MODELING AND MEASURING SERVICES TO SUPPORT BUSINESS PROCESS EXECUTION: FROM SOA ML AND QoS MODELS TO WEB SERVICES

Andrea Delgado
Instituto de Computación, Facultad de Ingeniería, Universidad de la República, adelgado@fing.edu.uy.

Abstract
When Service Oriented applications and services are modeled, the term Quality of Service (QoS) is used to refer to the collection of constraints and quality requirements for a service. It is important that QoS attributes are specified in early stages of the development process, and modeled in a way that can be recognized and understood by all stakeholders. Regarding Service Oriented Architecture (SOA) modeling, QoS usually includes security, performance and availability. There are many options to realize business processes with services, such as collaborations with partners, internal services, third parties services (SaaS) among others. In previous works we have proposed the automatic generation of services specified in the Service Architecture Modeling Language (SoaML) from business processes, and the associated code. As SoaML allows to model functional requirements only, we have extended our proposal to also take into account QoS modeling and generation from SoaML service models, enriching the specification of services with quality characteristics. We have also added a log facility to the generated WS client to register times regarding services execution, in order to evaluate the defined QoS.

Keywords: Automated service generation, quality of service, business process, BPMN 2.0, SoaML, QoS, Web Services

1. INTRODUCTION
Software modeling is performed traditionally mainly as system documentation, and is hardly updated when changes are introduced in the associated code. The problem with using models as mere documentation is that they rapidly diverge from the reality of the system implemented, becoming obsolete. Model Driven Development (MDD) [1][2][3] focus on software development based on models, using as first order artifacts models, their metamodels and the languages that allow automatic transformations between them. In this way, a model is successively transformed into another model of the same system, refining the level of abstraction until the code level. The Model Driven Architecture (MDA) [4] standard supports MDD defining three levels of abstraction: the Computational Independent Model (CIM) to specify requirements, the Platform Independent Model (PIM) to specify the solution design, and the Platform Specific Model (PSM) adding specific technological aspects for generating the associated code.

One key input model to guide the complete model-driven development process of software systems in organizations is the Business Process (BP) model, being the CIM. Business Process Management (BPM) [5][6] promotes an horizontal vision of organizations based on specifying the BPs they carry out crossing several organizational units. BP models are specified in a suitable notation such as the Business Process Model and Notation (BPMN 2.0) [7], where BP logic is defined by the process control flow which specifies the activities to be performed, their sequence and the possible paths to be selected based on defined conditions. BPs logic is then implemented by services to support the execution of specific parts of the process, ranging from a single activity to a group of related activities or sub-processes, or even a complete process.

Service Oriented Computing (SOC) [8] focus on services for the development of interoperable and distributed applications, implemented within a Service Oriented Architecture (SOA) [9][10], providing great flexibility to both business and technological changes, minimizing their impact into each other. In last year’s there have been many advances regarding technological aspects of services implementation, such as Web Services (WS) technologies, and frameworks to automatically generate code from their Web Services Document Language (WSDL) [11] specification, or from existing annotated code (i.e. Java classes or others), adding specific elements for their deployment within different contexts (i.e. servers such as JBoss, Apache Tomcat, Glassfish, among others).

However, not much have been defined or agreed regarding services modeling for specifying neither functional requirements nor non-functional characteristics. The SoaML (Service Oriented Architecture Modeling Language) [12] standard is one notation that have been developed to fill this gap providing support for services modeling. It is an UML profile and metamodel which defines specific stereotypes for services. SoaML allows
modeling key functional aspects only, regarding the design of services within a software system, but it does not include modeling of non-functional characteristics. For specifying this type of elements, which are known as Quality of Services (QoS) characteristics, some proposals exist such as the QoS&FT [13] standard, which is another UML profile and metamodel providing specific stereotypes to model quality elements of software systems, such as security, performance and availability, among others.

Regarding execution time, there is extensive support for service execution measurement and monitoring (as discussed in section 2 of related work) and Service Level Agreement (SLA) fulfillment, within an application server or other similar execution environments. Also, there is extensive support for the analysis of business process execution based on data registered in the so called execution event logs, e.g. by means of Process Mining [14][15] techniques and tools such as the ProM [16] framework. However, there is little support for assessing services execution within business process execution, from the process point of view.

This article is an extension of [17], which built upon previous work for the generation of services from BPs, mainly [18][19], by adding QoS characteristics to SoaML service models, for both modeling and generation of the corresponding code from the models. Our proposal aims to provide a complete vision based on BP and service models, in which one can navigate from BP models to service models implementing them and vice versa, and from the executable BPs to the corresponding services execution.

The new contributions of this article are: a) extending the definitions provided for the specification of service models in SoaML with QoS that are automatically generated from BPs, and the corresponding code generation, b) adding new concepts regarding the execution and measuring of BPs and services, and our approach for monitoring and evaluating the defined QoS at runtime, to assess BPs and services execution from the BPs point of view, along with the generated code to log services execution, and c) presenting these new elements in the example of application and the set of tools we have developed to support our approach.

We focus on providing complete traceability for functional and non-functional requirements from business to software and throughout the defined models, to the code associated to them, to help in assessing the impact of future changes. Our model-driven development approach is based on MDA using BPMN 2.0 to model (CIM level) and execute (PSM + code levels) BPs (since from version 2.0 the XML version of the model is also executable in BPMN 2.0 engines so we do not need to transform it to WS-BPEL [20] or other executable languages to be able to execute the BP), SoaML (PIM level) to model services, QVT [21] (Query/Views/Transformations) to automate model transformations between them, QoS (also PIM level) to specify non-functional characteristics of services, and generating from the PIM level the associated code (executable) in Java and WS (PSM + code levels). Then execution of BPs invoking the generated services is carried out in a BPMS platform from which internal and external services are invoked, registering execution data for BPs and services. This allows us to analyze real execution data to find improvement opportunities for both BPs and services.

The rest of the article is organized as follows: section 2 discusses related work and section 3 describes key elements of the SoaML and QoS standards for services functional and non-functional modeling. Then, in section 4 we present our SoaML and QoS modeling and code generation approach. In section 5 we present our vision to monitor and evaluate services execution from a business process point of view. Section 6 introduces an example to illustrate the approach and finally section 7 presents conclusions and future work.

2. RELATED WORK

Regarding model-driven approaches to generate service based systems (from business processes or not), several proposals have surfaced in last year’s. We have carried out a systematic literature review [22] in which we have identified existing key ones at the beginning of our research work, such as [23][24][25][26][27][28][29][30], and others that are contemporary to ours such as [31] (see [18]). For a deep discussion on each proposal and key elements to take into account for realizing BPs with services with a model-driven approach see [22][32]. To the best of our knowledge, our proposal is the only one generating SoaML service models from BPMN 2.0 BP models, both for collaborative and orchestration BPs, based on activities (single or grouped), obtaining the associated Java and WS code from the generated SoaML models.

We have also carried out an analysis regarding existing key QoS characteristics for services and BPs execution measurement [33]. We have defined a Business Process Execution Measurement Model (BPEMM) integrating measures for BPs execution grouped in three views: generic, lean and services. We have taken into account key quality characteristics as defined by the SEI in early works such as [34], and the update in [35] which discusses quality characteristics specifically in a SOA. Other early work regarding quality attributes for software is the ISO/IEC 9126 [36] quality model (superseded by the ISO/IEC 25010 SQuaRE [37]). Some other early works are specific for WS QoS characteristics such as [38] from W3C and newer ones including [39] from OASIS. Q-WSDL [40] extends the WSDL metamodel with specific QoS characteristics which are similar to the ones we provide in [33]. The QoS&FT [13] standard from OMG covers a wide spectrum of QoS modeling since it not only defines quality characteristics but also support for specifying quality models for any software systems. There are many other proposals that are not discussed here, [41] presents a systematic literature mapping with an actual and complete view of the QoS state of the art.
To the best of our knowledge, for modeling services with SoaML and QoS few proposals exist: [42] which is contemporary to ours and we have taken into account for modeling aspects, supports the generation of fewer QoS elements than ours, and does not provide code generation or tool support, and [43] which refers to ours and differently uses WS-BPEL specifying QoS in BPs at CIM level, also supporting fewer QoS elements than ours.

3. MODELING AND MONITORING SERVICES AND QoS CHARACTERISTICS

There are not many options for modeling the functional aspects of a service system within a SOA. The SoaML standard, which has gained popularity in last year’s, provides an UML profile and metamodel extending the UML notation with specific concepts and stereotypes to model services. In this way, each concept is unambiguously defined over a well-known UML element, so stakeholders can easily understand what is being modeled. Although for QoS modeling several options exist and since we were already working with OMG standards we decided to use the QoS&FT standard, as it also provides an UML profile and metamodel extending the UML notation with specific concepts and stereotypes for defining QoS quality models. We primarily work with the service execution measures we have defined in [33] adding now the corresponding specification using the QoS&FT standard. Table I shows a selection of key service QoS concepts and definitions [33].

<table>
<thead>
<tr>
<th>Quality characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Time</td>
<td>Guaranteed time interval for the execution of the response of an event [30] in [29]</td>
</tr>
<tr>
<td>Throughput</td>
<td>Number of event responses completed over a given observation interval [30] in [29]</td>
</tr>
<tr>
<td>Capacity</td>
<td>Maximum achievable throughput without violating specified responses time [30] in [29]</td>
</tr>
<tr>
<td>Availability</td>
<td>Service readiness for usage [30] in [29]</td>
</tr>
<tr>
<td>Reliability</td>
<td>Service ability to keep operating over time [30] in [29]</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>Property that data be inaccessible to unauthorized users [30] in [29]</td>
</tr>
<tr>
<td>Integrity</td>
<td>Property that the data be resistant to unauthorized modification [30] in [29]</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Ability to be extended in functionality without affecting other services [31]</td>
</tr>
<tr>
<td>Adaptability</td>
<td>Ability to be changed to adapt to changes in requirements [31]</td>
</tr>
<tr>
<td>Modifiability</td>
<td>Ability to make changes to a system in a quickly and profitable way [31]</td>
</tr>
<tr>
<td>Testability</td>
<td>Degree to which test criteria can be defined and testing can be conducted [31]</td>
</tr>
<tr>
<td>Interoperability</td>
<td>Ability to share information and operate with it under semantic agreements [31]</td>
</tr>
<tr>
<td>Auditability</td>
<td>Degree to which audits can be supported by maintaining system data records [31]</td>
</tr>
<tr>
<td>Operability and Deployability</td>
<td>Ability to operate in an environment of automated operations and recovery [31]</td>
</tr>
</tbody>
</table>

3.1 Modeling functional requirements with SoaML

SoaML defines key concepts regarding services modeling such as provider, consumer, participant, contract, role (provider, consumer), component, interface, service interface, among others. Fig. 1 shows an excerpt of the SoaML metamodel which extends UML with specific concepts for services, such as consumer, provider, service, and request.

It also defines a set of key diagrams to provide several views of the services network, such as: Services Architecture, Services, Service Contracts, Participants and Message Types. The Services Architecture diagram shows the participants involved in the services interaction, the defined services and the roles played by each participant within each service (provider, consumer); in the Services diagram each service is specified by means of its interface, operations, input and output parameters (messages), and uses between interfaces; the Participants diagram adds to each Participant involved the ports in which the defined services are offered or requested (depending on being a provider or consumer); the Service Contracts diagram shows the roles within each service (provider, consumer) and can include a sequence diagram to show the interaction; the Message Type diagram defines the messages to be used as parameters for the operations, their structure, etc.

3.2 Modeling non-functional requirements with QoS

The QoS&FT standard defines two modeling frameworks regarding QoS and Fault Tolerance concepts, from which we will only present the QoS ones, which is the one we use. It defines key concepts that have to be present to specify a QoS model: category, characteristic, dimension, value, constraint (offered, required, contract), among others. Fig. 2 shows an excerpt of the QoS&FT metamodel which extends UML with those key concepts.
It also defines key diagrams to model such quality requirements: QoS category in which non functional characteristics are conceptually grouped (e.g. Performance), showing the QoS characteristics (e.g. Response Time) which are quantifiable and defining their QoS dimensions (Processing Time, Latency), to specify measures over a characteristic. Then when the QoS model is instantiated to define non functional characteristics for a particular system, two more elements have to be specified: QoS value which instantiates a characteristic with a specific value for each defined dimension (e.g. Processing Time = 4000 ms), and QoS constraints to specify whether the QoS defined is: QoS offered (provided by the element), QoS required (required by the element) or belongs to a QoS contract (agreed between participants over an element).

3.3 Adding QoS definitions to WS

There are many standards known as WS* which are extensions to the WS specification, that defines how to add QoS to WS. WS-Policy [44] is a framework that allows defining policies to express conditions for the interaction between providers and consumers. Taking into account the policies exposed by a provider, a consumer can decide whether to use or not a service. A policy contains a list of alternative assertions, where an assertion defines a requirement or capacity, e.g. a certain type of encryption for a message, or the use of a username token for authentication.

WS-Agreement [45] defines a language and a protocol to define service provider's capacities, build agreements and monitor the agreements at runtime. An agreement between a provider and a consumer specifies both the requirements of the consumer and the resource availability and QoS that the provider guarantees. An agreement is defined by a name, a context including participants and duration of the agreement, which is expressed as terms, which can be service or guarantee terms. The first one defines the service to which the terms apply, and the second the QoS agreed over the service. Based on them, management systems are able to monitor services and enforce the agreement.

Fig. 3 shows the definition of a policy regarding the use of username tokens to secure the interaction (a), and the message including the security element defined (b).

WS-Policy [44] provides a set of SOAP extensions to guarantee the integrity and confidentiality of messages. It defines three main security mechanisms: security tokens as part of the message, message integrity and message confidentiality. A security token is a set of claims defined by an entity such as name, identity, key, privileges, among others. To check the origin and integrity of messages signs are used, and for message confidentiality encryption mechanisms are defined.

3.4 Monitoring and evaluating QoS in WS

After the services code is obtained from the generated SoaML and QoS service models as presented before, the business process is implemented and deployed in a BPM System or platform such as Activiti [47] or Bonita [48] (to mention two of the most used open source BPMS nowadays) and the services are linked to the process. In execution time, the business process will invoke the services as defined in the BPMN 2.0 model, and the QoS defined for the services should be registered in order to evaluate their complainant to the specified non-functional requirements.

Monitoring software execution generally implies to collect and analyze execution data on real time, while evaluating software execution can be viewed as a post-mortem analysis of data from a selected period of time.

As presented in [41] a monitor system [49] has to provide two main functions which are: to intercept and to log messages between the involved parties, i.e. services consumers and providers. Also, two strategies can be implemented in such a monitoring tool: passive monitoring...
where the tool sniffs the interaction between parties minimizing the interaction with the monitored system, and active monitoring also known as testing, where the tool also queries the service provider in order to evaluate the QoS defined. Four types of configurations can be set in order to add a monitoring tool to a given execution system [49], regarding the services consumer and provider, and the monitor:

1) Consumer, provider and monitor run all in the same system A
2) Consumer and monitor run in the same system A, the provider runs in a different system B
3) Consumer runs in a system A and the provider and monitor run in a different system B
4) Consumer, provider and monitor run in three different systems A, B and C

Fig. 4 shows these configurations where the consumer role corresponds to the business process (i.e. any service task in the BP which invokes the generated services).

Also, from the point of view of BPs invoking services to execute tasks automatically, several cases can be differentiated depending on whether services are executed inside or outside the organization. Hereby, the following three main cases can be differentiated [19]: (1) a BP invokes a service from a partner organization within a collaborative BP, (2) a BP invokes a service in the cloud (SaaS) that meets the defined criteria both functional and non functional based on QoS defined for the service and (3) a BP invokes a service defined and implemented within the limits of the organization. Case (1) corresponds to a collaboration, cases (2) and (3) correspond to an orchestration (i.e. a single process without defined interactions).

Another option for gathering data of services execution is to register the timestamps for the times in which the service is invoked, waits for the request to be processed, process the request and answers the invocation. Most application servers such as Jboss, Tomcat and Glassfish register data from the execution of applications and Web Services, which can be accessed and processed in order to select those that correspond to the services under evaluation.

4. **SoaML+QoS services modeling and generation**

Our proposal for modeling functional and non functional characteristics for services and automatic generation is depicted in Fig. 5, showing the MDA models we have defined for the CIM, PIM, PSM and corresponding code. The rectangle at the bottom of Fig. 5 containing steps 4, 5 and 6 shows the extensions we have made to our previous work for modeling functional requirements and generation in steps 1.2 and 3 [19]. Steps 4, 5 and 6 add the modeling of non functional requirements and generation, starting at the PIM level by adding QoS characteristics to the SoaML service models [17]. In all steps the BP models are specified in BPMN 2.0 since it is the CIM, PIM and PSM. The execution extension (c.f. section 5) on the right, shows that once the BP is implemented (step 7 by adding elements such as user forms and conditions in gateways), the service tasks will invoke the implemented Web Services (step 8) which are executing, by means of a WS client we also generate for each one. This client also includes code to log execution times for services (step 9), along with BP times (step 9).

4.1 **SoaML+QoS service modeling**

Once the SoaML service models have been generated from the BPMN 2.0 models, QoS characteristics for services can be added to the SoaML diagrams. To obtain the SoaML service models which specify functional requirements for services, steps 1 and 2 of Fig. 5 were performed. In step 1 we model the collaborative and orchestration BPs in BPMN 2.0 and in step 2 we generate the SoaML service models automatically from single or grouped service tasks in the BP models. The provision of services in SoaML is specified by Service ports attached to participants, which are typed with the defined service interface. Service contracts specify agreements between participants over a service.

In Step 4 a QoS model is defined (e.g. [33] or any other), and in Step 5 the QoS are added to the SoaML model, defining the SoaML+QoS model. QoS are added to service ports and to service contracts when they correspond to an agreement over a service. To do so, we specify a QoS value instantiating a QoS characteristic of a QoS model, to a service provider or a consumer, or to service contracts, as needed. In Step 6 we generate Java and WS code from the SoaML or SoaML + QoS models. Fig. 6 shows the domain model for SoaML and QoS key concepts, specified in UML, with QoS concepts on the left and SoaML ones on the right.
Fig. 5. Generation process from BPMN 2.0 to SoaML, and from SoaML or SoaML+QoS to the code extended from [17]

As shown in Fig. 6, on a service provider the QoS offered constraint is added to the service port connected to the defined QoS value. In the same way, the QoS required constraint is added to a request port connected to the defined QoS value, and a QoS contract constraint is added to a service contract connected to the defined QoS value. In this way, it is clear when a service offers or requires a QoS characteristic, and when it corresponds to an agreement between participants. In Table 2 we show the definitions for SoaML and QoS modeling with an example of use.

As it can be seen in Table 2, on the one hand the SoaML model defines two participants a Seller and a Buyer, which has a service and a request port respectively to specify services. The Receive Order service is provided by the seller, and consumed by the buyer (in the Send Order port). A service contract specifies the agreement between both participants, defining the roles provider for the seller and consumer for the buyer. On the other hand the QoS model defines two categories: Performance and Dependability, with characteristics Response Time and Availability respectively. Each one has dimensions with definition formulae to be able to calculate the associated measure.

### TABLE 2 Modeling SoaML and QoS characteristics

<table>
<thead>
<tr>
<th>Participant</th>
<th>Service Port</th>
<th>Request Port</th>
<th>Service Contract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seller</td>
<td>Receive Order</td>
<td>--</td>
<td>Provider</td>
</tr>
<tr>
<td>Buyer</td>
<td>--</td>
<td>Send Order</td>
<td>Consumer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QoS model</th>
<th>QoS characteristic</th>
<th>QoS dimension</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>Response Time</td>
<td>Processing Time</td>
<td>CT - ST(^1)</td>
</tr>
<tr>
<td>Performance</td>
<td>Response Time</td>
<td>Latency Time</td>
<td>ST - ET(^2)</td>
</tr>
<tr>
<td>Performance</td>
<td>Response Time</td>
<td>Response Time</td>
<td>PT + LT</td>
</tr>
<tr>
<td>Dependability</td>
<td>Availability</td>
<td>Availability</td>
<td>MTTF/MTTF-MTR(^3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SoaML + QoS model</th>
<th>SoaML characteristic</th>
<th>QoS constraint</th>
<th>QoS value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receive Order</td>
<td>Response Time</td>
<td>QoS offered</td>
<td>4000 ms</td>
</tr>
<tr>
<td>Send Order</td>
<td>Response Time</td>
<td>QoS required</td>
<td>4000 ms</td>
</tr>
<tr>
<td>Service contract</td>
<td>Availability</td>
<td>QoS contract</td>
<td>99.0 %</td>
</tr>
</tbody>
</table>

\(\text{PT} = \text{Completion Time (CT)} - \text{Start Time (ST)}\)  
\(\text{LT} = \text{Start Time (ST)} - \text{Enabled Time (ET)}\)  
\(\text{MTTF} = \text{Mean Time To Failure}, \text{MTTR} = \text{MT To Repair}\)
Then, to put together both models, we associate each QoS defined to the SoaML element we want. In this case, the provided service in the Receive Order port has the QoS response time as QoS offered with the defined value 4000 ms. In the same way, the consumer has the QoS response time as QoS required added to the Send Order port, with the same value. Finally, a QoS value of 99.0 % is defined for the characteristic availability and added as a QoS contract to the service contract defined between the buyer and seller participants.

It is worth noticing that in our approach and following the definitions in the QoS standard, we allow the definition of different QoS models containing different categories, characteristics and dimensions. To be added to a SoaML model, the selected ones have to be instantiated with defined values to be added to SoaML elements in the service model. Additionally, the same QoS model can be instantiated with different values for different SoaML models, allowing the definition of a complete set of QoS characteristics in a unique QoS model which can be used throughout the organization. This provides the same advantages as having a centralized repository of services that can be used throughout the organization: unifying definitions, avoiding duplications and impacting changes in a single place that can be easily disseminated to the complete organization.

4.2 SoaML+QoS service generation

Once the PIM models are in place, i.e. the SoaML service model, the QoS model, and the SoaML + QoS model, the corresponding code can be generated. We provide two options for doing this: in the first place WS and Java code can be generated directly from SoaML models as shown in Step 3 in Fig. 5. We will not discuss this case here since it corresponds to previous work and is already presented in [18][19]. Secondly, Step 6 shows the QoS generation we have added to our previous definitions, taking as input the SoaML + QoS model and adding specific WS* capacities to the previously defined WSDL and Java generation. The WS* we have analyzed so far and for which we currently provide code generation from SoaML + QoS service models are:

1) WS-Policy generic, which allows the generation of generic policies from different QoS characteristics
2) WS-Agreement for service contracts, which allows specifying agreements between parties regarding a service
3) WS-Security Username Token, which allows to control the access to services

Fig. 7 shows the definition of mappings from SoaML + QoS elements for the first two: (a) WS-Policy, shown on the left and (b) WS-Agreements, shown on the right. A WS-Policy is inserted inside the port tag in the WSDL of the service, as defined in the SoaML port element. From the QoS dimension elements such as the name and unit for the measure are taken, and from the QoS value the actual value defined for the dimension of the QoS characteristic is taken. To define the WS-Agreement, the names of the service contract, provider and consumer are taken from the SoaML service contract element, constituting the general data of the agreement. Also the service name is taken and placed as the name of the service in the service and guaranteed terms specified by the agreement. We also add to the guaranteed terms the name of the QoS dimension of the characteristic and the unit to measure it, which are taken from the QoS dimension element. The corresponding value is taken from the dimension QoS value and also added to the description.

The third WS* corresponds to WS-Security, specifically to the option of security token for authentication. From the

![Fig. 7. Mappings from SoaML + QoS to: (a) WS-Policy and (b) WS-Agreements definitions for QoS generation](image_url)
defined mechanisms we provide generation for the user name token as a security policy which is added to the WSDL. In this case the QoS dimensions and QoS value for the characteristic are not needed, since we will generate specific elements for the deployment of the WS in the selected server and the corresponding client. In the first case we generate a Java class and add information to the descriptors, to use the class as a handler for performing the security checking. As for the client we also generate a class to add the credentials to the SOAP message is generated.

Finally, we have also implemented a specific policy regarding the Response Time for a service, to be able to log the times from the point of view of the consumer. In this case we only add the logging option to the generated client, but no mapping definitions are needed apart from the ones already presented for WS-Policy.

5. QoS MONITORING AND EVALUATION

As presented in section 3.4 there several elements to be taken into account when defining how to tackle the monitoring and evaluation of services execution, based on the defined QoS. A key decision to make is regarding which configuration to use to execute the monitor system, or for service execution times, where to log the corresponding timestamps, and how to process them.

5.1 BPs and service execution and measurement

Fig. 8 shows the execution traces of tasks and associated services from a general case involving two orchestrations (A and B) within a collaboration (C).

![Fig. 8. Execution traces of tasks and associated services within A, B and C processes](image)

From the point of view of Organization A, the execution times that can be easily accessed are: tasks A1 and A3 from the registered data within the process engine executing the process, and service S1 which executes internally in an application server (or other infrastructure) of organization A.

When it comes to execution time measures for task A2 and service S2, which belongs to Organization B, they could be difficult to obtain. Instead, we can directly analyze the total execution time of activity A3 in the process of Organization A, which includes the invocation to activity A2 from the process in Organization B, which in turn is executed invoking its internal service S2. Similarly, in the case of collaborative BPs (1) where instead of invoking internal services, each organization in the participant orchestrations also invokes external services (SaaS) (2), service execution measures regarding execution times could be more difficult to obtain. In these two cases, upper bounds for service times could be better measured using measures from BP tasks invoking them [33], i.e. execution times of task A3 in organization A will be an upper bound for execution times of service S2 in organization B, from the point of view of A. Nevertheless, partners and external providers could also provide their own execution measures to assure the SLA defined for their services.

In the case of a BP in an orchestration invoking internal services from the same organization, a service execution log will be available from the application server (or similar infrastructure) in which services execute. These service logs can be analyzed along with the BP execution log to be able to relate tasks and service execution times. This will provide the organization with an integrate view of the execution of both the defined BPs and the software services and infrastructure that support them, within the selected BPMS platform and server environment. In Fig. 8, this case corresponds from the point of view of Organization A, to the execution times for service S1 which executes in Organization A, invoked from activity A1 from the BP orchestration.

5.2 Relating BP and service execution logs

Most BPMS provide automatic registration for a set of execution core data, such as BP cases and corresponding tasks identifiers, BP cases and tasks start and complete time, and tasks performers. To obtain the execution data registered one only need to query the BPMS database and to format the results to be analyzed into the selected business intelligence tool. However, the monitoring and execution analysis that can be done based on the data registered within the BPMS has a clear focus on the BPs and does not provide support per se for services execution monitoring and analysis. To obtain services execution data we need to select the records referring to the invocation and execution of the services we want to analyze, and this can be done in two different sides: the invocation of the service from the consumer point of view, or the reception of the invocation and processing from the provider point of view.
We can match the timestamps for service execution with typical timestamps related to BP tasks execution such as enabled, started and completed time. In previous work [33] we have defined six base measures to relate BP tasks execution times with the corresponding service execution times for each automated task, to measure the response time of a service. For the reader to understand our approach we present these base measures in Table 3, which are used as a basis for the calculation of the service execution times measures we will discuss in the following. Other measures we have defined regarding other QoS characteristics that are important to measure and analyze within services execution such as Throughput, Capacity, Dependability and Security are not presented here and can be seen in [33].

TABLE 3 Example of base measures defined for service execution times within BPs execution

<table>
<thead>
<tr>
<th>Measure</th>
<th>Measure definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1 (time)</td>
<td>Invoke time of a service from the activity in the BP (IT = timestamp)</td>
</tr>
<tr>
<td>M2 (time)</td>
<td>Enabled time of a service (ET = timestamp)</td>
</tr>
<tr>
<td>M3 (time)</td>
<td>Start time of a service (ST = timestamp)</td>
</tr>
<tr>
<td>M4 (time)</td>
<td>Complete time of a service (CT = timestamp)</td>
</tr>
<tr>
<td>M5 (time)</td>
<td>Fulfilled time of a service (FT = timestamp)</td>
</tr>
<tr>
<td>M6 (time)</td>
<td>Answer time from the service to the activity in the BP (AT = timestamp)</td>
</tr>
</tbody>
</table>

Fig. 9 shows the relation defined between BP tasks execution and service execution, regarding the base measures presented in Table 3.

As discussed in [50], a key issue when analyzing service execution is the correlation between instances. Although in our proposal we are not working with the complete collaborative BP, we also need to correlate the task executed within the BP with the corresponding service instance, i.e. the one that executed with the data sent by the invoking activity from the BP. There is also an important aspect to take into account here regarding the synchronization of the server clocks so we can match times of service execution in the application server with times of activities execution in the BPMS.

The left side of Fig. 9 shows the BPMS process engine where BP tasks are executed, and the right side of Fig. 9 depicts an Application server where the corresponding services are executed. Task A1 in the BP will invoke service S1. Once the execution of task A1 starts (1) the invocation of service S1 is performed at the defined endpoint, and so service S1 receives the invocation (1). The application server then logs the enabled (1.1), start (1.2) and complete (1.3) times for service S1, and after that the service response (2) is returned to task A1, which is then marked as completed (2). From the point of view of the BP, we can also log the invocation of the service in the task A1 which will give us the response time of the service as seen from the BP (consumer of the service).

As mentioned before, from these base measures regarding the response times of service executions within BPs, we can calculate several derived and indicator measures [33] as presented in Table 4.

TABLE 4 Example of derived measures defined for service execution times within BPs execution

<table>
<thead>
<tr>
<th>Measure</th>
<th>Measure definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>M7 (derived)</td>
<td>Service processing time (SPT = CT - ET)</td>
</tr>
<tr>
<td>M8 (derived)</td>
<td>Service latency time (SLAT = ET - AT)</td>
</tr>
<tr>
<td>M9 (derived)</td>
<td>Service response time (SRT = SLAT + SPT)</td>
</tr>
<tr>
<td>M10 (derived)</td>
<td>Service answer time from the BP (SAAT = AT - IT)</td>
</tr>
<tr>
<td>M11 (indicator)</td>
<td>Service response time vs. Service latency time index (SLAT / SRT)</td>
</tr>
<tr>
<td>M12 (indicator)</td>
<td>Average service response time in all BP cases (ASRT)</td>
</tr>
<tr>
<td>M13 (indicator)</td>
<td>Average service answer time in all BP cases (ASAT)</td>
</tr>
</tbody>
</table>

Thereby, derived measures are calculated using other base or derived measures. Indicators, in turn, imply the definition of a decision criterion to evaluate the obtained measurement results, including different ranks and numeric limits, which allow us to give an answer regarding the result. We use labels for the numeric limits that have to be instantiated for each organization, to allow flexibility for taking into account the organizational context.

We also use the semaphore metaphor to indicate whether a result is good (“Green”), average (“Yellow”) or bad (“Red”). In the decision criterion if the times are below the L1 label then the “Green” label is applied, the L1 label will be defined by the organization regarding their goals of service execution times. If the times are between L1 and L2 label then the “Yellow” label is applied, and if the times are above the L2 then the “Red” label is applied. The decision criterion is defined at the end of Table 4 and is used within all the indicators presented in the table.

5.3 Logging service execution from BPs

We have implemented a first step in gathering data from services execution within our generation approach, based on the configuration in which the monitor runs in the same system as the service consumer (cf. section 3.4) with the passive strategy, i.e. logging invocations to the services not invoking them directly. In this approach our logging facility
registers the timestamp before the invocation and the timestamp after the service returns the response.

For doing this, when the code for the service proxy client is generated, it includes the code to log the invocations and responses from the generated services. The proxy client we generate will be included in the implementation of the service tasks in the business process in order to invoke the services generated. Fig. 10 shows our service execution logging approach for a business process executing in an Organization A and for the three cases of services invocation as discussed in section 3.4: Case 1) collaborative BP, Case 2) invocation of SaaS and Case 3) invocation within the organization borders.

Fig. 10. Configuration approach to log service QoS execution from the WS Client running in Organization A

6. EXAMPLE OF APPLICATION

To illustrate the use of our proposal we extended an example regarding a real BP for "Major Ambulatory Surgery (MAS) for a Patient" from a Hospital, adding QoS characteristics to the generated SoaML service model [18][19]. For the QoS model we use as an example our BPEMM [33] proposal (but any other QoS model can be used to define the QoS characteristics to be added to services) instantiating selected characteristics from the service view to show both the modeling and the generation from SoaML + QoS service models. The BP defines three participants: the Hospital, the Patient and the Central Health entity which controls the medical records of the Patient. When an appointment is requested from a Patient, the Hospital registers the assigned date, sends it to the patient and asks the Central Health registry for the medical record of the patient. When the date arrives the patient goes to the Hospital where he undergoes some checking exams to be able to have the surgery. If the results are ok the surgery is performed, if not the patient is returned home and the surgery scheduled for another date.

6.1 SoaML service model

The SoaML service models we have generated from the BPMN 2.0 model can be seen in [18][19]. We generate the complete set of defined diagrams: Services Architecture, Participants with service and request ports, Services with Interfaces, Operations and Parameters, Messages, and Service Contracts. We will present the ones regarding Participants and Service Contracts in the example of the SoaML + QoS model we use as basis for code generation.

6.2 BPEMM service view QoS model

The service view of the BPEMM model defines measures for several QoS characteristics to evaluate service execution. These measures include (cf. Table 1 and Table 3): Response Time, Throughput Time, Capacity, Availability, Reliability and Confidentiality. Each characteristic has dimensions with formulae for the measures, as defined in [33] such as:

![Diagram of Major Ambulatory Surgery (MAS) for a Patient BP from a Hospital to be implemented with services](image-url)
6.3 SoaML + QoS service model

To add QoS characteristic to the SoaML model we use the QoS model of BPEMM we have created, and the Participants and Service Contract diagrams from the SoaML service model we have generated. Fig. 13 shows the SoaML + QoS service model visualized in the SoaML+QoS Eclipse plug-in we have developed (SoaML Toolkit) [51]. It presents two diagrams: (a) Participants with service and request Ports, QoS values for selected characteristics, and QoS offered and required constraint; and (b) Service Contracts with roles provider and consumer, QoS values for selected characteristics and QoS contract constraints.

In Fig. 13 (a) three provided services are shown: two by the Hospital participant which are "ReceiveRequestforAppointment" and "ReceivePatientMedicalRecord", and one by the Central Health participant "ReceiveRequestforPatientMedicalRecord". To all of them we have attached as QoS required the QoS value Access Control of the Security characteristic, defining the authentication for using them. Additionally we added as QoS value for Response Time of 5000 ms as QoS offered to the first service, which is in turn composed of a processing time of 4000 ms and a latency time of 1000 ms, as shown by the dimensions. In Fig. 13 (b) two service contracts are presented for the "ReceiveRequestforAppointment" service and for the "ReceiveRequestforPatientMedicalRecord" service. For both of them a QoS value for Availability is defined and attached as QoS contract constraint. For the second service contract a QoS value for the Throughput time characteristic is also attached.
6.4 Generation from SoaML + QoS service model

We have added the generation of QoS characteristics to our previously developed Eclipse plug-in, extending the dialogs offered to the user with the options we have discussed. Fig. 14 shows the window corresponding to the selection of options regarding the QoS detected on the SoaML + QoS service model.

Fig. 14. Generation options for QoS characteristics

On the left side of Fig. 14 the QoS characteristics that are detected in the SoaML + QoS are shown, which is dynamic allowing new characteristics to be added. On the right side of Fig. 13 the generation options are selected to be applied to each characteristic. This options are fixed as they are implemented using the mappings we have defined, as presented in Section 4. However, since the names and values are taken from the service models, they can be used to generate code for any QoS characteristic which is compatible with the defined mappings (i.e. generating different policies to be included in the WSDL, any agreement between participants, and the basis for defining services security).

As we did in the SoaML tool [52], for each participant a project is generated including now not only the WS and Java definitions but the QoS additions to the WSDL and the new XML for the agreements (if any) and the classes for security. Fig. 14 presents a screenshot of the Eclipse SoaML Toolkit plug-in showing the server and client project generated for the Hospital BP on the left side.

6.5 Logging service QoS execution from BPs

For the WS-Policy Response Time QoS the WS client that is generated includes code for logging the request and response times for the service, from the point of view of the BP which will play the role of service consumer. In Fig. 15 a screenshot from our Eclipse SoaML Toolkit plug-in [51] is presented, showing the generated code for logging the invocations of the service in the client on the right side, along with the log java class on the left side. As defined by ProM, we use the events "start" and "complete", logging the "start" event before invoking the service, and the "complete" event after the answer is received.

Fig. 15. Screenshot from the Eclipse SoaML Toolkit plug-in [51] for services modeling and code generation in SoaML +QoS
To log the times as shown in Fig 15, we added the library log4j2 to the WS client project generated along with the configuration file log4j2.xml in which the file to register the execution logs and its format can be defined, among other things. The configuration file of the log4j2 library was defined following the format MXML/XES [53] from the ProM framework, since we have also developed a plug-in (cf. section 6.6) to evaluate the execution of both BPs and services.

6.6 Evaluating service execution from BPs

For the calculation and visualization of the complete set of measures we have defined within the BPEMM [33], we have developed a ProM BPEMM plug-in [54] which takes as input several files regarding BP and services execution, and shows the measurement results for the measures defined as part of the model. For specific service execution measurement, a log containing services execution data has to be provided. Fig. 15 shows the format of the event log we generate to be imported into the ProM BPEMM plug-in.

This log can come from the client application invoking the service or from the application server executing it, depending on what you want to measure. Also other required input files has to be provided: the BP event execution log (MXML/XES formats), the BP model in BPMN 2.0 format, and a configuration file or context data in which information regarding some values for the measures that are needed to calculate them.

The service execution log file must also be in MXML or XES formats, and can be obtained from the WS client or from the provider in the application server, selecting the timestamps which correspond to the start and complete times of each service execution within each BP case.

Fig. 16. Extract of a generated ProM event log for services

Fig. 17 shows a screenshot of service execution average times for the service “Receive Patient medical record” implementing the corresponding activity, along with the BP cases in which the service was executed. Selecting each BP case from the rectangle with the BP case number, you can visualize graphically the exact execution times for the service within the selected BP case. For now we have not implemented the view providing the information about the relation between BPs activity execution with the corresponding service execution as shown in Fig. 10, we present them in separated views, one for BP activities (see [33] for more information on BPs execution measurement) and a different one for service execution, shown in Fig. 17.

Fig. 16. Screenshot of the ProM BPEMM plug-in with service average times for the service “Receive Patient medical record”
Although with these two views it is possible to show graphically real execution data to analyze services execution within BPs execution, we are working on integrating the view relating BP activities times to the corresponding execution times of the services they invoke, which will provide an easier way to analyze the measures than with the separated views we have now.

7. CONCLUSIONS AND FUTURE WORK

To be able to specify services in a complete way, both functional and non-functional requirements have to be modeled. Although this is not new, it is still a difficult task to find support to integrate both in one model, not to mention the automatic generation of models and code to support the full specification of services. Standards such as SoaML and QoS provide the means to express functional aspects and QoS characteristics of services, respectively. However, they have been mainly used separately without linking concepts from both notations. Our model-driven approach allows specifying services in a SoaML + QoS model, by selecting and adding QoS characteristics from QoS models to the SoaML models we generate automatically from business processes.

We have extended our vision for the automatic development of services from BPMN 2.0 models to SoaML models, adding QoS models, and generating code in Java and WS with QoS. We have also extended our tool support for SoaML adding two modeling plug-ins to specify QoS models and SoaML + QoS models, with associated code generation for selected WS* such as WS Policy, Agreements and Security, allowing a more complete specification of services. We have also included logging facilities in the generated WS client to be able to register the timestamp associated with both the invocation to the service and the response, which we use to evaluate services execution from the BP point of view. Also, processing services log from the provider point of view helps providing more information for the service execution analysis, although we have not included this yet.

The main benefit of our approach is to support both the modeling, code generation and execution measurement and monitoring of services in Java and WS from BP models executing in a BPMS platform. We propose a complete theoretical and practical approach to go through BPMN 2.0 to SoaML + QoS models allowing traceability between all software development artifacts, and to execution. Logging service execution apart from BP execution and analyzing and defining the relation between both executions allow us to present data for analyzing real execution, to find improvement opportunities. As current and future work we continue extending the mappings and generation from BPMN 2.0, SoaML and QoS to support more constructions of each language, and also to generate code for other platforms. We are also working on integrating a monitor tool to provide also service execution analysis in real time.

8. ACKNOWLEDGMENT

I would like to thank the students which participated in the development of the SoaML and QoS Eclipse plug-ins, and the ProM BPEMM plug-in.

9. REFERENCES


Authors

Andrea Delgado received her Ph.D. in Computer Science from the University of Castilla - La Mancha (UCLM), Spain and from Universidad de la República (PEDECIBA), Uruguay (2012). She is an Assistant Professor at the Instituto de Computación, Facultad de Ingeniería, Universidad de la República leading the business process research line. Her main research interests are in business process, service oriented computing and model driven development.