FROM RESEARCH TO PRACTICE: DOES AI PROMOTE OR PREVENT THE USE OF AN MBSE TOOL?

Asma Charfi, Takwa Kochbati and ChokriMraidha

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France

ABSTRACT

In this paper, we will investigate the role that can play the AI in adopting a Model Based System-Engineering (MBSE) tool. The MBSE approach is widely adopted in the development of complex systems (real time systems, cyber physical systems, system of systems, etc.) however, in the practice, the tools implementing this approach are facing several problems and are far from being adopted by system provider. We contend that the integration of AI into the appropriate MBSE phase can yield advantageous outcomes. Furthermore, we propose that further exploration into the implementation of AI techniques (such as Machine Learning and NLP) in MBSE tools is necessary to align with the requirements of stakeholders.

KEYWORDS

MBSE, AI, ALM, NLP, MBSE Tool

1. INTRODUCTION

The INCOSE (International Council on Systems Engineering) defines Systems Engineering as "*an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem*"[1].

Model Based System Engineering (MBSE) has been widely adopted in the development cycle of complex systems. Indeed, MBSE allows managing the complexity of the system by developing models for each stage in the development cycle (Requirement model, Function model, logical and physical models...) as shown in Figure 1 from [2].

This cycle is evolving (from a classical V cycle) to fit the new functional (the capabilities, the missions, the operations) and non-functional requirements (the quality attribute such as carbon footprint, safety, security, performance…) of the complex system (system of systems (SoS), Cyber Physical System (CPS)…). Some studies such as [3] and [4] show the advantage of adopting Model Based System Engineering over Document Based System Engineering. This study [4] over 700 projects in different countries around the world shows that schedule adherence and average project duration are improved by using modelling tools. Only 19% of modelling tool users reported projects being behind schedule, compared to 30% of non-users.

David C. Wyld et al. (Eds): SOEN, SIPP, PDTCA, ITE, CCSIT, NLPCL, DaKM, BIGML, AISC -2023
pp. 21-32, 2023. CS & IT - CSCP 2023 DOI: 10.5121/csit.2023.131202

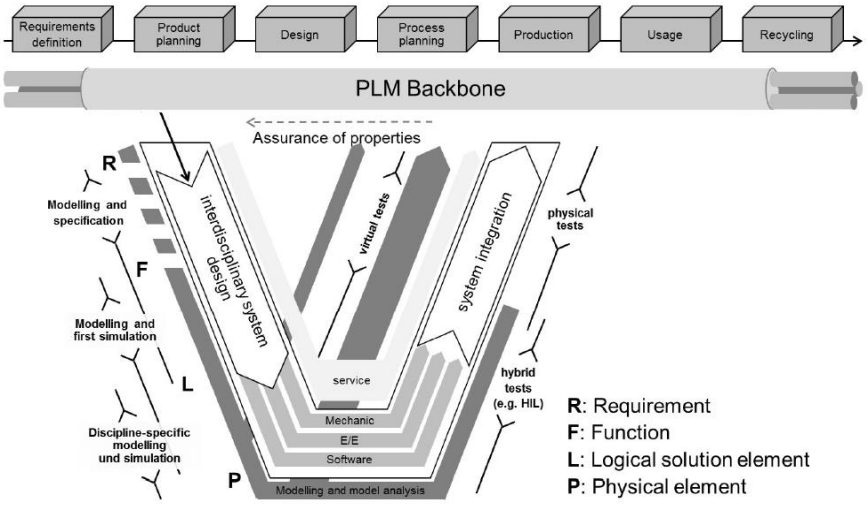

Figure 1. Extended V model for an MBSE approach of a Cyber Physical System

One of the MBSE pillars is to promote an automatic transition between the different stages of the system development process, from requirements specification through design and implementation to validation (Figure 1). Moreover, keeping a traceability link between the different models of the system is one of the pillars of an MBSE approach [1]. This will ensure that the final system will satisfy the functional and non-functional requirements identified at the highest level of the development cycle.

We can easily conclude that providing an MBSE tool to allow companies to take advantages of this promising approach is challenging (which modelling language is used to build the models? how to ensure traceability between models? how to automate the implementation and the test? what about validation and verification?). Some MBSE tools are already used in industries such as [5], [6], [7], etc. However, (1) the lack of powerful automation MBSE tools implementing all the modelling levels and ensuring traceability for all phases of the MBSE approach as well as (2) the social barrier that consider the implementation phase in the development cycle as the centric and more important phase, play a crucial role to hinder the adoption of the MBSE approach.

The discipline of AI can be described as the science of mimicking human mental faculties on a computer [8]. AI systems include modules that enables the generation of types of learning. For instance, Machine Learning (ML) is a type of artificial intelligence technique that makes decisions or predictions based on data. Natural Language Processing (NLP) is the branch of AI, which enables computers to understand, interpret, and manipulate human languages. In fact, AI techniques together with suitable technology have enabled systems to perceive, predict, and act in assisting humans in a wide range of applications.

With the emergence of the AI techniques, we can argue that integrating AI in an MBSE tool will promote the adoption of this approach.

In the next section, we will present the related works on using AI techniques in a system engineering process. The section 3 presents the result of adopting some AI techniques in an MBSE tools as well as our implemented approach. Section 4 presents a use case and section 5 a discussion followed by a conclusion.

2. RELATED WORKS

In [9] the author present the AI augmented system-engineering approach where he tries to map some system engineering activities to AI technologies. For example, machine-learning techniques [10], [11], [12] as well as ontologies and Knowledge bases (Kbs) [13], [14] can be used in the design phase.

In [15] the author propose a recommender system to assist design modelling with UML. The system recommends class attributes when constructing UML class diagrams. Commonly used recommendation techniques include collaborative filtering (CF) and knowledge-based (KB). Although a recommender system can assist the designer and enhance the quality of the produced UML class diagram, it can indeed clutter up the existing Modelling tool that already suffer from a cumbersome multi views IHM (semantic view, graphical view, properties view…).

The work presented in this paper [16] aims at fostering model-driven engineering at enterprise scale, by contributing to a next generation of cognitive model-based engineering tools and this by proposing a generic LSTM neural network architecture to infer heterogeneous model transformations, mainly the code generation from the design. However, despite the limitations already mentioned in the same paper: (the lack of datasets to train the ANN, the ANNs are unable to perform operations with values and evaluation of expressions), we argue that the automatic generation of the code from the system design is not the main functionality that a stakeholder will use when adopting an MBSE tool.

In fact, only structure part of the code is generated from structural models. Behaviours are best coded directly in 3rd generation languages and are not generated from the design. It is easier for the developer and even for designer to write directly the code than building an activity diagram or a state machine diagram and then generate the same code from it. Authors in this paper [17] propose to delete the code generation step in the MBSE approach and directly compile UML models since the code generation prevents some footprint code optimizations that is relevant for constrained-embedded systems.

In this paper [18] an evaluation of NLP techniques for requirements traceability is presented, the authors conclude that NLP is likely not a practical approach to requirements traceability. In fact, for this study, the amount of time needed to resolve the false positives may be greater than the time spent conducting the process without automated tools in the first place. In the case of this study, the number of false positives requiring human review rendered histogram distance counterproductive, involving more work to manually resolve than to manually establish traces without the use of automatic tools.

In the next section, we will present how the use of the NLP techniques to generate Design model from the requirements (first and second step of the V cycle of an MBSE approach) can be beneficial to the MBSE tool user.

3. AI TECHNIQUES TO MANAGE REQUIREMENTS AND DERIVE SYSTEM DESIGN

In the previous section, we presented related works that rely on adding AI techniques in different phases of the MBSE approach to enhance the adoption of this approach using more performant MBSE tools to produce more performant and reliable systems.

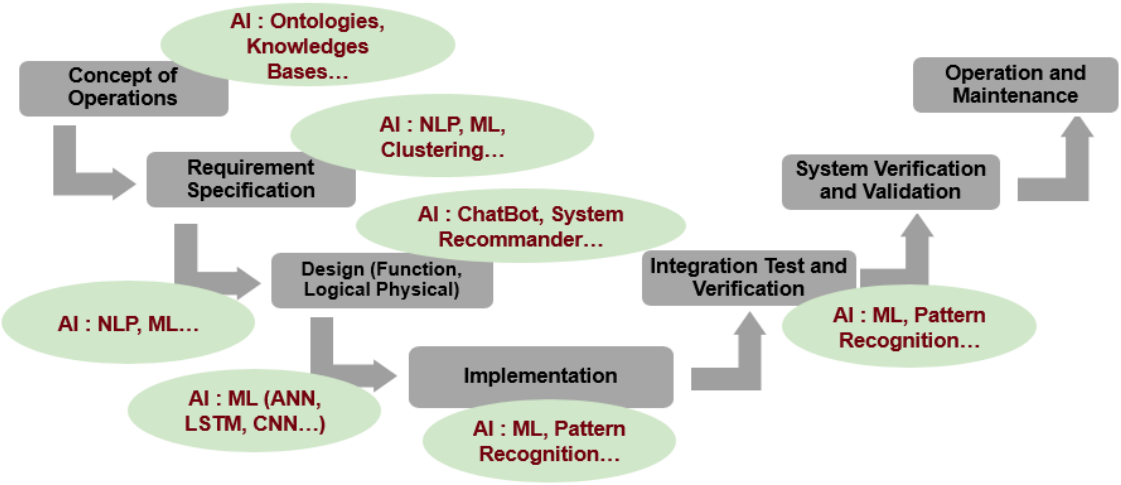

Figure 2. A possible view of an extended MBSE process with IA techniques

Figure 2 presents a classical V cycle of an MBSE approach extended with some AI techniques. We believe according to our experience in using and developing MBSE tools, and taking into account the feedback of the industrial stakeholders about the use of MBSE tools, that embedding the AI techniques (NLP and ML) to manage the requirements and automatically generating the first level design of the system is the most interesting way to enhance the adoption of the MBSE tool.

This section is organized as follow: the first subsection (section 3.1) introduces the approach and its advantages and drawbacks. The second subsection (section 3.2) presents the implementation and some specific NLP and ML techniques.

3.1. Approach definition

The first step in the development cycle of a complex system is the specification of the requirements (Figure 2). However, for complex systems, the number of requirements is exponentially increasing due to the increasing number of stakeholders involved in the process (mainly for CPS). This exponential growth in the number of requirements hinders the efficient management and the understanding of the gathered requirements and consequently, it hinders the construction of high quality systems. Requirements elicitation has a significant impact on information systems quality and success, as the errors introduced at the beginning stages of development are the hardest and most expensive to correct [19].

Hence, the exponential growth of the number of requirements raises difficulties in managing manually the requirements and having a clear view of the expectation and scope of the system to be designed. Here, lies the importance of adopting graphical models as they ensure having a clear view of the expectation and scope of the system to be designed [20]. In fact, adopting models helps to represent and communicate what is important among stakeholders, keep track of the gathered requirement throughout the project, and helps developers deal with the complexity of the solution being developed. Additionally, with the shift to Agile development, requirements are continuously changing, which makes it difficult to capture all requirements for a non-trivial system before development [21]. It is therefore critical to keep track of the requirements throughout the project. Agile software development methods such as Scrum have become

widespread in the industry. Several industrial and open source tools has emerged to manage not only the system development but also all the product lifecycle Management (PLM).

Comparing to MBSE tool, PLM tools have been more successful, because of their user friendly web interface (even if some MBSE tools migrating to a new web interface like [5]) and there capabilities to manage the entire lifecycle of an application (ALM) or a product (PLM).
An important factor that help user adopting an ALM tool is the ability to interact with the system without learning a new language. For MBSE, the user has to be familiar at least with the standards modelling languages like UML [22], SysML [23], etc. For example, for ALM tool that implements the well-known and heavy used Scrum Agile approach, the system requirements are edited in the tool as user stories.

We identified three challenges to be considered in adopting an MBSE approach to develop complex system:

1) How to help system developer manipulating all the complex system requirements usually defined in natural languages?
2) How to help system developer to keep traceability between the requirements and the design of the system?
3) How to help system developer to follow both Agile and MBSE approaches?

Our approach proposes a solution for each identified problem. (1) Rely on the AI techniques (NLP and ML) to manage the complexity and the growing number of requirements. (2) Automatically generate the system design from the classified requirements. (3) Integrate both services in an ALM tool to follow an agile approach and hide the complexity of modelling language for the end user (3). Figure 3 shows the proposed approach.

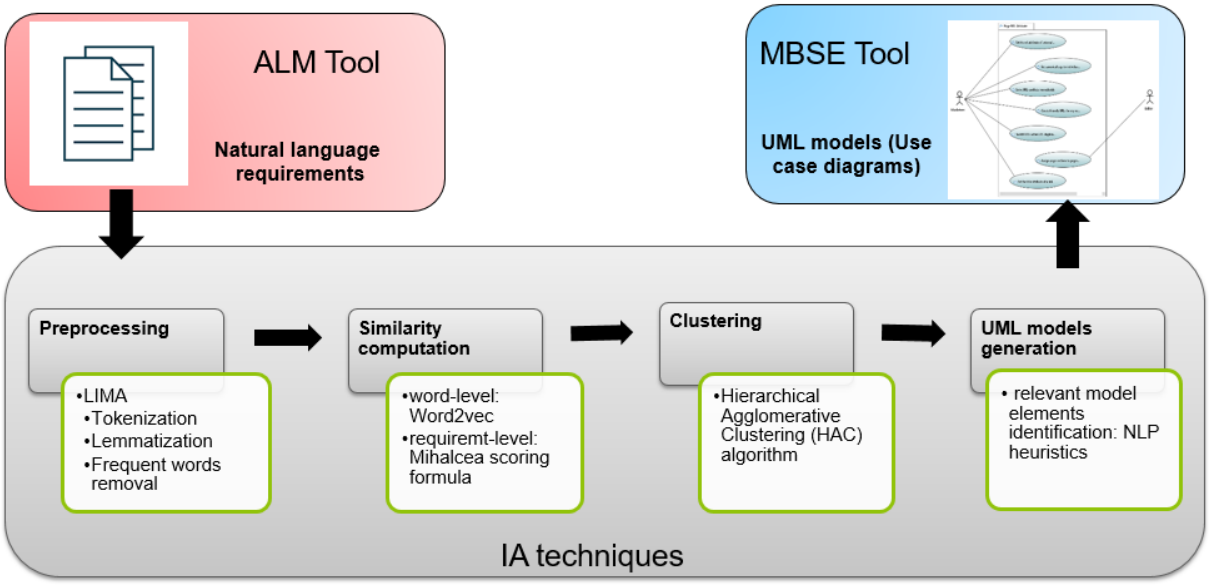

Figure 3. The Proposed Approach

It is worth to say here that the two first solutions proposed by our approach have been already presented in more details in [24]. In this paper, we enhance the approach technically by (1) relying on LIMA NLP tooling [25] instead of the NLTK and SpaCy libraries and (2) integrating the solution in the Papyrus MBSE tool that imports the requirements from the Tuleap ALM tool [26].

3.2. Technical Implementation

Among Agile engineering practices, user stories are widely adopted [27], involving potential stakeholders in the requirement elicitation process by writing their needs in natural language.

A user story is a requirement expressed from the perspective of an end-user. It is a semi-structured natural language description of requirements. The structure of user stories follows a compact template that describes the type of user, what they want and (optionally) why [28]. The most used template is as follow:

As a « type of user », I want « goal », [so that « some reason »]

Several NLP techniques can be applied to tune the user stories. In this paper [29], authors gave a systematic review of NLP techniques used in the processing, similarity computation and clustering of user stories as shown in Table 1.

Table 1. NLP tooling for the treatment of user stories

Tools	Features
SpaCy	Tokenization, Part-of-speech (POS) Tagging, Dependency Parsing, Lemmatization, Similarity.
Stanford CoreNLP	Tokenization, Part-of-speech (POS) Tagging, Lemmatization.
Natural Language ToolKit (NLTK)	Part-of-speech (POS) Tagging
word2vec	semantic arithmetic
WordNet	Semantic similarity
LingPipe Toolkit	end-of-sentence detection
PropBank	semantic propositions
TreeTagger	Part-of-speech (POS) Tagging
Stanford POS tagger	Part-of-speech (POS) Tagging

In our approach, we have used LIMA to pre-process the user stories and extract the main model elements in a UML use case diagram: the Actor, the Use case and the Association between Actor and Use case [22]. In general, LIMA is a fast and efficient NLP tool, especially for languages other than English (LIMA supports more than 60 languages). NLTK is a user-friendly NLP library with a wide range of tools and educational resources. CoreNLP is a scalable NLP library with a server-based architecture. In our previous work, we used NLTK library for the preprocessing step and Spacy for defining the rules to extract UML elements from already clustered requirements.

Preprocessing is the first step of the approach in which, the input functional requirements expressed in natural language are normalized through several steps like tokenization (the decomposition of a sentence into a set of individual words), stop-words removal (the elimination of common words), punctuation removal… Figure 4 shows the result of the preprocessing step using LIMA.

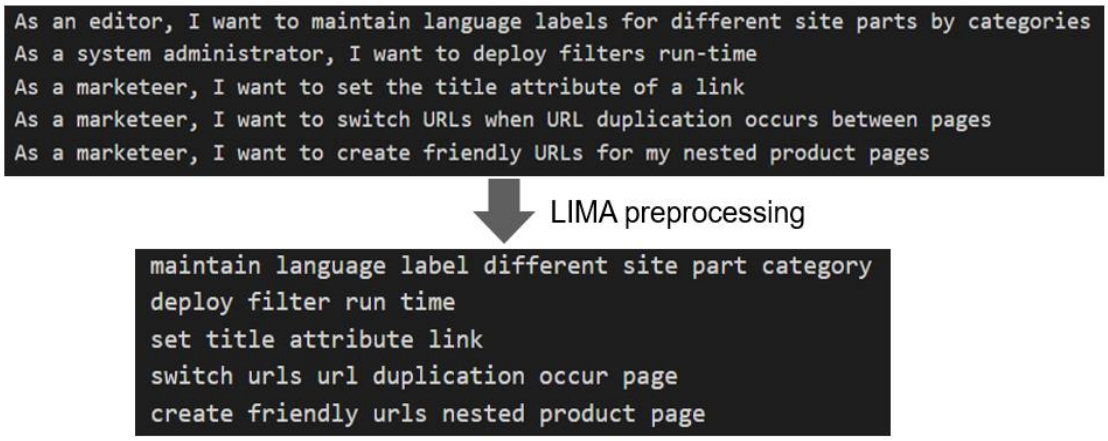

Figure 4. LIMA Preprocessing

In fact LIMA allowed us to enhance easily with a simplified syntax the extraction of an Actor and the Use Case as shown in the Figure 5: (Actor is referred here as REQ_ANNOUNCER and the use case as REQ_GOAL).

With DeepLima [25] the version of LIMA with deep learning techniques, we aim to enhance the result of the preprocessing as well as UML element extraction.

```
@As::(a|an|the) ((^I){1-10}) (,?) (I|i) @Verb_req:REQ_ANNOUNCER:
@So:(,?):((^.){1-n}):REQ_GOAL:
```

Figure 5. Example of LIMA Rule for extracting Actors and Use Cases

For more details about the Similarity Computation and the Clustering steps, please refer to our previous works [24] and [30].

In the next section, we will presents a case study of the approach using the ALM tool Tuleap and the MBSE tool Papyrus.

4. CASE STUDY: TULEAP AND PAPYRUS

Tuleap is an ALM tool that provides a platform for managing software development projects. ALM tools facilitate the entire software development process, from project planning and requirements gathering to coding, testing, and deployment.

Papyrus is an open source MBSE tool to develop complex system; it offers several views to design models according to OMG modelling languages like UML and SysML.

As shown in Figure 3, the idea is to:

- Import or edit the system requirements defined as user stories in a ALM tool (here Tuleap will be used as example)
- Execute several AI techniques to parse and classify the requirements
- Use Papyrus (in a hidden way from the end user) to generate for free a visual graphical representation of the requirements

4.1. Description of the Case Study

We evaluated our approach on the CMS Company case study [24]. This case study involves a company developing complex Content Management System (CMS) products for large enterprises. 34 user stories are supplied. Figure 6 shows a snippet of the user stories of the CMS case study and the UML use case diagram that is automatically generated from this snippet thanks to the AI NLP algorithms.

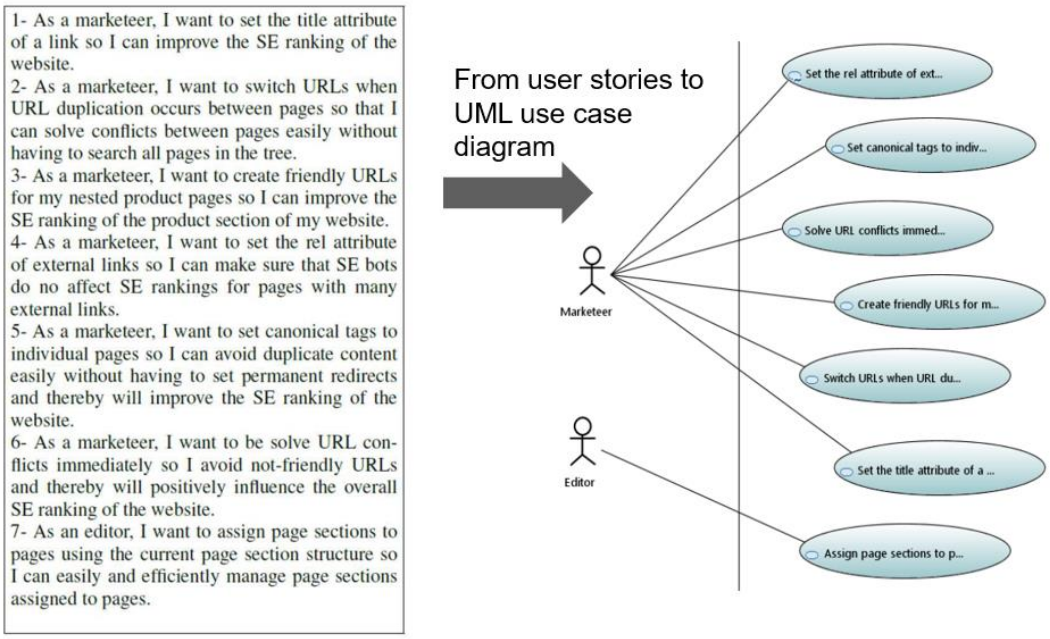

Figure 6. Snippet of CMS company user stories and the UML Use Case diagram generated from it

Each user story is tracked in the Tuleap tool as a new Artifact (the user story text is copied in the Task Title field of each Artifact as shown in the Figure bellow.

Artifact ID	Task title	Status
218725	As a marketeer, I want to set the title attribute of a link so I can improve the SE ranking of the website.	Todo
218726	As a marketeer, I want to switch URLs when URL duplication occurs between pages so that I can solve conflicts between pages easily without having to search all pages in the tree	Todo

Figure 7. Tuleap IHM to store the user stories

All user stories are imported in Papyrus thanks to the available REST API of Tuleap, and then all the embedded AI algorithms are executed (LIMA preprocessing, similarity computation, clustering and UML elements extraction). After that, Papyrus automatically generate the UML use case diagrams from the UML semantic file. The tooling architecture is presented in the Figure 8. The Rust programming language is used to launch all NLP algorithms and generate the UML use case diagram according to each Cluster.

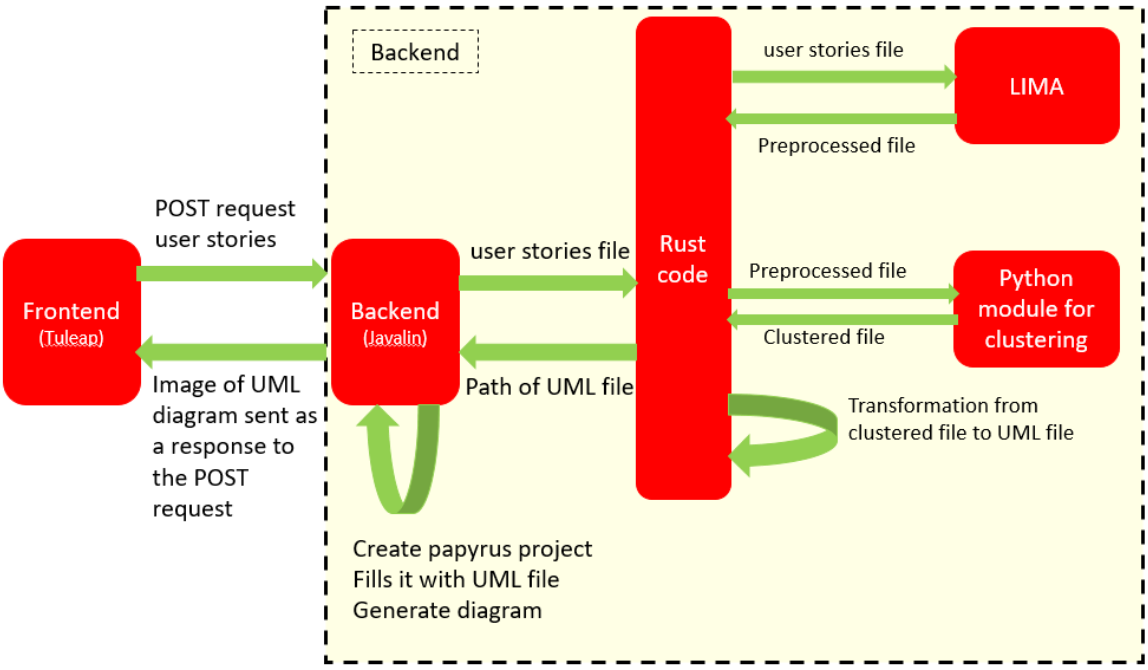

Figure 8. The web interface between Papyrus (back end) and Tuleap (front end)

4.2. Evaluation of the approach

In order to evaluate the accuracy of the generated use case model, we specified True Positive (TP), False Negative (FN) and False Positive (FP) elements applied to actors, use cases and their relationships.

Table 2 summarizes the evaluation of the generated UML use case model in terms of Precision, Recall and F-measure (well-known measures in the Information Retrieval (IR) field) [31].

Table 2. Evaluation of the used IA techniques using Precision, Recall and F-mesure for the UML element extraction as well as execution time of all AI algorithms

	Actors	Use Cases	Relation-ships
	TP FP FN	TP FP FN	TP FP FN
	34 0 0	32 2 2	28 4 4
Precision	100%	94%	88%
Recall	100%	94%	88%
F-measure	100%	94%	88%

Case Study	Phase	Execution Time
CMS Company	preprocessing	5s
	Clustering (similarity computation+HAC +labelling)	28s
	UML use case generation	2s
	Total	35s

For actors, F-measure value is equal to 100% as we succeeded to extract all the actors. For use cases and relationships, F-measure values take high-range (94%) and (88%) respectively. In relationships detection, these values are due in particular to inclusion or extension relationships between use-cases that are not supported by our approach. The execution time of all IA algorithms in this platform is acceptable (35 s for this case study as shown in Table 2). However, it increases with the increase of the number of the user stories.

5. DISCUSSION

We are convinced that AI algorithms are of great help for the MBSE approach if they are used to verify, validate, analyse, and compare models but not to cognify the MBSE tools that already suffer from adoption problems due to the complexity of the modelling language itself and the lack of fluidity of the user interface. Thus, integrating AI in the design phase and the code generation phase (Figure 2) is not the priority to enhance MBSE tool adoption: an automatic code generation is not heavily used by stakeholders since it is usually a partial step (only the structure part of the code is automatically generated).

The presence of AI in some systems can play the opposite role and, contrary to what one might think, AI can create a certain reticence for the designer manipulating a cognified tool (with embedded chatbot or a recommender system).

We have also noticed that ChatGPT (an example of a now day excessively used AI) [32] reveals in some cases a reluctance towards the not always convincing answers of this AI. For example, by asking chatGPT to generate a UML use case diagram from a user story, the AI generate a diagram that does not respect the semantics of the UML language. We much prefer that the AI explain that it has not be trained to answer this kind of question or that it only knows how to answer in text mode. This example shows that we cannot always trust what AI produces. Many works are investigating the ethics and trust in the AI such as [33].

Moreover, if integrating AI solutions in an MBSE process of a complex system is beneficial, this does not mean that running AI software in the tools implementing this process will be also beneficial. In fact, there is usually a huge gap between the theoretical and practical aspect. For example, for real time and embedded systems, the constraints on response times and data confidentiality lead to a preference for local data processing, as close as possible to equipment and sensors, rather than through the Internet. The challenge is not just to embed an AI but also rather to embed an AI that consumes computing power, memory space and energy in a device that is usually constrained in all these aspects and this without degrading performance.

6. CONCLUSION

In this paper, we presented several works for integrating AI techniques to enhance the adoption of the MBSE approach. We argue that integrating AI techniques in the first and second phase of the MBSE lifecycle (requirement specification and design generation) can promote the use of an MBSE approach to develop complex system. Otherwise, relying on a standalone MBSE tool and trying to integrate AI technique in the design phase or the code generation phase can prevent and slow down the adoption of such tool. We believe that integrating an MBSE tooling into an ALM or PLM tool is beneficial for both MBSE and Application Agile approaches.

ACKNOWLEDGEMENT

The authors would like to thank all Papyrus development Team and mainly Sébastien Gérard as well as Eliot Barbot for the work on the internship to develop the REST web service. We thank also the LIMA development Team and mainly Gael De Chalendar for his help.

REFERENCES

[1] J. W. a. Sons, Incose Systems Engineering Handbook V4., INCOSE, 2015.
[2] J. H. a. T. B. I. Graessler, «V-MODELS FOR INTERDISCIPLINARY SYSTEMS ENGINEERING,» INTERNATIONAL DESIGN CONFERENCE - DESIGN, n° %1https://doi.org/10.21278/idc.2018.0333, 2018.
[3] T. T. C. M. S. G. G. Z. Morayo Adedjouma, «From Document-Based to Model-Based System and Software Engineering: Experience Report of a Selective Catalytic Reduction System Development.,» EduSymp/OSS4MDE@MoDELS, pp. 27-36, 2016.
[4] C. Rommel, «MBSE Solutions & Software/System Modeling Tools.,» https://www.vdcresearch.com/vdcc/wp-content/uploads/2020/12/2020-Software-System-Modeling-Tools-EB-VDC.pdf, 2020.
[5] Papyrus. Available: https://www.eclipse.org/papyrus/
[6] Capella. Available: hthttps://www.eclipse.org/capella/
[7] Cameo. Available: https://www.3ds.com/products-services/catia/products/no-magic/cameo-systems-modeler/
[8] A. Hopgood, The state of artificial intelligence., Adv. Comput., 65:3–77, 2005., 005.
[9] A. M. Madni, Exploiting augmented intelligence in systems engineering and engineered systems. Insight, pp 31–36, INSIGHT, 2020.
[10] Z. B. Y. Y. Y. e. a. Yang, Exploiting augmented intelligence in the modeling of safety-critical autonomous systems., Form Asp Comp 33, 343–384. https://doi.org/10.1007/s00165-021-00543-6, 2021.
[11] P. T. E. H. D. A. &. M. Rajendran, Human-in-the-loop Learning for Safe Exploration through Anomaly Prediction and Intervention., SafeAI@AAAI, volume 3087, of CEUR Workshop Proceedings, 2022.
[12] F. E. H. R. A. e. a. ARNEZ, Towards Dependable Autonomous Systems Based on Bayesian Deep Learning Components, 18th European Dependable Computing Conference (EDCC). IEEE, 2022. p. 65-72., 2022.
[13] F. N. a. C. M. Luis Palacios Medinacelli, Augmenting model-based systems engineering with knowledge., In Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, 2022.
[14] M. B. B. K. I. G. M. B. Thomas Hagedorn, Knowledge Representation with Ontologies and Semantic Web Technologies to Promote Augmented and Artificial Intelligence in Systems Engineering, Volume23, Issue1 March 2020.
[15] M. Savary-Leblanc, Augmenting software engineers withmodeling assistants, Phd Thesis , 2021.
[16] L. B. •. J. C. •. S. L. •. S. Gérard, «A generic LSTM neural network architecture to infer heterogeneous model transformations,» Software and Systems Modeling (2022) 21:139–156.
[17] C. M. P. B. Asma Charfi, «An Optimized Compilation of UML State Machines,» 2012 15th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing.
[18] R. E. G. a. M. K. C. D. Laliberte, «Evaluation of Natural Language Processing for Requirements Traceability,» 2022 17th Annual System of Systems Engineering Conference (SOSE), , Vols. %1 sur %2Rochester, NY, USA, 2022, pp. 21-26, doi: 10.1109/SOSE55472.2022.9.
[19] B. Brügge and A. Dutoit, «Object-Oriented Software Engineering Using UML, Patterns, and Java,» 2009.
[20] e. a. Rodi Jolak • Maxime Savary-Leblanc • Manuela Dalibor, «Software engineering whispers: The effect of textual,» Empirical Software Engineering (2020) 25:4427–4471.
[21] P. Krutchen., « The rational unified process: An introduction. 2000.».
[22] O. U. M. L. (. U. OMG, «https://www.omg.org/spec/UML/,» V 2.5.1, 2017.
[23] OMG, «https://www.omgsysml.org/SysML-2.htm».

[24] T. K. p. Thesis, Bridging the gap between natural language system requirements and architecture design models. Software Engineering, niversité Paris-Saclay, 2021. English. NNT: 2021UP ASG076ff. tel-03411393,, 2 November 2021.
[25] O. S. N. &. B. L. ZENNAKI, «A neural approach for inducing multilingual resources and natural language processing tools for low-resource languages.,» vol. Natural Language Engineering, n° %1, 25(1), 43-67. doi:10.1017/S1351324918000293.
[26] Tuleap. [En ligne]. Available: https://www.tuleap.org/.
[27] J. T. a. M. J. E. C. E.-M. Schön, Agile requirements engineering: A systematic literature review., Comput. Stand. Interfaces, 49:79–91, 2017.
[28] S. H. M. K. a. I. M. Y. Wautelet, «Unifying and extending user story models,» EMNLP, 2014.
[29] I. K. a. S. D. a. F. C. Raharjana, «User Stories and Natural Language Processing: A Systematic Literature Review,» IEEE ACCESS, 2021.
[30] S. G. S. L. C. M. Takwa Kochbati, «From word embeddings to text similarities for improved semantic clustering of functional requirements,» SEKE, 2021.
[31] P. R. a. H. S. C. D. Manning, «Introduction to information retrieval,» 2005.
[32] ChatGPT. [En ligne]. Available: https://openai.com/blog/chatgpt.
[33] M. Ryan, «In AI We Trust: Ethics, Artificial Intelligence, and Reliability,» Sci Eng Ethics 26, 2749–2767 (2020). https://doi.org/10.1007/s11948-020-00228-y.

AUTHORS

Asma Charfi is a project manager and researcher at the Embedded and Autonomous Systems Design Laboratory of the CEA LIST in France. She obtained a PhD in computer science, on the topic of UML model compilation and execution. Her current research is related to model-based software engineering for cyber physical systems she is involved in several industrial and European collaborative R&D projects. She is a committer of the open-source Eclipse Papyrus project.

Takwa Kochbati is a PhD student at CEA LIST and SOM Research Lab. Her PhD research focuses on the cognification of model-driven engineering. By using AI techniques. She graduated in computer science engineering from the National School of Computer Science (ENSI, Tunisia) in 2017

Chokri Mraidha is leading the Embedded and Autonomous Systems Design Laboratory of the CEA LIST institute in France. He got a master degree in distributed computing in 2001 and a PhD in Computer Science in 2005. He is involved in UML-based OMG standards for design of real systems like SysML and MARTE and the AUTOSAR standard for automotive. He is working in European and French research projectsdeveloping model-based approaches for design and verification of architectures for critical real-time systems for automotive, railway, aerospace, and robotics. His research and development interests include methods, design principles and smart tools for the engineering of efficient and trustworthy software for autonomous systems.