Collaborative Discussion 1

Project Failures Study

Initial post

Three main reasons commonly lead to software project failure.

1. insufficient risk assessment

Healthcare.gov, aka Obamacare Website, was Launched in October 2013 and suffered from inadequate risk assessment and short load modelling. "The website simply was not designed to handle the massive number of individual users. Plenty of missed opportunities to fix the website before its launch." (Anthopoulos et al., 2016), resulting in technical issues such as slow performance and system crashes. Despite testing phase red flags, the launch proceeded, pointing to a need for proper risk evaluation.

2. Insufficient Quality Control

Quality control is crucial in software development. One notable example is Boeing's 737 MAX aircraft. The quality issue of new Maneuvering Characteristics Augmentation System (MCAS) was ignored, leading to a system flaw that caused two fatal crashes in 2019 (BBC, 2019).

Following these incidents, Boeing had to ground the 737 MAX and make significant changes to the aircraft. The company also faced billions of dollars in fines and settlements. This highlights the dire consequences of insufficient quality control.

3. Unrealistic Expectations

Creating unrealistic expectations can lead to significant backlash if the software does not deliver. This point was evident with the launch of Cyberpunk 2077 by CD Projekt Red in 2020. The highly anticipated game was expected to provide immersive gameplay and high-quality graphics but was riddled with bugs, poor performance on older consoles, and lacked promised features (Diaz, 2020).

The discrepancy between the promised product and the actual game led to a severe hit to the company's reputation, the withdrawal of the game from the PlayStation Store, and lawsuits against CD Projekt Red. This case emphasizes the importance of setting and managing realistic expectations in software development.

According to Lehtinen et al. (2014), there are various reasons for software project failures, and no single root cause exists. Instead, a software project failure results from a combination of several factors.

Reference

Anthopoulos, L., Reddick, C.G., Giannakidou, I. and Mavridis, N., 2016. Why e-government projects fail? An analysis of the Healthcare. gov website. Government information quarterly, 33(1), pp.161-173.
Anthopoulos L, Reddick CG

BBC News. (2019, May 9). Boeing admits knowing of 737 Max problem for a year before telling airlines. BBC News. [online] 6 May. Available at: https://www.bbc.com/news/business-48174797

Diaz, J. (2020, December 19). ‘Cyberpunk 2077’ and the Death of the ‘Cool’ Video Game. The New York Times. [online] 6 May. Available at: https://www.nytimes.com/2020/12/19/style/cyberpunk-2077-video-game-disaster.html

Lehtinen, T.O., Mäntylä, M.V., Vanhanen, J., Itkonen, J. and Lassenius, C., 2014. Perceived causes of software project failures–An analysis of their relationships. Information and Software Technology, 56(6), pp.623-643.

Peer review Michael's initial post

Hi Michael, thank you for your insightful post.

Lehtinen's argument resonates well - there is rarely a single, isolated reason leading to project failure. Instead, it is typically a culmination of various mistakes that result in a project's downfall (Lehtinen et al., 2014). For instance, the analysis of the failed Obamacare Website project highlighted multiple reasons for its failure. It was not a single error but a series that led to its downfall (Anthopoulos et al., 2016).

However, how does this series of errors start? Insufficient quality control can result in a low-quality product. However, poorly designed software could inherently have low testability, which might be the natural starting point of the problems.

Humphrey's data illustrates that as the project size increases, the chances of its success decrease. In the paper, Humphrey pointed out that "the hierarchical management style does not work well for managing large software projects", and in further he concluded as "if we want to have successful large-scale software projects, we must develop a project management system that is designed for this purpose." (Humphrey, 2005)

If we agree with his arguments, can we then deduce that the root cause behind many project failures is a mismatch between the management style and the nature of software development? Alternatively, there is no one-size-fits-all solution for all software projects. We might need to adapt our approach based on each project's unique characteristics to guide it appropriately.

Reference

Anthopoulos, L., Reddick, C.G., Giannakidou, I. and Mavridis, N., 2016. Why e-government projects fail? An analysis of the Healthcare. gov website. Government information quarterly, 33(1), pp.161-173.

Humphrey, W.S., 2005. Why big software projects fail: The 12 key questions. The Journal of Defense Software Engineering, 18, pp.25-29.

Lehtinen, A., Mäntylä, V., Vanhanen, J., Itkonen, J. & Lassenius, C. (2014) Perceived causes of software project failures – An analysis of their relationships. Information and Software Technology 56(6): 623–643.

Peer review Andrea’s initial post

Hi Andrea, thanks for your great post.

I noticed a common thread among the three major reasons for project failure you listed: a lack of deep understanding about the project among those involved. These issues often surface at the earliest stages of the project, and it might be too late to rectify them once they are recognized.

This is analogous to software development. Even the most excellent coding skills cannot salvage a poorly designed architecture or flawed business logic.

We could speculate why these issues persist. One could argue that those involved lack the necessary experience to tackle complex projects, or that a consensus has not been established across the team and stakeholders. These points could be valid. However, data suggests that as a project's size increases, its chances of success drastically decrease (Humphrey, 2005). Considering these large projects often belong to major software companies, it's difficult to believe they consistently lack the necessary resources or knowledge.

Andriole (2017) argues that traditional monolithic software, particularly large-scale projects, are almost destined to fail due to their inherent complexity, which can be overwhelming. Moreover, there's a high risk of misunderstandings among those involved.

Reference

Andriole, S.J., 2017. The death of big software. Communications of the ACM, 60(12), pp.29-32.

Humphrey, W.S., 2005. Why big software projects fail: The 12 key questions. The Journal of Defense Software Engineering, 18, pp.25-29.

Summary post

Thank you for the replies.

Lehtinen et al argue there are various reasons that could lead to project failure (Lehtinen, A., et al., 2014). As Michael mentioned, each reason could differently impact a project's failure (Michael, 2023), but no single reason alone causes a project to fail. Instead, failure is usually the accumulated result of multiple factors.

Data shows that as project size increases, the likelihood of success decreases (Humphrey, 2005). This is because large, especially monolithic, software architectures inevitably introduce complexity that is difficult for any one person to fully understand (Andriole, 2017). "Complexity influences project planning and control; it can hinder the clear identification of goals and objectives, it can affect the selection of an appropriate project organization form, or it can even affect project outcomes."(San Cristóbal, Diaz, Carral, Fraguela, and Iglesias 2019). When people try to fully plan a complex project, predictability is low. We should doubt how accurate the plan really is, especially if using a waterfall approach, since correcting errors can be cost-intensive (Thesing, Feldmann & Burchardt, 2021).

In contrast, agile project management planning is incremental, continuous, step-by-step, flexible, short-term in detail, and based on a long-term vision(Thesing, Feldmann & Burchardt, 2021). This reduces complexity and improves predictability, enabling a more accurate plan. If we look at many successful software or systems, the common pattern behind them is that the software or system started basic and evolved multiple times to become what we see today. However, many large projects tend to generate a full picture and detailed plan at the beginning. If the initial direction of a project is wrong, the following stages will be disastrous.

Similar trends appear in software architecture and technology, e.g., programming languages evolving from machine code to human-readable natural language, and architecture moving from monolithic to microservices (Andriole, 2017). The goal in all cases is reducing complexity and improving understanding.

However, the adoption of agile approaches in safety-critical systems is still not widespread, due to the perception that agile methods may not suit their development (Kasauli et al., 2018) and the tight regulations on them, which can constrain agile methods (Islam & Storer, 2020).

In summary, highly complex projects involve a lot of uncertainty. When people plan and organize activities, resources, and risk assessments based on limited understanding, the results can be distorted, putting the whole project at high risk. Case studies show that using appropriate ways to reduce complexity and improve predictability on a project could help address this problem.

Reference

Andriole, S.J., 2017. The death of big software. Communications of the ACM, 60(12), pp.29-32.

Humphrey, W.S., 2005. Why big software projects fail: The 12 key questions. The Journal of

Defense Software Engineering, 18, pp.25-29.

Islam, G. and Storer, T., 2020. A case study of agile software development for safety-Critical systems projects. Reliability Engineering & System Safety, 200, p.106954.

Kasauli, R., Knauss, E., Kanagwa, B., Nilsson, A. and Calikli, G., 2018, August. Safety-critical systems and agile development: A mapping study. In 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp. 470-477). IEEE.

Lehtinen, A., Mäntylä, V., Vanhanen, J., Itkonen, J. & Lassenius, C. (2014) Perceived causes of software project failures – An analysis of their relationships. Information and Software Technology 56(6): 623–643.

Michael, B. (2023). In reply to Wang Wang Re: Initial Post. [online] University of Essex Online. Available at: https://www.my-course.co.uk/mod/forum/discuss.php?d=156907 [Accessed 29 May 2023].

San Cristóbal, J.R., Diaz, E., Carral, L., Fraguela, J.A. and Iglesias, G., 2019. Complexity and project management: Challenges, opportunities, and future research. Complexity, 2019.

Thesing, T., Feldmann, C. and Burchardt, M., 2021. Agile versus waterfall project management: decision model for selecting the appropriate approach to a project. Procedia Computer Science, 181, pp.746-756