DOMAIN MODEL AND MODELING WORKBENCH FOR MODELING PDCA-MANAGEMENT SYSTEMS

Walter S.A. Schwaiger[1], Christian Fischer-Pauzenberger[1], Mathias Cammerlander[1]
[1] Technische Universität Wien, Institute of Management Science, Theresianumgasse 27, 1040 Wien, Austria
{walter.schwaiger, christian.fischer-pauzenberger, mathias.cammerlander}@tuwien.ac.at

Abstract. Plan-Do-Check-Act (PDCA)-management systems [1] are cybernetic management systems that underly all management systems standards from the International Organization for Standardization (ISO, https://www.iso.org). Furthermore they play an essential role in the Resource-Event-Agent (REA)-business management ontology [2] and in the OntoREA© Accounting and Finance Model [3]. The peculiarity of PDCA-management systems lies in the fact that they contain structural and behavioural components so that they are difficult to model by using the UML language. In this article this modeling problem is solved by the development of a PDCA-domain model (Ecore), i.e. a domain specific language (DSL) and a corresponding PDCA-modeling workbench (EMF) that allows the user to construct context-specific PDCA-applications.

Keywords: Model driven engineering, method engineering, meta-modeling platforms, semantic PDCA-domain model, Ecore PDCA-domain model, PDCA-DSL, PDCA-modeling workbench, Eclipse Modeling Framework (EMF), Xtext grammar language.

Table of Content

1 Introduction

PDCA-management systems are cybernetic planning and control systems where at the beginning of the planning period objectives are set in the plan activity and over time the achievement of the objectives is controlled. The controlling consists of two activities, i.e. the check activity where the measured performance is the feedback information that is compared with the objective, and the act activity where the resulting deviation between the objective and the performance determines the control input for the act activity. As can already be seen from this informal definition, the PDCA-management systems contains behavioral components in form of activities as well as structural components in form of the information objects that are exchanged between the activities.

This peculiarity of PDCA-management systems makes it difficult to model them by using UML modeling tools as these tools are specialized either on the structural part (e.g. UML class diagrams: UML-CD) or on the behavioral part (e.g. UML activity diagrams: UML-AD). In the literature both approaches where applied so far. By modeling PDCA-management systems with the UML-AD language [1] the PDCA activities are modelled as UML activities together with the accompanying flows of the corresponding information objects which are modelled as UML objects. By modeling PDCA-management systems with the UML-CD language [2] both components, i.e. the structural and the behavioral components are modeled as UML classes.

The conceptual modeling of the PDCA-management systems is surely beneficial for understanding and communication purposes. But furthermore it should also provide a solid foundation for the model driven engineering of management systems applications. To attain this, an executable domain specific language (DSL) with a corresponding modeling workbench should be established.

The establishment of such an executable DSL for the PDCA-domain (PDCA-DSL) is the primary research objective of this article. In order to achieve this objective a meta-modeling approach [4] is taken and an adequate modeling method is chosen. *Modelling methods consists of two components: a modelling technique, which is divided in a modelling language and a modelling procedure, and mechanisms & algorithms* (modeling infrastructure) *working on the models described by the modelling language.* [5, p.183]. The modeling language consists of an abstract syntax, a concrete syntax (visual notation) and the semantics. For the definition of the abstract syntax the graphical editor “Ecore Tools” will be used. Ecore Tools is based upon the meta-meta-modeling language [6] “Ecore” and it is available in the “Eclipse Modeling Framework” (EMF) [4]. This Java-based modeling method is chosen as it allows the establishment of the meta-model for the PDCA-domain (PDCA-DSL) and the corresponding PDCA-modeling workbench (executable DSL).

The article is structured as follows: In the next section two semantic PDCA-domain models are presented. Next to that the PDCA-domain model (Ecore), i.e. the PDCA-DSL is constructed. After that the establishment of the PDCA-modeling workbench is addressed. In the final section the paper is summarized and concluded.

2 PDCA-management systems: Semantic PDCA-domain modeling

So far, the semantic PDCA-domain was modelled in the literature in two ways. In Fig. 2 (Fig. 3) the modeling in the UML-AD (UML-CD) language is shown.

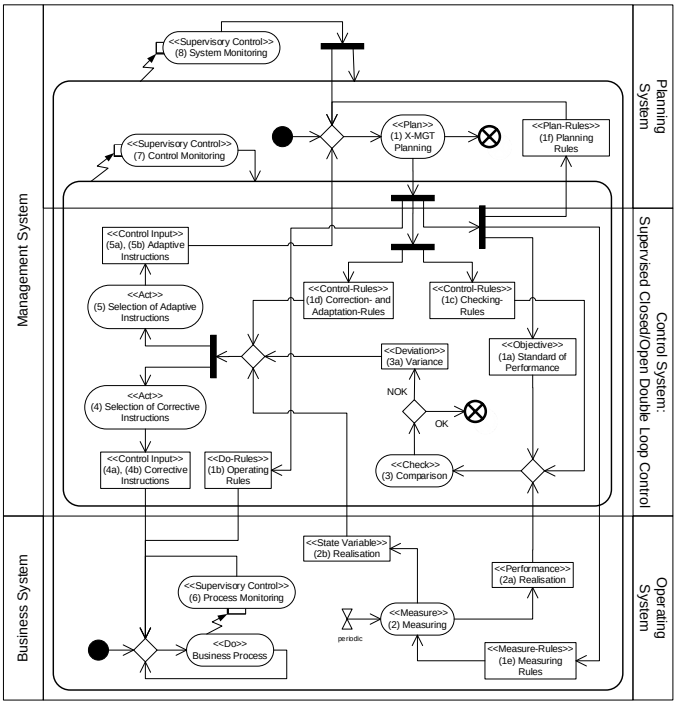

Fig. 1: Semantic PDCA-domain model [1] – UML-AD language

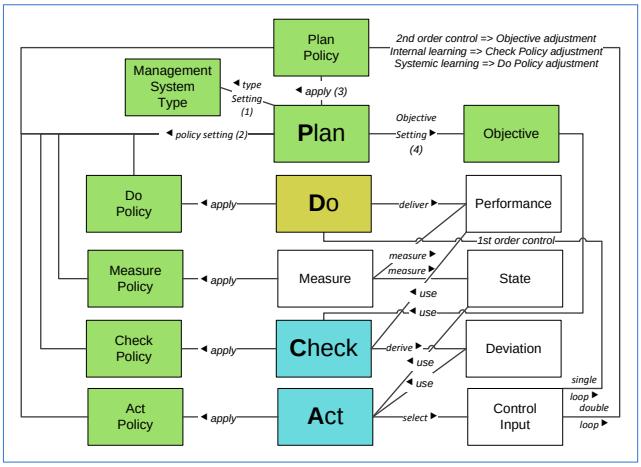

Fig. 2: Semantic PDCA-domain model [2] – UML-CD language

The semantics describes the meaning of a modelling language (PDCA-DSL) *and consists of a semantic domain and the semantic mapping. The semantic domain describes the meaning by using ontologies* (PDCA-domain in Fig. 2 and Fig. 3), *mathematical expressions etc. The semantic mapping connects the syntactical constructs with their meaning defined in the semantic domain.* [5, p. 186].

The semantic PDCA-domain models identify the elements that exist in PDCA-management systems and equivalently express their meanings. Of special importance is the interconnection between the activities and the informational object flows. E.g., in the `Plan` activity (3) the `Plan Policy` is applied to set (1) the management system type, (2) the policies for the `Do`, `Measure`, `Check` and `Act` activities and (4) the `Objective`. Over time (sand clock symbol) the `Performance` is measured, compared with the `Objective` and the `Deviation` is determined that induce a corresponding `Control Input`.

3 PDCA-DSL: PDCA-domain model – Meta-modeling (EMF)

After having specified the semantic domain it can be mapped with the syntactical constructs of the abstract syntax. In order to establish an executable PDCA-DSL the *Eclipse Modelling Framework (EMF)* is used [7]. It *supports the Ecore standard, which is a simplified implementation of MOF. Ecore is currently the de facto standard for metamodelling, and is used in the Eclipse implementation of UML, along with many other general-purpose and domain-specific language (DSL) tools.* [4, p.401].

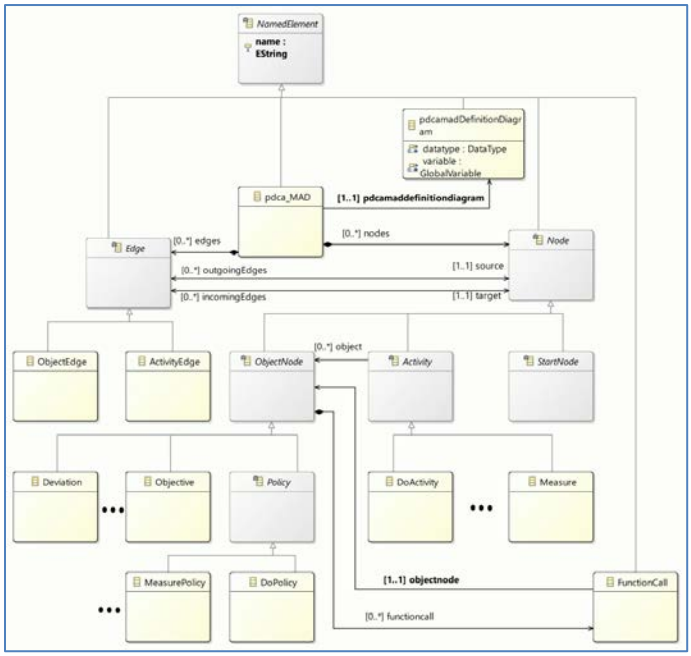

Fig. 3: PDCA-domain model excerpt (PDCA-DSL) – Ecore meta-modeling language

Fig. 3 shows an excerpt from the *PDCA-meta-model* – in Ecore's EMF notation – which defines the abstract syntax for the PDCA-domain. It was produced with the graphical editor Ecore Tools (https://wiki.eclipse.org/Sirius/Tutorials/DomainModelTutorial). This meta-model is formulated in the Ecore meta-modeling language, it represents the *PDCA-domain model* and defines the *PDCA-DSL*. The PDCA-DSL is based upon a graph-theoretical foundation by using `Node` and `Edge` constructs. The peculiarity of PDCA-management systems is integrated by the inclusion of two different node types, i.e. the `Activity` node that covers the behavioural components and the `ObjectNode` for covering the structural components.

4 PDCA-DSL: PDCA-modeling workbench (Eclipse Sirius)

The concrete syntax for the PDCA-DSL is established by the diagram editor of "Eclipse Sirius" (https://wiki.eclipse.org/Sirius/Tutorials/StarterTutorial). Furthermore, this editor allows the building of the PDCA-modeling workbench. This workbench is the tool that supports the user in constructing a specific PDCA-management system for the context under consideration. In Fig. 4, exemplarily a "closed double loop management system" – e.g. like the Balanced Scorecard management system from Kaplan/Norton – is generated with the workbench to demonstrate its applicability. Furthermore, the example PDCA model shows that the primary research objective of the article is achieved as the structural and behavioral peculiarities of PDCA-management systems are fully covered in the PDCA-DSL and they are provided to the user (modeler) of the corresponding PDCA-modeling workbench.

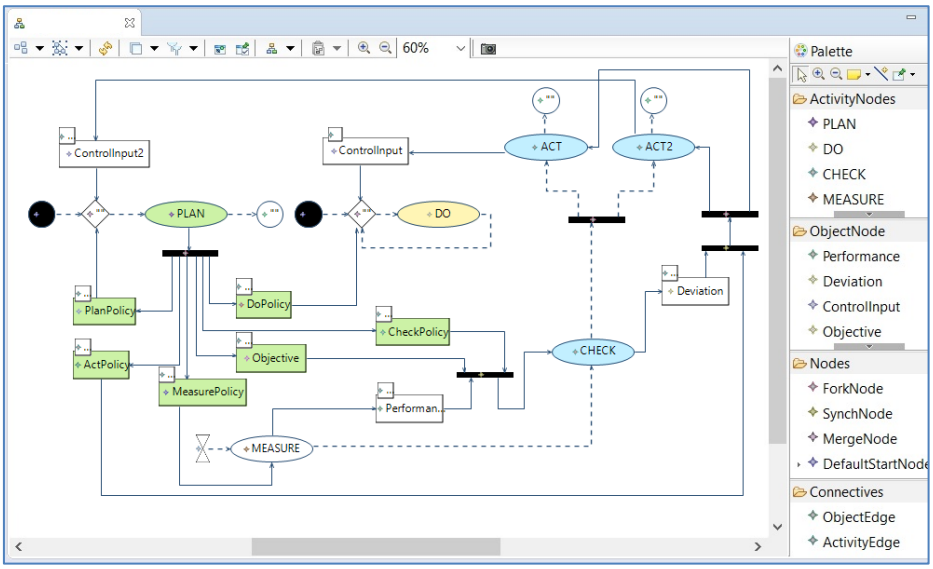

Fig. 4: PDCA-modeling workbench (Sirius) – Closed double loop management system

5 Conclusion

The primary research objective of this article was the establishment of a PDCA-DSL and a corresponding PDCA-modeling workbench based upon the semantic PDCA-management system models that are expressed in the non-executable UML-AD and the UML-CD language. For this purpose a two-step approach was taken: Firstly, the semantic content of the two equivalent PDCA-domain models was mapped into the abstract syntax that constitutes the PDCA-meta-model and describes the PDCA-DSL. This mapping was done by the construction of the PDCA-meta-model in the graphical editor Ecore Tools which is available in the Eclipse Modeling Framework and is based upon the meta-modeling language Ecore. Secondly, for the abstract syntax a concrete syntax (visual notation) was constructed in the diagram editor of the Sirius Eclipse plug-in, and upon this syntax a graphical designer with palette tools for the PDCA-modeling workbench was established.

The PDCA-DSL (meta-model, abstract syntax) and the corresponding PDCA-modeling workbench include the peculiarity of PDCA-management systems of covering structural and behavioural components at the same time. This match shows that the primary research objective is met. For demonstrating the functioning of the PDCA-modeling workbench a closed double loop management system was created.

It has to be mentioned that the current version of the PDCA-modeling workbench is limited to visual modelings only, so that currently no computational functionalities are available. In the PDCA-DSL (Fig. 3) the inclusion of such functionalities is already foreseen. This can be seen by the node `FunctionCall` that is attached to the `ObjectNode` so that it is also available in the `Policy` node. This means, the policies can be equipped with functions for performing special tasks. Such functions are e.g. the calculations needed in the PDCA activities for the dynamic replication of option contracts [3].

Such an integration of computational functionalities can be achieved in further research by the “Xtext” grammar language (https://www.eclipse.org/Xtext/documentation/301_grammarlanguage.html). This language is also related to the Ecore modeling language. Consequently, it can “communicate” with the PDCA-DSL and can provide the additional behavioural functionalities. *Building a domain-specific language (DSL) for structural parts of an application has always been rather easy with Xtext. But structure alone is not sufficient in many cases. When it comes to the behavioral aspects users often fall back to implementing them in Java. The reasons are obvious: expressions and statements are hard to get right and extremely complex and therefore costly to implement. This document introduces and explains a new API, which allows reusing predefined language constructs such as type references, annotations and fully featured expressions anywhere in your languages...* (https://www.eclipse.org/Xtext/-documentation/201_sevenlang_introduction.html).

References

1. Abmayer, M., Schwaiger, W.S.A.: Accounting and management information systems – a semantic integration. In: Weippl, E., Indrawan-Santiago, M., Steinbauer, M., Kotsis, G., Khalil, I. (ed.) 15th International Conference on Information Integration and Web-based Application and Services (iiWAS 2013). pp. 345–351. ACM, ISBN 978-1-4503-2113-6, Vienna, Austria (2013).
2. Schwaiger, W.S.A.: REA Business Management Ontology : Conceptual Modeling of Accounting , Finance and Management Control. In: España, S., Ivanović, M., and Savić, M. (eds.) CAiSE'16 Forum at the 28th International Conference on Advanced Information Systems Engineering. pp. 41–48. http://ceur-ws.org, Ljubljana, Slovenia (2016).
3. Fischer-Pauzenberger, C., Schwaiger, W.S.A.: OntoREA© Accounting and Finance Model: Hedge Portfolio Representation of Derivatives. In: Buchmann, R.A. and et al. (eds.) 11th IFIP WG 8.1. Working Conference, PoEM 2018. pp. 372–282. Springer Nature Switzerland, Vienna, Austria (2018).
4. Paige, R.F., Kolovos, D.S., Polack, F.A.C.: A tutorial on metamodelling for grammar researchers. Sci. Comput. Program. 96, 396–416 (2014).
5. Karagiannis, D., Kühn, H.: Metamodelling Platforms. In: Bauknecht, K., Min Tjoa, A., and Quirchmayer, G. (eds.) Proceedings of the Third International Conference EC-Web 2002 – Dexa 2002. pp. 182–203. LNCS 2455, Springer, Berlin, Heidelberg, Aix-en-Provence, France (2002).
6. Sutîi, A.M., Brand, M. Van Den, Verhoeff, T.: Exploration of modularity and reusability of domain-specific languages : an expression DSL in MetaMod. Comput. Lang. Syst. Struct. 51, 48–70 (2018).
7. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF - Eclipse Modeling Framework. Addison-Wesley, Upper Saddle River, NJ et al. (2009).