Washington D.C., 2015

Henderson, Fergus. “Software Engineering at Google.” ArXiv.org, 30 Jan. 2020, arxiv.org/abs/1702.01715.

MacLeod, Laura, et al. “Code Reviewing in the Trenches: Challenges and Best Practices.” IEEE Software, vol. 35, no. 4, 2018, pp. 34–42., doi:10.1109/ms.2017.265100500.

A. Rauf and M. AlGhafees, "Gap Analysis between State of Practice and State of Art Practices in Agile Software Development," 2015 Agile Conference, 2015, pp. 102-106, doi: 10.1109/Agile.2015.21.

Schildt, Daniel. “List of Companies That Use GitHub.” GitHub, 2 June 2020, github.com/d2s/companies/blob/master/src/index.md.

Titus Winters, Tom Manshreck, and Hyrum Wright. *Software Engineering at Google: Lessons Learned from Programming Over Time.* Sebastopol, CA: O'Reilly Media, 2020.

W. A. Lawrence-Fowler, L. M. Grabowski and C. F. Reilly, "Bridging the divide: Strategies for college to career readiness in computer science," 2015 IEEE Frontiers in Education Conference (FIE), El Paso, TX, 2015, pp. 1-8, doi: 10.1109/FIE.2015.7344317.

Woodward, M. (2018, June 11-14). *How Microsoft does DevOps* [Conference presentation]. O'Reilly Velocity Distributed Systems & DevOps Conference, San Jose, CA, United States. https://www.oreilly.com/library/view/velocity-conference-/9781492026051/video320679.html