MATURE: A Model Driven bAsed Tool to Automatically Generate a langUage That suppoRts CMMI Process Areas spEcification

David Musat[1], Víctor Castaño[1,2],
Jose A. Calvo-Manzano[1], and Juan Garbajosa[1]

[1] Technical University of Madrid (UPM), Madrid, Spain
[2] International Center for Numerical Methods in Engineering (CIMNE)
dmusat@syst.eui.upm.es, vcastano@cimne.upc.edu,
joseantonio.calvomanzano@upm.es, jgs@eui.upm.es

Abstract. Many companies have achieved a higher quality in their processes by using CMMI. Process definition may be efficiently supported by software tools. A higher automation level will make process improvement and assessment activities easier to be adapted to customer needs. At present, automation of CMMI is based on tools that support practice definition in a textual way. These tools are often enhanced spreadsheets. In this paper, following the Model Driven Development paradigm (MDD), a tool that supports automatic generation of a language that can be used to specify process areas practices is presented. The generation is performed from a metamodel that represents CMMI. This tool, differently from others available, can be customized according to user needs. Guidelines to specify the CMMI metamodel are also provided. The paper also shows how this approach can support other assessment methods.

Keywords: Model Driven Development (MDD), Model Driven Architecture (MDA), Domain Specific Language (DSL), Domain Specific Model (DSM), Capability Maturity Model Integration (CMMI), Spice ISO/IEC 15504.

1 Introduction

Nowadays, an increasing number of organizations suffer from symptoms of bad performance such as missed commitments, inadequate management visibility and quality problems. These are the most readily visible consequences of real harms that directly affect productivity. During the last two decades several models and methods for process improvement have arisen as good solutions to help these organizations to improve their processes. Six Sigma [1], Lean Thinking [2], the Theory of Constraints [3], ISO 9000 Quality Standards [4] or the CMMI suite [5] are some of the current examples of popular approaches for process improvement.

Born and based on the software industry, Capability Maturity Model Integration (CMMI) is a suite of products used for improving processes that helps

A. Riel et al. (Eds.): EuroSPI 2010, CCIS 99, pp. 48–59, 2010.

to integrate traditionally separate organizational functions and sets process improvement goals and priorities. Not so many tools support all kind of CMMI related activities. However the support level provided is often very limited, and its capabilities to be customized according to user needs are quite short[6–8].

In general terms, all forms of engineering rely on models as essential tools to understand the complex real-world systems. Model Driven Engineering (MDE) [9] is a discipline based on the use of models for software development through model transformations. The idea of models, modeling and model transformations are the basis of a set of software development approaches that are known as Model Driven Development (MDD)[12][13].

Domain Specific Model Driven Development (DSMDD)[10] is a way to automate a problem resolution when it happens repeatedly. DSMDD is embedded in MDD processes. From a Domain Specific Model (DSM) it is relatively easy to derive a language that is near the domain, a Domain Specific Language (DSL)[11].

A CMMI constellation is a set of CMMI components designed to meet the needs of a specific area of interest. In the CMMI constellations, the configuration of a process model is the basic pillar of the complex structure of process improvement. The definition of a model for process improvement can be seen as a DSM and, as a result, a DSL can be derived from it. It is possible to extract the elements from the CMMI-DEV v1.2 framework, to produce a Domain Specific CMMI Language. Developing a graphical tool embedded in a MDD process, will facilitate user without programming skills the development of DSLs to support documentation, verification and validation tasks.

The goal of this work has been producing a graphical artifact that effectively models organizational processes. The motivation factor stays on the versatility to create DSLs from a DSM.

The structure of the paper is as follows: In section 2 the research strategy followed in explained, section 3 presents the notion of CMMI constellations and its metamodel specification, section 4 presents how to deploy this metamodel in a domain specific model driven development process and get the pursued DSL; section 5 describes how the metamodel components are specified by defining a description of five views; section 6 presents MATURE, the tool that automate the derivation of DSLs through DSMs. Section 7 presents a case study applied to MATURE. Finally, conclusions and further work are presented in section 8.

2 Research Strategy

The research strategy was structured as follows:

- Study of CMMI from an MDD and DSM point of view
- Extraction of the main CMMI components.
- Composition of a CMMI constellations metamodel.
- Definition of the graphical metaphors per each metaclass of the metamodel.
- Introduction of the metamodel in a model management tool.
- Definition and generation of the tool supporting CMMI constellations model definition.

- Application of that tool to a case study.
- Automatic derivation of the DSL associated from that case study.
- Evaluation of results with respect to initial goals

3 Defining a Metamodel for CMMI

Next we will present the definition of a metamodel supporting CMMI constellations and the associated graphical metaphor for each of the metaclasses conforming the metamodel.

3.1 Definition of the CMMI Constellations Metamodel

Metamodels define models and establish properties in a precise way; a metamodel is a model of models [12]. For this reason, the definition of CMMI constellations can be specified by means of a metamodel (see Fig. 1). This metamodel contains a set of inter-related metaclasses that define the CMMI complete constellations. Metaclasses, their properties and relationships let exist process areas and its components be well defined.

The Process_area metaclass (see Fig. 1) represents the set of related practices in an area. As the 22 process areas in the case of the development constellation are already defined in [5], the type of the name is an enumerator (PA_Name) that contains the 22 possible values. To reflect the high level relationships between process areas a Related_process_area reflexive association is provided with a 0..21 cardinality because the association of a process area to itself is not allowed.

The specific_goal and generic_goal metaclasses (see Fig. 1) are specializations of the goal_statement metaclass. Both metaclasses of the metamodel correspond respectively to Specific Goal and Generic Goal CMMI constellation components. They inherit the attributes of the goal_statement metaclass, i.e. the Title of the goal, the Notes associated with the goal and the Goal Number, always beginning with SG or GG whether the type of the goal is specific or generic respectively.

A specific goal is implemented by many Specific Practices, reflected in the metamodel with the SpecificPractice metaclass (see Fig. 1), specialization of the PracticeStatement metaclass. These Specific Practices are defined by means of Subpractices and Typical Work Products metaclasses, SPSubpractice and Typical_Work_Product (see Fig. 1) respectively in the metamodel. Moreover, a Generic Goal is implemented by many Generic Practices, reflected in the metamodel with the GenericPractice primitive, specialization of the PracticeStatement abstract metaclass (see Fig. 1). These GenericPractices are defined by means of Subpractices and Generic Practice Elaborations, GPSubpractice and Generic_Practice_Ellaboration metaclasses respectively in the metamodel. Both Specific Practice and Generic Practice inherit the attributes from the practice statement metaclass. They include the Title of the practice, the Number of the practice always beginning with SP or GP depending on the kind of the practice and the Notes associated with the practice.

To define the metamodel primitives it is necessary to specify the descriptive components in [5]. For this reason, the Purpose Statement and the Introductory

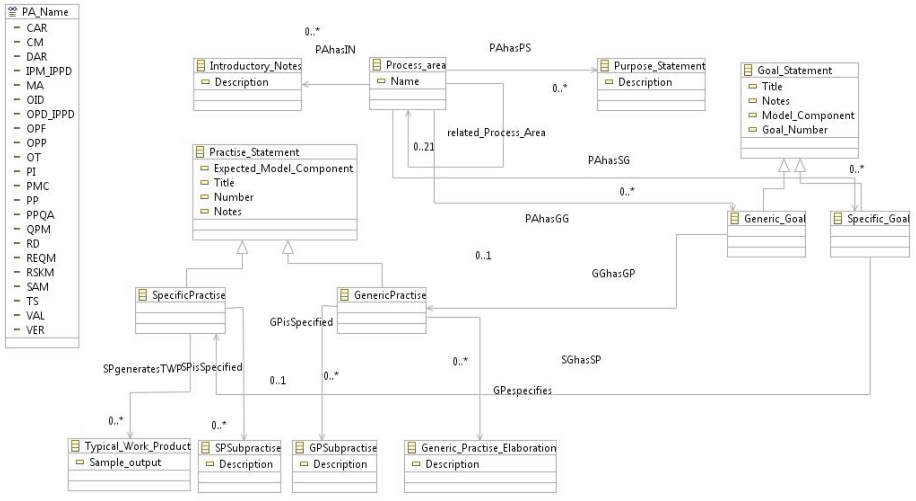

Fig. 1. CMMI constellations metamodel

Notes are represented by the Purpose_Statement and Introductory_Notes metaclasses (see Fig. 1). A set of acronyms was used to define the relations between the metaclasses. The logical path followed to determine the names of the relations was extracting the first characters of the names of the related primitives and composing the name of the relation as sourceAcronym-Verb-TargetAcronym, e.g. a relation between the Process Area (source) and the Specific Goal (target) name is PAhasSG (see Fig. 1). To finalize the specification of the process area components in [5], some supporting informative components are described. In our approach these components were not included since they can be specified in the form of annotations or informative notes. This metamodel provides the

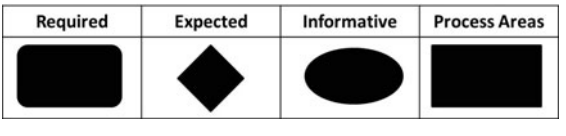

Fig. 2. Representation of the CMMI constellation elements

necessary metaclasses to model any of the CMMI constellations (development, acquisition and services). However, to use these primitives it is essential to define a graphical metaphor for modeling them. The graphical metaphor conceived in this paper was composed depending on the nature of the metaclass (required, expected and informative). Required components are represented by rounded rectangles, expected components are represented as diamonds and informative components are represented as ellipses. The process area graphical metaphor was not already defined. Due to the globalism it implies it was represented as a box (see Fig. 2).

3.2 CMMI in MDD

Model Driven Architecture (MDA)[14] [15] proposes a four level abstraction architecture that form a hierarchy or model architecture. In the upper layer (see M3 layer in Fig. 3), Model Object Facility (MOF)is defined. MOF is the language that provides the way to build a metamodel (see M2 layer in Fig. 3). The CMMI constellations metamodel presented in this paper is included in this last level. When a model (see M1 in Fig. 3) is built following a metamodel, the model conforms to that metamodel. Going to the lowest level, having a model and a real coded system by means of that model (see M0 layer in Fig. 3), that system is an instance of that model. In this lowest level, the data of the specific instances of the model is defined. The case study presented in this paper is placed in this layer. Since the CMMI constellation metamodel is included in layer M2

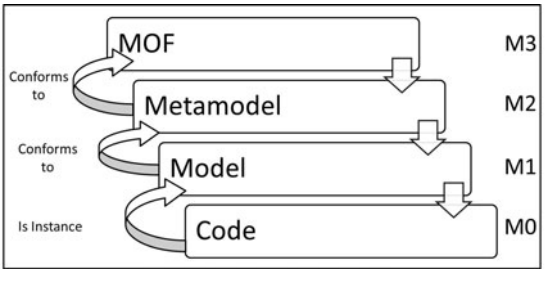

Fig. 3. CMMI for Development in MDD

it is possible to build or develop models at layer M1 using its primitives and guaranteeing model correctness.

4 Strategy for Implementing CMMI in Domain Specific MDD

The development approach followed in this work is based on Domain Specific Model Driven Development and solves the problem of the manual definition of a CMMI Domain Specific Language. Deriving the CMMI Domain Specific Language from the CMMI domain specific model takes some steps detailed in Fig. 4. The first one is to introduce the CMMI metamodel and the graphical

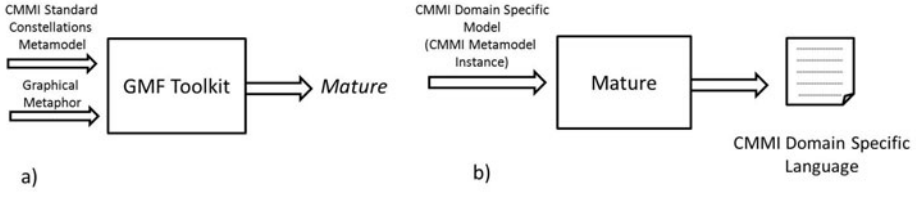

Fig. 4. Steps to generate a Domain Specific Language

metaphors defined in section 3 in a model management tool that supports the definition of graphical editors by means of a model, Graphical Modeling Framework (GMF) [16]. Then GMF generated MATURE automatically. MATURE is a tool specifically devised for our purposes within this research(see section 1). To get a domain specific language with MATURE, the next step was getting the definition of a domain specific model, an instance of the CMMI metamodel, into MATURE framework. Once the model is already defined, the domain specific language is automatically generated.

5 CMMI Constellations Views

To simplify the complexity of the CMMI constellations metamodel five different views are presented in this paper: configuration, descriptive, generic, specific and general views.

- **Configuration View:** specifies the preliminary configuration of the CMMI constellations metamodel. As a CMMI constellations model has twenty-two process areas in the case of development, it would be better to have the definition of the process areas isolated from the rest of the components. As a result, this view is composed by the process areas component and the link that relates one process area with others. Having this view properly configured, the user will have defined a set of process areas with high level relationships between them.
- **Descriptive View:** specifies the necessary informative components to implement a process area in the CMMI constellations model. This view is composed by two kinds of components (i) process areas and (ii) informative components. The former are also included in the Configuration View, although in this view, the relations between process areas are not shown. The latter include the Introductory Notes primitive, which describes the major concepts covered in a process area, and the Purpose Statement primitive, which describes the purpose of a process area.
- **Generic View:** specifies the generic definition of a process area. It was called generic because it can be applied to more than one process area. This view is composed by all the necessary primitives for modeling the generic part of a process area, including the generic goal, generic practices, generic practice elaborations and subpractices (GPSubpractice in Fig. 1).
- **Specific View:** specifies the specific definition of the specific part regarding a process area. It was called specific because can be only applied to one process area. This view is composed by all the necessary components for modeling the specific part of a process area, including the specific goals, specific practices, typical work products and subpractices (SPSubpractice primitive in Fig. 1).
- **General View:** Provides a way to have a general perspective of the CMMI constellations metamodel. In this view, it is possible to link process areas with the generic and specific goals once they have been implemented. This

view includes all the previously view components (required, expected and informative), with two more relations, the relation between a process area and its generic goal, and the relation between a process area and its specific goal.

6 The MATURE Tool

MATURE is a tool that supports the MDD paradigm and offers as modeling primitives the necessary concepts to define a Domain Specific CMMI Model. These concepts are based on the metamodel defined in section 3. Building the Domain Specific Language is provided by letting the user define the domain specific model from that language is derived, with a friendly and graphical environment.

MATURE was developed using Graphical Modeling Framework (GMF) [16] from Eclipse [17] platform, with the target of providing a free CMMI software modeling tool. As a result, the CMMI constellations metamodel has been specified in Ecore using Eclipse Modeling Framework (EMF)[18]. Furthermore, the graphical metaphor has been defined according to the metamodel concepts. In this way, MATURE has been automatically generated with GMF.

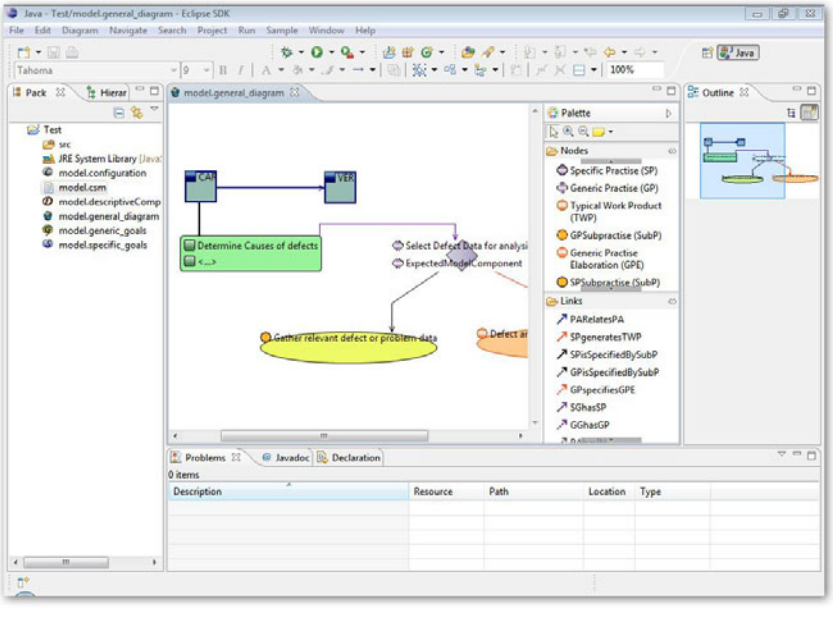

Fig. 5. Mature environment

MATURE graphical user interface is shown in Fig. 5. On the left side, six different documents associated and created for the Causal Analysis Resolution (CAR) process area domain are shown. Five of them correspond to the

diagrams that let the graphical modeling of the process which extensions are *general_diagram, specific_goals, generic_goals, descriptive* and *configuration*. The other one corresponds to the XML document generated by means of the model being developed (*csm* extension) that contains the DSL. On the right side, the modeling tool palette can be seen, it lets, by drag and drop, model in a more intuitive way the process model. This pallet separates the elements of the process model (top part) from the links among them (bottom part) in two different sections. As we are at tool level, metaclasses are known as primitives. The elements of the process model are: process areas that are to be configured (*Process_area*), purpose of the process areas (*Purpose_Statement*), major concepts addressed in a process area (*Introductory_Notes*), the unique characteristics that must be present to satisfy a process area (*Specific Goal*), the characteristics that must be present to institutionalize the processes that implement a process area (*Generic_Goal*), the description of an activity that is considered important in achieving the associated specific goal (*Specific_Practice*), lists of sample output for a specific practice (*Typical_Wok_Product*), detailed description that provides guidance for interpreting and implementing a specific or generic practice (*Subpractice*), description of an activity that is considered important in achieving the associated generic goal (*Generic_Practice*) and guidance on how the generic practice should be applied uniquely to the process area (*Generic_Practice_Elaboration*). The section of the supporting informative components (*Notes, Examples, Amplifications* and *References*) was not included in the metamodel, but a Note tool is available to support these elements. Each modeling primitive that MATURE provides, has a different representation that characterizes them when they are dropped in the modeling canvas. The representation has been defined following the classification of elements of a process area (expected, required and informative).

MATURE offers five different views of the process model that is being defined to provide a higher abstraction level to the user. The views built are exactly those explained in section 5 and are automatically generated and associated to the same XML[19] file. This means that one change on one view of the model affects the rest of the model and as a result, to the domain specific language associated too.

7 Case Study: Causal Analysis and Resolution (CAR)

The case study selected to present the MATURE tool, corresponds to the part of the process model that defines the Causal Analysis and Resolution (CAR) process area, explained in [5]. Using the Causal Analysis and Resolution process area, project members identify the causes of selected defects and other problems and take action to prevent them from occurring in the future. The final goal of the whole modeling process was the definition of a Domain Specific CMMI Language that supports computerized documentation tools in a precise and friendly way for the CMMI experts.

Taking into account the particular case presented in [5], the different concepts that formed this process area were identified(see Fig. 6). The selected Causal Analysis and Resolution process model has Verification (VER) as a related process area. The specific goal to achieve the implementation of the CAR process area is *"Determine Causes and Defects"*, the Specific Practice to implement that specific goal is *"Select Data for Analysis"*. This specific practice has a Subpractice, *"Gather relevant defects or problem data"*, and a Typical Work Product, *"Defect and data selected for further analysis"*.

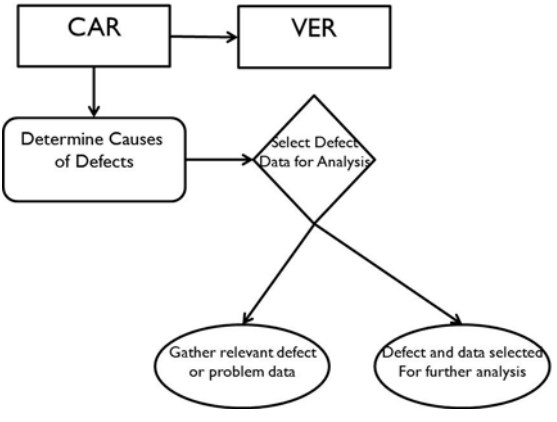

Fig. 6. Causal Analysis and Resolution model

The model explained above and shown in Fig. 6 was built in the MATURE framework to observe if MATURE offers the necessary support to build a domain specific model and, as a result, a domain specific language, without any special effort and expertise of programming knowledge, just knowing the domain.

A view to see the configuration of the model was developed (see Fig. 7). In this view, the ProcessAreas and the high level relationships between them (*PArelatesPA*) can be configured. When a process area is dropped onto the canvas, its name can be selected from a list of 22 process area names. In this case study, the Causal Analysis and Resolution and the Verification process areas are related.

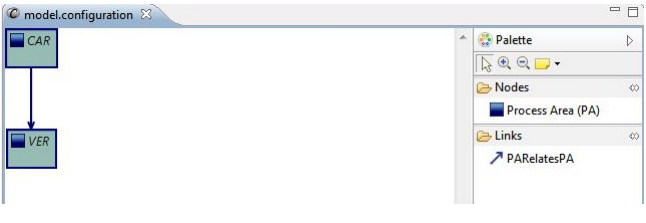

Fig. 7. Modeling of the configuration view of the model in MATURE

A view of the generic part of the model is shown in Fig. 8. The *ProcessArea* links a *SpecificGoal* with the relation *PAhasSG*. The *Specific Goal* links the *SpecificPractice* trough the relation *SGhasSP*. The SpecificPractice links the S*PSubpractice* and the *TypicalWorkProduct* with the relations *SPisdefinedBy-Subp* and *SPgeneratesTWP* respectively. Views for the generic part of the model are also provided in MATURE. In this case study those parts of the model were omitted.

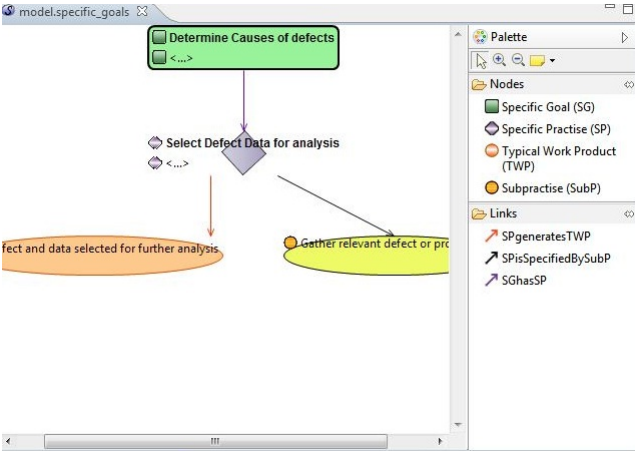

Fig. 8. Modeling of the Specific Part of the model with MATURE

The Domain Specific CMMI Language is automatically generated as an underlying XML document that can be seen in tree view in Fig. 9. This document will be the input, after some model transformations, to a documentation environment.

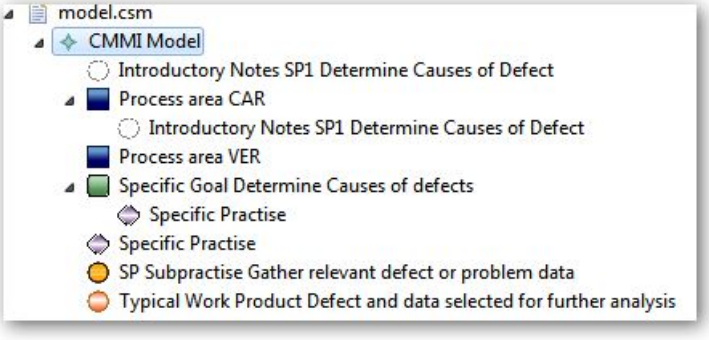

Fig. 9. Generated Domain Specific Language in tree format

8 Conclusions and Future Works

MATURE is a tool that is embedded in a Model Driven Development process and provides a framework that lets the user translate the CMMI generic model into a domain specific model, generating automatically a Domain Specific Language with multiple possibilities of transformation.

MATURE has been applied to a case study. It has been possible to show how MATURE accomplishes all the automation needs of the CMMI approach. A domain specific language representing a CMMI model has been created. This language can be further transformed depending on the needs of the user, working as input for many different tools and it can be graphically redefined.

The application of our proposal to CMMI constellations provides the following benefits: (i) having an assistant tool with graphical representation of the CMMI model, (ii) automate the process improvement model, being possible to customize it according to user needs, and limiting the derived documentation instead of having a bunch of papers, with all the benefits it implies, (iii) integrating CMMI in the MDD approach and in the MOF infrastructure, (iv) once the model is built, the resulting domain specific language can be further transformed so that many other different tools can be integrated, (v) building of models using metamodeling primitives guarantee the model correctness; (vi) a friendly graphical metaphor to establish variability in architectural components.

As future work, the support of the CMMI constellation metamodel is planned to be extended. As a consequence, MATURE will support other different parts of the CMMI constellations. One step upwards will be to specify a meta-metamodel for covering some other process improvement existing models such as ISO/IEC 15504[20] or ITIL[21]. In the mid term, the intention is to use MATURE to support software development innovation assesment models.

Acknowledgment

The work reported here has been partially sponsored by the Spanish MEC (DSDM TIN2008-00889-E), and MICINN (INNOSEP TIN2009-13849).

References

1. Tennant, G.: SIX SIGMA: SPC and TQM in Manufacturing and Services, p. 6. Gower Publishing, Ltd. (2001) ISBN 0566083744
2. Womack, James, P., Daniel, T. J. (eds.): Lean Thinking. Free Press, New York (2003)
3. Cox, J., Goldratt, Eliyahu, M.: The goal: a process of ongoing improvement. North River Press, Croton-on-Hudson (1986)
4. Bamford, R., Deibler, W.: ISO 9001: 2000 for Software and Systems Providers: An Engineering Approach, 1st edn. CRC-Press, Boca Raton (2003) ISBN 0849320631, ISBN 978-0849320637
5. CMMI® for Development Version 1.2, Software Engineering Institute, Carnegie Mellon University (August 2006)

6. Callis Author, http://www.callis.dk/resources/docs/ProductSheet%20-%20Callis%20Author%20ver.%202.1.pdf
7. Interim, http://www.man-info-systems.com/index_files/FreeTools.htm
8. Appraisal Assistant, http://www.sqi.gu.edu.au/AppraisalAssistant/about.html
9. Beydeda, S., Book, M., Gruhn, V.: Model-Driven Software Development. Springer, Heidelberg (2005)
10. Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Inst. for Software-Integrated Syst., Vanderbilt Univ., Nashville, TN, USA
11. Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. 37(4), 316–344 (2005), http://doi.acm.org/10.1145/1118890.1118892
12. Selic, B.: The pragmatics of model-driven development. IEEE Software 20(5) (September-October 2003)
13. Schmidt, D.C.: Model-Driven Engineering. IEEE Computer 39(2) (2006)
14. Object Management Group (OMG), Meta-Object Facility (MOF) 1.4 Specification. TRformal (2002-04-03)
15. Kleppe, A., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture Practice and Promise. Addison Wesley, Reading (2003)
16. GMF: The Eclipse Graphical Modeling Framework (GMF), http://www.eclipse.org/modeling/gmf/
17. Eclipse. Eclipse - an open development platform, http://www.eclipse.org
18. Eclipse Modeling Framework Project (EMF), http://www.eclipse.org/modeling/emf/
19. Extensible Markup Language (XML), W3C, http://www.w3org/XML/
20. ISO/IEC 15504, Information Technology, Process Assessment (2004)
21. ISO/IEC 20000, Service Management (2005)