Model-Driven Engineering for Implementation and Testing of Large Language Model Architectures

Jesús Carreño-Bolufer[1]

[1]Valencian Research Institute for Artificial Intelligence (VRAIN),
Universitat Politècnica de València
Camino de Vera s/n, Valencia, Spain

jecarbo@vrain.upv.es

Abstract. *Large Language Model (LLM) architectures such as DeepSeek-V3 demonstrate reductions in computational costs through the design of efficient architectures, but their development reveals technical debt in the software engineering principles associated with LLM development, partly caused by the multidisciplinary nature of the field. Consequently, it leads to increased development costs and challenges in software quality. To address these issues, this thesis proposes the integration of Model-Driven Engineering into the LLM development life cycle. A conceptual metamodel formalises LLM architectural constructs (RQ1), enabling automated code generation via model transformations (RQ2) and facilitating Model-Based testing (RQ3).*

1. Motivation

The adoption of artificial intelligence (AI)-based applications is growing rapidly in almost all domains. In particular, generative artificial intelligence (GenAI) [Banh and Strobel 2023] has captured much of the attention due to technologies such as Large Language Models (LLMs), including ChatGPT or Google Gemini [Baldassarre et al. 2023, Gozalo-Brizuela and Garrido-Merchan 2023].

LLMs are able to perform advanced tasks such as grammar correction, cross-domain reasoning, arithmetic computations and code execution [Bommasani et al. 2022]. There are multiple benchmarks to evaluate their performance in different areas [Zhao et al. 2024]. However, achieving results that exceed the state of the art requires a significant amount of computational resources. The training of GPT-4 is estimated to have cost around $78 million, while that of Gemini Ultra amounted to $191 million [Perrault and Clark 2024].

In the face of these high costs, approaches have emerged to develop more efficient models [Wan et al. 2024]. A remarkable example is DeepSeek-V3 [DeepSeek-AI 2024], an open-source LLM whose capabilities are comparable to those of models such as LLaMa-3 [Grattafiori et al. 2024] and GPT-4 [OpenAI 2023], but with a cost and training time up to ten times lower [towards data science 2025]. This has been possible thanks to optimisations in its architecture, which has improved efficiency during the training stage [DeepSeek-AI 2024].

In this way, we find synergies with software engineering, leading to an emerging line of research called SE4AI [Martínez-Fernández et al. 2022]. This line proposes the adoption and/or creation of software engineering processes that optimise the construction of intelligent systems.

In the context of software engineering, Model-Driven Engineering (MDE) [Pastor and Molina 2007] has been noted for reducing the complexity of software development through higher levels of abstraction and automation of manual tasks. In this thesis, we explore the adoption of model-based technologies to reduce cost and effort in AI development (MDE4AI) [Rädler et al. 2024]. In particular, we focus on building LLMs from design and quality assurance, considering the architecture design, implementation and testing stages.

2. Problem statement and Existing Solutions

In the field of artificial intelligence, there are several subcategories. This thesis is framed within the subdomain of Deep Learning (DL) which is part of the set of intelligent applications based on Machine Learning (ML) [Banh and Strobel 2023]. As illustrated in Figure 1, within Deep Learning is Generative Artificial Intelligence, from which LLMs emerge.

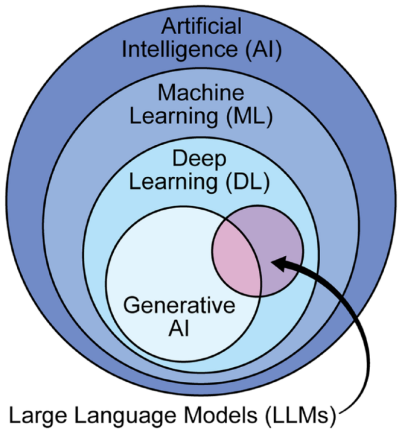

Figure 1. Classification of LLMs within artificial intelligence [Shahab et al. 2024].

The development life cycle of an LLM model can be decomposed into several phases and stages [Amershi et al. 2019, Google-Cloud 2025, Zhang et al. 2019]. This process encompasses five major phases: Analysis (1), Data (2), Software (3), Model (4) and Deployment (5) (see Figure 2).

To reduce the costs associated with LLMs, optimisation focuses mainly on the training stage (4.1), specifically on decreasing the computational costs [Hoffmann et al. 2022]. It has been shown that improving the architectural design of the AI model allows for more computationally efficient training[DeepSeek-AI 2024], achieving adequate results with lower demand on resources [Wan et al. 2024]. Examples of design decisions are the selection of parameters such as the size of the model, the number of layers, the number of heads in the multi-head attention mechanism, or the use of Mixture of Experts (MoE) in the neural network, where a router determines which components of the network are activated, thus optimising the use of computational resources.

However, to reach this goal there is a technical debt in applying software engineering [Liu et al. 2020, Sculley et al. 2015, Foidl et al. 2019, Alahdab and Çalıklı 2019] in the earlier stages of architecture design, implementation and testing (see Figure 2, Phase 3). This lack of SE principles increases development cost and decreases software quality in aspects such as maintainability, reusability and testability [ISO/IEC 25010:2023 2023].

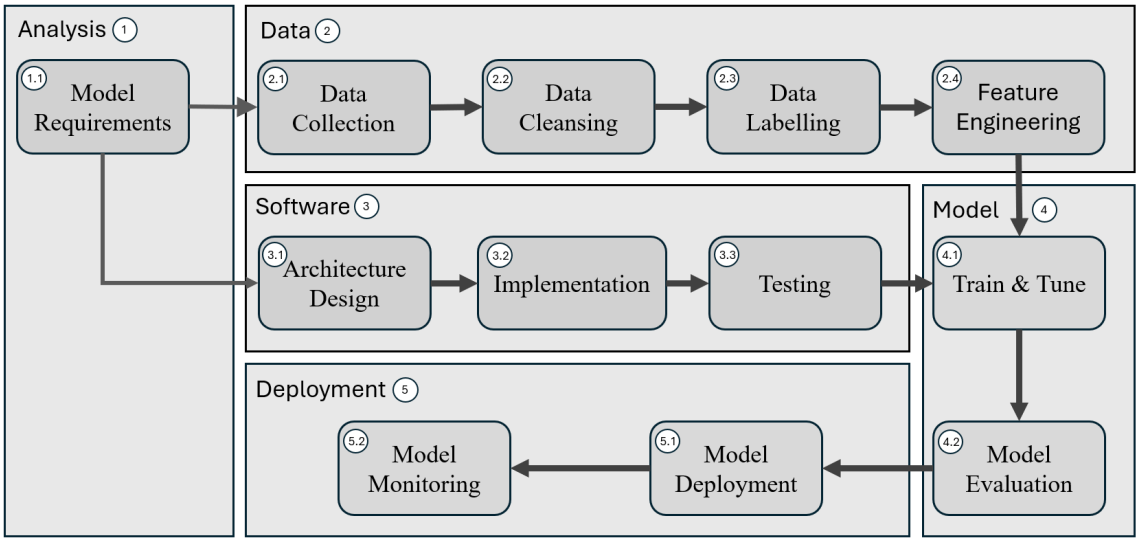

Figure 2. LLM development life cycle. The arrows guide the workflow, but it may loop back to any previous stage.

One of the main factors behind this technical debt is the multidisciplinary nature of the field. AI model development projects involve diverse profiles, such as data scientists, software engineers, domain specialists, cloud infrastructure experts, among others [Foidl et al. 2019].This requires developers to possess knowledge in software engineering and ML to build correct AI models. Furthermore, compared to other types of software, modules in these systems tend to be coupled, which hinders their modularity and evolution.

To address these challenges, proposals have emerged that are framed within the Software Engineering for Artificial Intelligence or SE4AI research domain. [Martínez-Fernández et al. 2022]. More specifically, the MDE4AI [Rädler et al. 2024] line applies the concepts of Model-Driven engineering to encapsulate the complexity associated with the design and automate the implementation of AI models. This is possible by means of model transformation rules that automate the generation of software artefacts. In this domain, the systematic review by Naveed [Naveed et al. 2024] explores the intersection of MDE and Machine Learning. In this review, effort reduction is highlighted, followed by quality improvement. In addition, we find solutions for the automatic generation of neural network code. [Díaz et al. 2015, Al-Azzoni 2020], and modelling for complexity reduction of cyber-physical systems based on deep reinforcement learning [Gatto et al. 2019]. On the other hand, Hartmann's proposal [Hartmann et al. 2019] aims to build ML algorithms into independent and reusable units with MDE techniques.

Considering the reviews and papers previously analysed, no MDE proposals have been identified that address the architecture of the AI model of an LLM in a comprehensive way. The proposals that apply MDE techniques focus on specific components, such as neural networks, or on general aspects of Machine Learning. This justifies and supports the main objective of the thesis, which is to comprehensively address the design and implementation of the architecture for LLMs using an MDE approach.

3. Objectives and Research Questions

The main objective of this thesis is the application of Model-Driven engineering to guide the architecture design, implementation and testing stages in the development of AI models associated with LLMs. In this way, by abstracting and encapsulating specific technical knowledge, as well as automating the implementation, we reduce the effort, cost, and errors inherent to manual coding [Pastor and Molina 2007].

As specific objectives, the proposal is expected to reduce costs and development times. Furthermore, it is expected to reduce the technical complexity inherent to LLM systems, facilitating the reuse of knowledge through the use of conceptual models [Pastor and Molina 2007]. Finally, it is expected to improve the quality of the resulting LLMs through the application of model-based testing techniques on the defined models [ref model based]. To address these objectives, the thesis focuses on the application of MDE in the Software Phase (see Figure 2, Phase 3), addressing the following research questions:

- **RQ1** Is it possible to characterise the conceptual constructors needed to define the software architecture of an LLM in a complete way?
- **RQ1** Is it possible to automatically implement the architecture of an LLM using model transformation rules?
- **RQ3** Is it possible to automatically generate test suites for testing an LLM from its conceptual model?

4. Research approach

For the development of this thesis, research approach based on Design Science Research (DSR) [Wieringa 2014] is defined. Thus, the research process is structured in the following phases:

- **Awareness of the Problem:** Inefficiency in the specification of Large Language Models (LLMs) and quality problems in the products developed due to the lack of software engineering practices in their architecture design, implementation and testing.
- **Suggestion:** A solution based on MDE is proposed to improve both development efficiency and software quality in LLMs.
- **Development:** The central artifact of the thesis is the conceptual model that describes the constructs associated with the architectural design of an LLM.
- **Evaluation:** The evaluation will be carried out considering the efficiency of the development, evaluating aspects such as reduction of implementation time, reduction of complexity and ease of integration of components in the process of building LLMs. Additionally, quality improvement will be analysed considering aspects such as reusability, maintainability and testability [ISO/IEC 25010:2023 2023]. The proposal will be applied in two case studies framed in European projects i) Music360[1]: Analysis of the impact of music in different contexts [Giachetti et al. 2023], and ii) Better [2]: Precision medicine based on genomic analysis.

[1]Music360. https://music-360.eu/

[2]Better. https://www.better-health-project.eu/

Our proposal consider three iterations that also applies the Design Science principles, these iterations are aligned with the research questions proposed, as follows.

1. **Iteration 1 (Year 1)**. Definition and generation of a conceptual model for LLMs architecture specification. Aligned with RQ1. This proposed conceptual model will be supported by a MOF metamodel that formalises its abstract syntax [Object Management Group 2019]. In addition, specific model editors will be implemented to provide notational support to the conceptual model, as well as model transformation tools to automate the generation of software artifacts.
2. **Iteration 2 (Year 2)**. Definition and implementation of Model-to-Text (M2T) transformation rules to generate the LLMs architecture software and configuration of components according the to LLM conceptual model instantiation [Pastor and Molina 2007]. Aligned with RQ2. The transformations will be supported by existing open-source technologies to facilitate its communication to the community and application by practitioners.
3. **Iteration 3 (Year 3)**. Definition and implementation of the suite for the generation of test cases from the LLMs models. Aligned with RQ3. The implementation of this test cases generation will be based on the principles of model-based testing considering approaches such as Marin's [Marín et al. 2017] in our research team.

5. Current status

As a starting point for the development of the thesis, the activities in the life cycle of an LLM were identified (Figure 2). Considering this life cycle, the thesis applies MDE principles framed within the software phase (see Figure 3, Phase 3). Thus, the main definitions for addressing the Research Questions and planning proposed are given below.

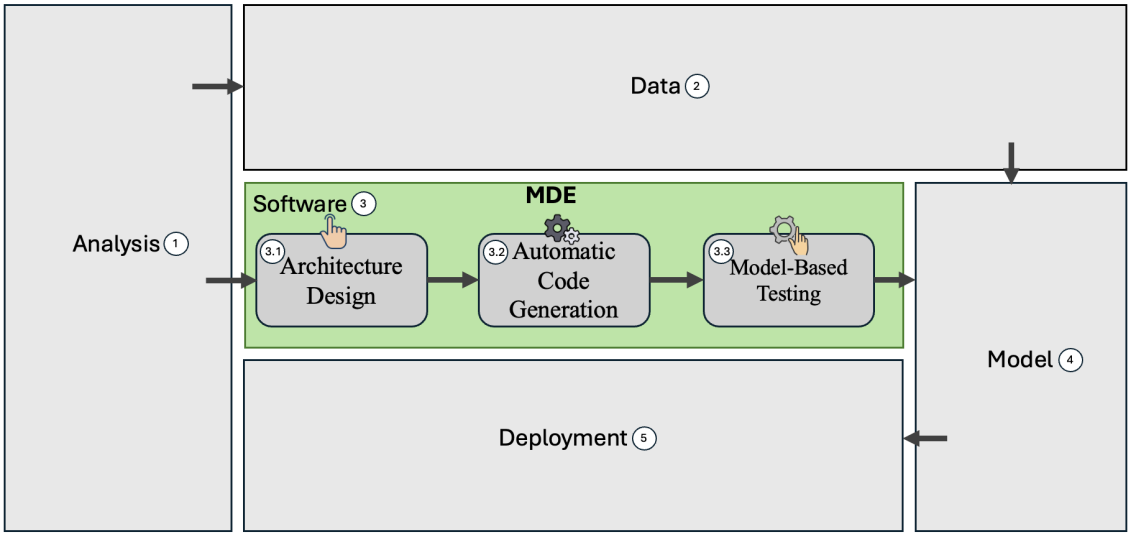

Figure 3. LLM Development life cycle highlighting the Software Phase where the MDE solution is applied.

- **Architecture Design (Stage 3.1)**: Associated with RQ1-Year 1. This stage is supported by metamodel and a modelling tool that will allow the metamodel to be instantiated in accordance with the requirements established in the analysis phase (Phase 1).

- **Automatic Code Generation (3.2)**: Associated to RQ2-Year 2. The implementation stage will be automated by means of model transformation rules for code generation in frameworks such as PyTorch[3]. This process will be automatic and transparent to the designer, adopting a Model is the Code approach [Pastor and Molina 2007]. In this way, modifications to the architecture during the life cycle can be made at the design level, avoiding direct programming on the code.
- **Model-Based Testing (3.3)**: Associated with RQ3-Year 3. The testing of the LLM model, before and after training (Phase 4), can be supported on the conceptual model designed, especially in those techniques that require configuration or parametrisation of the architecture. An example of this is the Mutation Testing technique [Çetiner et al. 2024], which can be applied for assessing the robustness of Deep Neural Networks (DDNs) and Transformer-based [Vaswani et al. 2023] models.

At the time of writing this paper, the thesis has been in progress for three months, including the definition of the methodology and a proposed solution. Currently, RQ1 is being addressed, whose objective is the conceptualisation of the architecture of an LLM, specifically the Transformer architecture [Vaswani et al. 2023], used as a reference in different LLMs such as GPT-4 [OpenAI 2023] and DeepSeek [DeepSeek-AI 2024]. In order to properly establish the conceptual constructs associated with the configuration of the LLM architecture, a systematic literature review is being carried out to identify the design decisions made by experts in the field.

Ackowledgments

Work partially funded by Horizon Europe RIA programme Music360 grant No 101094872.

References

Al-Azzoni, I. (2020). Model driven approach for neural networks. In *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*, pages 87–94.

Alahdab, M. and Çalıklı, G. (2019). Empirical analysis of hidden technical debt patterns in machine learning software. In Franch, X., Männistö, T., and Martínez-Fernández, S., editors, *Product-Focused Software Process Improvement*, pages 195–202, Cham. Springer International Publishing.

Amershi, S. et al. (2019). Software engineering for machine learning: A case study. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300.

Baldassarre, M. T., Caivano, D., Fernandez Nieto, B., Gigante, D., and Ragone, A. (2023). The social impact of generative ai: An analysis on chatgpt. In *Proceedings of the 2023 ACM Conference on Information Technology for Social Good*, pages 363–373.

Banh, L. and Strobel, G. (2023). Generative artificial intelligence. *Electronic Markets*, 33(1):63.

[3]PyTorch. https://pytorch.org/

Bommasani, R. et al. (2022). On the opportunities and risks of foundation models.

DeepSeek-AI (2024). Deepseek-V3 Technical report.

Díaz, V. G., Espada, J. P., García-Bustelo, B. C. P., and Lovelle, J. M. C. (2015). Towards a standard-based domain-specific platform to solve machine learning-based problems. *Int. J. Interact. Multim. Artif. Intell.*, 3:6–12.

Foidl, H., Felderer, M., and Biffl, S. (2019). Technical debt in data-intensive software systems.

Gatto, N., Kusmenko, E., and Rumpe, B. (2019). Modeling deep reinforcement learning based architectures for cyber-physical systems. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, pages 196–202.

Giachetti, G., Catalá, D., de Miguel, B., Carrascosa, C., de Miguel, M., and Pastor, O. (2023). Music360: Modeling the value of music. In *CAiSE Research Projects Exhibition*, pages 105–110.

Google-Cloud (2025). Ai platform: Machine learning solutions overview. Available: `https://cloud.google.com/ai-platform/docs/ml-solutions-overview` [Accessed: 20-02-2025].

Gozalo-Brizuela, R. and Garrido-Merchan, E. C. (2023). Chatgpt is not all you need. a state of the art review of large generative ai models. *arXiv preprint arXiv:2301.04655*.

Grattafiori, A. et al. (2024). The llama 3 herd of models.

Hartmann, T., Moawad, A., Fouquet, F., and Le Traon, Y. (2019). The next evolution of mde: a seamless integration of machine learning into domain modeling. *Software & Systems Modeling*, 18:1285–1304.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

ISO/IEC 25010:2023 (2023). Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model . Technical report, International Organization for Standardization.

Liu, J., Huang, Q., Xia, X., Shihab, E., Lo, D., and Li, S. (2020). Is using deep learning frameworks free? characterizing technical debt in deep learning frameworks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pages 1–10.

Marín, B., Gallardo, C., Quiroga, D., Giachetti, G., and Serral, E. (2017). Testing of model-driven development applications. *Software Quality Journal*, 25:407–435.

Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M., and Wagner, S. (2022). Software engineering for ai-based systems: A survey. *ACM Trans. Softw. Eng. Methodol.*, 31(2).

Naveed, H., Arora, C., Khalajzadeh, H., Grundy, J., and Haggag, O. (2024). Model driven engineering for machine learning components: A systematic literature review. *Information and Software Technology*, 169:107423.

Object Management Group (2019). Meta Object Facility (MOF) Core Specification, Version 2.5.1. Technical Report formal/2019-10-01, Object Management Group (OMG). Available: https://www.omg.org/spec/MOF/2.5.1/PDF [Accessed 20-02-2025].

OpenAI (2023). GPT-4 Technical Report.

Pastor, O. and Molina, J. C. (2007). *Model-driven architecture in practice: a software production environment based on conceptual modeling*, volume 1. Springer.

Perrault, R. and Clark, J. (2024). Artificial intelligence index report 2024.

Rädler, S., Berardinelli, L., Winter, K., Rahimi, A., and Rinderle-Ma, S. (2024). Bridging mde and ai: a systematic review of domain-specific languages and model-driven practices in ai software systems engineering. *Software and Systems Modeling*, pages 1–25.

Rädler, S., Berardinelli, L., Winter, K., Rahimi, A., and Rinderle-Ma, S. (2024). Bridging mde and ai: a systematic review of domain-specific languages and model-driven practices in ai software systems engineering. *Software and Systems Modeling*.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Shahab, O., El Kurdi, B., Shaukat, A., Nadkarni, G., and Soroush, A. (2024). Large language models: a primer and gastroenterology applications. *Therapeutic Advances in Gastroenterology*, 17.

towards data science (2025). DeepSeek V3: A New Contender in AI-Powered Data Science. Available: https://towardsdatascience.com [Accessed 14-02-2025].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). Attention is all you need.

Wan, Z., Wang, X., Liu, C., Alam, S., Zheng, Y., Liu, J., Qu, Z., Yan, S., Zhu, Y., Zhang, Q., Chowdhury, M., and Zhang, M. (2024). Efficient large language models: A survey.

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer.

Zhang, J. M., Harman, M., Ma, L., and Liu, Y. (2019). Machine learning testing: Survey, landscapes and horizons.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2024). A survey of large language models.

Çetiner, G., Yayan, U., and Yazici, A. (2024). Mutation-based white box testing of deep neural networks. *IEEE Access*, 12:160156–160174.