Approachable Code: Developing With Abstraction in Mind

Structuring a Club for Busy Introverts

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Ian Harvey

October 27, 2022

ADVISORS

Benjamin Laugelli, Department of Engineering and Society

Rosanne Vrugtman, Department of Computer Science

Introduction

The Student Game Developers club (SGD) at the University of Virginia (UVA) is a club made up of a variety of individuals, each passionate to contribute towards making video games. It is a club reliant on the necessity that its members dedicate significant amounts of time each semester towards this process despite their already busy schedules. This entails the learning of niche, complex software that many find scary while learning. It follows that the club struggles to retain the members it manages to bring in each year. In order to ensure the club's continued existence, its members are tasked with structuring it so that the technically complex aspects of game design do not drive away individuals due to their fears of technical inadequacy.

After analyzing the current structure of SGD's projects, I propose the implementation of known professional coding standards to reduce the effect unfamiliarity has on new members. These standards include naming conventions, code flexibility, hierarchical structuring, and example-based programming that simplify the process of understanding and writing code. As each SGD project is reliant on both technical and social aspects, an understanding of how both factors work together is critical for the technical project's success. To develop this understanding, I will use the STS Framework of actor-network theory to determine how current niche and messy code produces an environment where new members do not see a benefit to working on these projects.

Understanding the technical project as a solution to the technical problem, but not the social aspect, increases the likelihood the problem will reappear as new people present new coding styles to the club. Therefore, addressing both the social and technical aspects of this challenge produces a more lasting and proper solution to this problem. In the next sections, I examine, first, a technical project aimed at improving the club's code structure and, second, an

STS project examining the social factors of this challenge. The insights gained from observing sociotechnical aspects present in the STS project will allow further tuning of the technical project to accommodate new members.

Technical Project

The Student Game Development club's code consists of unchecked, niche, hastily put together script files that are extremely difficult for an outsider to understand. James Connors (2023), former SGD club president, attributes this in part to the fact that the club's game directors, "won't make a very maintainable codebase when they start." This is in part because the club's directors are often asked to independently create a demo to present to the club at the start of the semester in the hopes of getting people interested in joining their project. This results in the creation of poorly written, barely functional code that serves as the basis of the team's project for its entire duration. Code like this is a problem because a codebase's readability plays a significant role in a programmer's comprehension capabilities (Johnson et al., 2019), and when code becomes incomprehensible, a programmer finds it much more difficult to contribute to a codebase (Storey, 2005). Inexperienced members are being asked to add onto this codebase, but they often struggle to do so. Calls to misnamed sections of code do not provide what an individual would expect them to provide. Certain scripts are too functionally rigid to be reused anywhere else. The placement of files inside of the folder hierarchy is a jumbled mess. Overall, the code is unapproachable by anyone who did not themselves write it.

Directors have attempted to address this issue through the use of version control, allowing them to review the code and ensure that it is of quality. This, in theory, prevents bad code from ever being used in the main build that everyone bases their changes on. It also allows for work on simultaneous changes so long as they do not overlap in what they are changing, theoretically increasing the project's throughput. But because of time crunches and a lack of specification for good code format, directors are often left feeling forced to accept bad code out of a fear that their projects will not progress without it. This causes the codebase to become increasingly unapproachable throughout the semester, reducing work output and efficiency, and damaging the quality of the final game.

This technical project tries to avoid that pitfall by introducing a strict standard for code quality, introducing requirements for naming conventions and script reusability, and providing template code for programmers to use to base their changes on. When it comes to code comprehensibility, individuals newer to programming often fail to map value to code when it is named incorrectly (Gross and Kelleher, 2009). In addition, when they are provided with code similar to what they need to produce, they are able to understand the code much better (Sadowski et al., 2015). In addition to this, all members will be made aware of the code that was introduced into the main branch throughout the week during team meetings. The code's functionality will be explained, its current usage will be demonstrated, and suggestions on how it may further be used or applied to current jobs will be provided. These implementations build on the version control approach and may provide as an example as to how one may more easily attain the aforementioned benefits of said practice.

The evaluation of success for this projection will be dependent on three factors. First, the rate over a one-month period at which non-developer individuals contribute to the codebase in this project in comparison to the rate at which they contributed within previous projects. Second, the number of members initially interested in programming for the project who subsequently did not contribute anything throughout the course of the semester. Third, the rate and reason for which members of the group required assistance from the director of the project in implementing their code. These three factors respectively will help evaluate the ability of new members to comprehend and contribute to the codebase, the likelihood an individual was able to contribute to the codebase, and finally the level of standalone independence new members have when introduced to the codebase.

STS Project

Game development is a highly technical process that requires a carefully procured subset of niche programmatic knowledge in order for one to succeed. The Student Game Development club at UVA is a club that takes the complex process of game development and attempts to make the experience of working in such a field more accessible to students interested in doing so. SGD experiences a consistent pattern in which they begin a semester full of individuals interested in participating in such a project and end said semester with a fraction of them remaining. Multiple meeting notes of officers from the club recognize this phenomenon and make comments wondering why exactly it always tends to happen (Student Game Developers at UVA, 2019).

It is believed that people leave the club for reasons such as the lack of gender diversity within the club, the stress of an added responsibility the projects present, or the overall intimidation factor of learning Unity, the game engine software the club uses to produce its games (Xu, 2023). While these factors certainly contribute to attrition in the club and working to address them has slightly reduced its effect, they overlook the effects that other more pressing factors have on people’s decisions to stick with the club. For one, the individuals joining the club by nature of the club are comprised mostly of introverted individuals. Introverted individuals tend to stick with activities that they find more beneficial to personal development (Toma, 2015,

p. 123). Secondly, college students stand to benefit academically from good time management in their already busy schedules (Britton and Tesser, 1991). Finally, the club's code is not written with a quality that makes changes easily implementable, causing them to take more time than initially intended.

In looking at these factors on top of the currently considered factors, one is able to develop a better overall understanding as to why the club consistently fails to maintain a stable membership count. Overall, new members consider a single benefit in joining the club: learning game development. However, this single benefit is largely outweighed by the numerous detriments joining the club presents. This, altogether, makes it unlikely that new members will choose to stick with the club.

Using the science, technology, and society (STS) framework of actor-network theory (ANT) I plan to argue how it is a lack of diversity, added stresses, intimidation, introverted personalities, time management, and poor code quality that all comes together to cause the club to fail in this avenue. More specifically, all of these factors work in tandem as observable detriments to new members who join the club. As they consider these elements together, they are convinced to stop contributing to the club, thus further contributing to the club's membership attrition. The theory I will use to argue this, ANT, is a theory developed by Michel Callon, Bruno Latour, and John Law. It makes the claim that engineers build networks comprised of actors, both human and non-human in nature, that relate to certain parts of a network such as aspects in technical, social, natural, and conceptual standings. ANT makes the claim that the relationship between all of these actors in a network is what determines whether or not a network succeeds (Cressman, 2009). When supporting my argument, I will draw on information gathered primarily from interviews with club officers, responses to club wide surveys, and reports on sociology and psychology, such as Shannon Paige Toma's report on the effect personality has on college experiences (Toma, 2015).

Conclusion

The results of the technical project will provide insights into the effect introducing standardized coding has on the likelihood that an individual will participate in, and continue participating in, the activities of clubs with a high technical barrier to entry. My STS project will provide insight into the reasons why individuals currently choose to abandon participation in said club in the first place. With the results of the STS project, I will be provided a better understanding of the factors affecting individuals' choices to leave the club and will be able to apply my knowledge regarding their tendencies and desired environments to my technical project in order to alter the environment of the club to their expected liking. In applying my knowledge from the STS project to the technical project, I am able to reduce the effects that the technically complex environment of SGD has on new members by making the code of the club more approachable. In turn, the new environment will encourage people to stay in the club and help resolve the club's problems with attrition.

References

Britton, B., & Tesser, A. (1991). Effects of time-management practices on college grades. *Journal of Educational Psychology*, *83*, 405–410. https://doi.org/10.1037/0022-0663.83.3.405

Connors, J. (2023, October 19). Interview with James Connors. personal.

Cressman, D. (2009, April). A brief overview of Actor-Network Theory: Punctualization, heterogeneous engineering & translation.

Gross, P., & Kelleher, C. (2009). *Non-programmers identifying functionality in unfamiliar code: Strategies and barriers.* https://openscholarship.wustl.edu/cse_research/19

Johnson, J., Lubo, S., Yedla, N., Aponte, J., & Sharif, B. (2019). *An empirical study assessing source code readability in comprehension*. https://doi.org/10.1109/ICSME.2019.00085

Sadowski, C., Stolee, K. T., & Elbaum, S. (2015). How developers search for code: A case study. *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 191–201. https://doi.org/10.1145/2786805.2786855

Storey, M.-A. (2005). Theories, methods and tools in program comprehension: Past, present and future. *13th International Workshop on Program Comprehension (IWPC'05)*, 181–191. https://doi.org/10.1109/WPC.2005.38

Student Game Developers at UVA. (2019, September 22). Meeting notes 9/22. personal.

Toma, S. P. (2015). *Personality and the College Experience: How Extraversion-Introversion Measures Shape Student Involvement and Satisfaction*. [Doctoral Dissertation, University of California, Los Angeles]. UCLA Electronic Theses and Dissertations

Xu, C. (2023, October 22). Interview with Catherine Xu. personal.