Non-Functional Requirements for Business Processes in the Context of Service-Oriented Architectures

Oliver Charles
agilTech Information Technologies GmbH
Am Krebsgraben 15
D-78048 Villingen-Schwenningen, Germany
oliver.charles@agiltech.de

Bernhard Hollunder
Furtwangen University of Applied Sciences
Robert-Gerwig-Platz 1
D-78120 Furtwangen, Germany
hollunder@hs-furtwangen.de

***Abstract*—We present novel concepts to formalize and apply non-functional requirements (NFRs) for business processes in the context of Service-Oriented Architectures (SOAs). Today, popular languages for modeling business processes do not support the specification of NFRs in a systematic manner. However, there is a strong demand to explicitly address such requirements when designing and deploying software systems. In this paper, we elaborate an extension for BPMN (Business Process Model and Notation) towards the modeling of NFRs. A key feature is the tool independent representation of NFRs, which will be achieved by applying the widely used WS-Policy standard. Our approach also covers the mapping of the specified NFRs to the technical level represented by BPEL (Business Process Execution Language). For the monitoring of NFRs we exploit techniques from Complex Event Processing (CEP). A key characteristic of our solution is its coherence: from NFRs modeling at design level to their technical enforcement and dynamic validation during execution. The feasibility of our approach has been demonstrated by a proof of concept implementation based on NetBeans, Glassfish ESB, IEP as CEP implementation, and the BPEL Service Engine.**

Keywords-Non-functional requirements, Business process, BPMN, BPEL, SOA, WS-Policy, Web services, Quality of service

I. INTRODUCTION

When introducing a Service-Oriented Architecture (SOA) for some enterprise, the definition of appropriate business processes as well as services plays a crucial role. A business process can be viewed as a well-defined sequence of activities to achieve a particular business goal. In order to exchange data with back-end systems (e.g., ERP systems, specific business applications and database systems), business processes typically use course-granular services, which hide the technical details of the services' implementation. Today, services are often realized with the Web services technology. In other words, a business process within a SOA composes a set of Web services in such a way that higher business goals will be obtained.

When employing Web services in the area of so-called mission critical business applications, "pure" Web services are not sufficient. This is because in such an environment non-functional requirements (NFRs) such as message reliability, confidentiality, availability and performance must be addressed. The importance of NFRs for Web services has been stressed elsewhere (see e.g., [1] or [7]). There are proven standards such as WS-SecurityPolicy [2] bringing selected NFRs to Web services.

As Web services are composed by business processes, the interaction of NFRs at service level on the one hand and at process level on the other hand must be clearly defined. Hence, it is crucial to assign – explicitly or implicitly – NFRs to business processes such as time and resource consumption, auditability and scalability (e.g., as described by Adam and Doerr in [3]).

In the past, there has been much work on modeling functional requirements of business processes. The most prominent approaches used in SOA infrastructures are the *Business Process Model and Notation* (BPMN) and the *Business Process Execution Language* (BPEL). While BPMN primarily focuses on the graphical representation of business processes, BPEL tackles technical aspects such as the mapping to Web services to be invoked during process execution. It should be noted that there is broad tool support, for an overview see e.g., [4].

Currently, there is only very limited support for specifying NFRs for business processes, though. In fact, the BPMN and BPEL do not provide language features for including NFRs features. As a consequence, when transforming a process model into an executable format the application developer must pollute the business logic with mechanisms for realizing the desired NFRs. This approach, however, would strongly limit the reusability and adaptability, if the solution should be deployed in an environment where different sets of NFRs must be supported.

In this paper, we present a novel approach for formalizing NFRs for business processes that overcomes these deficits. Special focus lies on its coherence, because we not only cover the modeling of NFRs at design level, but also their technical enforcement and their dynamic validation during execution. Our approach comprises the following aspects:

- Modeling of NFRs with BPMN and BPEL by exploiting standard extension mechanisms.
- Enforcement strategy for NFRs based on Web service handlers.
- Usage of standards such as WS-Policy to formalize NFRs at the technical level.
- Static and dynamic validation of NFRs.
- Tool support and proof of concept.

The paper is structured as follows. The next section will give a short introduction to the underlying technologies required to understand our approach. Related work will be

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-165-6

discussed in Section three, followed by a detailed description of our solution. Section five will cover the proof of concept implementation. Conclusions and open issues are part of the final section.

II. FOUNDATIONS

This section briefly introduces the most important concepts and techniques as required for the understanding of our approach. We start with *Business Process Management*, which is the general area our results apply to. Then we provide background information to *Non-Functional Requirements*, followed by a short review of *WS-Policy*, which is a well-known and widely used standard for formalizing NFRs for Web Services and SOAs.

Due to limited space, this paper does not give an introduction to BPMN and BPEL. Hence, we assume an understanding of the basic concepts of these technologies.

A. Business Process Management

Business Process Management (BPM) includes concepts, methods, and techniques to support the design, implementation, enactment, monitoring, and strategy alignment of business processes. In the context of SOAs, BPM focuses on how business processes can be automated using SOA infrastructure elements. The target is not only a high automation of processes, but also to enable development and management to react in a flexible and agile manner on changing business or technical requirements.

BPM covers the following topics:

- Strategy phase
- Design phase
- Execution phase
- Monitoring phase.

As the name of the first phase indicates, the main focus is the elaboration of the mid- to long-term alignment of an enterprise and how IT can be leveraged to automate and optimize business processes. Having defined the strategic goals, in the design phase the identified business processes are brought to "IT-level". This includes a proper description from which an implementation will be derived. The usage of graphical modeling languages – in particular BPMN and BPEL – is not only advantageous for the domain experts, but also helps bridging the gap to the implementation level. While the execution phase is concerned with the usage of the implemented business processes by clients, the goal of the monitoring phase is to receive data regarding the runtime behavior such as identification of bottlenecks, quantity of invoked processes, and performance analysis.

As already mentioned in the introduction, our solution considers NFRs at the design, implementation, and monitoring level. That is the reason why we term it *coherent*.

B. Non-Functional Requirements

In system and software engineering there are mainly two categories of requirements: *functional* and *non-functional requirements*. A functional requirement describes a specific business or technical functionality of a system in terms of the input/output behavior. In contrast, a non-functional requirement addresses a quality of service (QoS) attribute of the implementation. In software engineering, there was (and still is) much research on NFRs for software systems. Standardization organizations such as ISO have identified manifold aspects (see e.g., ISO/IEC 9126 [6], which is superseded by ISO/IEC 25000 [5]).

There are several publications that consider NFRs in the specific context of SOA, e.g., by O'Brien, Merson, and Bass [7]. OASIS [1] gives a classification of different types of NFRs (which are called quality factors). Besides others, the following topics are covered:

- duration and response time
- throughput
- availability and reliability
- standard conformance
- observability
- security aspects such as confidentiality, authentication, authorization, integrity, and non-repudiation
- pricing and accounting
- robustness.

Let us make some remarks. Even though we can find in the literature characterizations of NFRs, there are often differences regarding their exact meaning and definition. Some of them can be described by a formula; e.g., response time, duration, and availability. The behavior of other NFRs such as integrity can be defined in terms of functions for digital signature. Robustness is an example for an NFR that has diverse facets such as error tolerance, often described as the ability to deal with erroneous input. A business process, for example, should not crash or run into an inconsistent state if it is called with invalid parameter values.

C. WS-Policy

WS-Policy [8] is a specification of the W3C and provides a policy language to formally describe "properties of a behavior" of services. A WS-Policy description is a collection of so-called assertions. A single assertion may represent a capability, a requirement or a constraint and has an XML representation. An example for an assertion is

```
<Performance max_runtime_minutes="15"/>,
```

which formalizes a condition for the runtime behavior of a particular business process.

WS-Policy introduces operators to form policies, which are basically sets of assertions. Policies can be attached via the WS-PolicyAttachment [9] specification to other entities such as a BPEL process description and a Web service's WSDL. We will come back to this issue when introducing our solution.

III. RELATED WORK

In [10], Pavlovski and Zou present an approach to model NFRs for business processes in a graphical manner. They introduce extensions for BPMN, the enforcement on the technical level (e.g., in BPEL) has not been elaborated, though. For the modeling of NFRs, Zou and Pavlovski propose two extensions of BPMN: i) an "operating condition" artifact and ii) a "control case" artifact. With an

operating condition artifact, a business process modeler should be able to connect NFRs such as security, performance or availability to activities of the BPMN process model. The use of the control case artifact is optional and is introduced to refine an operating condition artifact. From a more technical point of view, a control case artifact is a reference to a table containing detailed information about the modeled NFRs.

The approach of Rodriguez et al. [11] also tackles the modeling of NFRs within BPMN. However, their solution is restricted to the modeling of security requirements. They do not extend the standardized artifacts of BPMN, but rather implement new Business Process Diagram (BPD) core elements. In this context it is described how to extend the BPD meta-model towards the coverage of security issues. The mapping of "security-enhanced" process models to the technical level (as in the approach in [10]) is not addressed.

Tai et al. [12] explain a new idea about how transactional behavior can be modeled as NFRs within BPEL. To express this with XML, the authors use WS-Policy [8] in combination with WS-PolicyAttachment [9]. They directly attach WS-Policy descriptions to selected BPEL elements within the process document. Proposed elements are for example `<partnerLink>` or `<scope>`. To enforce the attached WS-Policy descriptions, Tai et al. assume a coordination middleware, which executes the BPEL-process taking into account the NFRs.

Charfi et al. present in [13] another approach to model non-functional requirements with BPEL. Their approach is based on well-known standards and specifications such as WS-Policy, WS-PolicyAttachment and XPath. It has to be mentioned that their approach is not a completely new one but a combination of the mentioned standards.

To sum up, there are several approaches that extend process models towards NFRs. However, they either focus on BMPN or BPEL. As we will see in the next section, our solution – beside other features – includes the mapping from BPMN to BPEL.

IV. THE OVERALL ARCHITECTURE

A. Modeling NFRs in BPMN

BPMN does not provide explicit language constructs for modeling NFRs. Basically, there are two options to overcome this limitation: i) introducing new language features optimized for modeling NFRs, and ii) applying existing artifacts in a specific way. A disadvantage of the first alternative would be missing support by existing BPMN tools. Therefore, we pursue the second approach.

A so-called text annotation is a standard artifact of BPMN, which allows one to attach auxiliary information to model elements. The following figure gives an example:

Figure 1. QoS artifact for BPMN.

At the left hand side there is some business process activity. In order to impose NFRs for this activity, we assign a text artifact. In this approach, we distinguish between arbitrary text annotations and those, which formalize NFRs. The latter are called "QoS artifacts" and are text artifacts with a particular content and specific syntax.

In our approach, we support the following syntax: The prefix "QoS" indicating a QoS artifact is followed by a category name, which specifies a particular NFR. In the previous example, we impose a performance restriction to the modeled activity. Finally, a set of attribute/value-pairs define the specific properties for the NFR.

The content of a QoS artifact is a text with some well-defined structure, which will be mapped to XML. In order to support syntax checking, we have defined XML schemas for the supported NFRs. Due to lack of space, we omit the description of the schemas.

We have defined a library comprising well-known QoS artifacts. Each QoS artifact comes with a modeling manual describing its meaning, formalizing the required syntax by means of an XML schema, and optional modeling examples. It should be noted that the set of predefined QoS artifacts could be extended by additional NFRs basically by defining its XML schema.

To sum up, our NFRs modeling approach is a light-weight solution, which reuses standard artifacts supported by BPMN tools. As a consequence, process models can be exchanged between different tools without losing NFRs model information. The usage of XML schemas not only specifies the specific syntax but also allows the automated validation. Last but not least, the QoS artifacts library can be reused in different settings.

B. Modeling NFRs in BPEL

As mentioned above, BPEL does not support the modeling of NFRs in a direct manner. In [13] it has been shown how to overcome this limitation by applying the standards WS-Policy, WS-PolicyAttachment and XPath. The main idea is to link BPEL process elements to a WS-Policy description. Such a description contains WS-Policy assertions formalizing NFRs (see Section II-C). A well-known set of assertions for the security domain has been introduced in [2].

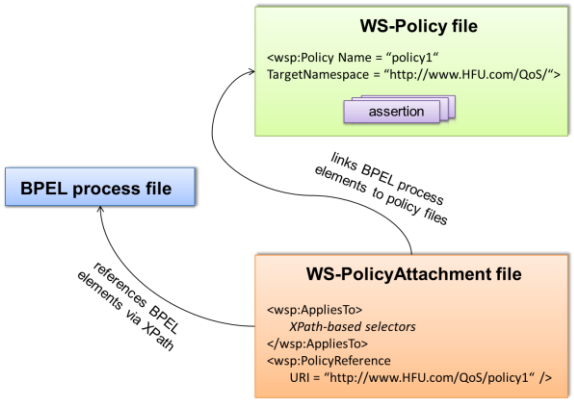

Figure 2. Assigning WS-Policy to BPEL.

Figure 2 depicts the linkage between the BPEL process and the policy description. A WS-PolicyAttachment file contains an <AppliesTo> entry referring to the BPEL element to which the WS-Policy description should be applied. The latter is linked via the <PolicyReference> element, which is also introduced by the WS-PolicyAttachment specification. As we apply XPath for selecting the targets, this approach exclusively uses well-known and widely supported specifications.

This concept clearly separates i) the logic of the business process and ii) the required NFRs. As a consequence, both parts of the overall application can evolve independently from each other, which has a positive effect on maintainability, reusability and adaptability of the solution. As an example, consider a WS-Policy file that formalizes a particular set of NFRs. The policy can be applied to several business applications. As a consequence, this not only increases reusability of the required "NFRs patterns" but also guarantees conformance to corporate compliance rules.

C. Transformation of NFRs – From BPMN to BPEL

Having described how to represent NFRs within BPEL, we are now able to consider the mapping from QoS artifacts in a BPMN model to WS-Policy descriptions for BPEL. It should be noted that we do not consider the general transformation rules mapping BPMN elements to BPEL elements, because they are part of most BPMN/BPEL modeling tools.

To map NFRs we proceed as follows: For each QoS artifact, we create both a WS-PolicyAttachment file as well as a WS-Policy file. The assertions contained in the policy description correspond to the NFRs of the QoS artifacts. These assertions in turn have references to the XML schema definition and the modeling manual, respectively. After all WS-Policy documents have been created, they are used by the corresponding WS-PolicyAttachment files to link the required policies to BPEL process elements as already described.

D. Enforcement of NFRs

This section is concerned with the question how the modeled NFRs can be enforced. Basically, we observe that there are two targets to which the modeled NFRs will be applied: i) the business process itself, and ii) the communication between a BPEL service and an underlying Web service. From a modeling perspective, we use the following convention: if the category name of a QoS artifact starts with "WSComm_", the latter target is meant, otherwise the NFR applies to the business process.

If a policy relates to the business process itself, which means that the described prefix is not set by the BPMN modeler, the Web service developer has to extend the Web service's application logic, i. e., the source code.

If a policy relates to the service communication, the Web service developer has the responsibility to enforce the NFRs with the help of interceptors (also called handlers), which can be installed in SOA infrastructures and manipulate the outgoing and incoming messages. Details can be found e.g., in [14].

In order to enforce a specified behavior, typically an appropriate WS-Policy description will be attached to the Web service's WSDL as well as to BPEL process elements (see Figure 3). This policy may for example specify that the invoker (e.g., the BPEL service) must encrypt the parameter values passed to the Web service, which in turn is able to decrypt these values. For standard NFRs (such as security and reliable messaging) Web services frameworks typically provide respective handlers. For other NFRs such as accounting and resource consumption specific handlers must be configured.

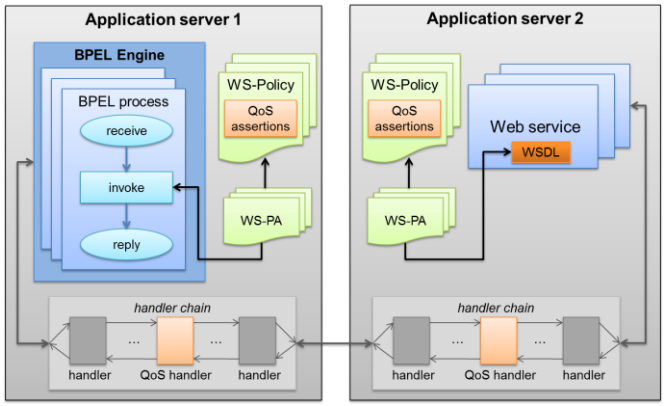

Figure 3. QoS enforcement through handlers.

To support several NFRs, all the required handlers must be installed. This can be achieved by using so-called handler chains supported by Web services frameworks. Before a request is delivered to the service implementation, each handler will be invoked.

E. Validation of NFRs

Our architecture also includes components for validating NFRs. We distinguish between static and dynamic validation. During the static validation process, the NFRs contained in a BPMN process diagram will be checked against the WS-Policy descriptions of the underlying Web services implementations. Static validation can be automated by applying WS-Policy compatibility algorithms such as WS-Policy intersection [8] and semantic policy differencing [15].

Performance is an example of an NFR where dynamic validation must be applied. As the actual execution time of a process depends on factors, which are not determinable *a priori* (e.g., server consumption, network latency and user interaction), a monitoring system is required in order to continuously observe the infrastructure. To provide a monitoring system with the required data, so-called sensor components (such as JMX and NFRs handlers, see [16]) can be installed in SOA infrastructures.

This system will inform, for instance, the system administrator if some NFRs are violated. Depending on the severity of the violation (e.g., leakage of sensible data) actions may be immediately performed such as shutting down a service or a server.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-165-6

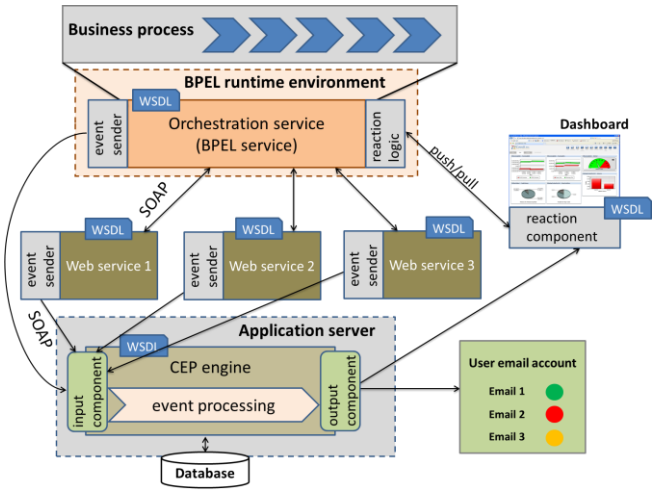

Figure 4. Dynamic validation with CEP [17].

Complex Event Processing (CEP) [18] has been introduced as a technology to find correlated data items in a continuous flow of data. The data items to be selected are specified by patterns defined, for instance, with the Continuous Query Language (CQL). It turned out that the conditions, which indicate a violation of an NFR during execution, can be appropriately defined as CEP patterns. Figure 4 illustrates the integration of an abstract CEP engine in a BPEL/Web services environment. We have identified the following components:

- input component
- output component
- CEP engine
- reaction component
- event senders.

The input component receives events from the BPEL engine and the Web services, respectively. The so-called event senders, which are specific implementation of the above mentioned sensor components, inform the CEP engine about significant actions in the business application (e.g., transition within the business process, Web service invocation, passing of non-encrypted sensible data, etc.). Subsequently, the input component passes the received events to the CEP engine. As soon as the CEP engine detects data items that match a CEP pattern, a new (complex) event will be created. The output component has the responsibility to pass it to a user (e.g., via SMTP) or to the reaction component, which in turn will inform the orchestration service, or to a management system (via Web service invocation) about the violation of an NFR.

V. PROOF OF CONCEPT

The overall architecture presented in the previous section is quite generic and can be instantiated in different ways. In order to show the feasibility of our approach we have developed a proof of concept implementation based on NetBeans IDE and Glassfish ESB. This combination comprises the following tool set:

- *BPMN/BPEL designer* to model business processes.
- *BPEL runtime environment* for executing BPEL processes.
- *Web services* development, deployment and runtime environment.
- *Intelligent Event Processing* (IEP) service engine as implementation of a CEP engine.

The BPMN/BPEL designer allows the graphical modeling of business processes according to the BPMN and BPEL languages. It should be noted that only those BPMN elements are supported by the tool, which can be mapped to the XML BPEL process file.

One of these elements is the documentation artifact, analogous to the common BPMN text artifact that allows the attachment of comments to elements in a BPEL process. These comments are transformed to the common BPEL tag `<documentation>` within the underlying XML BPEL process file. To avoid this intrusion, we extended the BPMN/BPEL designer by a new QoS artifact (see Figure 5) with which it is possible to implement our introduced transformation process as described above.

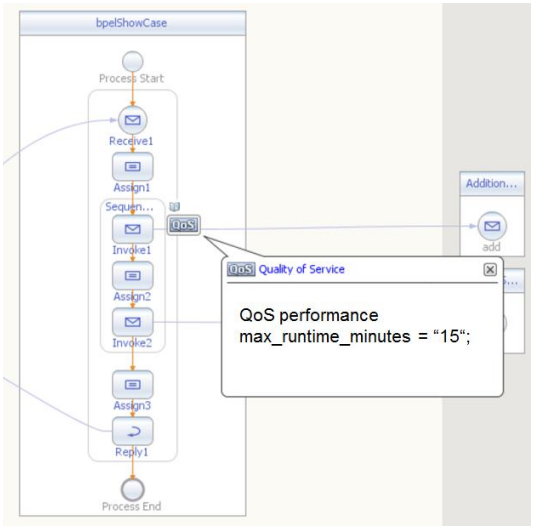

Figure 5. QoS artifact in the BPMN/BPEL designer.

With the NetBeans composite application display it is possible for a Web service developer to attach handlers via the context menu not only to the BPEL service but also to the Web services, which are invoked. This enables an easy configuration of handlers required for enforcing the defined NFRs.

The IEP service engine comes with a graphical modeling language for selecting, transforming and aggregating events. This modeling language also provides predefined types for output components, e.g., datasets, database tables and dashboard formats. It is also possible to generate WSDL interfaces for the input components and their implementations as Web services, which can be used by the event senders.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-165-6

Hence, IEP enables the validation of a business process during its execution.

Independent of IEP, it is also possible to make theoretical commitments before process execution. For example, a Web service developer wants to check if the modeled runtime of a business activity complies with the modeled runtime of the Web services. Therefore, the WS-Policy assertion of the business activity has to be checked against the sum of runtime assertions of the Web services to be invoked. Unfortunately, this functionality is not provided yet by NetBeans and Glassfish ESB, respectively, so that this check has to be accomplished manually.

VI. CONCLUSIONS AND FUTURE WORK

In software engineering, there has already been much work on non-functional requirements. This is motivated by the fact that nearly all deployed application systems must not only fulfill the desired business logic, but should also guarantee aspects such as robustness, scalability, security, performance and reliability. Although NFRs should be especially considered when designing applications according to the SOA principle, there is currently only partial support – both from a conceptual as well as technical point of view.

In our work, we have presented a coherent concept for formalizing, applying, enforcing, and monitoring NFRs for business processes. A driving force of our solution is the commitment to well-known standards and widely used technologies such as BPMN, BPEL, WS-Policy, CEP, and others. As a consequence, the conceptual framework of our solution can be instantiated in several ways based on existing tools such as NetBeans and Glassfish ESB.

This demonstrates the high impact of our results on software engineering practice. Specifically, our approach is a further step towards improving the development of business application with well-defined NFRs. We support the well-known separation of concerns principle by flexibly attaching NFRs to business processes.

Our work can be extended in several ways. In order to leverage our solution, further NFRs should be formalized. This includes the definition of the required QoS artifacts for BPMN and their mapping to corresponding WS-Policy assertions. To disseminate our approach in software engineering practice, additional proof of concept implementations would be quite helpful; especially an instantiation with the Visual-Studio IDE and the .NET technology.

ACKNOWLEDGMENT

We would like to thank Markus Schalk for the extensive support during the elaboration of the architecture and the proof of concept. This work has been partly supported by the German Ministry of Education and Research (BMBF) under research contract 17N0709.

REFERENCES

[1] OASIS, Web Services Quality Factors 1.0 (07/2010), Retrieved April 16, 2011, from http://docs.oasis-open.org/wsqm/wsqf

[2] OASIS, WS-SecurityPolicy 1.3 (03/2009), Retrieved October 11, 2010, from http://docs.oasis-open.org/ws-sx/ws-securitypolicy

[3] S. Adam and J. Doerr, "Towards Early Consideration of Non-Functional Requirements at the Business Process Level", Proceedings of International Conference on Information Resources Management, pp. 227-230, 2007.

[4] OMG, Object Management Group / Business Process Management Initiative, Retrieved March 03, 2011, from http://www.bpmn.org

[5] ISO/IEC, "Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE", ISO/IEC 25000, 2005.

[6] ISO/IEC, "Software engineering – Product quality", ISO/IEC 9126-1 to 9126-4, 2001-2004.

[7] L. O'Brien, P. Merson and L. Bass, "Quality Attributes for Service-Oriented Architectures", in Proceedings of International Workshop on Systems Development in SOA Environments (SDSOA '07), pp. 1-5, 2007.

[8] W3C, Web Services Policy 1.5 (09/2007) – Framework, Retrieved April 03, 2011, from http://www.w3.org/TR/ws-policy

[9] W3C, Web Services Policy 1.5 (09/2007) – Attachment, Retrieved April 03, 2011, from http://www.w3.org/TR/ws-policy-attach

[10] C. Pavlovski and J. Zou, "Non-Functional Requirements in Business Process Modeling", in Proceedings of the Asia-Pacific Conference on Conceptual Modelling, pp. 103-112, 2008.

[11] A. Rodriguez, E. Fernández-Medina and M. Piattini, "A BPMN Extension for the Modeling of Security Requirements in Business Processes, IECE Trans. Inf. & Syst, pp. 745-752, 2007.

[12] S. Tai, R. Khalaf, and T. Mikalsen, "Composition of Coordinated Web Services", Middleware, pp. 294-310, 2004.

[13] A. Charfi, R. Khalaf and N. Mukhi, "QoS-Aware Web Service Compositions Using Non-intrusive Policy Attachment to BPEL", in Proceedings of the 5th International Conference on Service-Oriented Computing, pp. 582-593, 2007.

[14] E. Hewitt, Java SOA Cookbook: SOA Implementation Recipes, Tips, and Techniques, O´Reilly, 2009.

[15] B. Hollunder, "Domain-Specific Processing of Policies or: WS-Policy Intersection Revisited," Proceedings of the 7th IEEE International Conference on Web Services (ICWS), pp. 246-253, 2009.

[16] A. Wahl, A. Al-Moayed, and B. Hollunder, "An Architecture to Measure QoS Compliance in SOA Infrastructures", Proceedings of the Second International Conferences on Advanced Service Computing (Service Computation 2010), pp. 27-33, 2010.

[17] O. Charles, M. Schalk, and B. Hollunder, "CEP meets SOA", OBJEKTSpektrum: Vol 5, pp. 28-32, 2010.

[18] D. Luckham, "The Power of Events", Addison Wesley, 2007.

Copyright (c) IARIA, 2011. ISBN: 978-1-61208-165-6