A review on the infrastructure and tool support for Model-Driven Engineering in the automation of the petrochemical industry*

Carlos Eduardo Xavier Correia [1], **Leandro Buss Becker** [1], **Fabio Paulo Basso** [2]

[1]Federal University of Santa Catarina (UFSC)
Florianópolis - SC - Brazil

[2]Federal University of Pampa (UNIPAMPA)
Alegrete - RS - Brazil

carloscorreia0002@gmail.com, leandro.becker@ufsc.br, fabiobasso@unipampa.edu.br

Abstract. *Automation is a fundamental part of the oil industry, responsible for ensuring productivity and process safety. The application of control systems is carried out through different software applications, which are often unable to exchange data (models) within each other. This is a significant limitation for the overall design process. To address this issue, the use of the Model-Driven Engineering for Petrochemical Industry Automation (M4PIA) platform is proposed. M4PIA is capable of accommodating the software applications Automated Procedures Module (MPA) and Environment for Modeling, Simulation, and Operation (EMSO), both of which are used by the Brazilian state-owned company Petrobras.*

1. Introduction

As pointed out in [Frohm et al. 2006], many companies are betting on automation as a differentiator to remain competitive in the market. They seek to eliminate their productive deficiencies through the automation of their processes, whether by reducing production costs, increasing efficiency, or expanding production.

It is not different within the oil and gas industry. All processes have strong ties to automation systems, which are essential for maintaining high production levels, quality, and process safety [Klatt and Marquardt 2009]. By working to keep the system operating at its optimal point, these control systems not only provide the mentioned benefits, but also have the ability to anticipate issues that could lead to process shutdowns [Seborg et al. 2010].

Additionally, the implementation of control systems has brought significant complexity to the processes. Supervision, control, and actuation systems have been developed, all with the aim of operating in real-time to ensure a high degree of reliability. In this sense, several software programs have been developed to perform these tasks, whether through simulation or process supervision. Therefore, considering that multiple software applications are typically used in automating an industrial plant, there is a need to model each of the platforms used, at least once [Damo et al. 2019].

*Authors thank the funding agencies CAPES and CNPq. Authors at UFSC are also supported by the PRH-ANP/MCT nº 2.1 program.

In the petroleum industry there is a high number of processes and software applications being used. As there is rarely interoperability between these software applications, it becomes necessary to create new models for each program in which the plant is operated. Consequently, this not only leads to a waste of time but also results in high operational costs to maintain a consistent level of reliability in the plant's operations. Thus, it becomes evident that the oil and gas industry faces significant challenges regarding the modeling of its processes.

Therefore, in an effort to mitigate the aforementioned challenges, we observed that model-driven approaches, such as Model-Driven Engineering (MDE), are not only gaining popularity but also proving to be highly suitable for the development of industrial applications [Vepsäläinen et al. 2010]. By adopting such approaches, it becomes possible not only to address inherent deficiencies in the industry, but also to incorporate or enhance important characteristics for this sector.

In this context, the present work aims to present the recent results from our research group related with using MDE to design systems for the petroleum industry. In [Damo 2019], the M4PIA metamodel was developed, allowing to perform direct engineering procedures. In the work [Cruz 2021], using the M4PIA metamodel, a reverse engineering strategy is developed. Finally, in [de Souza Moura 2022], a Domain-Specific Language (DSL) is proposed to assist in modeling the M4PIA models.

The rest of the paper is structured as follows: Section 2 presents the background necessary for understanding the work, as well as related studies. Section 3 explains the development of the M4PIA platform. Section 4 reports on the procedure and results of the proposed DSL. Finally, Section 5 presents the conclusions drawn from the study, as well as future perspectives for the work.

2. Background

In this section, we will provide the reader with an understanding of the methodologies and tools used in the development of the studied works. Additionally, it also presents some related works to aid in contextualizing the discussed subject.

2.1. Model-Driven Engineering (MDE)

Unlike conventional software development methods, Model-Driven Engineering (MDE) is a methodology where models, rather than programs, are the primary results of the development process [Schmidt 2006]. A model can be understood as an abstraction, representing a simplification of the context. In the context of MDE, system modeling involves developing models that describe the characteristics or behavior of the system [Sommerville 2011].

By applying the concepts of MDE, the development process can be solely focused on the objective related to the system, without being concerned about the peculiarities and complexities of the program that will execute it [Schmidt 2006]. In this sense, MDE can provide a development approach with lower risk of errors, increased productivity, and reusability of existing artifacts [Sommerville 2011].

Models with different levels of abstractions are related through transformations. In the Model-Model (M2M) transformation, it is possible to perform direct or reverse engineering, where we move from a model with a higher level of abstraction to a lower level or from a lower level to a higher level, respectively. Exclusive to direct engineering, the Model-Text (M2T) transformation produces the source code of a specific platform. On the other hand, the Text-Model (T2M) transformation, which comes from reverse engineering, generates the model based on the source code of the specific platform.

2.2. Model-Driven Reverse Engineering (MDRE)

Reverse Engineering (RE) is an approach used for understanding software, being employed to gain the necessary knowledge to perform maintenance, documentation, or reengineering of the system [Rugaber and Stirewalt 2004]. Proven to be highly valuable over the years, RE has assisted many maintenance teams in comprehending the structure in which the software was developed.

The application of model-driven techniques to address reverse engineering problems is known as Model-Driven Reverse Engineering (MDRE). It is defined as the creation of descriptive models to represent the behaviors or specificities of a legacy system [Favre 2004]. According to [Raibulet et al. 2017], the process of MDRE consists of: (i) Converting the legacy software into models without unnecessary loss of information, and; (ii) Using these models to generate the desired output models through transformations.

2.3. Domain-Specific Language (DSL)

A Domain-Specific Language (DSL) refers to a language generated to address a specific purpose, as opposed to general-purpose languages [Mailund 2018]. They provide a precise way to perform tasks and achieve specific objectives within a particular context. According to [van Deursen et al. 2000], the application of DSLs promotes increased productivity, reliability, ease of use, and flexibility.

Another important aspect that DSLs add to the system is the communication among the various departments responsible for the project, as the DSL facilitates the understanding of the software by all parties involved [Otto 2017]. In this sense, it becomes more cohesive, fostering collaboration between teams, and increasing the efficiency of the system development process.

2.4. Automated Processing Module (MPA)

The MPA software [Satuf et al. 2009] was developed by Tecgraf/PUC-Rio upon request and in collaboration with Petrobras. It serves to support the development and execution of automation systems in oil platforms. MPA adopts a graphical language based on flowcharts to define procedures for monitoring, diagnostics, and action within the plants. Initially, it was developed to assist in the commissioning of oil extraction platforms at the company's research center (Cenpes) [Guisasola and Maia 2009]. However, currently, it is used in various sectors such as level control in vessels and anti-roll protection.

MPA is composed of an execution server and a configuration and supervision application. In the application, industrial plants are modeled using objects configured in the LUA [JANEIRO 1993] programming language, and the diagrams to be executed are created. The server is responsible for executing the diagrams, communicating with the supervisory system through a specific bridge. In this way, MPA can not only monitor, but also act on the various variables that make up the plant.

2.5. Environment for Modeling, Simulation and Operation (EMSO)

EMSO [Soares and Secchi 2003] is a tool based on algebraic-differential equations. It provides a comprehensive development environment where users can describe equipment, model processes, perform optimizations, simulate, and visualize results. Currently, it is maintained by an alliance of national universities and petrochemical companies.

The EMSO contains three main elements: models, devices, and flowcharts. Models are used to describe the devices mathematically. Devices are instances of the models, used to represent the actual equipment. The flowchart pertains to the process being analyzed, depicting the interaction between the various devices present in the plant.

2.6. Related Works

In [Teixeira et al. 2020], a problem related to the different tools used in MDE is identified. The authors demonstrate that the significant diversity of tools, each with its specific artifact, produces results that affect the interoperability of systems. In the context of MDE, such a problem poses a considerable risk to the methodology, as it is commonly used due to the benefits provided by artifact reuse, such as increased productivity. To address this issue, they conducted research to categorize the different tools in MDE. As a result, they presented nine different tools that use uncommon properties compared to the ones typically applied in MDE.

Software test automation is one of the most challenging activities in Software Engineering. In this regard, [Lima et al. 2021] proposed the use of the DSL Teasy, enabling the application of Model-Based Testing (MBT) to web applications. MBT is a methodology for conducting software tests in which behaviors are described and validated through models. As a result of a real application, Teasy was able to detect 78.57% of the functional non-conformities in the tested system.

3. MDE Approach

The MDE approach proposed by [Damo et al. 2019, Cruz 2021] is composed of the following components: (i) domain-specific languages to represent each level of abstraction of the system; (ii) four sets of model transformations to ensure automatic integration between the models; (iii) two sets of code generators; and (iv) two sets of model generators. The entire infrastructure was developed in the Eclipse environment, using the Eclipse Modeling Framework (EMF) [Steinberg et al. 2008] for metamodeling, the QVTo framework [Foundation 2023b] for performing model transformations, the Acceleo engine [Foundation 2023a] for code generation, and the Antlr framework [Parr 2023] for obtaining models from source code.

3.1. M4PIA Infrastructure

In model-driven engineering, higher-level abstraction models are chosen to initiate the modeling stages because they can more accurately and easily describe the processes. Thus, [Damo et al. 2019] proposed the use of two levels of abstraction: Platform Independent Model (PIM) and Platform Specific Model (PSM). The PIM models are responsible for describing the technical details of the systems. On the other hand, the PSM models describe the behaviors provided by the PIM models in the context of a specific software or program. Through model refinement, PIM models become PSM models due to the defined transformations.

Described in detail in [Damo et al. 2019], the M4PIA infrastructure was developed with the aim of assisting MDE in applications for simulation, control, and supervision of petrochemical platforms. The M4PIA metamodel (PIM metamodel) is the main element responsible for representing the entire platform-independent application. Additionally, the MPA and EMSO metamodels were developed, both being PSM metamodels, to represent the software discussed in Sections 2.4 and 2.5, respectively.

The M4PIA metamodel, shown in Figure 1, is a class diagram developed using EMF and its Ecore metamodel. It expresses the structure and requirements of the process through classes, interfaces, and relationships between different elements. Below is a brief description of the metamodel.

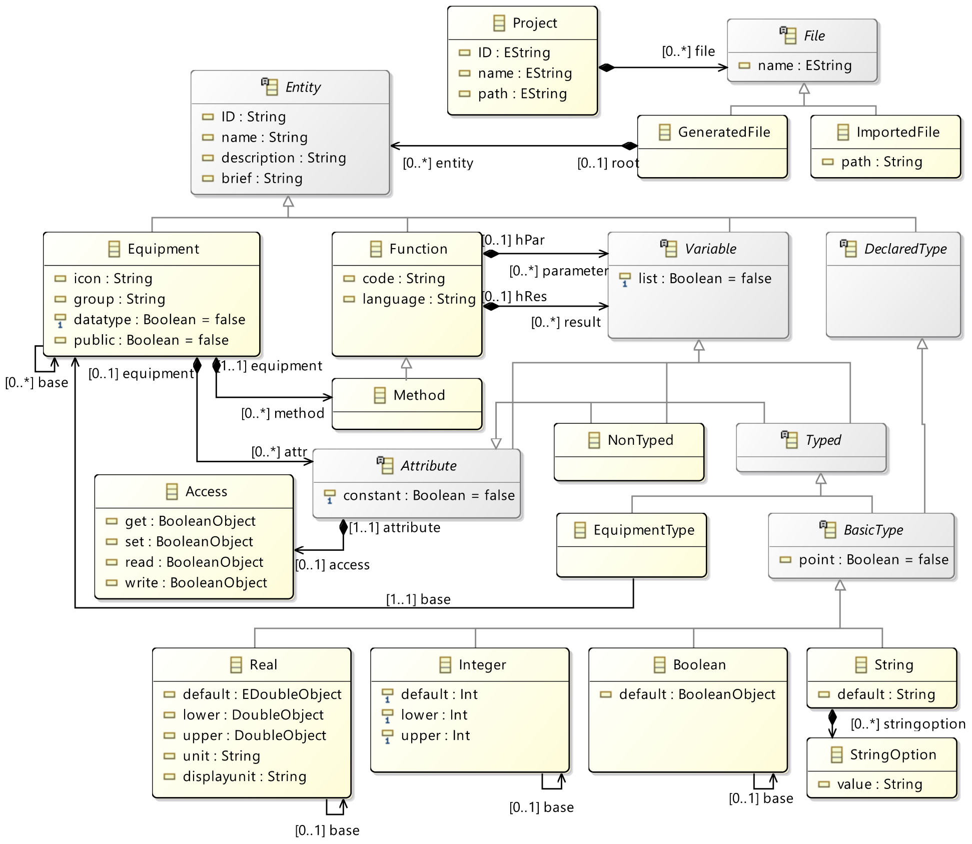

Figure 1. M4PIA metamodel [Damo et al. 2019].

The class *Project* stores and relates the different models present in the development. It can be composed of *Files* of two types: *ImportedFile* or *GeneratedFile*. The first type corresponds to files imported as libraries, while the second type represents groups of entities generated by the infrastructure itself.

Promoting the basic structure to more specialized classes, the *Entity* class is at the top of the modeling hierarchy. For example, the *Equipment* class is of type entity and contains attributes and methods, defined to represent the characteristics and dynamics of the equipment present in the platform.

The *Function* class is used to represent an operation and can contain a language and code. It is also associated with the *Variable* class, which can act as either a parameter or a result of the function. It can be further specialized, such as the *Method* class, which is defined to have an exclusive connection with *Equipment*.

The *Variable* class represents a logical variable that can be of type *NonTyped* or *Typed*. Typed variables can be of *EquipmentType* or *BasicType*, such as *Real*, *Integer*, or *Boolean*. The *Attribute* class is a specialization of the *Variable* class and is related to the *Access* class to define the read/write operations of objects.

Finally, the M4PIA metamodel allows recursion through the *Attribute* class. This means that it allows an equipment or function to be part of another equipment as attributes, for example, creating a hierarchical relationship between elements in the model.

3.2. Transformations

Originally, [Damo 2019] developed the M4PIA infrastructure, allowing only direct engineering operations, starting from the M4PIA PIM model to the PSM model (MPA or EMSO) and subsequently to the source code. In order to complement the tool, [Cruz 2021] developed the set of transformations necessary to perform reverse engineering, generating the PSM model from the source code and then the M4PIA PIM model. This addition to the infrastructure enables the tool to perform both forward and reverse engineering processes, enhancing its capabilities and versatility.

Currently, the tool has eight transformations, including: (i) four M2M transformations; (ii) two M2T transformations; and (iii) two T2M transformations. Through these transformations, it is possible to obtain the source code of the EMSO software starting from the source code of the MPA software. The process involves first going from the MPA source code to the MPA PSM model, then to the M4PIA PIM model, followed by the EMSO PSM model, and finally to the *.emso* extension file. The described procedure can be followed with the Figure 2.

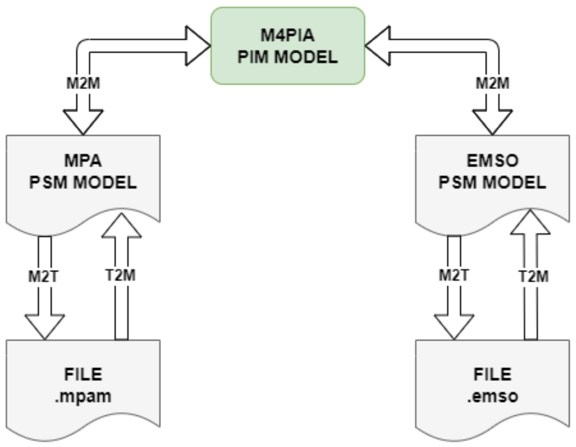

Figure 2. Set of supported transformations. [Damo 2019, Cruz 2021].

Through the presented transformations, the M4PIA infrastructure enabled the reuse of projects produced in MPA or EMSO. In other words, it made it possible to automatically convert a project created in MPA to EMSO or vice versa. This provided interoperability between different software used in the industry. Therefore, a project that would previously start from scratch will now have a more advanced starting point, thereby optimizing the overall system performance.

4. Graphical DSL

The graphical DSL proposed by [de Souza Moura 2022] consists of two different levels of abstraction. This decision was based on feedback from experts in the petrochemical industry who highlighted the need for different visualization modes of the model, one for development and another for maintenance and contextualization purposes. Thus, the Deployment Diagram (DD) and the Conceptual Diagram (CD) were developed. The entire structure of the DSL was implemented using the Sirius framework [Foundation 2023c] present in Eclipse.

4.1. Conceptual Diagram

The Conceptual Diagram was developed with the purpose of providing context to the user about the model process. It aids in model maintenance by offering transparent information, presenting attributes and methods used. The Conceptual Diagram provides a clear view of the process and how the equipment is interconnected, making it easier not only to understand each individual equipment but also the entire process. Figure 3 illustrates a system modeled from the Conceptual Diagram perspective.

In the Conceptual Diagram, equipment representation is structured using a parent **container**, which contains three child containers arranged in a vertical stack. These child containers represent the compartments for basic attributes, equipment attributes, and methods. Each of these compartments has child nodes presented in lists, where specific properties are defined, including the domain class and the expressions of fundamental semantic candidates for accurate selection of these elements.

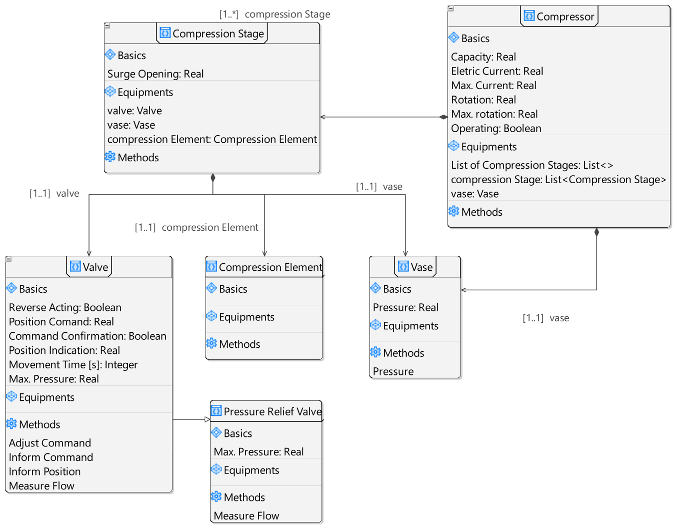

Figure 3. Conceptual modeling view [de Souza Moura 2022].

4.2. Deployment Diagram

The Deployment Diagram was developed solely with the purpose of implementing the plant model. It accelerates the modeling process by providing a ready-made structure for creating new objects in the M4PIA metamodel. The Deployment Diagram presents only the components and their relationships, omitting the remaining information to allow the developer to focus entirely on the process implementation. Figure 4 depicts a model from the perspective of the Deployment Diagram.

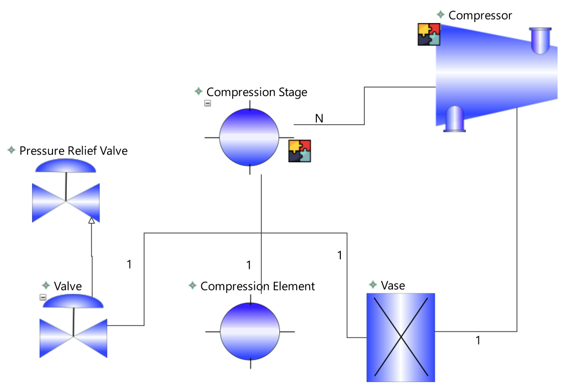

Figure 4. Deployment modeling view [de Souza Moura 2022].

In the Deployment Diagram, equipments are represented by a **node** using a default 3D cube format with spatial notation. A child artifact is created as a *puzzle*-shaped border node to indicate composition by other equipment, rendered only when the equipment is composed of others. Additionally, the display characteristic of equipment can be modified based on the textual attribute *icon* of the candidates themselves, obtained from applying the filter by equipment type attributes using semantic candidates.

The relationships in both diagrams are represented by **edges**. However, they differ in the definition of the arrowhead decorator style. The Deployment Diagram does not use a decorator for its edges, whereas the Conceptual Diagram uses a white triangular arrowhead to represent inheritance and a black diamond arrowhead to indicate composition, similar to the UML class diagram.

5. Conclusions and Future Works

The purpose of this work was to conduct a review of the papers [Damo 2019, Cruz 2021, de Souza Moura 2022], presenting everything from the basic concepts for understanding to the development of the proposed solutions. It provided the opportunity not only to validate the feasibility of the proposed MDE and DSL approaches, but also to identify the deficiencies and issues that need to be addressed.

To evaluate the proposed solutions in [Damo 2019, Cruz 2021], a comparative analysis of the performance was conducted between the process utilizing the tool and the process not employing it. In [de Souza Moura 2022], a quasi-experiment was carried out involving experienced professionals from the oil & gas sector. Thus, based on the development of this work and the discussed evaluation sections, it is evident that the obtained results are satisfactory and that the developed tool is well-received by stakeholders. The M4PIA infrastructure enabled interoperability between the MPA and EMSO software, optimizing plant modeling in the petroleum industry. The DSL, in turn, facilitated the use of the M4PIA tool.

However, some issues are noted regarding the results of the works. Concerning the

M4PIA tool, there is a lack of integration between the developments of [Damo 2019] and [Cruz 2021] since they were proposed in separate development environments. As a result, even though the tool exists, it cannot be fully utilized due to the lack of integration. As for the DSL, although it facilitates the modeling of M4PIA models, there is still a need to model each equipment from scratch for every new project. Considering that the petroleum industry has a set of widely-used equipment, something could be done to mitigate this and further streamline the modeling process.

As future work, the development of solutions for the aforementioned issues is proposed. For integration, the use of the same development environment with the assistance of plugins to ensure the proper functioning of the M4PIA infrastructure. Additionally, for the DSL, the creation of a library within the DSL itself containing the most common equipment used in the oil industry is suggested. This would not only make the tool usable, but also enhance its capabilities, making it even more powerful and efficient.

References

Cruz, M. V. S. (2021). Engenharia reversa baseada em modelos para aplicações de simulação, controle e operação de plantas na indústria petroquímica. Master's thesis, Federal University of Santa Catarina.

Damo, T. P. (2019). Engenharia baseada em modelos para aplicações de simulação, controle e operação de plantas na indústria petroquímica. Master's thesis, Federal University of Santa Catarina.

Damo, T. P., Becker, L. B., and Basso, F. P. (2019). Model-Driven Engineering Infrastructure and Tool Support for Petrochemical Industry Automation. *Advances in Science, Technology and Engineering Systems Journal*, 4(4):174–187.

de Souza Moura, J. (2022). Uma dsl gráfica de suporte para a infraestrutura m4pia. Tcc (graduation), Federal University of Pampa.

Favre, J.-M. (2004). Foundations of model (driven) (reverse) engineering : Models - episode i: Stories of the fidus papyrus and of the solarus. In *Language Engineering for Model-Driven Software Development*.

Foundation, E. (2023a). Acceleo. Available at: `https://eclipse.dev/acceleo/`. Accessed on: 16 July 2023.

Foundation, E. (2023b). Eclipse qvt operational. Available at: `https://projects.eclipse.org/projects/modeling.mmt.qvt-oml`. Accessed on: 16 July 2023.

Foundation, E. (2023c). Sirius. Available at: `https://eclipse.dev/sirius`. Accessed on: 18 July 2023.

Frohm, J., Lindström, V., Winroth, M., and Stahre, J. (2006). The industry's view on automation in manufacturing. *IFAC Proceedings Volumes*, 39(4):453–458. 9th IFAC Symposium on Automated Systems Based on Human Skill and Knowledge.

Guisasola, T. and Maia, R. (2009). Mpa, um sistema de controle de plantas industriais. In *Lua Workshop 2009*, Rio de Janeiro, RJ, Brasil. Available at: `https://www.lua.org/wshop09/mpa.pdf`. Accessed on: 15 July 2023.

JANEIRO, P. U. C. D. R. D. (1993). The programming language lua. Available at: `https://www.lua.org/`. Accessed on: 15 July 2023.

Klatt, K.-U. and Marquardt, W. (2009). Perspectives for process systems engineering—personal views from academia and industry. *Computers Chemical Engineering*, 33(3):536–550. Selected Papers from the 17th European Symposium on Computer Aided Process Engineering held in Bucharest, Romania, May 2007.

Lima, Y., Rodrigues, E., Basso, F., and Oliveira, R. (2021). Teasy: A domain-specific language to reduce time and facilitate the creation of tests in web applications. In *Anais do III Workshop em Modelagem e Simulação de Sistemas Intensivos em Software*, pages 40–49, Porto Alegre, RS, Brasil. SBC.

Mailund, T. (2018). *Domain-Specific Languages in R: Advanced Statistical Programming*. Apress, USA, 1st edition.

Otto, L. (2017). *DSL: Quebre a barreira entre desenvolvimento e negócios*. Casa do Código.

Parr, T. (2023). Antlr. Available at: `https://www.antlr.org/`. Accessed on: 16 July 2023.

Raibulet, C., Arcelli Fontana, F., and Zanoni, M. (2017). Model-driven reverse engineering approaches: A systematic literature review. *IEEE Access*, 5:14516–14542.

Rugaber, S. and Stirewalt, K. (2004). Model-driven reverse engineering. *IEEE Softw.*, 21(4):45–53.

Satuf, E., Pinto, S. F., and Dias, B. Q. (2009). Automatic alignment system for pra-1 pumping platform (in portuguese). In *5th Rio Automation Congress*.

Schmidt, D. (2006). Guest editor's introduction: Model-driven engineering. *Computer*, 39(2):25–31.

Seborg, D. E., Mellichamp, D. A., and Edgar, T. F. (2010). *Process Dynamics and Control*. John Wiley & Sons.

Soares, R. d. P. and Secchi, A. (2003). EMSO: A new environment for modelling, simulation and optimisation. In *Computer Aided Chemical Engineering*, volume 14, pages 947–952. Elsevier.

Sommerville, I. (2011). *Software Engineering*. Pearson.

Steinberg, D., Budinsky, F., Paternostro, M., and Merks, E. (2008). *EMF: Eclipse Modeling Framework*. ddison-Wesley Professional.

Teixeira, P., Lebtag, B., and Basso, F. (2020). Diversity of mde toolboxes and their uncommon properties. In *Anais do II Workshop em Modelagem e Simulação de Sistemas Intensivos em Software*, pages 1–10, Porto Alegre, RS, Brasil. SBC.

van Deursen, A., Klint, P., and Visser, J. (2000). Domain-specific languages: An annotated bibliography. *SIGPLAN Not.*, 35(6):26–36.

Vepsäläinen, T., Sierla, S., Peltola, J., and Kuikka, S. (2010). Assessing the industrial applicability and adoption potential of the aukoton model driven control application engineering approach. In *2010 8th IEEE International Conference on Industrial Informatics*, pages 883–889.