



HAL
open science

Generating reliable services' composition using A-policies: a model-driven approach

Genoveva Vargas-Solar, Valeria de Castro, Plácido Antonio A Souza Neto,
Javier A Espinosa-Oviedo, Esperanza Marcos, Martin A Musicante, José-Luis
A Zechinelli-Martini, Christine Collet

► To cite this version:

Genoveva Vargas-Solar, Valeria de Castro, Plácido Antonio A Souza Neto, Javier A Espinosa-Oviedo, Esperanza Marcos, et al.. Generating reliable services' composition using A-policies: a model-driven approach. the Jornadas de Servicios Web, Sep 2014, Cadiz, Spain. hal-01584802

HAL Id: hal-01584802

<https://hal.science/hal-01584802v1>

Submitted on 9 Sep 2017

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.

Generating reliable services' composition using A-policies: a model-driven approach*

Genoveva Vargas-Solar¹, Valeria de Castro⁴, Plácido Antonio de Souza Neto^{5,6},
Javier A. Espinosa-Oviedo³, Esperanza Marcos⁴, Martin A. Musicante⁵,
José-Luis Zechinelli-Martini^{2,1}, and Christine Collet³

¹ French Council of Scientific Research, LIG-LAFMIA, Grenoble, France

`Genoveva.Vargas-Solar@imag.fr`

² Fundación Universidad de las Américas, San Andrés Cholula, Puebla, México

`joseluis.zechinelli@udlap.mx`

³ Grenoble Institute of Technology, Grenoble, France

`Christine.Collet@grenoble-inp.fr`, `Javier.Espinosa@imag.fr`

⁴ Universidad Rey Juan Carlos, Móstoles, Spain

`Valeria.deCastro@urjc.es`, `esperanza.marcos@urjc.es`

⁵ DIMAp - UFRN, ForAll - Formal Methods and Language Research Laboratory,
Natal - RN, Brasil

`mam@dimap.ufrn.br`

⁶ Instituto Federal do Rio Grande do Norte, Natal - RN, Brasil

`placido.neto@ifrn.edu.br`

Abstract. This paper presents an approach for modeling and associating *A-Policies* to services' based applications. It proposes to extend the SOD-M model driven method with (i) the π -SCM an *A-Policy* services' composition meta-model for representing non-functional constraints associated to services' based applications; (ii) the π -PEWS meta-model providing guidelines for expressing the composition and the policies; and, (iii) model to model and model to text transformation rules for semi-automatizing the implementation of reliable services' compositions.

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

1 Introduction

Service oriented computing is at the origin of an evolution in the field of software development. An important challenge of service oriented development is to ensure the alignment between IT systems and the business logic. Thus, organizations are seeking for mechanisms to deal with the gap between the systems developed and business needs [2]. The literature stresses the need for methodologies and techniques for service oriented analysis and design, claiming that they

* This work is partially financed by the projects ORCHESTRA ECOS-ANUIES, CLEVER STIC-AMSUD, MASAI, NATIONAL SCIENTIFIC AND TECHNOLOGICAL NETWORK OF SERVICE SCIENCE and CAPES/STIC-AMSUD Brasil, BEX 4112/11-3.

are the cornerstone in the development of meaningful services' based applications [13]. In this context, some authors argue that the convergence of model-driven software development, service orientation and better techniques for documenting and improving business processes are the key to make real the idea of rapid, accurate development of software that serves, rather than dictates, software users' goals [19].

Service oriented development methodologies providing models, best practices, and reference architectures to build services' based applications mainly address functional aspects [1, 6, 18, 14]. Non-functional aspects concerning services' and application's "semantics", often expressed as requirements and constraints in general purpose methodologies, are not fully considered or they are added once the application has been implemented in order to ensure some level of reliability (e.g., data privacy, exception handling, atomicity, data persistence). This leads to services' based applications that are partially specified and that are thereby partially compliant with application requirements.

The objective of this work is to model non-functional constraints and associate them to services' based applications early during the services' composition modeling phase. Therefore this paper presents π -SOD-M, a model-driven method that extends the SOD-M [18] for building reliable services' based information systems. Our work proposes to extend the SOD-M [18] method with (i) the notion of *A-Policy* [10] for representing non-functional constraints associated to services' based applications. We also (ii) defines the π -PEWS meta-model [17] providing guidelines for expressing the composition and the *A-policies*. Finally, our work (iii) defines model to model transformation rules for generating the π -PEWS model of a reliable services' composition starting from the extended services' composition model; and, model to text transformations for generating the corresponding implementation.

The remainder of the paper is organized as follows. Section 2 gives an overview of our approach. It describes a motivation example that integrates and synchronizes well-known social networks services namely Facebook, Twitter and, Spotify. Sections 3, 4, and 5 describe respectively the three key elements of our proposal, namely the π -SCM and π -PEWS meta-models and the transformation rules that support the semi-automatic generation of reliable services' compositions. Section 6 analyses related work concerning policy/contract based programming and, services' composition platforms. Section 7 concludes the paper and discusses future work.

2 Modeling reliable services' compositions with π -SOD-M

Consider for instance the following scenario. An organization wants to provide the services' based application "To Publish Music" that monitors the music a person is listening during some periods of time and sends the song title to this person's Twitter and Facebook accounts. Thus, this social network user will have her status synchronized in Twitter and Facebook (i.e., either the same title is

published in both accounts or it is not updated) with the title of the music she is listening in Spotify. For developing this services' based application it is necessary to compose the following services calling their exported methods: a) The music service Spotify exports a method for obtaining information about the music a given user is listening: `get-Last-Song (userid): String` ; b) Facebook and Twitter services export a method for updating the status of a given user `update-Status (userid, new-status): String`.

The "To Publish Music" scenario starts by contacting the music service Spotify for retrieving the user's musical status (activity `Get Song`). Twitter and Facebook services are then contacted in parallel for updating the user's status with the corresponding song title (activities `Update Twitter` and `Update Facebook`).

Given a set of services with their exported methods known in advance or provided by a service directory, building services' based applications can be a simple task that implies expressing an application logic as a services' composition. The challenge being ensuring the compliance between the specification and the resulting application. Software engineering methods (e.g., [1, 6, 18, 14]) today can help to ensure this compliance, particularly when information systems include several sometimes complex business processes calling Web services or legacy applications exported as services.

2.1 Modeling a services' based application

Figure 1 shows SOD-M that defines a service oriented approach providing a set of guidelines to build services' based information systems (SIS) [18]. Therefore, SOD-M proposes to use services as first-class objects for the whole process of the SIS development and it follows a Model Driven Architecture (MDA) [15] approach. Extending from the highest level of abstraction of the MDA, SOD-M provides a conceptual structure to: first, capture the system requirements and specification in high-level abstraction models (computation independent models, CIMs); next, starting from such models build platform independent models (PIMs) specifying the system details; next transform such models into platform specific models (PSMs) that bundles the specification of the system with the details of the targeted platform; and finally, serialize such model into the working-code that implements the system. As shown in Figure 1, the SOD-M model-driven process begins by building the high-level CIMs and enables specific models PSMs for a service platform to be obtained as a result [18]. Referring to the "To Publish Music" application, using SOD-M the designer starts defining an E3value model ⁷ at the CIM level and then the corresponding models of the PIM are generated leading to a services' composition model (SCM).

Now, consider that besides the services' composition that represents the order in which the services are called for implementing the application "To Publish Music" it is necessary to model other requirements that represent the (i) conditions imposed by services for being contacted, for example the fact the Facebook

⁷ The E3 value model is a business model that represents a business case and allows to understand the environment in which the services' composition will be placed [9].

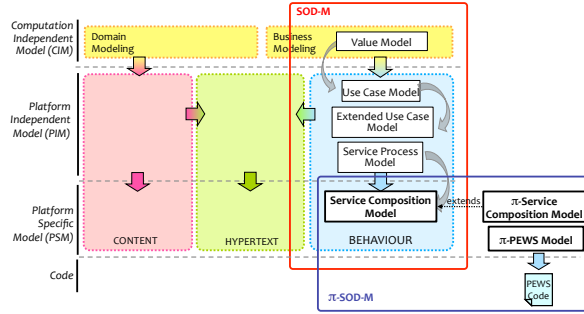


Fig. 1: SOD-M development process

and Twitter require authentication protocol in order to call their methods for updating the wall; (ii) the conditions stemming from the business rules of the application logic, for example the fact that the walls in Facebook and Twitter must show the same song title and if this is not possible then none of them is updated.

2.2 Modeling non-functional constraints of services' based applications

Adding non-functional requirements and services constraints in the services' composition is a complex task that implies programming protocols for instance authentication protocols to call Facebook and Twitter in our example, and atomicity (exception handling and recovery) for ensuring a true synchronization of the song title disseminated in the walls of the user's Facebook and Twitter accounts.

Service oriented computing promotes ease of information systems' construction thanks, for instance, to services' reuse. Yet, this is not applied to non-functional constraints as the ones described previously, because they do not follow in general the same service oriented principle and because they are often not fully considered in the specification process of existing services' oriented development methods. Rather, they are either supposed to be ensured by the underlying execution platform, or they are programmed through ad-hoc protocols.

Our work extends SOD-M for building applications by modeling the application logic and its associated non-functional constraints and thereby ensuring the generation of reliable services' composition. As a first step in our approach, and for the sake of simplicity we started modeling non-functional constraints at the PSM level. Thus, in this paper we propose the π -SCM, the services' composition meta-model extended with *A-policies* for modeling non-functional constraints (highlighted in Figure 1 and described in Section 3). π -SOD-M defines the π -PEWS meta-model providing guidelines for expressing the services' composition and the *A-policies* (see Section 4), and also defines model to model transformation rules for generating π -PEWS models starting from π -SCM models that will support executable code generation (see Section 5).

3 π services' composition meta-model

The *A-policy* based services' composition meta-model (see in Figure 2) represents a workflow needed to implement a services' composition, identifying those entities that collaborate in the business processes (called BUSINESS COLLABORATORS⁸) and the ACTIONS that they perform. This model is represented by means of a UML activity diagram. This model includes the typical modeling elements of the activity diagram such as ACTIVITYNODES, INITIALNODES and FINALNODES, DECISIONNODES, etc., along with new elements defined by SOD-M such as BUSINESS COLLABORATORS, SERVICEACTIVITY and ACTION. A BUSINESS COLLABORATOR element represents those entities that collaborate in the business processes by performing some of the required actions. They are graphically presented as a partition in the activity diagram. ACTION, a kind of EXECUTABLENODE, are represented in the model as an activity. There are two types of actions: i) a Webservice (attribute Type is WS); and ii) a simple operation that is not supported by a Web Service, called an ACTIVITYOPERATION (attribute Type is AOP). The SERVICEACTIVITY element is a composed activity that must be carried out as part of a business service and is composed of one or more executable nodes.

Figure 2 illustrate the service composition model of the "To Publish Music" scenario. There are three external business collaborators (*Spotify*, *Twitter* and *Facebook*⁹). It also shows the business process of the "To Publish Music" application that consists of three service activities: *Listen Music*, *Public Music* and *Confirmation*. Note that the action *Publish Music* of the application calls the actions of two service collaborators namely *Facebook* and *Twitter*.

Instead of programming different protocols within the application logic, we propose to include the modeling of non-functional constraints like transactional behaviour, security and adaptability at the early stages of the services' composition engineering process. We model non-functional constraints of services' compositions using the notion of *A-policy* [10, 5], a kind of pattern for specifying *A-policy* types. In order to represent constraints associated to services compositions, we extended the SOD-M services' composition model with two concepts: RULE and A-POLICY.

The RULE element represents an event - condition - action rule where the EVENT part represents the moment in which a constraint can be evaluated according to a condition represented by the CONDITION part and the action to be executed for reinforcing it represented by the ACTION part.

An *A-policy* groups a set of rules. It describes global variables and operations that can be shared by the rules and that can be used for expressing their Event and Condition parts. An *A-Policy* is associated to the elements BUSINESSCOLLABORATOR, SERVICEACTIVITY and, ACTION of the π -SCM meta-model.

⁸ We use CAPITALS for referring to meta-models' classes.

⁹ We use *italics* to refer to concrete values of the classes of a model that are derived from the classes of a meta-model.

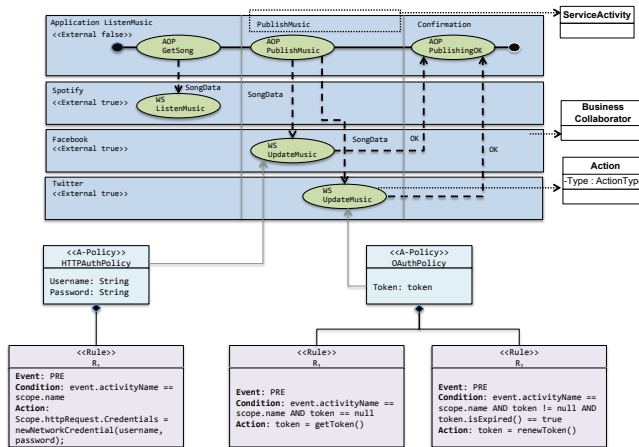


Fig. 2: Services' composition model for the business service "To publish music"

Given that *Facebook* and *Twitter* services require authentication protocols in order to execute methods that will read and update the users' space. A call to such services must be part of the authentication protocol required by these services. In the example we associate two authentication policies, one for the open authentication protocol, represented by the class *Twitter OAuthPolicy* that will be associated to the activity *UpdateTwitter* (see Figure 2). In the same way, the class *Facebook HTTPAuthPolicy*, for the http authentication protocol will be associated to the activity *UpdateFacebook*. OAuth implements the open authentication protocol. As shown in Figure 2, the *A-policy* has a variable *Token* that will be used to store the authentication token provided by the service. This variable type is imported through the library *OAuth.Token*. The *A-policy* defines two rules, both can be triggered by events of type *ActivityPrepared*: (i) if no token has been associated to the variable *token*, stated in by the condition of rule R_1 , then a token is obtained (action part of R_1); (ii) if the token has expired, stated in the condition of rule R_2 , then it is renewed (action part of R_2). Note that the code in the actions profits from the imported *OAuth.Token* for transparently obtaining or renewing a token from a third party.

HTTP-Auth implements the HTTP-Auth protocol. The *A-policy* imports an http protocol library and it has two variables *username* and *password*. The event of type *ActivityPrepared* is the triggering event of the rule R_1 . On the notification of an event of that type, a credential is obtained using the *username* and *password* values. The object storing the credential is associated to the scope, i.e., the activity that will then use it for executing the method call.

Thanks to rules and policies it is possible to model and associate non-functional properties to services' compositions and then generate the code. For example, the atomic integration of information retrieved from different social network services or for providing security in the communication channel when the payment service is called.

Back to the definition process of a SIS, once the *A-policy* based services' composition model has been defined, then it can be transformed into a model (i.e., π -PEWS model) that can support then executable code generation. The following Section describes the π -PEWS meta-model that supports this representation.

4 π -PEWS meta-model

The idea of the π -PEWS meta-model is based on the services' composition approach provided by the language PEWS[3,17] (*Path Expressions for Web Services*), a programming language that lets the service designer combine the methods or subprograms that implement each operation of a service, in order to achieve the desired application logic. Figure 3 presents the π -PEWS meta-model.

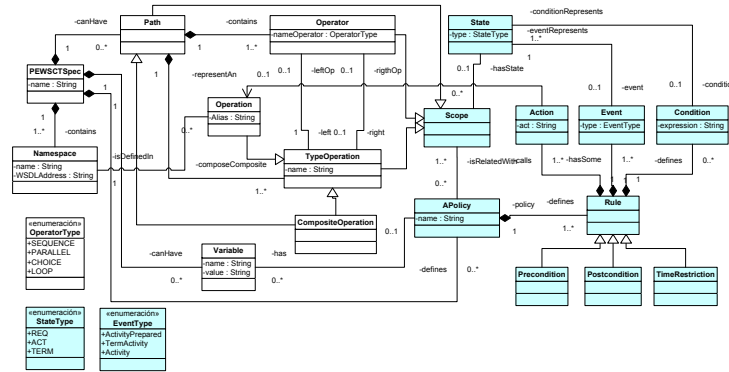


Fig. 3: π -PEWS Metamodel

NAMESPACE represents the interface exported by a service, OPERATION that represents a call to a service method, COMPOSITEOPERATION, and OPERATOR for representing a services' composition and PATH representing a services' composition. A PATH can be an OPERATION or a COMPOUND OPERATION denoted by an identifier. A COMPOUND OPERATION is defined using an OPERATOR that can represent sequential and parallel composition of services, choice among services, the sequential and parallel repetition of an operation, or the conditional execution of an operation.

A-Policies that can be associated to a services' composition: A-POLICY, RULE, EVENT, CONDITION, ACTION, STATE, and SCOPE.

As shown in the diagram an A-POLICY is applied to a SCOPE that can be either an OPERATION (e.g., an authentication protocol associated to a method exported by a service), an OPERATOR (e.g., a temporal constraint associated to a sequence of operators, the authorized delay between reading a song title in Spotify and updating the walls must be less then 30 seconds), and a PATH (e.g.,

executing the walls' update under a strict atomicity protocol - all or nothing). It groups a set of ECA rules, each rule having a classic semantics, i.e. *when an event of type E occurs if condition C is verified then execute the action A*. Thus, an *A-policy* represents a set of reactions to be possibly executed if one or several triggering events of its rules are notified.

The class SCOPE represents any element of a services' composition (i.e., operation, operator, path). The class A-POLICY represents a recovery strategy implemented by ECA rules of the form EVENT - CONDITION - ACTION. A *A-policy* has variables that represent the view of the execution state of its associated scope, that is required for executing the rules. The value of a variable is represented using the type VARIABLE. The class A-POLICY is specialized for defining specific constraints, for instance authentication *A-policies*.

Given a π -SCM model of a specific services' based application, it is possible to generate its corresponding π -PEWS model thanks to transformation rules. The following Section describes the transformation rules between the π -SCM and π -PEWS meta-models of our method.

5 Transformation rules

Figure 4 shows the transformation principle between the elements of the π -SCM meta-model used for representing the services' composition into the elements of the π -PEWS meta-model. There are two groups of rules: those that transform services' composition elements of the π -SCM to π -PEWS meta-models elements; and those that transform rules grouped by policies into *A-policy* types.

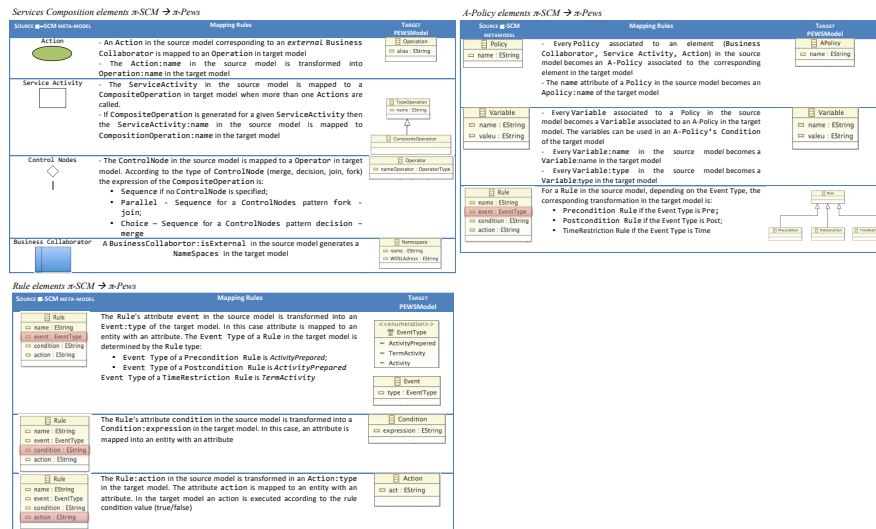


Fig. 4: π -SCM to π -PEWS transformation

6 Related works

Current standards in services' composition implement functional, non-functional constraints and communication aspects by combining different languages and protocols. The selection of the adequate protocols for adding a specific non-functional constraints to a services' composition (e.g., security, transactional behaviour and adaptability) is responsibility of a programmer. As a consequence, the development of an application based on a services' composition is a complex and a time-consuming process. This is opposed to the philosophy of services that aims at facilitating the integration of distributed applications.

Software process methodologies for building services based applications have been proposed in [14, 16, 7, 4], and they focus mainly on the modeling and construction process of services based business processes that represent the application logic of information systems. *Design by Contract* [8] is an approach for specifying web services and verifying them through runtime checkers before they are deployed. A contract adds behavioral information to a service specification, that is, it specifies the conditions in which methods exported by a service can be called. Contracts are expressed using the language *jmlrac* [12]. The *Contract Definition Language* (CDL) [16] is a XML-based description language, for defining contracts for services. There are an associated architecture framework, design standards and a methodology, for developing applications using services. A services' based application specification is generated after several [11] B-machines refinements that describe the services and their compositions.

As WS-* and similar approaches, our work enables the specification and programming of crosscutting aspects (i.e., atomicity, security, exception handling, persistence). In contrast to these approaches, our work specifies policies for a services' composition in an orthogonal way. Besides, these approaches suppose that non-functional requirements are implemented according a the knowledge that a programmer has of a specific application requirements but they are not derived in a methodological way, leading to ad-hoc solutions that can be difficult to reuse. In our approach, once defined *A-Policies* for a given application they can be reused and/or specialized for another one with the same requirements or that uses services that impose the same constraints.

7 Conclusions and future work

This paper presented π -SOD-M for specifying and designing reliable service based applications. We model and associate policies to services' based applications that represent both systems' cross-cutting aspects and use constraints stemming from the services used for implementing them. We extended the SOD-M method, particularly the π -SCM (services' composition meta-model) and π -PEWS meta-models for representing both the application logic and its associated non-functional constraints and then generating its executable code.

Non-functional constraints are related to business rules associated to the general "semantics" of the application and in the case of services' based applications,

they also concern the use constraints imposed by the services. We are currently working on the definition of a method for explicitly expressing such properties in the early stages of the specification of services based applications. Having such business rules expressed and then translated and associated to the services' composition can help to ensure that the resulting application is compliant to the user requirements and also to the characteristics of the services it uses.

Programming non-functional properties is not an easy task, so we are defining a set of predefined *A-policy* types with the associated use rules for guiding the programmer when she associates them to a concrete application. *A-policy* type that can also serve as patterns for programming or specializing the way non-functional constraints are programmed.

References

1. A. Arsanjani, S. Ghosh, A.A.T.A.S.G.K.H.: SOMA: A method for developing service-oriented solutions. *IBM System Journal* **47**(3) (2008)
2. Bell, M.: *Service-Oriented Modeling: Service Analysis, Design, and Architecture*. (2008)
3. Cheikh, B., Mirian, H.F., Martin Alejandro, M.: Composing Web Services with PEWS: A Trace-Theoretical Approach. In: *ECOWS*. (2006) 65–74
4. Ervin, R., Dimitris, D., Anthony J. H., S.: A survey of service oriented development methodologies
5. Espinosa-oviedo, J.A., Vargas-Solar, G., Zechinelli-Martini, J.L., Collet, C.: Non-Functional Properties and Services Coordination Using Contracts. In: *In proceedings of the 13th Int. Database Engineering and Applications Symposium (IDEAS 09)*, Cetraro, Italy, ACM (2009)
6. et. al, A.B.: *SOA Development Using the IBM Rational Software Development Platform: A Practical Guide*. In: *Rational Software*. (2005)
7. George, F., Sooksathit, M.: Towards software development methodology for web services. In: *SoMeT*. (2005) 263–277
8. Heckel, R., Lohmann, M.: Towards contract-based testing of web services. In Pezzé, M., ed.: *Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS 2004)*. Volume 116. (2005) 145–156
9. J., Gordijn, J.A.: Value based requirements engineering: exploring innovative e-commerce idea. *Requirements Engineering Journal* **8**(2) (2003)
10. Javier-Alfonso, E.O., Genoveva, V.S., JosÓ-Luis, Z.M., Christine, C.: Policy driven services coordination for building social networks based applications. In: *In Proc. of the 8th Int. Conference on Services Computing (SCC'11)*, Work-in-Progress Track, Washington, DC, USA, IEEE (July 2011)
11. Jean-Raymond, A., Matthew K. O., L., David, N., P. N., S., Ib Holm, S.: The b-method. In Prehn, S., Toetenel, W.J., eds.: *VDM Europe (2)*. Volume 552 of *Lecture Notes in Computer Science*, Springer (1991) 398–405
12. Leavens, G.T., Cheon, Y., Clifton, C., Ruby, C., Cok, D.R.: How the design of jml accomodates both runtime assertion checking and formal verification. In: *FMCO*. (2002) 262–284
13. M. Papazoglou, P. Traverso, S.D.F.L.: *Service-Oriented Computing: State of the Art and Research Challenges*. *IEEE Computer* **40**(11) (2007)
14. Mike P., P., Willem-Jan, v.d.H.: Service-oriented design and development methodology. *Int. J. Web Eng. Technol.* **2**(4) (2006) 412–442

15. Miller, J., Mukerji, J.: Mda guide. (2003)
16. Nikola, M.: Contract-based Web Service Composition. PhD thesis, Humboldt-Universität zu Berlin (2006)
17. Plácido A., S.N., Martín Alejandro, M., Genoveva, V.S., José-Luis, Z.M.: Adding Contracts to a Web Service Composition Language. LTPD - 4th Workshop on Languages and Tools for Multithreaded, Parallel and Distributed Programming (September 2010)
18. V. De Castro, E. Marcos, R.W.: 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. *International Journal of Cooperative Information Systems* **18**(2) (2009)
19. Watson, A.: A brief history of MDA (2008)