Realizing Quality Attributes of Service-based Business Processes: A Model-driven Approach

Fabrício Teles, Nelson Rosa
Centro de Informática
Universidade Federal de Pernambuco
Recife, Brazil
{fst, nsr}@cin.ufpe.br

Fernando Lins
Departamento de Estatística e Informática
Universidade Federal Rural de Pernambuco
Recife, Brazil
fernando.aires@deinfo.ufrpe.br

***Abstract*— The agnostic, integrated and automated treatment of non-functional (quality) attributes into a business process realization is still a challenge. While functional attributes are already addressed by existing approaches, non-functional ones are commonly neglected. A systematic way of dealing with such attributes from the most abstract business level to their technical realization through service compositions is a complex task. In this context, this paper presents a model-driven approach, in which quality attributes are transformed until their technical enforcement at execution level. Additionally, two key characteristics of our approach are the agnostic treatment of the quality attribute being considered and its projection on standards such as URN (User Requirement Notation), UML for QoS and WS-Policy. To illustrate the proposed approach, an illustrative scenario of Supply Chain Management (SCM) is used.**

Keywords— Quality Attributes, Model Driven Engineering, Service Oriented Computing, Business Process Management

I. INTRODUCTION

Business Process Management (BPM) lifecycle [24] consists of four main stages, namely process design, system configuration, process enactment and diagnosis. To address quality attributes since the early stages (process design) until the final realization (process enactment) is a complex task. A possible strategy to address quality attributes in the whole BPM lifecycle is to define them separately from the functional definition, i.e., by defining catalogues of quality characteristics using well-known standards for quality definition (e.g., GRL/URN [8], i* [32], NFR framework [2]), and linking them to the business processes in a non-invasive way, i.e., without defining new language features and simply reusing existing ones, such as the so-called 'text annotation' standard artifact of BPMN.

Quality characteristics of business processes (e.g., time and resource consumption, auditability, scalability, security) are arguably difficult to capture in business process modeling, since the focus of this early stage is the discovering of functional characteristics [22]. Moreover, existing standards for business process modeling (e.g., BPMN) do not readily support the expression of quality characteristics of business. To extend their notations, however, may not bring good results, since desirable requirements such as readability, extensibility, standardization and tool support may be lost after that.

Looking at the stages of the BPM lifecycle (e.g., system configuration, process enactment), it is possible to observe that there is a relevant support for the treatment of quality attributes. Specifically in the context of SOC infrastructures, there are several works that consider quality characteristics of services [6][26]. Additionally, OASIS [14] gives a classification of different types of quality factors of services. For the enforcement of quality attributes at execution time there is the support of well-known specifications, such as WS-Policy [26] and WS-SecurityPolicy [15]. Additionally, to support the diagnosis stage, dynamic validation of quality attributes with Complex Event Processing (CEP) [12] solutions have also been proposed [1].

Despite the (independent) support for defining quality attributes at various stages of BPM lifecycle, it is worth observing the lack of systematic approaches for a (integrated) treatment of quality attributes at these various stages. For instance, nowadays it is difficult to translate an abstract quality attribute of the business process definition (e.g., quality attributes in GRL) into its respective operational definition (e.g., QoS policies in WS-Policy), in such way that we can ensure the abstract quality definition is being fully considered at execution time.

This paper presents the "MOdel-driven QUality Attributes for business processes" (MOQUA), which aims to provide a systematic way of treating with quality attributes through the whole BPM lifecycle. MOQUA proposes a forward model-driven engineering process in which abstract business processes models and their quality attributes are sequentially refined through model transformations until production of executable business processes.

As unique contributions, this paper proposes (1) an integrated service-oriented and model-driven forward engineering process for the treatment of quality attributes of business processes, (2) a model transformation chain (CIM to PIM and PIM to PSM) for the realization of these quality attributes, (3) an integrated use of standards such as User Requirement Notation (URN) [8], UML for modeling QoS and Fault Tolerance [18], and WS-Policy [26].

This paper is organized as follows. Next section presents the envisioned MOQUA solution, detailing the model-driven approach used to realize quality attributes. In order to illustrate

this approach, Section III presents a use case in the context of the Supply Chain Management (SCM) application [30]. Section IV presents related works and shows the main differences between the existing approaches and the one presented in this paper. Finally, in Section V some conclusions and future work are presented.

II. MOQUA

MOQUA (MOdel-driven QUality Attributes for business processes) defines a forward model-driven engineering process in which abstract business process models along with their quality attributes are sequentially refined through model transformation until the production of executable business processes. By using a Model-Driven Engineering (MDE) approach, we aim to automate the translation from abstract to concrete business processes.

Fig. 1 shows the relationship between MOQUA and BPM, using the BPM lifecycle [24]. The BPM lifecycle describes the various phases to operationalize business processes. In the *design phase*, processes are (re)designed. In the *configuration phase*, designs are implemented by configuring a process-aware software system. After configuration, the *enactment phase* begins when the operational business processes are executed using the system configured. In the *diagnosis phase*, the operational processes are analyzed to identify problems and opportunities for improvement [24].

Based on the BPM lifecycle, this paper proposes a model-driven approach to support the *process design*, *system configuration* and *process enactment* stages. Fig. 1 illustrates an overview of the proposed approach, highlighting the model-driven viewpoints (CIM/PIM/PSM), and their relation with the BPM lifecycle phases (*process design* with CIM, *system configuration* with PIM/PSM and *process enactment* with Code). Additionally, it is presented the forward engineering process, in which abstract business models (CIM = [*Business* + *Quality*] *Conceptual Model*) are refined into platform independent models (PIM = [*Business* + *Quality*] *SW Design Model*) that are finally refined into executable business models (PSM = [*Business* + *Quality*] *SW Operational Model*). Central in this approach are the model to model (M2M) and model to text (M2T) transformations responsible for the vertical model refinement, and the weaving models responsible for the horizontal and fine-grained correspondence between model elements from the models of the same viewpoint.

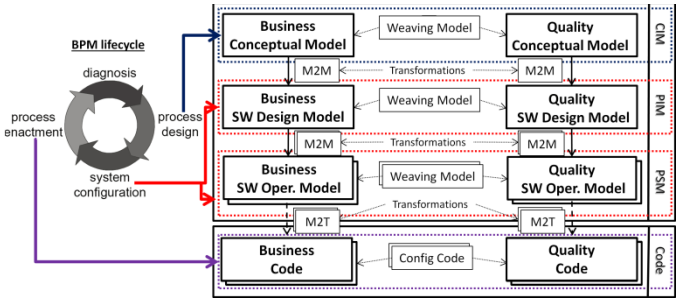

Fig. 1. BPM lifecycle and MOQUA conceptual solution

A. *Methodological and technical decisions*

This section puts forward the methodological and technological decisions that drive MOQUA technical design and elaborates on the rationale behind each decision.

Adherence to MDA principles: Our solution adheres to the four principles that underlie the OMG's view of MDA [13]: (1) Models should be expressed in a well-formed notation that forms the cornerstone for understanding the software system. On this point, the semantic of our models is based on metamodels. (2) The building of the software system can be organized around a set of models by applying sequential model refinement, organized into an architectural style of layers and transformations. We provide M2M transformations to support any change between models while M2T transformation are associated with (automated) code production. (3) A formal underpinning for describing models in a set of metamodels facilitates meaningful integration and transformation among models, and is the basis for automation through software. Our metamodels are compliant with the OMG's MetaObject Facility (MOF). (4) The acceptance and adoption of a model-driven approach require industry standards to provide openness to consumers, and foster competition among vendors. Our solution is based on the use of well-founded business process and quality standards.

Separation of concerns: We propose to treat the quality attributes of the business process separately from its functional definition. For this, catalogues of quality attributes must be defined, and linked to the business process in a non-invasive way, i.e., by using (weaving) annotations in the business process to define the semantic relation of the business process elements and the quality attributes definition.

Agnostic approach: We propose to capture the non-functional (quality) intentions at business level as they apply to the business but avoid addressing specific software attributes. The quality intentions are identified in high-level business modeling, as first-class entities in the software development. The idea is to deliver a software solution centered from the beginning on the business process and also the quality attributes that are really significant to the business needs. Although the agnostic treatment, it is worth observing that a considerable number of quality attributes may appear in the context of business processes that can be grouped in broad dimensions, such as suitability, accuracy, security, reliability, understandability, learnability, time efficiency and resource utilization. Some of them can be accomplished by means of software solutions, but others do not. We are interested in the ones realizable by software.

Extensive use of standards: Besides the MDA's orientation to use standards, we highlight additional benefits related to the using of standards: (1) to promote consistency and communication through common languages and notations (e.g., BPMN, GRL/URN), (2) to support reusability and extensibility by designing the software solution using Domain Specific Languages (DSLs) based on UML profiles (e.g., SoaML, QoS&FT), and (3) fostering acceptance and adoption by using technologies with broad tool support such as WS-BPEL and WS-Policy.

In the following, we focus on the standards that were chosen to realize each model-driven viewpoint (CIM/PIM/PSM) of MOQUA conceptual solution (Fig. 1). At the CIM viewpoint, the *Business Conceptual Mode*l is expressed in BPMN [16], and the *Quality Conceptual Model* is expressed in the GRL notation of URN [8]. These models serve as the input for the forward MDE process.

Besides the many benefits of using a standard modeling notation, the use of BPMN is justified as follows: (1) BPMN is readily understood by a considerable number of business users - from business analysts to technical developers; (2) BPMN has a standardized metamodel and serialization format, which allows users to exchange business process models in several ways; and (3) BPMN provides mapping mechanisms to view business processes models as executable processes described in business process execution languages, such as WS-BPEL.

Additionally, at the CIM viewpoint, we use GRL to represent quality attributes of business processes. GRL supports a goal-oriented modeling and reasoning about requirements, especially on quality attributes [8]. Since BPMN does not provide direct modeling features to analyze process alternatives in terms of high-level quality attributes or business (soft) goals, a goal-oriented modeling language serves as a good alternative to meet those needs [31]. In the context of business process modeling, a GRL goal model captures business goals, alternative means of achieving goals, and the rationale for goals and alternatives, along with an explanation of why an alternative is preferable or not.

GRL is based on two mainstream goal-oriented modeling languages: i*[32] and the NFR Framework [2], having relevant benefits over them: (1) GRL supports representation of numeric-based and qualitative-based quality attributes, (2) it provides additional first-class elements such as strategies (for analysing GRL models), metadata (for extensibility purposes), and (3) it provides a clear separation of model elements from their graphical representation, so that it enables a scalable and consistent representation of multiple views of the same goal model.

At the PIM viewpoint, the *Software Design Model* is expressed using the metamodel of the Service oriented architecture Modeling Language (SoaML) [17], and the Quality Design Model is expressed using the metamodel of the UML for modeling Quality of Service and Fault Tolerance (QoS&FT) [18].

SoaML provides a metamodel and a UML profile for specifying and designing services within a service-oriented architecture. SoaML is founded on the separation of logical implementation of a service from its possible physical realizations on various platforms. Additionally, SoaML is intended to work in synergy with business level languages like BPMN. Hence, these characteristics are aligned to our approach. Also, model-driven mappings from BPMN to SoaML are nowadays an active research topic [3][11][21] and we take advantage of these mappings.

Additionally, at the PIM viewpoint, we use QoS&FT to represent quality attributes at the software design level. QoS&FT provides a metamodel and a UML profile for the description of quality of service (QoS) and fault-tolerance (FT) properties in the context of a system, service or computational resource. As major benefits of adopting QoS&FT, we have: (1) the ability to define a broad range of QoS attributes, (2) a QoS framework having a metamodeling foundation that provides a hierarchy of QoS meta-classes with well-formed verification rules and semantics, and (3) a general QoS catalogue that includes a set of general QoS characteristics and categories that can be reused to define specialized QoS attributes.

Finally, at the PSM viewpoint, the *Business Operational Model* is expressed in some executable/interpretable language (e.g., WS-BPEL, jPDL) to represent the executable business process. Similarly, the *Quality Operational Model* is expressed in some policy definition language (e.g., WS-Policy, XACML) to enforce quality characteristics (e.g., security, reliable messaging, or access control) at execution time. In this paper we propose to realize the PSM using WS-BPEL and WS-Policy, since they provide integration in synergy to operationalize the business processes and their quality attributes. But it is worth observing that the MOQUA conceptual solution (Fig. 1) is not limited to those languages/technologies, since the PSM viewpoint could be realized using distinct business execution and quality enforcement languages.

B. Architecture

This section presents an overview of the model-driven architecture to support the definition and handling of the models and transformations proposed in the conceptual solution (Fig. 1). Fig. 2 depicts the proposed architecture through three layers. First, the *Modeling Layer (ML)* represents how business processes (functional) and quality attributes (non-functional) can be modeled. Second, the *Model Transformation Layer (MTL)* details the model transformation chain to transform the business process and the quality attributes into executable artifacts. Finally, the *Operational Layer (OL)* represents how the business process is executed and how the quality attributes are enforced at execution time.

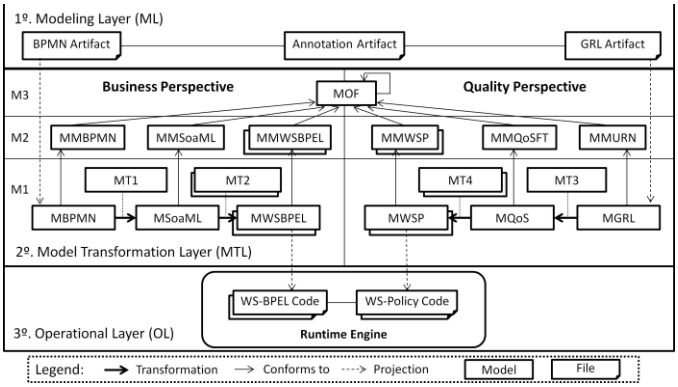

Fig. 2. MOQUA Architecture: modeling, model-transformation and execution

At the *Modeling Layer (ML)* the business process definition (*BPMN artifact*) and the quality attributes definition (*GRL artifact*) are linked by means of an annotation weaving artifact (*Annotation artifact*). This weaving artifact represents how the non-functional (quality) assessments relate to the process

elements, i.e., which specific activities, data objects, resources or any other element involved in the process are traced to the softgoals and descriptions of the quality attributes definition. To define the *BPMN artifact* we use the Eclipse BPMN2 Modeler tool and to define the *GRL artifact* the choice is the Eclipse jUCMNav tool. Finally, the *Annotation artifact* is based on the weaving modeling approach of the AMW (ATLAS Model Weaver) tool.

The *Model Transformation Layer (MTL)* in centered in the transformation chains to refine the models projected from the ML. This layer is structured through two perspectives (*Business* and *Quality*) in accordance with the conceptual solution previously defined (Fig. 1). MTL is also subdivided in three horizontal sub-layers (M1, M2 and M3). M1 defines the set of models of each perspective (e.g., *MBPMN* stands for the BPMN model, *MGRL* stands for the GRL model). Similarly, M2 presents the definition of the metamodels (e.g., *MMBPMN* stands for the BPMN metamodel, *MMURN* stands for the URN model), and finally M3 represents the central meta-metamodel that is based on MOF.

The *Operational Layer (OL)* comprises a *Runtime Engine* to support the execution of the business process (*WS-BEPL Code*) and the enforcement of the quality attributes (*WS-Policy Code*). The combined use of the Apache ODE engine (for business process execution) with the Apache Rampart module (for the quality attributes enforcement, specifically security) is an example to the *Runtime Engine*. The eclipse Business Process Management (eBPM) project is also a valid example and is our choice, since it aims to provide not only runtime support but a complete BPM solution.

C. Models and Transformations

Central in each perspective, the *Model Transformation Layer (MTL)* is the focus of this section, since the *Modeling Layer (ML)* and *Operational Layer (OL)* adopt existing tools, as already mentioned. MTL is defined atop of the Eclipse Modeling Framework (EMF), a metamodeling framework that provides integrated facilities to define, edit and handle models and metamodels. EMF has a great potential in the metamodeling field and has been broadly used to define new metamodeling tools during the last years.

The metamodels are defined in terms of Ecore meta-metamodel, the metamodeling language of EMF (Ecore provides an implementation for the essential part of the OMG's MOF standard). From an Ecore metamodel, it is possible to generate a set of plug-ins that bundle Java programming code to provide runtime support for graphically editing, manipulating, reading, and serializing models conforming to such metamodel. In order to automate the transformations, we use ATLAS Transformation Language (ATL) [9]. ATL has been widely accepted as de facto standard for model to model transformation development [25].

Fig. 2 (left side) depicts that in the *business perspective*, the *MBPMN* conceptual model (conforms to *MMBPMN*) must be translated into an *MSoaML* design model (conforms to *MMSoaML*), and finally this intermediate model must be translated into a *MWSBPEL* operational model (conforms to *MMWSBPEL*). To realize these two sequential translations, we defined two ATL transformations, namely MBPMN2MSoaML (*MT1*) and MSoaML2MWSBPEL (*MT2*).

MT1 was designed in accordance with the BPMN-to-SoaML transformation illustrated in [21], in which the BPMN source model comprises three different views: the structural, behavioral and data view, so that the transformation contains three different transformation types to map each view into the target model. The mapping results are consolidated in one resulting SoaML target model. The SoaML model also contains 3 views: the structural model can be seen as a SoaML participant diagram, behavioral model as the UML activity diagram and the data model as UML class diagram. The transformation of BPMN structural view into SoaML participant model is detailed in [3] and is also implemented in the SHAPE project (http://www.shape-project.eu/), the transformation of the BPMN behavioral view into the UML activity model is quite simple and is described in [11], and the transformation of BPMN data view into UML class model is trivial [21]. So that, the detailed mappings of these transformations is anything but new, and therefore is not included here. Additionally, MT2 is performed by applying mapping rules adapted from [7] that derive data types (XSD), service interfaces (WSDL) and executable business logic (WS-BPEL) from UML models.

Fig. 2 (right side) depicts that in the *quality perspective* the quality attributes model expressed in GRL must be translated into a QoS design model. Next, this intermediate model is translated into a concrete policy description model expressed in some policy definition language, such as WS-Policy. This final model represents how the quality attributes can be enforced at execution time. To realize these model translations we defined two sequential transformations: MGRL2MQoS (MT3) and MQoS2MWSP (MT4). The following subsections will focus on the realization of MT3 and MT4.

D. The MGRL2MQoS Model Transformation

To illustrate MT3, we introduce a simplified security attribute called *SimpleSecurity* (Fig. 3). This high-level model represents a security attribute defined by means of *signed* and *encrypted* communication. It is an abstract view of protection policy assertions proposed in WS-Security Policy [15]. The *SimpleSecurity* GRL model is a graphical representation of the GRL model (conforms to URN metamodel).

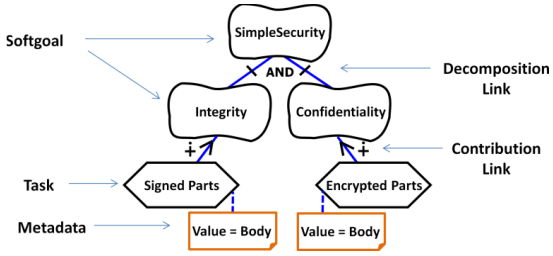

Fig. 3. SimpleSecurity GRL model (graphical notation)

MT3 presents a basic transformation pattern in which a source tree-structure needs to be transformed into a target tree-structure (see Fig. 4). This kind of transformation can be realized by a Depth First Traversal (DFS) algorithm, in which all nodes of the source tree are traversed, to generate the nodes

of the target tree. Generalizing the tree-structures presented in Fig. 4, it was defined the five mapping rules (MR) to MT3:

[MT3.MR1] *The high-level node of the source tree (Intentional Element of Softgoal type) needs to be mapped into a high-level node of the target tree (QoS Context).*

Since a QoS Context allows describing the context of a quality expression by means of multiples QoS Characteristics and its constraints [18], this context can be defined (in an abstract way) through a high-level Softgoal, which in turn is defined by means of quantifiable Softgoals. For instance, the *SimpleSecurity* Softgoal can be understood as an abstract definition of a QoS Context that defines a Simple Security requirement.

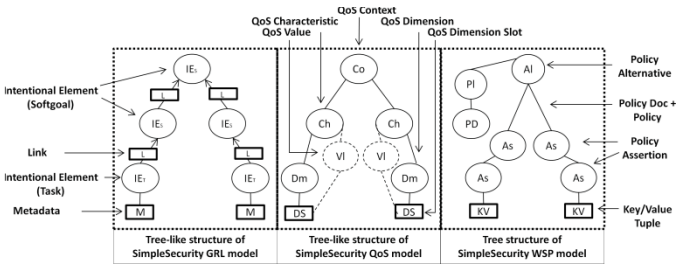

Fig. 4. Tree structure of the SimpleSecurity (GRL, QoS and WSP) models

[MT3.MR2] *The intermediate-level nodes of the source tree (Intentional Elements of Softgoal type) need to be mapped into the intermediate nodes of the target tree (QoS Characteristics), preserving the hierarchy of the nodes.*

The Softgoals that define the abstract context of a quality expression (also a Softgoal) have a direct correspondence with the QoS Characteristics that define a QoS Context. For instance, the *Integrity* Softgoal defines how to address the *SimpleSecurity* abstract context (Softgoal) and has a direct correspondence with the *Integrity* QoS Characteristic that defines the *SimpleSecurity* QoS Context. It is important to note that although not shown in the example in Fig. 4, an intermediate-level Softgoal can be defined as a hierarchy of others Softgoals. So, this hierarchy of Softgoals in the source model needs to be preserved as a hierarchy of QoS Characteristics in the target model.

[MT3.MR3] *The low-level nodes of the source tree (Intentional Element of Task type) need to be mapped to low-level nodes of the target tree (QoS Dimension).*

QoS Characteristics can be quantified in different ways (e.g., absolute values, minimum and maximum values, statistical values) and a QoS Dimension represents a dimension that quantifies a QoS Characteristic. Since Tasks can be used to represent the way to address (or operationalize) Goals and Softgoals [8], Tasks assume the (abstract) role of quantifying Softgoals in GRL. For instance, *Signed Parts* and *Encrypted Parts* Tasks quantify the *Integrity* and *Confidentiality* Softgoals.

The QoS metamodel defines metaclasses to represent the taxonomy of non-functional characteristics (e.g., *QoSCharacteristic*, *QoSDimension*) and also metaclasses to represent the values (instances) for these characteristics (e.g., *QoSValue*, *QoSDimensionSlot*). QoS Value represents an instance of QoS Characteristic and fixes it with specific values (QoS Dimension Slots). In this way, we have identified two additional mapping rules to map the values applied to the quality attributes:

[MT3.MR4] *The intermediate-level nodes of the source tree (Intentional Elements of Softgoal type) also need to be mapped to intermediate-level nodes of the target tree that represents the values of the QoS Characteristics (QoS Values), preserving the hierarchy of the nodes.*

MT3.MR4 is similar to MT3.MR2 since for each QoS Characteristic created from the MT3.MR2, it is also needed to create a QoS Value representing the value of this characteristic. For instance, *Integrity* Softgoal is mapped into *Integrity* QoS Characteristic (MT3.MR2) and also into a QoS Value that represents the *Integrity* QoS Characteristic quantification (MT3.MR4).

[MT3.MR5] *The metadata of low-level nodes of the source tree (Metadata of Intentional Element of Task type) needs to be mapped into the value of low-level nodes of the target tree (QoS Dimension Slot of QoS Dimension).*

QoS Dimension Slots represent possible values that a primitive QoS Dimension can assume, and a QoS Value sets a value for each QoS Dimension of the QoS Characteristic being quantified. Conversely, Tasks can be enriched with Metadata to set the value that a Task must respect. So, the Metadata value of a Task has a direct correspondence with the QoS Dimension Slot defined to a QoS Dimension. For instance, the *SignedParts* Task could assume three values (Header, Body or Attachments), but in the *SimpleSecurity* context it assumes the *Body* value. This specific value is mapped into a *Body* QoS Dimension Slot that defines the value of the *SignedParts* QoS Dimension. The defined mappings were realized by means of five ATL model transformation rules. The complete code of MT3 is available at [23].

E. The MQoS2MWSP Model Transformation

The mapping rules that serve as basis to the *MT4* have a similar, but more direct, transformation pattern than the one defined to the *MT3*. Similarly, there is a source tree structure to be transformed into a target tree structure (see Fig. 4) and an imperative DFS algorithm that is applied to traverse all nodes of the source tree, generating the nodes of the target tree. Generalizing the tree-structures presented in Fig. 4, we have defined the following mapping rules (MR) to MT4:

[MT4.MR1] *The high-level node of the source tree (QoS Context) needs to be mapped into the high-level node of the target tree (Policy Alternative). The Policy Document and the Policy that wraps the Policy Alternative also needs to be created.*

A Policy allows the technical specification of requirements over entities related to a Web service based system. A Policy is defined by a list of Policy Alternatives, which in turn defines a context of constraints on the use of a service. So, it is possible to understand a Policy Alternative as a technical realization of a QoS Context. Since a Policy Alternative is encapsulated by a Policy, within a Policy Document, these last two elements must also be created by the previous mapping rule.

[MT4.MR2] *The quantification of intermediate-level nodes of the source tree (QoS Value of QoS Characteristic) needs to be mapped to intermediate-nodes of the target tree (Policy Assertion).*

A Policy Alternative is a list of Policy Assertions, which in turn represents a requirement, capability, or other property applied to an entity (e.g., endpoint, message, resource, operation) with which a Policy can be linked. Since a QoS Characteristic and its specific QoS Value define a QoS requirement and its applied value, it is direct to map then to a Policy Assertion that defines the same requirement at technical level.

[MT4.MR3] *The quantification of low-level nodes of the source tree (QoS Dimension Slot of QoS Dimension) needs to be mapped into the low-level nodes of the target tree (nested Policy Assertion). The specific value associated with the quantification (Key/Value Tuple) also needs to be created.*

A Policy Assertion can be defined by nested Policy Assertions, and these inner assertions can assume specific values defined by means of a Key/Value Tuple. Since a QoS Dimension and its constrained QoS Dimension Slot are responsible to set a QoS Value, it is feasible to map then to a Policy Assertion and a Key/Value Tuple that defines the property (key) and its value applied to the assertion. The aforementioned mappings were realized by means of three ATL model transformation rules. The complete code of MT4 is available at [23].

III. USING MOQUA

In order to showcase the proposed approach, this section presents an illustrative scenario named WS-I Supply Chain Management (SCM), initially introduced in [30] and then elaborated in [5]. WS-I SCM has been broadly used to validate Web Services related applications [4][10].

A. Scenario Description

The functional specification of the WS-I SCM scenario is a typical business-to-consumer (B2C) one and consists of a Retailer (offers consumer electronic goods to consumers), a Warehouse (stocks items for the Retailer) and a Manufacturer (executes a production run to build finished goods). Fig. 5 depicts the associated business process model (from the Retailer's collaboration perspective).

Considering this scenario, it is relevant to address security issues in order to guarantee that executions occur in a safe environment, which depends on services delivered over the Internet. According to [29], relevant security requirements appear when the Retailer submits an order to the Warehouse (integrity and authentication), when the Warehouse confirms the order to the Retailer (integrity and authentication), when the Warehouse requires a production to the Manufacturer (integrity, authentication and confidentiality) and when the Manufacturer sends the report to the Warehouse (integrity and authentication).

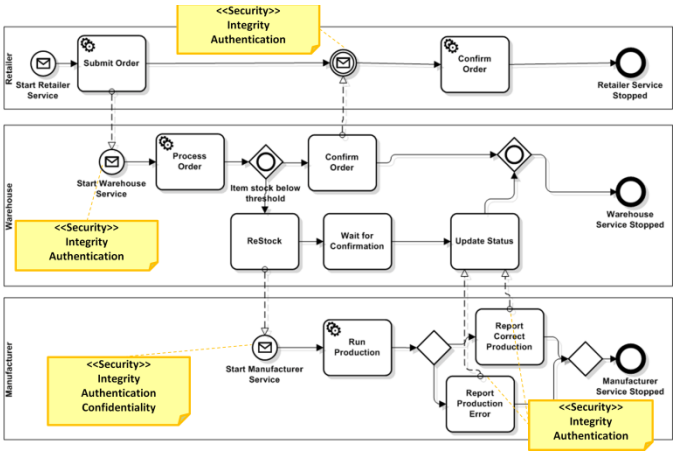

Fig. 5. WS-I SCM: Business Process Model with Security intentions

The non-functional intentions, which are represented by text annotations (metadata) in the business process, can be enriched with specific properties in order to better specify their enforcement [5][30]. For instance, the non-functional intentions Integrity, Confidentiality and Authentication have the properties Integrity Algorithm, Crypto Algorithm and Token Key Algorithm set to "SHA1", "AES256" and "RSA15", respectively. The hierarchical structure of the Security attribute that encompasses these non-functional intentions is represented by the GRL model depicted in Fig. 6. A general Softgoal (Security) is defined through other three Softgoals (Integrity, Confidentiality and Authentication). Each inner Softgoal is defined by means of valuable Tasks (e.g., Crypto Algorithm for Confidentiality, Token Type for Authentication). Finally, the Tasks are enriched with metadata values (e.g., SHA1 for Integrity Algorithm Task).

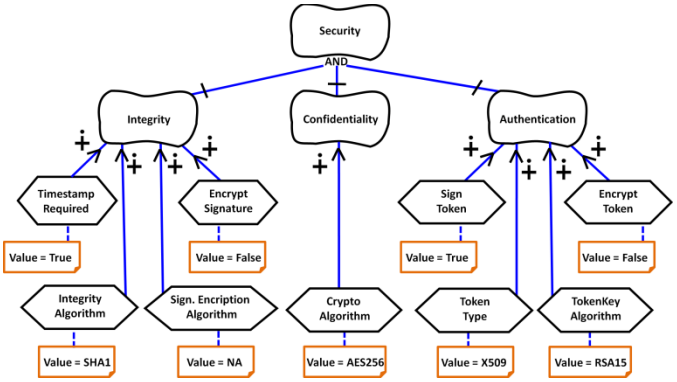

Fig. 6. WS-I SCM: Security attribute in GRL graphical notation

B. Practical Use

Following MOQUA approach, the GRL graphical model of Fig. 6 was projected into a GRL model conforming to URN metamodel (Fig. 7a). This model represents the input for the forward model transformation chain defined for the quality attributes. After applying the MT3, we have as result a QoS model (Fig. 7b) that is represented by a Security QoS Context, based on three QoS Characteristics (Integrity, Confidentiality and Authentication), so that each one is quantified by a set of QoS Dimensions (e.g., Crypto Algorithm for Confidentiality).

It is worth observing that each QoS Characteristic has a valid QoS Dimension Slot (e.g., AES256 for CryptoAlgorithm).

The application of MT4 to the Security QoS model yields the Security policy model (Fig. 7c). This policy model is represented by a Security Policy that has one Policy Alternative defined by three Policy Assertions (Integrity, Confidentiality and Authentication). Each Assertion is quantified by a set of inner Assertions (e.g., Crypto Algorithm for Confidentiality Policy Assertion) with valid values (e.g., AES256 for the Crypto Algorithm Assertion).

The generation of the WS-Policy document (code) is not directly performed from the WSP model by means of a M2T tool as might be expected. In contrast, the WSP model is first transformed into a XML model (according to a metamodel available in the Eclipse MoDisco tool suite) for which a XML extractor is available. This approach represents an alternative mechanism for the generation of (code) artifacts in MDE, in which a model is sequentially translated into different models, being the final model syntactically expressed (concrete syntax) in the desired target format [5].

Listing 1 is an excerpt of the projection (model to text) result of the Security policy model previously defined. It specifies security enforcement mechanisms (and related properties). Configurations are provided both for the authentication token type (lines 1-6) and for the cryptographic algorithm (lines 8-10), which are used, for example, in the communication process between Warehouse and Manufacturer. The complete code is ready to be deployed in a security enforcement module (e.g., Apache Rampart) of a services orchestration engine (e.g., Apache ODE).

Listing 1. Excerpt of WS-SecurityPolicy Document

```
1   ... <sp:InitiatorToken>
2       <wsp:Policy>
3           <sp:X509Token sp:IncludeToken="
    http://schemas.xmlsoap.org /ws/2005/07/
        securitypolicy/IncludeToken/AlwaysToRecipient" />
4       ...
5       </wsp:Policy>
6   </sp:InitiatorToken>
7   ...
8   <sp:AlgorithmSuite>
9       <wsp:Policy> <sp:AES256/> </wsp:Policy>
10  </sp:AlgorithmSuite> ...
```

IV. Related Work

Existing approaches applying model-driven development to realize business processes and their quality attributes have already been proposed. However, most of them do not support quality attributes (in a structured way) since abstract business level (CIM level) or do not provide a clear way to derive final artifacts to enforce the quality attributes at execution level. Some of these approaches are described as follows.

Ortiz and Hernandez [19][20] propose the use of model-driven development and aspect-oriented modeling for treating with extra-functional concerns of Web services. However, unlike our approach, Ortiz's proposal starts from the PIM-level, using the OASIS SCA (Service Component Architecture) specification and a UML profile for modeling services with extra-functional aspects. The UML profile is used to define the PIM, from which a specific service model (based on a JAX-RPC), and three additional specific models (an aspect-oriented one, a policy-based one and a soap tag-based one) are generated. Finally, code skeletons are generated from the specific models by applying additional transformation rules.

Wada et al. [27] use a UML profile (UP-SNFR) to define non-functional requirements inside UML models for SOA. This profile models security in a low level, defining technical security properties (e.g., algorithms, tokens, timeout) of the system. In Wada et al. [28] the original approach is extended, by now starting from a BPMN model (functional) and a Feature model (non-functional) that are weaved to generate the UP-SNFR model. Although defining non-functional properties early, the business and design models share the same abstraction level to define the non-functional aspects. Our approach, however, proposes goal-oriented modeling at business level to represent non-functional intentions of business processes. The goal is to provide an easy and understandable way to represent non-functional expertise at business level.

Gallino [5] proposes a model-driven approach for the development of a service-oriented solution that respects non-functional aspects. The proposal provides a model-driven process in which non-functional intentions are first transformed

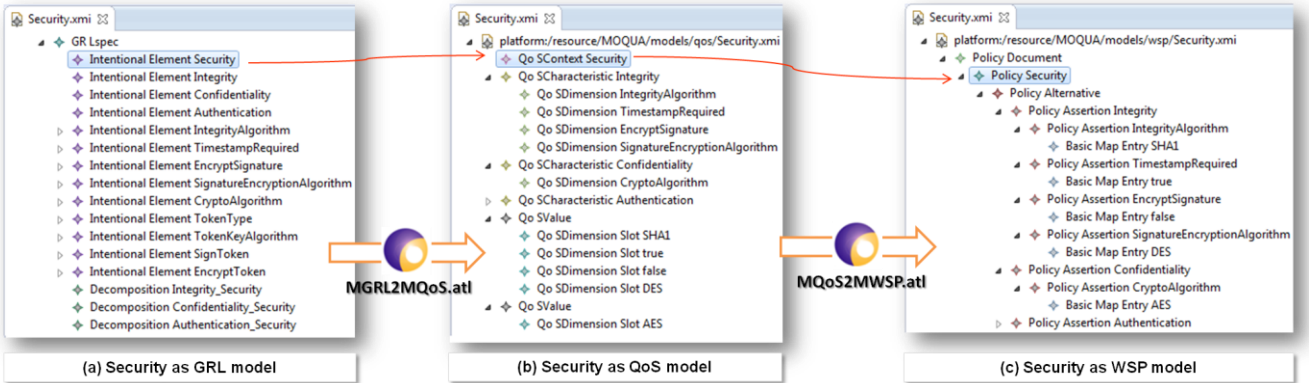

Fig. 7. WS-I SCM: Model Transformation Chain for Security attribute

into a QoS design model (PIM) and finally into a policy model (PSM). The use of well-known standards such as BPMN, QoS&FT and WS-Policy is one of its strengths, which is a point of convergence with us. A main difference to us is that Gallino's solution does not provide the means to model non-functional properties at CIM level, since the non-functional intentions in the BPMN models are directed mapped into QoS-design models (PIM) conform to the QoS&FT metamodel.

V. CONCLUSION AND NEXT STEPS

This paper introduced the MOQUA approach that proposes a forward model-driven engineering process to produce executable business models with quality awareness. In order to carry out the treatment of quality attributes, the proposed approach presents a model transformation chain defined around standards to represent quality attributes, such as GRL, QoS&FT and WS-Policy. The MDE techniques that underlie MOQUA were evaluated against the proposals of the main related work, providing a comparative view of our approach and the ones defined in related work.

The relevance of this work is primarily related to the possibility of dealing with quality attributes since the early stages of software development. Even in the business modeling phase, it is possible to think and to define the quality requirements, enabling an anticipated treatment of potential problems in the later stages of development. For this purpose, GRL models are defined at business-level to represent the quality attributes, and the model-driven process automates the derivation of policies models to represent the restriction over the services that will realize the business processes.

In terms of future work we are investigating: (1) how our approach could be scalable to other quality attributes apart from those related to QoS&FT and WS-Policy; (2) how our approach could address quality attributes for business processes realizable by dynamic service composition, which are popular at services related applications; (3) Finally, in terms of tools, we are evaluating the integration of the different Eclipse tools that support MOQUA architecture, particularly the Eclipse Business Process Management (eBPM).

REFERENCES

[1] Charles, O. and Hollunder, B., Non-Functional Requirements for Business Processes in the Context of SOA, 6th Int. Conf. on Software Engineering Advances, 2011.

[2] Chung, L., Nixon, B. A., Yu, E. and Mylopoulos, J., Non-Functional Requirements in Software Engineering, Int. Series in Sofware Engineering, vol. 5, p. 476. Springer, 1999.

[3] Elvesaeter, B., Panfilenko, D., Jacobi, S. and Hahn, C., Aligning Business and IT Models in SOA using BPMN and SoaML, ACM 1th Int. Workshop in Model-Driven Interoperability, 2010.

[4] Erradi, A., Maheshwari, P., and Tosic, V., Policy-Driven Middleware for Self-Adaptive Web Services Composition, in Middleware 2006, Springer, LNCS, vol. 4290, pp. 62–80.

[5] Gallino, J. P. S., "Application of model-driven techniques to the design of non-functional concerns of service-oriented software systems," PhD Thesis. ETSIT-UPM, Spain, 2012.

[6] Galster, M. and Bucherer, E., A Taxonomy for Identifying and Specifying Non-Functional Requirements in Service-Oriented Development, IEEE Cong. - Services - Part I, 2008.

[7] Gebhart, M. and Bouras, J., Mapping between Service Designs Based on SoaML and Web Service Implementation Artifacts, 7h Int. Conf. on Software Engineering Advances, 2012.

[8] ITU-T, Recommendation Z.151 (10/2012) User Requirements Notation (URN) - Language Definition, Switzerland, 2012.

[9] Jouault, F. and Kurtev, I., Transforming Models with ATL, Proc. of the Model Transformations in Practice Workshop at MoDELS 2005, vol. Satellite, pp. 128–138, 2005.

[10] Kalavathy, G. M., Rathinam, N. E., and Seethalakshmi, P., "Self-adaptable media service architecture for guaranteeing reliable multimedia services," Multimedia Tools Appl. 57, 3, pp. 633–650, 2012.

[11] Lemrabet, Y., Touzi, J. and Bourey, J. -P., "Mapping of BPMN models into UML models using SoaML profile," 8th Int. Conf. of Modeling and Simulation - MOSIMS10, 2010.

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

[13] Mellor, S. J., Scott, K., Uhl, A., Weise, D., MDA Distilled: Principles of Model-Driven Architecture, Addison-Wesley, New York, 2004.

[14] OASIS, Web Services Quality Factors, v.1.0, 2010.

[15] OASIS, Web Services Security Policy, v.1.3, 2009.

[16] OMG, Business Process Model and Notation, v.2.0, 2011.

[17] OMG, SOA Modeling Language, v.1.0.1, 2012.

[18] OMG, UML for modeling QoS and Fault Tolerance, v.1.1, 2008.

[19] Ortiz, G. and Hernandez, J., Service-Oriented Model-Driven Development: Filling the Extra-Functional Property Gap, Int. Conf. on Service Oriented Computing, pp. 471–476, Chicago (USA), 2006.

[20] Ortiz, G. and Hernandez, J., A Case Study on Integrating Extra-Functional Properties in Web Service Model-Driven Development, 2nd I. C. on Internet and Web Applications and Services. IEEE C.S., 2007.

[21] Panfilenko, D. V., Hrom, K., Elvesæter, B. and Landre, E., Model Transformation Recommendations for Service-Oriented Architectures, 'ICEIS (2)' , SciTePress , pp. 248–256, 2013.

[22] Pavlovski, C. J. and Zou, J., Non-Functional Requirements in Business Process Modeling, Asia-Pacific Conference on Conceptual Modelling, pp. 103–112, 2008.

[23] Teles, F., MOQUA: Model-driven Quality Attributes in BPM, Technical Report, pp. 14–35, UFPE, 2013, http://www.cin.ufpe.br/~fst/moqua/ (accessed on April 2014).

[24] van der Aalst, W. M. P., ter Hofstede, A. H. M. and Weske, M., Business Process Management: A Survey, Int. Conf. on BPM (BPM 2003), LNCS 2678, pp. 1–12, 2003.

[25] Vara, J. M., Marcos, E., "A framework for model-driven development of information systems: technical decisions and lessons learned," J. Syst Software 85(10):2368–2384, 2012.

[26] W3C, Web Services Policy 1.5 - Framework, W3C Recommendation, 2007.

[27] Wada, H., Suzuki, J. and Oba, K., Modeling Non-Functional Aspects in Service Oriented Architecture, IEEE Int. Conf. on Services Computing, pp. 222–229, 2006.

[28] Wada, H., Suzuki, J. and Oba, K., Early Aspects for Non-Functional Properties in Service Oriented Business Processes, IEEE Congress on Services, pp. 231–238, 2008.

[29] WS-I, Sample Applications Security Architecture Document, 2006, available at http://www.ws-i.org/deliverables/ (accessed on April 2014).

[30] WS-I, Supply Chain Management Sample Architecture Specification, 2003, http://www.ws-i.org/deliverables/ (accessed on April 2014).

[31] Weiss, M. and Amyot, D., "Business process modeling with URN," Int. J. of E-Business Research 1(3), pp. 63–90, 2005.

[32] Yu, E. S. K., "Towards modelling and reasoning support for early-phase requirements engineering," 3th IEEE Int. Symposium on Requirements Engineering, 1997.