LLM-Assisted Requirements Engineering in Agile MDD: Industry Insights and Validation

Tjerk Spijkman*†, Bente Molenkamp†, Steffen Beudeker*, Sietse Overbeek†, and Fabiano Dalpiaz†
*fizor. – Utrecht, the Netherlands
Email: {tjerk.spijkman, steffen.beudeker}@fizor.com
†Utrecht University – Utrecht, the Netherlands
Email: bente_molenkamp@hotmail.com, {s.j.overbeek, f.dalpiaz}@uu.nl

***Abstract*—Despite the growing interest in integrating Large-Language Models (LLMs) within software development, there is limited empirically-grounded guidance for teams to effectively apply this technology in industry. We explore the use of LLMs for generating requirements artifacts within a low-code consultancy organization that builds low-code development applications following a custom Agile model-driven development (Agile MDD) process. Through the analysis of multiple project cases within the company, we construct a method as an accurate representation of the employed Agile MDD approach. We then identify high-potential use cases for LLM adoption within the Agile MDD method. For these use cases, we engineer reusable LLM prompts that generate requirements artifacts. We validate the generated output for three of such use cases: generation of user stories, of acceptance criteria, and of data models. The team members of four projects were asked to express their opinion on the automatically generated artifacts. The results show high appreciation for the artifacts, which were found to be mainly relevant and similar to requirements that were included in the initial specification. The generation of the data model was rated less positively than the other use cases. Besides providing detailed insights on the inclusion of LLMs in the company's Agile MDD process, we share our results to provide guidance for other software teams seeking to leverage LLMs in Agile MDD.**

***Index Terms*—Agile MDD, LLM, Prompt Engineering, Requirements Engineering, Requirements Specification, User Stories, Data Modeling.**

I. Introduction

Agile development is currently the mainstream development method in software companies. According to Digital.AI's survey results, 71% of the respondents state they use Agile in their software development life-cycle (SDLC) [1].

Model-driven development (MDD), the use of models as primary artifacts for software development, is also a rising trend [2]. Through various platforms and toolkits, MDD aims to simplify and formalize the various activities and tasks of the SDLC [3]. For example, this can be done through standardizing development steps into reusable models and facilitating conversion into code. MDD is at the basis of low-code development [4]–[6], which aims at mitigating the lack of IT personnel by increasing intuitiveness and speed of development [7].

Agile model-driven development (Agile MDD) strives to leverage the adaptivity, flexibility and user-centricity of Agile with the abstraction level and living documentation of MDD [8]. The executable models of MDD can serve as a means of communication and contribute to shared understanding [9]. in Agile MDD, Agile artifacts such as user stories and scenarios guide the MDD development. Research on the effective combination of these two approaches is limited.

Similarly, there is growing reliance on Large Language Models (LLMs) in industry, with widespread adoption yet a notable absence of standardization [10]. Due to the novelty of the field, however, there is a significant gap in research on effective applications and processes to utilize AI.

In this paper, we investigate the use of LLMs in Agile MDD from an industrial perspective. The creation and maintenance of RE artifacts is a time consuming activity [11], and we surmise that LLMs may be able to support practitioners and improve efficiency. To investigate this, we conduct a study within a low-code consultancy company, wherin we aim at identifying the suitability of LLMs for generating RE artifacts in their custom Agile MDD process. We have one main research question that is refined into three sub-questions:

MRQ. To what extent can LLM-based RE artifacts be effectively generated within an Agile MDD process?

RQ1. In which use cases can we employ LLMs to generate RE artifacts?

RQ2. What is a re-usable set of prompt templates that can be applied in the identified use cases?

RQ3. What is the perceived effectiveness of the LLM-generated RE artifacts by project team members?

We first characterize the Agile MDD process in order to identify candidate artifacts and activities for LLM use (RQ1). Then, we design re-usable prompt templates for five use cases within the Agile MDD process (RQ2). Finally, we select three use cases that are most relevant for the creation and improvement of a requirements specification. With these use cases, we validate the perceived effectiveness of three kinds of generated outputs, which are user stories, acceptance criteria, and data models, according to the team members from four projects (RQ3). To enable a more homogeneous comparison across cases, all these projects are in the Warehouse Management Systems domain.

This industry-academia collaboration allows us to: (i) create prompts that are clearly positioned in a real-world development process, and (ii) validate the generated RE artifacts with project team members, who are experts with detailed knowledge of the domain and of the projects.

Through this investigation, we aim to provide validated insights from industry context on the effectiveness of LLMs in the generation of Requirements Engineering (RE) artifacts. In this research, we utilize ChatGPT-4o through the web interface because we focus on the use case of a practitioner interacting with the LLM chat interface. While this resembles how LLMs can be used by practitioners without integration in their IDEs, this comes with some limitations; using the chat, indeed, did not allow us to adjust parameters such as the LLM temperature.

While we perform this study in a specific industry context, we share the steps used to design the prompts and the results of the empirical validation. In particular, we provide an online appendix[1], including an example mock-project (part A), and the data gathered from our surveys (part B) for replicability and as inspiration for using LLMs in similar contexts. Additional parts are specified in later sections.

The remainder of the paper is organized as follows. Sect. II reviews the relevant background. Sect. III details the research method. We describe the Agile MDD context in Sect. IV, present the engineered prompts in Sect. V, and report on the validation in Sect. VI. After discussing threats to validity in Sect. VII, we present lesson learned and the conclusions in Sect. VIII. Finally, we outline future directions in Sect. IX.

II. Background

We review related literature on LLMs and their use in RE (Sect. II-A), prompt engineering (Sect. II-B), and agile MDD (Sect. II-C).

A. LLMs in Software and Requirements Engineering

Since the public release of ChatGPT-3, LLMs have seen a surge in public use. This has only increased with improvements to the OpenAI models (currently offering GPT-4o and GPT-4.5) and new contenders like Deepseek. LLMs have already been used in many software engineering research efforts. Focus areas include code generation [12], [13], testing [14], and code summarization [15]. In the industry, commercial systems like ChatGPT, GitHub's Copilot, Amazon CodeWhisperer, and Google Gemini have seen a rapid increase in functionality and adoption. A recent study with thirty tech professionals found that nearly half the respondents stated that LLMs can significantly accelerate software development [16].

LLMs are a hot topic in RE too [17]. Applications include tasks such as requirement elicitation [18], generating interview scripts [19] and dealing with ambiguitiy in requirements [20]. Voria *et al.* [21] combine the NLP and LLM functions in a pipeline to generate requirements specifications from transcriptions. Many of such articles present novel applications of LLMs in various use cases. However, while benchmarks exist for activities like creative writing, commonsense reasoning or dialogue ability, these do not focus on industry relevance [22], thereby creating a gap for specialized industry evaluations. Our study constitutes an industry-specific evaluation.

[1]The online appendix can be found at https://doi.org/10.5281/zenodo.15754874.

B. Prompt Engineering

Prompt engineering is emerging as a critical activity for the effective use of LLMs. Prompt engineering focuses on design and optimization of prompts to achieve desired outputs from LLMs [23]. These prompts are natural language instructions that are given to an LLM to enhance and refine interactions. Like design patterns in software engineering [24], prompt patterns are reusable designs to support LLM interaction [23]. These patterns include providing a persona to the LLM [25], specifying an output template [23], and using step-by-step instructions through the chain-of-thought pattern [26]. In this research, we applied patterns, as shown in the tables in Sect. V, to design the prompts used and evaluated in the industry study.

C. Agile Model-Driven Development

Model-driven development (MDD) is a software development method that applies models and modeling tools for abstraction of development [3]. Using these models in development helps to establish a shared domain understanding between stakeholders [27]. MDD techniques boost productivity, minimize manual implementation effort, increase system quality, and provide flexibility [28]. Research has suggested approaches to integrate agile practices in MDD. For instance, MERODE [9] integrates MDD in Agile processes by including software architecture in the process.

Industry has seen an increase in MDD application through the rise of low-code platforms [5], [6]. Mendix states that MDD is the foundation of low-code development [4]. These low-code application platforms (LCAPs) are defined by Gartner as "platforms for accelerated development and maintenance of applications, using model-driven tools for the entire application's technology stack, generative AI and prebuilt component catalogs" [2]. This includes platforms like Thinkwise, Mendix, Betty Blocks, and Power Apps. Gartner's report [2] states that, in 2024, 10% of software development organizations use LCAPs as their main development platform with a projected growth over the next years.

Combining generative AI and Agile MDD, Sadik *et al.* [29] suggest an approach to combine ChatGPT and PlantUML to convert UML diagrams into textual representations to be used as prompt. However, further research on this combination is extremely limited. Interestingly, from the industry side, LCAPs are introducing and pushing LLM functionality. For instance, Thinkwise supports AI code reviews, test automation and model generation [30]. Similarly Mendix has recently introduced Mendix AI Assistance with new functionality including page generation and user story generation [31].

III. Research Method & Industrial Case

We opt for context-driven software engineering research [32] and investigate a specific low-code development context. In particular, we apply the design cycle of Wieringa's design science [33]. We perform problem investigation by analyzing three MDD projects, leading to the construction of a Process Deliverable Diagram (PDD) [34]. PDDs are a modeling approach within method engineering that allows to precisely

characterize the activities and deliverables of a method [34]. We analyze the methods used in the three cases and generalize those into a single PDD that represents the specific Agile MDD process used at the case company.

A. Industry Context

Our research is conducted at fizor., a low-code consultancy company located in the Netherlands, with approximately 50 employees at the time of writing. fizor. uses low-code platforms provided by technology partners to build complex software systems within core processes for customers. These applications are developed using Agile MDD.

Customer applications are developed to customer specific requirements defined for the project, and managed based on an agreed scope and timeline. These projects start with an analysis phase, where a requirements specification and functional design are the main deliverables. This is followed by a sprint 0, where the initial data model is designed and the project management context is defined. Finally, a two-week sprint process includes the main activities of the software project.

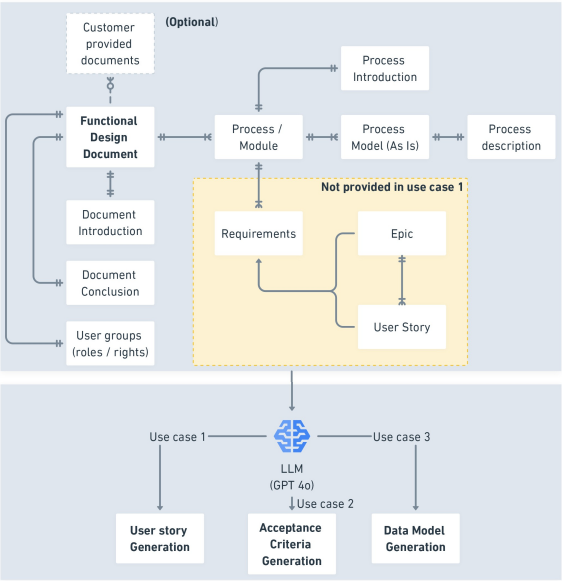

Fig. 1. Overview of the input and output RE artifacts for the validation (RQ3).

In this research, we use the analysis outputs as the main artifacts to feed the LLM prompts and generate RE artifacts that can contribute to the project specification. Fig. 1 visualizes the input artifacts that were used for the generation tasks in this research. We chose to focus on the functional design document as this was leading for the project scope and would be available for all projects in this industry context. Optionally, documents provided for the project were also included when they were available for additional context. The documents provided to the LLM were first reviewed for sensitive data based on the company guidelines on AI use. Although we cannot share the original documents due to confidentiality agreements, we provide a mock case document in the online appendix.

While the overall development context at fizor. is domain agnostic, this research focuses on four projects with a similar scope for validation of the outputs in a specific context. All these projects are in the Warehouse Management System (WMS) domain, and developed in the Thinkwise low-code platform (MDD). *ProjA* is a project focusing on replacing a legacy application for warehouse management that includes purchasing and sales processes, *ProjB* concerns an existing ERP application in the low-code platform that is extended with warehouse management functionality still being used through the legacy ERP system. *ProjC* is an extension of an ERP application to include WMS functionality. *ProjD* is a new WMS system, that replaces a legacy application and integrates with an existing ERP system for order data. For each project, we selected a single process as the focus for the artifacts, as we wanted to set a specific scope for LLM and participants.

B. Problem Investigation

In the first phase of Wieringa's design cycle, we have analyzed the Agile MDD process at the case company. We did so through semi-structured interviews with team members of three industry projects (Table I), of which *ProjD* is also used for the validation. The interviews were aimed at gathering insights into project activities, their sequence, and the project artifacts. Participants were also asked about their views on using generative AI and to share ideas for using LLMs in their day to day workflows. We created a process deliverable diagram that describes the Agile MDD method at the company (see Sect. IV). The resulting model provided insight in the artifacts and process used at fizor. In addition to its representational function, the model was used to identify five use cases for the use of LLM-generated artifacts.

TABLE I
PROJECTS STUDIED IN THE PROBLEM INVESTIGATION PHASE (RQ1).

Project	LCAP	Interview participants
ERP replacement project	Novulo	Lead developer
Project planning tool	Betty Blocks	Business analyst
WMS replacement project (*ProjD*)	Thinkwise	Business analyst, lead developer

C. Treatment Design

For treatment design, we aimed to employ LLMs for these five identified use cases. Using the available documents for these three projects, we iteratively engineered the prompts for the five use cases as described in Sect. V.

D. Treatment Validation

For the validation, we had to limit the effort required by the team members. Thus we decided to focus on the use cases for generation of (i) user stories, (ii) acceptance criteria, and (iii) data models. These prompts shared the same inputs with project context and fit in the focus of RE artifacts within the Agile MDD process.

The prompts were then applied on a set of four more homogeneous industry projects, as described in Sect. III-A. In order to replicate the most likely industry interaction utilizing LLMs through the web interface, we used a single LLM chat session to generate artifacts. This limits the output bias as we rely on the non-determinism of the LLM, instead of running the prompts repeatedly on the same inputs and selecting which results to use.

We used the prompts as they are, without specific project-based tweaks through follow-up prompting. We made this choice to validate the prompt templates as a re-usable tool that can be applied to other cases.

The generated outputs were then fed into a set of three surveys per case to validate the outputs. We designed one survey per each use case, for a total of twelve surveys. The surveys were distributed to industry professionals involved in the case projects. This meant a limited potential sample size, especially as there was a overlap in the project teams across the cases. To mitigate threats to validity, and limit the required effort from participants, we opted to involve each participant for a single case. The outputs were not *explicitly* compared to the project artifacts. This was done implicitly by relying on the participants' knowledge, since they were project members and, thus, familiar with the project artifacts.

IV. LLM in Agile MDD process for Generation of artifacts (RQ1)

For the problem investigation part of the research (RQ1), we created a holistic representation of the Agile MDD method used at the company. This was done in order to determine use cases for LLMs to support and enhance the Agile MDD process. To achieve this, we performed method analysis [34] based on three (at the time) ongoing projects, each relying on a different LCAP: Thinkwise, Betty Blocks and Novulo. With use cases, we refer to the combination of the two sides of the PDD, activities and their resulting deliverables.

For each of these cases, a Process Deliverable Diagram (PDD) was created. To create and validate the project-specific PDDs, interviews were conducted with the project's business analyst or lead developer. During these interviews, we also discussed how participants would envision using LLMs in their workflow. Findings from these interviews regarding the LLM requirements are provided in the online appendix (part C).

To combine the resulting PDDs into a company-specific one, we compared method fragments across the projects, for example, sprint preparation or project preparation. These fragments were assembled into a single PDD to represent the Agile MDD method used at fizor. This PDD forms the foundation for LLM use case identification. From the method analysis, we found the 'Functional Design Document' to be a central RE artifact, and identified a set of activities and

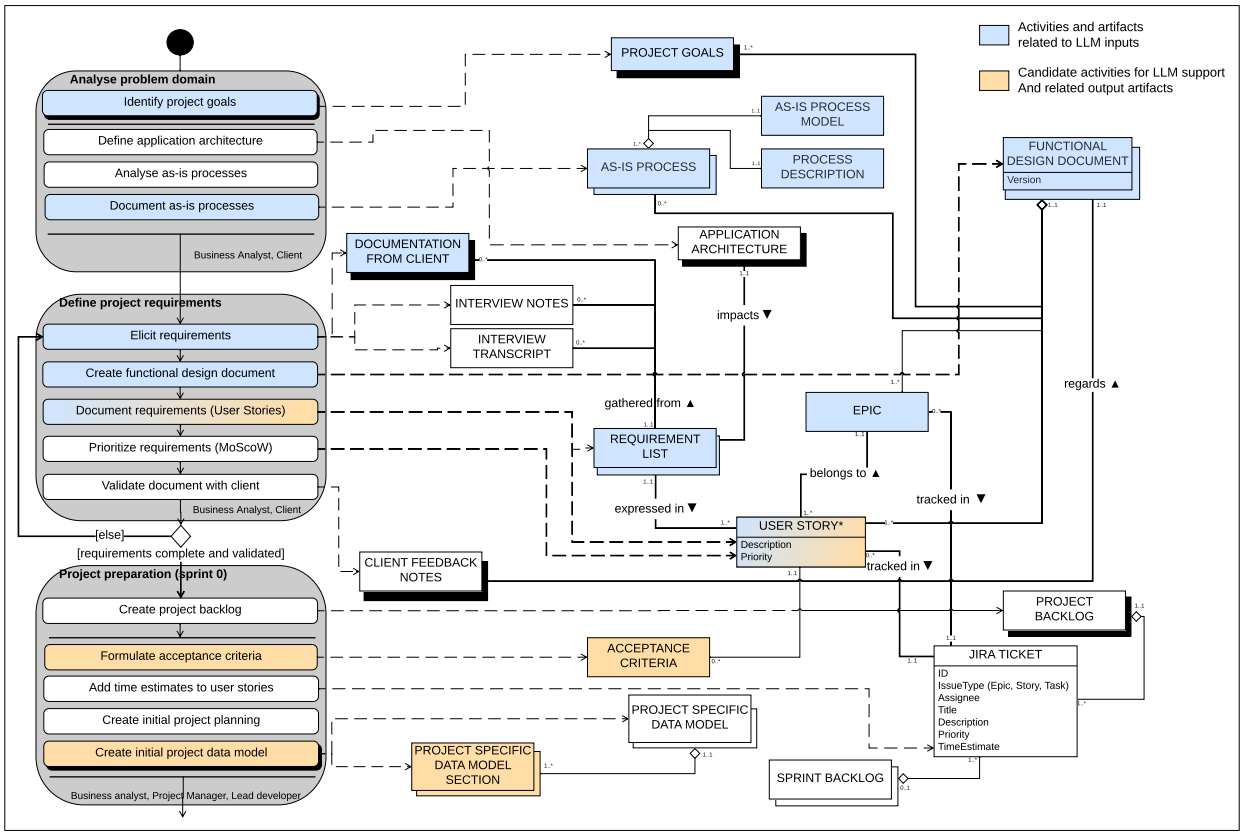

Fig. 2. PDD representing the first three activities and sub-activities of the Agile MDD method, with highlighted activities and artifacts.

artifacts (concepts in the PDD notation) around this document. Fig. 2 highlights the activities and artifacts related to the LLM inputs in blue, and the candidate activities and outputs in orange. While the figure only includes the first three steps, the full PDD is in the online appendix (part D).

From this central artifact, we opted to focus on the activities in the project startup (analysis of the problem domain, definition of project requirements and the project preparation). This falls in the window between the pre-requirements and post-requirements specification phases [35], specifically the refinement and extension of the requirements specification. Note that these activities can also be relevant for requirements identified later in the project. Based on the method analysis and interviews, we determined a list of potential use cases for utilizing LLMs.

From the identified use cases, we selected five for which we would create prompt templates as part of the treatment design phase of the design cycle. We limited our use cases to five due to time and scope constraints. The selection criteria were: (i) the generated output needs to be an artifact with a consistent structure content and explicit intended use; (ii) the use case task should allow the creation of a re-usable prompt template; and (iii) input documentation needs to be available that can be used as input for the generation of the artifact. This led to the following five use cases:

1) *User Story Generation.* corresponds to the activity 'Document Requirements (user stories)' as depicted in Fig. 2. This step involves the analyst formulating the requirements in the form of user stories (US), such as: "As a warehouse manager, I want to make inventory mutations so that I can correct inaccurate inventory data". The analyst translates as-is processes into requirements for the to-be processes [36]. These user stories are the primary artifact in Jira, the employed project management tool. This not only supports validating requirement completeness, but also improves efficiency by accelerating specification.

2) *Acceptance Criteria Generation.* During the project preparation phase, user stories are enriched with acceptance criteria (AC), which specify the detailed conditions to be met. For example, the user story in the previous paragraph can have the AC 'Stock can be both added and subtracted (+ and –)' for an inventory adjustment story. AC are added to USs and inform both estimation and development during sprints. Interviewees noted the activity is perceived as tedious and time-consuming, especially since customers rarely provide them. Automating this step could accelerate the specification process.

3) *Data Model Generation.* In LCAPs, MDD generally centers on the data model, which is initially designed by the project team and expanded during development. Automating this process with LLMs could accelerate development by generating a tailored model that, through MDD toolkits, enables rapid application prototyping.

4) *Process Model Generation.* Business analysts document current processes in BPMN to communicate with stakeholders and identify knowledge gaps. These models are included in the functional design as context for the application. Utilizing LLMs to generate these models could significantly reduce the time needed for the design document.

5) *Meeting Summarization.* During the project, requirements are typically gathered through interviews or discussions [37]. AI facilitates transcript availability [38], or interview notes can be available. LLMs can generate summaries from transcripts or notes, reducing the need for manual note-taking. This allows analysts to focus on the conversation and easily share insights with the broader team.

V. Prompt Engineering (RQ2)

To determine the prompts, we followed Marvin *et al.*'s structured prompt engineering [39]. Zero-shot prompting was used, and we adopted best practices and guidelines from literature [40], [41]. Additionally, prompt patterns [23] were incorporated when applicable. Through iterative testing and refinement, prompts were engineered based on the initial three projects mentioned in Sect. IV. We provide the prompts for the three use cases, while the other two prompts are in the appendix (part E). Overall lessons learned on the prompt engineering will be discussed in Sect. VIII.

A. User Story Generation

This use case aims to generate a comprehensive set of user stories based on the provided contextual information. The functional design document and, optionally, customer provided documents (e.g., legacy system documentation) are provided to the LLM. To improve the formatting, the prompts (Table II) provide a user story template. The prompts initially returned a relatively low number of user stories with high abstraction. To remedy this, we adapted them to request user stories for a specific section of the attached document.

TABLE II
Prompts for the generation of user stories.

Prompt 1: Provide context	**Patterns**
You are a business analyst for a software development project described in the [project documentation]. Read this [project documentation] carefully. Acknowledge that you have received and read the document carefully and wait for my next prompt. I will then provide you with [some additional information] based on which I will ask you to write a comprehensive set of user stories.	persona specification, input project context, task definition, confirmation request
Prompt 2: Provide input and task instructions	**Patterns**
Attached you will find [requirement input documentation]. Based on the information in the attached document, write a comprehensive set of user stories for [input document section]. Write the user stories in the following format, "As [user], I want [desired functionality] so that [reason/goal]". Make sure that all features of [input document section] are covered. Ensure that each user story provides a detailed description of what a user wants to be able to do.	task-specific input, constraints and guidelines, expected output format

B. Acceptance Criteria Generation

This use case aims at defining clear and testable acceptance criteria for a provided set of user stories. The prompts (Table III) also use the project documentation as additional context, which is the functional design in our case. Chain-of-thought prompting is applied here: we first provide the persona and request the LLM to analyze the context information. A second prompt then provides the user stories and the task to generate acceptance criteria. Optionally, focus on specific functional or non-functional criteria can be added. To prevent the addition of unnecessary formatting such as headers that contribute to the token limit, formatting instructions are included.

TABLE III
PROMPTS FOR THE GENERATION OF ACCEPTANCE CRITERIA.

Prompt 1: Provide context	**Patterns**
You are a business analyst for the project described in the attached [project documentation]. Read the document carefully. Confirm that you have read the document and wait for my next prompt. I will then ask you to help write acceptance criteria for a set of user stories from the project.	persona specification, input project context, confirmation request, task definition.
Prompt 2: Provide task instruction	**Patterns**
I want you to generate acceptance criteria for the following user stories. Make sure to align the criteria with the information outlined in the [project documentation]. Each criterion should be clear and concise. For each user story, list the user story first and then directly follow it with the corresponding acceptance criteria in bullet-point format. Here is the list of user stories: "[user story set]"	task definition, task-specific input, constraints and guidelines, expected output format
Optional: Specify project focus e.g.: Prioritize criteria for [performance OR functionality, OR usabilty]	extra constraints and guidelines
Optional: Specify AC template e.g.: Use the 'Given, when, then' template for each criterion.	extra constraints and guidelines

C. Data Model Generation

For the generation of data models, it is important for the LLM to fully understand the context and process to be supported by the data model. Therefore, the first prompt (Table IV) informs the LLM of the context, role and task expectations. Additionally, since ChatGPT-4o did not perform well in generating model visualizations, we stated that the descriptions should be formatted for use in the desired visualization tool [23]. The prompts also contain extra instructions regarding the relationships, to improve overall correctness and completeness of the generated model. Additionally, the prompts include instructions for link or bridge tables as they would otherwise not be included. Output formatting instructions are also provided. The third prompt is used to translate the output to a visualization tool. For this research we used PlantUML.

TABLE IV
PROMPTS FOR GENERATING A DATA MODEL FOR A PROCESS.

Prompt 1: provide context	**Patterns**
You are a low-code developer for the project described in the attached [project documentation]. The project will be built on [MDD platform]. I will ask you to create a relational data model for a certain part of the project. While creating the data model, I want you to adhere to the [MDD platform] data modeling guidelines specified in [modeling guidelines documentation]. Carefully read the attached project description and the guidelines. Confirm that you have carefully read the provided information and then wait for my next prompt.	persona specification, input project context, task definition, provide modeling guidelines, confirmation request
Prompt 2: provide task instructions	**Patterns**
I want you to create a data model, more specifically a [specific modeling approach], based on the [data model input documentation]. Make sure to follow the aforementioned modeling guidelines when describing the model. Take extra care when determining the relations (one-to-one, many-to-one, and link tables for many-to-many relationships) between the entities, to make sure they adhere to the provided input. Make sure to not forget to use link tables to incorporate many-to-many relationships. For each entity, provide a concise name, its attributes, and a description of the entity. For each relationship, provide between which entities it lies, whether it is a one-to-many or one-to-one relationship and your reasoning for including it.	task definition, task-specific input, constraints and guidelines, expected output format
Optional: Provide instructions to refine the generated model description if necessary.	
Prompt 3: format for visualization tool	**Patterns**
Format the generated data model description output so that I can provide it to [visualization tool] to visualize it.	visualization generator

D. Meeting Summarization

This use case aims to generate accurate and detailed meeting summaries. Optionally, the LLM can be prompted to focus specifically on key decisions, meeting action points, requirements, or other explicitly defined topics.

We added specific instructions to improve the level of detail, focus on certain items, and to boost comprehensiveness. Additionally, formatting instructions were added to write paragraphs instead of bullet points and adding headers.

For this use case, it was also required to split the transcripts into sections in order for the LLM to handle the entire meeting duration. Timestamps were also included in the instructions so outputs could be more easily traced back to the transcript. The meeting summarization prompt template can be found in the online appendix (part E).

E. Process Model Generation

The aim of this use case is to generate a process model for the domain. A first prompt provides the LLM with role

and project context. The LLM is explicitly asked to confirm it reviewed the context in the prompt. Then, the process model generation task is triggered. This prompt provides specific guidelines to identify and outline all tasks, sub-processes, events and decision nodes. General BPMN naming conventions and constraints were included to provide a focus on their identification by the LLM. Finally, similar to data model generation, the outputs are transformed for a visualization tool. The prompt templates for process model generation can also be found in the online appendix (part E).

VI. Perceived Effectiveness of the LLM-generated RE artifacts (RQ3)

To validate the LLM prompt templates, we assessed the perceived effectiveness of the outputs. Making use of the industrial embedding of this research, we evaluated the LLM-generated outputs with the team members of the projects.

While gauging the opinion of the team members increases the authenticity of the results, this also poses limitations. Indeed, the potential set of participants is low, and there is a necessity to limit the required effort from the industry experts. Thus, we selected three use cases based on their importance for the Agile MDD process at the company. We excluded summarization as this is not currently an activity in the process, and left out process generation because we prioritized requirement-centric artifacts. A uniform survey was designed to validate the outputs across the four industry cases (Sect. III-A). The survey focused on:

1) the perceived quality of the generated artifacts;
2) the perceived artifact applicability in the projects.

The former concern was addressed by means of a set of questions for each of the generated artifacts, for example, for one user story. The latter was studied via questions regarding the whole generated output for a given use case.

To limit the participants' cognitive effort, all artifacts were generated for the same business process within a given project. These artifacts are user stories, acceptance criteria, and data models. The focus processes and participant information are shown in Fig. 3, participant experience is based on the function title. Each participant was involved in a single case and is therefore unique for our validation (Sect. III)

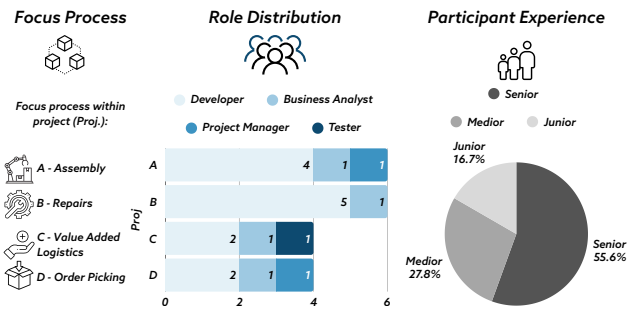

Fig. 3. Focus processes & participant information.

To maximize the data collection outputs, we also included the responses of two of the authors of this paper, as they were also project members for two different projects: *ProjA* and *ProjB*. These authors filled in the survey before accessing any of their colleagues' opinions. For transparency, we provide the survey responses in the online appendix (part B), clearly highlighting the results by the authors who are also project members. The appendix (part A) also contains example surveys for replicability.

Participants were provided with the input documents used by the LLM in order for them to possess additional context while answering the surveys.

In the following, we first discuss results per each use case, and then provide a comparison across the use cases.

A. User Story Generation

We presented a set of fifteen LLM-generated user stories to each participant. Each user story was accompanied by the categorization assigned by the LLM. The participants were asked to answer three statements on a 4-point Likert scale with an additional option "I'm not able to answer". An example showing the format is as follows:

> *'As a planner, I want to generate an assembly forecast based on historical sales data so that I can prepare for the next four weeks production needs'*
> *(Category; Forecast and Planning)*
> Please rate your agreement with the following statements:
> - The user story is in the scope of the application process.
> - The user story could have been included in the project specification as is.
> - The user story has been or will be implemented in the project.

The goal was to determine whether or not the LLM would include stories that could impact scope creep, if they were fit for specification in the project management tooling, and whether they ended up in the actual application. The results, organized by question and case, can be seen in Fig. 4. The individual responses are in the online appendix (part B).

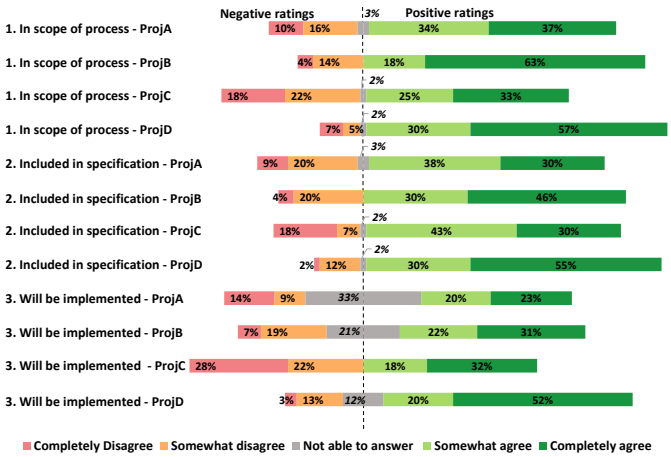

Fig. 4. Results for the user story statements.

The results show an overall positive rating for all three statements. Most stories are perceived to be in scope of the process and included in the specification. Both statements have a macro-average of 74.6% positive ratings (somewhat or completely agree), though the first statement has a higher rate of complete agreement. However, the results show that validating the generated user stories is important, as 25% of them is considered out of scope (40% for *ProjC*). The statement about the stories actually being implemented in the project has a higher number of 'not able to answer', probably due to the detailed level of application knowledge required, which is uneven among team members. The percentage of negative ratings (28%) is similar to that of the other statements.

Some user stories show only positive agreements for all statements. An example: "As a system administrator, I want the system to auto-assign repairs to technicians based on past experience so that the most qualified technician gets the task." (*ProjB* – thirteen "Completely agree", four "Somewhat agree", and one "Can't answer" over the three statements). This story also matches closely the process described in the design. In comparison, the story "As a printer operator, I want to print task sheets and corresponding labels for the assembly items so that workers can identify and track their progress." (*ProjA*) had two participants strongly disagreeing on the three statements, three participates rate the statements positively, and there was one participant with mixed results (two "Completely agree", seven "Somewhat agree", two "Can't answer", one "Somewhat disagree", and six "Completely disagree" over the three statements). This could be due to the current process being paper based, but the implementation supporting this functionality within the application (so no printed sheets).

B. Acceptance Criteria Generation

For the validation of the generated acceptance criteria, we opted for a format change and decided to validate each acceptance criterion individually, as opposed to the set of acceptance criteria for a single user story. Therefore, we also chose a different format to keep the effort requirement manageable: participants were asked to provide input on the *correctness* and *relevance* of each acceptance criterion.

We selected up to fifteen user stories contained in the functional design for the focus process. The LLM generated four acceptance criteria for each of these requirements. For each acceptance criterion, participants were asked to fill in check boxes to indicate if they deemed the criterion correct, relevant, both or neither. In this context, correctness means that it is indeed required for the project; relevance indicates it is related to the user story. For instance, for the user story from *ProjB*: "As a mechanic, I want to record the root cause of a defect so that I can trace what was wrong with the article", the AC "Root cause data must be included in repair reports" was considered relevant but not correct, because reports were not part of the project.

Table V shows the distribution of ratings across the cases. Averaged over all cases, participants found 62% of the acceptance criteria correct and 61% of them relevant. This indicates that, although they cannot be used as such, the resulting acceptance criteria contain a good amount of usable information. In particular, 37% of all answers indicate both relevance and correctness of the acceptance criteria.

TABLE V
COMBINED RESULTS FOR THE ACCEPTANCE CRITERIA VALIDATION.

	Correct	Relevant	Both	Neither	% Correct	% Relevant
ProjA	102	75	128	31	68.5%	60.4%
ProjB	85	95	126	54	58.6%	61.4%
ProjC	48	57	41	62	42.8%	47.1%
ProjD	48	54	127	11	72.9%	75.4%
Combined	283	281	422	158	61.6%	61.5%

Differences exist across projects though, with the best results in *ProjD* and the worst in *ProjC*. This is visible by the percentages in Table V, and also when examining the box-plots in Fig. 5. The latter figure also shows higher variance for correctness than for relevance, indicating better agreement among the team members when evaluating whether the generated acceptance criteria relate to the user story.

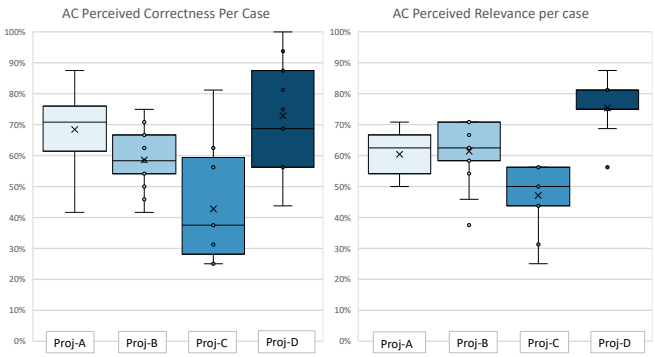

Fig. 5. Results for the acceptance criteria survey statements.

C. Data Model Generation

For the validation of the generated data models, participants were presented with a PlantUML visualization, like that shown in Fig. 6 for *ProjB*.

They were then asked, for each table, to rate the following statements on a 4-point Likert scale:

- The table is in the scope of the application process.
- The columns are within the scope of this table.
- The relationships linked to this table are accurate.
- The table and columns have been, or will be implemented in the project.

The findings from the questionnaire can be seen in Fig. 7. Here, we observe a less positive image, with a higher share of negative ratings compared to the user story statements. Especially the statement about relationships for *ProjC* & *ProjD* are rated negatively. When comparing statements, the

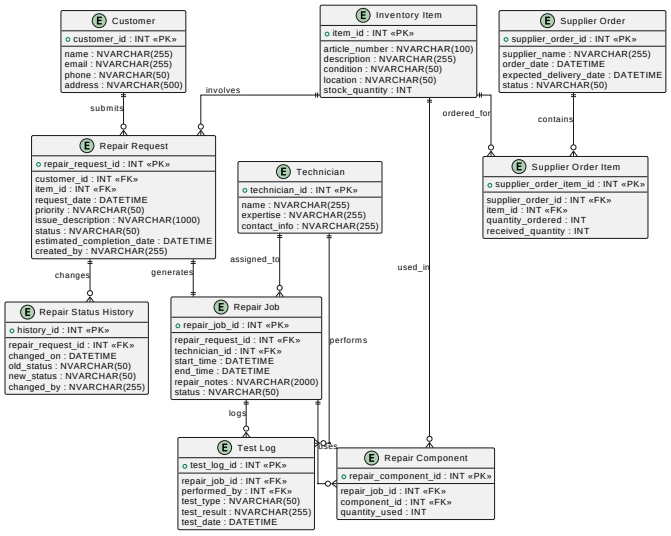

Fig. 6. Generated data model for *ProjB*.

worst average scores relate to the question on whether each table has been implemented. This has a similar number of positive and negative statements, namely 48% and 41%. This indicates a low precision in the outputs, with many generated tables perceived to be irrelevant for the actual implementation in the project.

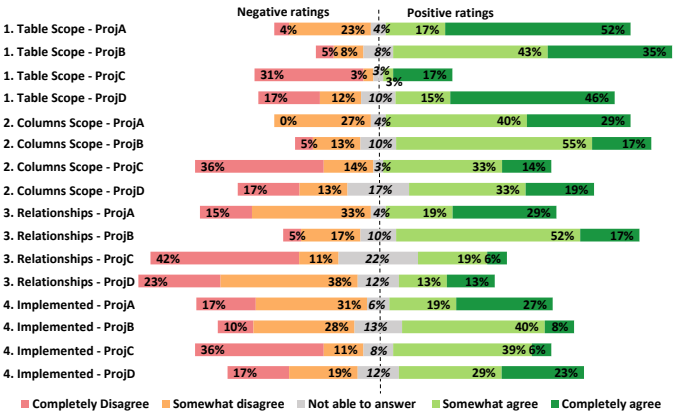

Fig. 7. Results for the data model statements.

D. Comparing the Use Cases

For each of the surveys, participants were also asked to provide their opinions on the use case in general. For each case, we asked about (a) 'similarity to the human created artifacts', (b) 'using the generated artifacts as a starting point', and (c) 'willingness to use AI in this use case'. For the data model, we added a question (d) on the relationships between tables. Moreover, we split data model generation into two use cases: (3A.) as extra context in the requirements specification, and (3B.) as actual data model implementation in the application. The survey results are visualized in Fig. 8. We combine the results between projects to reflect on the findings of the use case.

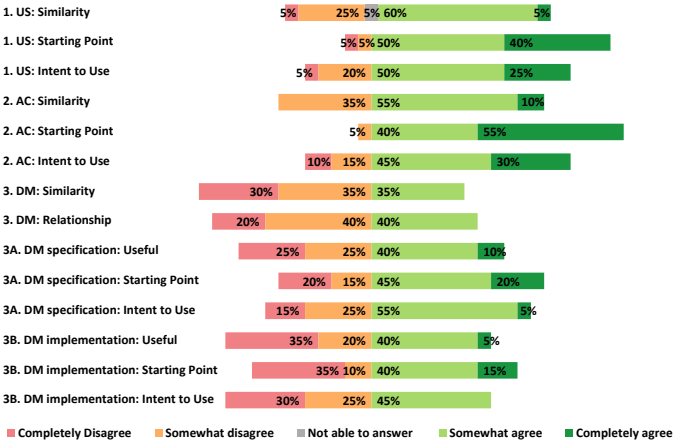

Fig. 8. Statements regarding the use cases as a whole.

While the generation of user stories and acceptance criteria have similar values for all three statements, AC generation has a slightly higher 'strongly agree rating'. We observe a high intent to use (c) of 75% (somewhat agree + completely agree) for the first two use cases, and lower values for the data model: 60% for specification (c1) and 45% for implementation (c2). The generally high intention to use can be explained by the fact these outputs can be generated with no additional effort, and can therefore be seen as a good starting point compared to starting from scratch. The better results for (c1) than (c2) is explained by the fact that, in the former use case, the generated data model is an appendix to the specification, rather than the database that guides the MDD application.

In the survey comments, the participants mentioned that using the user story outputs as a reference would be useful, but directly using them would take more time as validation is necessary to ensure correctness. Another participant was concerned that the LLM output may not cover the entire scope. Additionally, motivating business analysts to actually check the entire output was perceived as a challenge. The role part of the story was mentioned as a risk since it didn't match to roles used in the application and specification. Participants also mentioned in the comments that they are a nice starting point but need to be specified further. For further prompt improvement or additional use cases, they also noted that the description normally included in the project management tooling was missing. Two participants specifically mentioned that the generated stories are 'pretty close' to what is used in the project.

For the acceptance criteria, one participant mentioned that their quality was better than what is generally provided by the customer. However, another mentioned that while the outputs are a good starting point, they should not be given to a developer without checking between the business analyst and customer to ensure this is what was agreed, also focusing on existing price arrangements. Again, they mentioned tediousness: checking all the generated acceptance criteria for, say, a hundred user stories is not something a product owner would

look forward to. This is confirmed by the fact that the survey had an average completion time of thirty-four minutes for fifteen stories. Other comments also mentioned the lack of specificity and consistency with project terminology.

For the data models, we observed a lack of consistency between the case outputs. Three of the models used the crow's foot notation, while one used a composition for each relation. Similarly, relationship labels were different for the cases. Examples include 'logs', 'has many', 'contains', and 'from'. The participants mentioned that the table naming did not follow the naming conventions provided by the low-code development platform documentation[2].

VII. Threats to validity

We review the relevant threats to validity using the taxonomy by Yin [42], which is recommended by Wohlin *et al.* [43].

Construct validity refers to the adequate choice of operational measures. Using a survey is a limiting factor, because we cannot verify whether the participants have read and understood the survey instructions, nor if the questions are understood correctly. This is an inherent limitation of our research design, which we decided for as a trade-off between effort and thoroughness. We validate the statements in isolation rather than applying an AI-supported method in the company. This may result in a different, probably more positive, opinion. Additionally, although the input files provided as context contained business process models in the BPMN notation, we did not specifically instruct the LLM to process those. Thus, we cannot be sure whether and how the LLM has used these in generating the outputs.

Internal validity is concerned with uncontrolled factors that may affect causality. First, the use of ChatGPT as an LLM poses a threat, as it provides different responses to the same question, thereby influencing the outputs to validate. We took a 'first answer' approach to prevent further impacting the validity by cherry picking, and having multiple cases mitigates this threat. However, a different output by the LLM might have impacted the findings. Additionally, to limit the participant effort, we limited the data sets validated in the surveys, i.e., fifteen user stories. Also, we measure the perceived effectiveness of the use cases through Likert surveys. While we see a high percentage of '(strongly) agree', this may be related to a positive bias, as the artifacts are generated at 'no cost'.

External validity regards the extent to which the findings can be generalized. Since we performed a context-specific industry study, generalizability is limited. However, since the prompts were designed based on the method in Sect. IV and validated within this context, we may expect similar results in other contexts where Agile MDD is used.

VIII. Conclusions and Lessons learned

We conclude this paper by addressing the research questions and, while doing so, by highlighting the lessons learned for the case company.

[2]https://docs.thinkwisesoftware.com/docs/sf/guidelines_data_modeling

RQ1. In which use cases can we employ LLMs to generate RE artifacts? In order to gain the required contextual knowledge for the Agile MDD process used at our case company, we identified a set of process steps and artifacts that led to the PDD shown in Fig. 2. This precise characterization is a starting point for the identification of use cases where LLMs can assist in generating artifacts. Additionally, through interviews with team members, we identified five RE-relevant use cases (see Sect. IV). These use cases focus on the generation of (a) user stories, (b) acceptance criteria, and (c) data models, the (d) summarization of meetings; and (e) process model generation. Through the research, the case company has documented its own Agile MDD process in a precise manner, and has identified concrete use cases where LLMs can be used by potentially (semi-)automating existing activities where artifacts are generated. Beyond the validated use cases, this approach supports further method engineering both for integration of LLMs and general process improvements. Additionally, the explicit models facilitate communication about the 'way of working' with new employees or customers.

RQ2. What is a re-usable set of prompt templates that can be applied in the identified use cases? We selected the five use cases identified in RQ1 to then iteratively design a re-usable set of prompts to generate RE-related artifacts in the context of the considered Agile MDD process. By assembling these prompts, the case company increased its internal knowledge on practical aspects for the effective use of LLMs (more specifically, ChatGPT-4o) to automatically generate project artifacts. The prompts also function as examples that can be adapted to facilitate other use cases. *Task breakdown* was a factor for each of the use cases: splitting up the generation into more focused prompts, i.e., per module resulted in more complete and specific output. *Context window* was important to handle the input documents and to repeat instructions for, e.g., formatting to ensure they are still being followed. In our case, we experienced issues with more than fifty user stories, which could not all be processed together, or with large input documents. The *input documentation* has an impact on the abstraction level of the generated outputs. However, we found the functional design, one of the key artifacts in the Agile MDD process (Fig. 2) to be an effective input document. Especially for data model generation, we found that *providing guidelines* on how to generate a data model was beneficial to obtain models that met our expectations. In our case, the data modeling guidelines from Thinkwise were used for the modeling. Additionally, *chain of thought prompting* [26] improved results, especially for the data model and process model generation.

RQ3. What is the perceived effectiveness of the LLM-generated RE artifacts by project team members? From the validation, we observe an *overall positive perceived effectiveness* for the set of generated artifacts. The results were generally more positive than expected, and quite some participants indicated they are inclined to use the prompts in their projects. We also identified some considerations in using the generated artifacts. For example, it is crucial to *validate the outputs*

with respect to *scope*: several of the generated artifacts are in general sensible and could be relevant, but they may fall outside the boundaries of the agreed-upon project with the client, the budget, and the project timeline. Furthermore, in some cases, we also found artifacts that are not related to their 'parent' artifact. For example, a user story that does not pertain to its epic, or an acceptance criterion outside of the scope of a user story. When comparing the generated outputs for each of the cases, we also observe a lack of *consistency* especially in the grouping of the user stories, and in the elements that are included in the data model. These findings indicate that LLM-generated outputs, at least those created with ChatGPT in our context, are a starting point for further processing but are certainly not ready to be used without any validation.

MRQ. To what extent can LLM-based RE artifacts be effectively generated within an Agile MDD process? Although imperfect we could, generate outputs for all of our five use cases by means of a reusable prompt template that does not require follow-up interaction with the LLM. Therefore, the artifacts can be created via an API call and then provided to the team members for further refinement. Especially the user stories and the acceptance criteria were deemed to be a good starting point and/or refinement of documentation in the Agile MDD process at the case company. More than half of the project members had a positive opinion regarding their intention to use AI for these use cases, except for using the data model for the application implementation. While the generated outputs require human validation for use, one key lesson learned is that LLMs can lead to efficiency gains for requirements specification tasks in the Agile MDD process by providing team members with initial, to be refined inputs.

IX. Future Research

To enhance the generality of our findings, we encourage researchers to replicate this study in different contexts, possibly using the materials provided in the online appendix. While our work is based on real project data, a logical follow-up is to embed the prompts in the company's practices.

For application to other contexts, we suggest using our provided prompts with available project documentation and reviewing the outputs. The unavoidable differences in different industrial contexts entail that adaptations will be necessary for either the input documents (the mock documentation in our appendix serves as an example) or the prompts.

To further support applicability, our future work includes the formalization of our user cases and prompts into reusable method fragments [44] that include instructions for their application to various contexts.

Future studies could also explore the use of the prompts beyond the early project phases, particularly during sprints. Finally, while we studied the applicability of the general-purpose model ChatGPT-4o, investigating the impact of domain-specific and fine-tuned models presents a promising direction to improve accuracy and usefulness of the outputs.

Acknowledgments

We would like to thank all the employees of fizor. who participated in our study. Their involvement has been essential throughout the research cycle reported in this paper.

References

[1] Digital.AI, "The 17th state of agile report," Digital.AI, Tech. Rep., 2023, [Accessed 29-03-2025]. [Online]. Available: https://info.digital.ai/rs/981-LQX-968/images/RE-SA-17th-Annual-State-Of-Agile-Report.pdf?version=0

[2] O. Matvitskyy, K. Davis, and A. Jain, "Magic quadrant for enterprise low-code application platforms," Oct 2024, [Accessed 29-03-2025]. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2J42UZEC&ct=241017&st=sb

[3] B. Hailpern and P. Tarr, "Model-driven development: The good, the bad, and the ugly," *IBM Systems Journal*, vol. 45, no. 3, pp. 451–461, 2006.

[4] M. DiCesare, "Model-Driven Development: The Foundation of Low-Code — mendix.com," https://www.mendix.com/blog/low-code-principle-1-model-driven-development/, 2024, [Accessed 29-03-2025].

[5] A. C. Bock and U. Frank, "Low-code platform," *Business & Information Systems Engineering*, vol. 63, pp. 733–740, 2021.

[6] D. Di Ruscio, D. Kolovos, J. de Lara, A. Pierantonio, M. Tisi, and M. Wimmer, "Low-code development and model-driven engineering: Two sides of the same coin?" *Software and Systems Modeling*, vol. 21, no. 2, pp. 437–446, 2022.

[7] M. Overeem, "Evolution of low-code platforms," Ph.D. dissertation, Utrecht University, 2022, https://doi.org/10.33540/1197.

[8] S. Hansson, Y. Zhao, and H. Burden, "How MAD are we? Empirical evidence for model-driven agile development," in *Proceedings of the Workshop on Extreme Modeling*, ser. CEUR Workshop Proceedings, vol. 1239, 2014, pp. 2–11.

[9] M. Snoeck and Y. Wautelet, "Agile MERODE: A model-driven software engineering method for user-centric and value-based development," *Software and Systems Modeling*, vol. 21, no. 4, pp. 1469–1494, 2022.

[10] A. Urlana, C. V. Kumar, A. K. Singh, B. M. Garlapati, S. R. Chalamala, and R. Mishra, "LLMs with industrial lens: Deciphering the challenges and prospects–a survey," *arXiv preprint arXiv:2402.14558*, 2024.

[11] A. Karim Jallow, P. Demian, A. N. Baldwin, and C. Anumba, "An empirical study of the complexity of requirements management in construction projects," *Engineering, Construction and Architectural Management*, vol. 21, no. 5, pp. 505–531, 2014.

[12] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "An empirical study of the non-determinism of ChatGPT in code generation," *ACM Transactions on Software Engineering and Methdoology*, vol. 34, no. 2, Jan. 2025.

[13] V. Jain, J. M. Chatterjee, A. Bansal, U. Kose, and A. Jain, *Computational Intelligence in Software Modeling*. Walter de Gruyter GmbH & Co KG, 2022.

[14] C. Lemieux, J. P. Inala, S. K. Lahiri, and S. Sen, "CODAMOSA: Escaping coverage plateaus in test generation with pre-trained large language models," in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, 2023.

[15] T. Ahmed and P. Devanbu, "Few-shot training LLMs for project-specific code-summarization," in *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2022, pp. 1–5.

[16] A. S. Pothukuchi, L. V. Kota, and V. Mallikarjunaradhya, "Impact of Generative AI on the software development lifecycle (SDLC)," *International Journal of Creative Research Thoughts*, vol. 11, no. 8, 2023.

[17] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through Generative AI: Assessing the role of LLMs," in *Generative AI for Effective Software Development*, A. Nguyen-Duc, P. Abrahamsson, and F. Khomh, Eds. Switzerland: Springer, 2024, pp. 129–148.

[18] K. Ronanki, C. Berger, and J. Horkoff, "Investigating ChatGPT's potential to assist in requirements elicitation processes," in *Proceedings of the Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2023, pp. 354–361.

[19] B. Görer and F. B. Aydemir, "Generating requirements elicitation interview scripts with large language models," in *Proceedings of the IEEE International Requirements Engineering Conference Workshops (REW)*. IEEE, 2023, pp. 44–51.

[20] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Automated handling of anaphoric ambiguity in requirements: A multi-solution study," in *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, 2022, pp. 187–199.

[21] G. Voria, F. Casillo, C. Gravino, G. Catolino, and F. Palomba, "RECOVER: Toward requirements generation from stakeholders' conversations," *IEEE Transactions on Software Engineering*, 2025.

[22] Z. Li, W. Qiu, P. Ma, Y. Li, Y. Li, S. He, B. Jiang, S. Wang, and W. Gu, "An empirical study on large language models in accuracy and robustness under Chinese industrial scenarios," *arXiv preprint arXiv:2402.01723*, 2024.

[23] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, and D. C. Schmidt, "A prompt pattern catalog to enhance prompt engineering with ChatGPT," *arXiv preprint arXiv:2302.11382*, 2023.

[24] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: Elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.

[25] OpenAI, "Prompt engineering best practices for ChatGPT," 2024, accessed: 2024-10-25. [Online]. Available: https://help.openai.com/en/articles/10032626-prompt-engineering-best-practices-for-chatgpt

[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," in *Proceedings of the International Conference on Neural Information Processing Systems (NIPS)*, 2022, pp. 24 824–24 837.

[27] S. W. Ambler, *Agile modeling: Effective practices for extreme programming and the unified process*. John Wiley & Sons, 2002.

[28] V. Kulkarni, S. Reddy, S. Barat, and J. Dutta, "Toward a symbiotic approach leveraging Generative AI for Model Driven Engineering," in *Proceedings of the ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MODELS)*. IEEE, 2023, pp. 184–193.

[29] A. R. Sadik, S. Brulin, and M. Olhofer, "Coding by design: GPT-4 empowers Agile Model Driven Development," in *Proceedings of the International Conference on Model-Based Software and Systems Engineering (MODELSWARD)*, 2024, pp. 149–156.

[30] Thinkwise, "Unleashing the potential of GAI for advanced application development," https://www.thinkwisesoftware.com/ai, 2025, [Accessed 29-03-2025].

[31] D. Roest, "Mendix Release 10.21 - AI AI AI, boosting developer productivity — Mendix — mendix.com," https://www.mendix.com/blog/mendix-release-10-21-ai-ai-ai-boosting-developer-productivity/, Mendix, 2025, [Accessed 29-03-2025].

[32] L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh, "The case for context-driven software engineering research: Generalizability is overrated," *IEEE Software*, vol. 34, no. 5, pp. 72–75, 2017.

[33] R. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[34] I. van de Weerd and S. Brinkkemper, "Meta-modeling for situational analysis and design methods," in *Handbook of research on modern systems analysis and design technologies and applications*. IGI Global, 2009, pp. 35–54.

[35] O. Gotel and A. C. W. Finkelstein, "An analysis of the requirements traceability problem," *Proceedings of IEEE International Conference on Requirements Engineering (ICRE)*, pp. 94–101, 1994.

[36] T. Spijkman, F. Dalpiaz, and S. Brinkkemper, "Requirements elicitation via fit-gap analysis: A view through the grounded theory lens," in *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 2021, pp. 363–380.

[37] D. Zowghi and C. Coulin, "Requirements elicitation: A survey of techniques, approaches, and tools," in *Engineering and managing software requirements*. Springer, 2005, pp. 19–46.

[38] T. Spijkman, X. de Bondt, F. Dalpiaz, and S. Brinkkemper, "Summarization of elicitation conversations to locate requirements-relevant information," in *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer, 2023, pp. 122–139.

[39] G. Marvin, N. Hellen, D. Jjingo, and J. Nakatumba-Nabende, "Prompt engineering in large language models," in *Proceedings of the International Conference on Data Intelligence and Cognitive Informatics (ICDICI)*. Springer, 2023, pp. 387–402.

[40] S. Arvidsson and J. Axell, "Prompt engineering guidelines for LLMs in requirements engineering," University of Gothenburg, 2023, BSc thesis. [Online]. Available: https://hdl.handle.net/2077/77967

[41] OpenAI, "Prompt engineering guide," 2024, accessed: 2024-10-25. [Online]. Available: https://platform.openai.com/docs/guides/prompt-engineering

[42] R. K. Yin, *Case study research and applications*. Sage Thousand Oaks, CA, 2018, vol. 6.

[43] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2014, pp. 1–10.

[44] S. Brinkkemper, M. Saeki, and F. Harmsen, "Assembly techniques for method engineering," in *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 1998, pp. 381–400.