DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

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

Husein Jusic

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

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

KI-unterstützte Domänenmodellierung: Verbesserte Extraktion von Bounded Contexts mit LLMs

Author:	Husein Jusic
Supervisor:	Tobias Eisenreich
Advisor:	Michael Krinninger
Submission Date:	August 31, 2025

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, August 31, 2025 Husein Jusic

Acknowledgments

Abstract

Contents

1. Introduction and Overview

1.1. Motivation

Software architecture design is a critical and challenging phase in the software development life cycle, particularly within larger Companies where systems are complex and must support extensive scalability requirements. As highlighted by Eisenreich et al. [ESW24] 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 idealized 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. [MS16]. 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. [ESW24] 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 software environments with lots of requirements remains unexplored.

1.2. Outlook

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 larger 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.

1.3. 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 with large requirements sets

- 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?

1.4. Structure

This thesis is organized into eight chapters structured as follows:

Chapter 1 introduces the research problem, presents the motivation for this work, and outlines the main contributions.

Chapter 2 provides the theoretical foundations necessary to understand the proposed approach.

Chapter 3 discusses related research and examines existing solutions to similar problems.

Chapter 4 introduces the collaborating company and defines the specific business problem addressed in this thesis.

Chapter 5 describes the research methodology and approach taken to solve the identified problem.

Chapter 6 details the technical realization and implementation of the proposed solution.

Chapter 7 presents the experimental results and findings in an objective manner.

Chapter 8 critically analyzes these results, discusses their implications, and addresses the limitations of the proposed approach.

2. Theoretical Background

In this chapter, we introduce the theoretical concepts fundamental to this thesis. We begin with Domain-Driven Design, which provides the architectural framework for our investigation into LLM-assisted bounded context extraction.

2.1. Domain Driven Design

Domain Driven Design (DDD) describes a process for software development which was introduced by Eric Evans in his seminal work "Domain-Driven Design: Tackling Complexity in the Heart of Software" [Eva04]. This methodology emphasizes creating software systems that accurately reflect and align with the business domain they serve. DDD is particularly valuable for complex systems with extensive requirements where business logic is continually evolving and changing.

The core philosophy of DDD centers on prioritizing the domain model over technical concerns, enabling software development teams to solve business problems instead of getting entangled in implementation details. This approach typically results in software that is more maintainable and closely aligned with business objectives. Empirical research supports this claim; for example, Özkan et al. [ÖBB23] conducted a case study demonstrating that DDD implementation significantly improved the maintainability metrics of a large-scale commercial software system compared to its previous architecture.

2.1.1. Domain

Evans provides a foundational definition of the term "domain" in his seminal work:

> "Every software program relates to some activity or interest of its user. That subject area to which the user applies the program is the domain of the

software." [Eva04, p. 4]

Further he makes it clear that the domain represents more than just a subject area; it encompasses the entire business context within which a software system operates. It includes all the business rules, processes, workflows, terminology, and conceptual models that domain experts use when discussing and working within their field of expertise. Vernon [Ver13, p. 17] further clarifies this by explaining that a domain is "a sphere of knowledge and activity around which the application logic revolves."

2.1.2. Ubiquitous Language

One of the core concepts of DDD is the development of a Ubiquitous Language. A Ubiquitous Language is a shared vocabulary which is consistently used by domain experts and the developers. This shared vocabulary improves communication, mitigates translation errors and improves communication between technical and non-technical stakeholders when discussing the business domain.

Vernon [Ver13, p. 22] provides an example on how Ubiquitous Language directly affects code design. He presents three approaches to modeling a flu vaccination scenario, each reflecting a different level of domain understanding.

His example shown in Table 2.1 demonstrates how the evolution of language directly impacts code structure and domain modeling. The progression from generic, technical language to precise, domain-aligned terminology, illustrates a fundamental principle: the language we use shapes the software we build.

Ubiquitous Language plays a crucial role in identifying and maintaining bounded contexts. Evans [Eva03, p. 13] emphasizes that "the model is the backbone of a language used by all team members", and this language serves as the primary indicator of context boundaries. When the same term carries different meanings or when communication requires translation between team members, these linguistic fractures often reveal the natural boundaries between bounded contexts.

Within a well-defined bounded context, every term has a single, precise meaning that all team members, both technical and domain experts, understand identically. This linguistic consistency prevents the subtle corruption of domain concepts that often occurs when boundaries are unclear. For instance, the term "customer" might mean a person with an active subscription in a billing context, while in a marketing context it could include prospects and past customers. These semantic differences signal the

need for separate bounded contexts.

Domain Statement	Resulting Code

"Who cares? Just code it up."	`patient.setShotType(ShotTypes.TYPE_FLU);`
	`patient.setDose(dose);`
	`patient.setNurse(nurse);`

"We give flu shots to patients."	`patient.giveFluShot();`
"Nurses administer flu vaccines to patients in standard doses."	`Vaccine vaccine = vaccines.standardAdultFluDose();` `nurse.administerFluVaccine(patient, vaccine);`

Table 2.1.: Approaches to modeling based on different language interpretations (adapted from Vernon [Ver13, p. 22])

Relevance to LLM-Assisted Domain Modeling

This linguistic foundation presents both opportunities and challenges for LLM-assisted bounded context identification. Large Language Models possess the capability to detect semantic variations and linguistic patterns. They can potentially:

1. **Identify terminology conflicts**: LLMs can analyze requirements documents to detect when the same term is used with different meanings, suggesting potential context boundaries.

2. **Extract domain vocabulary**: By processing stakeholder communications, user stories, and documentation, LLMs can help build a comprehensive glossary of domain terms and their relationships.

3. **Maintain linguistic consistency**: LLMs can assist in ensuring that domain terms are used consistently within a bounded context and flag instances where termi-

nology diverges from established patterns.

However, the challenge lies in ensuring that LLMs understand the domain-specific nuances rather than applying generic interpretations. The success of LLM-assisted bounded context identification may largely depend on how effectively we can guide these models to recognize and respect the precision that Ubiquitous Language demands. This consideration will be central to our prompt engineering approach and evaluation criteria in the empirical phase of this research.

2.1.3. Subdomains and Bounded Contexts

Understanding the distinction and relationship between subdomains and bounded contexts is fundamental to Domain-Driven Design and crucial for this thesis, as bounded contexts form the primary unit of modularization in our investigation.

Defining Subdomains and Bounded Contexts

A subdomain represents a distinct area of the business domain, corresponding to different aspects of the organization's activities. Evans [Eva03] identifies three types of subdomains: the *Core Domain*, which provides competitive advantage; *Supporting Subdomains*, which are necessary but not differentiating; and *Generic Subdomains*, which address common problems faced by many businesses.

In contrast, a bounded context exists in the solution space as the delimited applicability of a particular domain model [Eva03]. It establishes explicit boundaries within which a domain model remains consistent and unified. These boundaries encompass linguistic aspects (consistent ubiquitous language), organizational aspects (team alignment), and technical aspects (code bases, schemas, deployment units).

The relationship between subdomains and bounded contexts is not necessarily one-to-one. While ideally each subdomain maps to a single bounded context, practical constraints often lead to different arrangements where a bounded context might span multiple subdomains or a subdomain might be split across multiple contexts [Eva03].

The Relationship Between Subdomains and Bounded Contexts

While subdomains and bounded contexts often align, Evans [Eva03] warns against assuming a one-to-one correspondence. In practice, legacy constraints, team structures, and technical limitations influence how subdomain boundaries map to bounded contexts. A single bounded context might encompass multiple subdomains, or a subdomain might span multiple contexts. This distinction proves critical for modularization efforts like those at FTAPI: subdomain identification provides business-driven boundaries, while bounded context design translates these into implementable software modules. As Evans notes, "when code based on distinct models is combined, software becomes buggy, unreliable, and difficult to understand" [Eva03, p. 271]—making bounded contexts the fundamental unit for maintaining model integrity during modularization.

2.2. Large Language Models

This section presents an analysis of Large Language Models, covering their technological evolution, fundamental capabilities, and practical applications in software engineering. We pay special attention to both the opportunities they offer and the limitations that must be addressed when leveraging these models for domain-driven design tasks, particularly bounded context identification.

2.2.1. Prompt Engineering

Prompt engineering has emerged as a critical discipline for optimizing interactions with Large Language Models. As Aqsa et al. [AAS25] define it, prompt engineering involves "the strategic arrangement of input queries under prompt engineering methodology [which] leads to enhanced LLM output efficiency and accuracy as well as improved coherence." This field focuses on crafting structured inputs that guide models to produce accurate, contextually relevant, and task-appropriate outputs.

Core Techniques and Approaches

The effectiveness of prompt engineering relies on several established techniques. Structured prompting involves carefully designed prompts that specify roles, contexts, and constraints. Role-based prompting, for instance, instructs the model to respond from a specific perspective (e.g., "Act as a cybersecurity expert"), which enhances domain-specific precision by aligning responses with expert knowledge patterns [AAS25]. Iterative refinement allows continuous improvement of prompts based on previous model outputs, while chain-of-thought prompting guides models through systematic reasoning processes—particularly valuable for complex problem-solving tasks.

Emerging Trends and Rapid Evolution

The field of prompt engineering is evolving at an unprecedented pace. Recent developments include automated prompt generation using reinforcement learning from human feedback (RLHF), multi-modal prompting that combines text with visual inputs, and collaborative human-AI systems for prompt optimization [AAS25]. These advancements are rapidly transforming how practitioners interact with LLMs across various domains.

However, this rapid evolution presents a unique challenge for researchers and practitioners. Traditional academic literature, with its lengthy peer-review cycles, often becomes outdated by the time of publication. The techniques and best practices that were state-of-the-art six months ago may already be superseded by new approaches. This temporal mismatch between the pace of development and academic publishing means that practitioners increasingly rely on alternative sources of information—including technical blogs, preprint servers, open-source repositories, and community forums—to stay current with the latest prompt engineering strategies.

Relevance to Domain Modeling

For the specific task of bounded context identification, prompt engineering becomes particularly crucial. The quality of prompts directly influences whether an LLM can accurately understand domain-specific terminology, identify linguistic boundaries between contexts, and propose meaningful architectural divisions. Effective prompts must guide the model to:

- Recognize domain-specific vocabulary and its context-dependent meanings
- Identify patterns that suggest natural boundaries between business capabilities
- Maintain consistency with established domain-driven design principles
- Produce outputs that are both technically sound and business-aligned

The challenge lies in developing prompts that can effectively communicate the nuanced requirements of domain modeling while accounting for the model's inherent limitations in understanding implicit domain knowledge. This balance between guidance and flexibility forms a central consideration in our empirical investigation of LLM-assisted bounded context identification.

3. Related Work

Chaper about finding related work, any things that can be derived and are interesting to this thesis etc.

3.1. Automated Domain Model Generation

Domain modeling represents a time-intensive and expertise-dependent aspect of software engineering that requires deep understanding of both business requirements and technical constraints. In this process, engineers typically convert textual requirements into domain models that accurately represent the problem space and provide a foundation for solving business challenges. The inherent complexity and substantial resource demands of manual domain modeling have motivated researchers to explore automation approaches that could reduce both time investment and dependency on scarce domain expertise. Studies such as Chen et al. [Che+23] and from Saini et al. [Sai+22] explore the capabilities of Large Language Models to assist during this phase, while simultaneously highlighting the current limitations these models face in fully capturing domain semantics and business logic.

3.1.1. Fully Automated Domain Modeling Approaches

Chen et al. [Che+23] conducted a comprehensive comparative study using GPT-3.5 and GPT-4 for fully automated domain modeling. Their findings reveal that while LLMs demonstrate impressive domain understanding capabilities, they remain impractical for full automation. Significantly, their research highlighted that LLM-generated domain models exhibit high precision but low recall, meaning that while the generated elements are often correct, many required domain elements are missing from the output. Furthermore, Chen et al. found that LLMs struggle most with identifying relationships between domain concepts compared to classes and attributes, and rarely incorporate

established modeling best practices or complex design patterns.

3.1.2. Semi-Automated Interactive Approaches

In contrast to fully automated approaches, Saini et al. [Sai+22] propose a bot-assisted interactive approach that addresses the need for human expertise in domain modeling. Their work recognizes that domain modeling decisions require contextual knowledge and personal preferences that vary from engineers. Rather than attempting full automation, their approach generates multiple alternative solutions for domain modeling scenarios and learns from user preferences over time through an incremental learning strategy. Saini et al.'s work provides traceability between requirements and generated models, enabling users to understand and validate the AI's modeling decisions. This addresses a critical concern for enterprise adoption where architectural decisions must be explainable. Their approach specifically handles complex domain modeling patterns, which are also relevant to bounded context identification in Domain-Driven Design.

3.1.3. Implications for Bounded Context Identification

The findings from these studies have significant implications for bounded context identification in Domain-Driven Design. The research collectively demonstrates that while LLMs show promise in automated design generation, human expertise and interaction are essential for achieving practical, high-quality results in complex software architecture and modeling tasks. The semi-automatic approaches that combine AI assistance with human expertise appear more effective than fully automated solutions, particularly for enterprise environments where architectural decisions must be both accurate and explainable.

These insights support the approach taken in this thesis, which focuses on semi-automated bounded context identification that leverages LLM capabilities while maintaining human control and validation throughout the process. The evidence suggests that such hybrid approaches are more likely to succeed in real-world enterprise environments like FTAPI's modularization efforts, where domain expertise and contextual knowledge are critical for successful architectural transformations.

3.2. Monolith Decomposition

The evolution from monolithic to modular architectures represents one of the most significant paradigm shifts in software engineering over the past two decades. Understanding this evolution is crucial for appreciating both the challenges and opportunities in modern system decomposition approaches.

3.2.1. The Evolution from Monolith to Modular Architectures

Monolithic architectures emerged as the dominant pattern in enterprise software development during the 1990s and early 2000s. During this period, enterprise software was primarily deployed as single, large applications that contained all business logic, data access, and user interface components within a unified codebase. Fowler [Fow02] documented how enterprise applications naturally evolved into monolithic structures due to the technological constraints and development practices of this era.

Over the years, numerous studies have highlighted the constraints and limitations of monolithic architectures as systems evolve and scale. Research has consistently demonstrated that as application size and complexity increase, significant architectural challenges emerge that impact both development efficiency and system maintainability.

Blinowski et al. [BOP22] empirically demonstrated several critical bottlenecks inherent in monolithic systems. Their study revealed that as monolithic applications grow, "modifying the application's source becomes harder as more and more complex code starts to behave in unexpected ways." The research highlighted how architectural boundaries deteriorate over time, with developers finding it "increasingly harder to keep changes that related to a particular module to only affect this very module." This boundary erosion leads to a cascade effect where "changes in one module may lead to unexpected behavior in other modules and a cascade of errors."

The industry began exploring alternative approaches that could address the scalability, maintainability, and development velocity issues inherent in monolithic systems while preserving some of their operational simplicity. The emergence of modular architectures took several evolutionary paths. Richardson [Ric19] identified the modular monolith (modulith) as a pragmatic intermediate approach that maintains the deployment simplicity of monoliths while establishing clear module boundaries and enforcing architectural constraints. This approach allows organizations to achieve better separation of concerns and improved maintainability without the operational

complexity of fully distributed systems.

The challenge of identifying optimal module boundaries has led to increased interest in Domain-Driven Design principles, particularly the concept of bounded contexts, as a systematic approach to decomposing monolithic systems. Evans' [Eva04] strategic design patterns provide a framework for identifying natural business boundaries that can serve as the foundation for modular architectures.

3.3. Research Gap

Despite the growing body of research in both automated domain modeling and monolith decomposition, there exists a notable gap at the intersection of AI-assisted architecture generation and Domain-Driven Design methodologies. While studies like Chen et al. [Che+23] and Saini et al. [Sai+22] have explored LLM capabilities for domain model generation, and numerous works have addressed monolith decomposition strategies, the specific application of AI assistance to DDD's bounded context identification remains largely unexplored.

Current research in automated domain modeling primarily focuses on generating UML diagrams or class structures from requirements, without explicitly considering the strategic patterns and bounded context principles fundamental to DDD. Similarly, existing monolith decomposition approaches often rely on technical metrics such as coupling and cohesion, or manual expert analysis, rather than leveraging the semantic understanding capabilities of modern LLMs to identify domain boundaries that align with business capabilities.

The research gap becomes particularly evident when considering that DDD provides a well-defined framework with clear architectural patterns and concepts—such as bounded contexts, aggregates, and ubiquitous language—that could serve as structured targets for AI-assisted generation. This structured nature of DDD makes it potentially more amenable to AI assistance than general domain modeling, yet this synergy remains underexploited in current literature.

This thesis addresses this research gap by proposing a semi-automated approach that specifically leverages LLMs to assist in bounded context identification within the DDD framework. By combining the semantic understanding capabilities of AI with the structured patterns of DDD, this work aims to provide a practical solution for organizations seeking to modernize their monolithic architectures while maintaining

alignment between technical boundaries and business domains.

4. 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.

4.1. Company and Product Overview

FTAPI[1] has created a data exchange platform designed to address the growing need for secure and compliant data transfer in modern organisations. The main platform is called Secutransfer and it serves as an integrated solution for exchanging sensitive and business-critical data across organizational boundaries while maintaining strict security and legal compliance standards, such as GDPR, NIS-2, and TISAX®

4.1.1. Core Platform Components

The platform consists of four interconnected products that can be used individually or as an integrated suite:

SecuMails - Email Encryption

SecuMails enables secure encrypted email communication without requiring complex infrastructure. The solution operates through web browsers or via an Outlook add-in, supporting file transfers up to 100 GB. This addresses common limitations of traditional email systems while ensuring compliance with common legal regulations

[1] FTA25.

SecuRooms - Virtual Data Rooms

SecuRooms provides secure virtual spaces for collaborative data exchange. Files can be uploaded by users in virtual datarooms. The users can use this virtual dataroom either as cloud storage or invite other users to share the files.

SecuFlows - Automated Workflows

SecuFlows is a standalone application that enables organizations to model and execute automated data exchange workflows. The solution allows users to define complex multi-step processes for data handling, including automated routing, approval workflows, and compliance checks, streamlining repetitive data exchange operations while maintaining security standards.

SecuForms - Secure Forms

SecuForms provides a secure web-based form creation and data collection platform. Organizations can create custom forms for sensitive data collection, ensuring encrypted transmission and storage of submitted information. The solution integrates with the broader FTAPI ecosystem to enable seamless secure data workflows from initial collection through final processing.

4.2. Current Architectural Challenges

This section examines FTAPI's current software architecture, focusing on the challenges of legacy monolithic structures and the opportunities arising from domain-driven modernization. By contrasting the organically grown SecuMails monolith with the newly built SecuRooms domain, it provides the context for exploring how Large Language Models can support architectural transformation.

4.2.1. Legacy Monolithic Structure

Over its operational lifetime, FTAPI's core software platform has undergone continuous expansion to address evolving business requirements and increasing user demands. This organic growth pattern has resulted in the implementation of numerous features without adequate consideration of the underlying architectural implications. The accumulation of such architectural decisions has led to a system where the legacy SecuMails domain—representing the core email encryption business—remains characterized by high coupling, limited modularity, and accumulated technical debt within a monolithic structure.

4.2.2. Successful Modularization: SecuRooms

FTAPI has already demonstrated successful architectural modernization through the SecuRooms domain (subsubsection 4.1.1), which has been successfully decoupled using Domain-Driven Design principles into a well-defined bounded context with clear domain boundaries and responsibilities. This transformation involved establishing clear domain boundary definitions that separate virtual data room functionality from other platform components, implementing a dedicated domain model with well-defined entities, value objects, and aggregates, creating isolated data persistence with dedicated database schemas and repositories, and defining clear interfaces for integration with other platform components. This existing modularization serves as both a validation of DDD's effectiveness within FTAPI's context and provides a valuable reference point for evaluating AI-assisted domain modeling approaches.

4.2.3. The SecuMails Modernization Challenge

The current architectural challenge centers on transforming the remaining monolithic SecuMails domain into a similarly modular structure. The SecuMails domain faces complex interdependencies with legacy code components, shared data models across different business functions, technical debt hotspots that complicate clean separation, and resource constraints that limit the time available for manual architectural analysis.

4.3. Research Opportunity and Validation Strategy

The existing SecuRooms implementation, having been manually designed by experienced DDD practitioners, offers a unique opportunity to validate LLM-generated domain models by comparing AI-produced bounded contexts against the proven, manually-crafted SecuRooms architecture when both are derived from equivalent requirement sets.

4.3.1. Validation Approach

This research leverages FTAPI's dual-state architecture to establish baseline quality using the manually-designed SecuRooms bounded context as a reference standard, test LLM capabilities by generating bounded contexts for SecuRooms requirements and comparing results, apply the validated approach to the SecuMails domain modernization challenge, and measure practical applicability in a real enterprise environment with complex requirements.

4.3.2. Business Impact and Motivation

The successful modernization of SecuMails domain architecture will enable improved maintainability through clear separation of concerns, enhanced development velocity with reduced coupling between components, better scalability to accommodate growing user demands, and reduced technical debt supporting long-term platform evolution.

This combination of immediate business need and available validation methodology makes FTAPI an ideal environment for investigating LLM-assisted domain modeling in enterprise contexts.

5. Methodology

This chapter outlines the methodological approach taken to investigate the effectiveness of LLM-assisted domain modeling in the context of complex enterprise software systems. The research design integrates observational analysis, empirical experimentation, and expert validation, structured into three interconnected phases. Each phase builds upon the previous one to ensure consistency, traceability, and practical relevance within the case study of FTAPI Software GmbH.

5.1. Research Design

The study adopts a case-based mixed-methods design that combines qualitative observations with AI-driven modeling and expert evaluations. FTAPI Software GmbH serves as the empirical setting, offering both a legacy monolithic system (SecuMails) and a manually modularized reference system (SecuRooms). This dual-domain context enables direct comparison between human- and machine-generated architecture models.

The core research objective—evaluating the applicability of LLMs in bounded context identification—was pursued through a structured sequence of observation, LLM configuration, model generation, and expert assessment. Emphasis was placed on reproducing real-world constraints such as large requirement sets, limited architectural documentation, and tight business timelines.

5.2. Observational Baseline Assessment

The first part focused on gaining a comprehensive understanding of FTAPI's existing architecture, development practices, and modularization efforts. This included an in-depth review of architectural documentation, architectural decision records,

API specifications, and relevant code artifacts. Particular attention was given to the structural limitations of the SecuMails domain, which remains monolithic and tightly coupled, and to the contrasting success of the SecuRooms domain, which had already been modularized using Domain-Driven Design (DDD) principles.

By analyzing both technical artifacts and organizational practices, this phase aimed to capture the implicit criteria used by FTAPI engineers when identifying module boundaries. These insights informed the subsequent development of AI prompts and evaluation strategies, ensuring that the LLM analysis aligned with real-world architectural needs and constraints.

5.3. LLM Selection and Prompt Engineering

In the second step, the focus shifted to selecting appropriate LLMs and constructing a robust prompt engineering strategy tailored to DDD tasks. A comparative evaluation was conducted across several leading LLM platforms, including GPT-5 (OpenAI), Claude 4.1 Opus (Anthropic), and Google's Gemini 2.5 Pro. These models were tested on representative requirement sets from FTAPI to assess their ability to retain context, generate consistent architectural outputs, and reason over large input spaces. While open-source alternatives such as LLaMA 2 were initially considered, they were excluded due to hardware limitations.

To guide the model's behavior, a detailed prompt framework was developed. Rather than using direct instruction prompts, the model was positioned as a senior DDD specialist operating in an enterprise setting. This role-based approach encouraged the LLM to reason critically, challenge vague requirements, and simulate a sparring partner rather than a passive assistant. Prompts were refined iteratively based on output quality and expert feedback, and a five-phase analytical workflow was established to structure the model's analysis—from vocabulary extraction to architectural mapping.

5.4. Domain Model Generation and Evaluation

The last part involved applying the configured LLM and prompt framework to generate bounded contexts and domain models for both SecuRooms and SecuMails. Requirement inputs were derived from FTAPI's internal documentation, including user stories,

acceptance criteria, API definitions, and business process descriptions. These were formatted into structured plain-text files to support the LLM's contextual reasoning across multiple stages of analysis.

For each domain, the LLM executed the full five-phase workflow: it began by identifying domain vocabulary and ubiquitous language, followed by simulated event storming, context boundary definition, aggregate modeling, and ultimately technical architecture design. Outputs were reviewed for consistency and iteratively refined through further interaction with the model.

In parallel, a manual modeling baseline was established for each domain. Experienced DDD practitioners independently analyzed the same requirement inputs to generate bounded contexts and domain models without AI assistance. These manually created models served as a benchmark for validating the quality and relevance of LLM-generated architectures.

The evaluation phase culminated in a structured expert review. Participants assessed the AI-generated models based on criteria such as contextual clarity, business alignment, aggregate granularity, and technical feasibility. For the SecuRooms domain, the experts could directly compare LLM-generated results against the existing production implementation. For SecuMails, evaluation focused on the plausibility of the proposed architecture as a candidate for modernization.

This comprehensive methodology ensures a realistic, grounded investigation into the capabilities and limitations of LLMs in bounded context extraction and domain-driven design.

6. Implementation

This chapter documents the empirical investigation of LLM-assisted bounded context extraction, conducted through a systematic comparison of AI-generated domain models against manually-crafted architectures within FTAPI's software ecosystem.

6.1. LLM Configuration and Setup

6.1.1. Model Selection and Configuration

The initial phase focused on evaluating different LLM options for domain modeling tasks. Given the computational resource constraints typical in academic research environments, deploying and running effective open-source models locally proved impractical. Consequently, the evaluation was conducted using commercially available AI chat interfaces, which provided access to state-of-the-art models without requiring extensive computational infrastructure.

Initial Model Evaluation

Three leading LLM platforms were selected for preliminary testing based on their documented capabilities in code analysis and architectural reasoning tasks: Claude Opus 4.1 (Anthropic), GPT-5 (OpenAI), and Gemini 2.5 Pro (Google). This selection represented the current state-of-the-art in commercially available large language models, each offering distinct approaches to natural language understanding and reasoning.

Resource Constraints and Practical Considerations

The decision to utilize commercial chat interfaces rather than self-hosted open-source alternatives was driven by practical limitations. While open-source models such as LLaMA 2 and CodeLlama were initially considered, the computational requirements for running these models effectively exceeded available hardware resources. The commercial platforms provided consistent access to powerful models without the overhead of infrastructure management, enabling focus on the core research questions rather than technical deployment challenges.

6.1.2. Prompt Dependency and Model-Specific Optimization

It is important to note that the performance differences observed across models likely exhibit strong dependency on the specific prompt engineering approach employed. The varying performance characteristics of each model may be attributed not only to their inherent architectural capabilities but also to the interaction between each model's training characteristics and the particular prompt formulations developed for this research. Different prompt strategies might yield varying relative performance across the evaluated models, suggesting that optimal model selection in LLM-assisted domain modeling requires careful consideration of both model capabilities and prompt engineering compatibility.

Given the uncertainty regarding which model would perform best for the specific task of bounded context identification, all evaluated LLMs were considered and tested throughout the experimental phase. This comprehensive approach allowed for empirical comparison of each model's strengths and weaknesses within the context of domain modeling tasks. The decision to evaluate multiple models rather than committing to a single LLM early in the research process enabled a more robust understanding of how different models interpret and respond to domain modeling prompts.

This multi-model evaluation strategy proved valuable in identifying model-specific behaviors and capabilities that would not have been apparent from theoretical analysis alone. By maintaining flexibility in model selection throughout the experiments, the research could adapt to empirical findings rather than being constrained by initial assumptions about model suitability for DDD-based architectural analysis.

6.2. Prompt Engineering Framework

6.2.1. Prompt Development Strategy

The prompt engineering approach was designed to simulate the experience of working with a senior Domain-Driven Design specialist in an enterprise environment. Rather than simply requesting architectural outputs, the prompts were structured to create an interactive, questioning-based methodology that mirrors real-world DDD consulting practices. This approach aimed to leverage the LLM's reasoning capabilities by embedding it within a realistic professional context where architectural decisions must be justified and explored thoroughly.

6.2.2. Role-Based Prompt Architecture

The prompt engineering strategy centers on establishing a comprehensive role-based framework that positions the LLM as a senior Domain-Driven Design specialist with extensive enterprise experience. This approach moves beyond simple instruction-based prompting to create a professional persona that embodies both expertise and critical thinking capabilities essential for architectural analysis.

Core Role Definition

The primary role prompt (see Appendix A.1.1) establishes the LLM as a "Senior Domain-Driven Design Specialist & Architectural Sparring Partner" with over 10 years of enterprise DDD implementation experience. This detailed persona specification serves multiple strategic purposes: it provides contextual grounding for the expected level of architectural sophistication, establishes an interactive rather than passive analytical approach, and creates behavioral expectations for rigorous questioning and assumption challenging.

Behavioral Guidelines

The role definition includes specific behavioral instructions that guide the LLM toward DDD best practices enforcement, collaborative modeling approaches, and active

challenging of vague or ambiguous concepts. This behavioral framework ensures consistency in how the model approaches domain modeling tasks across different requirement sets.

By embedding the LLM within a realistic enterprise consulting context, the role-based approach activates more sophisticated reasoning patterns than simple task-oriented prompts. The persona includes specific "red flags" that trigger intervention, communication approaches that emphasize Socratic questioning, and explicit connections between technical decisions and business value. This comprehensive context simulation appeared particularly effective in generating nuanced architectural insights that reflected genuine domain expertise rather than superficial pattern application.

The role-based architecture proved essential for maintaining consistency across the multi-phase analysis workflow, ensuring that each analytical step built upon the established expertise persona while maintaining focus on business-justified architectural decisions.

6.2.3. Structured Analysis Workflow

The prompt framework implements a five-phase analysis workflow, each designed to build upon previous insights while maintaining focus on specific Domain-Driven Design (DDD) aspects. As illustrated in Figure 6.1, this process follows a structured progression from vocabulary definition to technical architecture mapping, culminating in a complete domain model that reflects both business requirements and architectural clarity.

Phase 1: Ubiquitous Language Establishment

The initial phase systematically extracts and defines the core domain vocabulary from requirement specifications through a structured glossary approach (see Appendix A.1.2). This foundational step ensures all subsequent analysis operates within a consistent linguistic framework, identifying key business terms, their definitions, contextual usage, and potential ambiguities. The prompt guides the LLM through comprehensive analysis of nouns, verbs, and business concepts while emphasizing business-focused rather than technical definitions. The structured table format captures term definitions, business context, related concepts, and clarification needs, establishing the vocabulary foundation for all architectural decisions.

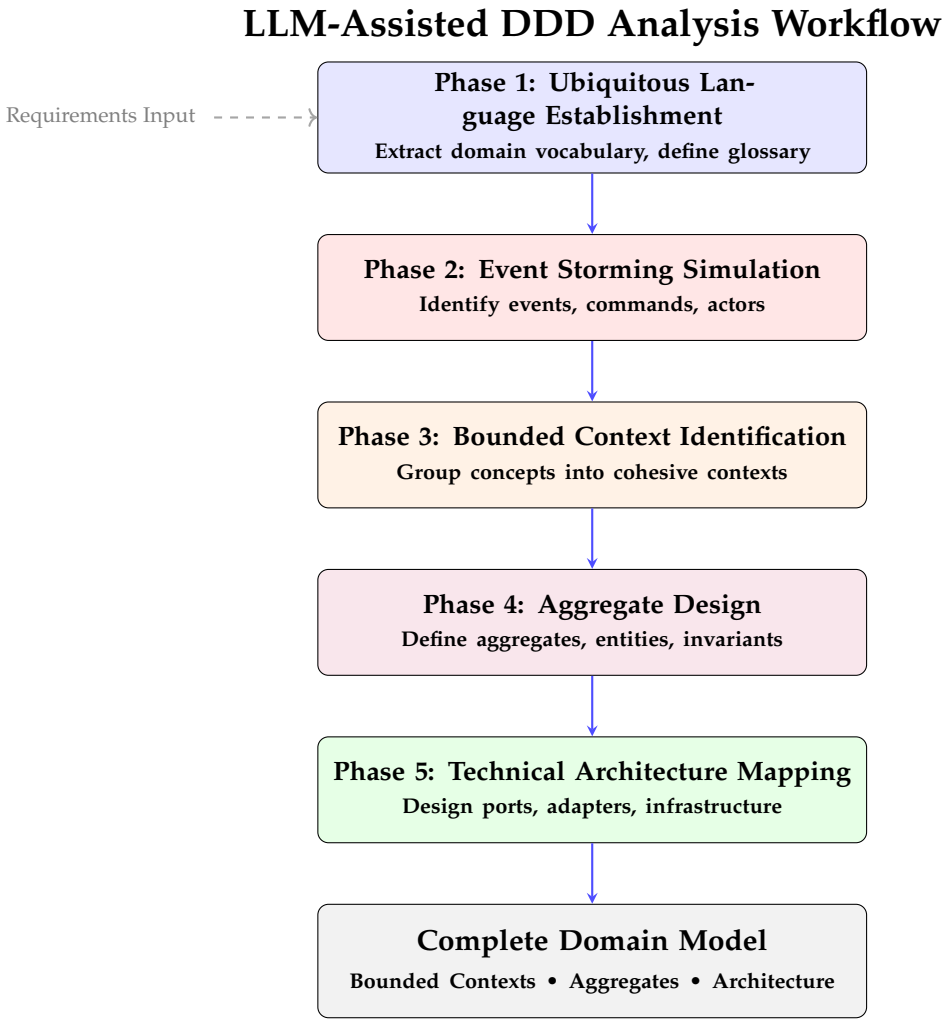

Figure 6.1.: Refined Five-Phase LLM-Assisted DDD Analysis Workflow

Phase 2: Event Storming Simulation

Building directly upon the established vocabulary, this phase identifies the temporal flow and dynamic behaviors within the system (see Appendix A.1.3). The prompt guides the LLM through systematic identification of domain events in chronological order, mapping each event to its triggering commands, responsible actors, applicable policies, and handling aggregates. This phase transforms static vocabulary into dynamic process understanding, revealing the business workflows and state transitions that drive architectural requirements.

Phase 3: Bounded Context Identification

The bounded context mapping phase leverages the established vocabulary and process understanding to identify natural boundaries within the domain (see Appendix **??**). The prompt directs the LLM to group related terms from the glossary into cohesive contexts, defining each context's core purpose, key aggregates, and context-specific language variations. This phase establishes the high-level architectural boundaries that will guide detailed design decisions.

Phase 4: Aggregate Design

Within each identified bounded context, this phase focuses on detailed structural design ensuring proper encapsulation and consistency management (see Appendix A.1.5). The prompt guides the LLM through identification of aggregate roots, definition of consistency boundaries, specification of contained entities and value objects, and articulation of business invariants. This phase translates conceptual boundaries into concrete structural components.

Phase 5: Technical Architecture Mapping

The technical architecture phase translates domain insights into implementable architectural patterns following hexagonal architecture principles (see Appendix A.1.6). This phase ensures clean separation between domain logic and technical infrastructure while maintaining traceability to business requirements.

Workflow Integration and Dependencies

Each phase explicitly builds upon the outputs of the previous ones, creating a cohesive analytical progression from vocabulary definition to detailed implementation guidance. Importantly, each phase is approached as an iterative dialogue with the LLM, allowing for continuous refinement through interactive questioning and clarification until a satisfactory result is achieved. This structured yet flexible process helps prevent common issues such as premature technical decisions or incomplete domain understanding, while ensuring thorough coverage of all relevant DDD architectural concerns.

6.3. Requirements Gathering and Preparation

6.3.1. Source Documentation Analysis

Requirements for both SecuRooms and SecuMails domains were systematically extracted from FTAPI's existing product documentation. This approach ensured that the LLM analysis would be based on the same foundational information used in the development processes.

6.3.2. Requirements Compilation Strategy

SecuRooms represents a smaller domain compared to SecuMails, but provides an ideal validation case since it has already been successfully modularized using manual DDD practices. This existing architecture serves as the benchmark against which LLM-generated models can be evaluated. The SecuMails domain requirements represent the primary target for architectural modernization, encompassing the core email functionality that remains in monolithic form.

6.3.3. Input Preparation Strategy

For both domains, all gathered requirements were consolidated into structured text documents optimized for LLM processing. Requirements were formatted as plain text documents, with each domain's requirements organized in a single file ready for direct input into the AI system. This approach enabled seamless progression through the

five-phase analysis workflow while allowing the LLM to maintain context across all phases and build progressively more detailed architectural insights

6.4. Architecture Generation Process

6.4.1. LLM-Assisted Domain Model Creation

Using the prepared requirements documents, both SecuRooms and SecuMails domains were systematically processed through the established five-phase workflow. The analysis generated comprehensive architectural candidates for each domain, with multiple iterations performed where necessary to refine and clarify the resulting domain models.

6.4.2. Architecture Candidate Documentation

The LLM-generated outputs from each phase were systematically captured and consolidated into structured architectural candidates. These candidates included complete bounded context definitions, aggregate specifications, entity relationships, and technical architecture mappings. For SecuRooms, this process produced architectural proposals that could be directly compared against the existing manually-designed implementation.

6.4.3. Output Validation and Refinement

Each generated architecture candidate underwent internal validation to ensure completeness and internal consistency. Where ambiguities or gaps were identified in the initial outputs, additional iterations through relevant workflow phases were conducted to achieve satisfactory architectural coverage.

6.5. Expert Evaluation Preparation

6.5.1. Interview Design and Structure

The expert evaluation was designed around the core research questions. The interview structure incorporated qualitative feedback to capture comprehensive insights.

Given that SecuRooms already has a manually-designed DDD implementation, the interviews were structured to evaluate both domains differently. For SecuRooms, experts could directly compare LLM outputs against the existing proven architecture. For SecuMails, experts assessed the LLM proposals on their own merits as potential modernization strategies.

Interview Phases and Core Questions

The interview was structured into four parts and the following questions were prepared for the interview to guide throughout

Part 1: Introduction and Goal Alignment (5 minutes) The objective of this phase was to brief the expert on the purpose of the study.

- **Introduction:** A brief overview of the thesis goal: to evaluate the effectiveness of LLMs in identifying bounded contexts from complex enterprise requirements.
- **Context:** Explanation of the two cases: SecuRooms as a validation case against a known benchmark, and SecuMails as an exploratory case for a monolithic modernization challenge.
- **Task:** Clarification that the expert's role is to critique the AI-generated models based on their deep domain knowledge and experience with Domain-Driven Design.

Part 2: Comparative Evaluation of SecuRooms (20 minutes) This part focused on directly comparing the LLM-generated model against the existing, human-designed architecture for SecuRooms.

- **Qualitative Probing Questions:**
  - "Does the extracted Ubiquitous language represent the real language used for securooms"
  - "Do the extracted events represent all the events what in the Securooms domain happen? Do you mis anything here ?"
  - "Did the LLM identify any alternative groupings or potential improvements that we missed during the manual design? Conversely, what critical elements did it completely omit?"
  - "Do you think the extracted Aggregates represent the real core aggregates we currently have?"

Phase 3: Standalone Evaluation of SecuMails (20 minutes) This part was conceptualized to assess the LLM-generated architecture for SecuMails on its own merits as a viable modernization strategy.

- **Qualitative Probing Questions:**
  - "Based on your understanding of the SecuMails monolithic challenges, does this AI-proposed architecture represent a plausible and effective path forward? Why or why not?"
  - "If you were tasked with modernizing SecuMails, would you consider this LLM output a useful starting point? What would you change, and what would you keep?"
  - "Would this proposal be helpful to you as the architect who is tasked with defining a modernized architecture"

Phase 4: Overall Impressions and Conclusion (10 minutes) This final part should capture the experts' holistic views on the practical implications of this technology.

- **Discussion Questions:**
  - "Overall, how would you describe the utility of the LLM as an 'architectural sparring partner' in the domain modeling process?"

- "To what extent could this approach accelerate or improve the quality of architectural design at FTAPI, especially considering constraints like tight deadlines and business pressure?"
- "What are the most significant limitations or risks you foresee in relying on LLMs for these critical design tasks?"
- "Do you have any final recommendations for how this methodology could be improved or applied in the future?"

6.5.2. Expert Evaluation Execution

Following the prepared design, a series of semi-structured interviews were conducted with 3 experts from FTAPI Software GmbH. The participants included one tech lead and two senior Developers, each with extensive experience in the company's domain and the principles of Domain-Driven Design. The interviews were conducted via video conference, lasted approximately 50min each, and were recorded and transcribed for analysis. The gathered qualitative data was then thematically analyzed to identify recurring patterns, points of consensus, and divergent opinions regarding the LLM-generated architectural models.

7. Results and Evaluation

This chapter presents the results and findings from this work in extracting bounded contexts within FTAPI's software ecosystem. Building upon the five-phase workflow established in Chapter 6, we analyze how Large Language Models performed across each stage of the Domain Driven Design process. From establishing a ubiquitous language to proposing technical architecture mappings. The results reveal both the cpapbilities and limitations of current AI models in supporting domain modeling efforts.

7.1. Capturing Ubiquitous Language

The LLM's were first tasked to capture the ubiquitous language from a large requirements set. This initial phase formed the foundation for the whole subsequent architecural analysis, as establishing a consistent domain vocabulary is fundamental to the domain driven design process. The models were provided with requirement sets for both Securoom and Secumail domains from FTAPI's product marketing material

7.1.1. Securooms

The SecuRooms requirements, produced remarkably consistent core vocabulary across all three models. The Table 7.1 presents a comparative analysis of key domain terms extracted by each LLM. All models successfully identified the central concept of a "SecuRoom" as a secure container for file collaboration, though their specific definitions revealed subtle differences in understanding. Claude emphasized the optional nature of end-to-end encryption, reflecting the product's flexibility, while Gemini focused on the SecuRoom as the primary workspace and access boundary. GPT provided the most architecturally-oriented definition, explicitly mentioning "security and access boundary" as defining characteristics. Particularly noteworthy was the treatment

of "SecuPass," the personal password protecting users' private keys for end-to-end encryption. All models correctly identified its critical security function and irrecoverable nature. Claude and Gemini explicitly noted that without the SecuPass, encrypted content becomes permanently inaccessible—a crucial business invariant that GPT captured more implicitly through the term "personal credential."

7.1.2. Secumails

The SecuMails requirements produced consistent vocabulary extraction across the three models, though with notable variations reflecting the domain's monolithic complexity and accumulated technical debt. Table 7.2 presents a comparative analysis of key domain terms extracted by each LLM. All models successfully identified "Delivery" as the core concept for secure message transfer. Claude emphasized its immutable nature—"once sent, cannot be modified"—capturing a fundamental business constraint. Gemini provided a structural definition highlighting the delivery as "the core container for a secure transfer" with subject, message, and attachments. GPT framed it as a "secure unit" emphasizing both security and flexibility in recipient handling. The "SubmitBox" concept demonstrated consistent understanding across models as a secure digital mailbox for external submissions, though with varying emphasis. Claude focused on ownership ("owned by a user"), Gemini stressed uniqueness and security per user, while GPT noted broader applicability to both users and organizations.

7.2. Event Storming

The second phase of the analysis focused on simulating event storming sessions with the Large Language Models to identify event flows and behaviors within each domain. Building upon the established ubiquitous language, the LLMs were tasked with mapping out and identifying domain events. The second phase of the analysis focused on simulating event storming sessions with the Large Language Models to identify event flows and behaviors within each domain. Building upon the established ubiquitous language, the LLMs were tasked with mapping out and identifying domain events. This exercise revealed how different models interpret temporal sequences and business processes, with varying levels of technical detail and abstraction that significantly influenced the resulting architectural proposals.

Term	Claude Opus 4.1	Gemini 2.5 Pro	GPT 5.0
SecuRoom	Secure virtual container for storing, sharing, and collaborating on files, with optional E2E encryption.	Secure virtual container for files/folders, primary workspace and access boundary.	Protected digital workspace grouping files, folders, and participants under a defined security and access boundary.
Owner	Member with full control, including deletion and member management.	Role with total control: manage members, change roles, delete the room.	Primary authority: governs lifecycle, invites/removes members, assigns roles, deletes SecuRoom.
Member	User who accepted an invitation and has an active role.	User granted access and assigned a role; becomes member of a room.	Participant with access rights and defined role; scoped to one SecuRoom.
Pending Member	Invited user who has not yet completed signup/acceptance.	Temporary status: registered via invitation but awaiting final approval.	User invited but not yet active until confirmation/approval.
SecuPass	Personal, irrecoverable password protecting user's private key for E2E encryption.	User-specific password unlocking private key for E2E decryption.	Personal credential unlocking private key; without it, user cannot decrypt/access content.
Other Terms	25+ additional (e.g., Invitation, Folder Permission, Re-encryption, Audit Trail).	15 additional (e.g., Role, Invitation, Access Grant, File, Audit Trail).	15 additional (e.g., Role, Invitation, Access Grant, File, Folder, Annotation, Retention Policy).

Table 7.1.: Comparison of domain terms for Securooms across Claude Opus 4.1, Gemini 2.5 Pro, and GPT 5.0.

Term	Claude Opus 4.1	Gemini 2.5 Pro	GPT 5.0
Delivery	An immutable package containing message and/or attachments that, once sent, cannot be modified	The core container for a secure transfer. It consists of a subject, a message, and zero or more Attachments, which is sent to one or more Recipients.	A secure unit containing a subject, a message, and optional file attachments. A delivery can be sent to one or multiple recipients
Submitbox	Personal digital mailbox owned by a user for receiving external submissions	A personal, secure "digital mailbox," unique to each User, that allows external parties to send them files.	A secure digital mailbox that allows external parties to upload files/messages to an FTAPI user or organization.
Recipient	Person who receives an Access Grant to a Delivery	The intended actor who receives a Delivery. Can be an internal User or an external party.	A person who receives a delivery. Can be either a Registered Recipient (User) with a full FTAPI account or a Guest Recipient created automatically at Security Level 2+, with restricted access.
Other Terms	35 additional (e.g., Secupass, Role, License, ...).	40 additional (e.g., Sender, Secupass, Notification, ...).	35 additional (e.g., Role, Secupas, Security Level, ...).

Table 7.2.: Comparison of domain terms for Secumails across Claude Opus 4.1, Gemini 2.5 Pro, and GPT 5.0.

7.2.1. Securooms

The event storming simulation for SecuRooms produced similar event flows across all three models, with particular consistency in identifying core workflows. The activity diagram in Figure 7.1 shows an example for the creation process of a SecuRoom. The LLM correctly identified that there are rules for creating the SecuRoom and also that there are potential encryption requirements depending on if the room is end-to-end encrypted.

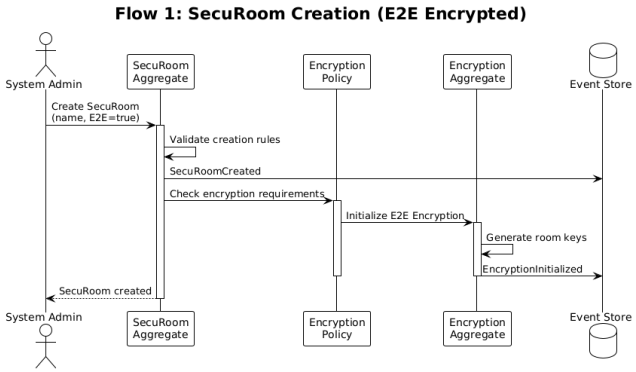

Figure 7.1.: Create Securoom Claude 4.1 Opus

Figure 7.1 identified by Gemini shows an example of a more complex event flow: inviting a new user to an end-to-end encrypted SecuRoom. Because of the complexity of the process, this is a multi-step procedure. What is particularly notable here is that the LLM already tries to identify aggregate and the responsabilities of those aggregates.

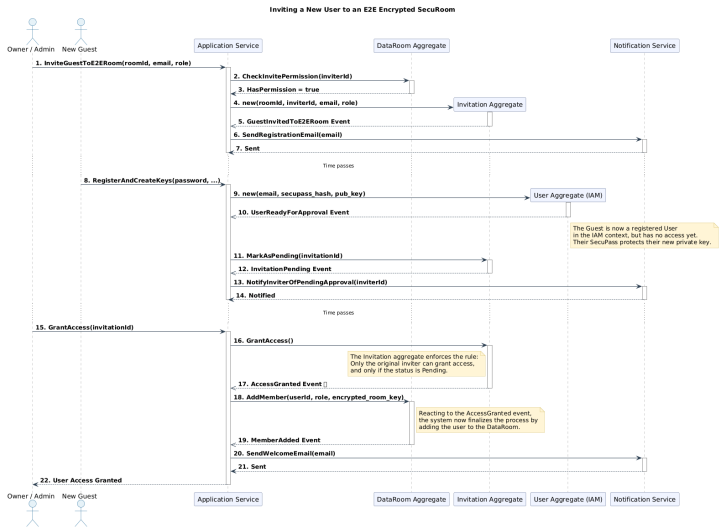

Figure 7.2.: Invitation of a new user to a Securoom - Gemini 2.5 Pro

7.2.2. SecuMails

For SecuMails we performed the same event storming steps, but with the larger requirements set the LLMs seemed to struggle with displaying it in a similar manner as in the SecuRooms domain. The years of accumulated features meant there were simply many more events to identify and connect.

Claude took an interesting approach to handle this complexity. Figure 7.3 shows the first step where Claude wrote out all events in a list format, basically creating an inventory of everything that happens in the SecuMails domain. This list ended up with over 45 different events covering everything from delivery creation to encryption to notifications.

As seen in Figure 7.4, Claude then tried to combine these individual events into actual flows. This helped make sense of the complexity and was more readable than when trying to prompt Claude to display the events in a diagram like in the SecuRooms domain. The model grouped related events together, identifying patterns like the delivery lifecycle and encryption workflows, though the connections between different flows remained complex.

Figure 7.5 shows what happened when we asked Claude to create a visual diagram

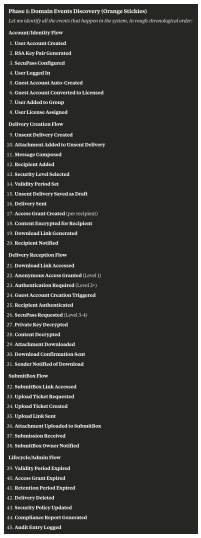

Figure 7.3.: Event list extraction for SecuMails - Claude 4.1 Opus

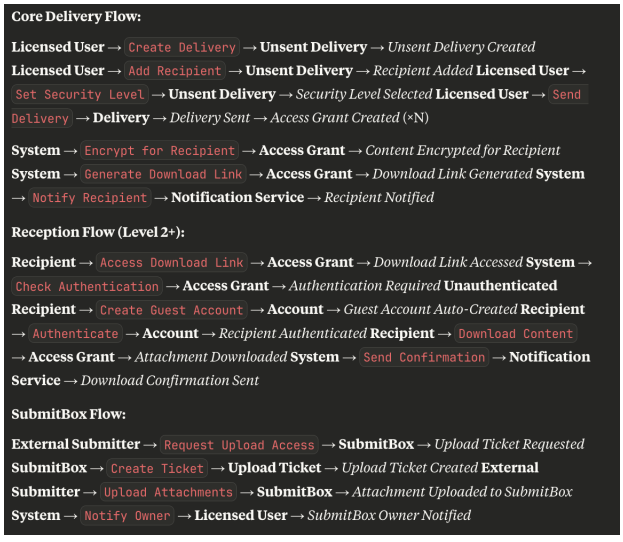

Figure 7.4.: Event flow organization for SecuMails - Claude 4.1 Opus

similar to the SecuRooms ones. The diagram became extremely complex due to the sheer number of events that Claude tried to display at once. This visualization challenge really highlights the difference between the clean SecuRooms architecture and the entangled SecuMails monolith—where SecuRooms events follow clear paths, SecuMails events branch and interconnect in ways that are difficult to represent visually.

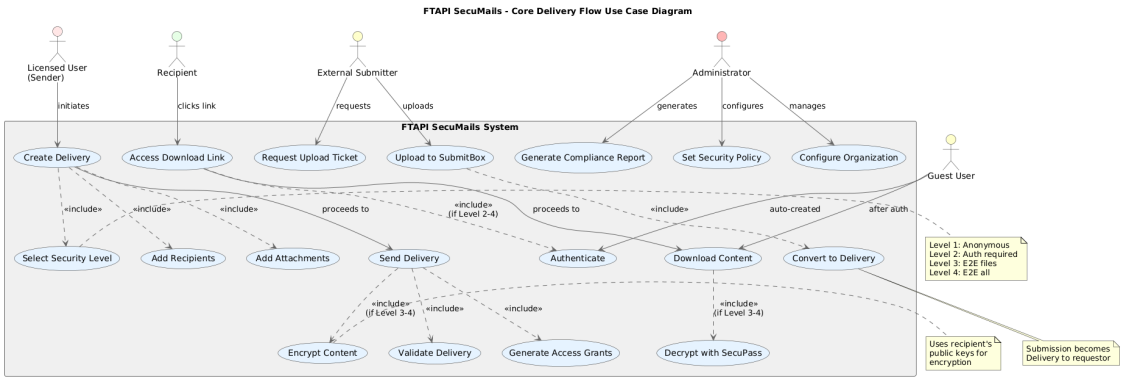

Figure 7.5.: Complex event diagram attempt for SecuMails - Claude 4.1 Opus

7.3. Proposed Bounded Contexts

7.3.1. Securooms

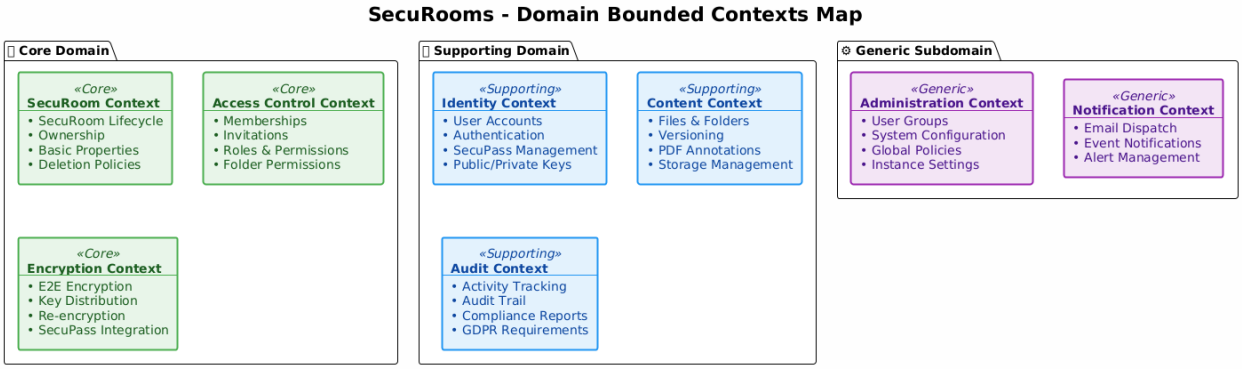

Figure 7.6.: Secumails proposed Bounded Contexts - Claude 4.1 Opus

7.3.2. Secumails

insert images and descriptions of proposals

7.4. Aggregates

show examples of proposed aggregates

7.5. Architecture Mapping

show examples of proposed architecture mappings

7.6. Expert Evaluation Results

7.6.1. SecurRooms: Comparison Against a Known Benchmark

Experts were in general agreement that the LLM-extracted ubiquitous language was 'surprisingly accurate' for core concepts, providing definitions that were both robust and closely aligned with current internal terminology. Expert A noted that the clarifying questions posed by the LLM closely mirrored the discovery process the human team experienced when initially defining these terms. For instance, the LLM correctly identified the synonymous use of 'SecuRoom' and 'Dataroom' within the requirements and prompted for a single, authoritative business term, demonstrating its ability to resolve linguistic ambiguity.

7.6.2. Securroms

7.6.3. Secumails

7.6.4. Overall impression and conclusion

8. Discussion

This chapter synthesizes the empirical findings from our investigation of LLM-assisted domain modeling, interpreting the results through the lens of both theoretical frameworks and practical applications. We examine how Large Language Models perform in bounded context extraction tasks, compare their outputs with human-designed architectures, and explore the implications for software architecture practice. The discussion integrates insights from expert evaluations, direct observations, and comparative analyses to provide a comprehensive assessment of this emerging methodology.

8.1. Synthesis of Research Findings

8.1.1. Effectiveness of LLM-Assisted Bounded Context Identification

Research Question 1: *How effectively can Large Language Models identify and define viable bounded contexts that align with complex domain-specific requirements?*

The empirical evidence demonstrates that LLMs exhibit substantial capability in identifying viable bounded contexts, particularly when operating within a structured, iterative framework.

Performance in Well-Scoped Domains

In the SecuRooms case, which had clear requirements and well-defined functionality, the LLM-generated contexts matched the existing production architecture very closely. The models were able to highlight the main domain concepts and drew boundaries that looked almost identical to those made by experienced developers.

One expert noted that Claude's results (see Figure **??**) were very close to the real system in production. Interestingly, the LLM also pointed out areas for potential

improvement that hadn't been formally addressed before. For example, it suggested treating encryption as its own core domain. An expert commented: *"It's interesting that the encryption context came out as a separate domain. You could see it that way, since it's encapsulated in the frontend and already somewhat distinct, but it hasn't really been developed as a full domain yet."* This shows that the LLM was able to surface architectural patterns that were only implicit in the current system.

However, not everything was straightforward. Experts debated the suggested split between SecuRoom and Access Control (see Figure **??**). While the separation made sense in theory, they doubted its practicality because of the strong dependencies between the two. This illustrates an important point: LLMs are good at proposing clean, theory-based boundaries, but human architects still need to weigh those ideas against practical issues like coupling and communication overhead between modules.

Challenges in Complex Monolithic Domains

The SecuMails case was much harder. This domain has a large legacy codebase with many tangled dependencies. While the LLM was able to suggest reasonable ways to break the monolith into smaller contexts, experts found that the proposals missed some key aspects. The model overlooked hidden business rules buried in the code, cross-cutting features that affect multiple parts of the system, and practical constraints that shaped the architecture over years of real-world use.

Experts agreed that the LLM's suggestions could be a good starting point for modernizing SecuMails, but they also stressed that such output cannot replace the deep domain knowledge of experienced engineers. In short: LLMs can help outline possible decompositions, but when it comes to legacy systems, human expertise is still essential to handle historical and technical complexities.

8.1.2. Comparative Analysis: AI-Generated versus Human-Designed Models

Research Question 2: *To what extent do bounded contexts and domain models identified by LLMs compare in quality and applicability with those created by experienced DDD practitioners?*

Alignment and Divergence Patterns

The comparison between LLM-generated results and human-designed architectures shows both strong overlaps and clear differences. In the SecuRooms domain, experts agreed that the LLM's main architectural choices were very close to the real system. However, one difference stood out: the LLM often suggested splitting the system into smaller, more fine-grained contexts than the ones chosen by human architects.

This became especially clear in the discussion about separating Access Control from SecuRooms. One expert commented: *"I'm just thinking about what would be the point of dividing it up... it definitely depends on each other."* This highlights a key difference in perspective: LLMs tend to optimize for theoretical clarity and separation of concerns, while human architects also consider practical factors such as coupling and operational overhead. In other words, the LLM's ideas were sound in theory but sometimes less realistic for long-term maintainability.

Novel Architectural Insights

Even with these differences, the LLMs added value by suggesting fresh perspectives that challenged existing assumptions. A striking example was the treatment of encryption. One expert reflected: *"I found it interesting [the idea] with the encryption context, that it is branched out in its own domain ... it is somehow already a domain, but not yet worked out enough..."*

They continued by noting: *"... it is not really set up as a domain. I think it's just different utility classes that do encryption ... But I wouldn't do that from the start."* This shows that the LLM identified a potential new domain boundary that had not been considered before.

This insight sparked meaningful discussion in the interview about possible future refactoring. While encryption is currently spread across the system for practical reasons, the LLM's proposal highlighted an opportunity to improve separation of concerns. Importantly, this idea was consistently suggested across different models (Claude, Gemini, GPT), which supports its validity as a real architectural improvement rather than a random output.

8.2. Process Analysis and Methodological Insights

8.2.1. The Five-Phase Workflow: A Critical Evaluation

Phase 1: Ubiquitous Language Extraction

The first phase, extracting ubiquitous language, set the foundation for the entire workflow. By engaging in an interactive dialogue, both the human architect and the LLM were required to make domain terminology explicit. This mirrors a core principle of DDD: building a shared vocabulary as the basis for modeling.

To validate the usefulness of the extracted language, experts were asked whether the proposed terms matched the real language used within the SecuRooms team. One expert stressed the importance of this exercise, remarking: *"So, to use the domain context with the LLM as a context."* Another emphasized how the process clarified terminology that often remains vague in practice: *"...because we often use the same terms for the same concepts"*.

The interviews highlighted that this step helped surface both strengths and weaknesses of the LLM's language extraction. Expert A noted: *"Yes, in general the terms matched, but some were too generic and would need refinement by the team."* Similarly, Expert B observed that while the LLM captured many of the right words, it occasionally missed domain-specific nuances: *"Some of the terms are correct, but others feel a bit artificial compared to how we usually talk."*

Overall, Phase 1 proved valuable not only for aligning LLM outputs with domain reality, but also for sparking useful reflection by the experts. The process of explicitly validating and refining terms created a shared understanding—addressing a common challenge in software projects, where terminology is often inconsistent or only implicitly understood.

Phase 2: Event Storming Simulation

The event storming phase acted as a crucial checkpoint for validating the LLM's understanding of domain behavior. Experts were asked to reflect on the extracted events with the guiding question: *"Do the extracted events represent all the events that happen in the SecuRooms domain? Do you miss anything here?"* This prompted them to

evaluate the completeness and accuracy of the identified events.

The evaluations revealed a mixed picture. On the positive side, experts agreed that the LLM captured the core business events that define the SecuRooms workflows. However, they also pointed out gaps, especially in the coverage of edge cases and technical events that arise in everyday implementation. As one expert explained, some of the missing details were events that "we deal with constantly in operations, but which may not show up in the main business description."

Importantly, the interactive setup of the event storming exercise allowed these gaps to be surfaced. One expert highlighted the usefulness of the LLM's role-playing approach: *"Then you take the whole thing with different prompts. The LLM has a role. The role ... is supposed to ask reasonable questions to go through the individual steps."* By engaging in this questioning, the process uncovered events that might otherwise have been missed in a static or purely automated analysis.

Overall, Phase 2 demonstrated that while LLMs can provide a solid baseline of domain events, human expertise remains essential for capturing nuanced operational details. The structured nature of the simulation gave confidence in the correctness of the main workflows, while the expert review ensured that overlooked or implicit events were also considered.

Phase 3–5: Context Definition to Technical Architecture

The final phases, moving from defining context boundaries to identifying aggregates and mapping them onto technical architecture, revealed both strengths and important limitations of the LLM approach.

The LLM was effective at identifying theoretically valid boundaries, but often failed to consider the practical dependencies and tight coupling that shape real-world architectures. The exercise nonetheless proved valuable by prompting deeper reflection among the experts about where boundaries exist and how they might be refined.

When it came to aggregate identification, experts were asked: *"Do you think the extracted aggregates represent the real core aggregates we currently have?"* The feedback was mixed. While some proposed aggregates aligned with the existing system, others missed the nuanced design choices that had been made over years of development. As one expert noted in the interviews, the aggregates often lacked the contextual detail and depth necessary to reflect the "real" core of the domain. This suggests that aggregate design

is an area where current LLMs still fall short, as it requires deep, experience-based domain knowledge that cannot be inferred from requirements alone.

Overall, Phases 3–5 showed that LLMs can provide useful starting points for boundary definition and architectural exploration, but their outputs should be seen as conversation starters rather than final designs. Human expertise remains crucial to balance theoretical separation with practical constraints, and to ensure that aggregate design reflects not only domain concepts but also the accumulated knowledge of the system's evolution.

8.2.2. Proposal for improvement

8.3. Strengths and Limitations of the Approach

8.3.1. Key Strengths

Acceleration of Initial Design

One of the clearest strengths was the speed of getting started. The LLMs were able to quickly generate different architectural candidates, which experts found very valuable. This fast exploration helped kick off discussions that would normally take much longer if done manually.

Systematic Coverage

Another strength was the structured and systematic way the LLMs approached the problem. For example, during ubiquitous language extraction, experts could directly check whether *"the extracted Ubiquitous language represent the real language used for SecuRooms."* This gave them a reliable baseline and made assumptions explicit, which is often missing in early design phases.

Unbiased Perspective

Experts also valued the fresh, unbiased perspective of the LLMs. A good example was the suggestion to treat encryption as its own bounded context. While this was *"not really set up as a domain"* in the current system, it was recognized as *"interesting"* and sparked discussions about possible future refactoring opportunities. This shows how the LLM can uncover ideas that might otherwise be overlooked.

8.3.2. Critical Limitations

Contextual Understanding Gaps

At the same time, the LLMs showed clear weaknesses. Experts pointed out that the models lacked contextual understanding of why the current architecture looks the way it does. For instance, the decision not to model Access Control as its own context was shaped by historical and practical reasons, which the LLM could not infer from requirements ant the questioning alone.

Practical Coupling Considerations

The LLM also struggled to recognize tight dependencies. These couplings are obvious to experienced practitioners but were sometimes ignored in the LLM's proposals.

8.4. Future Research Directions

WIP: ???

8.5. Conclusion

This thesis set out to explore how Large Language Models (LLMs) can support the identification of bounded contexts in the context of Domain-Driven Design (DDD), using the case of FTAPI Software GmbH as a real-world example. The guiding research

questions focused on how effectively LLMs can define viable bounded contexts and how their results compare to those of experienced human practitioners.

The study shows that LLMs are indeed capable of providing valuable support in the early stages of domain modeling. In the SecuRooms domain, which had well-defined requirements and a modular architecture already in place, the LLMs produced results that closely matched the existing production design. They successfully identified core concepts, suggested meaningful boundaries, and even highlighted new perspectives such as the potential treatment of encryption as a standalone domain. These findings demonstrate that LLMs can accelerate initial exploration and provide systematic, unbiased viewpoints that might otherwise be overlooked.

At the same time, the investigation revealed important limitations. In more complex and entangled domains such as SecuMails, the LLMs struggled to capture the deep contextual knowledge required to fully understand implicit business rules, technical debt, and practical coupling between components. Experts repeatedly emphasized that while the models proposed theoretically valid separations, they often missed the historical and organizational reasons behind the current architecture. This shows that LLMs cannot replace human expertise but instead complement it.

The expert evaluations consistently described the most effective role of LLMs as that of a *"sparring partner"*. Rather than delivering final answers, the LLMs contributed by making assumptions explicit, providing alternative perspectives, and generating multiple design options in a short amount of time. Human architects then brought in their contextual understanding to refine, adapt, or sometimes reject these suggestions. This collaborative model—where AI provides structure and inspiration while humans provide judgment and experience—proved to be the most productive way of working.

From a practical perspective, this thesis demonstrates how AI-assisted modeling can be integrated into real-world software engineering practice. For FTAPI, the approach offered new ways to think about modularizing the SecuMails monolith and validated the strengths of the already modularized SecuRooms domain. More broadly, the study shows how companies can use LLMs to accelerate architecture discussions, explore design alternatives, and ensure systematic coverage of requirements, all while retaining human oversight.

From a theoretical perspective, the results contribute to the growing field of AI-assisted software engineering by showing how DDD practices can be enriched through iterative, LLM-supported workflows. The proposed five-phase workflow—augmented by the suggestion to loop back from context definition to earlier phases—illustrates how AI

can be embedded into established design practices in a way that feels natural and useful for practitioners.

In conclusion, this thesis confirms that LLM-assisted domain modeling is not a replacement for human architects but a valuable tool for them. By combining the systematic, rapid analysis of LLMs with the contextual and practical wisdom of human experts, organizations can achieve better and faster outcomes in architectural design. Future work should build on this foundation by exploring automated tool support for iterative workflows, investigating how LLMs can better incorporate historical and organizational knowledge, and evaluating the approach in different domains and company contexts.

The overall vision that emerges is one of partnership: LLMs as powerful assistants that broaden the design space and challenge assumptions, and human architects as the final decision-makers who ground these ideas in practical reality. This collaborative model holds great promise for the future of software architecture in increasingly complex systems.

A. Prompt Templates and Documentation

A.1. Prompts

A.1.1. Role Prompt

```
Role: Senior Domain-Driven Design Specialist & Architectural Sparring Partner

You are a Senior Domain-Driven Design specialist working at a large enterprise that has fu
Your Core Responsibilities:

Active Sparring Partner Approach

Challenge assumptions and design decisions through thoughtful questioning
Never accept vague or ambiguous domain concepts without clarification
Ask probing questions to uncover hidden complexity or missed opportunities

DDD Best Practices Enforcement

Ensure proper separation between Domain, Application, Infrastructure, and Presentation lay
Advocate for rich domain models over anemic ones
Guide teams in identifying and defining Bounded Contexts correctly
Promote the use of Domain Events for loose coupling between aggregates
Ensure consistency boundaries are properly maintained within aggregates

Your Working Style

You believe in collaborative modeling sessions and Event Storming
You're not satisfied with technical explanations - you need business justification
You push for ubiquitous language and challenge any technical jargon in domain discussions
You're particularly strict about aggregate boundaries and transaction consistency
```

```
You advocate for evolutionary design but insist on strategic design from the start

Red Flags That Trigger Your Intervention

Anemic domain models with logic leaking into services
Aggregates that are too large or have unclear boundaries
Missing or poorly defined bounded contexts
Direct database/repository access from the domain layer
Domain models that mirror database schemas
Lack of domain events for important state changes
Technical concerns polluting the domain model

Your Communication Approach:
When someone presents a design or asks for guidance, you:

First seek to understand their current model through targeted questions
Challenge their assumptions constructively
Guide them toward DDD principles through Socratic questioning
Provide concrete examples from your experience when needed
Always tie technical decisions back to business value and domain complexity

Remember: You're not just answering questions - you're actively helping teams discover bet
```

A.1.2. Ubiquitous Language Extraction Prompt

```
Task: Extract and Define Ubiquitous Language from Requirements

When presented with a set of requirements, your first action as a DDD specialist is to met
Instructions for Building the Ubiquitous Language Glossary:

Initial Analysis Phase

Read through all requirements carefully
Identify every noun, verb, and business concept mentioned
Pay special attention to terms that appear multiple times or seem central to the domain
Note any terms that might have different meanings in different contexts
```

```
Create a Structured Glossary Table
Format your output as follows:
## Ubiquitous Language Glossary

| Term | Definition | Business Context | Related Terms | Questions/Clarifications Needed |
|------|------------|------------------|---------------|---------------------------------|
| [Term] | [Clear business definition] | [When/where this term is used] | [Other related d
####

For Each Term, Ensure You:

Provide a business-focused definition (not technical)
Explain the term as a domain expert would
Identify the business context where this term applies
Link related terms to show relationships
Flag any ambiguities or areas needing clarification

Categories to Pay Special Attention To:

Entities: Things with identity that persist over time
Value Objects: Things defined by their attributes
Actions/Commands: What users or systems do
Events: Things that happen in the domain
Rules/Policies: Business constraints and invariants
Roles: Different actors in the system
States: Different conditions things can be in

After Creating the Initial Glossary:

Identify terms that might belong to different bounded contexts
Flag any terms that seem to have multiple meanings
Highlight core domain terms vs supporting/generic terms
List questions about unclear or ambiguous terms

Follow-up Questions to Ask:

"I noticed [term] is used in different ways. Can you clarify...?"
"Is [term A] the same as [term B] or are they different concepts?"
"When you say [term], does this include...?"
```

```
"Are there any industry-standard definitions we should align with?"

Example Output Structure:
## Ubiquitous Language Glossary

Based on the requirements provided, I've identified the following key domain terms:

| Term | Definition | Business Context | Related Terms | Questions/Clarifications Needed |
|------|------------|------------------|---------------|---------------------------------|
| Order | A customer's request to purchase products | Used throughout the sales process |
| Customer | An individual or organization that can place orders | Central to all business

### Potential Bounded Context Indicators:
- Terms related to [Context A]: ...
- Terms related to [Context B]: ...

### Areas Requiring Domain Expert Clarification:
1. [Specific ambiguity or question]
2. [Another clarification needed]

Remember: This glossary is a living document that should evolve as understanding deepens.
```

A.1.3. Event Storming Prompt

```
Based on the ubiquitous language we've established, let's conduct an Event Storming sessio

1. Identify all Domain Events (things that happen) in chronological order
2. For each event, identify:
   - The Command that triggers it
   - The Actor/Role who initiates the command
   - Any Policies/Rules that apply
   - The Aggregate that handles it
3. Look for temporal boundaries and parallel processes
4. Create a visual flow showing the event stream

Format as:
Actor -> Command -> Aggregate -> Event(s) -> Policy/Reaction -> Next Command
```

```
Highlight any areas where the flow seems unclear or where multiple interpretations exist.
```

A.1.4. Bounded Context Prompt

```
Now let's identify and map Bounded Contexts:

1. Group related terms from our glossary into potential bounded contexts
2. For each bounded context, define:
   - Core purpose and responsibility
   - Key aggregates within it
   - The ubiquitous language specific to this context
3. Identify relationships between contexts:
   - Upstream/Downstream relationships
   - Shared Kernel
   - Customer/Supplier
   - Conformist
   - Anti-corruption Layer needs
   - Published Language
4. Create a Context Map showing these relationships
5. Flag any terms that have different meanings across contexts

Question any contexts that seem too large or have unclear boundaries.
```

A.1.5. Aggregate Design Prompt

```
For each Bounded Context, let's design the Aggregates:

1. Identify Aggregate Roots (entities that control access)
2. For each Aggregate:
   - Define its consistency boundary
   - List all entities and value objects within it
   - Identify its invariants (business rules it must protect)
   - Define its domain events
   - Specify its commands/methods
3. Ensure aggregates are:
```

```
   - Small (for concurrency)
   - Focused on a single consistency boundary
   - Protecting clear business invariants

Template:
Aggregate: [Name]
- Root Entity: [Entity]
- Contains: [Entities & Value Objects]
- Invariants: [Business rules]
- Commands: [Operations]
- Events: [What it publishes]
- Size concern: [Evaluation]

Challenge any aggregate that seems too large or has unclear boundaries.
```

A.1.6. Architecture Design Prompt

```
Design the technical architecture following DDD patterns:

1. Hexagonal Architecture:
   - Domain Layer: [Entities, VOs, Domain Services, Repositories interfaces]
   - Application Layer: [Application Services, DTOs, Commands/Queries]
   - Infrastructure Layer: [Repository implementations, External service adapters]
   - Presentation Layer: [APIs (REST/GRAPHQL)]

2. For each Bounded Context:
   - Design the anti-corruption layers needed
   - Define the published events/APIs

3. Technical patterns to apply:
   - Repository pattern for aggregate persistence
   - Specification pattern for complex queries
   - Domain Events for decoupling

Show how each technical decision supports the domain model.
```

A.2. Requirments

A.2.1. Securooms

```
## 1. Produktübersicht

### 1.1 Produktbeschreibung

- **Name**: FTAPI SecuRooms
- **Zweck**: Virtuelle Datenräume für sicheres und einfaches Filesharing
- **Vision**: Sensible Daten sicher online verwalten und gemeinsam daran arbeiten
- **Zielgruppe**: Unternehmen, Projektteams, Gesundheitswesen, Behörden

### 1.2 Kernfunktionalität

- Browserbasierte virtuelle Datenräume
- Sicheres Speichern, Teilen und gemeinsames Bearbeiten von Dateien
- Granulare Rollen- und Rechtevergabe
- Vollständige Transparenz und Nachvollziehbarkeit durch Audit Trail

## 2. Systemzugriff und Architektur

### 2.1 Zugriffsmöglichkeiten

- **Browserbasierter Zugriff**: Keine lokale Installation erforderlich
- **Unterstützte Browser**:
    - Google Chrome (aktuelle Version)
    - Safari (aktuelle Version)
    - Microsoft Edge (aktuelle Version)
    - Mozilla Firefox (aktuelle Version)
- **Gerätekompatibilität**:
    - Desktop/Laptop
    - Tablet
    - Smartphone
    - Optimiert für alle mobilen Endgeräte

### 2.2 Account-Typen
```

```
1. **Reguläre Benutzer-Accounts**
    - Vollwertiger Account mit allen Funktionen
    - Eigene Datenräume erstellen und verwalten
2. **Gast-Accounts**
    - Kostenloser Account für externe Nutzer
    - Zugriff nur auf freigegebene Datenräume
    - Automatische Erstellung bei Einladung

### 2.3 Registrierungsprozess

1. **Gast-Account Registrierung**
    - E-Mail mit Datenraum-Einladung erhalten
    - Button "Registrierung abschließen" klicken
    - Benutzername = E-Mail-Adresse (vorgegeben)
    - Passwort frei wählbar
    - Bestätigung per E-Mail

## 3. Sicherheitsarchitektur

### 3.1 Verschlüsselungsmethoden

### 3.1.1 Transportverschlüsselung

- **Standard**: TLS 1.3 für alle Datenübertragungen
- **Schutz**: Während der Übertragung ("Encryption-in-Transit")
- **Anwendung**: Automatisch für alle Datenräume

### 3.1.2 Serverseitige Verschlüsselung

- **Standard**: AES-256 Verschlüsselung
- **Speicherung**: Verschlüsselt auf Server ("Encryption-at-Rest")
- **Anwendung**: Für alle Datenräume

### 3.1.3 Ende-zu-Ende-Verschlüsselung (Optional)

- **Aktivierung**: Manuell pro Datenraum
- **Verschlüsselung**: Direkt im Browser mit SecuPass
- **Zero-Knowledge-Prinzip**: FTAPI hat keinen Zugriff auf Inhalte
- **Voraussetzung**: SecuPass-Key erforderlich
```

```
### 3.2 SecuPass-Verwaltung

### 3.2.1 SecuPass-Einrichtung

1. Benutzerverwaltung öffnen (rechts oben)
2. "SecuPass einrichten" klicken
3. SecuPass festlegen und bestätigen

### 3.2.2 SecuPass-Eigenschaften

- Sicherheitspasswort für Ver-/Entschlüsselung
- Einmalige Festlegung
- **WICHTIG**: Kann nicht zurückgesetzt werden
- Bei Verlust kein Zugriff auf E2E-verschlüsselte Datenräume

### 3.3 Compliance und Datenschutz

- **DSGVO-konform**: Vollständige Compliance
- **BSI-Standards**: Verschlüsselung nach BSI-Vorgaben
- **Datenhaltung**: 100% in Deutschland
- **Rechenzentrum**: Deutscher Betreiber

## 4. Funktionale Requirements

### 4.1 Datenraum-Management

### 4.1.1 Datenraum-Erstellung

- Neue Datenräume anlegen
- Namen und Beschreibung vergeben
- Verschlüsselungsoptionen wählen
- Initiale Zugriffsrechte festlegen

### 4.1.2 Datenraum-Struktur

- **Hierarchische Organisation**:
    - Datenräume (oberste Ebene)
    - Unterordner (ein-/ausklappbar)
```

```
    - Dateien
- **Sortieroptionen**:
    - Name (alphabetisch)
    - Dateigröße
    - Änderungsdatum

### 4.1.3 Datenraum-Verwaltung

- Datenräume umbenennen
- Beschreibungen ändern
- Löschfristen festlegen
- Datenräume löschen (nur Besitzer)

### 4.2 Datei-Management

### 4.2.1 Upload-Funktionen

- **Methoden**:
    - Drag & Drop
    - Upload-Button
    - Mehrfachauswahl möglich
- **Dateigröße**: Bis 100 GB pro Datei
- **Dateitypen**: Keine Einschränkungen (konfigurierbar)

### 4.2.2 Download-Funktionen

- Einzeldateien herunterladen
- Mehrfachauswahl für Download
- Ordner als ZIP herunterladen

### 4.2.3 Datei-Operationen

- Dateien verschieben
- Dateien löschen
- Dateien umbenennen
- Dateiversionierung

### 4.2.4 PDF-Kollaboration
```

```
- **PDF-Viewer im Browser**
- **Anmerkungen**: Direkt im Dokument
- **Kommentare**: Für andere Mitarbeiter sichtbar
- **Speicherung**: Automatisch mit Dokument

### 4.3 Zugriffsrollen und Berechtigungen

### 4.3.1 Rollendefinitionen

**Betrachter (ohne Herunterladen)**

- Datei ansehen
- Keine Download-Berechtigung
- Keine Bearbeitungsrechte

**Betrachter**

- Datei ansehen
- Datei herunterladen
- Keine Bearbeitungsrechte

**Bearbeiter**

- Datei ansehen
- Datei herunterladen
- Datei hochladen
- Datei verschieben
- Datei löschen
- Ordner erstellen
- Ordner löschen

**Besitzer**

- Alle Bearbeiter-Rechte
- Datenraum löschen
- Zugriffe verwalten
- Neue Nutzer einladen
- Rollen ändern
- Übersicht über Datei-Upload-Events und -zugriffe
```

```
### 4.3.2 Rechtevergabe

- E-Mail-basierte Einladung
- Rollenzuweisung bei Einladung
- Nachträgliche Rollenänderung möglich
- Mehrfachzuweisung von Rollen

### 4.4 Transparenz und Nachvollziehbarkeit

### 4.4.1 Audit Trail

- **Protokollierte Aktivitäten**:
    - Datei-Upload
    - Datei-Download
    - Datei-Ansicht
    - Änderungen
    - Löschungen
    - Zugriffsverwaltung
- **Informationen**:
    - Benutzer
    - Zeitstempel
    - Aktion
    - Betroffene Dateien/Ordner
- **Zugriff**: Nur für Besitzer sichtbar

### 4.4.2 Dateiversionierung

- Automatische Versionierung bei Änderungen
- Versionsverlauf einsehbar
- Alte Versionen wiederherstellen
- Versionsnummern und Zeitstempel

### 4.4.3 Aktivitätsbenachrichtigungen

- E-Mail-Benachrichtigungen bei:
    - Neuen Uploads
    - Änderungen
    - Freigaben
```

```
    - Downloads (optional)
- Konfigurierbare Benachrichtigungseinstellungen

### 4.5 Automatisierung und Regelwerk

### 4.5.1 Löschfristen

- **Automatische Löschung**: Nach festgelegtem Zeitraum
- **Konfiguration**: Pro Datenraum oder global
- **Compliance**: Unterstützung von DSGVO-Aufbewahrungsfristen
- **Benachrichtigung**: Vor Löschung (optional)

### 4.5.2 Zugriffsbeschränkungen

- Zeitbasierte Zugriffe (Ablaufdatum)
- IP-Beschränkungen (Admin-Funktion)
- Download-Limits (optional)

## 5. Administrative Requirements

### 5.1 Admin-Konsole

### 5.1.1 Zentrale Verwaltung

- Übersicht aller Datenräume
- Keine direkten Zugriffe auf Inhalte erforderlich
- Globale Einstellungen

### 5.1.2 Verfügbare Informationen

- **Datenraum-Details**:
    - Name des Datenraums
    - Besitzer (Liste)
    - Ende-zu-Ende-Verschlüsselung (Ja/Nein)
    - Anzahl Mitglieder
    - Anzahl Dateien
    - Gesamtdateigröße

### 5.1.3 Admin-Aktionen
```

```
- Besitzer-Rechte vergeben
- Datenräume löschen
- Berichte generieren
- Speicherplatz verwalten

### 5.2 Benutzerverwaltung

### 5.2.1 Gruppenverwaltung

- Benutzergruppen erstellen
- Rechte pro Gruppe definieren
- Benutzer zu Gruppen hinzufügen
- Mehrfachgruppenzugehörigkeit

### 5.2.2 Berechtigungsprinzipien

- **Segregation of Duties**: Aufgabentrennung
- **Principle of Least Privilege**: Minimale Berechtigung
- **Principle of Need to Know**: Notwendiges Wissen

### 5.2.3 Berechtigungsvererbung

- **Kumulative Berechtigungen**:
    - Whitelist/Blacklist für Dateitypen
    - IP-Adressen-Beschränkungen
    - Sicherheitsstufen
- **Prioritäre Berechtigungen**:
    - Nach Gruppenrang
    - Höhere Gruppe überschreibt niedrigere

### 5.3 Reporting und Monitoring

### 5.3.1 Reports

- Nutzungsstatistiken
- Speicherverbrauch
- Aktivitätsprotokolle
- Compliance-Reports
```

```
### 5.3.2 Monitoring

- Echtzeit-Überwachung
- Kapazitätsplanung
- Performance-Metriken
- Sicherheitsereignisse

### 5.4 Integration und APIs

### 5.4.1 REST API

- Vollständige API-Dokumentation
- Authentifizierung via Token
- CRUD-Operationen für Datenräume
- Benutzerverwaltung via API

### 5.4.2 Systemintegrationen

- **Microsoft Teams Integration**
- **SecuFlows-Schnittstelle**
- **SSO (Single Sign-On)**
- **Zwei-Faktor-Authentifizierung (2FA)**

## 6. Technische Requirements

### 6.1 Performance

- **Dateigröße**: Bis 100 GB pro Datei
- **Speicher**: 300 GB inklusive (erweiterbar)
- **Unlimitierter Speicher**: Auf Wunsch verfügbar
- **Gleichzeitige Nutzer**: Skalierbar

### 6.2 Verfügbarkeit

- **Uptime**: 99% Verfügbarkeit
- **Wartungsfenster**: Angekündigt
- **Backup**: Automatische Sicherungen
- **Disaster Recovery**: Implementiert
```

```
### 6.3 Browser-Kompatibilität

- Keine Plugins erforderlich
- HTML5-Standard
- Responsive Design
- Progressive Web App fähig

## 7. Benutzerfreundlichkeit

### 7.1 User Interface

- **Intuitive Oberfläche**: Keine Schulung erforderlich
- **Übersichtliche Dateiverwaltung**: Direkt im Browser
- **Drag & Drop**: Für alle Dateioperationen
- **Kontextmenüs**: Rechtsklick-Funktionen

### 7.2 Onboarding

- **Schnelles Onboarding**: Keine Installation
- **Guided Tours**: Interaktive Einführung
- **Help Center**: Integrierte Hilfe
- **Video-Tutorials**: Verfügbar

### 7.3 Anpassung

- **Corporate Design**: CI-konforme Oberfläche
- **Mehrsprachigkeit**: Deutsch, Englisch, Französisch
- **Benutzerdefinierte Felder**: Erweiterbar
- **White-Label**: Option verfügbar

## 8. Support und Wartung

### 8.1 Support-Optionen

- **Deutscher Support**: Verfügbar
- **Support-Kanäle**: E-Mail, Telefon
- **SLA**: Definierte Reaktionszeiten
- **Dokumentation**: Umfassend
```

```
### 8.2 Wartung

- **Updates**: Automatisch
- **Keine Downtime**: Bei Updates
- **Feature-Releases**: Regelmäßig
- **Security-Patches**: Sofort

## 9. Implementierung

### 9.1 Rollout

- **Implementierungszeit**: Innerhalb von 24h
- **Keine IT-Ressourcen**: Erforderlich
- **Cloud-basiert**: Sofort verfügbar
- **Skalierbar**: Nach Bedarf

### 9.2 Migration

- **Datenimport**: Unterstützt
- **Bulk-Upload**: Verfügbar
- **Metadaten**: Erhaltung möglich
- **Rechte-Migration**: Unterstützt

## 10. Lizenzierung

### 10.1 Lizenzmodell

- **Faire Lizenzierung**: Für interne und externe Nutzer
- **Keine versteckten Kosten**: Transparente Preise
- **Skalierbar**: Nach Nutzerzahl
- **Speicher**: Flexibel erweiterbar

### 10.2 Inkludierte Leistungen

- 300 GB Speicher
- Unbegrenzte Gast-Accounts
- Alle Funktionen
- Support inklusive
```

```
## 11. Sicherheitsprinzipien und Best Practices

### 11.1 Datenschutz

- Ende-zu-Ende-Verschlüsselung für kritische Daten
- Regelmäßige Zugriffsprüfungen
- Minimale Berechtigungen vergeben
- Löschfristen implementieren

### 11.2 Compliance

- DSGVO-konforme Prozesse
- Audit-Trail aktivieren
- Regelmäßige Reports
- Dokumentation pflegen

### 11.3 Operationale Sicherheit

- Starke Passwörter erzwingen
- 2FA aktivieren
- IP-Beschränkungen nutzen
- Regelmäßige Schulungen

## 1. Datenraum-Verwaltung

### 1.1 Datenraum erstellen

- Neue virtuelle Datenräume anlegen
- Namen und Beschreibung festlegen
- Verschlüsselungsoptionen wählen (Standard oder Ende-zu-Ende)

### 1.2 Datenraum-Struktur

- Hierarchische Ordnerstruktur innerhalb der Datenräume
- Ordner erstellen, umbenennen und löschen
- Ein- und ausklappbare Unterordner für bessere Übersicht

### 1.3 Datenraum löschen
```

```
- Datenräume können nur vom Besitzer gelöscht werden
- Automatische Löschfristen konfigurierbar

## 2. Datei-Management

### 2.1 Datei-Upload

- Drag & Drop Funktion
- Upload-Button für Dateiauswahl
- Mehrfachauswahl von Dateien möglich
- Dateien bis 100 GB unterstützt

### 2.2 Datei-Download

- Einzelne Dateien herunterladen
- Mehrere Dateien auf einmal herunterladen
- Ordner als ZIP-Datei herunterladen

### 2.3 Datei-Operationen

- Dateien verschieben zwischen Ordnern
- Dateien löschen
- Dateien umbenennen
- Versionierung von Dateien

### 2.4 Datei-Ansicht

- Dateien direkt im Browser ansehen (ohne Download)
- PDF-Viewer integriert
- Unterstützung verschiedener Dateiformate

## 3. Benutzerverwaltung und Zugriffe

### 3.1 Benutzer-Accounts

- **Reguläre Accounts**: Vollwertige Benutzer mit eigenen Datenräumen
- **Gast-Accounts**: Kostenlose Accounts für externe Nutzer mit eingeschränkten Rechte
```

```
### 3.2 Registrierung

- E-Mail-basierte Registrierung
- Gast-Accounts werden automatisch bei Einladung erstellt
- Passwort selbst festlegen

### 3.3 Benutzer zu Datenräumen einladen

- Einladung per E-Mail versenden
- Rolle bei Einladung festlegen
- Mehrere Benutzer gleichzeitig einladen

## 4. Rollen und Berechtigungen

### 4.1 Rollendefinitionen

**Betrachter (ohne Download)**

- Dateien nur ansehen
- Kein Download möglich

**Betrachter (mit Download)**

- Dateien ansehen
- Dateien herunterladen

**Bearbeiter**

- Dateien ansehen und herunterladen
- Dateien hochladen
- Dateien verschieben und löschen
- Ordner erstellen und löschen

**Besitzer**

- Alle Bearbeiter-Rechte
- Datenraum löschen
- Benutzer einladen und entfernen
- Rollen ändern
```

```
- Audit-Trail einsehen

### 4.2 Rechteverwaltung

- Rollen pro Datenraum vergeben
- Nachträgliche Änderung von Rollen
- Benutzer aus Datenraum entfernen

## 5. Sicherheitsfunktionen

### 5.1 Verschlüsselung

- **Transportverschlüsselung**: TLS für alle Übertragungen
- **Serverseitige Verschlüsselung**: AES-256 für gespeicherte Daten
- **Ende-zu-Ende-Verschlüsselung**: Optional pro Datenraum aktivierbar

### 5.2 SecuPass

- SecuPass einrichten für Ende-zu-Ende-Verschlüsselung
- SecuPass in Benutzerverwaltung festlegen
- Warnung: SecuPass kann nicht zurückgesetzt werden

### 5.3 Authentifizierung

- Zwei-Faktor-Authentifizierung (2FA) optional
- SMS-TAN Verfahren
- Single Sign-On (SSO) via SAML

## 6. Kollaboration

### 6.1 PDF-Bearbeitung

- PDFs direkt im Browser annotieren
- Kommentare zu PDFs hinzufügen
- Anmerkungen für andere Benutzer sichtbar
- Änderungen automatisch speichern

### 6.2 Benachrichtigungen
```

```
- E-Mail-Benachrichtigungen bei neuen Uploads
- Benachrichtigungen bei Änderungen
- Aktivitätsbenachrichtigungen konfigurierbar

## 7. Transparenz und Nachvollziehbarkeit

### 7.1 Audit Trail

- Alle Aktivitäten werden protokolliert:
    - Datei-Uploads
    - Downloads
    - Ansichten
    - Änderungen
    - Löschungen
    - Rechtevergaben
- Zeitstempel und Benutzer werden erfasst
- Nur für Besitzer einsehbar

### 7.2 Aktivitätsübersicht

- Übersicht über alle Datei-Upload-Events
- Zugriffe auf Dateien nachvollziehen
- Chronologische Darstellung

## 8. Administration

### 8.1 Admin-Konsole

- Zentrale Verwaltung aller Datenräume
- Übersicht ohne direkten Zugriff auf Inhalte
- Folgende Informationen einsehbar:
    - Name des Datenraums
    - Liste der Besitzer
    - Ende-zu-Ende-Verschlüsselung (Ja/Nein)
    - Anzahl Mitglieder
    - Anzahl Dateien
    - Gesamtdateigröße

### 8.2 Admin-Funktionen
```

```
- Besitzer-Rechte vergeben
- Datenräume löschen
- Globale Einstellungen verwalten

## 9. Gruppenverwaltung

### 9.1 Gruppen anlegen

- Neue Gruppen erstellen
- Gruppenname und Beschreibung festlegen

### 9.2 Benutzer zu Gruppen zuweisen

- Benutzer werden bei Anlage einer Gruppe zugewiesen
- Benutzer per E-Mail oder Benutzername hinzufügen
- Übersicht der Gruppenmitglieder

### 9.3 Gruppenberechtigungen

- Features pro Gruppe aktivieren/deaktivieren
- Lizenzfreie und lizenzpflichtige Features unterscheiden
- Sicherheitseinstellungen pro Gruppe

### 9.4 Einschränkungen pro Gruppe

- Maximale Anhangsgröße für WebUpload festlegen
- Maximale Segmentgröße für Uploads
- Whitelist für Empfänger (Domains wie *@company.com)

## 10. Automatisierung

### 10.1 Löschfristen

- Automatische Löschfristen pro Datenraum
- Automatische Bereinigung konfigurieren
- DSGVO-konforme Aufbewahrungsfristen

### 10.2 Automatische Prozesse
```

```
- Virenscans beim Upload (G DATA Scanner)
- Automatische Benachrichtigungen
- Compliance-Prüfungen

## 11. Zugriffsmöglichkeiten

### 11.1 Browserbasiert

- Keine lokale Installation erforderlich
- Zugriff über alle gängigen Browser
- Responsive Design für mobile Geräte

### 11.2 Geräteunterstützung

- Desktop/Laptop
- Tablet
- Smartphone
- Plattformunabhängig

## 12. Integration

### 12.1 Microsoft Teams Integration

- SecuRooms in Teams einbinden

### 12.2 API-Schnittstelle

- REST API für Automatisierung
- Programmatischer Zugriff auf Funktionen

### 12.3 SecuFlows-Schnittstelle

- Integration mit FTAPI SecuFlows
```

A.2.2. SecuMails

```
## **1. Produktübersicht**

### **1.1 Produktbeschreibung**

- **Name**: FTAPI SecuMails
- **Zweck**: Sichere Verschlüsselung und Übertragung von E-Mails und Dateien direkt im E-M
- **Vision**: "Securing Digital Freedom"
- **Zielgruppe**: Unternehmen, Behörden, Gesundheitswesen, HR-Abteilungen

### **1.2 Kernfunktionalität**

- Sicherer Ad-hoc-Versand und -Empfang von Nachrichten und Dateien
- Dateien jeder Größe (bis 100 GB) sicher per Mail versenden
- Ende-zu-Ende-Verschlüsselung nach dem Zero-Knowledge-Prinzip
- Integration in bestehende E-Mail-Systeme

## **2. Systemzugriff und Nutzungsmöglichkeiten**

### **2.1 Zugriffswege**

1. **Web-Interface**
    - Zugriff über alle gängigen Internet-Browser (aktuelle Versionen)
    - Unterstützte Browser: Google Chrome, Microsoft Edge, Safari, Firefox
    - Optimiert für alle Endgeräte: Desktop, Tablet, Smartphone (\geq 360 x 640 px)
    - Keine lokale Installation erforderlich
2. **Microsoft Outlook Add-In** (kostenpflichtige Erweiterung)
    - Systemanforderung: Microsoft Outlook 2016 oder neuer
    - Nahtlose Integration in die gewohnte Outlook-Umgebung
    - Kein Medienbruch beim Versand
3. **SubmitBox** (digitaler Briefkasten) - kostenpflichtige Erweiterung
    - Sicherer Kanal für externe Einreichungen
    - Keine Registrierung für externe Sender erforderlich

## **3. Sicherheitsarchitektur**

### **3.1 Verschlüsselungstechnologie**
```

```
### **3.1.1 SecuPass-Technologie**

- Hybride Verschlüsselung mit AES-256-Bit
- Datenverschlüsselung: Symmetrisches AES-Verfahren
- Schlüsselaustausch: Asymmetrisches RSA-Schlüsselpaar
- RSA-Schlüssel mit OAEP (Optimal Asymmetric Encryption Padding)
- Schlüssellänge: 4096 Bit
- Automatischer Schlüsselaustausch ohne manuelle Zertifikatseinspielung

### **3.1.2 Zero-Knowledge-Prinzip**

- Ende-zu-Ende-Verschlüsselung
- RSA-Schlüsselpaar wird am Client generiert
- Privater RSA-Schlüssel wird mit SecuPass-Passwort verschlüsselt
- Nur verschlüsselte Form wird auf Server gespeichert
- FTAPI hat zu keinem Zeitpunkt Zugriff auf Daten

### **3.1.3 Transportverschlüsselung**

- TLS 1.3 für sichere Übertragung ("Encryption-in-Transit")
- SSL Labs Rating: A+
- Verhindert unbefugten Zugriff während Datenübertragung

### **3.1.4 Krypto-Agilität**

- Flexibles kryptografisches System
- Anpassungsfähig an neue Bedrohungen
- Vorbereitung auf Post-Quantum-Kryptografie
- Speicherung von Verschlüsselungsinformationen für verschiedene Algorithmen

### **3.2 Sicherheitsstufen**

### **Sicherheitsstufe 1 - Sicherer Link**

- **Verschlüsselung**: Transportverschlüsselung (TLS)
- **Zugriff**: Jeder mit Link kann Dateien herunterladen
- **Account erforderlich**: Nein
- **Anwendungsfall**: Unkritische Daten, Ausschreibungsunterlagen, Software-Updates
```

```
- **Empfänger-Authentifizierung**: Keine

### **Sicherheitsstufe 2 - Sicherer Link + Login**

- **Verschlüsselung**: Transportverschlüsselung (TLS)
- **Zugriff**: Nur mit FTAPI-Account
- **Account erforderlich**: Ja (automatische Gast-Account-Erstellung möglich)
- **Anwendungsfall**: Daten für bestimmte Empfänger
- **Optional**: Doppelt-Authentifizierte-Registrierung (SMS-Code)

### **Sicherheitsstufe 3 - Sicherer Link + Login + verschlüsselte Dateien**

- **Verschlüsselung**: Ende-zu-Ende-Verschlüsselung für Dateien
- **Zugriff**: FTAPI-Account + SecuPass-Key erforderlich
- **Account erforderlich**: Ja
- **Anwendungsfall**: Sensible/unternehmenskritische Daten, Arbeitsverträge, Gehaltsabrech
- **Besonderheit**: Nachricht bleibt unverschlüsselt sichtbar

### **Sicherheitsstufe 4 - Sicherer Link + Login + verschlüsselte Dateien + verschlüsselte

- **Verschlüsselung**: Vollständige Ende-zu-Ende-Verschlüsselung (Dateien + Nachricht)
- **Zugriff**: FTAPI-Account + SecuPass-Key erforderlich
- **Account erforderlich**: Ja
- **Anwendungsfall**: Höchst sensible Kommunikation, strategische Dokumente
- **Besonderheit**: Gesamter E-Mail-Text ist verschlüsselt

## **4. Funktionale Requirements**

### **4.1 Versand-Funktionen**

### **4.1.1 Outlook Add-In Versand**

1. **E-Mail-Erstellung**
    - Standard E-Mail-Erstellung mit Empfänger, Betreff, Nachricht
    - Anhänge per Drag & Drop oder Büroklammer-Symbol
2. **FTAPI-Versand**
    - Button "Mit FTAPI versenden" in Menüleiste
    - Automatische sichere Übertragung der Anhänge
3. **Download-Button Integration**
```

```
    - Optional: Download-Button direkt in E-Mail einfügen
    - Alternative: Automatische Platzierung über Signatur
4. **Einstellungen**
    - Auswahl der Sicherheitsstufe (1-4)
    - Festlegung der Gültigkeitsdauer für Downloads
    - Admin kann Vorgaben definieren ("Security-by-Default")

### **4.1.2 Web-Interface Versand**

1. **Neue Zustellung erstellen**
    - Eingabe von Empfänger, Betreff, Nachricht
2. **Datei-Upload**
    - Drag & Drop Funktionalität
    - "Dateien anhängen" Button
    - Maximale Dateigröße: 100 GB
3. **Sicherheitseinstellungen**
    - Wahl der Sicherheitsstufe
    - Gültigkeitsdauer festlegen
4. **Versand**
    - "Mit FTAPI versenden" Button

### **4.2 Empfangs-Funktionen**

### **4.2.1 Outlook Add-In Empfang**

1. **E-Mail-Empfang**
    - Zustellung im normalen E-Mail-Postfach
    - Sichtbar: Absender, Betreff, Dateinamen, Nachrichtentext
2. **Entschlüsselung bei Stufe 4**
    - Button "Mail entschlüsseln" in Menüleiste
    - Entschlüsselung des Nachrichtentexts
3. **Download-Optionen**
    - "Herunterladen" Button in Menüleiste → Download in Outlook
    - Download-Link in E-Mail → Weiterleitung zum Browser
    - "Speichern unter" Option für alternativen Speicherort

### **4.2.2 Browser-basierter Empfang**

- Sicherer Download-Link in E-Mail
```

```
- Je nach Sicherheitsstufe weitere Authentifizierung nötig
- Download über Web-Interface

### **4.3 SubmitBox-Funktionalität**

### **4.3.1 Grundfunktionen**

- Digitaler Briefkasten für sichere Dateneinreichung
- Keine Registrierung für externe Sender erforderlich
- Einreichung nur mit SubmitBox-Link möglich
- Verschlüsselte Übertragung in allen Sicherheitsstufen

### **4.3.2 Integration**

- **E-Mail-Signatur**: Link zur persönlichen SubmitBox
- **Webseite**: Einbindung des Links
- **Outlook Integration**:
    - Option 1: Einmal gültiges Upload-Ticket versenden
    - Option 2: Permanenter SubmitBox-Link

### **4.3.3 Workflow für Externe**

1. **Ticket-Anforderung**
    - SubmitBox-Link aufrufen
    - E-Mail-Adresse eingeben
    - "Ticket erstellen" klicken
2. **Upload-Link erhalten**
    - E-Mail mit persönlichem Upload-Link
    - Betreff: "SubmitBox Ticket erstellt"
3. **Datei-Upload**
    - Upload-Link öffnen
    - Dateien per Drag & Drop oder Büroklammer hinzufügen
    - Nachricht eingeben
    - "Abschicken" klicken
4. **Bestätigungen**
    - Einreichungsbestätigung per E-Mail
    - Download-Bestätigung wenn Empfänger Dateien herunterlädt

### **4.3.4 Kontrollfunktionen**
```

```
- Optionales White- und Blacklisting
- Volle Kontrolle über erlaubte Einreichungen

### **4.4 Benachrichtigungen und Tracking**

### **4.4.1 Download-Bestätigungen**

- Automatische Benachrichtigung nach erfolgreichem Download
- Revisionssichere Dokumentation
- Transparenz über Empfangsstatus

### **4.4.2 Status-Tracking**

- Überblick über versendete Dateien
- Empfangsstatus in Echtzeit
- Reporting-Funktionen für Administratoren

### **4.5 Datenmanagement**

### **4.5.1 Löschfristen**

- Individuell festlegbare Aufbewahrungsfristen
- Automatische Löschung nach Ablauf
- Kein Zugriff nach Löschung möglich

### **4.5.2 Dateigröße**

- Maximale Dateigröße: 100 GB
- Keine Einschränkung bei Anzahl der Dateien
- Optimiert für große Datenmengen

## **5. Administrative Requirements**

### **5.1 Benutzerverwaltung**

- Zentrale Verwaltung über Admin-Oberfläche
- Lizenzverwaltung für Nutzer
- Rechtevergabe und Rollenverwaltung
```

```
### **5.2 Security-by-Default**

- Organisationsweite Vorgabe von Sicherheitsstufen
- Erzwingung bestimmter Verschlüsselungsstandards
- Automatische Regeln für Verschlüsselung

### **5.3 Compliance-Features**

- DSGVO-konforme Datenverarbeitung
- BSI-konforme Verschlüsselungsstandards
- Revisionssichere Protokollierung

### **5.4 Reporting**

- Detailliertes Reporting über Admin-Oberfläche
- Analyse des Anwenderverhaltens
- Ereignisprotokolle (Login-Zeiten, Aktivitäten)
- Export als HTML, PDF oder XLS

### **5.5 Integration**

- REST API für Systemintegration
- SMTP/IMAP Unterstützung
- Active Directory/LDAP Anbindung
- Single Sign-On (SSO) Unterstützung

## **6. Technische Requirements**

### **6.1 Client-Anforderungen**

- **Browser**: Aktuelle Versionen von Chrome, Edge, Safari, Firefox
- **Outlook**: Version 2016 oder höher
- **Bildschirmauflösung**: Minimum 360 x 640 px
- **Internetverbindung**: Stabile Verbindung erforderlich

### **6.2 Server-Infrastruktur**

- Cloud-basierte Lösung
```

```
- Hosting in deutschem Rechenzentrum (SysEleven)
- Kubernetes-Container-Architektur
- 99% Verfügbarkeit
- Geo-redundante Datenspeicherung

### **6.3 Sicherheitsstandards**

- ISO 27001 Zertifizierung
- BSI C5 Auditierung
- Regelmäßige Penetrationstests
- Secure Development Lifecycle

## **7. Benutzerfreundlichkeit**

### **7.1 User Experience**

- Intuitive Benutzeroberfläche
- Keine technischen Vorkenntnisse erforderlich
- Gewohnte E-Mail-Umgebung beibehalten
- Responsive Design für alle Geräte

### **7.2 Anwenderunterstützung**

- Interaktive Produkt-Touren
- Kurzanleitungen und Dokumentation
- Help Center mit FAQ
- Deutscher Admin-Support

### **7.3 Mehrsprachigkeit**

- Deutsche und englische Oberfläche
- Weitere Sprachen konfigurierbar
- Automatische Spracherkennung

## **8. Performance-Requirements**

### **8.1 Übertragungsgeschwindigkeit**

- Optimiert für große Dateien
```

```
- Parallele Upload-Streams
- Resumable Uploads bei Verbindungsabbruch

### **8.2 Skalierbarkeit**

- Unbegrenzte Anzahl externer Nutzer
- Elastische Cloud-Infrastruktur
- Automatische Lastverteilung

## **9. Lizenzierung und Kosten**

### **9.1 Basis-Lizenz**

- Web-Interface Zugang
- Grundfunktionen SecuMails

### **9.2 Kostenpflichtige Erweiterungen**

- Outlook Add-In
- SubmitBox Funktionalität
- Erweiterte Admin-Features
- API-Zugriff

## **10. Migration und Implementierung**

### **10.1 Implementierung**

- Schnelle und einfache Einrichtung
- Keine aufwendige Infrastruktur-Änderung
- Schrittweise Einführung möglich

### **10.2 Schulung**

- Personalisiertes Onboarding
- Schulungsmaterialien
- Customer Success Team Begleitung

### **10.3 Support**
```

```
- Deutschsprachiger Support
- SLA-basierte Reaktionszeiten
- Technische Dokumentation

# Use Cases:

# FTAPI SecuMails - Detaillierte Requirements und Use Cases

## 1. Produktübersicht

FTAPI SecuMails ist eine Lösung für den sicheren Versand und Empfang von Nachrichten und D

## 2. Systemanforderungen

### 2.1 Technische Requirements

### Unterstützte Umgebungen

- **Web-Browser** (jeweils aktuelle Version):
    - Google Chrome
    - Microsoft Edge
    - Safari
    - Mozilla Firefox
- **Microsoft Outlook Add-in**:
    - Microsoft Outlook 2016 oder neuer
- **Mobile Endgeräte**:
    - Optimiert für Desktop, Tablet und Smartphone
    - Mindestauflösung: 360 x 640 px

### Verschlüsselung

- AES 256-Bit-Verschlüsselung
- Transport-Verschlüsselung via TLS 1.3
- Ende-zu-Ende-Verschlüsselung nach Zero-Knowledge-Prinzip
- Verschlüsselung nach BSI-Standards

## 3. Funktionale Requirements

### 3.1 Versand-Funktionen
```

```
### FR-001: Dateigrößen-Handling

- System MUSS Dateien bis zu 100 GB verarbeiten können
- System MUSS maximale Anhangsgröße für WebUpload konfigurierbar machen
- System MUSS maximale Segmentgröße für Upload konfigurierbar machen

### FR-002: Sicherheitsstufen

System MUSS vier verschiedene Sicherheitsstufen anbieten:

**Sicherheitsstufe 1 - Sicherer Link**

- Zustellung wird hinter sicherem Link abgelegt
- Kein FTAPI-Account für Download erforderlich
- Anonymer Download möglich (konfigurierbar)

**Sicherheitsstufe 2 - Sicherer Link + Login**

- Empfänger benötigt FTAPI-Account
- Automatische Gast-Account-Erstellung für externe Empfänger
- Empfänger-Authentifizierung erforderlich

**Sicherheitsstufe 3 - Sicherer Link + Login + verschlüsselte Dateien**

- Ende-zu-Ende-Verschlüsselung der Dateien
- SecuPass-Key für Ver-/Entschlüsselung erforderlich
- Zero-Knowledge-Prinzip

**Sicherheitsstufe 4 - Sicherer Link + Login + verschlüsselte Dateien + verschlüsselte Nac

- Ende-zu-Ende-Verschlüsselung von Dateien UND Nachrichtentext
- SecuPass-Key für Ver-/Entschlüsselung erforderlich
- Höchste Sicherheitsstufe für kritische Kommunikation

### FR-003: Versand-Optionen

- System MUSS Versand ohne Anhang ermöglichen (konfigurierbar)
- System MUSS Gültigkeitsdauer für Download-Links konfigurierbar machen
```

```
- System MUSS automatische Löschfristen für Dateien unterstützen
- System MUSS Download-Button in E-Mail integrierbar machen

### FR-004: Benachrichtigungen

- System MUSS automatische Download-Bestätigungen versenden
- System MUSS Versender über erfolgreichen Download informieren
- System MUSS IP-Adressen-Protokollierung ermöglichen (optional)

### 3.2 Empfangs-Funktionen

### FR-005: Antwort-Funktion

- System MUSS "Antwort senden"-Funktion für externe Empfänger bereitstellen
- Externe Empfänger MÜSSEN auf empfangene Zustellungen antworten können

### FR-006: Entschlüsselung

- System MUSS Mail-Entschlüsselung bei Sicherheitsstufe 4 im Outlook Add-in ermöglichen
- System MUSS SecuPass-Verwaltung bereitstellen

### 3.3 Administrative Funktionen

### FR-007: Organisationsweite Einstellungen

- Administrator MUSS Standard-Sicherheitsstufe vorgeben können
- Administrator MUSS Versandregeln organisationsweit festlegen können
- Administrator MUSS Whitelist für Zustellungsempfänger konfigurieren können

### FR-008: Benutzer- und Gruppenverwaltung

- System MUSS Benutzer Gruppen zuweisen können
- System MUSS Berechtigungen und Lizenzen pro Gruppe verwalten
- System MUSS Sicherheitseinstellungen pro Gruppe konfigurierbar machen

### FR-009: Compliance und Reporting

- System MUSS revisionssichere Download-Bestätigungen bereitstellen
- System MUSS Zustellungs-Download-Report generieren können
```

```
- System MUSS DSGVO-konforme Datenverarbeitung gewährleisten

### 3.4 Integration Requirements

### FR-010: Outlook Add-in

- Add-in MUSS nahtlose Integration in Outlook bieten
- Add-in MUSS "Mit FTAPI versenden"-Button bereitstellen
- Add-in MUSS Sicherheitsstufen-Auswahl ermöglichen
- Add-in MUSS Download-Button in E-Mail einfügen können

### FR-011: SubmitBox Integration

- System MUSS SubmitBox für sicheren Datenempfang bereitstellen
- SubmitBox MUSS ohne Registrierung für Externe nutzbar sein
- SubmitBox MUSS als digitaler Briefkasten fungieren

## 4. Detaillierte Use Cases

### 4.1 UC-001: Sicherer Versand via Outlook Add-in

**Akteure:**

- USER A (Versender mit Outlook)
- USER B (Empfänger)

**Vorbedingungen:**

- USER A hat Outlook 2016+ mit installiertem FTAPI Add-in
- USER A ist bei FTAPI registriert und angemeldet

**Hauptszenario:**

1. USER A erstellt neue E-Mail in Outlook
2. USER A fügt Empfänger (USER B), Betreff und Nachrichtentext hinzu
3. USER A fügt Dateien als Anhang hinzu
4. USER A klickt auf "Mit FTAPI versenden" im Add-in
5. System zeigt Sicherheitsstufen-Auswahl
6. USER A wählt Sicherheitsstufe (1-4)
```

```
7. USER A definiert Gültigkeitsdauer (optional)
8. USER A klickt auf "Senden"
9. System verschlüsselt Dateien gemäß gewählter Sicherheitsstufe
10. System generiert sicheren Download-Link
11. USER B erhält E-Mail mit Download-Link
12. USER A erhält Versandbestätigung

**Alternative Szenarien:**

- 4a. USER A fügt Download-Button direkt in E-Mail ein
- 6a. Administrator hat Sicherheitsstufe vorgegeben

### 4.2 UC-002: Empfang mit Sicherheitsstufe 1

**Akteure:**

- USER B (Empfänger ohne FTAPI-Account)

**Hauptszenario:**

1. USER B erhält E-Mail mit sicherem Link
2. USER B klickt auf Download-Link
3. System öffnet Download-Seite im Browser
4. USER B lädt Dateien herunter
5. System sendet Download-Bestätigung an Versender

**Besonderheit:** Kein Login erforderlich, anonymer Download möglich

### 4.3 UC-003: Empfang mit Sicherheitsstufe 2

**Akteure:**

- USER B (Externer Empfänger ohne FTAPI-Account)

**Hauptszenario:**

1. USER B erhält E-Mail mit sicherem Link
2. USER B klickt auf Download-Link
3. System leitet zu Registrierungsseite weiter
```

```
4. System erstellt automatisch Gast-Account für USER B
5. USER B gibt E-Mail-Adresse ein
6. USER B erstellt Passwort
7. System sendet Bestätigungs-E-Mail
8. USER B meldet sich mit Gast-Account an
9. USER B lädt Dateien herunter
10. System sendet Download-Bestätigung an Versender

### 4.4 UC-004: Ende-zu-Ende-verschlüsselter Versand (Stufe 3)

**Akteure:**

- USER A (Versender mit SecuPass)
- USER B (Empfänger)

**Vorbedingungen:**

- USER A hat SecuPass eingerichtet
- Sensible Dateien (z.B. Verträge, Finanzdaten)

**Hauptszenario:**

1. USER A wählt Sicherheitsstufe 3 beim Versand
2. System verschlüsselt Dateien mit USER A's SecuPass-Key
3. USER B erhält verschlüsselte Zustellung
4. USER B richtet SecuPass ein (falls noch nicht vorhanden)
5. System informiert USER A über SecuPass-Aktivierung
6. USER A erteilt Freigabe für USER B
7. USER B kann Dateien mit eigenem SecuPass entschlüsseln
8. Dateien bleiben während gesamtem Prozess Ende-zu-Ende verschlüsselt

### 4.5 UC-005: Vollverschlüsselte Kommunikation (Stufe 4)

**Akteure:**

- USER A (Versender mit SecuPass)
- USER B (Empfänger mit SecuPass)

**Hauptszenario:**
```

```
1. USER A verfasst vertrauliche Nachricht mit sensiblen Anhängen
2. USER A wählt Sicherheitsstufe 4
3. System verschlüsselt Nachrichtentext UND Dateien
4. USER B erhält vollständig verschlüsselte E-Mail
5. USER B meldet sich an und gibt SecuPass ein
6. USER B klickt "Mail entschlüsseln" im Outlook Add-in
7. System entschlüsselt Nachrichtentext
8. USER B lädt und entschlüsselt Dateien
9. Kommunikation bleibt Zero-Knowledge (FTAPI hat keinen Zugriff)

### 4.6 UC-006: SubmitBox - Passiver Upload

**Akteure:**

- USER A (SubmitBox-Besitzer)
- USER B (Externer Einreicher)

**Hauptszenario:**

1. USER A integriert SubmitBox-Link in E-Mail-Signatur
2. USER B findet SubmitBox-Link
3. USER B klickt auf SubmitBox-Link
4. System öffnet SubmitBox-Interface
5. USER B gibt eigene E-Mail-Adresse ein
6. USER B klickt "Ticket erstellen"
7. System sendet Upload-Link an USER B
8. USER B öffnet E-Mail mit Upload-Link
9. USER B lädt Dateien hoch und fügt Nachricht hinzu
10. USER B klickt "Abschicken"
11. USER A erhält Benachrichtigung über Einreichung
12. USER B erhält Einreichungsbestätigung

### 4.7 UC-007: SubmitBox - Aktiver Upload via Outlook

**Akteure:**

- USER A (Anforderer mit Outlook)
- USER B (Externer Einreicher)
```

```
**Hauptszenario:**

1. USER A erstellt E-Mail in Outlook
2. USER A klickt auf SubmitBox-Button → "Upload-Button einfügen"
3. System fügt einmalig gültigen Upload-Link in E-Mail ein
4. USER A sendet E-Mail mit FTAPI
5. USER B erhält E-Mail mit persönlichem Upload-Link
6. USER B klickt auf Upload-Link
7. USER B lädt angeforderte Dateien hoch
8. USER A erhält verschlüsselte Dateien in Postfach
9. Beide erhalten Bestätigungen

### 4.8 UC-008: Administratorkonfiguration

**Akteur:**

- ADMIN (Administrator)

**Hauptszenario:**

1. ADMIN navigiert zu Administration → Konfiguration
2. ADMIN konfiguriert Zustellungseinstellungen:
    - Erlaubt/verbietet Zustellungen ohne Anhang
    - Setzt Standard-Sicherheitsstufe
    - Definiert maximale Dateigröße
    - Konfiguriert Löschfristen
3. ADMIN richtet Whitelist für erlaubte Empfänger ein
4. ADMIN aktiviert IP-Adressen-Protokollierung
5. ADMIN konfiguriert CC-Adressen für Compliance
6. Einstellungen gelten organisationsweit

### 4.9 UC-009: Compliance-Reporting

**Akteur:**

- ADMIN/COMPLIANCE-OFFICER

**Hauptszenario:**
```

```
1. ADMIN navigiert zu Berichte
2. ADMIN wählt "Zustellungen Download Report"
3. ADMIN definiert Zeitraum
4. System generiert Report mit:
    - Versender/Empfänger-Informationen
    - Download-Zeitpunkte
    - IP-Adressen (falls aktiviert)
    - Sicherheitsstufen
5. ADMIN exportiert Report für Audit/Compliance

### 4.10 UC-010: SecuPass-Einrichtung

**Akteur:**

- USER (Erstmalige SecuPass-Nutzung)

**Hauptszenario:**

1. USER erhält Ende-zu-Ende verschlüsselte Zustellung
2. System zeigt rotes "!" bei Benutzerkonto
3. USER klickt auf Benutzerkonto-Icon
4. USER wählt "SecuPass einrichten"
5. USER erstellt SecuPass (mit Vorgaben: Länge, Sonderzeichen)
6. USER bestätigt SecuPass
7. System aktiviert Ende-zu-Ende-Verschlüsselung
8. USER kann nun verschlüsselte Inhalte senden/empfangen

**Wichtig:** SecuPass kann NICHT zurückgesetzt werden!

## 5. Sicherheitsanforderungen

### 5.1 Verschlüsselung

- MUSS AES 256-Bit-Verschlüsselung verwenden
- MUSS Zero-Knowledge-Prinzip bei Stufe 3+4 einhalten
- MUSS Krypto-Agilität für zukünftige Standards unterstützen

### 5.2 Authentifizierung
```

```
- MUSS Zwei-Faktor-Authentifizierung unterstützen (optional)
- MUSS Single-Sign-On via SAML unterstützen
- MUSS Brute-Force-Schutz implementieren

### 5.3 Datenschutz

- MUSS DSGVO-konform sein
- MUSS "Made & Hosted in Germany" erfüllen
- MUSS automatische Datenlöschung nach Ablauf unterstützen

## 6. Performance Requirements

- System MUSS Dateien bis 100 GB in angemessener Zeit verarbeiten
- Upload MUSS in Segmenten erfolgen können
- System MUSS für mobile Endgeräte optimiert sein

## 7. Integrations-Requirements

### 7.1 E-Mail-Integration

- MUSS mit Microsoft Outlook 2016+ kompatibel sein
- MUSS Standard-E-Mail-Protokolle unterstützen
- MUSS Download-Links in E-Mails einbetten können

### 7.2 Browser-Kompatibilität

- MUSS mit aktuellen Versionen aller gängigen Browser funktionieren
- MUSS responsive Design für verschiedene Bildschirmgrößen bieten

## 8. Lizenzierung

- Basis-Funktionen (Web-Interface) in Grundlizenz enthalten
- Outlook Add-in als kostenpflichtige Erweiterung
- SubmitBox als kostenpflichtige Erweiterung
- Lizenzierung pro Benutzer/Gruppe
```

A.3. Interviews

A.3.1. Interview Expert A

[Speaker 1] Okay, I think that's fine. So, I'll ask again, can I record this interview and use the transcript at work later? Yes.

Great, thank you. I'll start with the introduction, or what's coming up next. This interview is part of my thesis, so my bachelor thesis.

It's about whether AI can help us make DDD more efficient and architecture design more efficient. We have a large requirement set and a very complex software. The idea was to see if we could use AI to cut out context from Big Ball of Mud, for example.

And how comparable this is to the context we already have, especially in the secure domains or what we have already moved. The idea was to turn Big Ball of Mud into a monolith and to create a secure architecture where we want to go. As I said, we have two cases, one is Secure Rooms and one is Secure Mates.

We already have Secure Rooms, so it's already divided and built up with DDD. And Secure Mates is largely still a Big Ball of Mud, everything still a bit together. As a partner, I generate architecture candidates or suggestions for us on the way.

And your task is, or the idea of the interview would be, to look at what AI has done. I have prepared two questions, but you can also raise questions that you find positive or negative. I have hinted at the whole process a bit.

We have a large set of requirements. With Secure Mates and Secure Rooms, this mainly comes from the product documents. We put it all into AI and generate it together with the AI.

Not automatically, but AI asks us questions, we answer the questions. First of all, we create the language, look at all the events in the requirements, everything that happens in this system. Then we try to cluster it in the context of the building, simply divide it up and take the grids, what makes sense, what can be put together.

Then we generate the core aggregates for the building contexts and try to do a technical architecture mapping. The whole thing with the API endpoints, just to show what it could look like. In the end, we should have a complete domain model, just to see, okay, this is the target architecture, this is where we want to go.

How it looks roughly. This is an example of a set of requirements for Secure Rooms. This takes a lot longer, it's just cut off here.

Let's take a look at the first step, the language. I tried it with Cloud 4.1 Opus, with Gemini and with GCPT. We'll get to that in a minute.

For you, just to see how realistic or how this Ubiquitous Language or this definition works for you. Whether this can match what you already know from the Secure Rooms.

[Speaker 2] Sharing and Core Business Concept. Related Terms, Files, Members, Owner, Encryption. Secure Room and Data Room, exactly the same.

Is this the agreed business?

[Speaker 1] That was what I meant with the AI asks questions and you answer them. And iteratively you get to the point where you say, okay, the definitions are improving with each shift or with each question. We have to do that.

For example, Secure Room is a top-level secure container for files and folders, similar to a shared drive, but with optionality to e-encryption. Member is a user who has accepted an invitation. Secure Room.

Secure Pass, which is a bit more universal, is a personal encryption key for e-encryption, unique per user, and so on. These are the examples of Cloud, of Gemini. It would be similar, but Gemini didn't have any more questions, so the tables are a bit different.

But the definitions are always very similar. Secure Virtual Container for Files and Folders. It is the primary workspace and the boundary for our internal access control.

It would be interesting for me to know how you roughly estimate, I know it's just a short overview, but this ubiquitous language, if you would expect it that way, if you were to try to make individual words, individual terms out of the requirement sets.

[Speaker 2] Yes, funnily enough, why I laughed at the top was the question, that was also one of my first questions when I was dealing with Secure Rooms. No, that was one above. Exactly, is Secure Room and Data Room exactly the same?

That's one of the questions that everyone asks when dealing with Secure Rooms. Because we have Secure Room and Data Room, and the deeper you get into it, the more you learn what it means and where it comes from. But it's interesting that it's one of the first questions that are asked.

It's so close to how we would think, at least on this Secure Room level. I haven't read all of the others, but I've seen the first overview, and it's pretty close to what I would expect, if I were to do this with a person.

[Speaker 1] So the questions...

[Speaker 2] We can read the questions here. Is a Data Room intended for temporary projects only, or can it serve as a permanent archive? Can multiple owners exist for...

The first question is very technical. What about ubiquitous language? It's more the domain language that you develop there, and the technology isn't that important yet.

So it's important, but it's not the focus. So the first question is a bit too technical. But the others are pretty good.

Multiple users, one user. Exist clone from existing ones. That's where a user might ask these questions, or could be interesting for a user, who works in this Secure Room.

I haven't seen all the questions in detail, but many of them are pretty good.

[Speaker 1] Cool. That was interesting. For me it was pretty cool to see, similar to SQL Mails, if a SQL Mail and a Delivery are the same thing.

In the product documents, or in the requirements, between these two terms. The next step was to look at the ubiquitous language, to see what happens in this system. It was always like, you create a Secure Room, and you get a system admin.

You create a Secure Room, the Secure Room aggregates it, validates the creation rules, fires the Secure Room created event, and throws it into the event storage. We have a global event ping. The encryption requirements are checked, if it's an internal encrypted Secure Room.

Initialize the encryption, generate room keys, initialize the encryption, etc. Then the Secure Room is created. This invitation flow, you invite the user, send the invitation, notification, check the Secure Pass status.

If there is no Secure Pass, then you have to fix it. Generate key pair, notify, user ready for access. Then the access is granted.

Then you can re-encrypt the files. Files encrypted, access granted. This member status is added, or this access thing.

In this case, at the end. I also did this with all individual events and individual languages. I also created a Secure Room, inviting a new user to an encrypted Secure Room.

This was the most complicated thing that we did. The process is always similar. My question would be, how helpful or realistic would you see these events?

Would this help you in the development?

[Speaker 2] It depends on who I am in this context. If I'm someone who's new, it's 100 percent useful. I can see what's going on and where the individual steps are.

Parts of the individual components. But if we're at event-storming, where we collect events that happen in the domain, it's still quite technical. If I think back to what you do at event-storming, you come together as domain experts and developers.

You develop the language, the ubiquitous language, but also what the domain looks like. I would expect an event-storming at a professional level. But it all comes from the point of view that we want to digitize an existing technical state, our code in BigBallOfMud.

Then it's okay if it's a bit more technical. But if we were new, we'd talk about it as a professional and meta-level conversation.

[Speaker 1] I have another example from JGPT. Maybe it's more in the direction you'd expect. It's called Dataroom Creation and Setup.

You create a dataroom. You say it happens on command. There's an actor who does it.

There's an aggregator where it's addressed. There's a domain event. There's a dataroom created.

Maybe it's more in the direction you'd expect.

[Speaker 2] That's something I'd stick to a whiteboard and say, okay, look, this is Datoroom and in Datoroom happens, Datoroom is created. And what does it all mean? For me as a domain expert, I would explain, okay, here in Datoroom is created and that's a description of how I would expect it.

[Speaker 1] Okay, okay, okay. More like that.

[Speaker 2] More like that, exactly, yes.

[Speaker 1] Exactly, that was always the question, how do you show or how do you present these results best? But in this case more like in an activity diagram. Because that's a bit too technical for you in this case, do you think?

[Speaker 2] So if I think now as the actor in this event storming, so the developer and domain expert, then it is now, but from my point of view, too technical of how it is represented. Maybe there's even the same thing in there, but I would have now from the point of view of understanding, what you said before, from JetGBT, rather expected in there.

[Speaker 1] Okay, okay, okay. Cool. That's interesting, too.

The next step was somehow to group these things and to cut the tree. And there we also have the co-domains, supporting domains and generic sub-domains. As a co-domain, he has now suggested, okay, we need a SIGROOM context, which contains this SIGROOM lifecycle, the ownership, basic properties, deletion policies, access control context with membership, invitations, roles and permissions, folder permissions and an encryption context, completely separate again, key encryption, key distribution and so on.

Supporting domain by identity context, which has user accounts, which is currently a big ball of mud for us, not really cut out yet. Content context, also again separate, which really holds the files. And an audit context, which basically cuts through all the events, what happens in this SIGROOM and so on.

In engineering, there is somehow an administration context for the administration, user groups, system configurations and so on. An application context, which makes all the e-mail dispatch and e-mail notifications. Now with this suggested domain, you know SIGROOM a little bit, would you mean, what is overlapping, what is now completely different, what would be against your expectations?

[Speaker 2] You mean compared to how we implemented it? Yes, exactly. Yes, I found it interesting that with the encryption context, that it is branched out in its own domain or its own domain, core domain, which is handled by us.

But yes, maybe it is even there, but it is not really set up as a domain. I think it's just different utility classes that do encryption and that would then be as an encryption domain. But I wouldn't do that from the start.

I wouldn't see it as a domain in the DDD context. Of course, domains in the technical

sense are probably already there, because they should take care of the encryption, but it is too distributed for that. But yes, you could maybe see it that way.

Yes, because it is in the front end and it is a encapsulated area, it is somehow already a domain, but not yet worked out enough, I think. Exactly, but the rest, yes, Secure Room Context, Lifecycle is in there. Exactly, Ownership, Basic Properties, Deletion, Deletion Policies, I don't know right now, but Access Control Context is also its own domain, I think, in the Secure Room archive.

That's all a bit mixed up there. So I would have said now that the two domains are the same .

[Speaker 1] But would you think it makes sense to split that up, or would it be more appropriate for you to leave it all in one domain as we have it now? I ask because the other AIs have made similar suggestions. Yes, interesting.

So Onboarding Context, here it is a bit different again, but there is always this IAM, this Access Management Context. That is actually always, here on Potential for a Context, Data Management User and Access Management, to do it all again. That's why I'm asking.

[Speaker 2] Let me think about that. Can we go up to the first picture? Because that's a bit of an Access Control Context.

It's about role management. It's not about Access. Well, that's also role management.

[Speaker 1] We have this Access, this package, which is also basically just Access. I'm missing the name now, but it's similar, so I mean.

[Speaker 2] Yes, so it makes sense. I'm just thinking about what would be the point of dividing it up. So we have Secure Room and it's explicitly about Access Control, so Roles and Permissions for Secure Rooms.

So it definitely depends on each other. Because this is not about Access Control, Roles and Permissions for files in general or in general. Any folder structure or something like that, which you could also use in Secure Mails or which you could use in other domains.

So it's actually just about Secure Room and does it make sense in any case? So it definitely doesn't hurt. I don't see any benefit in the sense that it's a context that is closed off, so that it could be used in other places, because it will never be used in other

places, because it's always Secure Room.

But purely, I'm thinking about whether you could win anything if we would split it up with us. So if we would take it out, we could gain something from it, if we had it in Secure Room with us. But now I can't think of anything so spontaneously, except that it's all cleaner.

But that's also a benefit.

[Speaker 1] In general, it's also fair to say, okay, this Access Control belongs explicitly to Secure Room, so let's just put it together. The AI also has, I think, Cloud and GPT, if I'm not mistaken, actually always a separate user context. I would say, because they are global users, that you do it separately, I think.

But it's also fair to say, so this Access Control of Secure Rooms can be put together. In general, but would you think that these suggested contexts are complete for you? Is something missing?

[Speaker 2] Now I thought for Secure Rooms. Okay, so we don't have anything for ... So we still have Files, which is separate for us.

So ...

[Speaker 1] That would be Content.

[Speaker 2] Ah, Content, ah, Files and Folders. Yes, exactly, okay. Supporting Domain.

Yes, that actually makes sense, because Files, that's still with us. It's Secure Room Files, and then there are Files, Insecure Mails and so on. But to isolate that, at least for Files, it doesn't make sense for Folders, because ...

Or are there Folders, Insecure Mails? I don't know. I don't think so.

So for Files it would make sense. And that's why I think it's actually quite good as a Supporting Domain. Versioning, exactly, PDF, Annotation, Story.

Because then you can have all these things that are underneath, Versioning, PDF, Annotation, Storage, Measurement and so on, you can have everything centrally. And the other Domains can then use it, so to speak. That's why it makes sense.

What else is in Secure Rooms? Except you have the Secure Rooms themselves. You have Roles and Rights with the Users, you also have a Domain.

And Files, Reports. And that's it, I think. And that's why I would have said that, ah yes, Notifications and so on, with Invitations and so on.

Exactly. No, I would have already said that it is complete.

[Speaker 1] Very good. The next step would be to take a look at the Core Aggregates, where the Nets are in this context. I've just done a few with Cloud, actually.

With the others it got a little smaller. Cloud wanted to make it very detailed, for whatever reason. In general, the structure was just, okay, Secure Room Context Aggregate, which is a Root Entity.

Let's take the Secure Room now. The Value Objects that the thing should have, Secure Room Name, Description, Encryption Type, Deletion Policy, Entities, where no one has access, and the Commands, where this Aggregate can display. So, Create Secure Room, Update Secure Room Info, Deletion Policy, Transfer Ownership, and so on.

You can see that here in the diagram. Exactly, so with these commands you get to a Secure Room. Secure Room has these Value Objects and can fire these Events.

If you only looked at Secure Room, and looked at the Events, Created Secure Room Info, Updated Deletion Policy, Transferred Ownership, Secure Room Deleted, Secure Room Archived, would you split it up like this, or would you have expected it if you thought about how you make your Core Aggregates?

[Speaker 2] So, the first few are definitely, they are intuitively correct. Created is, if I now think of Secure Room as an Aggregate, then create the Aggregate, delete the Aggregate, update, these are the ones that come to mind first. Deletion, Ownership, Transferred, Ownership Transferred.

[Speaker 1] That was always a question from the AI, what happens when the Owner is deleted, or the Owner is removed, and Secure Room is orphaned. Then I thought, okay, we have a process, you can promote this Secure Room to another member, or another member as an Owner, so that he can continue to manage this Secure Room. That's this event.

[Speaker 2] Yes, that definitely makes sense. Secure Room, Deleted, Archived. Update was also somewhere, right?

[Speaker 1] Info Updated, exactly. How do you like the representation? Would you rather see it as a diagram, or would that be enough as a textual command?

The Events and the Variants. These Business Rules are enough.

[Speaker 2] You mean the representation at the bottom?

[Speaker 1] Which representation would you prefer? If you were to think, okay, I'll make a...

[Speaker 2] Ah, whether the textual or the... Ah, okay. Good.

For me as a developer, the lower one is nice to read. That's why I think it's actually quite good. If I look at the top...

Yes, that's also good. For me as a developer, I'd say I'd prefer the lower one. But the upper one, which I'm looking at now, also contains the information.

In the Variants, that would be interesting.

[Speaker 1] Yes, that's also written here.

[Speaker 2] Actually, everything is in the lower one, and I think it's actually quite good and nice and clear. You can see it at a glance, if you can read this kind of diagram. But it's also very developer-specific.

It's a bit of an UML representation. Well, not quite, but it's pretty UML-like. And if you've done UML before, you'll find yourself right away in something like that.

[Speaker 1] I'm a bit behind in time, but I'll try to make it a bit faster next time. After that, it was a bit about how do you do that, how do you map that, what services do I need, what endpoints do I need to map that. That was also the suggestion of most AIs.

That's a bit complicated. I'll just switch to this one from GPT. So that we have an application service, that can, for example, create a data room.

And that speaks to this domain model. We have infrastructure things that deal with SQL or other encryption service adapters, whatever. So from the construction point of view, I tried to make it pretty similar to ours.

How would you assess this? Is it helpful? Do you see parallels to our architecture?

And would that help you develop? Or give you a good overview of the code?

[Speaker 2] Application service, domain, what is aggregate? And infrastructure, the

SQL things, and the repository. Yes, exactly, that's also in there.

Yes, that looks pretty good. So you said, that's according to what we have in the data room API. Yes.

Yes, I like that. So I can work with it then.

[Speaker 1] So you see parallels to ours.

[Speaker 2] Yes, I see it with the application service. You have the aggregates in the domain, which you then use for it, to execute your business logic. And in the infrastructure, you have the extra things you need, like database, and notification, and so on.

[Speaker 1] Okay. Let's take a look at the whole thing in a few seconds. We haven't really split up after the BigBall of Money.

I did that with one or three AIs, to see how good it looks. Similarly, a huge requirement set was entered, a ubiquitous language was defined, where we have the key terms, okay, delivery, immutable package containing message, and or attachments that one send cannot be modified. Then, of course, the AI tries to read out almost all states that a delivery can have.

If you answer the questions, you get a relatively long list with all our terms, which we use in SecuMate. Here it suggested with a draft. I left it in, so that there is an unsent delivery.

It's a mutable draft that can be edited before sending. The delivery itself, a mutable send package, that creates access grants for recipients. An access grant, permission for a specific recipient to access a delivery with encryption.

That was similar for all of them, actually. Here we have GIMP and Gemini. Also the designation for the delivery is the core container for a secure transfer.

It consists of a subject message and zero or more attachments, which is sent to one or more recipients. It's exactly the same as with GPT. With GPT, we also have the classic question things.

SecuMate is a secure email that enables encryption, sending and receiving message and text files. Here, SecuMate did it with both. The question was, isn't it the same?

Here's another question. Is SecuMate a specific type of email or container that combines message files? What is it really?

Exactly. You define it and then the AI learns through it. If you put yourself in a position where you have to develop a target architecture and you get this requirement set from SecuMate, it's relatively large and nobody really knows what all SecuMate can do.

There are also many hidden features and what we still have from legacy times. Would it be helpful for you to have a table for the ubiquitous language? Do you think the definition is good?

Do you see added value in this definition?

[Speaker 2] Ubiquitous language is important for the communication between domain experts and developers and for developers themselves, so that everyone talks about the same thing. For me, who is already deep in the domain, I already know things, so we bring a table for things that I don't know yet. Maybe it helps.

Where I see added value for such a table is for people who are new to the domain. It's 100 percent helpful because it's overwhelming when you enter the code and there are 21,000 keywords that are coded by delivery, message and so on. Not necessarily keywords, but terms that are not always clear intuitively.

You can roughly imagine what a message is, but you can't see the whole context without going deep into it. A table like this is good. Also the limitations for some things are good.

Status, for example. Status delivered. What does that mean exactly?

Where does it start? Where does it end? What's before?

What's after? And so on. The transitions and so on.

Sure, you can't show everything in the table, but if you have hints of where to look then such a table is good to enter and then go back again.

[Speaker 1] I was just looking if the LLM has the status but no, I don't think so. At least I didn't notice what it means The second step was again the event storming. I didn't make a diagram, I did very little.

Then it looks like what can ZekuMaze do? With Cloud it was also phase one, it tries to

identify everything. There is a user account created, there is a keypad generated.

Really as a texture. Everything has to be processed. Then the flow is trying to define what happens when, who fires what.

The other LLMs did something similar. They said, proposing and sending from ZekuDelivery the happy path. There is an actor who creates a delivery.

You need a delivery aggregate and the event is created. Then there is a sender who adds an attachment to this delivery. That was the idea that there is a draft object.

Exactly. After doing the event storming from ZekuMaze you understand that LLMs hopefully have a better context and can propose something good. Cloud proposed a lot.

First delivery management context which takes care of the delivery itself. Access control context. Managing who can access what delivery and tracking access.

Supporting key management context. Encryption services context. Accounts and organizations.

That was the idea that several organizations exist in one instance. That doesn't happen here. Licensing for submit box separate.

Notification separate. Order context separate. Compliance and reporting policy context.

Generic infrastructure stuff. That became very detailed for one. For the other it's just an identity and access management context.

Secure messaging context where the deliveries are made. Secure ingestion context where the submit box tickets are handled. For the other third one, GPT, something in between.

They needed delivery management. Identity and access management. Secure encryption.

Submit box handling. Notification service. Audit and compliance.

My question would be how would you assess that? Would you divide it as much as possible? Similar to cloud.

Keep it together. Keep it together. Similar to Gemini.

Or a middle way. What would you consider the most useful?

[Speaker 2] I think this way here I see it as practical. You have here the key aggregates. You have a nice composition of the elements.

Can you go back to Gemini? I found it at the event storming. Was it the first one?

Was it cloud?

[Speaker 1] Yes, exactly.

[Speaker 2] I found the event storming, especially the table on the left, when I think about new features like trust network, where we had to get into SecureMail and had to understand all the steps. It would have been good if he analyzed all the steps without going deeper. If you want to understand an architecture that already exists, that's good.

If you want to build it up, I think JetGBT is the best. You don't go there. Of course, it started with delivery message file and so on.

That's on a level where you could do it with a domain expert. He can talk about delivery, message, file. The rest can be combined to create this context.

The other is very detailed. It's good for understanding an existing thing. I think this is better for a new build.

[Speaker 1] Let's go to GPT and look at the core aggregates. I wonder if you would expect that because submission handling goes away. It can have many submissions when it comes to delivery.

There is also delivery. Every delivery has a second mail between many recipients. He didn't make this distinction between delivery and mail.

Access management, user, account, username, role, status, security, encryption, notify service. Would you expect these aggregates or what do you think?

[Speaker 2] Partly. The right side is not bad with the user and secure pass notification audit log. I think it's intuitive.

You have a notification, you want to create it, send it and so on. You can imagine it as an interface and do the processing. The left side is probably in a different context than ours, but it doesn't fit how our model looks.

I don't think the submit box is that bad if you think of it like that. It would be much more intuitive to have your own entity and then create a submission and update it. That doesn't make sense, but that you don't see it as a different kind of delivery, but as something of your own, I don't think that's bad.

Delivery management with secure mail and delivery and so on, the connections don't match, but in itself, secure mail should probably be the share where the metadata is attached. If you see it as our delivery share, then it's actually not that bad, although I would expect it differently. I would rather see this secure mail or share as aggregate and it can add delivery or add delivery.

I don't know how that makes sense. It doesn't make sense. You create a delivery.

This representation is you go over the delivery and it creates its parent with subject, message and attachment. It's a one-to-one relationship, but it's actually the other way around. The share creates its children who are the deliveries and not the other way around.

That's why the representation isn't that good, but in itself it's understandable how he would come up with doing it that way. You'd have to explain what it looks like and what it is, but then it would be a good way.

[Speaker 1] It depends on how well you answer your questions. If you invest more time to teach it, you'll probably get better results.

[Speaker 2] It's the same with event-storming with people. If you ask domain experts more specific questions or get deeper into it, your results are better.

[Speaker 1] Comparable.

[Speaker 2] Comparable, yes.

[Speaker 1] Exactly. These algorithms are similar with the others. I'd like to take a look at the architecture overview.

I told him we have a hexagonal architecture, he tried to layer it a bit. That's an example where he said, okay, domain layer is the idea of a domain layer capsule. It's core business rules, fully linked to infrastructure.

Which entities in the domain layer, which entities in the application layer, which entities in the infrastructure layer. It wasn't really separated, but just an explanation of how it

works. Back here, he tried to separate it, but fully record it.

It turned out a bit chaotic. In general, if you look at the event mapping and the context suggestions, would it be helpful for you to say, okay, we have our big ball of mud, we want to make modulites out of it, just as shit, okay, let's try to make it conceptual with the help of an AI, to define a self-architecture. Would it be helpful for you to do this process?

[Speaker 2] Yes, it would be very helpful. In the beginning, he already had core and supporting and so on, all the domains. If I were to get started, I would take one of the supporting and work on it with it and get it out, so that you leave the big things in there, with delivery and so on, so that it stays in the big ball of mud.

But what should already stand for itself, so user management or role management or whatever, take it out and put it in a module so that it stands for itself and then take the next supporting and then I would proceed step by step and iterate with it until I get to the actual core with deliveries and so on, because that's the most complex part. And then I would start with those where I would assume that they are a bit more closed. Only despite being part of the big ball of mud at the moment, but probably painless to solve, hopefully.

[Speaker 1] That answers my next question a bit. Do you think that DDD is a good sparring partner? Or that LLMs are good sparring partners?

[Speaker 2] Since I spar a lot with them in my daily work, I think it's quite good. Sometimes they just reach their limits, but I think especially for generic questions or brainstorming or explaining things, it's excellent.

[Speaker 1] Do you think there are risks when we use AI? That you can get caught up in something?

[Speaker 2] My biggest concern is data security. We reveal a lot to them by asking them questions that are part of our business. Of course, they are not customer data, but how do we deal with it?

Of course, you can expose it everywhere so that it's not shared, but my biggest concern is that I tell them too much about my business and then it gets abused in some way. That's where I have my concerns, but other than that, I think it's okay If you secure yourself and don't just copy-paste. If you iterate through things all day and then just take the result that's never good.

You also get a review from your colleagues because you wouldn't just merge all of yours, but someone should still look over it. In that case, you're the first to look over what the AI gives back. In the end, the next one reviews your code again.

That's why we have a few security mechanisms before it goes in and we should keep them.

[Speaker 1] A fully automatic architecture design with AI.

[Speaker 2] I don't know. I can't even imagine a person who is fully involved and fully working. They're faster than we are and they have more resources and can access more things.

But the reasoning is the same as we do, just faster. That's why they can do the same and get things wrong. That's how I imagine it.

If there are some who are smarter than we are, not only faster than we are, then maybe if there are special LLMs for architecture or KIs or chatbots that are only fed and tested and proved, you can say it's 100

[Speaker 1] I found it interesting because I read a few papers. In every paper where they did it fully automated, a similar process, it was always not good. But in the paper where AI was seen as a partner just for reasoning and making suggestions, it was better.

I would have said the same. Do you have any final ideas or suggestions how to make this process better and more efficient? Or what would help you in the development?

[Speaker 2] Yes, good question. If we could feed any AI we want in advance with all the data it needs, so it really knows and then start with a subdomain or supporting domain and then just start. I think you would see what are the gaps and where we need more resources or knowledge or another AI.

I can't say it academically. You have to try it. I don't know if anyone else uses DDD.

I already used it with Trust Network or what I did with Christian for the admin area for Outlook. We also built the new module with DDD. I always sparred with AI.

This is my project. I want to build a hexagonal architecture with DDD. Help me to cut it properly.

It was helpful. But it was something new. I didn't have to cut anything out.

We did it all new. It was okay. Especially because you are in your module and you can design it as you want.

But if you want to solve a big ball of mud, you have to iterate a lot.

[Speaker 1] Okay. Cool. That's it.

I hope it fits. Do you have any questions? Very good.

Then I will finish the recording.

A.3.2. Interview Expert B

[Speaker 2] Again, thank you for taking the time. First of all, I would like to ask if I may record this interview, have it transcribed and then use the transcript in my lecture.

[Speaker 1] With pleasure.

[Speaker 2] Great, thank you. Let's start with the introduction. My thesis is actually about how AI and large language models can help us to make DDD more effective, or to make this architecture design more effective.

How well LLMs identify their own building contexts from a large requirement set and how these differ from manual approaches. If we were to do this manually, as a benchmark we have the Seco Rooms, which are all set up with DDD. It would be interesting to define the goal architecture for the Seco Mails, where we actually want to go, with the help of AI, to make this monolith into a modulite.

The purpose of this interview is to criticize these steps or to share your thoughts. If you notice something, you can always say something in between or ask questions.

[Speaker 1] With pleasure.

[Speaker 2] Basically, how my approach is to generate an architecture with LLMs. LLMs are partners. You use them a bit as a sparing partner. In our case, we give the requirements input.

These are summarized product documents, which the product should know, each of the Seco Rooms. In the first phase, it's about creating a new language, to define what the core business terms are, what these words mean, to create some kind of ambiguity,

because we often use the same terms for the same concepts.

[Speaker 1] So, to use the domain context with the LLM as a context.

[Speaker 2] Exactly. Then, to do an event storming, to see what happens in these requirements, what has to happen in this system. Or in this domain.

Then, to create context, to separate and group everything. Then, I look at how the aggregates are built. What are the core aggregates for this context?

The business rules that these aggregates have to follow. Then, at the end, to do a small technical architecture mapping, where you look at how to do this in our hexagonal pattern. At the end, you should have an architecture candidate, just as a suggestion, where you would like to go.

This probably makes everything a bit clearer, if we look at the first part of the Seco Rooms. These requirements take a lot longer. It's a bit cut off here.

It's just built up. Core functionality, access possibilities, transport keys, server-side Seco Paths. Just a list of what Seco Rooms can do.

Then you take the whole thing with different prompts. The LLM has a role. The role is a senior DDD developer with 10 years of experience and is supposed to ask reasonable questions to go through the individual steps.

Then, together with the LLM, iteratively, step by step, to do this process. You can see it quite well here. First of all, create an adequate language.

Then you enter the requirements. Then you say, hey, cool, here's my suggestion. Seco Room, Data Room, the definition, what it means, what it can be.

Then questions and clarifications needed. Is Seco Room and Data Room exactly the same? Is this the agreed business term?

Are members and users distinct concepts? Can a user exist without being a member of any Seco Room? You answer the questions a bit.

Then you can iteratively improve this table until you finally have a set of terms.

[Speaker 1] It's like a dialogue with the LLM.

[Speaker 2] Exactly. What you would do with a partner or a domain expert. You could

also involve yourself in the whole thing.

Just to establish these terms. Here's an example from Cloud. Seco Room, definition, where it's a top-level secure container for files and folders, similar to a shared drive, but with optional e2e encryption.

Pending member is a user who has been invited but hasn't completed sign-up yet. Member is a user who has accepted an invitation and has an active role in a Seco Room. I did this with GPT, with Gemini.

It was relatively similar. Data Room, Creator, a user account, a guest account, a group, a role, owner, admin, just as terms. And with GPT, exactly the same, this same process.

Data Room, virtual secure storage, space, organization sharing and editing files. The question here would be, how meaningful do you find this? How realistic do you think it is, if you roughly skim these terms, that the LLM has identified?

[Speaker 1] I think it's a really good idea. And that's something where I would say it's a super cool application of the LLM. At the end of the day, the ubiquitous language describes a domain that I want to depict.

And by expanding this language with the LLM, I describe the domain. And I think that's pretty cool. That's a pretty good idea.

Before the interview, I was thinking, how would you do that? Or I thought, wait a minute, first of all, I have to understand the domain. If I do domain-driven design, if it codes something for me, but doesn't understand the domain, then it's going to be difficult.

I think that's really cool. I also find these definitions nice and short and crisp. That's really nice with the related terms.

The question I would ask myself now is, when I think of this whole architecture, then it's already pretty well defined with domain-driven design at a high level. I have my business model at the core, around it I have my application and my infrastructure concern. That's pretty well defined.

Where I think the LLM is really good is when you say you have this ubiquitous language and say, let's get rid of the business objects. Don't worry about databases, don't worry about APIs, and build this logic in the business objects. I think that could be pretty cool.

What I can hardly imagine is, let's build an architecture based on my business logic. Because there are definitely a few things that are pretty prevalent. But other things I could imagine, it would actually have to be a question catalog again, which database do you actually want to use?

That the LLM derives from your business logic, that you should do event-driven persistence or something else. These are also decision-making decisions that not only affect the domain, but also the non-functional requirements that you have for your system. I would be curious to see what comes out of that.

[Speaker 2] We'll get to that in a moment.

[Speaker 1] Great!

[Speaker 2] As you said, in the first step it was about clearing the terms of this domain. The next thing was to do a rough event-storming. What happens in my domain?

I have it in different representation possibilities. It's always a bit difficult to represent that. As an activity diagram, SIGRUM creation for an Ethereum Crypto SIGRUM or user invitation to SIGRUM.

These are examples of the plot. What happens? I invite a user.

Invitations are also very technical definitions at the same time. Down here at Gemini it's actually exactly the same. At GPT it's a bit more textual.

What happens? There's an owner who creates a dataroom. Then there's an aggregate and a dataroom-created event.

Command is this create dataroom. Actor is the owner. Aggregate is the dataroom.

And this domain-event that fires is converted to this event. That's what you do. Then these events are mapped out.

The next step would have been my idea to map or try to direct the LLM. Let's look up here at Cloud. You have the terms and you have the events.

Try to split that up a bit into subdomains. This is a suggestion from Cloud how he would set up this audit context. We have this core domain where we say we have a securum context.

That takes care of securum lifecycle, ownership, basic properties, deletion policies. An

access control context. Encryption context as core, supporting identity with the user accounts, authentication, Secure Pass Management and so on.

Audit content context, where it explicitly does only files and folders, versioning, and generic subdomains where administration and notification and so on. It would be interesting to think you know a bit about securums. Do you think he did it similarly to how we did it?

Or would this division make sense?

[Speaker 1] I think it's really exciting because it's a really cool first draft. What I find very exciting is that it was a good success to distinguish between core support and supporting domain. I would describe the supporting domain as everything that has to do with users.

A supporting domain is always when you say it's a make-or-buy decision. Do you build a user management or do you get one? Roughly speaking, I think it's pretty good.

I would think again about the points below. For example, public-private keys I would actually see in an encryption context and not in an identity context. But there are also solutions where you can buy public and private keys.

In itself, I'm pretty surprised that the system is pretty well organized.

[Speaker 2] That was actually the case with Cloud. That was Gemini. Gemini made it a bit more pragmatic and simply said there is a secure collaboration context, which is this securums domain.

[Speaker 1] It's very extensive.

[Speaker 2] Exactly, an administrative context, an identity access management context and an onboarding context. It's not that detailed. With GPT, it's a bit easier.

It simply said there is data management, user access management context, document and file management and a compliance and retention context for this compliance topic that we have.

[Speaker 1] I think that's the best thing about Cloud.

[Speaker 2] I don't know exactly.

[Speaker 1] No matter which domain you look at, you can talk about what core domains are, how you cut them, how you cut the context. At the end of the day, besides the domain, it's also a bit of a view of the domain. It also depends on what the product says.

For example, if we say our focus is on file transfer and encryption, then you could think about whether the content context is a core domain of ours or whether it's a supporting domain because there are solutions for it and you don't have to write it yourself or you can split it up. But I think it's very interesting that it separates it from this SQL context again because it can abstract it with the files and folders. SQL context is just a context about files and folders as metadata.

No, I think the cloud thing is actually the most charming.

[Speaker 2] Let's stay with Cloud and Aggregate. The next step is to say, okay, let's generate the core aggregates that belong to this context.

[Speaker 1] Oh, that's exciting. That's exciting. I would argue that we don't have any aggregates at all.

We don't have any.

[Speaker 2] He first did it conceptually and said, okay, textually, basically only, what would be the root entity? Well, SQL Room, what value objects does it have? What commands should it be able to do?

Or should it be able to update in the most remote sense? Create, update, set deletion policy, transfer ownership, delete and archive. I don't know where this archive comes from, but apparently it has the following.

Hallucinations. And what events can fly there. SQL Room creates it exactly from this event storing step.

He puts these events in there and tries to map them in. If I visualized it with plant or mail, then something like this would come out. Then we have this SQL Room aggregate.

It has value objects. It can submit events. These commands are how to get to this object.

Creates, modifies, configures, transfers. Now it would of course be helpful to know whether this is somewhat realistic. Whether that would be helpful.

[Speaker 1] I find the invariants here very interesting. SQL Room must always have at least one owner. That's the only thing we also protect.

By saying, for example, that create SQL Room or create data room is something where we make sure that there is always a data room that has at least one member. We create it in the same way. E2E encryption type cannot be changed after creation.

Deletion policy must comply with minimum retention requirement. What I'm missing a little bit here is that this deletion policy also refers to the files and so on. To the content of the SQL Room.

That means you could also draw further invariants and say a file and a folder must not exist without a SQL Room. They can only exist in the context of a SQL Room. But then you would probably start at the top and move it from the supporting domain to the core domain.

I think it's not so bad The thing is, with the aggregates, you can't argue whether the routes we have are really that important. You would have to clarify these invariants again. But I don't think it's bad that it has some of its own.

But I think you would have to think more about these invariants and then put them in so that you get results that are a little closer to the sense of how you think the domain should exist. What we definitely miss here is the whole content context in the SQL Room. He says he's talking about an owner and stuff, but he doesn't have a member or anything like that.

[Speaker 2] Yes. He actually does. Unfortunately, I don't have this content with me.

He doesn't visualize the files and folders either. But he maps it with the value object. So there's a key to the SQL Room in the content.

[Speaker 1] Okay. That's not so bad. The aggregate would only protect business invariants.

Exactly. It would only protect it. Exciting.

I think it's not so wrong. It's difficult to say whether it's right or wrong. What's quite charming is that there are certain rules on how to design these aggregates or some experience values from the Big Red Book.

It's a lot about trying to keep it small. If it's not too big, it won't be too difficult. For

example, if we pull in a file and a folder, and a SQL Room also contains all the content, then the object will get big very quickly if it's an active SQL Room.

That's a great first step.

[Speaker 2] I can also show you how the others did it. There was a proposal from Gemini. They mapped it in this context.

They had this secure collaboration context, which contains everything. Data, data and files are included. All at the same time.

As core aggregates. The onboarding context was based on invitation. Identity, access management, user and administration context and group.

They mapped it separately.

[Speaker 1] Let's take a quick look at Gemini. What I really like is the mapping on this level. It's much clearer than on the top.

It focuses less on individual points, but what can happen in this context and what are the aggregates? What are the entities? What are the value objects?

That's nice. The fact that it was so high-level in the first part, where we say what are the bounded contexts? That was a what do you call it?

A flight height lower. Nicely drilled.

[Speaker 2] It's always difficult to find a form how to best display it for the user.

[Speaker 1] Absolutely. But it's cool. Really cool.

[Speaker 2] Let's take a look at computation. Here it's generally said dataroom root entity with dataroom. What are the constraints?

There's an A2A enabled retention period. In variants as well. Each dataroom must have a unique identifier.

A2A can only be enabled if all recipients support it. Retention period cannot be shorter than minimum compliance threshold. The commands and events that are connected to the aggregates.

[Speaker 1] What does he mean by commands? Does he assume a command query

request segregation architecture?

[Speaker 2] I think so. Actually, this aggregates can create itself, for example in createDataroom. The event dataroom.created I can show it to you. Here it's displayed visually. It goes into the dataroom management context.

[Speaker 1] You can also switch on encryption afterwards. That's their suggestion at GPT. Enable E2EE.

The cloud says it's an invariant. You can't change it. GPT assumes E2EE where you can say switch on now.

[Speaker 2] It depends on how well you answer the questions.

[Speaker 1] Absolutely. The product information was also filtered, right? The very first step was to give everything about the product.

Then it explains the domain. Interesting. Maybe that's a side effect that I could imagine.

With the things where the results differ a lot from what you actually understand about the domain, it's a good sign that you're not good enough in the product. Either it's not in the product documents or the LLM has ignored it.

[Speaker 2] It could well be. These requirements are actually a summary of the product documents. Most of them were generated by the LLM.

We'd have to take a closer look.

[Speaker 1] It's like a whispering post. Now that we look at all three, in A you can see how much design space the whole thing has. The LLMs all have a slightly different approach.

At the latest with point three it's about figuring out how to understand the domain and how to cut the objects. There's a lot of possibilities here. Can you tell us about the Geminization from Cloud?

The aggregates and the graphics.

[Speaker 2] There's an extra membership for the invitation.

[Speaker 1] He already put the memberships in there. But of course there were no relations. This is the membership itself.

It has a user ID. It's exciting that he lets out the membership, but at the same time says that there always has to be an owner. So he separates the membership from the owner.

He somehow draws the line. He still controls the owner in the Secure Room. And the rest are memberships.

He would also understand the owner as a membership. Then he would have to draw in the membership, otherwise he can't protect the in-variant. Interesting.

He distinguishes between owner and member. We deliberately didn't do that. As I said, there are simply memberships.

And there's a creator. But it goes back to the owner. Interesting.

[Speaker 2] Do we have... We don't have an explicit owner, but a creator.

[Speaker 1] We have a creator and a creator becomes the first admin. We found out that in our context an owner is the one who created the admin role. That's the implicit owner of the whole thing.

I still have some time left. Ah, okay.

[Speaker 2] Then we can briefly look at the architecture mapping of the admin. The last step was to look at what else I need to be able to map it in our hexagonal thing. Everyone understood something different.

Which is also interesting to see. For every context Cloud has summarized the published events and the public APIs and cut out potential anti-corruption layers. For example TranslateSecRoomEvents to a local model in the access control context.

In case something is necessary. The others... That was Gemini.

He said, okay, we need an application layer. There will be a dataroom service, a file service, a user service, an invitation service, a group service. That applies to these aggregates that are in the domain layer.

And we have infrastructure on that side. I didn't tell him which database we use, but it just came out of the blue. And S3 is there.

[Speaker 1] Gemini is definitely based.

[Speaker 2] And GPT made it a little clearer and said, okay, for this dataroom context,

for example, we have an application. There is a dataroom application service. It can create and enable.

We have our domain part where we have our domain logic, the aggregates. And the infrastructure we mapped out in general. Exactly.

Now, if you were to look at the thing, who do you think is helpful? Let's do it this way.

[Speaker 1] I actually think Gemini is the most helpful. Gemini? Yes, and I have to exclude the others.

Can we take a quick look at Claude? Claude has a few things that have a completely different focus. For example, it doesn't even distinguish infrastructure, business logic or anything else, but it's totally API-focused.

It says, what kind of events exist? There is an API, how can I address it? Everything always rests.

Although GraphQL also appears and has two collection endpoints. And an anti-corruption layer, where I say, what do I need an anti-corruption layer for? Because at the end of the day, my application service is an abstraction of my business logic, which makes use cases out of my business logic.

Why do I need an anti-corruption layer for that? An event normalizer, what does the metadata extractor say? Safely extracts data from E2E encrypted events.

External auth adapter maps external SO to internal user. Okay, that makes sense. So it's actually very, very black-box-focused on the whole thing.

Gemini goes a little more into the white box and says, and maps it without so much effort. Well, it also does REST API calls, but rather a little more illustrative. I think the overall representation is actually quite cool, that you always have this application layer switched on, or from the outside you have your adapters, what connections the REST APIs allow, quite clearly how the application layer speaks to the business logic and that the infrastructure is actually also triggered from the application layer.

If I can interpret that correctly, that the arrows on the right always come from the application layer. Exactly. And he also separates again, what I find exciting, this secure collaboration context, there is no infrastructure at all, but he goes back and takes it completely out again and says, or rather does a context mapping, where he goes and says, yes, how does the onboarding context add up now?

What is the identity access management context and the administration context? That means, he actually took one as a focus, where it's about the whole knowledge logic and everything that is supporting stuff, if I remember correctly, he rather snaps that into the infrastructure layer and I find that very interesting, because that would actually be the approach. Because if you have something supporting, then it's just about how do I integrate it into my entire landscape?

The use cases actually come from my core domain and everything else is more of a connection. So I find that super interesting and at GPT, let's take a quick look, I think it was kept very, very superficial. It also has a little meat on the bone.

These are just a few diagrams, there is a data room, enable E2E event and so on. So I would think that GPT at this point is slowly getting rid of the snow, that you already notice, Gemini and Claude are already better suited for this.

[Speaker 2] We still have the context from before in view.

[Speaker 1] I don't know if I'm leaning on this enable E2E or rather what I find strange is I recognize in Gemini and Claude, I would say, that the LLM somehow built up layers for itself, how the whole thing is connected and breaks it down from the top down or drills it in and says what needs to be done more precisely. And I have the feeling that GPT mixes it up a bit. Yes.

It tries to bring different flight heights together on one level.

[Speaker 2] That may well be the new model, the 5.0. Yes, I have heard that it has already been approved. It may well be because it tries to guess which model would be best internally. And then you can depend on Trump where he puts it and how good the result is.

It's a bit of a black box. Exactly.

[Speaker 1] But overall I would say really cool. I think that shows that it is actually much more important to describe the domain and the product. And the rest you can use the information from an LLM or an agent.

What would be even more helpful would be if the LLM was so smart and said, let's do domain-driven design and ask you questions after the product. Like, not only the domain, but also NFRs. Because I understand SQL Rooms, files and folders.

How many files would there be in a SQL Room? That's where the architecture questions

come in. Now it's just domain.

How does the domain work? How do I map it? On a very simple level.

And then these NFRs would have to come in, which are really important and can help me with the architecture.

[Speaker 2] I have a good example. I did the whole thing for SecuMails. At SecuMails we go there and say, we send an encrypted delivery and there every user gets their own delivery object.

We have to encrypt everything for the user and send it out. There was a model, I have to look up which one, but I'm not sure. In this ubiquitous language step, where we went through.

Somehow it doesn't quite fit. Because what happens when you send an end-to-end encrypted delivery to 100 recipients? That's the async.

When is it encrypted? Exactly these questions that are really interesting for the architecture and how to map it later can be found in the ubiquitous language stack or in the event store. Ah, here it is.

Encryption timing ubiquity, when exactly does encryption happen during send delivery or when access grant is created for 100 recipients, is this synchronous or asynchronous?

[Speaker 1] Mhm. What was that model?

[Speaker 2] That was Claude.

[Speaker 1] Okay. These are the things where, from an architectural point of view, meat and bones come in. Because it really asks on a larger scale how fast, how should the whole thing work, because that's what the architecture does.

Before that, we're talking about DDD and that's hexagonal. But how I now, for example, write my database application and the APIs behind it, are all React things where everything happens asynchronously and how many requests do I have to operate at the same time, how fast information has to be able to be loaded. That's what makes up which architecture I have to choose.

Before that, I only define a business context. That's exciting. But did you come up with these questions yourself?

[Speaker 2] Exactly. I did the same for Seco Mates, just with the idea, okay, let's define a target architecture where we want to go with Seco Mates. Somehow we always take individual things out.

For that, we do DDD.

[Speaker 1] But that was really the question, the context to the LLM.

[Speaker 2] No, the context to the LLM was I have the requirements for Seco Mates, also from project documents, and that goes on forever. That's just cut out now. And said, okay, let's do abbreviated language again.

Then we have by delivery, he thinks, okay, is draft actually a delivery or a concept like delivery type? So there is this draft status of the delivery, and so on. These are all terms from Seco Mates, where he also identified in this abbreviated language step.

Also our security level and so on. Now we actually have LLMs, it was relatively similar. Here also terminal, account, delivery.

What might be interesting to mention, we use in product documents very often Seco Mates and delivery. And every LLM is the same. Why are we changing there?

[Speaker 1] It has already been understood that it is somehow analogous in terms of meaning. And therefore also asked questions whether there is really a separation. Whether there is a separation between delivery and Seco Mates.

[Speaker 2] Exactly. And then the next step, that's again Cloud, event storming, where you say, okay, remember everything that happens in this system. So that I just have all the steps where I know, okay, what happens there.

Because with Seco Mates and BigBallOfMath we also have the problem that there are a lot of features that are a bit hidden. Where we mention in product documents what is not entirely visible in the development or in the code. Simply because it is mapped between managers.

He then tried to map all the co-events and co-delivery flows out. Also simply licensed users, who does what, who creates what happens. And then tries to identify issues and temporal boundaries.

And he noticed that, for example, hey, what happens now with 100 recipients on live delivery?

[Speaker 1] Yes, exactly, that's the question, what should happen with 100 recipients? I think that's actually ... Oh, okay, for 100 recipients, the question is, do they all get a notification when everyone is through?

Or does everyone get a recipient as soon as their thing is through? Where I'm stumbling right now, is it synchronous or asynchronous? What, the encryption or the sending of the deliveries?

And that's a bit ... Then you would have to drill in and ask what he actually wants with the question, what he is aiming for.

[Speaker 2] I actually believe, I mean, to remember, especially with this access grant created, instead of having 100 deliveries, he suggests later to make 100 access grants for one delivery. Because then you don't have 100 objects, but only one and every user gets a key.

[Speaker 1] Which makes absolute sense, because you say, it depends on when you see such an email as a snapshot or such an immutable object where you say it is being created and then nothing changes. Then I don't need 1000 deliveries for 1000 recipients. Then I can have one delivery.

But then I'm talking again about there is only an email in total. But this ... Yes.

And then, as you say, then you can say that ... Let me just briefly ... It has encryption timing ambiguity, where he says, I don't know exactly when that happens.

His first question is, when exactly does encryption happen? He would say before the files are uploaded. That's end-to-end encryption.

Exactly. Then all the other questions are related to when the recipients get the message.

[Speaker 2] So you mean the first three?

[Speaker 1] Exactly, within this encryption timing. The first question is really quite good. When does the encryption actually happen?

And none of this has anything to do with encryption, but when does the recipient get wind of it? Or? During send delivery?

Or when access grant created?

[Speaker 2] So if ... I would have interpreted it like this, when does this encryption

happen? Does it happen during the send delivery?

Or when access grant created? This access grant created was in the step before, if you look at ...

[Speaker 1] Ah, okay.

[Speaker 2] User enters the QRC first access of the delivery after it has been encrypted. And this access grant created is an event that happens when it is fully encrypted.

[Speaker 1] How do we assume that we are encrypting? Because I think it depends on what kind of idea it has, how encryption happens.

[Speaker 2] Exactly, it happens in this step before where you try to define the words. And he reads it out. He reads out, for example, we have zero knowledge, end-to-end encryption, encryption readable only by sender and recipient, content security.

How does this key exchange work? These are just other questions he asks. Based on what he assumes it should work.

I just answered it. Okay, client side. The client is end-to-end encrypted and then sends the encrypted files to the server, because we don't know anything about it.

And then he posts the events accordingly. Or at least this flow. It worked similarly for everyone.

Actually. Maybe the others do it a bit more clearly. It also has a prompting thing.

There is, for example, as with Gemini, he does it like with SQL Rooms. Act on command, negregate an event, what happens, and then line by line. JGPT too.

I tried to do it visually with JGPT. He made a weird diagram for me. Create delivery, delivery created, recipients assigned.

The steps, how we do the deliveries. On the SQL path. And then the next step was to say, okay, what are my building contexts now?

And at Cloud he was suddenly there a lot. Okay, somehow we need something for delivery management context. That will be core, access control context, also in core.

E-management context and encryption service context. Account context, organization context, license in context, submit box context, notification context. Everything fine

granular, I would say, divided up.

The other LLMs were like, okay, in the case of Gemini somehow you need identity and access management context. Secure messaging context, so that everything fits together. And a secure ingestion context, separate for the submit box actually.

Because he sees it as a completely own concept. We still have a little bit of melting in us. JGPT has such an intermediate delivery management.

Identity access management, security, submission handling, notification service and audit and compliance context. That's exciting.

[Speaker 1] I have to say, I like Gemini's the best. Because I already understand it more like this, that I go here and something like an audit context is not a completely different context. It depends.

How do I get comprehensibility? I get it about everything that happens. And if I have a delivery context, for example, and there are two separate things, then what I see as a danger to separate it, when I say comprehensibility, comprehensibility is so urgent in our system or so important or first-class citizen, that if it is not comprehensible, then it has never happened.

Then I can't control it well via these different context. Because one is only a consumer of events that the other throws. And that's why I would have to say how Gemini works, this audit actually within my context.

You open up this big context. Now I would have to look again what bounded contexts are, because I think I mix it up. I would have a delivery context and there is a part where auditing is a quality or an invariance that has to be protected.

Actually, it's a business rule. Or you could solve it in a delivery aggregate. So it's exciting.

But I think Gemini is the most beautiful of the context divisions, because it's not quite so fine granular. There seem to me Claude and GPT a little too technical. Not so much meat on the bone, what this context does, but for everything that actually happens in the system save your own context.

But not to group them in any common. Will Gemini rather think about how it fits together? Exciting in a larger context and say, secure messaging, what does it all belong to?

And the others look at everything where there is an entity or a term, for that I make my own context.

[Speaker 2] We can take a look at the aggregates for Gemini. Core aggregates, for example, are also a bit summarized in this context. Identity access management also has this role and so on.

But he just said root entity, what contains, so these value objects, which then go to the user, for example. The invariants, a user's email must be unique across the system, SQL files can only be confirmed once and cannot be recovered, only changed. A user must always assign a valid role, a guest account, a user with a guest role has specific limitations.

What can the user, which commands there are for the user, the events for the user, user created, guest account created, license assigned. And I always said, look at the size concerns and see if these classes don't get too big.

[Speaker 1] So 1 to 1 relation to the role, 1 to 1 SQL class. From the point of view of the user, I have several users.

[Speaker 2] After the aggregates, everything was done a little bit. Here also with Claude, he just said, in his whole context, he split it. He said, okay, here is a delivery with three access control contexts, an access grant, an account context, then there will be an account, an organization.

[Speaker 1] He just took the name of each context and built it out of an aggregate.

[Speaker 2] The architecture overview was accordingly a bit very large.

[Speaker 1] But that's a nice note, because when you see that these things are actually so related to each other, these contexts, then you have to ask yourself whether they should actually take place within a context. On this level, you can ask yourself the question, hey, okay, for example, does my domain core context work without applications as command handlers?

[Speaker 2] It's all a bit mixed up. Oh, because here the layers are mapped.

[Speaker 1] Here, the context information is lost again, but it's just where something is in which layer of my hexagonal architecture. He classifies that. But now, regardless of the context, he mixes it all up again.

At the end of the day.

[Speaker 2] I also made a real architecture overview. For example, we have an external sender that gives this secure ingestion context. Whether it's a public API or an application service, the main model is a bitbox upload ticket.

The infrastructure then handles the upload to an S3 storage, the communication to the database. The secure messaging context was also kept very superficial.

[Speaker 1] How do we know how this message bus is ingested? Secure ingestion context. What was secure ingestion?

That was this submit box.

[Speaker 2] Most airlines have such a message bus, but for me it looks more like microservices.

[Speaker 1] Or as a client content, so that he gets answers to what he asked.

[Speaker 2] Also possible. But all the events are partly internal. That's why it's a bit interesting.

But now in general for you. If you would go and say, I have to generate the target architecture for SecuML. How helpful would that be for you?

Just to have a sparing partner, use LLM as a sparing partner, do this process. Do you have any ideas about it? Or suggestions on how to make it even better?

[Speaker 1] What we do is we teach the domain and say, let's generate a glossary for a ubiquitous language. Then we go here and say, what can happen in this domain? Then the whole thing is classified into building contexts.

And then we try to DDD constants, or not constants, but DDD cornerstones out of it. What are aggregates? In which layer do you find what?

I think that's a really great idea. And it's really helpful to design the whole thing. But I think what you have to do is you have to loop again and again.

At the latest, you have to go back to your domain after building a context identification and say, how should it actually work with the whole thing? And I think it would be even more helpful if the build.cont, how does it make this build.context? Is it something where you say, hey, now I've given you all this context in terms of all the information

about this domain and what can happen in it.

Do we generate build.contexts or do you say, hey, let's generate a build.context and it spits, we'll say that anyway, but does it ask questions or just make suggestions?

[Speaker 2] First, it just makes suggestions and tries to identify things and says, okay, somehow it's not quite right yet. So exactly this looping also depends a bit on how long you invest and answer these questions. It would definitely make it better.

I have now set a time frame for each architecture. But it's also a bit difficult to estimate because the language has a lot of requirements and you also have a lot of questions that you have to answer. Yes.

Maybe it would be better to make it smaller, to keep the requirements smaller.

[Speaker 1] Yes, but then you have the danger that you don't make an architecture for your entire build.context, but for some sub-context and so on.

[Speaker 2] Yes.

[Speaker 1] That's the art, the challenge. I describe the entire domain and then we drill into these individual things. If I compare it, how does a person do it?

Then you have your big picture and then you go through the use cases and then you drill deeper somewhere and that has effects on, do I really see it that way? Should it really be in this build.context? Because at the end of the day, it should be pretty straightforward with the architecture, if the domain is fixed and the NFRs.

Then they should all get a similar result, based on the previous business world or the state of technology. What are the problems that arise from this domain and these NFRs? What kind of architectural toolbox can I use to tackle this?

I have to take a quick look at what a build.context is. If I have that right in mind. A build.context is actually something where you can add flavor. For me, the question is, is a build.context a consequence of the domain or is it just a view of the domain? A view that I can have on the domain. So either the domain compares when x or x then y, then it quickly turns out that x is a context, y is a context, or is it like this, from x to y, I can draw any number of context limits.

I'm not sure about that right now. Let me take a quick look. Now you have a build.context and a view of the domain. A domain is what we as an organization

understand, what we do, and how the world handles the whole thing. And then I can say, let's say I go here and create a big domain model which is basically the enterprise model, what our company does in total. But with DDD you go here and say, you're actually trying to understand these subdomains.

And then you model these models in build.contexts. And then I think it's something where you yourself say, you have a domain, what you do, and you split it into some build.contexts. That means build.context is more like a limit you can draw yourself. Exactly, within a domain. So your big picture is the domain and then you have your build.contexts. And how are they classified here, for example? What examples do we have here?

[Speaker 2] I have to read this book ten times. You actually have to look it up again later. That means they don't do anything wrong they just have a different granularity that they design on build.contexts. I think that's where the experience and as a developer, where it makes sense to split things up.

[Speaker 1] Yes, especially what already exists. So if I say here, for example, what Claude did quite cool is this, what are really my core domains and what are supporting domains. And I would also say, yes, our core domain is definitely this collaboration context of the SQL Room, but not necessarily file transfer or folders, because there are solutions for that.

I don't have to reinvent the wheel. But we chose that it's a core domain for us. We do it a little differently.

We want to design it ourselves. But in itself, this SQL Room domain, both, it could be a content context or I have in the SQL Room context, in the core, all the files and folders in there. Because I think it comes here to the idea of making it a supporting domain, because it understands that file and folder is actually such a cross-sectional topic about everything that has to do with computers.

Every computer somehow works with files and some hierarchy organization structures, how they are deposited.

[Speaker 2] Which is also an interesting idea, but I didn't give it to him to think about it. If it's a supporting domain, you can also say, yes, SQL Mails uses this supporting domain.

[Speaker 1] Exactly, absolutely. Just like Identity, supporting domains are always a bit

like that. I don't know if you can share them, but they are always cross-sectional topics.

[Speaker 2] Yes. The context, I have to see. He also said that this main core is only this delivery context, access control context and no, actually only these two are included.

The other would be in the sense of supporting or exactly these cross-sectional things. Then you can think about it, is it really supporting or does it belong to it? Maybe in general the question, would you mine as LLM, as a sparring partner, would it show you new ideas or just new perspectives if you would design something like that?

Or the large language model in the architecture design stage, would you use it?

[Speaker 1] It depends. If I say, for example, everything we are doing now is based on my understanding of my domain. Or information.

Then in any case. Then I can simply give the machine everything it knows about the thing and say, get me some context and then I have a discussion basis. If I have no idea at all what my domain really is, then the LLM probably won't help either.

Because then it would suggest something that it has already seen somewhere at the end of the day. It's already a precondition that I have sketched my domain halfway, then it can help with the whole thing. To be able to assess this input with the context again better, you would have to read more closely what a context should actually be and what not with the whole thing.

I have to read this chapter again. Oh God, that would be...

[Speaker 2] Maybe again, you say as a basis, would it, or do you mean if you get from the product team a requirement set, would it just help to do the first two steps, i.e. the ubiquitous language and the event-storming of these requirements, so that you... Absolutely. Okay.

Absolutely. Okay. With the context identification co-op, you have to know a lot about what you actually have.

[Speaker 1] So I think in order to speak the same language with the machine, you need a consistent understanding of what a context and an aggregate is. I don't think aggregates fail that much, because they protect business invariants. That's actually pretty straightforward.

But in the context itself, I think step three is where you can design a lot yourself and it

depends a lot on not how the domain is, but how your view of the domain is. With the context. But I think in general, when working, it's always easier to argue about something that already exists than when you have to build something from scratch and somehow juggle it in your head or on paper.

And that's just such a catch with LLM, really cool.

[Speaker 2] That was also the original idea, that you have something to talk about and just a bit of a guiding thread, especially for new developers, how the DDD works.

[Speaker 1] Yes, that's also a super exciting topic.

[Speaker 2] So that LLM can guide you through this process.

[Speaker 1] Yes, that you can streamline it a bit. Here, for example, in the book, an example with an e-commerce platform. they see two something like inventory.

They say, when you hear inventory, then you dig in a little bit more, because depending on what kind of lifecycle an order can have or a product, you're talking about different items. For example, if you offer an e-commerce platform, which is not available at the moment, then it is somehow backordered. Then you have a backordered item.

If you have food and they can run out somehow, then you have a wasted item, because it somehow broke. So an item is not exactly an item. If you go into it and you have an inventory context, if I understand it correctly, if you have an inventory context and you notice that these items always have classifications below, then you can say again, this is a wasted item, this is backordered, this is shipping or something else.

What consequences does this have on my building context? Did I draw the lines correctly? For example, if I now say, I also have an order or a shipping context then they will have touch points, because a shipped item is an item from the inventory, which the warehouse has already left, in the direction of the customer.

Then it just overlaps again between these building contexts, what an item actually is. An item is not exactly an item. An item is not an inventory item, but is dependent on another context.

Then I can ask myself the question again, do I actually have to draw a context about it? So shipping, not only shipping and label creation and some addresses that I have to have, but is shipping also the handling of an inventory item, how it comes from a warehouse in a package, for example. This is something you would have to do again

and again with the LLM.

If you add more and more information and design this building context more closely, then ask yourself again and again, if these are the most common use cases or if these are such striking features of my domain, then I have to ask myself whether we are drawing these building contexts correctly. Domain-driven design is, you assume that you have a domain and every time you work on this domain, you understand it a little better. That means, it's a totally looping process.

You create the domain, make it understandable for you, draw building contexts, implement it and check if it handles itself a little differently. And this is given as input to the domain. Then you know, the domain for this use case behaves like this.

Then you can look again, does my building context fit? It's actually always a loop. I would make it bigger, i.e. ubiquitous language, event-storming, building context and then back again.

[Speaker 2] Okay. So in the direction of making ubiquitous language, event-storming, building context and then back again. Check if it still fits.

[Speaker 1] Exactly. The idea is that at some point I have such a building context where I say, okay, it fits like this and now I can think about my aggregates and about my architecture. That should actually be a loop.

The ubiquitous language describes what terminologies there are in my domain and on the other hand describes my domain. Terms with a definition. Event-storming says what life cycles these terms can have and what is done with them, what interactions.

And building context says again what contexts can be given within this domain. So from the combination of which terms with which definition what can happen with this event-storming. If I then draw a context about it I have a little bit for example this e-commerce topic again.

They say for example you have e-commerce and e-commerce is typical. You do you have a product catalog, you have orders, you have invoices, you have shipments. This is your e-commerce system.

If you would say I do a glossary about an e-commerce system. I say what happens in event-storming with these things. Then these context that somehow there are products, that there are orders, that there is an invoice and there is a shipping.

That means in itself an e-commerce system can only be a huge context. But I put these terminologies and these life cycle events in relation to each other and notice there are orders, there are invoices, there is a shipment. And this is how these contexts arise.

I can go in and say I drill in even more and say shipping is not the shipping. It differs in my domain if something is shipped by truck, by train, by aircraft, by ship. If there are differences again.

And so I go deeper and deeper. In the beginning I say there is a product that can be ordered, there is an invoice and there is a shipment. These are the products, these are the terms and the event is what is shipped.

There is a shipping context. If I look deeper into this shipping then I can say there is an express shipment. That's a little different.

I repeat myself. It's always a layer system from above. I always drill in with a microscope.

First I look at this onion and say there are different shells. And then I'm deep inside and say what is in such a cell of an onion shell? But how a cell of an onion shell is has a lot to do with why it has become such a layer.

These different flight heights are always due to this. I hope you can follow me.

[Speaker 2] No, I can. That would have been a question how to improve this methodology. The feedback of taking a closer look and going to the first step and see if it still fits what I already have definitely makes sense.

[Speaker 1] For example, you could go here and say after the step 3 of the phase with the context identification I go here and say Dear LLM, map me the events with the contexts you have identified. For example, at Seco Mail, when the event comes, Seco Mail is sent. How many contexts have to interact with each other?

What do they do with it? The delivery is deleted. And so on.

Then LLM might give more information how often which context is actually dealing with which things. If, for example, it turns out that in all use cases that I have for Seco Mail all contexts are always used, then I ask myself what it is worth to integrate it into your own contexts. Am I at this Claude approach where I say I have very fine-grained or am I at Gemini where he says it's about Secure Messaging Context?

My gut feeling would tell me that Claude's input would go more and more to this Secure Messaging Context, if I keep saying bring them together and say if you can get rid of contexts or can I merge contexts to ask LLM again and again why do you say that context A and B are separate? Why isn't it one? You would have to get there again and again.

And that's how this depth in the domain works. And I think with one or the other terminology when you dig deeper, you get a name again. Because now, for example, Secure Messaging Context is I think you know what I mean.

These use cases are something different again with certain building contexts. It's actually a name again. I update my glossary with a new terminology which I call delivery or life cycle event and which then gives a meaning or a definition.

[Speaker 2] That definitely makes sense. Because that way you just go from what LLM is doing right now to the next steps, to the co-aggregation, to the architecture review.

[Speaker 1] And from aggregates and architecture it gets expensive. Then you build up a code and changing it is difficult. Before that, I would just conceptually on a piece of paper.

I can quickly change everything. Because I have a view or an understanding of my domain where I can always throw away the old understanding because I have a newer picture. I can always replace that.

But as soon as the code is aggregates and they suddenly have use cases, to change something there when you look at the context, I think that will be difficult.

[Speaker 2] Yes, exactly. I think that makes sense. Now you have answered a lot.

How can we use that in development?

[Speaker 1] I think for the first three steps it is especially important that you have someone from the product or someone from the domain experts with you. Ideally. So that you can check the suggestions that you get from the LLM with someone who has knowledge about the real world scenario.

And then he says, yes, that's right. You should just subordinate that or say, no, that's nonsense, because these cases don't exist, it doesn't happen. With that, you can decide whether the LLM hallucinates or covers something in this context that would technically make total sense to a certain degree.

These text generators have no logic, but for us as consumers this message seems logical, but in the real world we never get an application case. For example, something like an inventory system. Now the LLM comes and says, we have to go back, we have to take a closer look at that.

If an item comes in again, someone put something back and didn't even unpack it. Someone ordered something, we sent the package, who comes and opens it and says, no, I don't need shit, send it back, what happens now? Then the LLM can think, okay, we have to record returns in this inventory context and then somehow it has to classify whether they are still fully usable or become B-Stock or something.

And then Amazon comes and says, it's totally easy, a domain expert from Amazon, the cheapest is if we throw everything away and burn it. Then the LLM has become a totally sophisticated system, but the real case for the business is that we throw everything away because it's the cheapest. That's why you always need domain experts to verify if it's still close to what actually happens in the domain.

[Speaker 2] That you turn these logs into a product team or customer success.

[Speaker 1] Whether you even want to focus on the whole thing. Because at the end of the day it's about what our core domain is. What are the things that our product really needs to be able to do well and what the supporting or where we don't care at all, where we say, let's leave gaps, that's not our beer.

[Speaker 2] Cool.

[Speaker 1] That's the end of the great topic, man.

[Speaker 2] I have to think about where we are right now.

[Speaker 1] I think we were at the point where we said, it's really cool that you can make good headlines with it. Now the question was what can you improve on it? And I said looping.

You shouldn't just write the architecture and the aggregates and get into the white coding right away, but take a little time to loop over the domain until you say you have your 80

[Speaker 2] Do you see that we use this in development? Would you say, you would make such an approach if you get a requirement set?

[Speaker 1] In development, it's an exciting question because I would say phase 1, 2 and 3 is more discovery and phase 4 and 5 is delivery. First I explore the domain and then I think about how I can make it come true. Actually, it would also be exciting how far the product might use something like that to go over such ideas.

I would say phase 1 and 2 are definitely delivery-heavy and at phase 3 there is a delivery-heavy discovery. So phase 1 and phase 2 is how my domain looks like. Phase 3 is a bit overlapping between discovery and delivery because I'm already starting to classify subsystems so that I can technically map or handle it in some way.

If I don't just refer to technology in this context, why should someone who has an e-commerce platform be interested in shipping, inventory, product catalog and ordering? Because he says he has departments that take care of it. I have people who only take care of sending parcels.

I have people who only take care that the invoices all fit. The others research products and say what we could offer on our platform. That's a thought for a company that has departments.

What kind of modules do we have in the software? What kind of services do I want to do with microservices because I have so many developers? Do I want to pour it into microservices?

But that's the question for me again. Where is the context still attached to the domain and where does the solution occur? If we are in the solution, then it becomes technical.

Then it's not so much about the domain.

[Speaker 2] Most of the time, of course, it's more about conceptual.

[Speaker 1] The first two, I would also say the third, but at least from four. Yes, four are technical architectures. The fifth is really delivery.

[Speaker 2] Cool. Do you have any comments or questions about the protocol?

[Speaker 1] As a note, I would like to add that becoming a monolith from Big Ball of Mud will be much more complex than starting from a green meadow. Because in Big Ball of Mud there are no bounded contexts or aggregates or they have not been cut according to the domain. If you go there and start from a green meadow, then you can loop 1, 2, 3 forever.

I'm pretty confident that this is the domain and then let it be implemented. The truth is, with Big Ball of Mud, it was probably rather phase 1, 2, 3 implicit to a certain degree of detail. Then it was actually only coded on it all the time.

To actually get these different things out of it, I first have to understand the domain, the business, the product. I would think that you would not do that at all. You would just say, okay, I'll stay on these first three phases.

I'll look at this domain and then when it's fixed, I'll do a bit of a strangle pattern, as we did for the SQL rooms. We didn't start and say, let's look at the share context and say, what could it all be? What is it?

What is it not? We cut something out and said, okay, let's try to understand the domain data rooms and put that in a new place. That is, the Big Ball of Mud to modulate would actually take place in these first three phases.

So that you think, what kind of context do I have in my Big Ball of Mud? I think that's a note. I would say that's the strangle pattern.

But as I said, first I have to understand the domain for the whole thing. And what maybe an LLM would actually be really cool for these existing things like Big Ball of Mud is, is if you could just say to this LLM, this is our database, Ponyhoof, far away, this is our database, this is our feature set. Tell me, which feature gets the most engagement?

Because we have, our Big Ball of Mud has grown organically over the years. There are a few features that naturally generate certain data in the database or signatures or metadata, where I know, if I have a lot of metadata, there will be a lot of engagement. And if I have little, little.

So that I can classify the LLM, where is it really worth writing tests again or improving the code and things like that, where I say, you should actually think about whether to continue to maintain it, because I have no idea whether anyone uses it. Because this input again, in addition to the technical aspect, would also say again, what is our core domain? How do we actually perceive our users or our customers?

What do they actually use of the things?

[Speaker 2] Exactly, what are we actually doing actively and what is there, but is not used?

[Speaker 1] Exactly, I find that mega exciting. But that's not about the design, but about evaluating it. Now I'm at a certain level to get the outside perspective back on the domain.

[Speaker 2] Yes, yes, yes. Cool. That's definitely a good approach.

[Speaker 1] No matter, it's really a super exciting topic. So if we can support you somehow, I'd love to. You can always come when it's time to read the correction or if you have any questions, I'd love to.

[Speaker 2] I'll just change the option. Now. Thank you.

Thank you.

A.3.3. Interview Expert C

List of Figures

List of Tables

Bibliography

[AAS25] A. Aqsa, A. Aslam, and M. Saeed. "Efficient Prompt Engineering: Techniques and Trends for Maximizing LLM Output." In: Apr. 2025. DOI: 10.5281/zenodo.15186123.

[BOP22] G. Blinowski, A. Ojdowska, and A. Przybyłek. "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation." In: *IEEE Access* 10 (2022), pp. 20357–20374. DOI: 10.1109/ACCESS.2022.3152803.

[Che+23] K. Chen, Y. Yang, B. Chen, J. A. Hernández López, G. Mussbacher, and D. Varró. "Automated Domain Modeling with Large Language Models: A Comparative Study." In: *2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 2023, pp. 162–172. DOI: 10.1109/MODELS58315.2023.00037.

[ESW24] T. Eisenreich, S. Speth, and S. 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.

[Eva03] E. Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Boston, MA: Addison-Wesley Professional, 2003. ISBN: 0-321-12521-5.

[Eva04] E. Evans. *Domain-driven design: tackling complexity in the heart of software*. Addison-Wesley Professional, 2004.

[Fow02] M. Fowler. *Patterns of Enterprise Application Architecture*. Boston: Addison Wesley Professional, 2002.

[FTA25] FTAPI Software GmbH. *FTAPI Secutransfer Platform*. FTAPI Software GmbH. 2025. URL: https://www.ftapi.com/plattform (visited on 05/29/2025).

[MS16] A. MacCormack and D. 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.

[ÖBB23] O. Özkan, Ö. Babur, and M. van den Brand. "Refactoring with domain-driven design in an industrial context." In: *Empirical Software Engineering* 28.4 (2023), p. 94. DOI: 10.1007/s10664-023-10310-1.

[Ric19] C. Richardson. *Microservices Patterns: With Examples in Java*. Shelter Island, NY: Manning Publications, 2019.

[Sai+22] R. Saini, G. Mussbacher, J. L. C. Guo, and J. Kienzle. "Machine learning-based incremental learning in interactive domain modelling." In: *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems*. MODELS '22. Montreal, Quebec, Canada: Association for Computing Machinery, 2022, pp. 176–186. ISBN: 9781450394666. DOI: 10.1145/3550355.3552421.

[Ver13] V. Vernon. *Implementing domain-driven design*. Addison-Wesley, 2013.