Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Round-Trip Engineering for Petrochemical Industry Automation

Marcus Vinícius Silva Cruz * **Thaise Poerschke Damo** *
Leandro Buss Becker *

* *Automation and Control Systems Department, Federal University of Santa Catarina, Florianópolis, Brazil*
e-mail: marcus.v.s.c13@gmail.com, thaisedamo@gmail.com, leandro.becker@ufsc.br

Abstract: The petrochemical industry is becoming increasingly complex and, as a result, different software platforms are used to assist in system design. Generally, it happens that the same component of the physical plant is (re)modeled on different software platforms, so the reuse of these models becomes difficult, compromises system interoperability, and generates the need for rework in projects. Previous work proposed the infrastructure named M4PIA (Model-Driven Engineering for Petrochemical Industry Automation), which allows to represent industrial plants through different, compatible and object-oriented models. By means of model transformations, it supports automatic code generation from a high-level abstraction model to specific software platforms. The present work provides the integration of Round-Trip Engineering (RTE) in the M4PIA infrastructure so that from a platform-specific model it allows to obtain, through model-to-model (M2M) transformation, a platform-independent model. In order to validate the implemented RTE transformations, it was developed a case study related to a simplified gas compression system.

Keywords: Model-Driven Engineering, Round-trip Engineering, Model Transformations, Petrochemical Industry.

1. INTRODUCTION

Automation systems are of utmost importance for the Petrochemical industry to maintain quality, high production volume, plant safety and profitability (Klatt and Marquardt, 2009). The related control systems are in charge to maintain optimum and stable operation, anticipating to problems that might cause operation shutdown (Seborg et al., 2010).

Typically there exists several distinct software platforms involved for designing and deploying an automation plant. As highlighted in Damo et al. (2019), an industrial plant needs to be modeled several times, at least once for each platform that will operate on it. Developers should be aware of the characteristics/behavior of each process and equipment, to tailor their design according to the target platform. Thus, the platform specialist also needs to be a domain expert, with knowledge of the components to be modeled within each application. On each operating platform, there are plenty of equipment with individual control points, instances, process variables, and process attributes.

Therefore, it is possible to claim that there are advantages on designing a high-level model for representing the plant to be automated, specially when model transformations are used. Such practice is aligned with the principles of Model-Driven Engineering (MDE) that, according to Schmidt (2006), is a software development approach with emphasis on domain specification (high-level) models, contributing to improve design productivity, system understanding, serviceability, and evolution. Such high-level model, named Platform-Independent Model (PIM), goes through one or more model-to-model (M2M) transformations to constitute a Platform-Specific Model (PSM), which is further transformed into source code by using model-to-text (M2T) transformation.

Within this context, it was presented in Damo et al. (2019) a MDE infrastructure for the development of simulations, plant operation/supervision, and control applications of petrochemical industrial plants. Such infrastructure was named M4PIA (Model-Driven Engineering for Petrochemical Industry Automation). It starts from a higher abstraction instance and, through M2M transformations, a PSM model is generated to be used as basis for source-code generation (M2T transformation).

It happens, for instance, that to implement a complete MDE-based development process it is very important to create support for an approach characterized as Round-trip Engineering (RTE) (see Brambilla et al. (2012)). The main characteristic that distinguishes RTE from forward and reverse engineering is the ability to synchronize existing artifacts that evolved concurrently by incrementally updating each artifact to reflect changes made to the other artifacts. Furthermore, forward engineering can be seen as a special instance of RTE in which only the specification is present and reverse engineering can be seen as a special instance of RTE in which only the software is present.

Copyright lies with the authors

The present paper proposes a solution for enhancing the M4PIA infrastructure (see Damo et al. (2019)) for, rather than supporting a limited MDE Development devoted simply for code generation, also covering the aspects of RTE. To be more specific, this work aims to integrate the functionality of RTE with the M4PIA, giving the possibility of performing M2M transformation from PSM to PIM. Such as done in M4PIA, this work uses standardized MDE technology, providing insights and basis for other projects with similar goals.

Implementing RTE will allow for improved application evolution where changes made at the lowest levels - source code or Platform Specific Models - will not be lost after re-executing the transformations. This avoids the need for editing only at the high-level model, making it possible, to a certain extent, to edit and maintain models at any modeling level.

The remainder parts of this paper are organized as follows: Section 2 presents the concepts that provide the basis of this work and also discuss the related works. Section 3 presents the proposed RTE process with the M4PIA infrastructure, detailing the bidirectional transformation propose. Section 4 shows an application of the present proposal within a simplified gas compression system, also providing an initial evaluation. Finally, section 5 highlights our conclusions and future work perspectives.

2. BASIC CONCEPTS AND RELATED WORKS

2.1 Model-Driven Engineering

Model-Driven Engineering (MDE) is a software development philosophy in which models, not programs, are the main results of the development process (Schmidt, 2006). Modeling a system consists of building models that describe its characteristics and behavior. A model is an abstraction, that is, a deliberate simplification in a given context (Sommerville, 2011).

Models can represent distinct levels of abstraction and can be separated into: Platform-Independent Model (PIM) and Platform-Specific Model (PSM). The PIM is a specification that abstracts technical details of the system implementation. The PSM specifies how the functionality described in PIM is implemented on a given software platform or programming language.

Using MDE developers can focus on the problem of expressing domain concepts effectively without wasting time on the complexities of the implementation platform (Schmidt, 2006). MDE usage is seeing as advantageous because it reduces the possibility of errors, speeds implementation design processes, and enables the creation of reusable application templates (Sommerville, 2011).

An instantiation of a domain application is represented by a model, and all models are defined in accordance to their metamodels. At the metamodeling level, the elements that can compose the system model, the relevant domain concepts, as well as the possible relationships between these concepts and the rules for the combination of elements are identified, thus defining the structure, semantics, and the variables associated with the concepts. At the modeling level, real systems and applications are registered with their characteristics and specifications (Damo et al., 2019).

Models have semantic relationships between them represented at various levels of abstraction. In the so-called Model-to-Model (M2M) transformations, which allow the automation of this generation of models by mapping relationships. There are also reverse-engineering M2M transformations, in which a platform-specific model can be automatically generalized to a PIM model. Other typical transformations are Model-to-text (M2T), that generates source code in a given programming language), and its reverse, Text-to-model (T2M), that generates a PSM from the source code.

2.2 Model-Driven Reverse Engineering

Reverse engineering is the approach of understanding software and producing a model at a high level of abstraction, applicable for maintenance, documentation, or re-engineering (see Rugaber and Stirewalt (2004); Brambilla et al. (2012)). This technology has indeed proven useful in many projects to help a maintenance team gaining a better awareness of the structure and operation of a software system. But from a developer's perspective, there are two major problems: it is practically impossible to anticipate how long a reverse engineering effort will take, and there are no standards to evaluate the quality of the reverse engineering perform that the supply team works.

In order to overcome these challenges it exists the Model-Driven Reverse Engineering (MDRE). A known utilization of models by developers is to precisely specify systems before they are developed. In some instances, modeling tools can even create part or all the code automatically, tending to eliminate programming errors. MDRE adopts these characteristics of modeling technology, but applies them differently to address maintenance manager issues.

2.3 Related Works

Palacios-González et al. (2008) review existing tools that pursue the MDE paradigm. From authors' point of view, Eclipse Modeling tools are the best options for supporting MDE, as development is done with the standardized Query/View/Transformation Operational (QVTo) language.

Angyal et al. (2008) exhibit an approach based on differencing and merging of abstract syntax trees (AST) for code and model RTE. In their work, the AST is considered to be the PSM according to the taxonomy of models in MDA (Soley et al., 2000). It involves two distinct round-trip tasks: one between PIM and PSM, and other between PSM and code. The approach tries to prevent information loss during round-trip engineering by using a so called trace model which is used to synchronize the PIM and the PSM through AST.

Greiner et al. (2016) presents a Query/View/Transformation (QVT) implementation of a bidirectional model transformation. Therefore it was used the MoDisco framework (see Bruneliere et al. (2010)) to transform the Java source code into a model representation. QVT-R is used to formalize the bidirectional M2M transformation between the UML model and the Java one.

Nagowah et al. (2013) presented a RTE tool that follows MDE to generate a CRUD (Create-Read-Update-Delete) oriented application and to perform reverse engineering to better meet the requirements of Java developers.

Finished our literature analysis it was not possible to find other works using MDE in the petrochemical industry that contribute with the design and modeling of Plants and their Equipment. The present proposal focuses on such aspects, as discussed in the next section.

3. PROPOSED RTE PROCESS

In order to design models that are part of the system infrastructure, it is required for the designer to have expertise in the problem domain, so that she/he is capable of illustrating the concepts that are part of the respective domain, considering the need to have a description at a high-level of generality.

The infrastructure called M4PIA (Model-Driven Engineering for Petrochemical Industry Automation) was developed covering in its scope two software platforms devoted for the petrochemical industry: (1) MPA (Automated Procedures Module) (see Satuf and Pinto (2009)), which is used in the automation of oil platforms for development and execution of industrial process control and automation applications; (2) EMSO (Environment for Modeling, Simulation and Optimization) (see Soares and Secchi (2003)), which is an equation-based dynamic process that is applied for modeling and simulating petrochemical process. Follows a description of such software platforms.

3.1 MPA - Operation and Control Software Platform

The MPA software (Satuf and Pinto, 2009) was developed for oil and gas platforms automation, as a system for development and execution of industrial control and automation applications. MPA consists of an enforcement server and a setting/management application. Industrial plants are modeled using object-orientation and diagrams are used to define the maneuvers in the respective plant. The server is responsible for executing the configured operation maneuvers in the diagrams and handles the equipment interacting with the supervisory system through OPC (OLE for Process Control) communication bridges.

LUA programming language is used to model processes equipment. They are treated as specific classes in the preconfiguration phase of the application. In this step, *Attributes* and *Methods* of each *Class* are defined. For example, *Equipment* classes are described in a preconfiguration file and loaded at the MPA configuration stage. The developer uses the application to define the equipment instantiation which composes the plants and to model the operating maneuvers of plant equipment.

3.2 EMSO - Simulation Software Platform

EMSO (Soares and Secchi, 2003) is a software for modeling and simulating dynamic processes based on equations. It is composed by a graphical interface and its own object-oriented modeling and programming language. Such language was built from the combination of the best modeling aspects found in existing languages, resulting in a simpler language with a better code reusability.

The EMSO modeling language is composed of three major entities: *Models*, *Devices*, and *Flowsheets*. *Model* is the mathematical description of a device and can be composed of *Parameters*, *Variables*, *Equations*, *Initial Conditions*, and *Sub-Models* and can be based on a preexisting one. *Device* is an instance of the model and represents a real process equipment. *Flowsheet* means the process to be analyzed, that is composed of a set of devices.

3.3 M4PIA Infrastructure

Model-driven Engineering typically enforces higher abstraction levels along applications development. Such more abstract models are called Platform Independent Model (PIM), which go through successive refinements by means of the defined model-to-model (M2M) transformations to create Platform Specific Models (PSM). From the PSM models it is possible to automatically generates the source code for the application domain by means of model-to-text (M2T) transformations.

M4PIA infrastructure, detailed in Damo (2019), was built to support MDE of equipment class definition, including applications for simulation, control, and operation platforms in petrochemical industry. The main element is the M4PIA metamodel (PIM metamodel) that represents the entire domain of the desired applications to be created, independently of its implementation platform. Then it was created the two PSM metamodels, MPA and EMSO, which define the specificities of each platform. Based on the metamodels, it was defined the M2M transformations from PIM to PSM model.

Petrochemical industry applications inspired the definition of a set of generic elements, capable of being shared by a wide range of industrial automation system. The M4PIA metamodel, illustrated in Figure 1, is a class diagram designed using *Eclipse Modeling Framework* (EMF) and its *Ecore* metamodel, similar to UML. It consists in a set of classes, interfaces, and collaborations with their respective relationships, expressing results of the structure and requirements analysis of the problem domain and its components. Follows a brief description of the M4PIA metamodel.

The *Project* class relates to the way that models are represented and stored. It can be composed of several *Files* that can be imported as libraries (*ImportedFiles*) or be an entity group to have their source code generated by the infrastructure, such as in *GeneratedFile*.

The modeling hierarchy starts with the *Entity* class, which is abstract and serves to provide the basic structure for the more specialized classes, such as *Equipment*. Equipment is an entity type that can contain attributes and methods, defined to reflect physical characteristics and functions performed by the equipment within a plant.

The *Function* class represents a logical entity that can either represent a high-level or a low-level operation. A function also can have multiple *Variable* instances associated, acting as parameters or results, in addition it may have a language and an associated code, which textually describes

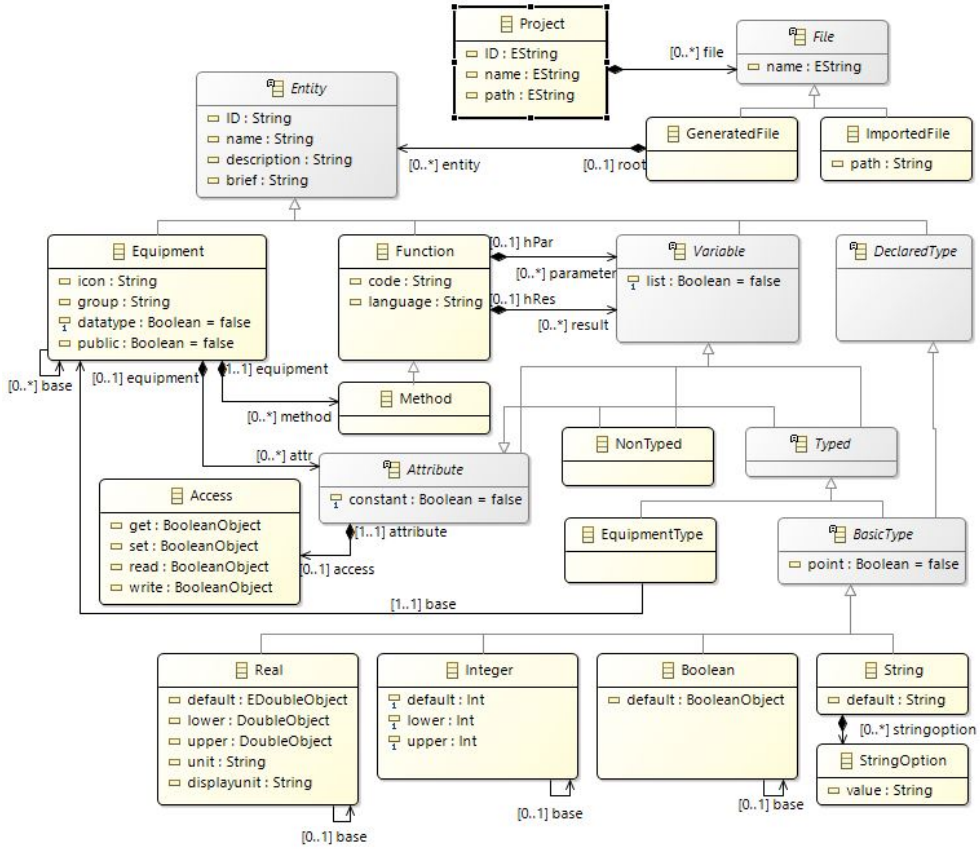

Fig. 1. The M4PIA metamodel (Damo et al., 2019)

.

the instructions to be further executed by the interpreter. The *Method* class is a specialization from *Function*, and has exclusive connection to an *Equipment*.

The *Variable* class represents a logical variable that can be *NonTyped* or *Typed*. Typed variables can be of *EquipmentType* or of *BasicType*, such as *Real*, *Integer*, *Boolean*, or *String*. An *Attribute* is a specialization of *Variable*. This metamodel hierarchy allows one to model situations where there are natural recursions in relationships between equipment, for example, when an equipment or a machine have other equipment as attribute.

Moreover, *Attribute* can be associated with access permissions to manage read/write operations by means of its association with the *Access* class. New data types can be defined through the *DeclaredType* and can also be used as the basis for variables and attributes.

3.4 Model-to-Model Transformations

M4PIA enables modeling a petrochemical plant at a higher-level of abstraction (PIM) and, from this model, it allows transforming from a M4PIA PIM model to a PSM model (MPA or EMSO). There is, however, the limitation of not supporting reverse engineering to allow refactoring.

The reverse engineering developed in this work uses the (MPA or EMSO) PSM to obtain a M4PIA PIM through M2M transformations. It is a first step towards the complete reverse engineering tool support. To develop the set of M2M transformations that can integrate with M4PIA, it was decided to use the Eclipse tool with the Query/View/Transformation Operational (QTVo) language. Figure 2 presents the two new M2M PSM-to-PIM transformations developed in this work, which add to the previously existing PIM-to-PSM transformations.

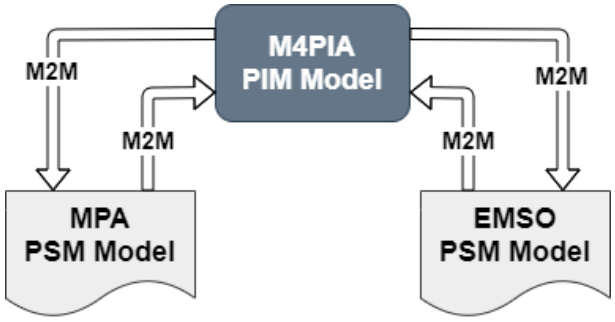

Fig. 2. Set of supported M2M Transformations.

Damo et al. (2019) propose the M2M transformation by PIM model (M4PIA) to PSM model (MPA or EMSO). The contribution to infrastructure is the transformation of each PSM metamodel to the PIM metamodel, from the lowest level of abstraction to the largest. Each class in the PSM metamodel has its respective representative in the PIM metamodel, which characterizes the rules of transformations, e.g., in the MPA metamodel, an *Equipment* matches a *Equipment* in M4PIA metamodel of higher level of abstraction, both with the same name. In the case of the EMSO metamodel, a *Model* corresponds to an *Equipment*

in the M4PIA metamodel. Each *mapping* in the QVTo code is responsible for linking the transformation to its corresponding concept at another metamodel level.

4. CASES STUDIES AND INITIAL EVALUATION

Damo (2019) presented a study case of the infrastructure M4PIA in the focused on the application of advanced control strategies for oil and gas extraction platforms. In order to represent the use of the proposed solution this section presents the implementation of a related subsystem, more specifically the control of a simplified gas compression system. The emphasis in this implementation is use the support the design and refinement iterations of the models and transformations, mainly the reverse engineering that is the add in the M4PIA infrastructure. The development were done whitin the Eclipse environment.

The compression system needs to process the gas leak through the inlet vessel. If the compressor fails, gas builds up and the pressure in the vessel increases. To prevent pressure from exceeding safety limits, part of the gas must be flared. An inlet gas flow below specified limits may lead the compressor to an unstable operating region (surge). To prevent compressor downtime, each compression stage has a recycling line with an anti-surge valve.

The simplified compression system represented in the Figure 3 is composed of one *Surge Tank* (a flaring valve with a knockout drum), one *Output Header* and two *Stage Compressors*. A compression stage has a *Heat Exchanger*, a *Suction Drum*, a *Compressor Element* and an *Recycle Valve*. The output header has three control valves: one exportation valve and two gas lift injection outlet valves.

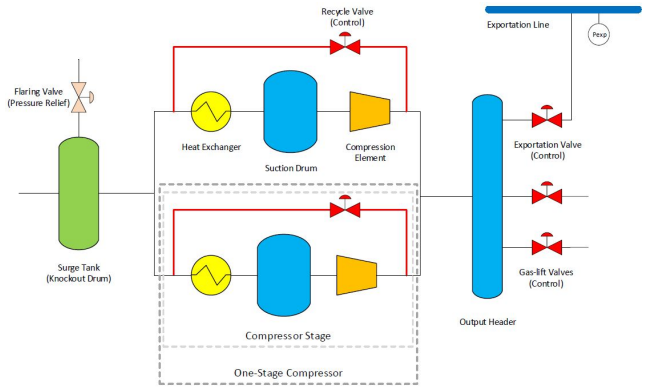

Fig. 3. Simplified Gas Compression System (Damo et al., 2019).

4.1 Initial Evaluation

To test the infrastructure transformation cycles, an empirical evaluation with four scenarios was proposed, as shown in Figure 4. Our goal is to test the Round-Trip Engineering in the infrastructure covering some possibles transformation with the source a PSM model (MPA or EMSO) and the target is a PSM model as well through M2M transformation. The intention is to detect possible information losses and similarity cases. Two models of the simplified gas compression system at the PSM level, one for MPA and the other for EMSO were used as starting point for performing the transformations to be evaluated.

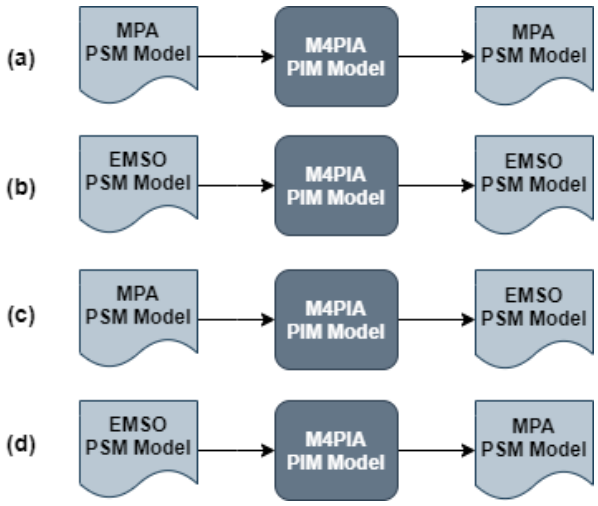

Fig. 4. The four case study scenarios analyzed.

In *case (a)*, taking a MPA PSM as starting point, the transformation output was equal to the original model, with exactly the same model with the same number of *Equipment*, *Methods Variables*, and *Attributes*. The same was true for *case (b)*, where an EMSO PSM was used as starting point and the transformation output was the equal to the initial model, with the same number of *Models*, *Variables*, *Parameters*, and *Equations*.

Case (c) started with a MPA PSM, aiming to transform it into an EMSO PSM. To help analyzing the transformation process, the generated EMSO PSM is compared with one created "manually" by an experienced designer, as shown in Table 1. *Models* and *Equations* obtained the same quantity of items. However, in MPA, for modeling the Compressor System and the Stage Compressor, *List* attributes were used - and it is not necessary to define the list size. In EMSO, however, it is necessary a *Parameter* therefore, this justifies two of the missing parameters in the generated EMSO model. The third missing parameter is due the need, in MPA, of a *Variable* to receive the return of a *Method*. In EMSO this is already integrated with *Equations*, so it generated a variable that will not be useful in the model, but also do not cause errors.

Table 1. MPA to EMSO Transformation (*c*): "Manual" vs. Generated EMSO

	"Manual" EMSO	Generated EMSO
Num. of Models	7	7
Num. of Variables	13	14
Num. of Parameters	3	0
Num. of Equations	2	2

Case (d) performed the transformation of an EMSO PSM to an MPA PSM. The same comparison approach was used and an MPA model was "manually" created by an experienced designer. Table 2 presents the related numbers from the "manually" created and the automatically generated MPA models. The differences occur for the inverse reason of *case (c)*, the initial EMSO model has two *Parameters* for defining the sizes of the List *Attributes*, but there is no need for that in the MPA, so there are two more (useless) *Attributes* in the Generated MPA. The missing *Variable* in the generated model is due to the fact that *Equations* in EMSO do not need a Variable to receive the result of the

operation, which incurs into an error. This helped us to make the proper adjustments in the transformation rules.

Table 2. EMSO to MPA Transformation (d): "Manual" vs. Generated MPA

	"Manual" MPA	Generated MPA
Num. of Equipment	7	7
Num. of Methods	2	2
Num. of Attributes	14	16
Num. of Variable	1	0

Finished the proposed analysis it was possible to conclude that the developed RTE solution is ready to be used in more complex systems. It can either use MPA or EMSO models as starting point, as transformations in both directions seem to work properly.

Furthermore, additional analysis should be done to allow reasoning about productivity gains that might be achieved by design teams when using the proposed RTE solution.

5. CONCLUSIONS AND FUTURE WORKS

This paper presented a round-trip engineering (RTE) solution implemented within a previously developed MDE infrastructure and tool support for Petrochemical Industry Automation (M4PIA). The solution uses standardized MDE technology, providing insights and basis for other projects with similar goals. It enables reverse engineering from (MPA and EMSO) PSM to M4PIA PIM, and is a first step towards the complete RTE support in M4PIA infrastructure.

For validation purposes, a simplified gas compression system was developed. It was observed that starting with a given x PSM and then going up and down (PSMx-PIM-PSMx), the resulting model is the similar. From x PSM aiming to obtain an y PSM (PSMx-PIM-PSMy), the resulting model showed few differences justified by the specifics of each tool. Nevertheless, the obtained y PSM is similar to a model designed directly in such target software.

The next step of this work should cover the implementation of Text-to-Model (T2M) transformations, thus allowing existing equipment libraries to be imported as M4PIA models, letting it available to all software platforms supported by the infrastructure. It is also desired to apply a case study with the modeling of a more complex petrochemical plant to double-check the complete transformation processes, analyze the losses, and the effectiveness of the proposed infrastructure. With the complete RTE implementation it is expected to profit from all benefits of MDE, like gains in applications maintainability, interoperability, traceability between different software platforms, to reduce development time, and to enhance compatibility between different models of the same concepts.

ACKNOWLEDGEMENTS

Authors thank the Petrobras Research Center (CENPES) for financing this work. Thanks also goes to Prof. Julio Normey-Rico and all other members of SCoPI/UFSC research group that contributed for this research. Finally, authors thank Prof. Fabio Basso (UNIPAMPA) for his collaboration.

REFERENCES

Angyal, L., Lengyel, L., and Charaf, H. (2008). A synchronizing technique for syntactic model-code round-trip engineering. In *15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008)*, 463–472. IEEE.

Brambilla, M., Cabot, J., and Wimmer, M. (2012). Model-driven software engineering in practice. *Synthesis Lectures on Software Engineering*, 1(1), 1–182.

Bruneliere, H., Cabot, J., Jouault, F., and Madiot, F. (2010). Modisco: a generic and extensible framework for model driven reverse engineering. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, 173–174. ACM.

Damo, T.P. (2019). Engenharia Baseada em Modelos para a Aplicação de Simulação, Controle e Operação de Plantas na Indústria Petroquímica.

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. doi:10.25046/aj040422.

Greiner, S., Buchmann, T., and Westfechtel, B. (2016). Bidirectional transformations with qvt-r: a case study in round-trip engineering uml class models and java source code. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 15–27. IEEE.

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.

Nagowah, L., Goolfee, Z., and Bergue, C. (2013). Rtet-a round trip engineering tool. In *2013 International Conference of Information and Communication Technology (ICoICT)*, 381–387. IEEE.

Palacios-González, E., Fernández-Fernández, H., García-Díaz, V., García-Bustelo, B.C.P., and Lovelle, J.M.C. (2008). A review of intelligent software development tools. In *IC-AI*, 585–590.

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

Satuf, E. and Pinto, S. (2009). e quaresma, b.,(2009). *Sistema automático de alinhamento para a Plataforma de rebombeio autônomo PRA-1*.

Schmidt, D.C. (2006). Model-driven engineering. *COMPUTER-IEEE COMPUTER SOCIETY-*, 39(2), 25.

Seborg, D.E., Mellichamp, D.A., Edgar, T.F., and Doyle III, F.J. (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, 947–952. Elsevier.

Soley, R. et al. (2000). Model driven architecture. *OMG white paper*, 308(308), 5.

Sommerville, I. (2011). Software engineering 9th edition. *ISBN-10*, 137035152.