A Situational Method to Enable Pattern-based Requirement Specifications as Low-Code/No-Code Software Models[*]

David Mosquera[1,2]

[1]ZHAW Zurich University of Applied Sciences, Gertrudstrasse 15, Winterthur 8400, Switzerland
[2]PROS-VRAIN: Valencian Research Institute for Artificial Intelligence, Universitat Politècnica de València, Camí de Vera, s/n, València 46022, Spain

Abstract

.**Main problem.** Requirement patterns have emerged as a means to support the reuse and systematic production of requirement specifications, improving both efficiency and quality—typically in the form of textual documents. In Low-Code/No-Code (LCNC) software development, however, requirements are represented as software models rather than textual artefacts, still requiring effort to transform requirement specifications into software models. Although several authors have proposed approaches to reduce the software modelling effort in LCNC tools—i.e., modelling assistance approaches—the end-to-end integration between pattern−based requirement specifications and software modelling in LCNC software development remains overlooked. **Solution.** Therefore, this PhD Thesis introduces MARPa: a **M**odelling **A**ssistance and **R**equirements **Pa**ttern-based situational method for LCNC software development. MARPa enables requirements engineers, stakeholders, and LCNC developers to collaboratively develop software using LCNC tools, reusing requirement patterns and leveraging modelling assistance—from pattern selection to software model creation and refinement. Following the Situational Method Engineering approach, we enable MARPa to be tailored to specific organisations and software development contexts. **Validation.** We validated MARPa through empirical studies aimed at evaluating MARPa's technical and social suitability, as well as their effects on subjects' effectiveness, efficiency, and perceptions during requirement specification using LCNC tools to analyze MARPa's technology acceptance. Our results show that integrating requirement patterns with modelling assistance following MARPa is technically feasible and holds the potential to improve LCNC development based on efficiency and effectiveness results. Furthermore, we identified challenges resulting from our empirical validations, providing a foundation for a research agenda.

Keywords

Situational Method, Modelling Assistance, Requirement Patterns, Model Driven Development, Low Code No Code

1. Introduction

Requirements engineering—as a socio-technical, iterative process to elicit, document, and manage the requirements of a system under development [1]—is the foundation of high-quality software [2]. Part of requirements engineering is the *specification* of *software requirements*. Software requirements are capabilities—functional or non-functional—that must be met in order for the software to solve a real-world problem [3]. Software requirements can be specified as documents, text descriptions, but also as *software models* [1]. A software model is an *abstract* representation of an existing reality or a reality to be created.

Software systems often interact and involve concepts of complex physical and virtual systems in which it is executed. Thus, requirements engineers create software models to describe abstractly how a software system should behave, react to certain events, and store data, among other software functionalities. Unlike physical systems as metro lines for civil engineers, software models hold

PoEM2025: Companion Proceedings of the 18th IFIP Working Conference on the Practice of Enterprise Modeling: PoEM Forum, Doctoral Consortium, Business Case and Tool Forum, Workshops, December 3-5, 2025, Geneva, Switzerland

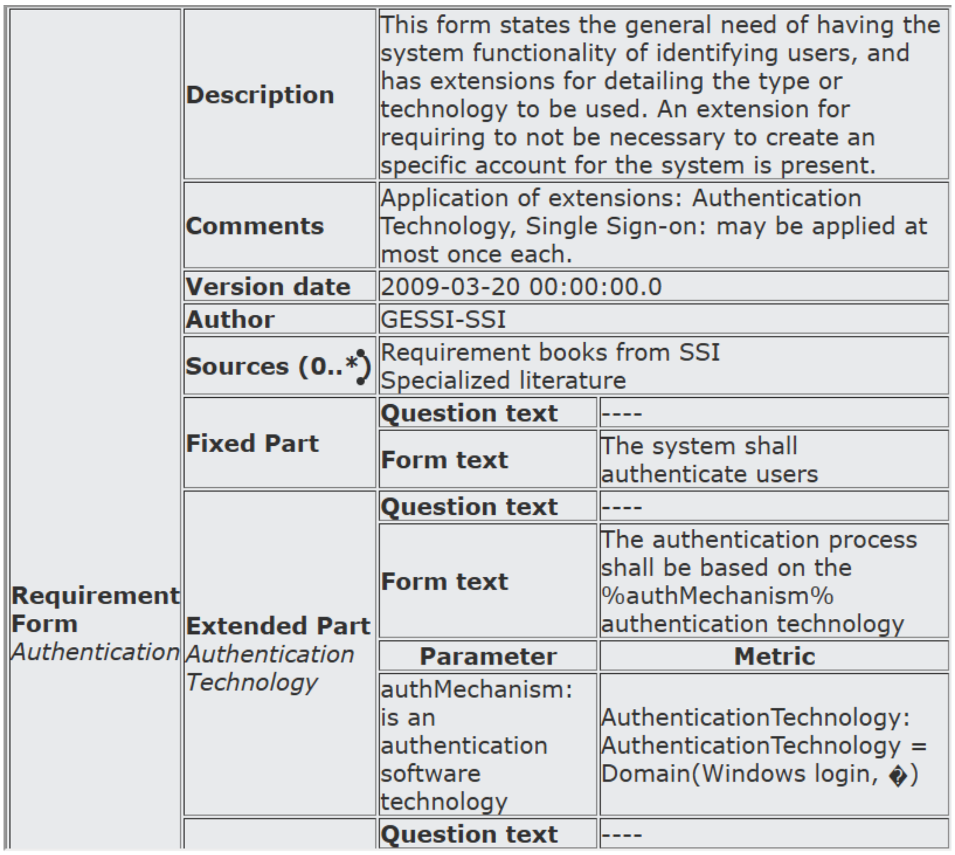
[*]Supervised by: Prof. Dr. Marcela Ruiz and Prof. Dr. Oscar Pastor.

mosq@zhaw.ch (D. Mosquera)
https://www.linkedin.com/in/jdmosquerat (D. Mosquera)
0000-0002-0552-7878 (D. Mosquera)

the potential to automatically generate the software systems they describe by using software model transformations [4] implemented in Low-code/No-code (LCNC) tools—a.k.a. model-driven development tools. LCNC is a paradigm for software development that relies on graphical/textual modelling languages and configuration, rather than traditional coding [5]. For example, a software data model specifies the data requirements of a system and can be automatically transformed into a database schema through an LCNC tool, thus removing the need to manually define data tables, attributes, and relationships in SQL.

Nevertheless, requirements specification as well as software modelling, are often error-prone and effort-intensive, which can threaten both the time-to-market and the overall quality of the software system under development [6]. Thus, several techniques have been proposed to ease the requirements specification and software modelling process, thereby speeding up and improving software development. In this paper, we focus particularly on two techniques: requirement patterns and software modelling assistance.

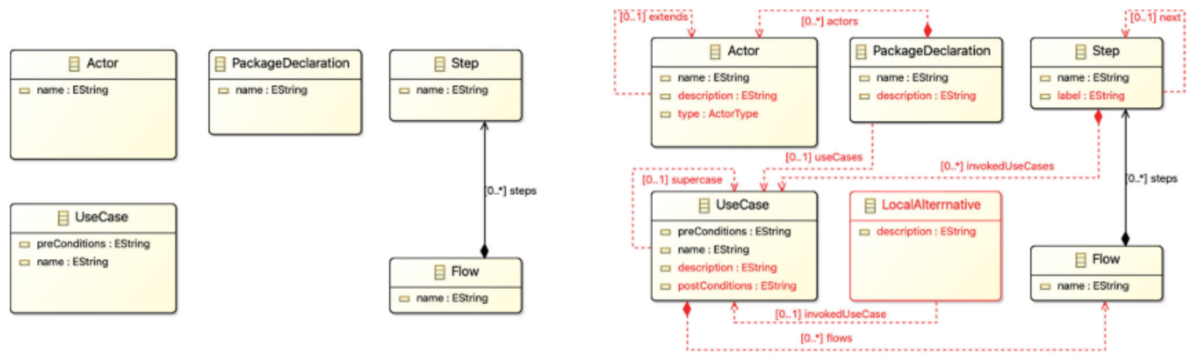
Requirements Patterns are reusable, experience-based specification that supports the reuse of high-quality software requirements to specify new requirements, reducing the time and effort required to produce new specifications [7]. Approaches in literature include pattern catalogues, specification languages, and pattern templates, among others [8, 7, 9]. An example of a requirement pattern is shown in Figure 1, where the *Authentication* pattern expresses the need to identify users and provides a template specifying the information to be documented when creating a new requirement specification.



Requirement Form *Authentication*	**Description**	This form states the general need of having the system functionality of identifying users, and has extensions for detailing the type or technology to be used. An extension for requiring to not be necessary to create an specific account for the system is present.	
	Comments	Application of extensions: Authentication Technology, Single Sign-on: may be applied at most once each.	
	Version date	2009-03-20 00:00:00.0	
	Author	GESSI-SSI	
	Sources (0..*)	Requirement books from SSI Specialized literature	
	Fixed Part	**Question text**	----
		Form text	The system shall authenticate users
	Extended Part *Authentication Technology*	**Question text**	----
		Form text	The authentication process shall be based on the %authMechanism% authentication technology
		Parameter	**Metric**
		authMechanism: is an authentication software technology	AuthenticationTechnology: AuthenticationTechnology = Domain(Windows login, �)
		Question text	----

Figure 1: *Authentication* requirement pattern template from PABRE catalogue [10].

Software modelling assistance refers to strategies—such as tools, techniques, and frameworks—that aim to ease the software modelling within LCNC tools, reducing software modelling effort and improving model quality [11, 12]. For example, Figure 2 shows a modelling assistant that suggests model elements and operations while modelling.



(a) Partial metamodel (b) Complete metamodel

Figure 2: Modelling assistance example: suggesting model elements [13].

Despite progress in research on modelling assistance frameworks and requirement patterns methods, we have identified gaps in both literature and practice. Modelling assistance frameworks [11, 14] have overlooked requirements engineering techniques, such as requirement patterns, focusing mainly on specific tasks related to software modelling. On the other hand, requirement pattern methods [10, 15] stop at the level of producing textual requirements specifications, overlooking the effort required to transform such specifications into software models. The proposals that do focus on generating software models from requirement patterns are tied to specific modelling languages (such as event and use case diagrams [16, 17]), which restricts their integration with LCNC tools with different software modelling languages.

In practice, LCNC tool providers have developed techniques that resemble the reuse of requirement patterns and modelling assistance to create software models. For instance, solutions such as OutSystems Forge and the Mendix Marketplace provide catalogues and wizards to reuse software models from templates [18, 19, 20]. However, these solutions provide ad-hoc support for reuse and automation but fail to provide method-level guidelines beyond their respective LCNC tool.

Therefore, in this Doctoral Consortium paper, we summarise the main technical research problem (MTRP) of the PhD thesis as:

> **(MTRP)** How to **design a method** that provides **guidelines and tools** for **leveraging the benefits of modelling assistance and requirement patterns** in the context of **LCNC software development**?

This paper is structured as follows: in Section 2, we introduce the research method; in Section 3, we review related works and motivation for this thesis; in Section 4, we introduce MARPa design in a nutshell; in Section 5, we overview the validation pipeline of the thesis; and in Section 6, we draw conclusions, review limitations, and propose a research agenda.

2. Research Method

In Design Science, the object of study is an artefact in context [21]. In this PhD thesis, the MARPa (**M**odelling **A**ssistance and **R**equirement **Pa**ttern method for LCNC software development) and its chunks are the artefacts we design and investigate in the context of LCNC software development. We propose a set of research questions to address our MRTP, followed by a design cycle.

2.1. Research Questions

MARPa research questions are divided into technical research problems (TRP)—a.k.a., design problems—and knowledge questions (KQ). **Technical research problems** aim to (re)design an artefact, contributing to achieving some goal [21]. **Knowledge questions** focus on learning about the world without calling for an improvement [21]. We propose the following TRP and KQ for MARPa:

- **RQ1. (KQ) What is the landscape of modelling assistance and requirement patterns?** To answer RQ1, we conduct research efforts to understand related works' goals, strategies, limitations, and users in LCNC software development, especially for modelling assistance and requirement patterns literature and practice. We conducted a systematic literature review and focus groups to answer RQ1, published in [12, 14].
- **RQ2. (TRP) How to design the MARPa method and develop tools to support it?** RQ2 contains the main TRP. To answer RQ2, we conduct Method Engineering efforts following the Situational Method Engineering principles [22] and propose a set of chunks, a process map, a metamodel, and a set of context criteria to tailor MARPa into specific contexts. Moreover, we develop tools that extend and support MARPa chunks, allowing us to introduce the LEMON framework [23, 24] and OntoTrace [25, 26].
- **RQ3. (KQ) What are the effects of implementing MARPa in context?** To answer RQ3, we perform empirical efforts to validate MARPa and its chunks in context in terms of efficiency, effectiveness, and satisfaction. Moreover, we collect data from experts about the potential technical and social suitability of MARPa tailored into a specific software development context. The results from validation are currently under review, in execution, and some have been published [27].

2.2. Design Cycle

We perform three tasks (T) in a design cycle to answer the proposed research questions: i) (T1) problem investigation; ii) (T2) treatment design; and iii) (T3) treatment validation. In Figure 3, we present our Design Science Cycle detailing activities conducted for each T1, T2, and T3 tasks.

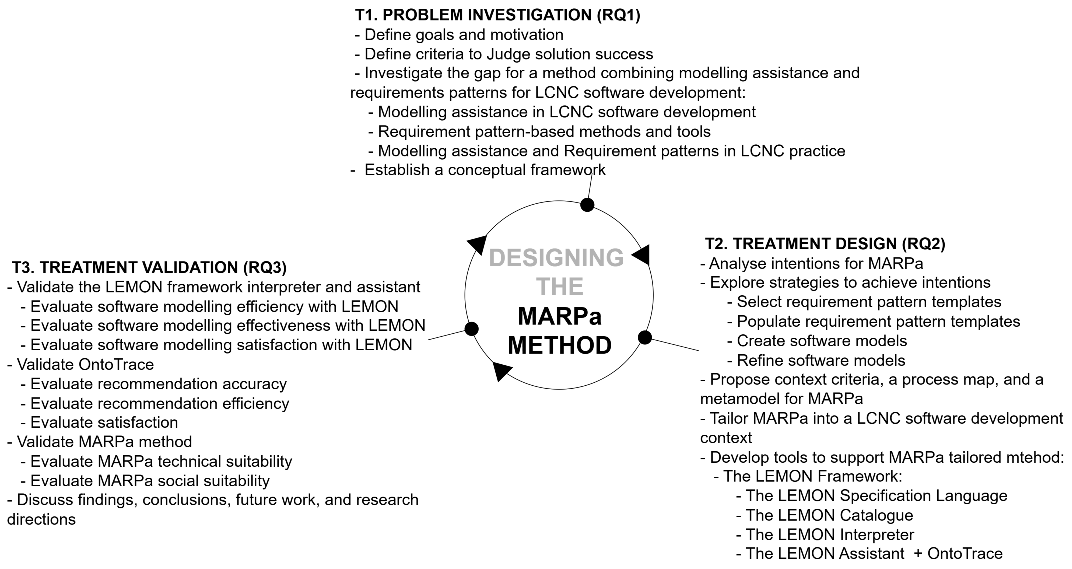

T1. PROBLEM INVESTIGATION (RQ1)
- Define goals and motivation
- Define criteria to Judge solution success
- Investigate the gap for a method combining modelling assistance and requirements patterns for LCNC software development:
- Modelling assistance in LCNC software development
- Requirement pattern-based methods and tools
- Modelling assistance and Requirement patterns in LCNC practice
- Establish a conceptual framework
DESIGNING THE MARPa METHOD
T2. TREATMENT DESIGN (RQ2)
- Analyse intentions for MARPa
- Explore strategies to achieve intentions
- Select requirement pattern templates
- Populate requirement pattern templates
- Create software models
- Refine software models
- Propose context criteria, a process map, and a metamodel for MARPa
- Tailor MARPa into a LCNC software development context
- Develop tools to support MARPa tailored mtehod:
- The LEMON Framework:
- The LEMON Specification Language
- The LEMON Catalogue
- The LEMON Interpreter
- The LEMON Assistant + OntoTrace
T3. TREATMENT VALIDATION (RQ3)
- Validate the LEMON framework interpreter and assistant
- Evaluate software modelling efficiency with LEMON
- Evaluate software modelling effectiveness with LEMON
- Evaluate software modelling satisfaction with LEMON
- Validate OntoTrace
- Evaluate recommendation accuracy
- Evaluate recommendation efficiency
- Evaluate satisfaction
- Validate MARPa method
- Evaluate MARPa technical suitability
- Evaluate MARPa social suitability
- Discuss findings, conclusions, future work, and research directions

Figure 3: Design cycle for designing the MARPa method.

For the sake of brevity, we do not present each individual design cycle in detail. Instead, we describe the design cycle that guides this PhD thesis, as it reflects the overall research method. Nevertheless, each contribution consists of smaller and more atomic design cycles. Although this may appear similar to a waterfall approach, the research was conducted in an agile manner: tasks T1, T2, and T3 were revisited whenever clarification of value, refinement of requirements, or additional artefact evaluation was needed.

3. Problem Investigation: The Landscape of Modelling Assistance and Requirement Patterns

In literature, several authors have proposed modelling assistance and requirement pattern approaches, as highlighted in secondary studies such as systematic reviews [12, 28, 8, 7]. On one hand, modelling assistance proposals aim to provide support for model creation, refinement, consistency checking, and testing—ranging from AI-driven recommenders [29, 13, 30, 31, 32, 33, 34] to validation and consistency checking tools [35, 36, 37, 38, 39, 40, 41]. Frameworks such as RF-IMA (Reference Framework for Intelligent Modelling Assistance) [42] and the emerging modelling assistance framework [14] clarify how assistants gather context and interact with users. However, these frameworks and tools overlook earlier requirements tasks and focus on easing modelling, missing the benefits of using requirement-centred techniques, such as requirement patterns.

On the other hand, requirement pattern research has introduced languages, templates, and catalogues [17, 43, 44, 45, 46, 47] as well as methods such as PABRE [10, 48] and CaRePa [15] to guide specification, selection, and reuse during elicitation. While effective for specifying text-based requirements, these methods overlook the effort of software modelling in LCNC tools. In practice, LCNC tools (e.g., OutSystems Forge [49], Mendix Marketplace [18], Oracle APEX [20]) partially combine (requirement pattern) templates and modelling assistance with marketplaces, wizards, and forges, but remain tool-specific and ad-hoc approaches.

Finally, recent LCNC development methods such as the Low Code Development Cycle (LCDC) [5], MSDeveloper [50], EasInnova [51], and situational methods for manufacturing companies [52] improve process guidance and stakeholder involvement. Yet, they do not explicitly integrate requirement patterns with modelling assistance.

Taken together, these findings highlight a research gap (and answer RQ1): to the best of our knowledge, there is no method that integrates requirement pattern reuse with modelling assistance to support an end-to-end transition from requirements elicitation/specification to model creation in LCNC software development. This gap motivates the design of MARPa (see Table 1), and thus, this PhD thesis.

PhD Industry-relevant project context. This PhD thesis was conducted in the context of the SHIFT project [53]—Smart Hospital: Integrated Frameworks, Tools, and Solutions—in which it contributes to the Patient and Staff Empowerment pillar. Within this pillar, one of the central questions concerns how to empower practitioners, patients, and other stakeholders to create digital health software automatically. Together with the industry partner Whatscount, we proposed to address this challenge. Whatscount is a young Swiss company that develops digital health software using its own proprietary LCNC tool, Posity Design Studio [54]. As part of their internal analysis, Whatscount observed that stakeholders often requested similar types of requirements—for instance, different clients frequently asked for visualisations of patient test data. Although the specific data types varied, the underlying requirement was essentially the same, exposing recurring LCNC model structures and patterns. Nevertheless, existing industry-tool-dependent solutions were not viable, since Whatscount relies on its own LCNC tool. At the same time, adopting modelling assistance techniques from the literature would not resolve the issue of recurring requirements, while introducing requirements patterns in isolation would generate overhead in their LCNC development process, because they use LCNC models themselves as requirement specifications, rather than traditional requirements documents. The case of Whatscount served as an industry-relevant case were the gap we highlighted in Table 1 is materialized.

Table 1
Related works vs MARPa.

Category	Ref	Name	Type	Include Requirement patterns	Include Software Models in LCNC	Include Software Modelling Assistance
Modelling assistance in LCNC software development	[29, 13, 30, 31] [33, 34, 32, 35] [36, 37, 40, 41] [38, 39]	Approaches for Modelling Assistance	Tool	NO	**YES**	**YES**
	[42]	RF-IMA	Framework	NO	**YES**	**YES**
	[14]	Emerging framework for Modelling Assistance	Framework	NO	**YES**	**YES**
	[49, 18, 20]	Ad-hoc Requirement Pattern-based Modelling Assistants	Tool	**YES**	**YES**	**YES**
Requirement pattern-based methods and tools	[17, 43, 44, 45, 46, 47]	Approaches for requirement pattern specification selection, reuse	Tool	**YES**	**YES**	NO
	[10, 48]	PABRE	**Method**	**YES**	NO	NO
	[15]	CaRePa	**Method**	**YES**	NO	NO
LCNC Software Development Methods	[5]	Low Code Development Cycle (LCDC)	**Method**	NO	**YES**	NO
	[50]	MSDeveloper	**Method**	NO	**YES**	NO
	[51]	EasInnova	**Method**	NO	**YES**	NO
	[52]	Situational Method for LCNC manufacturing development	**Method**	NO	**YES**	NO
(This Phd Thesis) Modelling assistance and requirement pattern-based method for LCNC Software Development		MARPa	**Method**	**YES**	**YES**	**YES**

4. MARPa Design in a Nutshell

We conceive MARPa as a Situational Method based on the Situational Method Engineering (SME) [22, 55] approach. SME supports the definition of new methods as a composition of method chunks, where each method chunk enables the satisfaction of a method intention in a specific way. Method chunks are black boxes composed of a process and a metamodel that act as a transformation engine to change a set of inputs into an output. This allows us to build MARPa in an agile and incremental manner, adding new intentions and strategies, and thereby chaining demand to satisfy various specific development contexts. Method chunks are the result of combining a strategy to achieve an intention.

MARPa covers four intentions: i) *selecting a requirement pattern template*, ii) *populating a requirement pattern template*, iii) *creating a software model*, and iv) *refining a software model*. By analysing such intentions, we provide MARPa with a catalogue of method chunks, a metamodel, a process map to assemble chunks into a tailorable method, and a set of context criteria for situation-specific method tailoring. In the following paragraphs, we provide a brief description of each intention from MARPa.

Select a Requirement Pattern Template. MARPa begins with the selection of a requirement pattern template—an instance of a pattern that can be later populated to specify a new requirement. With this intention, requirements engineers aim to find the requirement pattern template that best fits the software requirement under development.

Populate a Requirement Pattern Template. Having selected a requirement pattern template, the next intention comprises populating it. This allows the stakeholders to provide the requirement pattern template with their input—e.g., expressing how data should be displayed, how data should be stored, or how the process should be orchestrated, among others. As a result, the requirements engineer receives a populated requirement pattern template containing the customisations from stakeholders.

Create Software Model. Having the populated requirement pattern template, the next intention aims to create the software model (or models). LCNC developers represent the new requirement with the software model from the populated requirement pattern template. After creating the software model, if it is ready and does not require further refinement, the LCNC developer can use the LCNC tool to generate the software.

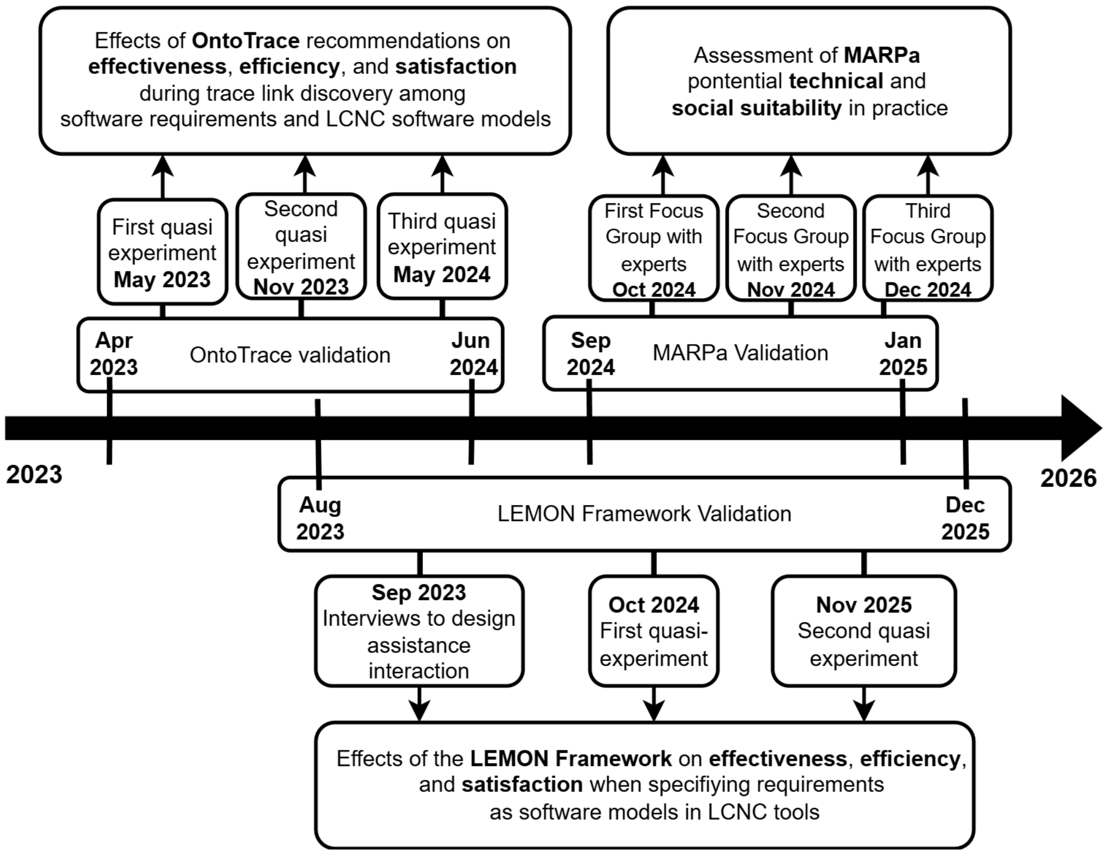
Refine software Model. If the previous intention produces a software model that is not ready for software transformation, it requires refinement. Through software model refinement, LCNC developers strive to enhance the quality of the resulting software models, thereby meeting the requirements that are being developed. When the refinement is finished, the LCNC developer can use the LCNC to generate the software.

MARPa tool support. We tailor and provide tool support for MARPa chunks tailored into the industry-relevant context of this PhD Thesis: Whatscount. As a result, we proposed the LEMON framework [23, 24], which consists of a domain-specific language for specifying software requirement patterns (M1), a requirement pattern catalogue (M2), an interpreter (M3), and an assistant (M4). To support the assistance, we designed and implemented a recommendation system named OntoTrace [25, 26], which provides recommendations based on the similarity of trace links between software requirements and software models.

5. MARPa Effect in Context: Validation Pipeline

The validation of this PhD thesis focused on the industry-tailored MARPa chunks, examining their effects within LCNC software development, thus addressing RQ3. Three complementary empirical efforts were conducted to assess: (i) the technical and social suitability of MARPa (Under review); (ii) the effects of the LEMON framework on effectiveness, efficiency, and satisfaction when specifying requirements as software models in LCNC tools (Under review); and (iii) the effects of OntoTrace recommendations on effectiveness, efficiency, and satisfaction during the tracing of software requirements within LCNC models (Published [27]). Although these evaluations are currently under publication and journal review, preliminary results suggest that MARPa is both technically and socially suitable, and that its chunks and tools hold the potential to enhance effectiveness, efficiency, and satisfaction in LCNC software development. Nevertheless, areas for improvement have been identified as a result of those validation efforts, leading to new research challenges and future work. We present the validation pipeline and timeline in Figure 4, which illustrates the number of quasi-experiments, focus groups, and interviews conducted during the validation cycle from 2023 to 2026.

Limitations and threats to validity. Throughout this PhD thesis, we have identified limitations and threats to validity. Firstly, by choosing validation efforts that relied on human participants, we encountered human-related factors that limited the generalizability of our findings; i.e., the experiments conducted were dependent on the characteristics and backgrounds of the participants involved. We decided to focus on human subjects as we were researching on how to assist LCNC users into creating software models. This motivates further industry-involved empirical efforts once MARPa is in place in this to understand their interaction and needs of other variety of backgrounds, allowing to provide a more generalizable sample of subjects. Moreover, both LEMON and OntoTrace were evaluated in specific contextual settings. Although these tools provide a general framework and a recommendation system applicable across domains, their assessment within Experimental Objects that fit a short session with human participants may differ from real-world scenarios. As an alternative, we could consider a data-driven experimental approach—such as the creation of a ground truth—instead of human-related experimentation. This could have further strengthened the evaluation and represents a promising direction for future work. This thesis and future work would benefit from creating such ground truths, breaking the human-factor dependency, and working towards a benchmark-oriented research approach.



Effects of OntoTrace recommendations on effectiveness, efficiency, and satisfaction during trace link discovery among software requirements and LCNC software models
Assessment of MARPa pontential technical and social suitability in practice
First quasi experiment May 2023
Second quasi experiment Nov 2023
Third quasi experiment May 2024
First Focus Group with experts Oct 2024
Second Focus Group with experts Nov 2024
Third Focus Group with experts Dec 2024
Apr 2023
OntoTrace validation
Jun 2024
Sep 2024
MARPa Validation
Jan 2025
2023
2026
Aug 2023
LEMON Framework Validation
Dec 2025
Sep 2023 Interviews to design assistance interaction
Oct 2024 First quasi-experiment
Nov 2025 Second quasi experiment
Effects of the LEMON Framework on effectiveness, efficiency, and satisfaction when specifiying requirements as software models in LCNC tools

Figure 4: Validation pipeline and timeline for MARPa and its chunks.

6. Lessons Learned and Research Agenda

This PhD thesis devised the first situational method to integrate requirement patterns and modelling assistance in LCNC software development, grounded in a catalogue of method chunks and validated through quasi-experiments and focus groups. This research journey has provided several lessons on designing, implementing, and evaluating MARPa. A key lesson is that technical feasibility alone is not sufficient: while MARPa chunks operationalised in the LEMON framework and OntoTrace proved effective and efficient in controlled experiments, adoption in practice requires addressing both technical refinements (e.g., catalogue growth, interaction, precision, recall, traceability) and social factors (e.g., stakeholder engagement, role displacement concerns). Another lesson is the importance of tailoring. The industry-relevant case study demonstrated that selecting and adapting the relevant MARPa chunks was crucial for aligning with industry needs, underscoring the value of situational method engineering as a foundation for method design in LCNC contexts. Future work builds on these insights; we propose a research agenda in Figure 5 to facilitate the full adoption of MARPa and its chunks.

Discussions and reflections. During execution of this PhD thesis, design decisions were made that influence how the results should be interpreted and delimit the scope of the contributions. Beyond our research agenda, there are reflections and open questions that point toward meaningful next steps:

- **Gen AI:** In this thesis, we used AI techniques that do not rely on training data—specifically, ontology-based recommendation systems—to avoid model dependency. However, given the outbreak of GenAI, this new technology could fundamentally change how requirement patterns are specified and how software models are created from them. Recent advances in GenAI for requirements pattern engineering [48] and for assisting LCNC development [56] show the relevance of this new technology. GenAI could support the creation, retrieval, and population of requirement pattern templates, or even replace some MARPa intentions by directly generating

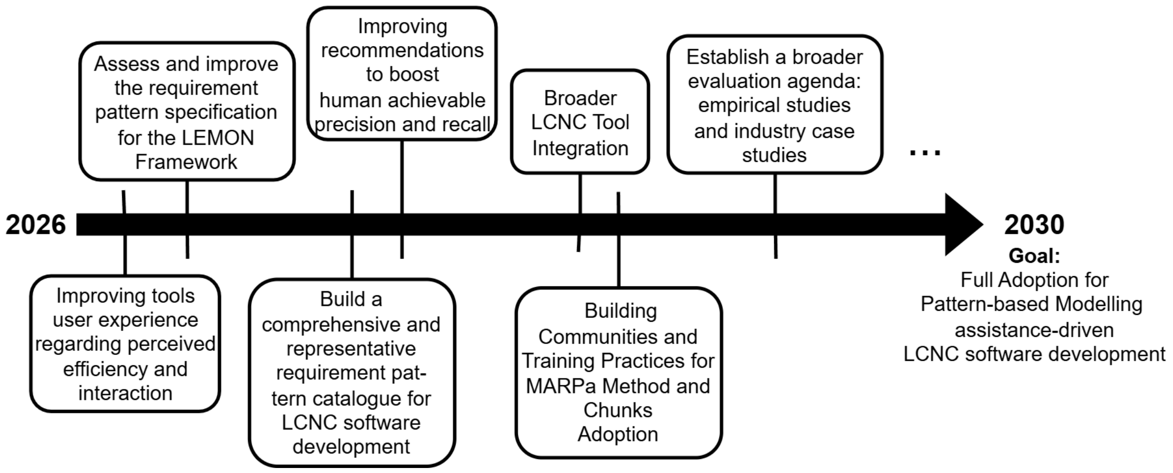

Assess and improve the requirement pattern specification for the LEMON Framework
Improving recommendations to boost human achievable precision and recall
Broader LCNC Tool Integration
Establish a broader evaluation agenda: empirical studies and industry case studies
...
2026
2030
Goal:
Full Adoption for Pattern-based Modelling assistance-driven LCNC software development
Improving tools user experience regarding perceived efficiency and interaction
Build a comprehensive and representative requirement pattern catalogue for LCNC software development
Building Communities and Training Practices for MARPa Method and Chunks Adoption

Figure 5: Research Agenda for MARPa and its chunks

pattern instances or LCNC models. This potential improvements can be driven by our research around MARPa, as a solid foundation for evaluating whether generative AI is necessary, what value it may bring, and how it could reshape the method and its intentions.

- **Integrability and scalability:** MARPa is a situational method, meaning it can grow by adding new chunks, intentions, and strategies that can later be tailored. Still, scalability remains an open question: How can the MARPa tools be extended to new domains while remaining aligned with the method? Tools such as LEMON and OntoTrace provide a basis for this interoperability, but further work is needed to support large-scale use. This challenge points to creating protocols or mechanisms for seamless integration between assistants, LCNC tools, and other modelling environments.
- **Requirement pattern life cycle:** In this thesis, we proposed tools and a method that allow to specify, select, (re)use and transform requirement pattern templates as LCNC software models. Once a template has been (re)used and its software model deployed, both the model and the underlying pattern may evolve with the time, introducing a new open point: the requirement pattern template and model life cycle. Managing this life cycle introduces an extra complexity: How should LCNC tools based on requirement patterns handle changes in requirements, templates, or generated models over time? How can traceability be maintained—possibly even bidirectionally (round-trip)? In this thesis, traceability was used to guide recommendations during pattern construction. This can serve as a foundation for more precise lifecycle-aware traceability mechanisms that link template evolution with model evolution.

We belive these discussions and reflections will guide future work, and novel progress in requirement pattern and modelling assistance research.

Declaration on Generative AI

During the preparation of this work, the author used ChatGPT and Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] M. Glinz, A Glossary of Requirements Engineering Terminology, International Requirements Engineering Board (IREB), 2014.

[2] M. Ochodek, S. Kopczyńska, Perceived importance of agile requirements engineering practices – a survey, Journal of Systems and Software 143 (2018) 29–43.

[3] P. Bourque, R. E. Fairley, I. C. Society, Guide to the Software Engineering Body of Knowledge (SWEBOK(R)): Version 3.0, IEEE Computer Society Press, 2014.

[4] S. Sendall, W. Kozaczynski, Model transformation: the heart and soul of model-driven software development, IEEE Software 20 (2003) 42–45.

[5] M. Pańkowska, Low code development cycle investigation, in: Proceedings of Ninth International Congress on Information and Communication Technology, 2024, pp. 265–275. URL: https://link.springer.com/10.1007/978-981-97-4581-4_19.

[6] A. Gupta, G. Poels, P. Bera, Using conceptual models in agile software development: A possible solution to requirements engineering challenges in agile projects, IEEE Access 10 (2022) 119745–119766.

[7] P. Mahendra, A. Ghazarian, Patterns in the requirements engineering: A survey and analysis study, in: WSEAS Transactions on Information Science and Applications, 2014, pp. 214–230.

[8] T. N. Kudo, R. F. Bulcão-Neto, A. M. Vincenzi, Requirement patterns: a tertiary study and a research agenda, IET Software 14 (2020) 18–26.

[9] C. Palomares, C. Quer, X. Franch, Requirements reuse and requirement patterns: a state of the practice survey, Empirical Software Engineering 22 (2017) 2719–2762.

[10] S. Renault, O. Mendez-Bonilla, X. Franch, C. Quer, Pabre: Pattern-based requirements elicitation, in: 2009 Third International Conference on Research Challenges in Information Science, 2009, pp. 81–92.

[11] G. Mussbacher, B. Combemale, J. Kienzle, S. Abrahão, H. Ali, N. Bencomo, M. Búr, L. Burgueño, G. Engels, P. Jeanjean, J.-M. Jézéquel, T. Kühn, S. Mosser, H. Sahraoui, E. Syriani, D. Varró, M. Weyssow, Opportunities in intelligent modeling assistance, Software and Systems Modeling 19 (2020) 1045–1053.

[12] D. Mosquera, M. Ruiz, O. Pastor, J. Spielberger, Understanding the landscape of software modelling assistants for mdse tools: A systematic mapping, Information and Software Technology 173 (2024) 107492.

[13] C. Di Sipio, J. Di Rocco, D. Di Ruscio, P. T. Nguyen, Morgan: a modeling recommender system based on graph kernel, Software and Systems Modeling 22 (2023) 1427–1449.

[14] D. Mosquera, M. Ruiz, O. Pastor, J. Spielberger, Assisted-modeling requirements for model-driven development tools, in: Research Challenges in Information Science (RCIS 2022), 2022, pp. 458–474.

[15] K. Kumar, R. K. Saravanaguru, Context aware requirement patterns (carepa) methodology and its evaluation, Far East Journal of Electronics and Communications 16 (2016) 101–117.

[16] A. R. da Silva, D. Savić, S. Vlajić, I. Antović, S. Lazarević, V. Stanojević, M. Milić, A pattern language for use cases specification, in: Proceedings of the 20th European Conference on Pattern Languages of Programs, 2015, pp. 1–18.

[17] S. Robertson, Requirements patterns via events/use cases, in: PLoP, 1996, pp. 1–16.

[18] Mendix, Mendix marketplace - industry templates, 2025. URL: https://marketplace.mendix.com/link/contenttype/106, [Accessed 02-05-2025].

[19] OutSystems, Search Forge assets from OutSystems, 2025. URL: https://www.outsystems.com/forge/list, [Accessed 02-05-2025].

[20] Oracle, Oracle appex: Using the create application wizard, 2024. URL: https://docs.oracle.com/database/apex-5.1/HTMDB/using-the-create-application-wizard.htm#HTMDB29252, [Accessed 01-12-2024].

[21] R. J. Wieringa, Design Science Methodology: For Information Systems and Software Engineering, Springer Berlin Heidelberg, 2014.

[22] B. Henderson-Sellers, J. Ralyté, P. J. Ågerfalk, M. Rossi, Situational Method Engineering, Springer Berlin Heidelberg, 2014.

[23] D. Mosquera, O. Pastor, J. Spielberger, Lemon: A tool for enhancing software requirements communication through requirements pattern-based modelling assistance, in: REFSQ2024 Posters and Demos Track, 2024, pp. 1–7.

[24] D. Mosquera, M. Ruiz, A. Martakos, A domain-specific language for specifying requirement patterns for model-driven software development, in: 15th Model-Driven Requirements Engineering Workshop (MoDRE), 2025, pp. 1–10.

[25] D. Mosquera, M. Ruiz, O. Pastor, J. Spielberger, L. Fievet, Ontotrace: A tool for supporting trace generation in software development by using ontology-based automatic reasoning, in: CAiSE2022: Forum, 2022, pp. 73–81.

[26] D. Mosquera, M. Ruiz, O. Pastor, J. Spielberger, Ontology-based automatic reasoning and nlp for tracing software requirements into models with the ontotrace tool, in: REFSQ2023, 2023, pp. 140–158.

[27] D. Mosquera, M. Ruiz, O. Pastor, Ontology-based nlp tool for tracing software requirements and conceptual models: an empirical study, Requirements Engineering 30 (2025) 341–369.

[28] M. Savary-Leblanc, X. Le Pallec, S. Gérard, Understanding the need for assistance in software modeling: interviews with experts, Software and Systems Modeling 23 (2023) 103–135.

[29] M. Savary-Leblanc, Improving mbse tools ux with ai-empowered software assistants, in: ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion, 2019, pp. 648–652.

[30] B. Adhikari, E. J. Rapos, M. Stephan, Simima: a virtual simulink intelligent modeling assistant, Software and Systems Modeling 23 (2023) 29–56.

[31] J. D. Rocco, C. D. Sipio, P. T. Nguyen, D. D. Ruscio, A. Pierantonio, Finding with nemo: a recommender system to forecast the next modeling operations, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems, 2022, pp. 154–164.

[32] L. Burgueño, R. Clarisó, S. Gérard, S. Li, J. Cabot, An nlp-based architecture for the autocompletion of partial domain models, Lecture Notes in Computer Science 12751 LNCS (2021) 91–106.

[33] S. Salemi, A. Selamat, Enhancement approachof object constraint language generation, Journal of Physics: Conference Series 933 (2018) 1–12.

[34] H. Agt-Rickauer, R.-D. Kutsche, H. Sack, Automated recommendation of related model elements for domain models, Communications in Computer and Information Science 991 (2019) 134–158.

[35] D. Ilic, E. Troubitsyna, L. Laibinis, S. Leppänen, Formal verification of consistency in model-driven development of distributed communicating systems and communication protocols, in: ISOLA 2006, 2006, p. 425–432.

[36] R. Sajjad, N. Sarwar, Nlp based verification of a uml class model, in: 2016 Sixth International Conference on Innovative Computing Technology (INTECH), 2016, p. 30–35.

[37] A. Paz, G. E. Boussaidi, H. Mili, checsdm: A method for ensuring consistency in heterogeneous safety-critical system design, IEEE Transactions on Software Engineering 47 (2021) 2713–2739.

[38] M. Ohrndorf, C. Pietsch, U. Kelter, T. Kehrer, Revision: a tool for history-based model repair recommendations, in: Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, 2018, pp. 105–108.

[39] J. Bürger, J. Jürjens, S. Wenzel, Restoring security of evolving software models using graph transformation, International Journal on Software Tools for Technology Transfer 17 (2014) 267–289.

[40] N. Almasri, B. Korel, L. Tahat, Verification approach for refactoring transformation rules of state-based models, IEEE Transactions on Software Engineering 48 (2022) 3833–3861.

[41] M. Babaei, J. Dingel, Efficient regression testing of distributed real-time reactive systems in the context of model-driven development, Software and Systems Modeling 22 (2023) 1565–1587.

[42] G. Mussbacher, B. Combemale, S. Abrahão, N. Bencomo, L. Burgueño, G. Engels, J. Kienzle, T. Kühn, S. Mosser, H. Sahraoui, M. Weyssow, Towards an assessment grid for intelligent modeling assistance, in: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2020, pp. 1–10.

[43] S. Srivastava, A repository of software requirement patterns for online examination system, International Journal of Computer Science 10 (2013) 247–255.

[44] A. Sleimi, M. Ceci, M. Sabetzadeh, L. C. Briand, J. Dann, Automated recommendation of templates

for legal requirements, in: 2020 IEEE 28th International Requirements Engineering Conference (RE), 2020, pp. 158–168.

[45] L. Sardi, A. Idri, L. Redman, H. Alami, J. Fernández-Alemán, A reusable catalog of requirements for gamified mobile health applications, in: Proceedings of the 17th International Conference on Evaluation of Novel Approaches to Software Engineering, 2022, pp. 435–442.

[46] R. Wahono, J. Cheng, Extensible requirements patterns of web application for efficient web application development, in: First International Symposium on Cyber Worlds, 2002. Proceedings., 2002, pp. 412–418.

[47] I. Darif, G. El Boussaidi, S. Kpodjedo, A. Paz, Utl: A unified language for requirements templates, in: Proceedings of the 40th ACM/SIGAPP Symposium on Applied Computing, ACM, 2025, p. 1489–1506.

[48] X. Franch, S. Gnesi, F. Paccosi, C. Quer, L. Semini, Leveraging Requirements Elicitation through Software Requirement Patterns and LLMs, Springer Nature Switzerland, 2025, p. 261–276. URL: http://dx.doi.org/10.1007/978-3-031-88531-0_19. doi:10.1007/978-3-031-88531-0_19.

[49] OutSystems, Outsystems: Application templates, 2024. URL: https://success.outsystems.com/documentation/11/building_apps/application_templates/, [Accessed 01-12-2024].

[50] B. K. Dolu, A. Cetinkaya, M. C. Kaya, S. Nazlioglu, A. H. Dogru, Msdeveloper: A variability-guided methodology for microservice-based development, Applied Sciences 12 (2022) 11439.

[51] M. Missikoff, A simple methodology for model-driven business innovation and low code implementation, Arxive (2020).

[52] J. Kirchhoff, N. Weidmann, S. Sauer, G. Engels, Situational development of low-code applications in manufacturing companies, in: Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2022, pp. 816–825.

[53] SHIFT, Blueprint for health care digitization - SHIFT | Smart Hospital; Intelligent Framework, Tools and Solutions, 2025. URL: https://future.hospital/en, [Accessed 17-11-2025].

[54] Posity, Posity AG and posity design studio, 2025. URL: https://posity.ch/EN/index.html, [Accessed 02-05-2025].

[55] I. Mirbel, J. Ralyté, Situational method engineering: combining assembly-based and roadmap-driven approaches, Requirements Engineering 11 (2006) 58–78.

[56] Mendix, Maia: AI-Assisted Development, 2025. URL: https://www.mendix.com/platform/ai/aiad/, [Accessed 02-05-2025].