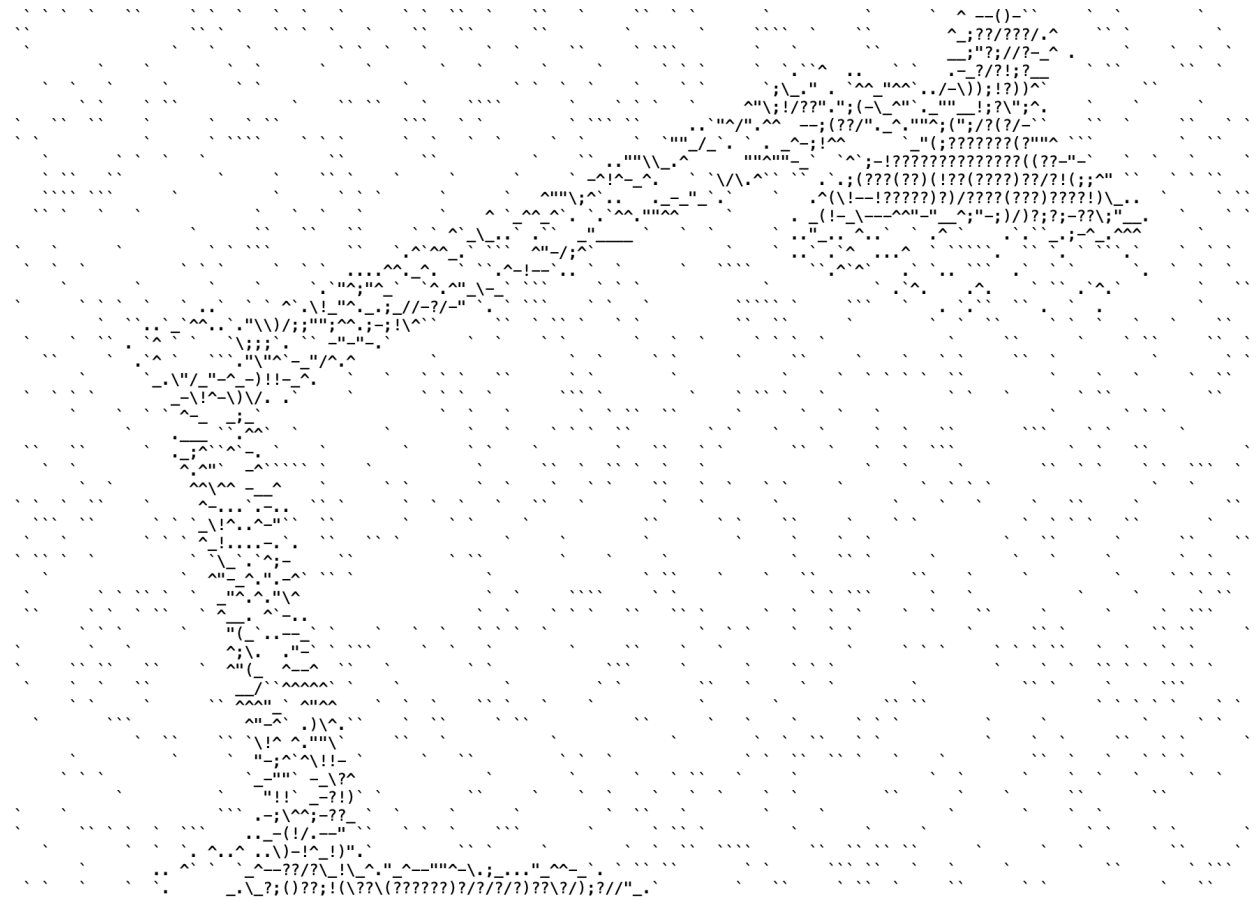




CHALMERS
UNIVERSITY OF TECHNOLOGY



Automating Maintenance Work Order Processing in Facility Management

Classification, Responsibility Routing, and Data Readiness in Maintenance Workflows

Master's thesis in Complex Adaptive Systems

MEGAN BROWN

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2026

www.chalmers.se

MASTER'S THESIS 2026

Automating Maintenance Work Order Processing in Facility Management

Classification, Responsibility Routing and Data Readiness in
Maintenance Workflows

MEGAN BROWN



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Architecture and Civil Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026

Automating Maintenance Work Order Processing in Facility Management
Classification, Responsibility Routing and Data Readiness in Maintenance Work-
flows
MEGAN BROWN

© MEGAN BROWN, 2026.

Supervisor: Elena Malakhatka, Chalmers Next Labs
Examiner: Holger Wallbaum, Department of Architecture and Civil Engineering

Master's Thesis 2026
Department of Architecture and Civil Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Automating Maintenance Work Order Processing in Facility Management
Classification, Responsibility Routing and Data Readiness in Maintenance Workflows

MEGAN BROWN

Department of Architecture and Civil Engineering
Chalmers University of Technology

Abstract

The Facility Management (FM) industry generates vast amounts of unstructured text data in the form of Maintenance Work Orders (MWOs). However, this data is frequently underutilised, often treated as an administrative burden rather than an analytical resource. Compared to hardware-dependent IoT (Internet-of-Things) and BMS (Building Management System) data, MWOs do not rely on modern building infrastructure and they can therefore provide valuable insights into building operations across diverse real-estate portfolios containing buildings of varying technical maturity.

The Goal of this thesis is to investigate how MWOs can be leveraged to automate administrative workflows and improve data quality within FM. Using a Design Science Research (DSR) approach in collaboration with Akademiska Hus, an exploratory data analysis established that traditional classification methods struggle with historical noise, overlapping categories, and the under-specified nature of occupant text inputs. To address these bottlenecks, two artefacts, *The Structurer* and *The Demarcation Agent*, were developed and evaluated.

The results indicate that while full autonomy is restricted by non-deterministic model behavior and incomplete technical details in user texts, a Human-on-the-Loop (HOTL) configuration provides a viable operational framework. Shifting the technician's role from manual triage to validation lessens administrative burden without the risk of automated routing errors. Furthermore, the pipelines introduce a standardised classification logic that reduces the subjectivity and variance inherent in manual logging, thereby improving data consistency for future maintenance analytics.

Keywords: artificial intelligence, facility management, maintenance work orders, large language models, design science research.

Acknowledgements

I would like to extend my sincere thanks to my supervisors, Elena Malakhata and Tobias Robertsson. Thank you, Elena, for always pushing me to expand, refine, and develop my ideas. Thank you, Tobias, for the weekly guidance and technical support that kept this project on schedule. I would also like to thank Peter Karlsson and everyone at Akademiska Hus for their warm reception and continuous support throughout this work. Gaining direct insight into your daily operations, systems, and practical challenges provided the crucial domain foundation upon which this thesis is built. Lastly, I would like to thank my friends and family who have been a great moral and mental support during this period which has been challenging at times.

Megan Brown, Gothenburg, September 2026

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AEC	Architecture, Engineering and Construction
AI	Artificial Intelligence
API	Application Programming Interface
BDA	Boundary Demarcation Agreement
BIM	Building Information Model
BMS	Building Management System
CMMS	Computerised Maintenance Management System
CSF	Critical Success Factor
DSR	Design Science Research
EDA	Exploratory Data Analysis
FM	Facility Management
HITL	Human-In-The-Loop
HOTL	Human-On-The-Loop
HCAI	Human-Centred AI
HVAC	Heating, Ventilation and Air Conditioning
IoT	Internet of Things
KPI	Key Performance Indicator
LLM	Large Language Model
MWO	Maintenance Work Order
NLP	Natural Language Processing
PoC	Proof of Concept
SVC	Support Vector Classifier

Contents

List of Acronyms	ix
1 Introduction	1
1.1 Background	2
1.2 Problem Statement	3
1.3 Scope	3
1.4 Aim and Objectives	4
1.5 Research Questions	4
1.6 Methodology: the Design Approach	5
1.6.1 Design Science Research	5
1.6.2 The Double Diamond Framework	5
1.6.2.1 Diamond 1: Discover and Define	6
1.6.2.2 Diamond 2: Develop and Deliver	7
1.7 Limitations	7
2 Phase 1: Discover and Define	9
2.1 Theory and Literature Review	9
2.1.1 Key Performance Indicators for FM	9
2.1.2 Digitalisation in FM: From IoT to MWOs	11
2.1.3 The Emergence of Agentic AI and Cognitive Automation	12
2.1.4 Organisational Readiness and Adoption	13
2.1.5 Human-Centered AI: Trust and Safety	14
2.2 Exploratory Data Analysis of MWOs	15
2.2.1 The Work Order Data	15
2.2.2 Data Preprocessing and Cleaning	16
2.2.3 Topic Modeling: Exploring clusters and patterns in the data	17
2.2.4 Initial Modeling: What do the classes look like?	19
2.2.5 Summary of EDA	20
2.3 Iterative Prototyping Sessions	21
2.3.1 Iteration 1: The As-Is Architecture	21
2.3.2 Iteration 2: From Pain Points to Prototypes	22
2.4 Feedback and Selection	24
2.4.1 Summary and Next Steps	26
3 Phase 2: Develop and Deliver	27
3.1 The Demarcation Agent	27

3.1.1	The Data: A look at boundary demarcation agreements	27
3.1.2	Architectural Decisions: RAG vs Direct Injection	28
3.1.3	System Integration (Future Work)	30
3.2	The Structurer	30
3.2.1	The Symptom Model	31
3.2.2	The Priority Model	33
3.2.3	System Integration (Future Work)	33
4	Results and Evaluation	35
4.1	Technical Performance	35
4.1.1	The Demarcation Agent	35
4.1.2	The Symptom Class Model	37
4.1.3	The Priority Model	40
4.2	Task Performance	41
4.2.1	Problem Resolution	42
4.2.2	Estimated Impact on FM KPIs	42
4.3	System Agency and Architectural Autonomy	43
5	Discussion	45
5.1	Characterising MWO Data and the Ontology Gap	45
5.2	Architectural Integration and Task Automation	46
5.3	Comparative Performance of Machine Learning and Language Models	47
5.4	Automation, Operational Readiness, and Human-in-the-Loop Design	47
5.5	Methodological Limitations and Ethical Considerations	48
6	Conclusion	51
6.1	Contribution	52
6.2	Future work	52
	Bibliography	53

1

Introduction

The operational phase of a building, including utilities, physical maintenance, and ongoing facility administration, accounts for approximately 75% of its total life-cycle costs [4]. To manage these costs and maximise efficiency, digital transformation is essential. Yet, the facility management (FM) industry has historically lagged behind other sectors, continuing to rely on conventional approaches instead of emerging technologies [2, 30]. As urbanisation continues and facility managers handle increasingly larger portfolios, developing maintenance practices that are truly scalable becomes imperative. [30]

Daily operations within FM are largely driven by maintenance work orders (MWOs). While these MWOs contain textual data capable of giving insight into building performance, they typically exist as large, unstructured repositories [17]. The lack of standardisation can make it challenging to extract insights or perform analytics [17], presenting a major barrier to scalability.

These maintenance requests represent the main link between building occupants and facility managers, but because the reports are generated by occupants, they consist of informal, natural language where descriptions may vary significantly in detail and clarity. As a consequence, technicians must dedicate significant time to reading and evaluating these records one by one before any decision-making or physical maintenance can begin [7, 37]. Beyond creating administrative bottlenecks, this repetitive task introduces risks for human error and inconsistent logging, which degrades the data quality over time [17, 37].

The unstructured nature of these databases represents a significant barrier to traditional data analytics and process automation [17], warranting an investigation into whether modern Artificial Intelligence (AI) or Machine Learning (ML) solutions handle these tasks better. This thesis explores the implementation and limitations of AI-driven workflows to automate processes within FM, using Akademiska Hus as a case study. Specifically, the research focuses on developing and evaluating two artefacts: a filtering pipeline for routing misdirected tickets, and a structuring pipeline for classifying and standardising MWO data to assist technicians and enable data-driven analysis. By moving from theoretical exploration to functional Proofs of Concept (PoCs), this work aims to evaluate to what extent applied AI and ML can reduce administrative bottlenecks and create more scalable maintenance operations.

1.1 Background

According to the International Facility Management Association (IFMA), facility management is defined as "a profession dedicated to supporting people", which "ensures the functionality, comfort, safety, sustainability and efficiency of the built environment" [22]. This highlights that FM is fundamentally a service-oriented discipline. A primary objective of FM is to optimise the end-user experience [28], rendering user satisfaction a key indicator of overall FM performance [1].

FM is generally carried out in two ways: preventive maintenance (routine checkups, planned work, and response to alarms on IoT devices) and reactive maintenance (carried out in response to maintenance requests issued by users) [37]. Within reactive maintenance, when an occupant detects a fault or issue in the built environment, they typically submit a maintenance request through a graphical user interface (GUI). The request is forwarded to a digital system, commonly a Computerised Maintenance Management System (CMMS). A technician or administrative employee subsequently reviews the request in the CMMS, before a technician is dispatched to resolve the issue.

To examine these workflows in a real-world setting, this thesis uses Akademiska Hus as a case study. Akademiska Hus is a Swedish state-owned real-estate company, managing 3.4 million sqm of real-estate in 15 university cities across Sweden [3]. Part of their role as property owners involves managing and responding to maintenance requests submitted by the people using their buildings. Akademiska Hus processes around 30,000 maintenance requests per year. Responding quickly and accurately to these requests is a primary factor in ensuring high satisfaction among their occupants [1].

Within the context of this report, a maintenance request refers to a complaint filed by a user (anyone inhabiting or utilising the premises, such as students or university staff). The maintenance request can be a complaint from the user or a report of a detected fault. Once a maintenance request is entered into the Computerised Maintenance Management System (CMMS), it becomes a maintenance work order (MWO) to be handled by a technician or facility manager (terms used interchangeably in this thesis). Furthermore, the term tenant is reserved for the organisational entity renting the property, such as the university itself.

Although the FM industry has widely adopted digital tools like CMMS to centralise operations, triaging incoming requests remains largely manual [33]. These requests typically combine structured metadata with user-generated free text, which may be ambiguous or lacking in relevant information. Hence, technicians spend a disproportionate amount of time on administrative tasks [37], such as ticket routing, classification, and cross-referencing issues with boundary demarcation agreements. This creates a bottleneck that grows as building portfolios become larger and more complex.

While traditional automation is successfully employed in FM for numerical processes, such as HVAC alarms and energy optimisation[41], it fails when applied to administrative workflows driven by text input. Firstly, facilities operate under unique contracts and boundaries of responsibility between property owner and tenant, which cannot easily be encoded into simple rules. Secondly, maintenance work orders (MWOs), often suffer from poor data quality due to manual data entry[7, 9]. In practice, this manifests as missing context, empty fields, and variable, non-standardised terminology for identical technical issues [7, 9].

Since managing these tasks depends on contextual judgment, these workflows remain manual across the industry, relying largely on human reasoning [37]. To address this limitation, recent research highlights the potential of Large Language Models (LLMs) to emulate these exact cognitive capabilities. As Wang et al. [39] note, LLMs have demonstrated potential to achieve human-like intelligence by leveraging massive datasets. Building upon this, a growing research area employs LLMs to power autonomous agents capable of human-like decision-making [39]. In a complex environment like facility management, it becomes interesting to evaluate these capabilities alongside more established data-driven techniques. This thesis therefore explores the feasibility and operational value of implementing different AI approaches to automate maintenance workflows.

1.2 Problem Statement

Based on the operational realities identified in the background, the automation of maintenance workflows faces two distinct barriers:

- **Textual Character and Automation Viability:** While literature notes that CMMS databases contain valuable data [17], the specific character and variability of the MWOs remains largely unexamined. It is uncertain how these texts are structured, whether they contain sufficient information for automated processing, and to what extent they can be successfully utilised by text classification models.
- **AI and Operational Friction:** The facility management sector is traditionally slow to digitise [30], meaning new technologies must adapt to legacy constraints rather than replace them [33]. Implementing AI or Large Language Models (LLMs) introduces two layers of friction: technical friction when integrating with CMMS systems, and operational friction if the tools disrupt the established routines of maintenance staff [33, 35]. The architectural frameworks required to deploy these models without causing workflow disruption remain undefined.

1.3 Scope

To ensure the feasibility of this research within the given time-frame, the project is bounded in the following way:

- **The Deliverables:** The deliverables are limited to 1-2 functional Proof of Concepts (referred to as artefacts throughout the report), hosted locally.
- **The System Environment:** The artefacts are developed using local Python environments, invoking LLMs via the Azure OpenAI API. The architecture is designed to be easily transferable to a Microsoft Azure/Foundry environment, but the actual deployment is out of scope.
- **The Data:** Development relies primarily on historical Maintenance Work Order data, with the possibility of employing other administrative or static policy data if needed. The use of IoT and BMS data is out of scope.

1.4 Aim and Objectives

The aim of this thesis is to investigate ways in which modern AI and ML can automate the MWO process in FM, with the goal of reducing manual administration for technicians and improving service quality. To accomplish this, the project is guided by the following objectives:

- Analyse the current system architecture and its bottlenecks.
- Propose solutions targeting the identified bottlenecks, and evaluate based on feasibility, KPIs, and stakeholder feedback.
- Develop a functional PoC, based on one or several of the previously proposed solutions.
- Assess the PoC based on KPIs, performance validation and technical metrics.

1.5 Research Questions

This thesis aims to investigate and answer the following research questions:

RQ1. How do user-generated maintenance requests differ from the information requirements of structured FM classification and responsibility-routing systems?

RQ2. How can AI-based artefacts be designed to support classification, prioritisation and responsibility-routing tasks in FM work order workflows?

RQ3. What are the comparative strengths and limitations of traditional ML and LLM-based approaches for classifying imbalanced and under-specified MWO datasets?

RQ4. What level of autonomy is appropriate for AI-supported FM workflows, and how can Human-on-the-Loop architectures reduce operational and contractual risk?

1.6 Methodology: the Design Approach

To effectively navigate the iterative design process required to develop valuable and relevant solutions, this research employs the Double Diamond framework integrated within a Design Science Research (DSR) approach.

1.6.1 Design Science Research

Design Science Research (DSR) is a research paradigm focused on the creation and evaluation of innovative artefacts to solve complex organisational problems. Rather than relying solely on theoretical problem definitions, DSR focuses on the practical needs, strategies, and limitations of the organisation. This ensures that the research delivers tangible value to stakeholders. A core concept within DSR is Design Knowledge, defined as the theoretical and practical knowledge gained through the act of developing an artefact. Therefore, the development of an artefact is framed as not only a deliverable product, but as the main vehicle for conducting research, which is then evaluated using established academic frameworks to ensure rigour [32].

This paradigm is relevant to this thesis. As operational challenges within FM are complex and context-specific, the problem cannot be understood in isolation. By adopting a DSR approach, the thesis shifts its perspective from simply "building a tool to fix a known issue" to designing an artefact to explore and answer the defined research questions.

1.6.2 The Double Diamond Framework

The Double Diamond framework is a methodological framework used to balance exploration and decision-making in an iterative design process. In the context of this thesis, it acts as the practical framework used to carry out the DSR approach. The framework is built on the alternation between divergent thinking (broadening the scope to explore possibilities) and convergent thinking (narrowing the scope to make definitive choices) [12].

The first diamond encompasses the Discover and Define phases, which are used to explore the problem space and define the challenge to be solved. The second diamond consists of the Develop and Deliver phases, shifting focus to the solution space, where solutions to the challenge are explored, developed and evaluated.

This framework dictates the workflow of the project and the structure of this report. Chapter 2 aligns with the first diamond (Discover and Define), detailing the initial exploration of the data and the bottlenecks in the system. Chapter 3 aligns with the second diamond (Develop and Deliver), documenting the iterative development

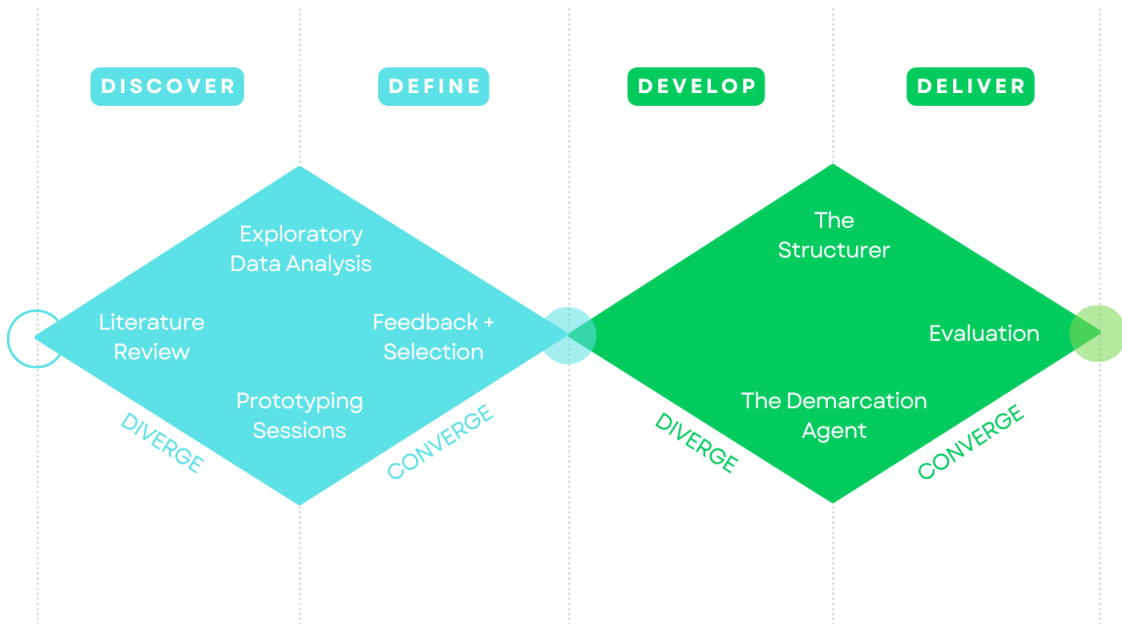


Figure 1.1: The Double Diamond Framework and how it was applied to this thesis, including the steps taken at each stage.

of the artefacts. The rest of this section details the specific methods and expected outcomes for each of the four phases as they were applied to this research.

1.6.2.1 Diamond 1: Discover and Define

The Discover and Define portions of the project were conducted through three primary activities: literature review, exploratory data analysis, and prototyping sessions. For each activity, the objective (why it was done) and the resulting product (what it generated) are detailed below.

1. Literature Review

- **Objective:** This step was conducted to establish a theoretical foundation for the project.
- **Product:** The key outputs of this review were a list of Key Performance Indicators (KPIs) tailored to the project, the theoretical justification for implementing a Human-on-the-Loop (HOTL) architecture, a framework for defining an AI agent, and an understanding of the State of the Art (SOTA) research on AI and FM, as well as research gaps.

2. Exploratory Data Analysis (EDA)

- **Objective:** The objective of the EDA was to understand the quality and limitations of the historical Maintenance Work Order (MWO) dataset provided by Akademiska Hus (the host organisation).
- **Product:** This produced a comprehensive understanding of the dataset's constraints and served as a foundation for understanding which AI solutions could feasibly be built during the prototyping sessions.

3. Prototyping Sessions

- **Objective:** Serving as the core of the "Define" phase, these sessions aimed to synthesise the insights from the literature review and data analysis with the actual day-to-day problems identified by the stakeholders. Through iterative prototyping and evaluation, a set of solutions were developed to target the problems.
- **Product:** This produced a diagram of the system architecture, which defines the environment that the artefacts will come to live in, as well as a definitive list of proposed solutions to be built in the next phases, making sure the solutions were scoped to be manageable within the time-frame of the thesis.

1.6.2.2 Diamond 2: Develop and Deliver

The Develop and Deliver portions of the project focused on building, testing, and refining the solutions proposed in the first diamond. The development was conducted iteratively and split into tracks: one for each of the proposed solutions. The key activities of this phase are detailed below.

Prototype Development (The Demarcation Agent and The Structurer)

- **Objective:** The objective of the development was to transform the conceptual solutions from phase 1 into functioning prototypes.
- **Product:** This activity produced the localised codebases and inference logic for the artefacts, functioning as initial Proof-of-Concept (PoC) implementations, ready for evaluation.

3. Evaluation and Validation

- **Objective:** The Deliver phase requires the solutions to be evaluated against the problems they were designed to solve. The objective of this activity was to evaluate the artefacts through technical performance as well as through other frameworks defined throughout the project (such as KPIs and design requirements).
- **Product:** The product of this is the final evaluation of the outcome, presented in Chapter 4.

1.7 Limitations

The findings and developments of this thesis are subject to the following limitations:

- **Lack of System Integration:** Since the scope was restricted to local PoCs, the solutions were not integrated into the organisation's CMMS API. Subsequently, the results are limited to evaluation on historical data and estimations of KPI metrics, as opposed to results from real world trials.
- **Generalisability:** The models were developed and evaluated using Akademiska Hus' historical MWO data, which follows a classification scheme specific to AH and consists mainly of texts in Swedish, relating to university campuses. The

exact results may therefore not be seamlessly transferable to other similar datasets in other languages, sectors or of different quality/using different classification schema.

- **Database Simulation:** In the PoC implementation, the retrieval of documents is limited to a static mapping, which is much simpler than implementing a live document retrieval backend.
- **AI Stochasticity:** Since LLMs are inherently stochastic in nature, there is always a risk for hallucinations, despite the use of resilient prompting techniques.
- **Lack of Empirical Study:** The final evaluation against KPIs could only be estimated and not validated, as no real-world trial was carried out.

2

Phase 1: Discover and Define

This chapter covers the Discover and Define phases of the project. This includes initial problem exploration through literature review and stakeholder interviews, data exploration and definition of feasible project artefacts. The chapter is laid out in a chronological manner, detailing the various steps taken to understand the context of the problem and developing the proposed artefact ideas.

2.1 Theory and Literature Review

This section describes the knowledge and insights gained from the initial literature review, focusing on the intersection of facility management (FM), data-driven administration, agentic AI, and organisational readiness. Specifically, the topics investigated were: current operational realities of FM, digitalisation and data-driven approaches in FM, key performance indicators (KPIs) and critical success factors (CSFs) of FM, the technological advancement from NLP to LLMs and agentic AI, human-in-the-loop (HITL) models, and implementation of AI in an organisation. Some of the keywords used were "facility management", "work order", "helpdesk", "data-driven", "digitalization", "building maintenance", "user satisfaction", "agentic AI", "Large Language Models", "human-in-the-loop", "implementation gap", "data readiness", "organisational readiness". Searches were primarily carried out via Scopus, and to some extent IEEE Xplore and Science Direct.

2.1.1 Key Performance Indicators for FM

To estimate the value of any technical development in FM, it is necessary to establish an understanding of how FM performance is measured. Lavy et al. [25] proposed a broad set of KPIs for facilities, sorted into four categories: Financial, physical, functional and survey-based. A majority of these KPIs can be considered to be directly tied to the quality of facility management, such as "Building physical condition", "Building maintenance costs", and "Occupants productivity". Recontextualising Lavy et al.'s [25] KPIs from an FM perspective, three overarching performance measures emerge: Operational Efficiency (time), Financial Performance (cost), and Service Quality (user satisfaction).

While financial and operational metrics are easily quantified, evaluating service quality requires a deeper look at the end-user experience[5]. One of the main responsibil-

ities of a facility manager is responding to requests and complaints issued by users. As such, it is undeniably important to consider user satisfaction when evaluating the success of the service. Abisuga et al. [1] propose a framework, building off of the commonly used SERVQUAL framework, with the addition of measures related to fairness and communication with the user. They found the most important factors to be time (Timeliness) and communication (Attentiveness).

However, achieving this high level of timeliness and user responsiveness relies heavily on the internal efficiency of the FM organisation. Hopland & Kvamsdal [19] ranked CSFs primarily from an internal operational perspective, focusing on the efficiency of the FM provider rather than end-user satisfaction. They found that important factors were generally related to clear instructions and plans surrounding decision-making (such as economic conditions, prioritisation, clear maintenance requests), as well as competence among facility managers.

Table 2.1: Synthesis of FM KPIs

Core FM KPI	Lavy et al. (2010)	Abisuga et al. (2020)	Hopland & Kvamsdal (2018)
Operational Efficiency	Physical & Financial : e.g. <i>Building Performance Index</i> and <i>Maintenance Efficiency Indicators</i> .	Timeliness: Minimising response times.	Input Optimisation: Early notification of issues, clear maintenance requests, systematic prioritisation protocols to streamline technician workflows.
Financial Performance	Financial: e.g. <i>Building Maintenance Costs, Operating Costs</i> and <i>Current Replacement Value</i> .		Economic Control: Managing economic constraints through clear decision-making guidelines.
Service Quality	Physical, Functional & Survey-based: e.g. <i>Indoor Environmental Quality, Occupant Productivity</i> and <i>Occupant Satisfaction</i>	Attentiveness, Fairness & Redress: Active and transparent user communication.	Internal Capability: Developing manager competence and robust service delivery practices.

These distinct perspectives (the financial goals, the user’s need for communication and the provider’s need for structure) can be synthesised into a cohesive framework, as outlined in Table 2.1.

As demonstrated, these KPIs are very interlinked. The operational framework proposed by Hopland & Kvamsdal [19] claims that efficiency requires systematic and clear inputs. Therefore, if a maintenance request is ambiguous or misrouted, efficiency drops. This bottleneck immediately causes an increase in resolution times and costs, directly affecting the timeliness and attentiveness demanded by users[1]. Therefore, to improve overall FM performance, this initial bottleneck needs to be optimised. To understand how this could be carried out, the efforts towards automation and digitalisation within the FM industry so far have been studied.

2.1.2 Digitalisation in FM: From IoT to MWOs

The digitalisation of FM initially centered around Information and Communication Technology (ICT) designed to digitise paper-based workflows [17]. This led to the widespread adoption of Computerised Maintenance Management Systems (CMMS), where work orders could be stored and handled by facility managers. As buildings became more digital, the industry shifted towards data-driven approaches. Modern buildings generate a vast amount of data, generally categorised into static data, such as Building Information Models (BIM), and dynamic data, gathered through Building Management Systems (BMS) and IoT sensors [33].

IoT-driven maintenance is an extensively explored topic within the academic field of data science and AEC/FM [28]. A significant portion of this literature focuses on Predictive Maintenance (PdM), which attempts to predict equipment failures before they occur using Machine Learning algorithms [33]. Although academic studies demonstrate promising theoretical results for IoT and BMS-driven PdM, real-world implementations often remain building-specific [33, 41], making them difficult to scale across an entire organisation. As highlighted by Xu et al., the widespread adoption of these solutions is hampered by differences in physical infrastructure, particularly in older buildings that lack modern sensor equipment [41].

To overcome these hardware limitations found in real estate portfolios, such as university campuses comprising both modern and historic buildings, administrative data provides a more widely available alternative. Specifically, Maintenance Work Orders (MWOs), submitted by users regardless of a building’s technological age, represent a rich and largely untapped data source. Unlike IoT-data, which is limited by the varying quality and presence of physical sensors, MWOs are generated continuously and fairly uniformly. This data therefore allows for insight into user satisfaction, user needs, component failure rates, error trends, and the overarching state of a building [17].

Unstructured MWO data is inherently more difficult to analyse [17, 37], especially before the advent of LLMs. This likely explains why the topic of MWOs as a data source remains largely unexplored [33]. Despite the challenges of unstructured text, early research using traditional Natural Language Processing (NLP) and classical Machine Learning (ML) has attempted to make use of MWOs. In the realm of data

structuring, Stenström et al. [37] investigated how NLP can be used to transform unstructured work orders into structured data, suitable for statistical analysis, while D’Orazio et al. [13] evaluated how different preprocessing pipelines affect the performance of classification of work orders. Building on this, other studies focused on operational triage, with Bouabdallaoui et al. [7] classifying problems based on user’s maintenance reports and Ensafi et al. [15] predicting resolution times.

While these early studies demonstrate the potential of MWO data, their applications remain limited to isolated tasks within the broader workflow. They also highlight how traditional ML struggles with the linguistic nuances of these casual texts. This suggests that a significant potential remains untapped. This thesis builds on the hypothesis that expanding automation beyond these narrow tasks can streamline larger segments of the maintenance workflow.

2.1.3 The Emergence of Agentic AI and Cognitive Automation

As demonstrated, traditional Natural Language Processing (NLP) has enabled basic classification of work orders. However, these methods are limited by their inability to understand semantic context, underlying meaning, or complex sentence structures. The introduction of the Transformer architecture in 2017 [38] revolutionised this field, leading to the development of Large Language Models (LLMs). Unlike previous models, LLMs excel at interpreting noisy, human-generated text [42].

The use of LLMs is no longer restricted to passive text generation. Through various prompting frameworks, LLMs can be utilised as the cognitive engine for ‘Agentic AI’ [39]. While a standard LLM generates text based on a prompt, an autonomous agent is designed to reason, plan, and execute tasks [21]. In the context of FM, an agent can go beyond merely classifying a work order to acting upon it.

To explore the use of AI agents for automating processes in FM, a solid framework for what defines an autonomous agent must be established. In the classical definition proposed by Franklin & Graesser [16], there are four core requirements that must be met for a computer program to be deemed an agent:

- **Reactive:** Must be able to react to changes in its environment.
- **Autonomous:** Must be able to control and decide its own actions.
- **Goal-oriented:** Must have a clear goal it works towards.
- **Temporally continuous:** Must be running continuously.

In the modern context of LLM-powered systems, however, the term is used a lot more loosely. Today, it often refers to any type of LLM pipeline that is capable of utilising reasoning, planning, tools, or communicating with other agents in a multi-agent system [40]. To formalise this, Wang et al. [39] proposed four interconnected modules that define an LLM-based AI agent:

- **Profile:** The profile assigned to the agent, describing its role (such as a "domain expert"), personality, and social information. This is most commonly defined in the agent's system prompt.
- **Memory:** The mechanism allowing the agent to store and recall information perceived in its environment. This can be implemented through in-context learning or through retrieval from vector databases.
- **Planning:** The ability to plan ahead and divide a complex task into suitable subtasks. This can be achieved through techniques ranging from simple Chain of Thought (CoT) prompting to complicated multi-path reasoning strategies.
- **Action:** The capability allowing the agent to act based on its decisions and interact with its environment. What constitutes an action is fairly open to interpretation, with a foundational example being the execution of a specific software function.

While these modules can be achieved through various approaches, Wang et al. [39] argue that all four are necessary to constitute a functioning agent. Rather than viewing these components as strict requirements during development, this framework will serve as the theoretical benchmark used to evaluate the developed artefacts in Section 4.3.

However, a technically advanced agent is useless if the organisation is not ready to actually use it. To move from theory to real-world deployment, it is necessary to first evaluate if the business environment can support these new workflows.

2.1.4 Organisational Readiness and Adoption

Going beyond theoretical research towards implementing AI within organisations presents a new set of challenges. Scaife [35] recommends viewing AI implementation in FM as a strategic endeavor, if AI projects are to be successfully integrated. To evaluate an organisation's readiness towards integration of AI workflows, Jöhnk et al. [23] developed a comprehensive AI readiness framework. Rather than viewing adoption purely as a technological challenge, the framework claims that readiness relies on five interconnected dimensions: strategic alignment, resources, knowledge, culture, and data. Using this framework to analyse the implementation gap within FM, three main challenges emerge: innovation culture, data readiness, and employee readiness.

Firstly, Jöhnk et al. [23] emphasise that an organisation's innovation culture relies on interdisciplinary communication. To uncover use cases for AI, an organisation must foster active collaboration between IT departments, AI specialists, and domain experts (such as facility managers and technicians) [23]. In the context of this thesis, this interdisciplinary alignment is essential to bridge the gap between technical capabilities and the practical nuances of building operations.

Secondly, the success of this innovation is limited by the organisation's data readiness and process maturity. Machine learning models are inherently data-dependent, meaning their performance mirrors the quality of the data they are trained on [23]. Scaife [35] suggests that one of the greatest hurdles to overcome in AI implementation is data availability, whereas Quinello & Costa [33] further argue the importance of data quality. Jöhnk et al. [23] highlight the importance of structured and standardised processes, as these generally output correct and complete data which is compatible with AI.

Thirdly, Jöhnk et al.'s framework highlights employee readiness as a bottleneck, arguing that successful implementation requires upskilling and a comfort with data-driven decision-making. However, as observed by Quinello & Costa [33], FM technicians are not usually familiar working with large-scale data analytics, so this may spark organisational resistance. To manage this friction, this thesis argues that AI-driven automation can be designed to minimise workflow disruption. By positioning the technology as a non-disruptive layer that operates silently within existing CMMS, meaning the staff does not need to actively engage with the it. Instead, the technology integrates into their established routines.

2.1.5 Human-Centered AI: Trust and Safety

Expanding on Jöhnk et al.'s framework, it is also important to consider the risks that come with non-deterministic models. Because LLMs are susceptible to stochastic errors and hallucinations, granting complete autonomy to an agent introduces risks. If a model misinterprets a maintenance request and uses its agency to make a financial expenditure or physical dispatch, establishing accountability becomes problematic [6].

Hosseini & Seilani [20] argue that a Human-In-The-Loop (HITL) architecture is suitable for organisations in the beginning stages of implementing AI for automation, as it is more controlled and allows employees to become comfortable and gain trust in the technology. HITL is an approach to designing AI systems where it is designed as a support system, suggesting actions which may be approved or rejected by human workers. Similarly, the expression Human-On-The-Loop (HOTL) describes an architecture where the model performs tasks autonomously but remains under the supervision of a human who can intervene to override decisions.

This approach is supported by Davenport & Ronanki [11], who emphasise the importance to divide decision-making between AI and humans based on the nature of the task, so that each part can play to its strengths and compensate for the other's weaknesses. Rather than viewing this division of responsibility as a compromise, Shneiderman [36] argues that human control and computer automation do not necessarily have to be two ends of a spectrum, but can coexist. Through human-centered design principles the full capabilities of AI can be leveraged whilst still allowing for humans to be in control of the outputs.

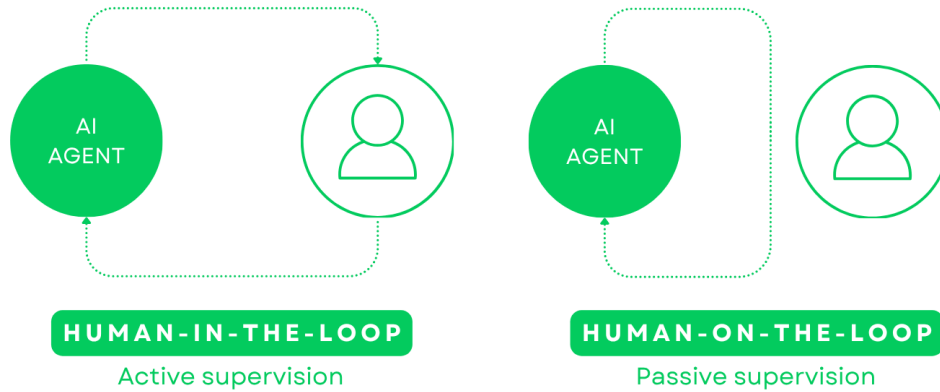


Figure 2.1: Conceptual illustration of the HITL and HOTL approaches

Within this thesis, this human-centered approach is used during the iterative prototyping sessions (Chapter 2.3), ensuring that the proposed solutions are relevant and suitable for future implementation, following the DSR framework.

2.2 Exploratory Data Analysis of MWOs

To get an initial understanding for the insights that could be gained from the historical work order data and the limitations and challenges of working with this data, exploratory data analysis was performed. This primarily consisted of overarching statistical and qualitative analysis, topic modeling and classification.

2.2.1 The Work Order Data

Initially, a dataset comprised of about 150,000 work orders, spanning the years 2016 to 2026, was extracted from Akademiska Hus' database for analysis. However, initial exploratory data analysis revealed that work orders filed prior to 2023 were classified using a different classification schema, leaving only around 77,000 from the timespan 2023-2026, usable in training a classification model, after removing duplicates and filtering out internal MWOs (filed by technicians).

The primary attributes used for analysis include the user's free-text problem description, spatial metadata (comprising Campus, Building, and Room attributes), the priority level (Low, Normal, High), and the dual-stage classification: the initial symptom class and the final root-cause class ("felorsak").

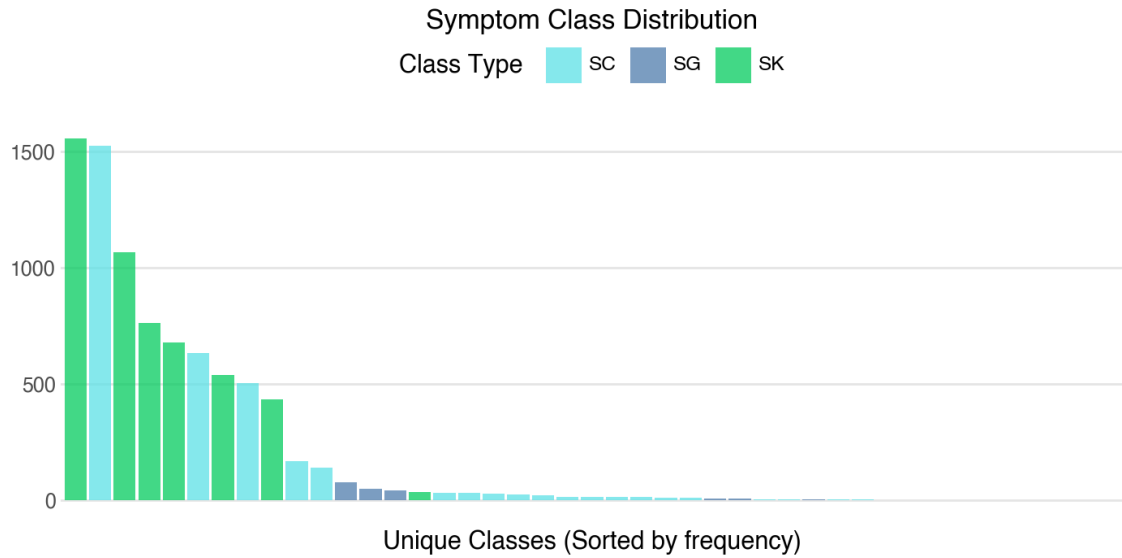


Figure 2.2: Class distribution of the MWOs. SC: technical classes (e.g. "broken door"), SK: perceived classes (e.g. "stuffy air"), SG: alarms and technical surveillance

Both the symptom and root-cause attributes use a shared hierarchical taxonomy consisting of 17 top-level classes, which expand into 24 classes at lower sub-levels (see Appendix A for the complete taxonomy). Notably, six of these domains pertain exclusively to indoor environmental quality (IEQ) anomalies (e.g., thermal or ventilation symptoms). These IEQ classes are strictly reserved for the initial symptom class, as a finalised technical root cause cannot logically be categorised as "stuffy air." As illustrated in Figure 2.2, the dataset shows extreme class imbalance across these 24 categories, presenting a significant challenge for traditional classification models.

Fig 2.3 shows that the length of the user's free text description varied significantly, with an average of approximately 36 words. The shortest MWO in the dataset was comprised of the single word "Stopp", signifying that a toilet was blocked. Further qualitative analysis of the dataset revealed that the tone varied greatly. Some MWOs were written as notes from other technicians, simply stating ID's of armatures that needed replacing, whereas other's were written by furious occupants ranting about the temperature in their office.

2.2.2 Data Preprocessing and Cleaning

Before further analysis of the data could be performed, some preprocessing was required. As a first step, the `langdetect` library [10] was used to detect the language of all the documents. Approximately 95.2% of the documents were detected to be in Swedish, which was deemed to be a large enough portion that the analysis could be limited to only documents in Swedish. The `spaCy` library [18] was used to lemmatise the texts, as well as remove stop words, punctuation, numbers, and make lowercase. Stop words included the standard set of Swedish stop words, along with some context specific stop words, manually selected after reviewing the documents.

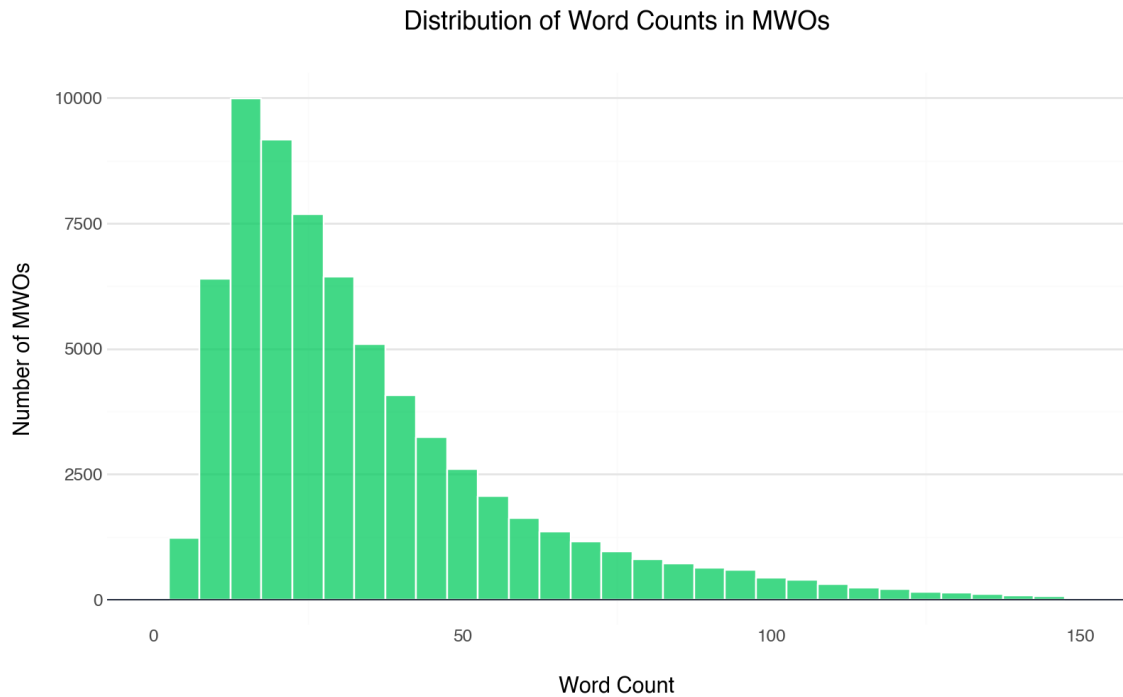


Figure 2.3: Histogram showing the distribution of word counts of the user’s free text submissions. The histogram is cut as 150 words, although there were instances of longer MWOs in the dataset

Place names were, to some degree, removed by filtering out any words ending in "gatan" or "vägen".

2.2.3 Topic Modeling: Exploring clusters and patterns in the data

Topic modeling is an unsupervised machine learning technique similar to clustering, used to group text documents. Instead of looking for distinct clusters in the feature space it looks for "topics". These are found by looking for words that often appear together and grouping them together into topics[14]. Words can belong to more than one topic and documents are assigned topics based on the words they contain. The purpose of this was to determine the main topics of the maintenance requests, to investigate whether the classes in the data represented the actual topics that users were talking about and whether the needs of the users were being properly addressed by the facility managers.

Several different models were evaluated for this task, both statistical models and transformer models, but the best results were achieved with multilingual BERTopic. The full topic modeling pipeline consisted of the previously described preprocessing, a count vectoriser, followed by multilingual BERTopic.

Topic modeling revealed that the predefined classes cover most of the topics found

Topic	Count	Percentage (%)	Top Words (Translated)
0	4131	5.7%	ventilation, bad, air
1	2659	3.7%	window, close, glass
2	2236	3.1%	noise, annoying, loud
3	2094	2.9%	lift/elevator
4	1529	2.1%	toilet, water, flush
5	1326	1.8%	smell, bad, drain
6	975	1.4%	lift, still, standing
7	922	1.3%	dishwasher, machine, dishes
8	805	1.1%	interior, building, contact, department
9	798	1.1%	students, teachers, bad, air

Table 2.2: The top 10 topics found in topic modeling

in the text, meaning the existing taxonomy is comprehensive enough to capture the core issues. However, the analysis highlighted an ontology gap between the language used by occupants and the technical structure of the classification scheme. Users tend to focus on objects that appear to be broken (such as coffee makers, dishwashers, doors) or disruptions to the environment which make it difficult to work (such as sounds, smells, temperature differences).

The taxonomy captures these environmental disruptions to some degree through the IEQ classes, corresponding to "Cold room", "Hot room", "Bad Air", "Noise" and "Bad lighting". However, the text data shows that these categories contain a lot of overlap. For instance, complaints regarding "Hot room" and "Bad air" frequently appear within the same maintenance request.

A similar misalignment is present regarding physical property damage, such as doors, appliances, and lighting. While the topic modeling shows that users frequently report specific items like microwaves or dishwashers, the formal classification scheme groups these issues based on the required technical expertise or underlying building system. Therefore, specific appliances are categorised under broad umbrella terms such as "Furniture" or "Appliances". From an operational perspective, grouping these components under broad categories like "Interiors" or "Furniture" aligns with how maintenance expertise is dispatched, but it creates an abstraction layer that hides the specific asset types mentioned in the raw text.

Akademiska Hus handles this workflow in two stages: an MWO is first classified based on symptom, and later assigned a root cause code once the issue is resolved. The topic modeling reveals that while the classes are functional to classify the final root cause (since they cover all possible faults in a building), they lack specific granularity when it comes to classifying the initial symptom. This confirms the mismatch between the operational symptom classification scheme and the descriptive language in the users' requests.

2.2.4 Initial Modeling: What do the classes look like?

To establish an understanding for how well a traditional ML algorithm can handle the maintenance request data, and to gain insight into the challenges which may be encountered during development of a more complex model, a classification model was trained to classify the symptom class based on the free text portion of the maintenance requests. This initial modeling was done as a part of data exploration, specifically to understand the class structure and how the classes correspond to the topics found during topic modeling.

The classification scheme consists of classification at three levels of detail. Data which was only classified at the top-level was left out of the training, since the classification guide used by technicians specifically required FMs to not use umbrella categories for classification. The dataset was split into 5 folds for cross validation, using stratified sampling. The texts were then tokenised using TF-IDF vectorisation. Lastly a Linear SVC , which has been shown to perform well on similar data [29, 34], was trained and evaluated at each step.

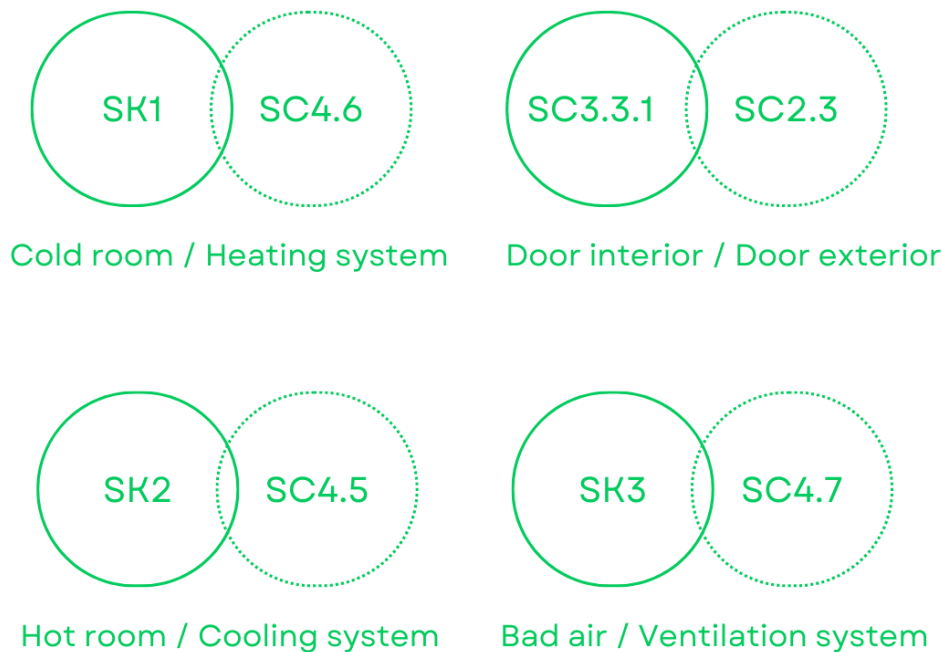


Figure 2.4: Illustration showing commonly overlapping classes. The dotted circles represent the minority class which is often confused with the other class.

The evaluation of this model revealed significant class overlap, highlighting fundamental challenges within the dataset. It became apparent that a large portion of the documents could objectively fit into more than one class, making strict classification problematic without additional heuristic rules (e.g., "always prioritise classes describing changes in indoor climate"). Fig 2.4 illustrates the most commonly over-

lapping classes, which directly reflects the overlaps identified during topic modeling in Section 2.2.3.

A qualitative analysis of texts that were misclassified with high confidence, alongside a random sampling of the dataset, revealed that this ambiguity primarily stems from three underlying causes:

- **Symptom vs Root Cause:** As seen in Section 2.2.3 users frequently describe the symptom they experience while simultaneously predicting the root cause. In such cases, the text matches both symptom-based classes and system-based classes. This is the most common source of confusion for the model, particularly between indoor climate symptoms and specific HVAC systems. Since the word "ventilation" appears frequently in two classes, it struggles to differentiate them.
- **Compound Issues:** A single maintenance request can contain multiple problems, or a chain reaction of events (e.g., "a leaking tap" leading to "water damage in the wall"). This is sometimes caused by a user wanting to reduce administrative effort by reporting several faults at once, which is something the current classification scheme is unfit for.
- **Under-specification:** Users generally lack the technical vocabulary and knowledge of details required by the classification schema. For instance, a user might simply report a "broken door," but the taxonomy requires the model to differentiate between interior and exterior doors.

2.2.5 Summary of EDA

In conclusion, the exploratory data analysis establishes three main constraints for the development of the AI artefacts in the subsequent phases:

- **Data Volume:** After filtering and removing internally generated maintenance records, a viable dataset of approximately 77,000 MWOs remain for training and evaluation.
- **The Structural Mismatch (Ontology Gap):** Topic modeling confirms that the information structure of occupant requests does not match the classification requirements of the CMMS. User requests focus on environmental symptoms or high-interaction assets (e.g., thermal discomfort, domestic appliances). However, the CMMS classification scheme is designed around asset responsibility and specific technical domains.
- **Limitations of Single-Label Classification:** Qualitative review of the misclassified requests reveals that MWOs frequently belong to more than one

class. Requests frequently combine an observed symptom with a suspected root cause, or describe a chain reaction of multiple physical faults, while the existing CMMS data structure demands one label per work order.

2.3 Iterative Prototyping Sessions

Using the DSR paradigm to ensure that the solutions were designed to respond to real needs in the organisation, the idea development phase was an iterative process, cycling between interviews and feedback sessions with stakeholders, and development phases diving deeper into the data, to get an understanding for possibilities and limitations.

The iterative process started out by understanding the underlying architecture of the system that the artefacts would be interacting with: The As-Is Architecture. The next iteration consisted of identifying bottlenecks through discussions with those working with the system, and finding solutions. The last iteration consisted of feedback to those solutions, analysing prerequisites needed to build the solutions, and thereby narrowing it down until a final selection of two solutions remained to be built.

2.3.1 Iteration 1: The As-Is Architecture

Before delving deeper into the bottlenecks, the system architecture itself had to be understood. Through interviews with facility managers and other key persons, the following insight was gained.

Currently, maintenance requests are filed by users through a web form on Akademiska Hus website. The user must fill in the following mandatory fields:

Table 2.3: Specification of MWO Input Fields

Field Name	Input Type	Constraints / Description
Title	Free text	Maximum 80 characters.
Problem description	Free text	Maximum 3000 characters.
Campus	Drop-down menu	Predefined list of university areas.
Building	Drop-down menu	Filtered based on selected campus.
Room	Free text	No character limit.
Contact information	Free text / Form	User contact details.

The request is then entered into Faciliate as a work order. Faciliate is a CMMS used by facility managers at Akademiska Hus. It allows technicians to handle MWOs and view documents and data relating to the facilities (such as the database containing information about installed IoT units). When first receiving a work order in Faciliate, a technician reads the text and sets a priority as well as an initial problem class,

also referred to as the symptom class (since the user tends to report the perceived symptom of some technical problem). Once the technician has dealt with the issue, they document it by setting another problem class, referred to as the "felorsak" (that is, the root problem which caused the symptom, often one and the same) and submitting a message to let the user know the issue has been resolved. This message may include details about how the problem was resolved.

In some cases external contractors are called in to help resolve issues (for example if it is beyond the in-house technicians capabilities or if the work order might take too long to complete). In this case, the technician puts in an order in another application called Kommers. This process is currently being reviewed, with plans of being moved into Faciliate. In the case where technicians need to examine sensor data, this is done in separate programs (depending on the system installed in the relevant building).

The following diagram was produced to illustrate the flow:

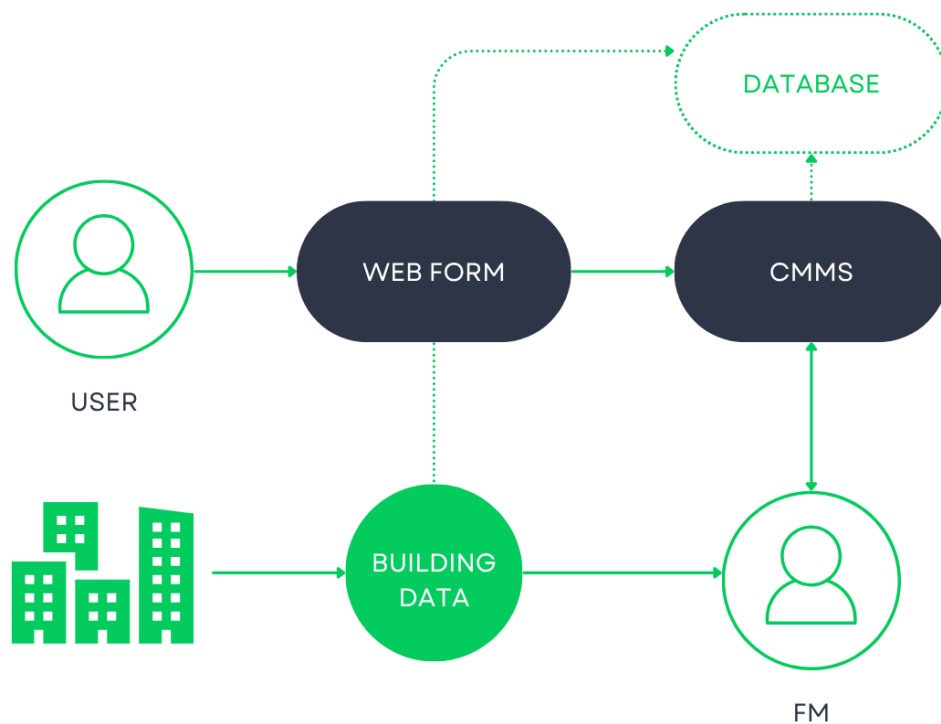


Figure 2.5: Flow chart illustrating the system architecture

2.3.2 Iteration 2: From Pain Points to Prototypes

In the first iteration of prototyping the goal was to identify bottlenecks and other areas for improvement in the system. In alignment with the initial ambitions of Akademiska Hus, the second iteration of the prototyping sessions focused mainly

on autonomous agents. The stakeholders expressed an interest in exploring how autonomous systems could reduce the manual administrative load within the CMMS environment. Therefore, the six initial conceptual artefacts were designed as autonomous or semi-autonomous entities, each assigned a specific operational profile and expected to independently execute tasks or communicate with users.

It became apparent through the interviews that there were many administrative tasks, currently done by the FMs, which could easily be automated or streamlined. Among these tasks were classification and prioritisation of the error reports. Moreover, the CMMS itself lacked a holistic overview where ongoing work orders could be easily viewed. If an FM had 10 active MWOs, they would only be able to identify them by location and priority, forcing them to re-read each MWO to remind themselves of the issue. Lastly, it was pointed out that it was very common for MWOs to be misrouted. When this happens, the FM must manually find the boundary demarcation agreement (BDA) between AH and the tenant to determine who bears the responsibility for resolving the issue. If the MWO is outside AH's domain of responsibility, the technician usually has to manually reroute the MWO to the correct place and in some cases has to act as middleman for communication between the initial filer of the complaint and the receiver of the MWO.

These issues go hand-in-hand with problems identified during the EDA (Section 2.2). The Linear SVC model struggled with class overlap and the "ontology gap" between user descriptions and technical CMMS classes. By automating triage and classification of the requests, the class structure might become more standardised, leading to better data quality for future digitalisation efforts or statistical analysis.

These insights generated the ideas for two artefacts:

- **The Structurer:** Structures the MWOs by automatically classifying and prioritising them.
- **The Demarcation Agent:** Automatically retrieves the correct BDA and compares the maintenance request against this list, so that misrouted requests can be rerouted before they enter the CMMS

Because the FMs are bogged down by the administrative work above, communication suffers. Moreover, the system itself does not support continuous dialogue.

One of the main takeaways from the interviews was the lack of communication channels between the users and the technicians. Currently, the technician can only contact the user via the CMMS in the closing message left once the MWO is completed. If the technician has follow-up questions or needs to contact the user for other reasons the technician has to email or phone the user, using the contact information given in the maintenance request. This correspondence does not always have a formal connection to the MWO and therefore, if for example the MWO needs to be reviewed at a later date, this information, which may be relevant to the resolution, is left out of the formal documentation of the case.

Based on this, the following artefacts were proposed:

- **The Follow-up Agent:** Analyses incoming maintenance requests to identify common issues + missing information, and immediately asks the user relevant follow up questions
- **The Informer:** Supplies user with important information, such as expected time to resolution, things they can do to "fix" the issue themselves if applicable and possibly send automatic updates

As stressed in Section 2.1.1, communication is one of the key factors towards user satisfaction with FM [1], which further justifies *The Follow-up Agent* and *The Informer*. *The Follow-up Agent* was also conceptualised in part in response to Hopland & Kvamsdal's [19] CSF pertaining to clear instructions for technicians. By automatically retrieving missing information from the user, this increases attentiveness as well as clarifying important details which lets the FM work more efficiently.

Lastly, predictive maintenance was a topic that appeared often throughout the literature review, and whilst a lot had been done pertaining to predictions based on IoT data, not much was found connected to MWOs. Based on this idea, two artefacts were proposed, exploring how MWOs can be used to give insight into future operations:

- **The Forecaster:** An agent with access to historical work order data, instructed to find patterns and predict upcoming maintenance requests (e.g. if someone complains about X this week, there will be another complaint about Y a few days later). Or analysing work orders to find hidden patterns/root causes (e.g. many people complaining about a smell or bad air in a building -> may be caused by the same root problem)
- **Weekly Prediction Report:** Agent that sends out a weekly report to facility managers, giving insights into expected complaints (for example based on weather and historical patterns). Many different data-sources could be added to this, such as the universities schedules, warning about upcoming events.

2.4 Feedback and Selection

In the second iteration of prototyping, the selection of artefacts was narrowed down through stakeholder feedback, measuring the value using KPIs (from literature review), as well as further analysis of the available data to determine feasibility.

Through stakeholder feedback, *The Forecaster* was eliminated due to low perceived operational value. Subsequently, the two communication-focused artefacts, *The Follow-Up Agent* and *The Informer*, were also eliminated. Although there was consensus that these artefacts could significantly improve *Service Quality* (as proposed

by Abisuga et al. [1]), their feasibility was debated due to the lack of data readiness.

To function effectively, both artefacts required details pertaining to how specific facility issues are resolved. An exploratory review of the historical "closing comments" logged by facility managers revealed that the data quality varied too widely and lacked sufficient technical detail. Furthermore, it was unclear whether standardised internal guidelines or policy documents existed to make up for this gap. While an alternative solution would be to manually draft new documentation specifically to support these artefacts, the primary objective of this project was to leverage existing historical data. Consequently, in alignment with the data readiness limitations highlighted by Jöhnk et al. [23], these communication artefacts were set aside for future development.

Lastly, the remaining artefacts were evaluated against the previously defined KPIs, alongside a more qualitative analysis of feasibility and immediate business value. Through this analysis the *Weekly Prediction report* was eliminated, as it didn't directly target any of the needs identified in the previous iteration, leaving *The Structurer* and *The Demarcation Agent* as the final two artefacts. These two directly targeted two of the main bottlenecks identified, and by reducing these bottlenecks, the service can run more efficiently, which directly connects back to KPIs identified in 2.1.1.

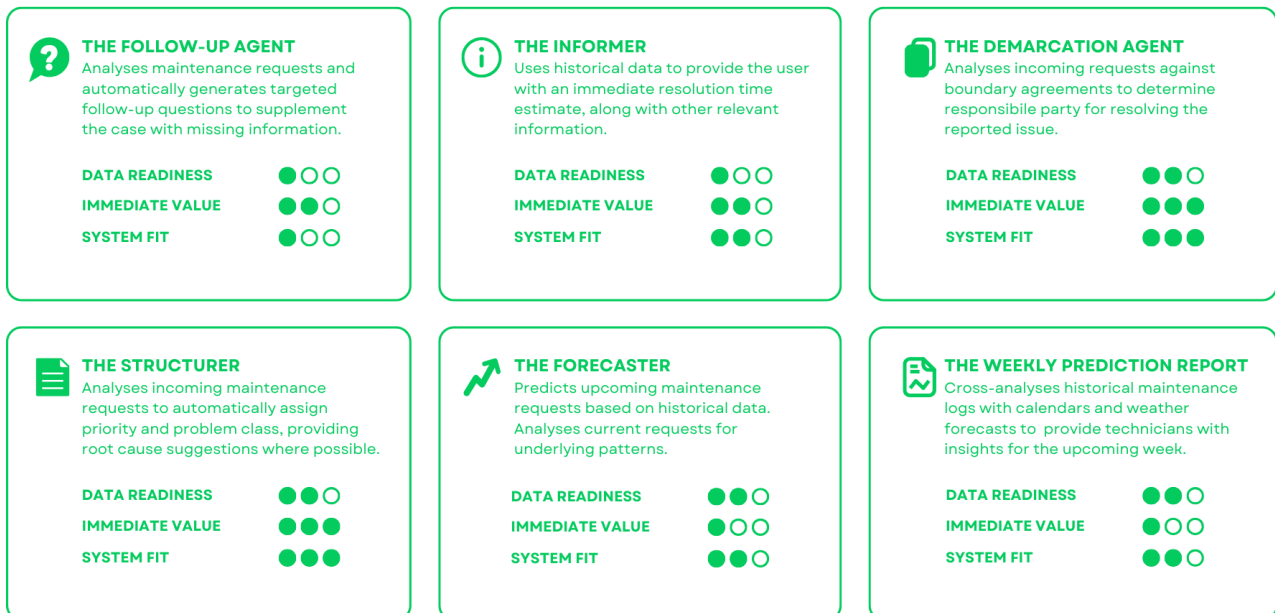


Figure 2.6: Evaluation of the proposed architectures against the selection criteria.

2.4.1 Summary and Next Steps

In summary, Phase 1 establishes the baseline operational and technological understanding required to build the automation pipeline. The phase resulted in the following key outputs:

- **Literature Review:** Provided the theoretical foundation for the project including the current state of FM and digitalisation efforts, organisational readiness for AI, AI approaches to minimise friction, and how success is measured in FM.
- **Exploratory Data Analysis:** Revealed the structural limitations and opportunities within the historical dataset, exposing data readiness challenges such as class imbalances and noisy human labeling patterns while establishing an initial statistical machine learning baseline.
- **Iterative Prototyping Sessions:** Stakeholder and technician interviews provided an understanding for the system architecture and bottlenecks. Thereafter an evaluation of initial automation concepts was carried out, leading to the two final artefacts: *The Structurer* and *The Demarcation Agent*.

The next steps involve the development and testing of these two selected artefacts, followed by a formal evaluation against organisational needs, KPIs, and performance metrics.

3

Phase 2: Develop and Deliver

This chapter covers the Develop and Deliver phases of the project, focusing on the development and implementation of the selected artefacts. Specifically, it details the design and iterative refinement of *The Structurer* and *The Demarcation Agent*, aimed at automating the classification, prioritisation, and filtering of MWOs.

3.1 The Demarcation Agent

As identified during Phase 1, the manual rerouting of misdirected requests constitutes a significant administrative burden. To automate this process, *The Demarcation Agent* was developed. The purpose of this agent is to function as a routing filter, evaluating incoming requests against contractual demarcation agreements to determine the responsible party. By autonomously performing this cross-referencing, the agent eliminates the need for manual reviews, streamlining the workflow of the facility managers.

3.1.1 The Data: A look at boundary demarcation agreements

A boundary demarcation agreement (BDA) is a part of the contract signed when a property is leased. This document details the distribution of maintenance responsibilities between the property owner (lessor) and the tenant.

At Akademiska Hus this document is structured as a table, where each row corresponds to one building component. For the purpose of building the PoC, a sample of 12 BDAs were studied, both older and more recent documents. The format of the tables varied throughout the dataset and some BDAs were appended to the full contracts, instead of saved as separate documents. In these cases there was usually more than one BDA (which can happen if certain areas of the building, such as lab environments, need to be treated differently). Lastly, all the documents had been scanned from paper. The first step taken was therefore to transform the documents to markdown using OCR, which was done with Azure’s Document Intelligence.

3.1.2 Architectural Decisions: RAG vs Direct Injection

Since the boundary demarcation agreements were not always saved as separate documents and since there was sometimes more than one agreement for the same facility, the first step was figuring out how to extract and store the documents so that they could easily be accessed by the agent.

Initially, a RAG(retrieval augmented generation)-pipeline was evaluated on the documents, since this required no modification to them. This configuration presented several technical limitations. Partly, because the boundary agreements are formatted as tables, which can accidentally be split during the chunking phase, causing the model to lose important context. But more importantly, the entire contract consists of very similar terminology and there are many sections that introduce high semantic noise, which could be confused for boundary demarcation agreements (such as the similar agreement pertaining to fire-safety). This introduced retrieval errors as a new source for, alongside the baseline stochastic errors and hallucinations of LLMs.

The RAG Pipeline was evaluated on the full annotated test set as a reference, but at this point the focus shifted to Direct Injection, where the LLM is provided with the entire document within its context window. As noted by Li et al.[26], Direct Injection often outperforms RAG since the model has access to the full context, avoiding retrieval errors.

Direct Injection, however, is limited by the context window size and the "Lost in the middle" phenomenon. Liu et al. [27] demonstrated this phenomenon, where LLMs often struggle to retrieve information located in the middle of long contexts. Although the performance varies between models, providing excessive irrelevant context introduces both cognitive load for the model and significant increase in model costs.

The context window of modern LLMs (such as GPT-5.1 at 400k tokens) easily accommodates for the full leasing contracts, which are typically under 200KB after converting to markdown, or approximately 50k tokens. But processing 30,000 requests annually, with 50k tokens of mostly irrelevant text per request, introduces a significant increase in model/API costs. Therefore, to mitigate both the risk of the "Lost in the middle" problem and excessive token costs, a preprocessing pipeline for extracting the boundary demarcation agreements was designed.

Preprocessing

As full system integration with the existing IT infrastructure falls outside the scope of this thesis, the architecture was developed and evaluated as a localised Proof-of-Concept (PoC). The primary objective was to validate the core logic and semantic reasoning of the agent. To optimise resource allocation by avoiding the architectural overhead of RAG and the high token costs of injecting full contracts, the final architecture was split into two processes: a preprocessing step for data extraction,

and a pipeline for live routing.

To isolate the boundary demarcation agreements from the contracts, the full, OCR-processed markdown files were sent to an LLM with a large context window. The model was prompted with strict instructions to return only the tables detailing maintenance responsibilities with their immediate surrounding context. To simulate the production environment's document retrieval system without creating a redundant database, the resulting agreements were saved locally and mapped to their respective property IDs using a static JSON dictionary, allowing for programmatic retrieval of the agreements in the main pipeline.

Before moving onto the main pipeline, the accuracy of the preprocessing was evaluated. If the extraction step modified the BDAs in any way or failed to extract the correct parts of the document, the rest of the pipeline would fail. The evaluation was done by assessing the performance on 5 BDAs of varying format. By manually extracting the BDAs and then comparing with the output from the LLM (using the Python `difflib` module), the results could be analysed to ensure that no important information was modified or left out. The extraction pipeline demonstrated high reliability, in the 5 BDAs tested, no information of critical importance was left out or modified. In all cases the responsibilities and the context surrounding the tables were preserved exactly as is. The manually extracted BDA and the BDA extracted by the LLM had a similarity of above 90% in all cases, with the LLM having removed unnecessary details, such as page headers. With the success of the preprocessing verified, the main pipeline was built.

The Direct Injection Pipeline

In the simulated environment, the input of a maintenance request triggers the workflow. First, the pipeline simulates a database query by passing the facility's Property ID to the JSON mapping, which retrieves the corresponding boundary demarcation agreement.

Next, a model with high reasoning capabilities (GPT-5.1) is invoked via Direct Injection. The system prompt is written with strict instructions to only use the boundary demarcation agreement to answer whether the maintenance request should be handled by Akademiska Hus or the tenant. To enforce deterministic outputs and reduce hallucinations, the model temperature is set to 0 degrees, and the model is required to return a JSON object containing two fields: a motivation and the responsible party. Generating the motivation first forces the model to map the user's text to a specific extract from the demarcation table, before making its final decision. The full pipeline is illustrated in Fig 3.1.

Furthermore, to handle the ambiguous requests identified in Phase 1, the model is required to output "Unsure/unclear" in cases where more information is needed to answer the question, such as the case where a door is broken but it is unclear whether the door is interior or exterior. This ensures that ambiguous cases fail safely

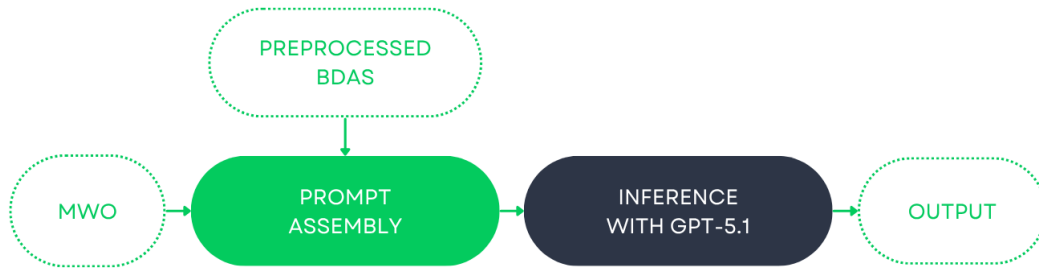


Figure 3.1: Flowchart showing the Direct Injection pipeline of *The Demarcation Agent*

and can be reviewed manually.

Three different prompting strategies were evaluated:

- **System Design:** This involved giving the model a specific "role" as an expert facility manager and highlighting the
- **Chain-of-Thought:** Instructing the model to break the task down into pre-defined steps, in hopes of improving accuracy.
- **Few-Shot:** Giving the model several examples of MWOs and their expected outputs as a reference (worth noting that this bears the risk of skewing the responses towards the output given in the examples).

The prompts were formulated iteratively to ensure good performance on edge cases and ambiguous inputs, without overfitting. This process revealed that the models were quite sensitive to small changes in prompt formulation.

3.1.3 System Integration (Future Work)

In a fully deployed production environment, the localised PoC would be integrated directly with the existing CMMS infrastructure. To achieve this, the core inference logic would be moved to a serverless architecture, such as Azure Functions, enabling event-driven execution triggered by incoming requests. Furthermore, the local mock environment would be replaced by a cloud-based repository (e.g. Azure Blob Storage or Microsoft Sharepoint) to host the preprocessed documents. Finally, the JSON output generated by the LLM would be pushed via the CMMS API, alongside the original web form submission.

3.2 The Structurer

The second artefact selected for development was *The Structurer*. The idea behind it was to have the artefact structure the freetext maintenance request into a format that is more easily overviewed by the FM. This included classifying the problem based on AH's classification scheme, setting a priority level, and structuring it into a readable format. The idea was also to include a feedback loop, where the FM

could disagree with the artefact’s evaluation, allowing for the artefact to improve with time.

Although the task could very well be done using an LLM, since there were many years of historical data available, it seemed a better idea to first investigate whether a classic ML-model could be trained on the classification tasks. Thus, the artefact was split into two completely separate ML pipelines: one for symptom class classification and one for priority classification. In this section the development phases of those two pipelines are presented.

3.2.1 The Symptom Model

The classification model for classifying symptom built on the initial baseline model developed during Phase 1. The performance of this model could use some improvement and the issue of the overlapping classes and ambiguous maintenance requests needed to be addressed.

To better help the model understand the features of each distinct class, mislabeled and ambiguous documents, such as those identified as belonging to several classes in Section 2.2, were filtered using the `cleanlab` library [8]. Cleanlab builds on Northcutt et al.’s [31] Confident Learning framework to identify label errors. This works by using an ML model to compute out-of-sample predicted probabilities via cross-validation, flagging instances where the assigned label strongly disagrees with the model’s prediction.

In this case, a classical Logistic Regression model was implemented as the baseline for the data cleaning phase. Both LinearSVC and Logistic Regression have been shown to perform efficiently on similar tasks, classifying high-dimensional text data [29, 34]. During initial data exploration, the LinearSVC yielded slightly better classification metrics and was therefore selected as the primary machine learning model for the final pipeline. To avoid reinforcing the model’s own biases during data cleaning and filtering, Logistic Regression was used as an independent architecture to identify and filter out historical label anomalies with `cleanlab`. Since the dataset contained some very small classes, protocols were put in place to award immunity to small enough classes.

Table 3.1: Effect of data cleaning on training splits. Values represent the absolute sample counts and the calculated mean across the 5-fold cross-validation loop.

Dataset Partition	Sample Count (N)
Full Raw Dataset	9,770
Mean Initial Train Fold	7,816
Mean Cleaned Train Fold	6,780

To prevent data leakage, this cleaning was only done on the training set of each iteration of the cross-validation loop. The test-data remained raw, to better replicate the results of the model on real-world data. Furthermore, to establish a robust and fair dataset for evaluation, a separate, distinct test-set was taken out of the main dataset before model training started, which was manually annotated.

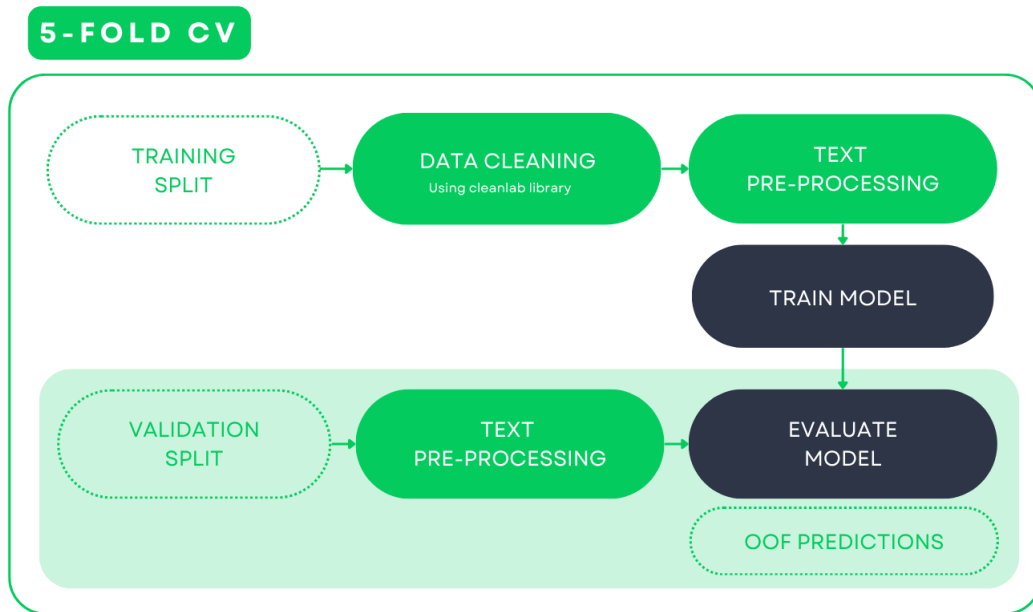


Figure 3.2: Main pipeline for training the symptom model. The model components represents any of the models mentioned

With the training data cleaned, several models were evaluated to find the best fit for this specific task. As stated, LinearSVC was used as the baseline statistical model. To investigate whether poor classification performance was a limitation of model complexity rather than data quality, two other models were evaluated. First, the Swedish KB-BERT [24] was introduced as a semantic middle-ground between classical machine learning and generative models. Unlike bag-of-words approaches (such as the SVC and Logistic Regression), KB-BERT leverages a transformer-based architecture capable of capturing context, syntax, and sentence structure. Finally, a zero-shot LLM pipeline was introduced.

These three models each provided different insight into the qualities of the data and the type of architecture needed to classify them. Classical models like the LinearSVC are deterministic, cheap to evaluate, and fast enough to run as lightweight, low-latency microservices. In contrast, generative LLMs introduce variable API token costs and higher operational latency, but they offer runtime rule-enforcement via prompting. This allows the system to remain insulated from historical data corruption and class overlaps, providing a baseline for what can be achieved without relying on historical human classification patterns.

The KB-BERT and LinearSVC were trained using the cross-validation setup shown in Fig 3.2 and the evaluated on the annotated test set. The LLM was evaluated directly on the test set, injecting the classification guidelines into the system prompt.

3.2.2 The Priority Model

Initially, this task was hypothesized to be a less complex machine learning task than the symptom classification. The historical maintenance request data included ground-truth labels for five priority classes. Given the lower dimensionality of the target variable, 5 classes vs the 24 symptom classes, and the intuitive assumption that the boundaries between varying levels of urgency would be more distinct, the classification seemed straightforward

After analysing the dataset further, substantial challenges emerged. The classes were extremely unbalanced, with 97.5% of the maintenance requests belonging to the 'Normal' class. The smallest class only had 56 samples, which wasn't nearly enough to define clear clusters in the raw input space, as demonstrated during initial exploratory runs using a Linear SVC.

To combat this issue, the dimension of the input was reduced using Chi-Square feature selection, which ranks the importance of the features through a chi-square test between the features and labels. This revealed that the model was sensitive to noise, such as specific building or place names, which correlated statistically with certain priorities during feature ranking but lacked semantic relevance. Further steps were taken to filter out these geographical markers, yet this yielded marginal improvements in classification performance.

The imbalanced data and the high dimensionality, suspected label noise in the historical record, made it difficult to produce operationally viable results. At this point a round of feedback with stakeholders revealed that the prioritisation was not actively integrated into current workflows. Therefore, rather than iterating further toward a production-ready implementation, the priority pipeline was repositioned as an experimental baseline to evaluate data readiness constraints. While the prioritisation task was initially conceptualised as a straightforward complement to symptom classification, the experimental evaluation revealed that the historical data lacked the readiness required for reliable priority classification. Therefore, the decision was made to exclude the priority model from the final system design, as its marginal operational utility did not justify further optimisation efforts.

3.2.3 System Integration (Future Work)

It is still possible to implement the symptom classification as a part of the maintenance request triage. An alternative solution would be to integrate the classification into the Demarcation Agent, using the LLM for classification.

4

Results and Evaluation

In alignment with the Design Science Research (DSR) paradigm introduced in Chapter 1, the developed artefacts must be evaluated through both technical means and operational relevance. This chapter presents the quantitative technical performance of the models alongside their qualitative performance within the organisation. Further theoretical insights and research value will be explored in Chapter 5.

4.1 Technical Performance

4.1.1 The Demarcation Agent

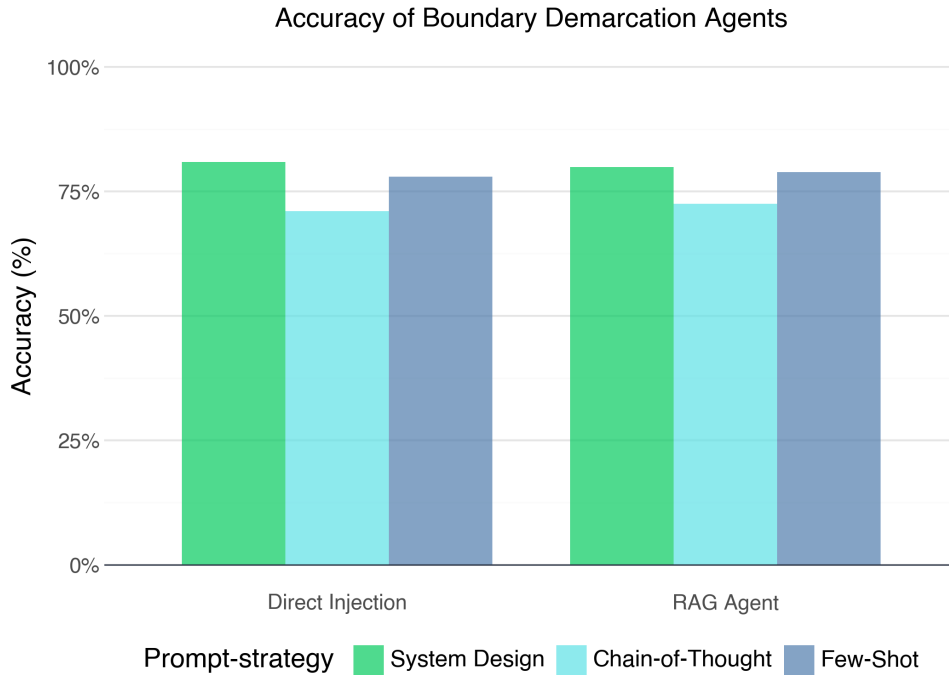


Figure 4.1: Comparison of accuracy between different combinations of model architecture and prompting strategy

Since there was no way of telling whether historical maintenance requests had been rerouted or not, a dataset of 200 MWOs were sampled from the historical dataset and manually annotated. This dataset included MWOs from all different symptom classes, to ensure broad coverage. Additionally, four edge cases were used, testing

the models on difficult MWOs, prompt-injection and blank texts.

The two different pipelines, detailed in Chapter 3.1, were evaluated on this set, primarily looking at accuracy. For each pipeline architecture, three different prompting techniques were evaluated: System Design, Chain-of-Thought, and Few Shot. The results of this evaluation are presented in Fig 4.1. This shows that the different architectures are comparable in accuracy, with performance mainly being tied to prompt-strategy. This suggests that refining the prompts might further improve the model, which is supported by the high sensitivity to prompt design encountered during the development phase.

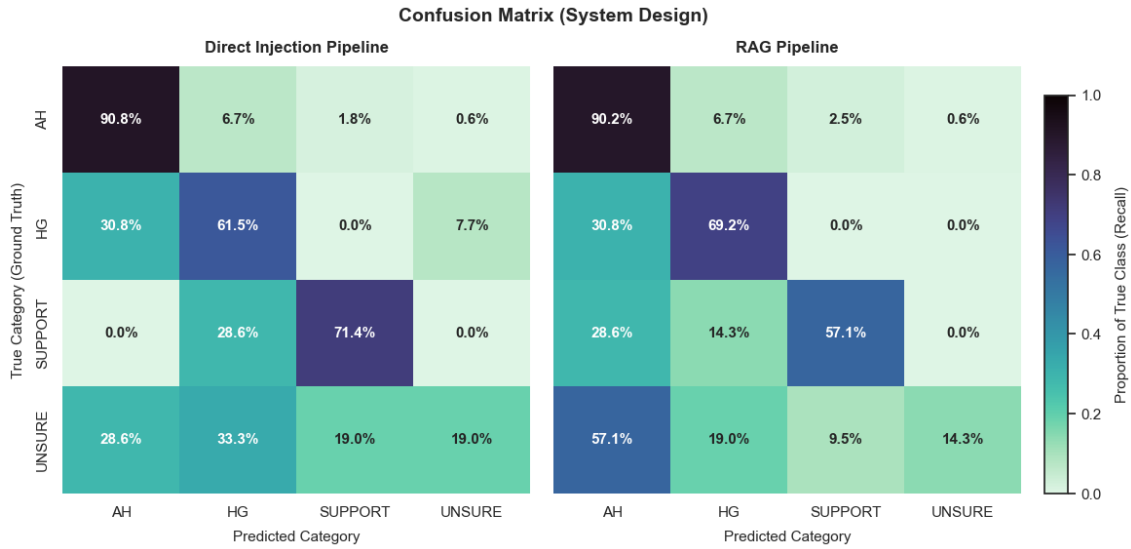


Figure 4.2: Normalised Confusion Matrix based on predictions using System Design prompt

Table 4.1: Cost and time analysis of The Demarcation Agent (Mean value per MWO, $N = 200$).

Architecture / strategy	Average Tokens	Average Latency [s]
Direct Injection (DI)		
System Design	13 526	2.37
Chain-of-Thought	13 778	4.53
Few-Shot	13 643	2.12
RAG Agent		
System Design	10 407	2.47
Chain-of-Thought	10 635	6.52
Few-Shot	10 710	2.69

Fig 4.2 shows that the Direct Injection Pipeline achieves higher recall for most classes. The performance of the two pipelines is comparable, although the RAG

pipeline more commonly misclassifies MWOs as AH’s responsibility. This can be explained by the fact that all prompts contained a clause stating that if the responsibility was unclear given the context, the model should rely on industry standard. If the RAG pipeline retrieved the wrong part of a BDA, it may not have the correct context and might then make an inaccurate generalisation. From an architectural perspective, this demonstrates that increasing system complexity via retrieval mechanisms does not necessarily yield better performance.

Aside from the accuracy, the latency (average time per MWO being processed) and the token cost were of interest. This is presented in Table 4.1. This shows that RAG is slightly cheaper, but not by much, and that latency primarily depends on prompting strategy.

As the test set was small, the misclassifications were reviewed manually. This analysis revealed several key difficulties. Summary of this is presented below.

- **Insufficient Detail:** User-descriptions often lacked the detail required of the BDA to define a responsible party. Some common examples include broken doors (which are usually owned by the property owner, but have complicated boundaries surrounding the locking mechanism), toilets and plumbing (responsibility often depends on where the blockage has occurred), light fixtures (responsibility usually depends on whether they are fixed installations or not)
- **Insufficient BDA:** Many of the BDAs omitted boundaries surrounding key building parts, such as windows, inner and outer walls, and electrical sockets. In these cases the BDA would usually state that the given boundaries were for parts of the building "affected by the tenant’s activities". This is interpreted to either mean that anything that isn’t mentioned in the BDA falls on the lessor, or that the objects that are not mentioned in the BDA should follow industry standard. Either way, this was difficult for the model to interpret and would need refinement in future work.

These failure modes highlight a challenge for automation. Because user descriptions are often under-specified relative to legal boundaries, the errors stem from a lack of information rather than algorithmic failure. This justifies the necessity of a HOTL design, to mitigate risks from misrouting.

4.1.2 The Symptom Class Model

Before the initial training of the model began, a completely untouched test set of 200 MWOs were withheld. This test set was manually annotated and the models were subsequently evaluated on both the raw test set (using the historical symptom class labels) as well as the manually annotated set. This was done to give a more nuanced look at the model performance. It makes sense to use the historical labels for evaluation as these were used to train the model and are more "realistic". That

4. Results and Evaluation

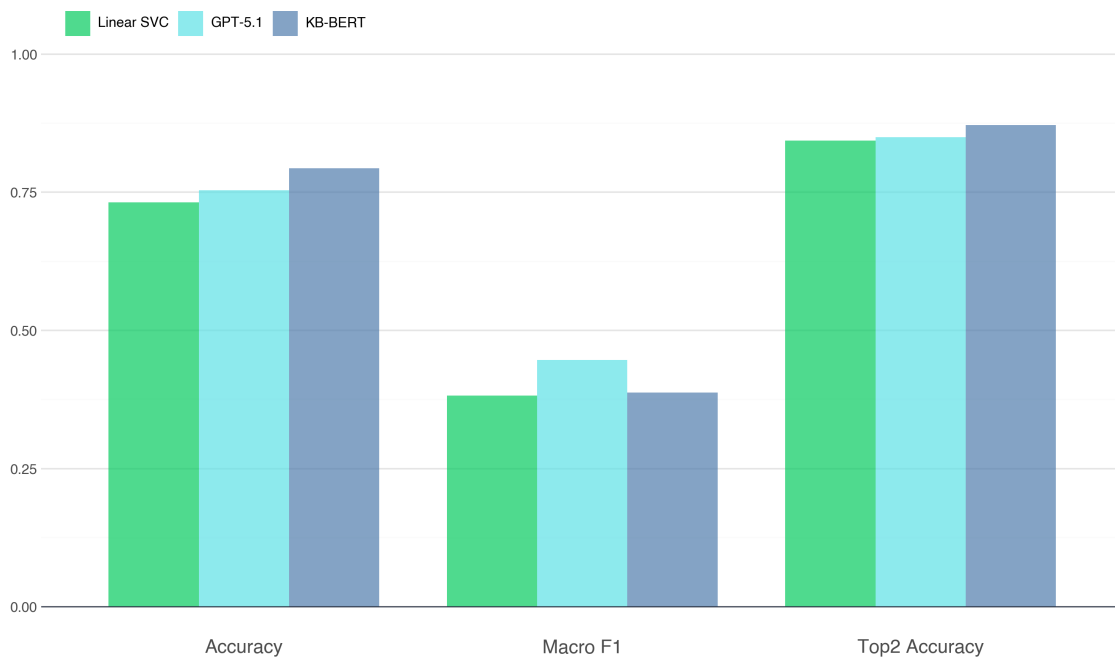


Figure 4.3: Accuracy, Macro F1 Score and Top-2 Accuracy of the three models, evaluated on the manually annotated dataset

being said, training on historical labels means that the model will learn the biases of the human classification. Therefore by evaluating the performance on the manually annotated dataset, the model is evaluated against what is hoped to be achieved.

The metrics used for evaluation were Accuracy, Recall, Precision, F1-score, and Top-2 Accuracy. In Section 2.2.4 it was shown that MWOs often belong to more than one class. To accommodate for this, Top-2 Accuracy was added to the evaluation, as this would show whether one of the model’s two first guesses were correct.

Even though the majority classes were downsampled, there was still a class imbalance in the test set which could lead to unfair accuracy estimates. Looking at the Macro F1 it appears as though the model struggles with the smaller classes. As summarised in Table 4.2, these metrics directly reflect the inherently different capabilities of a ML model trained on historical data and an LLM with the correct semantic class definitions. While the fine-tuned KB-BERT achieves the highest raw Accuracy (79.3%) by optimising for dominant classes, the LLM yields a significantly higher Macro Precision and Macro Recall. This shows that for highly imbalanced data, zero-shot or few-shot language models are better equipped to correctly classify small and sparse categories given the correct instructions, compared to traditional classifiers. It must still be noted that the performance of the LLM still leaves a lot to be desired.

To better understand limitations and challenges in classifying the dataset, a normalised confusion matrix of the LinearSVC predictions on the annotated test set

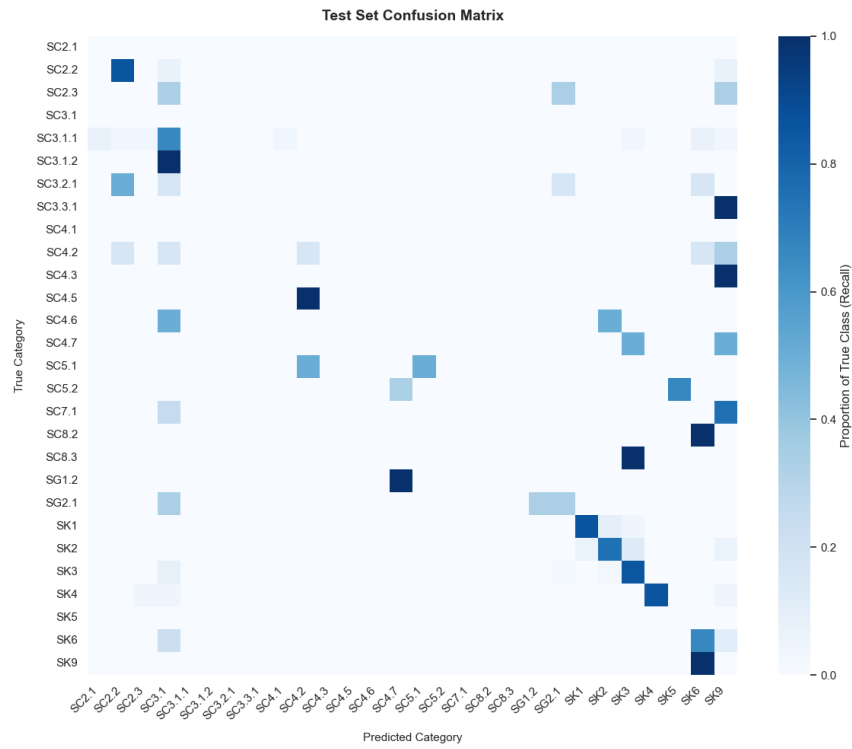


Figure 4.4: Normalised confusion matrix based on the symptom predictions of the final LinearSVC model on the annotated test set

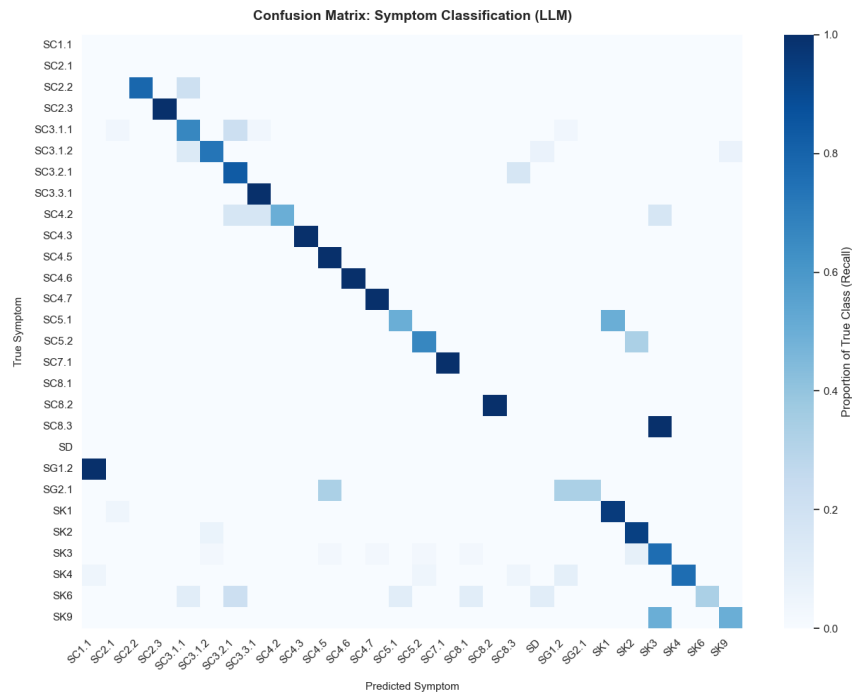


Figure 4.5: Normalised confusion matrix based on the symptom predictions of the LLM on the annotated test set

is presented in Fig 4.4. The matrix illustrates a clear performance constraint; the model primarily captures dominant environmental categories (SK-classes) and broad classifications (such as SC3.1), while consistently misclassifying specific technical SC-classes. This directly reflects the class overlaps identified during the exploratory phase (see Fig 2.4), confirming that user descriptions frequently conflate physical symptoms with assumed root causes. The class overlap is supported by the increase in Accuracy when considering Top-2 Accuracy, as this signifies the model guessed two classes, one of which matched the ground truth. Because the statistical classifiers were trained on raw historical labels containing human indexing bias, evaluating them against a strictly standardised test set demonstrates the limitations of relying on unstandardised training data.

Table 4.2 and Fig 4.5 demonstrate that the LLM achieved higher precision and recall than the SVC and BERT models. This may in part be explained by the LLMs understanding for semantics, but is more likely due to the fact that the LLM was given access to the same classification guidelines that were used during the manual annotation of the test set. The comparison is therefore somewhat unfair. The LLM is "trained" and tested on the true class labels, whereas the SVC and BERT were trained on historical classifications but tested on the true class labels.

Table 4.2: Performance Metrics of Symptom Model vs Dummy Classifier, evaluated on test set.

Model	Accuracy	Top-2 Accuracy	Macro Precision	Macro Recall	Macro F1-Score
Dummy Classifier	17.1%	-	0.7%	4.2%	1.2%
LinearSVC	72.1%	83.2%	43.3%	36.3%	37.5%
KB-BERT	79.3%	87.2%	39.0%	38.7%	38.7%
LLM	75.4%	84.9%	49.0%	46.3%	44.7%

4.1.3 The Priority Model

The performance of the Priority Model is presented in Fig 4.6. A manually annotated dataset for this model was not created, as the development was abandoned and the process of manually annotating is time-consuming. The evaluation is therefore based on the historical labels, which are not necessarily accurate.

In Fig 4.6 it appears as though all classes except "High" were classified with fairly high recall. This may appear to be a better result than the classification of the symptom class, but it is important to note that the evaluations are not directly comparable because of the huge difference in the number of target classes.

To further analyse the class with lowest recall ("High"), the feature importance was studied. It was found that the words most associated by the model with the "High"

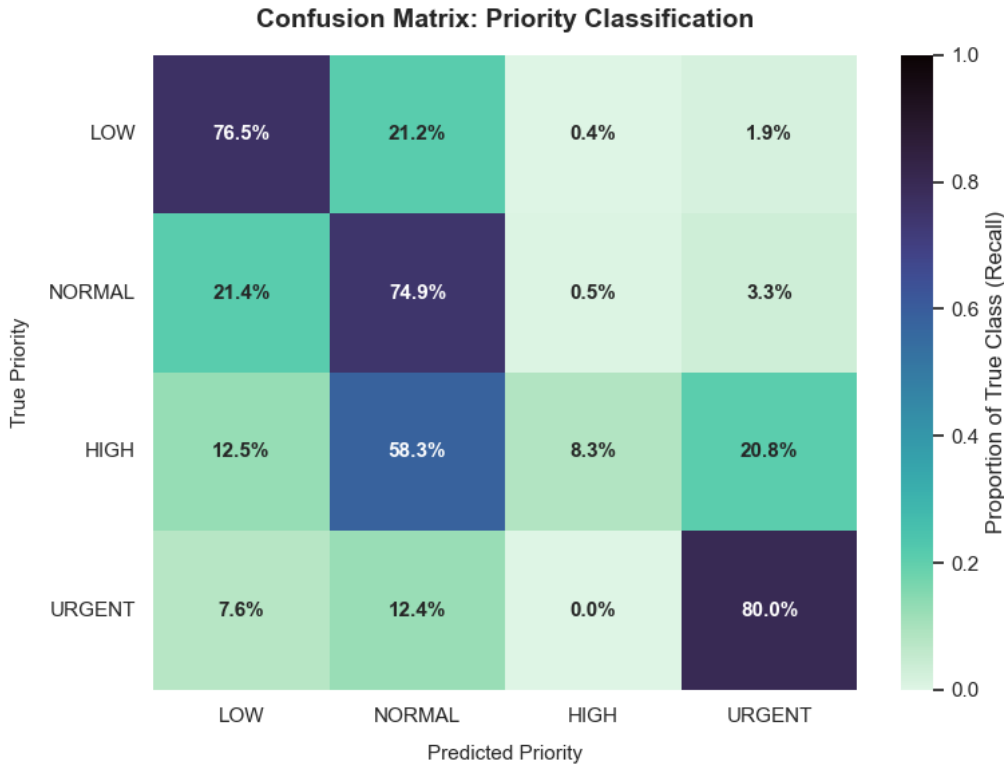


Figure 4.6: Normalised confusion matrix of the predictions of the SVC on the test set

priority class were "called", "call", "urgent", "extreme", "break-in", "medical centre". This indicates that the model was able to understand some of the key features of the class, but most likely because of the class imbalance or due to mislabeling was unable to fully capture it. This result demonstrates that priority logging in historical MWO data does not possess the structural readiness required for text-driven machine learning.

Table 4.3: Performance Metrics of Priority Model vs Dummy Classifier, evaluated on test set.

Evaluated Architecture	Accuracy / Micro F1	Macro Precision	Macro Recall	Macro F1-Score
Dummy Classifier (Most Freq.)	97.0%	24.3%	25.0%	24.6%
LinearSVC	74.8%	30.5%	59.9%	31.3%

4.2 Task Performance

A more qualitative performance analysis was also carried out. This focuses on the value created for Akademiska Hus, based on whether the solutions solve the problems defined in Phase 1 (see Section 2.3.2), and how well they automate the tasks.

4.2.1 Problem Resolution

Among the problems identified in Phase 1 (Section 2.3), the two artefacts mainly target the daily administrative load on technicians, the poor overview of MWOs in the CMMS, and the issue of misrouted tickets. Both artefacts successfully lower the administrative load by automating the classification and filtering processes.

While *The Structurer* aims to solve the problem of poor overview within the CMMS, this cannot be fully verified until the artefact is integrated. As of now, it is unclear whether the CMMS supports additional information display or changes in interface design.

The Demarcation Agent alleviates the issue of misrouted tickets by flagging MWOs that are suspected to be misrouted. However, a major part of this bottleneck remains in the current pipeline, as the artefact only outputs a decision recommendation. The physical task of changing fields and rerouting the ticket within the CMMS must still be performed manually by a technician. Automating the actual routing fell outside the scope of this project, as it would require a more advanced agent capable of interacting with external forms and APIs. While a future HITL system could let the agent execute the transfer after a technician's approval, the current prototype is strictly limited to an isolated decision-support tool.

Beyond addressing these inefficiencies, the artefacts also tackle the systematic issue of low data readiness of the MWOs identified during the EDA in Phase 1 (Section 2.2). By standardising the classification of MWOs, the data can be used more reliably for statistical analysis, decision support, and future digitalisation. Instead of viewing the models merely as tools for temporary time-savings, they function as infrastructure to improve the organisation's overall data quality.

4.2.2 Estimated Impact on FM KPIs

The tasks that the artefacts were designed to automate mainly targeted the KPI of *Operational Efficiency*. While a real-world trial of the artefacts, to measure *Operational Efficiency* in the form of time saved and integration with current workflow, fell outside the scope of the project, a theoretical estimation was made to assess the impact of the artefacts.

Akademiska Hus processes approximately 30,000 maintenance requests annually [3]. To evaluate the total operational impact, *The Structurer* and *The Demarcation Agent* are assessed as a combined pipeline rather than isolated tools.

Based on the stakeholder interviews conducted in Phase 1, a conservative estimate of 3 minutes per ticket is assumed for the entire manual triage process, which includes reading the text, cross-referencing BDAs, and entering classifications into the CMMS. This equates to an annual administrative burden of 1,500 hours. It is important to note that this 3-minute baseline represents a roughly estimated average. In

practice, the time required varies significantly based on the experience of the technician and the character of the MWO. While common issues may be processed rapidly by experienced staff, ambiguous requests (such as complex indoor climate issues) often require technicians to conduct inspections to definitively establish maintenance responsibility.

Using the developed artefacts, the human task is reduced to a validation step estimated at 30 seconds per ticket. This shifts the annual burden to 250 hours, yielding a theoretical reduction of 1,250 administrative hours annually.

As mentioned, this is only a theoretical estimate and cannot be verified, but it illustrates approximate scaling between the current workflow and the automated workflow, which in turn gives insight into the long-term scalability of the different approaches.

4.3 System Agency and Architectural Autonomy

Because the initial prototypes in Phase 1 were conceptually framed as "agents," evaluating the final implementations against a formal agent framework is useful to clarify their actual technical boundaries. The artefacts were not developed to achieve a specific theoretical architecture, but rather to evaluate the most viable way to resolve the administrative tasks and achieve implementation readiness within the existing system environment. Mapping the final pipelines against the structural modules defined by Wang et al. [39] (Profile, Memory, Planning, and Action) defines what was built.

Table 4.4: Architectural Classification Matrix on Agent Modules.

Artifact Configuration	Profile	Memory	Planning	Action	Architectural Classification
<i>The Structurer</i> (LinearSVC)	N/A	None	None	None	Statistical Classifier
<i>The Structurer</i> (Zero-Shot LLM)	Implemented	Implemented	None	None	LLM-Powered Pipeline
<i>The Demarcation Agent</i> (RAG)	Implemented	Implemented	Evaluated	None	Agentic Decision-Support System
<i>The Demarcation Agent</i> (Injection)	Implemented	Implemented	Implemented	None	Agentic Decision-Support System

This classification shows that fully autonomous agents may not be currently viable for production deployment in this context. While the LLM-powered pipelines effectively handle role definitions (Profile) and context retrieval (Memory), the "Action" module (defined as autonomous system modifications or direct database mutations) was intentionally omitted across all configurations. The development process revealed that because raw occupant text is consistently under-specified, granting autonomous capabilities to non-deterministic models introduces operational and contractual risks. Consequently, the artefacts are optimised as non-autonomous, Human-on-the-Loop (HOTL) decision-support systems rather than autonomous agents.

5

Discussion

This chapter discusses the findings presented in Chapter 4 within the theoretical frameworks established in the Literature Review (Section 2.1). Another aim is to critically reflect upon the methodological limitations of the DSR approach and other methodologies used in this thesis.

5.1 Characterising MWO Data and the Ontology Gap

One of the main focuses of this thesis was to study the challenges of working with MWOs as a data source. While MWOs offer a universal entry point across buildings regardless of their age, the results reveal a significant "ontology gap" between building occupants and FM providers. This gap represents an asymmetry in detail of descriptions. From the user's perspective, the complaint is highly specific to their experience. However, from the perspective of the CMMS and BDA rules, the text is fundamentally under-specified, as it omits the technical metadata and asset boundaries required to assign responsibility.

This ambiguity complicates the evaluation of all three developed models. As observed during the development phase, establishing a reliable "ground truth" within the historical datasets proved difficult. Because maintenance requests frequently describe overlapping symptoms that fit multiple categories, human technicians historically categorised these tickets with high subjectivity and inconsistency [37].

Through this study it was found that the oversimplification actually happens inside the CMMS taxonomy itself. As shown in the class distribution (Fig 2.2), a large majority of historical MWOs belong to very broad categories like "Interiors" and "Furniture". While dumping everything into these broad categories works fine for just sending out a technician, it creates a massive information gap for data analysis. By grouping completely different assets together, Akademiska Hus loses the ability to track how often specific appliances or components actually break down over time. Using automated text parsing would allow the organisation to keep and use the rich context from the user's text, improving data quality without forcing technicians to spend more time manually logging details.

Finally, the performance discrepancy between the symptom class and priority models highlights a fundamental limitation regarding the type of information that can be

extracted from the text. Whilst user descriptions clearly define a symptom, the operational priority is not as easily extracted. A user may express urgency through their choice of words, but the priority class within the CMMS is not necessarily tied to how urgent a user believes the issue to be. That being said, the fact that the model is able to pick up on words connected to urgency (as shown in Section 4.1.3) indicates that the urgency can sometimes be extracted.

5.2 Architectural Integration and Task Automation

To evaluate how different AI architectures can be applied to automation, the prototypes were developed and tested using different pipeline setups. A key focus was investigating whether complex agentic features add practical value compared to simpler direct approaches.

The development and evaluation phase demonstrated that the two core agentic features evaluated, CoT (Chain-of-Thought) prompting and RAG (Retrieval-Augmented Generation), showed diminishing returns. As presented in Table 4.1, CoT maximised the latency of the processing (for example, increasing average time per MWO from 2.37 to 4.53 seconds in the Direct Injection pipeline) while achieving sub-optimal accuracy relative to the other prompt strategies evaluated.

Furthermore, whilst the RAG agent architecture performed well quantitatively and lowered token consumption, it introduced retrieval errors. The RAG pipeline more commonly misclassified MWOs as Akademiska Hus' responsibility. This can be explained by the fact that all prompts contained a safety clause stating that if responsibility was unclear given the context, the model should rely on industry standards. If the RAG pipeline retrieved the wrong part of the BDA, it lacked the correct context and made an inaccurate generalisation. This shows that for this type of workflow, a simpler Direct Injection pipeline can handle the task equally well without the added architectural complexity and retrieval risks of a full RAG agent setup.

Generally speaking, the study shows that performance seems to be more strongly tied to prompt design rather than model architecture in this case. This is supported by the experience during development (see Section 3.1), model performance appeared to be very sensitive towards small changes in prompt formulation. With further refinement of the prompts it may therefore be possible to achieve better results from *The Demarcation Agent*.

Beyond prompt sensitivities, looking at the wider workflow automation reveals a major benefit regarding data consistency. In current manual operations, human technicians who are busy or tired naturally introduce inconsistencies, tagging identical technical issues differently based on personal habits. This is made worse by the overlapping classes and unclear instructions for classification. Automating the ingestion pipeline changes this by enforcing a set logic. Even if the models make oc-

casional mistakes, the mistakes are systematic and the classification becomes more objective. This creates a more structured, reliable database that can actually be used for tracking long-term trends or setting up future data-driven workflows.

5.3 Comparative Performance of Machine Learning and Language Models

The performance metrics presented in Table 4.2 provide a direct comparison between classical machine learning techniques and large language models when dealing with messy historical maintenance data.

During the development phase, building the AI functions proved relatively straightforward, whereas cleaning and structuring the historical data required many iterations. This underscores the AI readiness framework by Jöhnk et al. [23], which claims that an organisation's AI capability relies heavily on its data maturity. Because the training data for the statistical models was noisy and frequently mislabeled, the LinearSVC and BERT models struggled with the smaller classes, resulting in a lower Macro F1-score.

In comparison, the LLM achieved a higher Macro F1-score (44.7%) and higher Macro Precision (49.0%). However, when comparing these metrics, it is important to note that the evaluation is somewhat unfair to the machine learning models. The SVC and BERT models were trained entirely on historical CMMS data, meaning they had to learn from years of human labeling errors and overlapping categories (like the confusion between symptoms and systems). They were then tested against a perfectly clean, manually annotated set. The LLM on the other hand, skipped the messy history completely because the clean classification guidelines were injected directly into its prompt. Therefore, the LLM's higher performance is less about mathematically fixing a class imbalance, and more about its ability to follow strict rules. This makes it useful when clean training data is missing.

5.4 Automation, Operational Readiness, and Human-in-the-Loop Design

The evaluation of task performance demonstrates that substantial administrative efficiency can be achieved without requiring full system autonomy. Instead of aiming for an entirely automated pipeline, the performance and KPI estimations validate that a Human-on-the-Loop (HOTL) architecture is both practically and economically viable.

The Demarcation Agent encountered clear limitations due to the ontology gap and the lack of detail in user texts. As shown in Section 4.1.1, user descriptions often lacked the specific details required by the BDA to definitively define a responsible party. For instance, the model struggled to allocate responsibility because the text

failed to clarify whether a broken door pertained to the handle or the locking mechanism, or whether a broken light source was a fixed ceiling installation or a tenant's desk lamp. Consequently, a human technician is required not just for oversight, but to provide the tacit, building-specific context that the unstructured text does not contain.

For this reason, the pipelines were designed to autonomously generate recommendations for symptom classes, priorities, or responsibility, but the final execution remains manual. This choice directly reflects the core philosophy of Human-Centered AI, illustrating that the adoption gap in FM may be tied to the organisational requirement for trust and risk mitigation. The practical value of this setup is illustrated by the KPI estimation (Section 4.2.2).

It is important to acknowledge that a major bottleneck still remains in this prototype, as technicians must still manually execute the actual rerouting and field changes within the CMMS once a misrouted ticket is flagged. Automating the actual execution fell outside the scope of this project, as it would require a significantly more advanced integration capable of database mutations and interacting with APIs. However, these results prove that a full autonomous agent is not required to create value. By keeping the current prototype limited to an isolated decision-support tool, the organisation can still achieve significant time-savings while maintaining complete human oversight. In a future implementation, this could be expanded to a HITL architecture where the agent executes the data transfer automatically, requiring only a single confirmation from the technician.

5.5 Methodological Limitations and Ethical Considerations

During the course of the development of the pipelines, the necessary steps were taken to avoid data leakage and evaluate performance fairly. Despite this the evaluation is subject to certain limitations in the prompt engineering process. Specifically during development of *The Demarcation Agent*, the main prompt instructions were finalised after the manual annotation of the test dataset. As the manual annotation was time consuming it could not be redone on a new dataset. Therefore, the specific phrasing within the test set may have subconsciously influenced the prompt design. This data leakage bears the risk of overfitting the agent's instructions to the test set, potentially yielding overly optimistic performance metrics (see Fig 4.1).

Since this project followed the DSR approach, the workflow was largely action-driven. The research began exploratorily with data analysis and interviews, and the literature study followed as specific technical and organisational needs were identified. This approach directly impacted how the study was conducted. Looking back, if a comprehensive literature study had been completed first, a different path might have been chosen. Having that theoretical foundation upfront could have helped narrow the scope earlier, allowing the project to focus entirely on fully completing

and integrating *The Demarcation Agent* instead of splitting focus across multiple prototypes.

However, at the start of the project, it was difficult to predict exactly which specific topics would become relevant. Because the work is interdisciplinary the problem description started out broad to let the process itself shape the formulation of the problem. Speaking with domain experts gradually clarified which subjects required deeper study. While having expert knowledge in all three fields from the start would have enabled more targeted investigations during the initial discovery phase, working iteratively allowed the project to adapt to the practical data challenges as they appeared in the dataset.

From an ethical perspective, integrating LLMs into enterprise workflows requires strict data governance. Although MWOs generally lack sensitive personal information, they frequently contain names, phone numbers, and location data. To comply with GDPR data minimisation principles and organisational security policies, all models utilised in this study were accessed via secure enterprise agreements that prohibit the use of data for model training. However, for future production deployment, a local preprocessing pipeline designed to scrub Personally Identifiable Information (PII) before LLM ingestion may be necessary.

6

Conclusion

This thesis investigated the implementation of language models and machine learning to automate the processing of Maintenance Work Orders (MWOs) within facility management. Through a Design Science Research (DSR) approach, the project highlighted the challenges of real-world AI deployment, primarily regarding data readiness and the balance between automated efficiency and operational accountability. Furthermore, the topic of MWOs as a data source was explored.

The study demonstrates that full autonomy is currently impractical, not only due to the non-deterministic nature of LLMs, but because MWO data is often lacking detail. The observed "ontology gap" is a mismatch in descriptive granularity. While occupants describe their environmental symptoms with high specificity, they omit the technical detail required by boundary demarcation rules and CMMS schema. Because the raw text often doesn't contain enough information, full autonomy introduces financial and contractual risks. However, by limiting the agency of the solutions and using a Human-on-the-Loop (HOTL) approach, deploying reliable, language-model-powered pipelines is feasible. Rather than replacing human oversight, these artefacts demonstrate a way of automating the processing, classification, and filtering of MWO data.

In terms of operational value, the artefacts demonstrate the ability to reduce administrative bottlenecks. By shifting the technicians' burden from manual triage to validation, resources can be reallocated to physical maintenance, thereby elevating overall service delivery. The primary limitation of these solutions is their dependency on the quality of historical data. The contrast between symptom-heavy user descriptions and broad classification categories like "Interiors" or "Furniture" demonstrates that CMMS taxonomies fail to capture more granular details of the MWO data.

Finally, through the developed solutions a data-standardisation pipeline was established. By converting messy natural language into consistent, structured entries, these artefacts represent a move towards MWO data readiness. This creates a reliable foundation enabling future automated analytics within the industry.

6.1 Contribution

This thesis provides contributions to both academic research and industrial practice within facility management:

- **Academic Contribution:** The study contributes to cognitive automation literature by comparing generative zero-shot language models directly against trained statistical classifiers using noisy, imbalanced enterprise data. It provides an empirical example of how rule-enforcement via prompting can insulate a system from historical human labeling bias, and demonstrates how agency can be safely balanced with risk through HOTL design.
- **Industrial Contribution:** For Akademiska Hus and the broader FM industry, this work demonstrates a scalable methodology to extract value from unstructured data without requiring homogeneous IoT infrastructure. It also provides an example of how AI can be implemented with minimum friction to automate parts of the maintenance process.

6.2 Future work

The numerous prototype concepts filtered out during Phase 1 and the limitation of the developed artefacts to local PoCs clearly demonstrate several avenues in which this research could be expanded. The following tangible suggestions could be explored in future work:

- **CMMS Integration:** Future work should explore the technical integration of the artefacts directly into the CMMS API.
- **The Demarcation Agent:** As touched on in Section 4.2.1, a compelling expansion of *The Demarcation Agent* would be to allow it to automatically forward misrouted tickets to the correct entity.
- **Empirical Validation:** Finally, a long-term empirical study should be conducted within a live operational environment to validate the theoretical saving of 1,250 administrative hours and measure the direct impact on user satisfaction and FM efficiency.

Bibliography

- Abisuga, A. O., Wang, C. C., & Sunindijo, R. Y. (2020). Facility managers' responses to user post-occupancy feedback: A conceptual framework. *Facilities*, 38(7-8), 481–499. <https://doi.org/10.1108/F-10-2018-0119>
- Ahmed, V., Tezel, A., Aziz, Z., & Sibley, M. (2017). The future of big data in facilities management: Opportunities and challenges. *Facilities*, 35, 725–745. <https://doi.org/10.1108/F-06-2016-0064>
- Akademiska Hus. (n.d.). About us. <https://www.akademiskahus.se/en/About-us/>
- Alhaj, M. B., Liu, H., Abudayyeh, O., & Sulaiman, M. (2022). Development of a mobile application for occupant-centric facility maintenance management. *2022 IEEE World AI IoT Congress (AIIoT)*, 323–329. <https://doi.org/10.1109/AIIoT54504.2022.9817248>
- Amaratunga, D., & Baldry, D. (2000). Assessment of facilities management performance in higher education properties. *Facilities*, 18(7/8), 293–301. <https://doi.org/10.1108/02632770010340681>
- Bathae, Y. (2018). The artificial intelligence black box and the failure of intent and causation. *Harvard Journal of Law & Technology*, 31(2), 889–938.
- Bouabdallaoui, Y., Lafhaj, Z., Yim, P., Ducoulombier, L., & Bennadji, B. (2020). Natural language processing model for managing maintenance requests in buildings. *Buildings*, 10, 160. <https://doi.org/10.3390/buildings10090160>
- Cleanlab. (2026). Retrieved May 26, 2026, from <https://github.com/cleanlab/cleanlab>
- Conte, A., Bolland, C., Phan, L., Brundage, M., & Sexton, T. (2021). The impact of data quality on maintenance work order analysis: A case study in HVAC work durations. *Annual Conference of the PHM Society*, 13(1).
- Danilák, M. (2021). Langdetect. Retrieved May 26, 2026, from <https://github.com/Mimino666/langdetect>
- Davenport, T. H., & Ronanki, R. (2018). Artificial intelligence for the real world. *Harvard Business Review*, 96(1), 44–54.
- Design Council. (2007). Eleven Lessons: Managing Design in Eleven Global Brands. https://www.designcouncil.org.uk/fileadmin/uploads/dc/Documents/ElevenLessons_Design_Council%2520%25282%2529.pdf
- D'Orazio, M., Bernardini, G., & Di Giuseppe, E. (2023). Automated priority assignment of building maintenance tasks using natural language processing and machine learning. *Journal of Architectural Engineering*, 29(3), 04023027. <https://doi.org/10.1061/JAEIED.AEENG-1516>

- Egger, R., & Yu, J. (2022). A topic modeling comparison between LDA, NMF, Top2Vec, and BERTopic to demystify Twitter posts. *Frontiers in Sociology*, 7. <https://doi.org/10.3389/fsoc.2022.886498>
- Ensafi, M., Thabet, W., & Gao, X. (2024). Work order prioritization using neural networks to improve building operation. *Journal of Information Technology in Construction*, 29, 324–346. <https://doi.org/10.36680/j.itcon.2024.016>
- Franklin, S., & Graesser, A. (1997). Is It an agent, or just a program?: A taxonomy for autonomous agents. In J. G. Carbonell, J. Siekmann, G. Goos, J. Hartmanis, J. Van Leeuwen, J. P. Müller, M. J. Wooldridge, & N. R. Jennings (Eds.), *Intelligent Agents III Agent Theories, Architectures, and Languages* (pp. 21–35, Vol. 1193). Springer Berlin Heidelberg. <https://doi.org/10.1007/BFb0013570>
- Gunay, H. B., Shen, W., & Yang, C. (2019). Text-mining building maintenance work orders for component fault frequency. *Building Research & Information*, 47(5), 518–533. <https://doi.org/10.1080/09613218.2018.1459004>
- Honnibal, M., Montani, I., Van Landeghem, S., & Boyd, A. (2020). spaCy: Industrial-strength natural language processing in Python. <https://doi.org/10.5281/zenodo.1212303>
- Hopland, A. O., & Kvamsdal, S. (2018). On the ranking of critical success factors: The role of cost efficiency and score uncertainty for public facilities management. *Journal of Facilities Management*, 16(1), 26–37. <https://doi.org/10.1108/JFM-04-2017-0012>
- Hosseini, S., & Seilani, H. (2025). The role of agentic AI in shaping a smart future: A systematic review. *Array*, 26, 100399. <https://doi.org/10.1016/j.array.2025.100399>
- IBM. (2025, March). What is agentic architecture? <https://www.ibm.com/think/topics/agentic-architecture>
- International Facility Management Association. (n.d.). What is Facility Management? Benefits and Use Cases. <https://www.ifma.org/about/what-is-fm/>
- Jöhnk, J., Weißert, M., & Wyrski, K. (2021). Ready or not, AI comes— an interview study of organizational AI readiness factors. *Business & Information Systems Engineering*, 63(1), 5–20. <https://doi.org/10.1007/s12599-020-00676-7>
- Kungliga biblioteket. (2020, July). Bert-base-swedish-cased [Model]. <https://huggingface.co/KB/bert-base-swedish-cased>
- Lavy, S., Garcia, J. A., & Dixit, M. K. (2010). Establishment of KPIs for facility performance measurement: Review of literature. *Facilities*, 28(9/10), 440–464. <https://doi.org/10.1108/02632771011057189>
- Li, Z., Li, C., Zhang, M., Mei, Q., & Bendersky, M. (2024, October). Retrieval augmented generation or long-context LLMs? A comprehensive study and hybrid approach. <https://doi.org/10.48550/arXiv.2407.16833>
- Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023, November). Lost in the middle: How language models use long contexts. <https://doi.org/10.48550/arXiv.2307.03172>
- Marocco, M., & Garofolo, I. (2021). Integrating disruptive technologies with facilities management: A literature review and future research directions. *Automation in Construction*, 131, 103917. <https://doi.org/10.1016/j.autcon.2021.103917>

- Mo, Y., Zhao, D., Du, J., Syal, M., Aziz, A., & Li, H. (2020). Automated staff assignment for building maintenance using natural language processing. *Automation in Construction*, 113, 103150. <https://doi.org/10.1016/j.autcon.2020.103150>
- Naji, K. K., Gunduz, M., & Al-Qahtani, A. (2024). Unveiling digital transformation: Analyzing building facility management's preparedness for transformation using structural equation modeling. *Buildings*, 14(9), 2794. <https://doi.org/10.3390/buildings14092794>
- Northcutt, C. G., Jiang, L., & Chuang, I. L. (2022, August). Confident learning: Estimating uncertainty in dataset labels. <https://doi.org/10.48550/arXiv.1911.00068>
- Peppers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-1222240302>
- Quinello, R., & Kramer Costa, B. (2025). Critical success factors for the adoption of artificial intelligence in facilities management: Second-order systematic review. *Journal of Technology Management & Innovation*, 20(2), 103–118. <https://research.ebsco.com/linkprocessor/plink?id=a84850e4-8fbf-3fc3-8699-d47f92721b61>
- Qurat-Ul-Ain, Shaukat, A., & Saif, U. (2022). NLP based model for classification of complaints: Autonomous and intelligent system. *2022 2nd International Conference on Digital Futures and Transformative Technologies (ICoDT2)*, 1–6. <https://doi.org/10.1109/ICoDT255437.2022.9787456>
- Scaife, A. D. (2024). Improve predictive maintenance through the application of artificial intelligence: A systematic review. *Results in Engineering*, 21, 101645. <https://doi.org/10.1016/j.rineng.2023.101645>
- Shneiderman, B. (2020). Human-centered artificial intelligence: Reliable, safe & trustworthy. *International Journal of Human–Computer Interaction*, 36(6), 495–504. <https://doi.org/10.1080/10447318.2020.1741118>
- Stenström, C., Aljumaili, M., & Parida, A. (2015). Natural language processing of maintenance records data. *ResearchGate*. https://www.researchgate.net/publication/275335337_Natural_language_processing_of_maintenance_records_data
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. <https://doi.org/10.48550/arXiv.1706.03762>
- Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., Zhao, W. X., Wei, Z., & Wen, J.-R. (2024). A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), 186345. <https://doi.org/10.1007/s11704-024-40231-1>
- Weng, L. (2023, June). LLM powered autonomous agents. <https://lilianweng.github.io/posts/2023-06-23-agent/>
- Xu, K., Chen, Z., Xiao, F., Zhang, J., Zhang, H., & Ma, T. (2024). Semantic model-based large-scale deployment of AI-driven building management applications.

- Automation in Construction*, 165, 105579. <https://doi.org/10.1016/j.autcon.2024.105579>
- Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., ... Wen, J.-R. (2026, March). A survey of large language models. <https://doi.org/10.48550/arXiv.2303.18223>

DEPARTMENT OF ARCHITECTURE AND CIVIL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY