AI-Assisted Domain Modeling: Enhanced Bounded Context Extraction with LLMs

Bachelor Thesis

Organization Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Tobias Eisenreich
Student:	Husein Jusic
Timeframe:	

1 Introduction and Motivation

Software architecture design is a critical and challenging phase in the software development life cycle, particularly within larger SaaS Companies where systems are complex and must support extensive scalability requirements. As highlighted by Eisenreich et al. [1] designing domain models and software architectures is not only time-consuming but also significantly impacts the quality of service delivered by the resulting system.
In practice, the architecture design process in enterprise environments is constrained by tight deadlines, limited resources and business pressure which often leads software architects to pick suboptimal solutions: either taking the first viable architecture design without deep exploration of alternatives or creating very simple architectures that satisfy the immediate requirements but without considering long-term quality attributes. This stands in contrast to the ideal approach where multiple architecture candidates would be created, thoroughly evaluated and compared before settling for the most suitable solution.
The consequences of hastily constructed software architectures are well-documented in software engineering literature. Suboptimal architectures for example can lead to increased maintenance cost as analyzed by MacCormack et al. [2]. Specifically for SaaS Companies these consequences can translate to competitive disadvantages as their business model depends on maintaining a robust software foundation.
The recent quality advancements of LLMs present promising opportunities to address these challenges. Eisenreich et al. [1] have proposed a vision for semi-automatically generating software architectures using artificial intelligence techniques, particularly LLMs, based on software requirements. Their approach suggests leveraging AI to generate domain models and multiple architecture candidates, followed by a manual evaluation and trade-off analysis of the created architectures.
While their vision provides a valuable conceptual framework, its application specifically in large-scale SaaS software environments with lots of requirements remains unexplored.

This thesis aims to extend Eisenreich's vision by investigating how different LLMs can be utilized specifically in the context of large Saas Software. We will conduct an empirical study with a SaaS Company - FTAPI Software GmbH. By focusing specifically on the domains of large Saas Software and conducting research within an actual enterprise environment, this thesis aims to provide insights into the applicability of AI-Assisted architecture design and the specific considerations required when applying these techniques in larger-scale software development contexts.

1.1 Study Context: FTAPI Software GmbH

Founded in 2010, FTAPI Software GmbH has consistently pursued a clear vision: enabling organizations to maintain complete control over their data exchange—enhancing efficiency, security, and digital sovereignty. Today, approximately 2,000 companies and more than a million active users rely on FTAPI's platform for secure data exchange. The company's core software has evolved over the years to accommodate numerous requirements. Currently, a substantial portion of the service consists of a large monolithic structure that has become increasingly difficult to maintain. To ensure future readiness and sustain a reliable, robust software foundation, the development team is continuously working to transform this monolith into a more modular architecture (modulith). To accomplish this transformation, developers—alongside their regular tasks of implementing new features and fixing bugs—decompose individual parts of the software where possible into separate bounded contexts using domain-driven-design (DDD) as presented by Vernon [3, p.62]. Each bounded context consists of a single module designed to accomplish one clearly defined task. These modules, once separated from each other, become significantly easier to maintain. Through this thesis, we aim to investigate how Large Language Models (LLMs) can be effectively utilized to accelerate and improve FTAPI's modularization process. Specifically, we will explore how LLMs can assist software architects in identifying potential bounded contexts, defining domain models within those contexts, and establishing appropriate interfaces between them. This assistance has the potential to significantly advance FTAPI's architectural evolution toward a more maintainable, modular system based on sound domain-driven principles.

2 Research Question and Objectives

This thesis aims to explore and analyze how LLMs can be utilized in the industry with large requirement sets to help developers with creating and refining software architectures

- How effectively can Large Language Models (LLMs) identify and define viable bounded contexts that align with domain-specific requirements in large SaaS applications implementing Domain-Driven Design principles?

- To what extent do bounded contexts and domain models identified by LLMs compare in quality and applicability with those created by experienced DDD practitioners when analyzing complex SaaS application requirements?
- What modifications to Eisenreich's methodology are necessary when applying it to large-scale legacy systems undergoing incremental modularization?

3 Methodology

This research will employ a mixed-methods approach combining qualitative and quantitative techniques to evaluate the effectiveness of LLM-assisted software architecture design at a SaaS Company like FTAPI Software GmbH.

3.1 Phase 1: Baseline Assessment

First, we need to understand how FTAPI currently identifies and designs bounded contexts within their domain-driven approach. We will interview 4-5 software architects to learn about their current design process. We will observe how the software developers and domain experts identify and define bounded contexts for parts of the software monolith. To supplement these observations, we will analyze existing documentation of previously identified bounded contexts to understand FTAPI's domain model process.

3.2 Phase 2: LLM Setup and Prompt Engineering

In the second phase, we will prepare the AI tools and the inputs they need. We will select several modern LLMs to test, such as GPT-4, Claude 3, and LLaMA-3. We will gather requirements and documentation for 1-2 subdomains FTAPI wants to separate from their monolith. We will also collect information about one bounded context that has already been separated. Using this information, we will create specialized prompts for each LLM based on FTAPI's requirements and design guidelines.

3.3 Phase 3: Architecture Generation and Comparative Study

In this phase, we will adapt Eisenreich et al.'s semi-automatic architecture generation approach to focus specifically on bounded context identification and domain model generation. Using the prompts developed in Phase 2, we will task the selected LLMs with analyzing requirements and proposing bounded contexts along with their internal domain models and context maps showing relationships between contexts. These LLM-generated domain models will then be evaluated through structured interviews with professional developers and experienced DDD practitioners at FTAPI. The evaluation will focus on the correctness, completeness, and applicability of the proposed bounded contexts.

References

[1] Tobias Eisenreich, Sandro Speth, and Stefan Wagner. "From Requirements to Architecture: An AI-Based Journey to Semi-Automatically Generate Software Architectures". In: *Proceedings of the 1st International Workshop on Designing Software*. Designing '24. Lisbon, Portugal: Association for Computing Machinery, 2024, pp. 52–55. ISBN: 9798400705632. DOI: 10.1145/3643660.3643942. URL: https://doi.org/10.1145/3643660.3643942.

[2] Alan MacCormack and Daniel J. Sturtevant. "Technical debt and system architecture: The impact of coupling on defect-related activity". In: *Journal of Systems and Software* 120 (2016), pp. 170–182. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2016.06.007. URL: https://www.sciencedirect.com/science/article/pii/S0164121216300760.

[3] Vaughn Vernon. *Implementing domain-driven design*. Addison-Wesley, 2013.