**Research Proposal on Mitigating Risks in Software Development using LLMs**

Zorawar Jaiswal

Student ID: 203212561

California State University, Northridge

COMP 680

Prof. Harut Sepetjyan

Jan 28, 2025

**Abstract**

Software projects are never free of risks. The severity of risks may have dire consequences such as leading to a loss of life in case of medical software and airplane crashes, while on the other hand, it may lead to a loss of business as in the case of website downtime. Thus, identifying and mitigating risks in software in a preemptive manner is crucial for all domains. This work proposes on exploring a system that identifies software risks at each step of the Software Development Life Cycle by passing all commits, user stories, comments and memos to an LLM with background information such as team size, budget and scope of requirements to identify and estimate gaps in the system. Not only will this methodology boost risk management, it will also help in the planning phase and assist in vulnerability assessment.

**Introduction**

Events and activities that can have unwarranted repercussions in a software environment are known as software risks (Khanna et al., 2022). A risk-free software system is an ideal software system because software with risks can be life-threatening.

A very common example of risky software is the Ethiopian Airlines crash in 2019. One of the sensors failed in the system, which sent noisy data to the control system, which led to the undesired activation of the maneuvering system, leading to the loss of life of nearly 190 people (Agence France-Presse, 2022).

Risky software can also lead to huge financial losses, as seen in the case of the Y2K incident. The Y2K incident was expected to affect the airline and banks since the transition of date to a different century (1900s to 2000s). The systems were expected to crash since two-digit dates (year 1998 referred as 98) would have turned to 00 (Smithsonian, 2000).

Software risks can be mitigated through a variety of strategies. The management of software risks identifies, assesses, and mitigates potential risks through effective control; this minimizes the probability of failures, reducing losses due to unexpected events or known vulnerabilities.

Another latest example of the same is the Netflix outage caused due to the boxing match between Mike Tyson and Jake Paul. Netflix had a surge of load on its system leading to down time for most of the USA (Yoon, 2024).

This main focus of this paper is to analyze the various causes of development risks for a software system and identifying practices to mitigate the same through the use of tools, practices, and processes to ensure high quality and reliable systems.

**Literature Review**

Khanna et al., in their work identified the various causes of risks in a Distributed Agile Development environment. They majorly call out the requirement elicitation phase for a majority of the risks with issues such as ambiguous and contradicting requirements, and multiple product owners leading to confusion as the driving force of software risks. They propose introducing paired programming as one of the solutions for mitigating risks.

Haider et al., in their work focused on the open-source software systems and identified the risks for faults. The consensus came out to blame bugs in the source code and incomplete documentation since open-source codes may not undergo code reviews and beginners may make erroneous changes causing risks. They suggest to incorporate tools such as Jira and to improve communication within teams.

Khurana & Wassay, focused on the security risks in a software environment. They argue that security is the primary driver of risks in a software system. The inference is that security risks may be invoked due to fast evolving technology while the cybersecurity guidelines take a while to improve. The proposed solution is to have multiple standup calls to review updates.

Darandale & Mehta worked on the NASA MDP dataset and utilized various Machine Learning algorithms to (i) classify the risks & (ii) evaluate the risks. They concluded that Naïve Bayesian Classifier was the best performing model to estimate and evaluate the risks. The MDP dataset has a lot of features such as lines of code, complexity of code, effort, number of bugs, etc.

Pilliang et al., proposed novel work on the Stackoverflow 60K dataset from Kaggle by implementing sentiment analysis and K Means Clustering. They SO 60K dataset is a collection of user posts from stackoverflow. They used the user questions as sentiment analysis, ranked the sentiment analysis and estimated the likelihood of occurrence. Using this, they computed a risk matrix comprising of the severity and occurrence. This is the only work that was done on purely textual data.

Even though the recent research is adept in nature, there are still a lot of gaps that need to be addressed. Only one system identifies the risks in terms of texts and that too is based on sentiment likelihood. There is no present set of practices that addresses a dynamic need to risk assessment. Risks need to be assessed at the end of each step and this process needs to be natively present in the software design life cycle.

This study will explore the feasibility of a system that addresses risks dynamically through each stage of the system. Passing each commit statement, minutes of each meeting, and story board backlogs into a LLM would help identify the risks in term of decision making and development.

# **Methodology**

This study proposes an LLM-based approach to identify and assess software risks dynamically at each stage of the SDLC. The main stages involved would be data collection, model training, risk assessment, and validation.

For data collection, it is ideal to pull up commit messages from an open sourced github project and to gather case study reports of failed software systems (victims of risk systems). Sample user stories and Kanban boards should be picked up to ensure that risks are identified during the phase of communication as well.

Next up, an LLM agent would be created to assess the project specifics such as manpower, budget, requirements, code, and communication. It will help identify the gaps and risks in the project.

For validation, human validation and comparison with the existing datasets would be a good way. There are various metrics such as Root Mean Square Error that can be explored.

# **References**

Agence France-Presse. (2022, December 23). *Inquiry into 2019 Ethiopian Air Crash Confirms Software Failure*. Barrons. https://www.barrons.com/news/inquiry-into-2019-ethiopian-air-crash-confirms-software-failure-01671821708

Darandale, S., & Mehta, R. (2022). Risk assessment and management using machine learning approaches. *2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, 663–667. https://doi.org/10.1109/icaaic53929.2022.9792870

Haider, S., Khalil, W., Al-Shamayleh, A. S., Akhunzada, A., & Gani, A. (2023). Risk factors and practices for the development of open source software from developers’ perspective. *IEEE Access*, *11*, 63333–63350. https://doi.org/10.1109/access.2023.3267048

Khanna, E., Popli, R., & Chauhan, N. (2022). Identification and classification of risk factors in Distributed Agile Software Development. *Journal of Web Engineering*, *21*(6), 1831–1851. https://doi.org/10.13052/jwe1540-9589.2164

Khurana, S. K., & Wassay, M. A. (2023). Towards challenges faced in agile risk management practices. *2023 International Conference on Inventive Computation Technologies (ICICT)*, 937–942. https://doi.org/10.1109/icict57646.2023.10134188

Pilliang, M., Munawar, Hadi, M. A., Firmansyah, G., & Tjahjono, B. (2022). Predicting risk matrix in software development projects using bert and K-means. *2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 137–142. https://doi.org/10.23919/eecsi56542.2022.9946637

Smithsonian. (2000). *Y2K*. National Museum of American History. https://americanhistory.si.edu/collections/object-groups/y2k

Yoon, J. (2024, November 16). *Thousands report netflix livestream crashes during Mike Tyson-Jake Paul Fight*. The New York Times. https://www.nytimes.com/2024/11/16/business/media/netflix-outage-crash-boxing.html#:~:text=Users%20across%20the%20U.S.%20reported,the%20high%2Dprofile%20boxing%20match.&text=Tens%20of%20thousands%20of%20Netflix,livestream%20was%20failing%20to%20load.