An Integrated Approach for Model Driven Process Modeling and Enactment

Rita Suzana Pitangueira Maciel
Computer Science Department
Federal University of Bahia
Salvador - Bahia - Brazil
e-mail: ritasuzana@gmail.com

Bruno Carreiro da Silva, Ana Patrícia Fontes Magalhães
Faculdade Ruy Barbosa
Salvador Bahia Brazil
e-mail:{brunocarreiro, anapatriciamagalhaes}@gmail.com

Nelson Souto Rosa
Center of Informatics
Federal University of Pernambuco
Recife – Pernambuco - Brazil
e-mail: nsr@cin.ufpe.br

Abstract— **The adoption of MDA in software development is increasing and is widely recognized as an important approach for building software systems. However, there´s a lack of standard terminology and notation addressing design aspects of an MDA process. The available MDA tools and environments are particularly focused in defining and executing model transformations, while a development process involves other important definitions which should be carried out during the process enactment. This paper presents an integrated approach for MDA process modeling and enactment based on specializations of some SPEM 2 concepts. To support and evaluate our approach a tool was developed and applied in two case studies.**

Keywords-Model Driven Software Process; Software Process Metamodel;Model Driven Development; SPEM 2.0; MDA

I. INTRODUCTION

The description of a software process is called a process model. In many cases software processes adopted in organizations are poorly documented, or not documented at all. Even in cases where organizations keep their process specification and documentation they maintain and define process elements in *ad hoc* manner. The lack of a consistent and standard terminology with distinct notations and natural language negatively affects process comprehension and communication among the stakeholders. In such context process evolution is more difficult and enactment even more distant from the specification. Consequently, the process specification becomes obsolete and expensive to maintain. This can also negatively affect the team productivity and the quality of the software product [4; 23].

The need for a language or notation especially for process modeling is not new. In recent years several researchers have worked on the proposal of Process Modeling Languages (PMLs). A PML is a particular language to model and describe software processes. In this context, PMLs can be used for different purposes [4]: process understanding, design, training and education, simulation and optimization, support and enactment.

Several PMLs, such as APEL [2], PROMENADE [3], E^3 language [9] etc. were proposed in the last decade. The most recent initiative towards standardization is SPEM (*Software Process Engeneering Metamodel Specification)* [18]. SPEM 2.0 is a standard proposed and maintained by the Object Management Group (OMG). It is a metamodel based on MOF (Meta Object Facility) used to specify software processes. It also defines a UML profile in order to provide a mechanism to model processes with the UML language. There are well known development processes such as RUP (*Rational Unified Process*) which have been modeled using SPEM as a PML [11].

Another proposal focusing on software productivity and quality is Model Driven Architecture (MDA) [17], which is also an OMG standard. The MDA specification follows the Model-Driven Engineering (MDE) philosophy which shifts the focus of development activities to models and transformations leading to code generation.

In contrast to traditional development process models, an MDA process requires the selection of metamodels and mapping rules for the generation of the transformation chain which produces models and application code [14]. Thus, such a kind of process is conceptually driven by process automation. Once an MDA process is designed, it should be enacted several times. However, current MDA supporting tools are particularly interested just in defining and executing transformations which produce code and deployment artifacts from models (OptimalJ[1], AndroMDA[2], oAW[3] and others). Indeed, other activities in a software process are usually not considered. Tools for process modeling and specification (e.g. EPF[4]) are not integrated into a UML modeling tool nor a model transformation engine.

The initiative to integrate process design and enactment has been investigated [5]. Environments called PSEEs (Process-Centered Software Engineering Environment) have been proposed with different characteristics, features and contexts [1; 5]. However, most of them have shortcomings regarding process enactment. Some of them have enactable but proprietary and non-standard PML [25]. In some cases they have a restricted focus on the management view of a software development process alone [12]. To the best of our

[1] Compuware Corporation Home Page - http://www.compuware.com/
[2] AndroMDA.org Home - http://www.andromda.org/
[3] Official openArchitectureWare Home page - http://www.openarchitectureware.org/
[4] Eclipse Process Framework - http://www.eclipse.org/epf/

knowledge there is not a PSEE which supports the enactment peculiarities of an MDA process, i.e. an integrated environment including modeling and metamodeling, definition and execution of model transformations.

Our work presents an integrated approach for MDA process modeling and enactment. In order to achieve a standard notation we have specialized some of the SPEM 2.0 concepts to address MDA process definitions explicitly. We also designate a set of diagrams to model the MDA process structure and behavior. Therefore, during the process enactment tools can provide specific features regarding an MDA process (UML modeling, UML profile application, model transformations and code generation) besides the common features of any software process (role assignment, monitoring management, configuration and change management, measurement etc). Finally, we have developed an environment to support our approach.

We have previously presented an approach for the modeling of MDA processes [15], which was the first step in the direction of this work. Here, we present the integration of MDA process modeling with its enactment including a supporting environment and case studies. The text is organized as follows: in Section II we discuss the current approaches and tools for MDA processes; then in Section III we present our approach for MDA process modeling and enactment, including a supporting environment; In Section IV we describe two case studies we have carried out to assess our approach and finally in Section V we present conclusions and future works.

II. CURRENT APPROACHES, TOOLS AND ENVIRONMENTS FOR MDA PROCESSES

In recent years a number of research initiatives related to MDA have emerged. We divided these approaches in two categories: processes and methodologies for MDA; and languages and tools for model transformation.

Since the MDA standard does not define a software process to apply its concepts, naturally several works have concentrated on the specification of such processes including the definition of metamodels, modeling levels (CIM – Computational Independent Model, PIM – Platform Independent Model and PSM – Platform Specific Model) and model transformations. For instance, some of these proposals include MDA process for middleware specific services [14], MDD process for web applications [10], MDA methodology for e-learning systems [24] and fault tolerance distributed software families [6].

However, most of the approaches for MDA processes and methodologies are defined using a non-standard notation and language. Most of them are specified imprecisely in natural language with supplementary pictures and diagrams. In fact, there is a lack of consistent terminology since there is no unified language to specify MDA processes: each one adopts *ad hoc* notations and different concepts are used to define the activities and artifacts for the software development life cycle. Software process modeling using a unified and consistent terminology should make communication, understanding, reutilization, evolution, management and standardization of the process possible [7].

There is also a model-driven development process called OpenUP/MDD [19], which is a variation of the Open Unified Process for MDD. This process was specified according to the SPEM 2.0 standard using the EPF tool. The EPF provides an environment for software process modeling following SPEM. The OpenUP/MDD is a kind of generic model-driven development process defined to be customizable for each specific context. As a result, it is an instance (i.e. a metamodel instance) of the SPEM metamodel. Therefore, the only difference and advantage in contrast with the aforementioned MDA process and methodologies is that it is defined using the SPEM standard terminology and notation.

The second research initiative is related to model transformation, which is an essential activity for a model-driven development. Several languages for model transformation specification have been proposed and also a number of transformation engines to carry out the transformations. At present, there is a variety of open source and proprietary MDD/MDA tools with different characteristics and features. We can cite some interesting and mature tools such as: ATL[5] language and transformation engine; Mofscript language and transformation engine[6]; OptimalJ as a representative of a MDA proprietary tool; oAW framework; AndroMDA for model to code transformations, among others.

In spite of the high number of MDA tools already proposed as well as those used both in academia and industry, most particularly focus on model transformation execution and not on process design aspects. In other words, current MDA supporting tools are specifically interested in defining and executing transformations which produce code and deployment artifacts from models. As a result, users can only access the transformation chain. However, a development process involves other important definitions which should be carried out during the process enactment such as requirements analysis, testing, manual tasks etc. than just execute model transformations.

Despite the existence of these tools, some problems emerge due to the lack of integration among them. For instance, tools for process modeling and specification such as EPF can not be integrated into a UML modeling tool nor a model transformation engine. Furthermore, tools for UML modeling and model transformations encounter limitations in working together. They are usually restricted in terms of the interchangeability of models from different versions of the modeling language created in a bunch of available UML modeling tools.

[5] ATL Project - http://www.eclipse.org/m2m/atl/
[6] Mofscript Home Page - http://www.eclipse.org/gmt/mofscript/

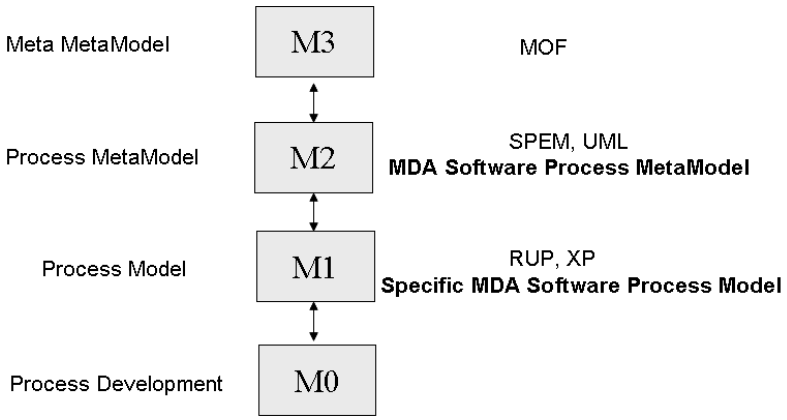

Figure 1. OMG model layers (adapted from [18]).

III. MDA PROCESS MODELING AND ENACTMENT

According to the OMG model layers shown in Fig. 1, a specific software development project is placed at level M0, i.e. the layer where a development team works on a project enacting a process which is specified in the level above (M1). RUP, XP, OpenUP/MDD and other processes are situated at M1. Process models in M1 are designed according to a process metamodel (i.e. a metalanguage to specify process models) which corresponds to level M2. For instance, SPEM was used to design the well-known RUP process model. As highlighted in Fig. 1, our approach is placed in level M2. Thus, an MDA process model (located in level M1), can be designed and will be available on the development of new projects in level M0. The proposed approach includes the following elements: (1) SPEM metamodel slice with a specialization of some concepts according to MDA; (2) indication of a set of diagrams for modeling the process structure and behavior.

The definition of MDA process concepts at metamodel level (M2) is important to provide a meaningful way to design software processes with explicit characteristics of an MDA process. Unlike the OpenUP/MDD process, we decided to add the MDA concepts at the metamodel level. Consequently, any MDA process definition modeled in M1 can be used during the process enactment in M0 providing specific features according to it. The MDA process metamodel, which is placed in level M2 is detailed in the next subsection.

A. MDA Process Modeling

Our approach is based on the metamodel illustrated in Fig. 2. This metamodel extends some of the SPEM 2 concepts specializing them for the MDA context.

The process specification needs static and reusable definitions such as *Disciplines, Tasks, Roles and WorkProducts* (from the method content package in Fig. 2). A *Role* defines a set of related skills, competencies and responsibilities of an individual or a set of individuals. Individuals should play their *Roles* performing *Tasks* that can be associated to input and output *WorkProducts*. A *Task* may comprise many *Steps* to describe a meaningful and consistent part of the overall work. The *Discipline* represents a collection of *Tasks* which are related to a major 'area of concern' within the overall project. *WorkProducts* are in most cases tangible artefacts consumed, produced, or modified by *Tasks.*

In our approach, the *WorkProduct* is specialized into four kinds of artefacts: the *UMLModel*, produced by a process role or automatically generated by a transformation during the process execution; *TransformationRule* contains the rules for model transformation and code generation during the process execution; *ExtraModel*, used only for documentation and are based on text or supplementary notations; and *Profile* to represent an UML profile to base the modeling on each phase. Transformation rules are used in MDA process to automatically transform UML models. Each transformation rule should refer to at least one source model and generate one or more target models. Based on the above definitions, the MDA process structure is specified according to the metamodel shown in the second part of Fig. 2. As illustrated, a *Process* has a life cycle composed of a set of sequential *Phases* performed in *Iterations*. In terms of MDA, these phases represent the modeling of CIM (Computational Independent Model), PIM (Platform Independent Model), PSM (Platform Specific Model) and Codification. Each *Modeling Phase* can be associated to UML profiles defined to address specific characteristics of a particular domain or platform.

Based on the metamodel presented in Fig. 2, an MDA process should be specified by the construction of three kinds of UML diagrams: class, use case and activity diagrams.

Table 1 presents the SPEM 2 stereotypes [18] (second column) extended in our metamodel and their usage in the three indicated UML diagrams (first column). The third column refers to the UML base element according to each SPEM stereotype. For example, in a use case diagram *Tasks* are modeled as use cases, while in the activity diagram they are modeled as actions.

.

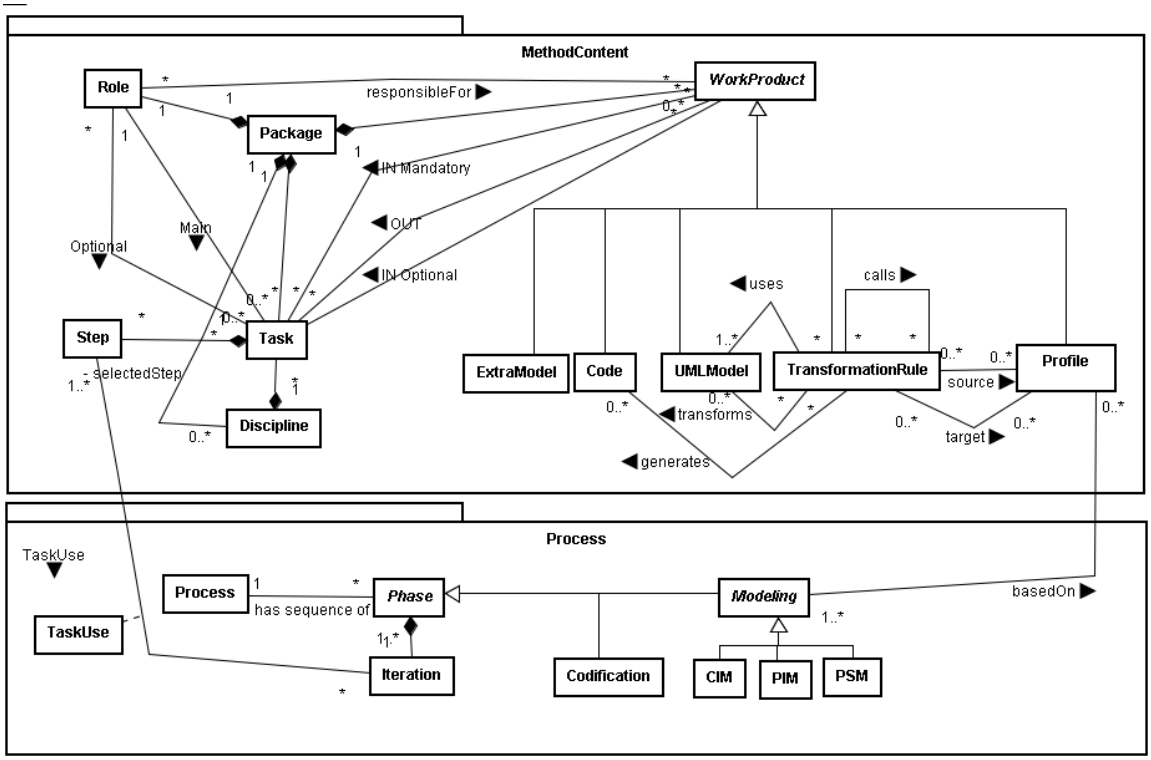

Figure 2. Specialized metamodel from some SPEM 2 concepts.

TABLE I. STEREOTYPES OF SPEM 2 ASSOCIATED TO UML DIAGRAMS AND ELEMENTS

Diagram	SPEM Stereotype	UML Element
Class	Package, Role, WorkProduct, Task, Step, Discipline, Phase, Iteration, TaskUse	Class
Use Case	Role	Actor
	Task, WorkProduct	Use Case
Activity	Task	Action
	Workproduct	Object

In this case, the class diagrams are used to specify the elements of a knowledge base (method content) and the process life cycle overall static structure. This is the first diagram that should be constructed as the elements are used to elaborate later diagrams. The use case diagrams are used to provide a specific view associating *Roles* to perform *Tasks* and also the used/produced *WorkProducts*.

The activity diagrams are used to model the process workflow, i.e, the behavior associated to the process execution in terms of *Phase/Iterations* and the selected *Steps* (*TaskUse*). This last diagram is also important because it defines when the transformations should be applied according to the workflow of tasks.

In summary, the model of an MDA process should contains definitions about phases and iterations, process roles, tasks, workproducts (consumed and produced) from different kinds, metamodels and transformation rules. All these definitions are input information for the process enactment supported by automation.

B. MDA Process Enactment

An MDA process is a software process too. There are manual, semi-automatic and automatic activities involved in the enactment of an MDA process. The use of a PML is essential to enable the process enactment with automation. Our approach provides a meaningful way to design MDA processes using a PML enabling further enactment of such a process with automated support. An environment for the enactment of a software process model is important to provide a variety of features such as software developer assistance, automation of routine tasks, invocation and control of software development tools, and enforcement of mandatory rules and practices [1].

The metamodel extension of the SPEM standard (explained in section III subsection A) was essential to enable the elaboration of process models with necessary and explicit information about MDA processes. By the definition of the process structure and behavior the enactment can be supported by a tool environment providing responsibilities assignment by roles definition to specific professionals in the software team, guidance to perform process tasks, management control, measurement, execution of model transformations, code generation and so on. All the necessary process information should be available in the MDA process model, which should follows the extended metamodel (Fig. 2) and the indicated diagrams (Table 1).

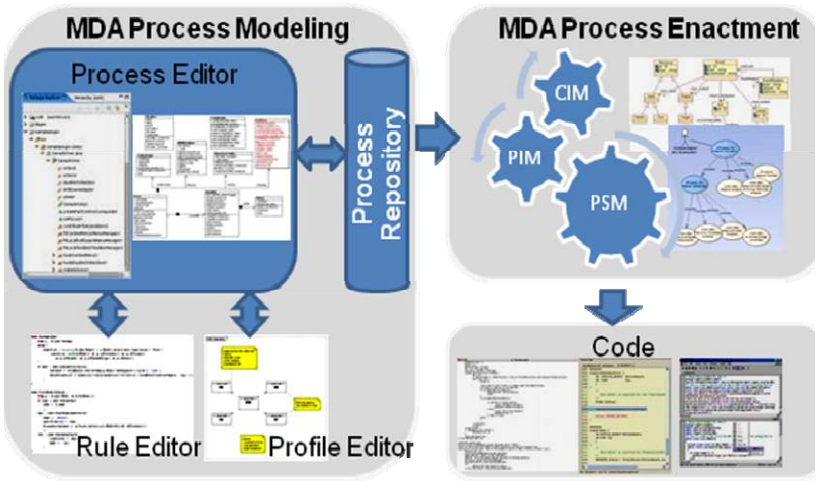

Figure 3. The Transforms Tool Overview.

C. Transforms: *Supporting Environment for MDA Process Modeling and Enactment*

An environment called *Transforms* has been developed to support the modeling and enactment of MDA processes according to our approach. This environment is divided into two main modules as illustrated in Fig. 3.

The first module provides design and customization of MDA processes offering a graphic editor to model the process structure and behavior. The set of graphic editors allows engineers to model their processes according to the proposed approach. It is also possible to specify a process using a breakdown structure and automatic generate diagrams to represent it graphically.

Besides these features, we integrated editors for two model transformation languages to write model-to-model and model-to-text transformations, and a UML profile editor to be used during the process specification. In this way, a user can create their own UML profiles and/or write their own transformation rules without going to other tools. All the process specification is stored in a process repository and so it becomes available to be enacted with automated support.

The other module is for the MDA process enactment. At this stage our environment provides the registration of professionals of a software team and the roles assigned to each one of them, the possibility to view all the process definitions (phases, iterations, artifacts etc.) including the tasks and their status, an integrated environment for UML modeling and model transformation execution, and a management view of process tasks and artifacts during the MDA software project execution. At the end of the process enactment the Transforms should provide all the created models by the project team, generated code and models by transformations, and the history of tasks (manually or automatically executed). Some screenshots and examples from a case study using our approach and the Transforms environment are presented in section IV.

IV. CASE STUDIES

This section presents two case studies we performed to evaluate the applicability of our approach in two different situations. First, in section IV subsection A we explain the modeling and enactment of an MDA process for development of specific middleware services. This case study was an initial effort to verify the feasibility of our approach. In section IV subsection B we present part of the experience in modeling a process of a real company which was important to assess the applicability of our work.

A. *An MDA Process for Specific Midleware Services*

Services Specific middleware services consist of a layer above the common middleware services that embody knowledge of a specific domain within the middleware. Domain-specific middleware services are not standardized. Their implementations are usually tightly coupled to the middleware platform. This implementation modeling requires considerable effort that certainly would not be rewarded if the service use were restricted to a specific middleware platform [21]. The MDA process goals presented in [13] encompass the specification and implementation of portable specific middleware services. This process was applied to the development of the InterDoc (Reference Architecture for Interoperable Services in Collaborative Writing Document Environments) [13; 14].

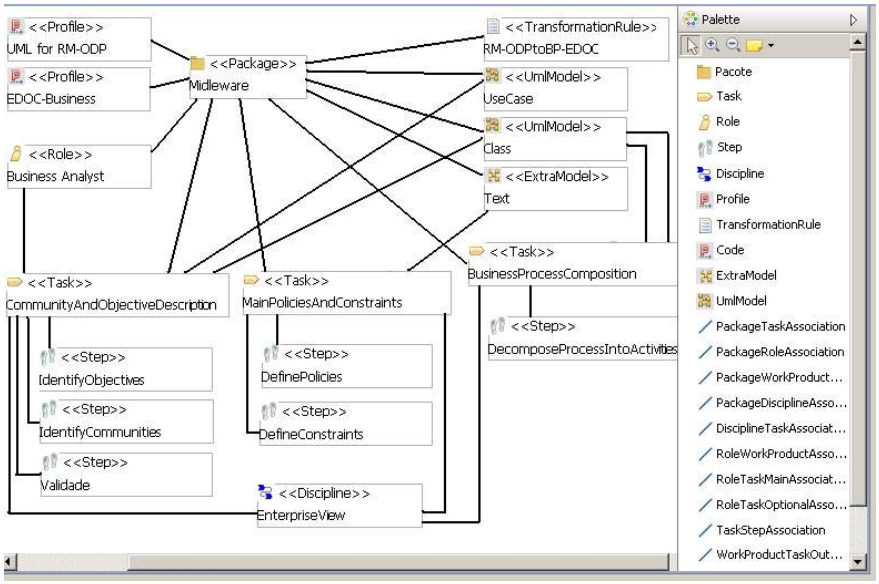

Figure 4. CIM phase static structure.

This MDA process includes the following elements: (1) Three modeling phases according to the MDA specification (CIM, PIM and PSM) (2) part of the EDOC (Enterprise Distributed Object Computing) [16] and UML for RM-ODP (Reference Model Open Distributed Processing) [8] metamodels (3) indication of a set of diagrams for each modeling phase (4) workflows to guide the modeling tasks and (5) mapping rules among the UML models to carry out the model transformations.

According to the MDA concepts the CIM Phase, called Domain Model, corresponds to the context in which the service should be applied. This category includes Enterprise and Information viewpoints. The PIM phase, called Design Model, describes the Computational view of the specified Domain model. The services to be offered and their operations are defined in this model. The Operational Model describes the application execution environment in a specific platform and corresponds to the specification of Engineering and Technology viewpoints. At this stage, the MDA process allows the definition of specific middleware services for CCM and EJB platforms.

The MDA process for specific middleware services was originally described without any standard language. Tables, illustrations and textual documents were used to represent the process specification. Tools were developed to support the automation of model transformations related to the process [22; 20]. However, the difficulty in understanding, reusing and evolving the process structure and behavior across development teams became evident.

In order to adopt the approach presented in section III, we mapped the process characteristics to the concepts and associations of the metamodel in Fig. 2. Six *disciplines* were defined to group related tasks: *Enterprise View*; *Information View*; *Computational View*; *Engineering View*; *Technology View*; and *Services Implementation*.

As described in section III, subsection A the class diagram is the first to be specified. It defines the overall structure of the MDA process. Two class diagrams were designed: one representing the *method content* and the other representing the process structure. Due to lack of space here, we present the method content class diagram for only a piece of the *Enterprise View discipline* and some of their related *Tasks, Roles* and *WorkProducts* in Fig. 4. Two UML profiles are used in the domain modeling (see their stereotype in the illustrated diagram): the *UML profile for RM-ODP*, and the EDOC UML profile. There might also be transformation rules to map elements from one phase to another, for example the *RM_ODPtoBP_EDOC* transformation. The *Enterprise View* discipline includes five tasks (not all illustrated in the diagram): *Community and Objective Description*, *Main Policies and Constraint Definition*, *Business Process Composition*, *Activities Definition* and *Behavior Definition*. Moreover, each activity may use one or more *WorkProducts* as input and/or produce new ones as output. For example, a use case and a class diagram are produced (see the *UmlModel* stereotype) as a result of the community and objective description.

Fig. 4 also illustrates the class diagram editor. The right palette organizes the necessary buttons to model the structural and static view of the method content. The editor only allows the modeling according to the metamodel defined in Fig. 2. The process life cycle is divided into three modeling phases (*CIM. PIM, PSM*) and codification. Each phase may comprise several *iterations* allowing incremental process development. *TaskUses* were selected, according to the steps previously defined in the method content, to be performed during the iterations. Fig. 5 shows the activity diagram for the CIM *phase* which is composed of a rich variety of *Tasks,* represented by their *TaskUses* and associated *WorkProducts*.

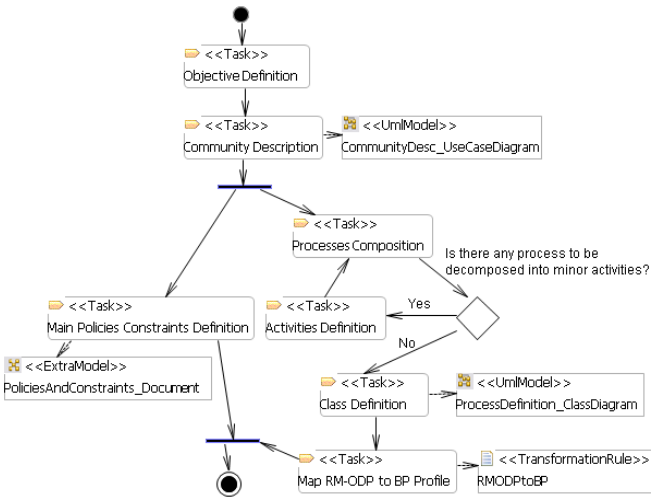

Figure 5. An activity diagram for the CIM phase.

The *Tasks* presented in Fig. 5 are included in the *Enterprise View Discipline* required to perform the CIM. The *activity* diagram starts with the domain *Objective Definition*, followed by the domain *Community Description*, which produces a *WorkProduct* (see the *UmlModel* stereotype). This artifact is a use case diagram related to the *Community Description*. Afterwards, the domain *Main Policies Constraints Definition* should be defined producing an artifact as result (see the *ExtraModel* stereotype in the illustrated diagram), which can be a text document or a diagram in any other additional notation. Concurrently, the *Process Composition Diagram* should be defined using a top-down approach, i.e. the business process is gradually broken down into fine-grained activities. The final task maps the models defined using the RM-ODP metamodel to a new representation based on the metamodel of the *BusinessProcess* profile. This last task comprises the execution of the model transformation.

Fig. 6 illustrates an ATL model transformation rule being edited on the Transforms MDA Process Editor. The left side depicts the process breakdown structure with its definitions, while in the right side the editor area is open with an ATL transformation rule called *BPtoCCA*.

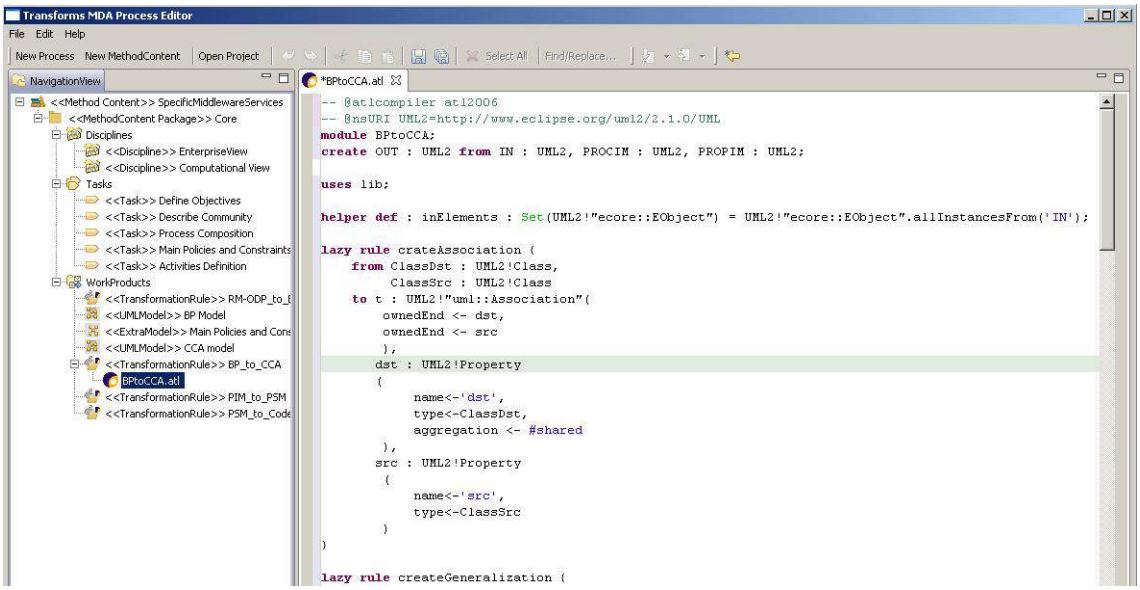

Figure 6. Creating a transformation rule on Transforms MDA Process Editor

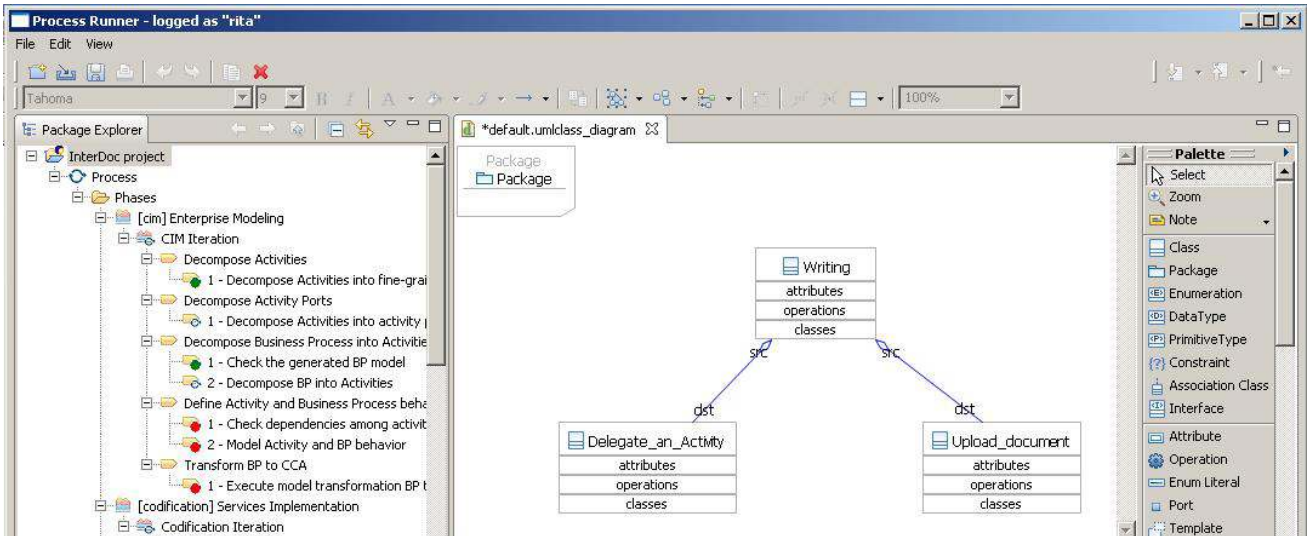

Figure 7. Creating a transformation rule on Transforms MDA Process Editor.

Fig. 7 shows a screenshot of the Transforms MDA Process Executor taken during the case study. The breakdown structure on the left side represents the project execution in terms of the process definitions previously specified using the Transforms Pocess Editor. Tasks marked in green were already performed, while tasks in red have not been started. The right side of the Transforms executor is used to edit UML model (as shown in the Fig. 7) and to describe the tasks execution. Also, it is possible to execute model transformations by just clicking on the corresponding tasks.

B. PRODEB Process for Web Applications

We have performed a case study with the Data-Processing Company of Bahia State in Brazil (PRODEB). This study involved the modeling of the PRODEB process for the development of web applications using the MDA approach.

PRODEB had been using the AndroMDA tool for a couple of months during the development of a web-based application. However, they encountered limitations related to the tool environment and especially because the process definitions (phases, activities, artifacts, roles, transformations etc.) were not specified and documented. Therefore, the process knowledge had not been registered so far. Furthermore, as MDA is an emerging technology, not all professionals were familiar wiht it. Most of the PRODEB staff did not know the reason why they had to stereotype UML elements or why to elaborate some models, which are necessary activities for the AndroMDA tool. In this context, we worked together with the professionals from PRODEB team for a couple of months in order to model the PRODEB process using our approach.

The class diagram in Fig. 8 partially illustrates the PRODEB process structure. In this figure, we show the CIM phase called *Business*, two iterations – *Data Collection* and *Documentation Refinement*, and also the tasks used in each iteration.

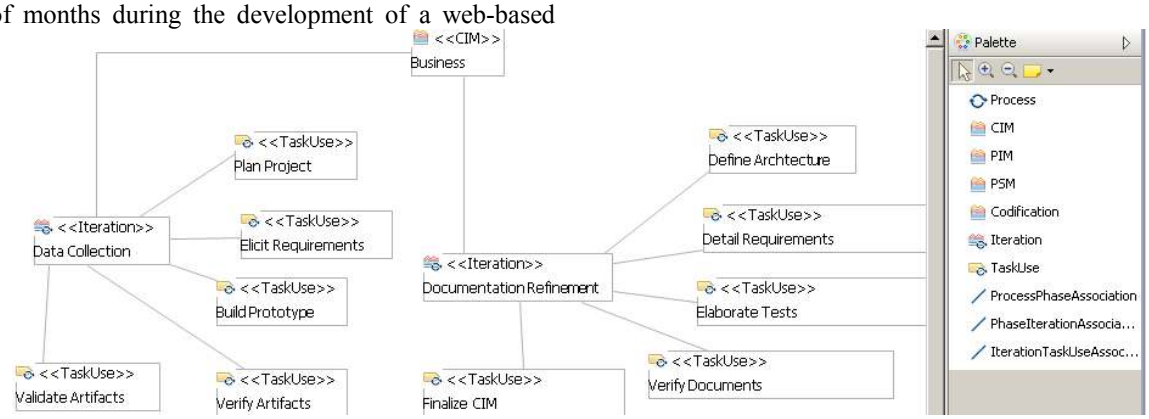

Figure 8. Partial class diagram for the business modeling phase.

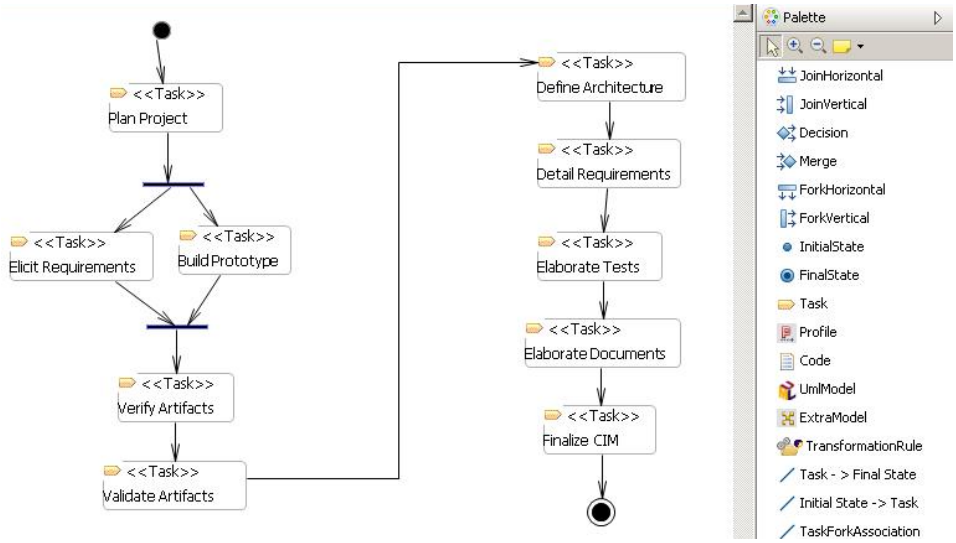

Figure 9. Activity diagram referring to the business modeling phase (CIM).

The activity diagram in Fig. 9 shows the workflow of tasks designed for the Business phase. All the tasks compose the behavior of the *Business* phase corresponding to the structural definition of the class diagram in Fig. 8. On the left side of Fig. 9, we organized the tasks from the *Data Collection* iteration, while on the right side we put the tasks from the *Documentation Refinement* iteration. The activity diagram initiates with the task *Plan Project*. Afterwards, the tasks *Elicit Requirements* and *Build Prototype* can be performed concurrently. Finally, the tasks *Verify Artifacts* and *Validate Artifacts* should be executed respectively in sequence. In the same phase, another iteration should be taken into account. The *Documentation Refinement* iteration comprises five tasks which are performed sequentially (*Define Architecture*, *Detail Requirements*, *Elaborate Tests*, *Verify Documents* and *Finalize CIM*).

C. *Results and Lessons Learned*

Based on the case studies, we have observed that difficulties in process comprehension, mostly in relation to the execution sequence of activities, were eliminated. At several moments developers gave suggestions on how to improve the specification of the current MDA process [20]. In particular, in the PRODEB case study, as the process was designed the professionals from PRODEB could better understand their own work and they used our meetings to discuss new definitions and elements to improve their process.

Additionally, we applied a questionnaire with 6 questions to collect data from their experience. The questionnaire was answered by professionals with different skills and functions in the company, such as business and system analysts, software designers and quality analysts. The results are summarized in Fig. 10.

We concluded that the results depicted in Fig. 10 were positive. Most of the answers ranged from *Good* to *Very Good*. There were only two absent answers and the *Insufficient* alternative wasn't checked for any question. Besides, we also collected commentaries which were optionally filled by the participants. Of course it is clear that this numbers are by no means statistically relevant. However, it represents just one of the important feedbacks we have gathered from the study. Also, we are considering new cooperative work to be conduced with PRODEB regarding another case study.

We can also compare an MDA process designed with SPEM 2.0 – e.g. the OpenUP/MDD - with those MDA processes we have presented in Section IV. In contrast to the OpenUP/MDD process, we found that our metamodel provides greater flexibility in the specification of MDA process elements. First, as we mentioned in Section III, the OpenUP/MDD is a process model at level M1, while our approach address the MDA process concepts at level M2. As a result, with OpenUP/MDD there is a process to be executed in a model driven development context, while with our approach there is a PML to specify model driven development processes according to some organization needs including the OpenUP/MDD itself.

Additionally, we found benefits during the enactment of the case study described in Section IV, subsection A. We could handle the process definitions designed based on our metamodel providing specific support for modeling, UML profile application, code generation, model transformation, task management and so on, using our environment. Therefore, we can conclude that our approach has contributed to the comprehension, evolution, reuse and enactment of the MDA processes we have worked on.

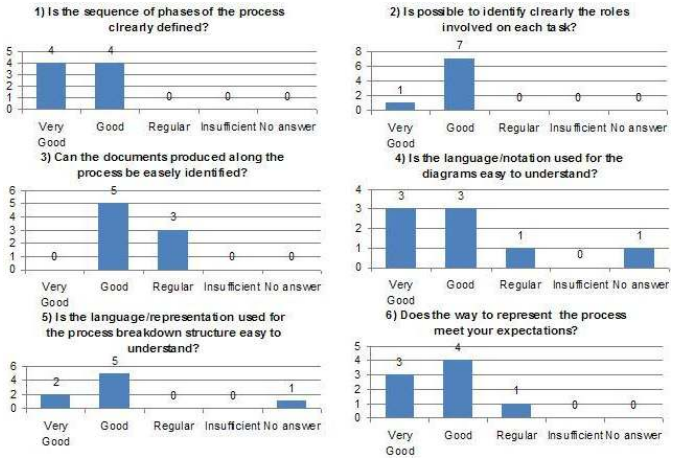

Figure 10. Results from the questionnaire.

V. CONCLUSIONS AND FUTURE WORKS

This paper has presented an approach for software process modeling and enactment based on the concepts of the SPEM 2 and MDA standards. We have specialized some of the SPEM 2 metamodel elements to provide a specific language to define model-driven processes according to the MDA. As the SPEM metamodel has a UML profile, our metamodel can be used through any UML modeling tool.

Our approach is based on several OMG standards (SPEM 2, UML 2, XMI and MDA), which guarantee the interoperability of models proposed by MDA. Therefore the methodologies and tools that assist in the development of applications and use these standards will have the same conceptual and notational framework. This aspect facilitates the understanding of models both by development teams and by process automation tools.

Moreover, we have developed an environment divided into two main parts. One for process modeling including: graphic editors specific for process modeling according to our metamodel, transformation rule editors, and also graphic forms and a breakdown structure as an alternative to process elaboration and organization. The second part of the environment is dedicated to the MDA process enactment which includes registration of the software team and role assignments, registration of task status, an integrated UML modeling tool and also engines for model transformation.

The contributions achieved in this current work represent a first effort towards a PSEE (Process-Centered Software Engineering Environment) for MDA software processes. Our ongoing and future work encompasses support for configuration management, providing traceability mechanisms across the process artifacts, and integrating an approach for model-driven testing. We are also planning new case studies to perform more in depth evaluations involving quantitative assessment with new goals and new target processes.

ACKNOWLEDGMENT

We would like to thank the PRODEB team and all the professionals who collaborated to us during the case study. This work is partially funded by Fapesb, project number 8694/2006, and grant number 0020/2006.

REFERENCES

[1] Ambriola, V., Conradi, R., and Fuggetta, A. (1997). Assessing process-centered software engineering environments. *ACM Trans. Softw. Eng. Methodol.* p. 283-328.

[2] Dami, S., Estublier, J. and Amiour, M. (1998) APEL: a Graphical Yet Executable Formalism for Process Modeling. *Automated Soft. Eng.*. Vol 5, January, pp. 61-96.

[3] Franch, X. and Ribó, J. M. (1999). "Using UML for Modelling the Static Part of a Software Process". In: UML99 - Beyond the Standard 2nd Int'l Conf., Fort Collins - USA, October 28-30.

[4] Fuggetta, A. (2000). "Software Process: A Roadmap". In: Proceedings of the Intl. Conference on the Future of Software Engineering. ACM, New York, NY, p. 25-34.

[5] Gruhn, V. (2002). Process-Centered Software Engineering Environments, A Brief History and Future Challenges. *Ann. Softw. Eng.* 14, 1-4 (Dec. 2002), 363-382.

[6] Guelfi, N. *et al*. (2003) "DRIP Catalyst: An MDE/MDA Method for Fault-tolerant Distributed Software Families Development". In: OOPSLA Workshop on Best Practices for Model Driven Software Development. Canada.

[7] Humprey, W., Kelner,M. (1989). "Software Modeling: Principles of Entity Process Models". SEI. Software Engineering Institute, Carnegie Mellon University. Pittsburgh, Pennsylvania, (CMU/SEI-89-TR-2).

[8] ISO. (2004). Use of UML for ODP system specification. Working Draft. ISO/IEC JTC1/SC7, 2004.

[9] Jaccheri, M. L., Baldi M., Divitini M. (1999). "Evaluating the requirements of software process modeling languages and systems". In: Process Support for Distributed Team-based Software Development. In: *PDTSD99*, Orlando, Florida, p. 570-578, August.

[10] Koch, N. (2006). "Transformation Techniques in the Model-Driven Development Process of UWE". In: Workshop Proc. of the 6th intl. Conference on Web Engineering (Palo Alto, California). ICWE '06, vol. 155. ACM, New York, NY, 3.

[11] Kroll, P. (2003). "The RUP: An Industry-wide platform for Best practices". Available at: http://www.ibm.com/developerworks/rational/library/873.html.

[12] Lima, A. *et al*. (2006). Gerência Flexível de Processos de Software com o Ambiente WebAPSEE. In *20º Simpósio Brasileiro de Engenharia de Software – Sessão de Ferramentas*, Florianópolis - Brasil.

[13] Maciel, R., Ferraz, C., Rosa, N. (2005). "An MDA Domain Specific Architecture to Provide Interoperability Among Collaborative Environments". In 19th Brazilian Symposium on Software Engineering, PUC-RIO, p. 120-135.

[14] Maciel, R., Silva, B. C. and Mascarenhas, L. A. (2006). "An Edoc-based Approach for Specific Middleware Services Development". In: 4th Workshop on MBD of Computer Based Systems, Postdam, Germany. Proc. IEEE Press, p:135–143.

[15] Maciel, R., Silva, B., Magalhães, A. and Rosa, N. (2009). "An Approach to Model-Driven Development Process Specification". In: 11th International Conference on Enterprise Information Systems, Milan. Proc. ICEIS'09. p. 27-32.

[16] OMG. (2002). EDOC - UML Profile for Enterprise Distributed Object Computing Specification. OMG Adopted Specification (ptc/02-02-05).

[17] OMG (2003). MDA Guide. Version 1.0.1 (omg/2003-06-01).

[18] OMG (2008). Software Process Engineering Metamodel Specification, Version 2.0, (formal/08-04-01).

[19] OpenUP Component – MDD (2008). Available at: http://www.eclipse.org/epf/openup_component/mdd.php.

[20] Pasini, K. *et al*. (2008). "Uma Solução para Apoiar um Processo de Desenvolvimento Dirigido a Modelos Usando openArchitectureWare". In: *IX Free Software Workshop / 9th Intl. Forum of Free Software*, Porto Alegre, Brazil, p. 121-126.

[21] Schantz, R., Schmidt, D. (2001). Middleware for Distributed Systems: Evolving the Common Structure for Network-centric Applications. *Encyclopedia of Software Engineering*, Wiley & Sons.

[22] Silva, B., Maciel, R. and Mascarenhas, L. (2006). "Transforms: Uma Ferramenta MDA/EDOC para Desenvolvimento de Serviços Específicos de Middleware". In: 20th Brazilian Symposium on Soft. Eng. – Tools Session. Florianópolis, p. 19-24.

[23] Sommerville, I. (2006). Software Engineering. 8th edition, Pearson Education.

[24] Wang, H., Zhang, D. (2003). "MDA-based Development of E-Learning System". In: 27th International Computer Software and Applications Conference, Texas. Proc. California: IEEE Press, p. 684.pi.

[25] Zamli, K. Z. and Lee, P. A. (2001). "Taxonomy of Process Modeling Languages". In: Proceedings of the ACS/IEEE international Conference on Computer Systems and Applications. IEEE Computer Society, Washington, DC, 435.