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

A Practical Exploration with FTAPI Software GmbH

Husein Jusic

Supervisor: Tobias Eisenreich
Chair of Software Engineering
Technical University Munich

husein.jusic@tum.de
h.jusic@ftapi.com

June 18, 2025

Who am I

- **Name:** Husein Jusic – Bachelor's Informatics @ TUM
- **Current Role:** Working student @ FTAPI ($\tilde{4}$ Years)

FTAPI

FTAPI Software GmbH Logo

- FTAPI started in 2010 with the idea: *"One platform to secure all business data exchange."*
- Millions of users now rely on FTAPI.
- But like many companies, the architecture... did not keep up.

The Harsh Reality: A Big Ball of Mud

- **Platform:** Secutransfer
- Monolithic codebase, entangled logic
- Hard to scale, harder to understand
- Domain knowledge scattered over lots of Services, Managers, ...

The Harsh Reality: A Big Ball of Mud

Antipattern:

Image generated with *ChatGPT + Sora (OpenAI)*

The Escape Plan: From Monolith to Modulith

- Gradual transformation
- Strategy: Domain-Driven Design (DDD)
- Goal: Code with clear boundaries and clear responsabilities

Spoiler: This is Really, Really Hard

- Domain knowledge lives in people's heads
- Legacy logic spans multiple domains
- No consistent language
- Time pressure from business side

Enter the AI Architect: LLMs in Software Design

- What if AI could help identify bounded contexts?
- Extract domain models from large requirement sets?
- Help engineers designing software architecutres?
- Speed up modularization efforts?

What the Research Says

- Fully automated generation of domain models?
  $\rightarrow$ **Doesn't work well.**
- Models are incomplete, inconsistent [ADD REFERENCE]
- **Why?** LLMs lack domain context and architectural intuition (WIP: explain no clear path to follow??)

Not Replacing Engineers – Augmenting Them

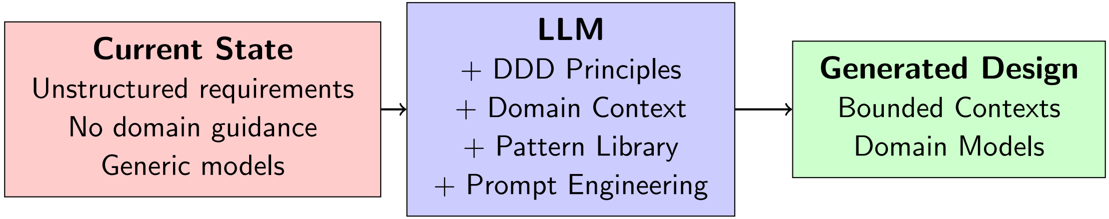
Semi-Automated Domain Modeling:

- Engineers + LLMs = **much better results**
- LLMs help with:
  - wip: add study results [ADD REFERENCE]

Industry Context

- Few studies analyze real-world, industrial use cases for AI-based domain model generation.
- Most research is based on synthetic or simplified examples, which limits practical applicability.
- **My Hypothesis:** Integrating Domain-Driven Design (DDD) into AI-assisted model generation could improve architectural quality.
- **Why?** DDD provides a structured and context-aware foundation — giving AI a clear "path" for modeling that current approaches lack.

From Chaos to Clarity: AI-Guided DDD

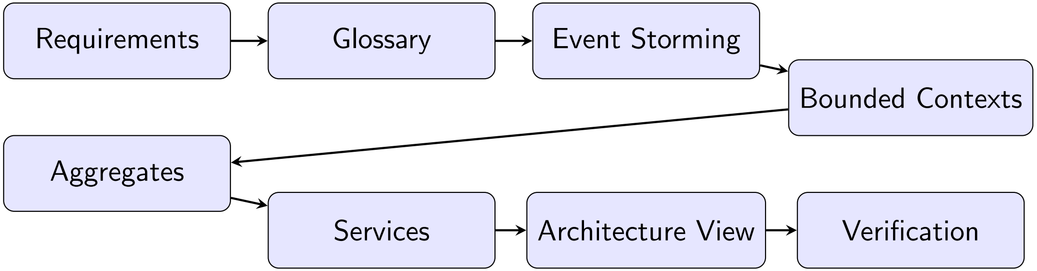


"Don't just throw requirements at AI – teach it to think like a domain expert."

In Progress: Preparation

- **Prompt Engineering:** Designed and refined a diverse set of prompts to guide the LLM effectively through different DDD stages.
- **Requirements:** Collected and analyzed detailed requirements from the platform.
- **Workflow:** Initiated the implementation of a Spring Boot–based AI application to orchestrate the domain modeling pipeline.

In Progress: Defining a Workflow to Create Architecture Candidates



"LLM supports the entire domain modeling pipeline – from raw text to architecture candidates."

TBD: Next Steps

- **Generate** – Derive bounded context candidates from collected requirements.
- **Evaluate** – Assess both the generation process and the resulting artifacts.
- **Verify** – Validate outcomes through interviews with experienced Domain-Driven Design practitioners.

The Goal

Can AI help to escape from the monolith faster and better? Research Questions:

- How effectively can Large Language Models (LLMs) identify and define viable bounded contexts that align with complex domain-specific requirements?
- 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 application requirements?

Questions?

Any questions or thoughts?

I'm happy to discuss!

Backup: Domain-Driven Design (DDD)

- **What is DDD?**
  A software design approach focusing on modeling complex domains based on collaboration between technical and domain experts.
- **Key Concepts:**
  - Ubiquitous Language
  - Bounded Contexts
  - Entities and Value Objects
  - Aggregates and Repositories
- **Why DDD?**
  Helps create clear, maintainable architectures aligned with real business needs.
- **Relation to AI:**
  Provides structure and domain knowledge that can guide AI in generating better, context-aware models.

References I

Eric Evans,
Domain-Driven Design: Tackling Complexity in the Heart of Software,
Addison-Wesley, 2003.