Supporting Non-functional Requirements in Services Software Development Process: An MDD Approach

Maria Valeria de Castro, Martin A. Musicante, Umberto Souza, Plácido A. Souza Di Neto, Genoveva Vargas-Solar

▶ **To cite this version:**

Maria Valeria de Castro, Martin A. Musicante, Umberto Souza, Plácido A. Souza Di Neto, Genoveva Vargas-Solar. Supporting Non-functional Requirements in Services Software Development Process: An MDD Approach. SOFSEM, Jan 2014, Novy Smokovek, Slovakia. pp.199-210. hal-01006226

HAL Id: hal-01006226
https://hal.science/hal-01006226

Submitted on 14 Jun 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Supporting Non-Functional Requirements in Services Software Development Process: An MDD Approach

Valeria de Castro[1], Martin A. Musicante[2], Umberto Souza da Costa[2], Plácido A. de Souza Neto[3], and Genoveva Vargas-Solar[4]

[1] Universidad Rey Juan Carlos – Móstoles, Spain
Valeria.deCastro@urjc.es
[2] Federal University of Rio Grande do Norte (UFRN) – Natal-RN, Brazil
{mam,umberto}@dimap.ufrn.br
[3] Federal Technological Institute of Rio Grande do Norte (IFRN) – Natal-RN, Brazil
placido.neto@ifrn.edu.br
[4] French Council of Scientific Research (CNRS) – Grenoble, France
Genoveva.Vargas-Solar@imag.fr

Abstract. This paper presents the π-SODM method an extension to the Service Oriented Development Method (SOD-M) to support the development of services software considering their functional and non-functional requirements. Specifically, π-SODM proposes: *(i)* meta-models for representing non-functional requirements in different abstraction levels; *(ii)* model-to-model transformation rules, useful to semi-automatically refine Platform Independent Models into Platform Specific Models; and *(iii)* rules to transform Platform Specific Models into concrete implementations. In order to illustrate the use of this methodology the paper describes how its use to develop a proof-of-concept.

Keywords: MDD, Service Oriented Applications, Non-functional Properties

1 Introduction

Model Driven Development (MDD) [12] is a top-down approach for designing and developing software systems proposed by the Object Management Group (OMG)[5]. MDD provides a set of guidelines for structuring specifications using *models* that specify a software system at different levels of abstraction or *viewpoints*:
Computation Independent Models (CIM): This viewpoint represents the software system at its highest level of abstraction. It focusses on the system environment, and on its business and requirement specifications. At this moment of the development, the structure and system processing details are still unknown or undetermined.

[5] http://www.omg.org/mda.

Platform Independent Models (PIM): This viewpoint focusses on the system functionality, hiding the details of any particular platform.

Platform Specific Models (PSM): This viewpoint focusses on the functionality, in the context of a particular implementation platform. Models at this level combine the platform-independent view with the specific aspects of the platform to implement the system.

Besides the notion of model at each level of abstraction, MDD requires the use of *model transformations* between levels. These transformations may be automatic or semi-automatic and implement the refinement process between levels.

MDD has been applied for developing service oriented applications. In Service-Oriented Computing [19], pre-existing services are combined to produce applications and provide the business logic. The selection of services is usually guided by the functional requirements of the application being developed. Some methodologies and techniques have been proposed to help the software developer in the specification of functional requirements of the business logic, such as the Service-Oriented Development Method (SOD-M) [9].

Ideally, non-functional requirements such as security, reliability, and efficiency would be considered along with all the stages of the software development. The adoption of non-functional specifications from the early states of development can help the developer to produce applications that are capable of dealing with the application context. Non-functional properties of service-oriented applications have been addressed in academic works and standards.Dealing with these kind of properties involves the use of specific technologies in different layers of the SOC architecture, for instance during the description of service APIs (such as WSDL[8] or REST [13]) or to express service coordinations (like WS-BPEL [1]).

Protocols and models implementing non-functional properties assume the existence of a global control of the artifacts implementing the application. They also assume that each service exports its interface. So, the challenge of supporting non-functional properties is related to *(i)* the specification of the business rules of the application; and *(ii)* dealing with the technical characteristics of the infrastructure where the application is executed.

This paper presents πSOD-M a methodology for supporting the construction of service-oriented applications, taking into account both functional and non-functional requirements. The goal of the methdology are to: *(i)* improve the construction process by providing an abstract view of the application and ensure the conformance to its specification; *(ii)* reduce the programming effort through the semi-automatic generation of models for the application, to produce concrete implementations from high abstraction models. Accordingly, the remainder of the paper is organized as follows: Sections 2 and 3 present, respectively, the SOD-M method of service software process and πSOD-M our proposed extension to deal with non-functional requirements. A proof of concept is developed in Section 4. Section 5 describes related works. Section 6 concludes the paper and gives final remarks.

2 SOD-M

The Service-Oriented Development Method (SOD-M) [9] adopts the MDD approach to build service-based applications. SOD-M considers two points of view: *(i) business*, focusing on the characteristics and requirements of the organization, and *(ii) system requirements*, focusing on features and processes to be implemented in order application requirements. In this way, SOD-M simplifies the design of service-oriented applications, as well as their implementation using current technologies.

SOD-M provides a framework with models and standards to express functionalities of applications at a high-level of abstraction. SOD-M meta-models are organized into three levels: CIM (*Computational Independent Models*), PIM (*Platform Independent Models*) and PSM (*Platform Specific Models*). Two models are defined at the CIM level: *value model* and *BPMN model*. The PIM level models the entire structure of the application flow, while, the PSM level provides transformations towards more specific platforms. The PIM-level models are: *use case*, *extended use case*, *service process* and *service composition*. The PSM level models are: *web service interface*, *extended composition service* and *business logic*.

The *value model* is a business model that describes a business case as a set of values and value activities shared by business actors. The *BPMN model* (business process model) is used to describe the business process related to the environment which the system will run. These two models represent the independent aspects of computing. The *use case model* is used to represent the business services to be implemented by the system, while the *extended use case model* is a behavioral model, to represent the system features as a way to implement the business services. The *service process model* describes the set of activities that must be performed on the system to implement a business service. Finally, the *service composition model* represents the full flow of business system. This model is an extension of the service process model, however, in more detail. These four models represent the platform independent aspects.

The SOD-M approach includes transformations between models: *CIM-to-PIM, PIM-to-PIM* and *PIM-to-PSM* transformations. Given an abstract model at the CIM level, it is possible to apply transformations for generating a model of the PSM level. In this context, it is necessary to follow the process activities described by the methodology. These three SOD-M levels have no support for describing non-functional requirements.The following section introduces π-SODM the extension proposed for considering these requirements.

3 πSOD-M

πSOD-M provides an environment for building service compositions considering their non-functional requirements. πSOD-M proposes the generation of a set of models at different abstraction levels, as well as transformations between

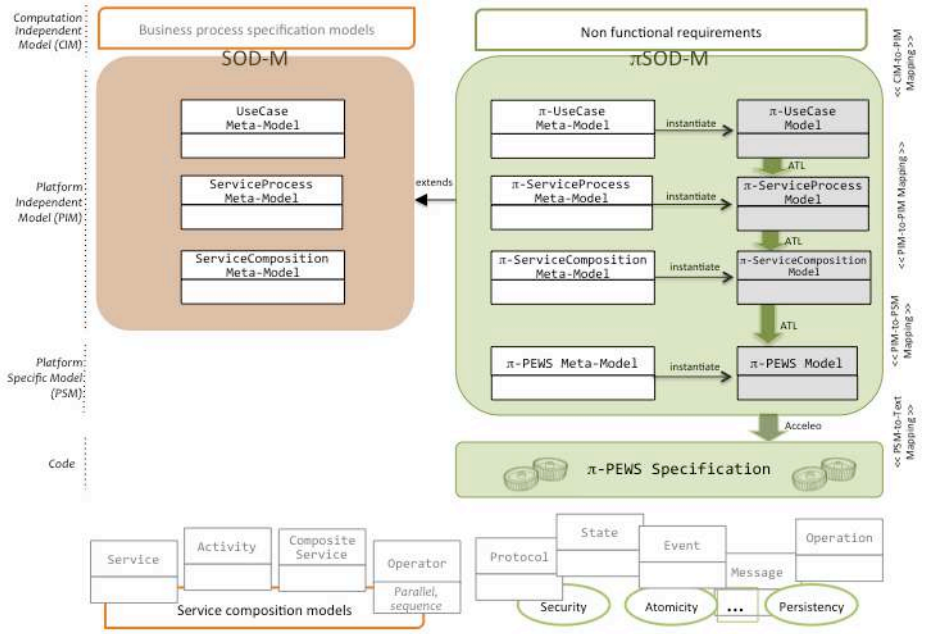


Fig. 1: πSOD-M.

these models. πSOD-M includes non-functional specifications through four meta-models that extend PIM SOD-M meta-models (see Figure 1): *π-UseCase*, *π-ServiceProcess*, *π-ServiceComposition* and *π-PEWS*.

The *π-UseCase* meta-model describes functional and non-functional requirements. Non-functional requirements are defined as *constraints* over processing and data. The *π-ServiceProcess* meta-model defines the concept of *service contract* to represent restrictions over data and actions that must be performed upon certain conditions. The *π-ServiceProcess* meta-model gathers the constraints described in the *π-UseCase* model into contracts that are associated with services. The *π-ServiceComposition* meta-model provides the concept of *Policy* [11] which put together contracts with similar non-functional requirements. For instance, security and privacy restrictions may be grouped into a security policy. *π-ServiceComposition* models can be refined into PSMs. Policies are associated to service operations and combine *constraints* and *reactive recovery actions*. Constraints are restrictions that must be verified during the execution of the application. Failure to verify the constraints will trigger exceptions to execute their corresponding recovery actions. An example of policy is the requirement of authentication for executing some of the system functions. The action associated to this policy may perform the authentication of the user. The *π-PEWS* meta-model is a PSM (see Figure 1). At the PSM level we have

lower-level models that can be automatically translated into actual computer programs. The π-*PEWS* meta-model is the PSM adopted in this work. π-*PEWS* models are textual descriptions of service compositions that can be translated into PEWS [3] or BPEL [1] code. Although PEWS is our language of choice, other composition languages can be used as target.

Thus, πSOD-M proposes a development process based on the definition of models (instances of the meta-models) and transformations between models. There are two kinds of transformations: Model-to-model transformations are used during the software process to refine the specification. Model-to-text transformations are the last step of the process and generate code.

πSOD-M environment is built on the top of Eclipse. We also used the Eclipse Modelling Framework (EMF) to define, edit and handle (meta)-models. To automate the transformation models we use ATL [14] and Acceleo [17].

In the next section we develop an example, to serve as a proof-of-concept. The example will show the actual notation used for models.

4 Proof of Concept: *Tracking Crimes*

Consider a tracking crime application where civilians and police share information about criminality in given zones of a city. Civilian users signal crimes using Twitter. Police officers can notify crimes, as well as update information about solving cases. Some of these information are confidential while other can be shared to the community of users using this application. Users can track crimes in given zones. Crime information stored by the system may be visualized on a map. Some users have different access rights than others. For example, police officers have more access rights than civilians.

In order to provide these functionalities, the application uses pre-existing services to provide, store and visualize the information. The business process defines the logic of the application and is specified in terms of tasks. Tasks can be performed by people or computers.

The business process and requirements specifications presented in Figure 2 are instances of the Computation-Independent models of Figure 1. The business process is represented as a graph while requirements are given as text boxes.

In our example, crime processing can start with one of two tasks: *(i) notify a crime*, or *(ii) track a crime*. Notified crimes are stored in a database. Tracked crimes are visualized in a map. The used can ask for detailed information. The application is built upon four services: twitter and an ad-hoc police service for notifying crimes, Amazon used as persistence service and Google Maps for locating and displaying crimes on a map.

Non-functional requirements are specified by rules and conditions to be verified during the execution of tasks. In our example we have the following non-functional requirements:

1. Twitter requires authentication and allows three login failures before blocking.
2. Crime notification needs privileged access.

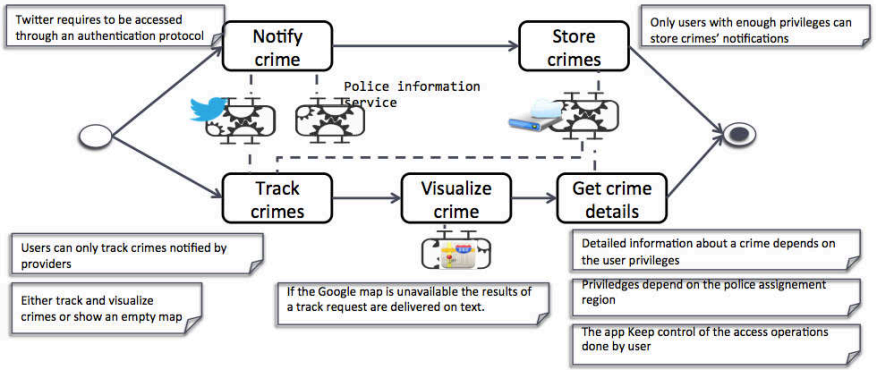


Fig. 2: Business process for the tracking crime example.

3. Civilian users can only track crimes for which they have clearance: Civilian population cannot track all the crimes notified by the police.
4. If Google Maps is unavailable, the results are delivered as text.
5. Querying about crimes without having proper clearance yields an empty map.
6. Access rights to detailed information depends on user clearance and zone assignment for police officers.
7. The application maintains a detailed log.

The idea about these requirements is to leave the application logic expressed by functional requirements as independent as possible from exceptional situations like the unavailability of a service and the conditions in which services are called for example through an authentication protocol. These requirements can be weaved as activities and branches of the composition or kept independent. The interest of the second option is that the maintenance and the evolution of the application logic can be easier. For example, the services called by the application are not hard coded (Twitter and Google Maps in the example), neither the actions to deal with exceptions (replacing anohter Map service or doing nothing).

Considering the example of tracking crimes, all the system restrictions are modelled as constraints. π-SODM provides three types of constraints: *value*, *business* and *exceptions behaviour* constraints. Each use case (model) can be associated to one or more constraints [6].

π-UseCase model: In our example we have five use cases (Figure 3), which represent the system functions (tasks) and constraints. We will not detail the functional part of the specification, due to lack of space. The constrains defined for our tracking crime example are:

- The Notify crime task requires that the user is logged in. This is an example of a *value constraint*, where the value associated to the condition depends on the

[6] For a more comprehensive account of πSOD-M the reader can refer to [22].

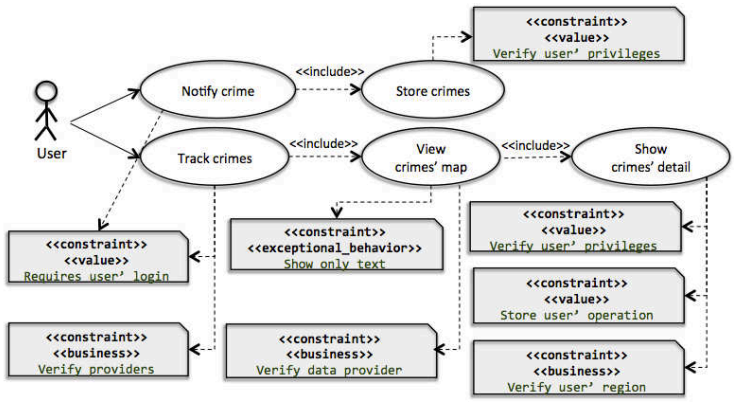


Fig. 3: π-UseCase Model

semantics of the application. In this case, it represents the maximum number of allowed login attempts;

- The Store crimes task requires the verification of the user's clearance (also a value constraint).
- In order to perform the Track crimes task, it is necessary that the notifier user is in the contact list of the requesting user. This is an example of *business constraint*. Additionally the requesting user must be logged in.
- For the View Crime Map task, the specification defines that if the Google Maps service is not available, the result is presented as text. This is an example of *exceptional behaviour constraint*. The availability of the Google Maps service is verified by a *business constraint*.
- The Show crime details task is specified to have three constraints: A *value constraint* is defined to verify the user's clearance level; A *business constraint* is used to ensure that the user's clearance is valid for the geographic zone of the crime; Another *value constraint* defines that the log is to be maintained.

π-ServiceProcess model: The model presented in Figure 3 is transformed, at this stage of the development, into a similar graph, where *(i)* the task nodes are better detailed, by refining the control and data flows; and *(ii)* constraints are transformed into *contracts* (pre- and post-conditions). The new model describes the application's activities and defines contracts for each activity or for parts of the application.

A model-to model transformation is defined in order refine the *π-UseCase* model of the application into the more detailed model. This (semi-automatic) transformation process is supported by a tool (described in [22]).

The *π-ServiceProcess* model defined for our tracking crime application is presented in Figure 4, where:

1. Tasks of the previous model are transformed into *actions*;

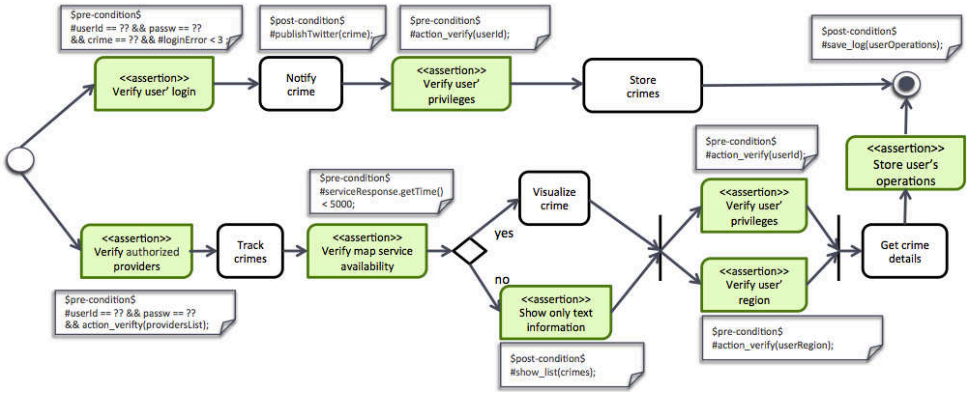


Fig. 4: π-*ServiceProcess* Model

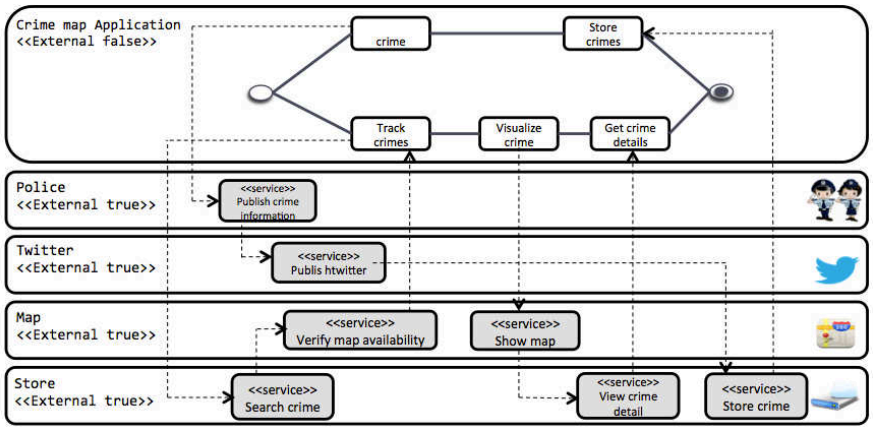
2. Actions are grouped into *activities* (in accordance to the business logic).
3. Constraints of the π-*UseCase* model are transformed into assertions.

π-*ServiceComposition model:* This model refines the previous model by using the activities to produce the workflow of the application. The model serves to identify those entities that collaborate with the service process by providing services to execute actions. This model identifies the services and functions that correspond to each action in the business process.

In the case of our crime tracking example, the model produced from the π-*ServiceProcess* model of Figure 4 is given in Figures 5a and 5b. Figure 5a shows how the crime application interacts with its *business collaborators* (external services and entities). The interaction occurs by means of function calls (denoted by dotted lines in the figure). Figure 5b shows the definition of three *policies*, which define rules for service execution. In our case we have policies for *Security*, *Performance* and *Persistence*.

π-*PEWS Model:* These models are produced by a model-to-text transformation that takes a π-*ServiceComposition* model and generates π-*PEWS* specification code. This code is a service composition program that can be compiled into executable code. π-*PEWS* models are expressed in a variant of the PEWS composition language. The π-PEWS program generated from the model in Figure 5 is partially presented in Figure 6. The figure shows a simplified program code, produced in accordance to the following guidelines:

1. Namespaces, identifying the addresses of external services are produced from the Business Collaborators of the higher-level model. We define four of them, corresponding to the Police, Twitter, Google Map and Amazon partners.
2. Specific operations exported by each business collaborator are identified to an *operation* of the program (Each operation is given an `alias`).



(a) π-ServiceComposition Model

```
policy class Compensation |[ Parallel-Operator scope ]|

    Status-Backup
        backup-Facebook, backup-Twitter;

    rule R1
        ON   Activity-Failure e1 AND
             Activity-Ended e2
        IF   e1.activityName == " update twitter " AND
             e2.activityName == " update facebook "
        DO   compensate(" update facebook " , backup-Facebook.status );

    rule R2
        ON   ...
```

```
policy class Status-Backup |[ Activity scope ]|

    String status;

    rule R1

        ON   Activity-Started event
        IF   event.activityName == scope.name
        DO   {
             status =    get-Status ( scope.name );
        }
```

(b) π-ServiceComposition Policies

Fig. 5: Service Composition and Policies.

3. The workflow in Figure 5a is translated into the text in line 11.
4. *Contracts* are defined in π-PEWS as having pre-conditions (`requires`), post-conditions (`ensures`) and actions (`OnFailureDo`) to be executed case a condition is not verified. Contracts are generated from Policies (such as those of Figure 5a.

5 Related work

Over the last years, a number of approaches have been proposed for the development of web services. These approaches range from the proposal of new languages for web service descriptions [1, 20] to techniques to support phases of the development cycle of this kind of software [6]. In general, these approaches concentrate on specific problems, like supporting transactions or QoS, in order to improve the security and reliability of service-based applications. Some proposals address service composition: workflow definition [24, 16] or semantic equivalence between services [3].

Works dealing with non-functional properties in service-oriented development can be organized in two main groups: those working on the *modeling of*

```
//Namespaces specify service URI
1 namespace twitter = www.twitter.com/service.wsdl
2 namespace googlemaps = maps.googleapis.com/maps/api/service
3 namespace amazondynamodb = rds.amazonaws.com/doc/2010-07-28/AmazonRDSv4.wsdl
4 namespace police = www.police.fr/service.wsdl
//Operations
5 alias publishTwitter = portType/publishTwitter in twitter
6 alias searchCrime = portType/searchCrime in amazondynamodb
7 alias showMap = portType/showMap in googlemaps
//Services
8  service notifyCrime = publishCrime  . publishTwitter
9  service trackCrime= searchCrime . verifyService
10 Service visualizeCrime =  showMap . getCrimeDetail
//Path
11 (notifyCrime.storeiCrime) || (trackCrime.visualizeCrime.getCrimeDetail)
//Contracts
12 defContract notifyCrimeContract{  isAppliedTo: notifyCrime
13   requires: userId == ?? && passw == ?? && req(notifyCrime)  < 3
14        (OnFailureDo: NOT(action_publish(crime));
15   ensures: publishTwitter(crime) == true  (OnFailureDo: skip); }
```

Fig. 6: π-PEWS code for the crime tracking example (partial, simplified).

particular non-functional properties or QoS attributes and those proposing *architectures or frameworks to manage and validate QoS attributes in web service composition processes*. The first group considers specific non-functional concerns (e.g. security) which is modelled and then associated to functional models of the application. The work of Chollet et al. [7] defines a proposal to associate non-functional quality properties (security properties) to functional activities in a web service composition model. Schmeling et al. [21] present an approach and also a toolset for specifying and implementing non-functional concerns in web service compositions. Non-functional concerns are modelled and then related to a service composition represented in a BPMN diagram. Ovaska et al. [18] present an approach to support quality management at design time. Quality requirements are modelled in a first phase and then represented in an architectural model where quality requirements are associated to some components of the model.

The second group of works dealing with non-functional requirements for services propose specific architectures or frameworks to manage and validate QoS attributes in service composition processes [25, 4, 15].

Despite the variety of techniques proposed, there is not yet a consensus on a software methodology for web services. Some methodologies address the service-based development towards a standard or a new way to develop reliable applications. SOD-M and SOMF [5] are MDD approaches for web services; S-Cube [19] is focused on the representation of business processes and service-based development; SOMA [2] is a methodology for SOA solutions; DEVISE [10] is a methodol-

ogy for building service-based infrastructure for collaborative enterprises. Other proposals include, the WIED model [23], that acts as a bridge between business modeling and design models, and traditional approaches for software engineering applied to SOC.

6 Conclusions

This paper presented the πSOD-M software method for specifying and designing service based applications in the presence of non-functional constraints. Our proposal enhances the SOD-M method with constraints, policies and contracts to consider non-functional constraints of applications. We implemented the proposed meta-models on the Eclipse platform and we illustrated the approach by developping a simple application.

πSOD-M is being used in an academic environment. So far, the preliminary results indicate that πSOD-M approach is useful for the development of complex web service applications. We are now working on the definition of a PCM-level meta-model to generate BPEL programs (instead of π-PEWS).

Acknowledgements

This research is partly supported by the National Institute of Science and Technology for Software Engineering (INES[7]), funded by CNPq (Brazil), grants 573964/2008-4 and 305619/2012-8; CAPES/UdelaR (Brazil/Uruguay) grant 021/ 2010; CAPES/STIC-AmSud (Brazil) grant 020/2010); MASAI project (TIN-2011-22617) financied by the Spanish Ministry of Science and Innovation and the Spanish Network on Service Science (TIN2011-15497-E) financed by the Spanish Ministry of Competitiveness and Economy.

References

1. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weeranwarana, S.: Bussiness process execution language for web services. Available at http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ (2003)
2. Arsanjani, A.: SOMA: Service-Oriented Modeling and Architecture. Technical report, IBM, http://www.ibm.com/developerworks/library/ws-soa-design1 (2004)
3. Ba, C., Halfeld-Ferrari, M., Musicante, M.A.: Composing web services with PEWS: A trace-theoretical approach. In: ECOWS 2006. (2006) 65–74
4. Babamir, S.M., Karimi, S., Shishechi, M.R.: A broker-based architecture for quality-driven web services composition. In: Proc. CiSE 2010. (2010)
5. Bell, M.: Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. John Wiley (2008)
6. Börger, E., Cisternino, A., eds.: Advances in Software Engineering (Revised Tutorial Lectures). In Börger, E., Cisternino, A., eds.: Lipari Summer School. Volume 5316 of LNCS., Springer (2008)

[7] www.ines.org.br

7. Chollet, S., Lalanda, P.: An extensible abstract service orchestration framework. In: Proc. ICWS 2009, IEEE (2009) 831–838
8. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web services description language (wsdl) 1.1. Technical report, World Wide Web Consortium (2001) Available in http://www.w3.org/TR/wsdl.
9. de Castro, V., Marcos, E., Wieringa, R.: Towards a service-oriented mda-based approach to the alignment of business processes with it systems: From the business model to a web service composition model. IJCIS **18**(2) (2009)
10. Dhyanesh, N., Vineel, G.C., Raghavan, S.V.: Devise: A methodology for building web services based infrastructure for collaborative enterprises. In: Proc. WET-ICE'03, USA, IEEE Computer Society (2003)
11. Espinosa-Oviedo, J.A., Vargas-Solar, G., Zechinelli-Martini, J.L., Collet, C.: Policy driven services coordination for building social networks based applications. In: Proc. of SCC'11, Work-in-Progress Track, USA, IEEE (2011)
12. Favre, L.: A rigorous framework for model driven development. In: Advanced Topics in Database Research, Vol. 5. Chapter I, IGP. (2006) 1–27
13. Fielding, R.T.: REST: Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine (2000)
14. Group, A.: Atl: Atlas transformation language. Technical report, ATLAS Group, LINA & INRIA (February, 2006)
15. Karunamurthy, R., Khendek, F., Glitho, R.H.: A novel architecture for web service composition. J. of Network and Computer Applications **35**(2) (2012) 787 – 802
16. Musicante, M.A., Potrich, E.: Expressing workflow patterns for web services: The case of pews. J.UCS **12**(7) (jul 2006) 903–921
17. Musset, J., Juliot, E., Lacrampe, S.: Acceleo référence. Technical report, Obeo et Acceleo (2006)
18. Ovaska, E., Evesti, A., Henttonen, K., Palviainen, M., Aho, P.: Knowledge based quality-driven architecture design and evaluation. Information & Software Technology **52**(6) (2010) 577–601
19. Papazoglou, M.P., Pohl, K., Parkin, M., Metzger, A., eds.: Service Research Challenges and Solutions for the Future Internet. In Papazoglou, M.P., Pohl, K., Parkin, M., Metzger, A., eds.: S-CUBE Book. Volume 6500 of LNCS., Springer (2010)
20. Salaün, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: Proc. IEEE International Conference on Web Services. ICWS '04, Washington, DC, USA, IEEE Computer Society (2004)
21. Schmeling, B., Charfi, A., Mezini, M.: Composing non-functional concerns in composite web services. In: Proc. ICWS 2011. (july 2011) 331 –338
22. Souza Neto, P.A.: A methodology for building service-oriented applications in the presence of non-functional properties. PhD thesis, Federal University of Rio Grande do Norte (2012) http://www3.ifrn.edu.br/∼placidoneto/thesisPlacidoASNeto.pdf.
23. Tongrungrojana, R., Lowe, D.: Wied: A web modelling language for modelling architectural-level information flows. J. Digit. Inf. **5**(2) (2004)
24. Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases **14**(1) (July 2003) 5–51
25. Xiao, H., Chan, B., Zou, Y., Benayon, J.W., O'Farrell, B., Litani, E., Hawkins, J.: A framework for verifying sla compliance in composed services. In: ICWS. (2008)