Prospectus

Software Architectural Patterns: A Performance Analysis
(Technical Topic)

Software Abstraction and the Impact of Methodical Software Design on Society
(STS Topic)

By

Vineeth Gaddam

10/22/19

Technical Project Team Members: Sai Konuri and Aman Garg

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

Signed: _____Vineeth Gaddan_____

Approved: _____ Date _____
Benjamin J. Laugelli, Department of Engineering and Society

Approved: _____ Date _____
Nada Basit, Department of Computer Science

Sociotechnical Problem

In 2017, it was reported that software failures had cost the economy over a trillion dollars and affected over 3 billion people around the world. A premium on efficiency of software architectures has risen due to the rapid increase in data storage and consumers using software products in the world. In a world that is so interwoven with software and data, having over 300 companies experience software failures and consumers being at risk of data theft is an issue (Raygun,2017). The basis for these issues can be traced to companies having poor guidelines and an inadequate implementation of different software architectures. It is pertinent that software failures begin to subside to prevent reoccurrences of incidents such as the Equifax hack, during which over 140 million records of user data were stolen by hackers, which equates to almost half the United States' population (Matsudaria). During the software development process, there needs to be a solid basis of communication between the developers and the one giving the requirements. The current approach in many software companies today is to have requirements set by a manager and then utilize a common software architecture, also called software framework, that may not be fully efficient for a given system. Many of the bugs can be traced to the miscommunication between the different developers and the requirements that may change but not be clearly established to the programmer (Matsudaria). This leads to bad software and hidden bugs that are not discovered which can lead to a company losing millions of dollars or putting consumers data at risk due to not having solid efficient software architecture. The technical solution to this issue is to develop a software architecture guide that will outline many different architecture patterns, with information on how and where to implement a certain pattern. The guide will allow for companies to gain more knowledge about how to have a more

efficient rigid structure that will be efficient from the moment an idea is hatched to the building of the product, testing the product, creating safety protocols, and maintaining the product. However, this does not fully address the social ramifications and issues that come with the implementations of the Software Architectures. For example, there are many factors that could cause a failure to happen such as the miscommunication between teams, ambiguity in requirements, and a bad workplace environment.

In other words, to fully resolve this issue, the solution will need to address not only the technical aspects but also the social aspects. The technical portion will address the problem of the ever-increasing data and importance of efficient frameworks. Social implications of the impacts of having a loose architecture and its effects to the public will be explored through the study of the Facebook Cambridge Analytic case.

Technical Problem

As the world continues to shift its focus to technology and society becomes ever more reliant on different kinds of software applications, the need for proper efficient implementation and maintenance of software frameworks becomes more pertinent. As more companies are becoming more accustomed to creating software, it is increasingly important to begin to have methodical strategies to create products and improve upon old methods (Karam & Mohsin,2018). The current practice of having requirements set by the managers and stakeholders, then utilizing a legacy software pattern that does not fully keep up to the current technologies, creates a huge problem and severely hurts the efficiency of the process. Software patterns refer to the general reusable guideline that teams will use to aid them in the development cycle (Shvets,n.d.). The

current implementations by many people lead to several issues. There is often too much lag in the communication between the and the software development team about the system requirements. The implementation of old software patterns that are utilizing outdated technology and not adapted to the massive influx of data can lead to slowing down of the systems and leave them vulnerable to attacks. This can further lead to getting hacked and having sensitive consumer information stolen and used for malpractices (Tan,2016). More awareness and analysis needs to be brought to the software architecture realm. It is very crucial that the structural choices that companies make regarding how they want their software architecture to look, to be the most informed, and applicable to the type of product that is being created. Once the architecture is implemented and the process is underway, it is very cost intensive to make core structural changes. So, it is important that a company implements the best architecture pattern that suits its needs during the initial stages of planning to avoid redesigning the architecture. (Moises,2018) As a whole, the architecture would allow companies to gain an understanding of how their software systems will behave; they will also be able to reuse certain elements and increase risk management.

The most common practice for developing software is the layered pattern, where each level starts off small and sends information to the next layer which is more general as you go up the chain. This idea is a solid choice for creating and handling a website, as programmers can easily split up the different logics, such as the user interface (ui) and data (Mallawaarachchi,2018). However, the program may not be as good for a more dynamic and responsive software that is more hands on. The project will help make the process of selecting or updating a software pattern simpler and allow people to have valuable information to make the

best choice for their product. The goal of this technical project is to design an information sheet in a pamphlet style that outlines the various different software architecture patterns and ideologies. This sheet will allow companies to plan out their architecture based on the product that is being developed. The pamphlet will have diagrams breaking down the different levels of how different teams communicate, how information is displayed and processed, and it will also provide metric data showing how efficient certain methods are for certain tasks. The pamphlet will also provide technologies to use to help implement the architecture. By the end, people will have a more thorough resource that can be consulted when designing an application that builds upon the current knowledge of architectures that companies utilize. The information for the pamphlet and the research was discovered by creating an open source project. JMeter is an application that allows for the testing of different frameworks to see how efficient the applications are during different circumstances and varying amounts of traffic. Ruby on Rails is an example software coding framework to make web applications (Castello,2019). The project will run tests on the application made by Ruby on Rails under different situations and give performance results. Then the information about the framework can be added to the pamphlet with information regarding the speed and the correct usage of the framework. Since it is open source, other people can load various frameworks to see performances and analyze what action to take. This will allow people to be able to tailor the design of the platform based on the type of service that is being created and have an up to date software pattern that will be efficient for their respective systems.

STS Problem

Taking a closer look at the 2015 Facebook Cambridge Analytic debacle. Looking at the different aspects of this scenario, it is evident that the company was impacted, millions of users were impacted, there was money lost, and it led to a potential hack into our voting democracy (Spangler,2018). The impacts of not having a solid architecture extend beyond the business realm and into the political and social worlds as well. This event outlines that even the biggest of companies can have problems with their software architecture. Looking at the different aspects of the situation, it was evident that there was miscommunication across different teams on how to handle form data from third parties, which allowed Cambridge Analyatica to access user personal data. Consumers had their sensitive information sold to the highest bidders and it led to calculated subliminal marketing regarding the presidential election, where certain candidates had more positive commercials appear on the user feed (Zialcita,2019). Now if we take a step back and look at the reasons why the miscommunication occurred, we can gain a better understanding of the scenario. The workers at Facebook had harsh deadlines, unclear requirements that kept changing without the approval of the team in some cases, and there was a lot of pressure on the workers to complete their tasks quickly due to the company being Facebook. This in turn also led society and the government to take a closer look at how technology companies are treated and how they are handling the personal data of their users. The software architecture could have been more secured and rigid and would have gravely helped Facebook from having this ripple effect that had an impact on not just the economical ecosystem, but the social ecosystem as well (Team,2019). It is my belief that if more precautions are taken early on and businesses focus on having a solid implementation of the software development

process, many of these issues can be addressed and situations like the Cambridge Analytical hack will not occur. Individuals are most prone to make the wrong decision when they do not have adequate information at hand. The same ideology applies in this context, where not knowing fully how to implement the right software architecture for a given situation can have many lasting consequences to not just the software, but to society as well. To see this, I am drawing on the science, technology, and society (STS) concept of Social Construction of Technology (SCOT), which outlines how different groups such as stakeholders and managers interpret technology and how their views are affecting the design process. In the Facebook case the managers wanted to have their product out for the public and rushed their work and were not thorough in their development which led to an insecure software that was open to an attack.

Conclusion

All in all, our technical research plan will deliver a pamphlet guideline that will consist of a bundle of information on the proper set up for different software architectures, and provide efficiency runtime stats, with the necessary tools required to utilize the architecture to its fullest. This design will take into account how big the teams are, how intensive the software is, and how to make sure a team is following through with the implementation of the architecture. The pamphlet will be easy to read and will contain diagrams and charts to explain the flow of certain patterns. While the STS paper will strive to bring to light and further explain how the malpractice of using a software pattern can lead to the creation of liable software that can be taken advantage of. The consequences for this can be people losing their livelihood, companies losing money, and data getting into the wrong hands. It will also outline how the work place

environment and communication within the company can lead to the architecture to not be fully utilized. So, all in all the results of the STS paper will help to analyze the social implications of companies not having a solid grasp of the software architecture during the software development process.

References

Castello, J. (2019, June 22). What is Ruby on Rails & Why Is It Useful? Retrieved from https://www.rubyguides.com/2018/10/what-is-ruby-on-rails/.

Karam, L., & Mohsin, A. (2018, November 28). Why is a good software architecture so important? Retrieved from https://apiumhub.com/tech-blog-barcelona/importance-good-software-architecture/.

Moisés. (2018, November 15). Efficient Software Architecture. Retrieved from https://thepracticaldeveloper.com/practical-software-architecture/software-architecture-issues-and-solutions/.

Madsudaria, K. (n.d.). Bad Software Architecture is a People Problem. Retrieved from https://queue.acm.org/detail.cfm?id=2974011.

Mallawaarachchi, V. (2018, April 27). 10 Common Software Architectural Patterns in a nutshell. Retrieved from https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013.

Shvets, A. (n.d.). Design Patterns and Refactoring. Retrieved from https://sourcemaking.com/design_patterns.

Spangler, T. (2018, March 29). Facebook's Cambridge Analytica Debacle: Why This Data-Privacy Storm Is Different, And What's Next. Retrieved from https://variety.com/2018/digital/news/facebooks-cambridge-analytica-scandal-analysis-regulation-1202734392/.

Tan, G. (2016, August 26). A Collection of Well-Known Software Failures. Retrieved from http://www.cse.psu.edu/~gxt29/bug/softwarebug.html.

Team, T. E. (2019, September 16). The 7 Most Important Software Design Patterns. Retrieved from https://medium.com/educative/the-7-most-important-software-design-patterns-d60e546afb0e.

Zialcita, P. (2019, October 30). Facebook Pays $643,000 Fine For Role In Cambridge Analytica Scandal. Retrieved from https://www.npr.org/2019/10/30/774749376/facebook-pays-643-000-fine-for-role-in-cambridge-analytica-scandal.