QUALITY REQUIREMENTS FOR SERVICE CHOREOGRAPHIES

Cesare Bartolini[1], Antonia Bertolino[1], Andrea Ciancone[2], Guglielmo De Angelis[1] and Raffaela Mirandola[2]
[1]*CNR – ISTI, Pisa, Italy*
[2]*Politecnico di Milano, Milano, Italy*

Keywords: Choreography, BPMN, Non-functional Requirements, MDE.

Abstract: The growing interest in the Service-oriented Architecture paradigm carries along an increasing popularity of choreographies, a flexible form of service composition to manage interactions within a business process. The BPMN provides and intuitive graphical notation to model choreographies, however there are aspects that BPMN choreography diagrams are unable to display, related not to *what* the participants must do, but to *how* they should do it. These non-functional requirements are usually expressed using separate, though connected, models. This paper introduces Q4BPMN, an approach that aims at improving the expressiveness of choreography diagrams by enhancing them with non-functional annotations. This way, the participants in the choreography will be eased in knowing what is expected from them, and designers can exploit the underlying formalism to support analysis and monitoring facilities.

1 INTRODUCTION

The Service-oriented Architecture (SOA) paradigm is widely adopted for the development of software systems through the dynamic composition of network-accessible loosely-coupled services. In SOA, the development focus shifts from in-house custom design and implementation of system components, to those activities concerning the identification, selection, and composition of services offered by third parties. At the same time, the environment in which software systems operate is highly changing and evolving; hence, the service compositions should dynamically adapt, while meeting both functional and non-functional requirements concerning the required quality of service (QoS) levels.

Whereas the *orchestration* composition (Peltz, 2003) connects services together using ad-hoc language and execution engines, a service *choreography* only describes the expected collaboration and the messages exchanged among services. In the following, we will be referring to the notation for service choreographies defined by OMG in the standard Business Process Model and Notation (BPMN) version 2.0 (Object Management Group, 2010). In that specification, a choreography establishes a "procedural contract" among the participating services, but this definition only concerns functional aspects. Given the increasing importance of quality aspects in service-based systems, we claim that *a choreography specification should also establish quality contractual agreements, or Service Level Agreements (SLAs), at the choreography level*. Such SLAs would not entail a specific concrete service, but rather the abstract description of participants. In other words, SLAs attached to a choreography description would postulate the expected quality levels to be guaranteed between participants playing the choreography roles. As intuitive and simple as such an idea is, we have not found any proposal for it in the literature.

The current practice, in the presence of quality constraints over a choreography, is to use the BPMN language for functional specification of actions flow, and, separately, other languages for the specification of existing quality constraints. Although this approach allows the specification of the requirements to be traced back to the functional specification, such a process requires to be managed by a modeling expert, and does not make the SLAs immediately visible.

In this paper, we propose an approach to directly annotate the BPMN *Choreography Diagram* with quality requirements the services entering the choreography will have to abide by. The benefit of this improvement is twofold: on the one hand, the

Bartolini C., Bertolino A., Ciancone A., De Angelis G. and Mirandola R..
QUALITY REQUIREMENTS FOR SERVICE CHOREOGRAPHIES.
DOI: 10.5220/0003936401430148
In *Proceedings of the 8th International Conference on Web Information Systems and Technologies* (WEBIST-2012), pages 143-148
ISBN: 978-989-8565-08-2

choreography designer can clearly and easily state which are the requirements for the choreography and its roles, at the same level of abstraction of the tasks' flow, and without the need to make use of separate models; on the other hand, services willing to participate as actors in the choreography are immediately aware of what is requested from them. The annotated choreography will then act as a binding contract between the entity managing the choreography and the service providers acting in it.

In the following, we refer to the augmented BPMN notation with its underlying implemented transformation as the Q4BPMN (Quality for BPMN) approach.

2 BACKGROUND

The work presented in this paper builds atop BPMN *choreography diagrams*. The purpose of a choreography diagram is to give a very high-level overview on how the various companies, business units, partners etc. (depending on the actual scope of the business process) relate among themselves. However, BPMN "as is" does not cover non-functional properties.

An interesting related work remarking the need to overcome such a lack of notation is (Bocciarelli and D'Ambrogio, 2011). This paper introduces PYBPMN (Performability-enabled BPMN), a notation for the description of a business process in terms of both functional and non-functional properties, specifically addressing the performance and reliability characterization of a business process. In particular, PYBPMN is a lightweight BPMN extension that addresses the specification of both performance and reliability properties of a business process.

Even though the approach in (Bocciarelli and D'Ambrogio, 2011) and ours look similar, they actually differ for some relevant technical aspects. First of all, the extension provided by PYBPMN only involves those elements of BPMN modeling business processes, those elements natively conceived and used for modelling service orchestrations. Q4BPMN allows to annotate service choreographies with non-functional aspects to provide a binding contract between the entity managing the choreography and the service providers acting in it; thus the BPMN modeling elements that Q4BPMN extends are not covered by PYBPMN. Also, PYBPMN only deals with annotations concerning performance and reliability properties, while Q4BPMN natively supports the definition of both new quality properties, and new metrics evaluating them.

Our approach also relies on a language called Property Meta-Model (PMM), which was defined for specifying observable properties of the system. PMM (Di Marco et al., 2011; Bertolino et al., 2011) describes a property that can be ABSTRACT, DESCRIPTIVE, or PRESCRIPTIVE. An ABSTRACT property indicates a generic property that doesn't specify a required or guaranteed value for an observable or measurable feature of a system. A DESCRIPTIVE property represents a guaranteed/owned property of the system. A PRESCRIPTIVE one indicates a system requirement. In the two latter cases, the property is defined taking into account a relational operator with a specified value.

Properties are grouped into *classes*, differing on which features of the system they address. In the notation proposed in this research, the property class is a feature that allows the classification of the properties into several groups, which generally correspond to the requirement types in the metamodel of some languages designed for this purpose. With respect to this work we will focus on the following classes of properties that PMM defines: performance, security, dependability (Di Marco et al., 2011).

Q4BPMN aims at leveraging the introduced non-functional annotations to apply model transformation methodologies for the generation of analysis-oriented target models (including performance and reliability models). A growing interest focuses on model-driven development, starting from domain-specific source models, possibly augmented with suitable annotations. Many proposals focusing on a particular type of source design-oriented model and a particular type of target analysis-oriented model exist, with the former including, for example, UML, Message Sequence Chart, Use Case Maps, formal or ADL languages, and the latter including Petri nets, queueing networks, layered queueing network, stochastic process algebras, Markov processes (see (Balsamo et al., 2004; Koziolek, 2010; Bartolini et al., 2006)). To have an overview the topic, see for example the WOSP conference series (ACM, 2011).

3 LINKING QUALITY REQUIREMENTS IN CHOREOGRAPHY DIAGRAMS

The idea envisioned in this paper aims at changing the way a BPMN choreography diagram looks, to highlight its quality requirements in addition to the workflow and the interactions between the participating actors. Of course, this change is not supposed to be just a fancy visual change, but its objective is to be an aid for several of the stakeholders:

- the choreography designer will be facilitated in expressing the quality requirements, without the need to create a separate model for them;
- the service provider willing to participate in the choreography under a specific role will have a more immediate understanding of what is required on his/her part;
- the testers, monitors and in general all the stakeholders in charge of verifying that the requirements are met will more efficiently compare the requirements with the properties of the service provided, and will thus be able to easily detect any violations in the contract.

The proposed Q4BPMN annotation must therefore be supported by an underlying model, capable of expressing the quality requirements. The annotated BPMN model can be then used to derive different models using an MDE (France and Rumpe, 2007) approach. The so-obtained Q4BPMN model can be analyzed to obtain information useful both to highlight intrinsic features of the choreography and to detect possible model criticalities. A model designer can use the Q4BPMN information about quality requirements for tasks and participants to evaluate their impact on the overall quality requirements. At the same time, possible participants can use this information to understand what is the quality level required oh their part.

3.1 Q4BPMN Annotations

The annotations we introduce mostly concern *choreography tasks*. A choreography task is not a task in the traditional definition from programming languages, but rather an interaction between two or more actors. A choreography task can generally be split into a set of activities in an activity diagram, or a set of tasks in a BPMN orchestration. This means the non-functional requirements encompass not the single activity executed by one of the partners, but their whole interaction, including the whole set of messages exchanged between them. So, for example, if a requirement states that an activity has a maximum execution time of 100 ms, it means that the whole interaction must occur within that time, starting from the instant the activity is fired. However, it is also possible that an annotation is not related to an activity, but to one of the roles participating in it. For instance, if a specific service is expected to be very reliable, a non-functional requirement could state the maximal admissible rate for the failures the service can raise.

Finally, quality requirements might be related to the whole choreography and not to a single task. In this case, the requirement will involve the choreography as a whole, which could mean that every single task must be compliant with the requirement (in case of requirements that are independent throughout tasks, such as a specific security protocol), or that there is no particular requirement on the single task, as long as the whole choreography requirement is fulfilled (as in the case of cumulating properties such as response time).

Some additional information can be carried along with the annotations. A quality requirement might have an *isHard* attribute, representing the importance of that requirement. An *isHard* attribute set to `true` means that that is a *hard* requirement. Hard requirements are those that must necessarily be met by a service willing to participate in the choreography, lest it be rejected altogether. On the other hand, non-hard requirements are called *soft*, and although they are not strictly mandatory, the respect of these requirements would constitute a preference criterion in the case of two or more services applying for the same role.

3.2 Implementation within MagicDraw

The aforementioned quality annotations have been successfully applied to a BPMN diagram inside the commercial tool MagicDraw[1]. While there is currently no specific BPMN implementation to display the annotations in a clear graphical format (an issue which is currently being discussed with the MagicDraw developers), the current implementation of Q4BPMN is a UML profile that implements the feature of PMM (see Section 2), by defining a set of stereotypes and tagged-values for introducing the quality requirements inside the BPMN diagram.

In the graphical view of the diagram, annotations will be displayed inside the task boxes. The choreography tasks will display the stereotype «Q4Task» or «Q4Participants», and additional details will be shown about the instance representing the requirement. The specific content of the requirement will be expressed in an associated diagram.

To highlight a requirement meant for the whole task, the task will be extended with the «Q4Task» stereotype. Examples of the various annotation types are shown in Figure 2, whereas Figure 3 contains the details of the requirement. The *Accept Request* task shows an example of annotations relating to a choreography task. A requirement meant only for a participant and not for the whole task, on the other hand, will be shown by applying the stereotype «Q4Participants», as shown in the *Request Taxi Service* task. Quality requirements aimed at the

[1] https://www.magicdraw.com/

whole choreography are shown with the stereotype «Q4Choreography». This kind of notation is also extended to those choreography tasks representing a sub-choreography.

In the properties of each requirement the (lack of) presence of an *isHard* attribute shows if the requirement is (not) hard, in the meaning explained above.

Table 1 summarizes all Q4BPMN stereotypes and their meaning on different BPMN elements.

Table 1: Stereotype summary.

Stereotype	Target	Meaning
«Q4Task»	Loop task	Every single execution of the task must be executed under the required circumstances. The requirement is not related to the whole execution of the loop.
«Q4Task»	Sequential task	Every single instance of the task must be executed under the required circumstances. The requirement is not related to the whole execution of the multi-instance task.
«Q4Task»	Parallel task	All instances of the task must be executed under the required circumstances. The requirement is therefore also related to the whole execution of the multi-instance task.
«Q4Task»	Sub-choreography	The whole sub-choreography must be executed under the circumstances mentioned above.
«Q4Participants»	Participant	The participant must fulfill the requirement. The requirement is not task-wide.
«Q4Choreography»	Start event	The whole choreography must be executed under the required circumstances.

4 REFERENCE SCENARIO

Figure 1 shows a sample choreography, developed using MagicDraw. The process depicted in there is an excerpt from one of the demonstration scenarios under development in the CHOReOS European project[2]; it represents a taxi request to a system connected to a number of taxi companies.

The process starts with a citizen requesting a taxi using a Mobile Internet Device (MID participant). Initially, the MID asks the requestor for confirmation about his or her need for transportation. Once the requestor has confirmed that he or she is still in need,

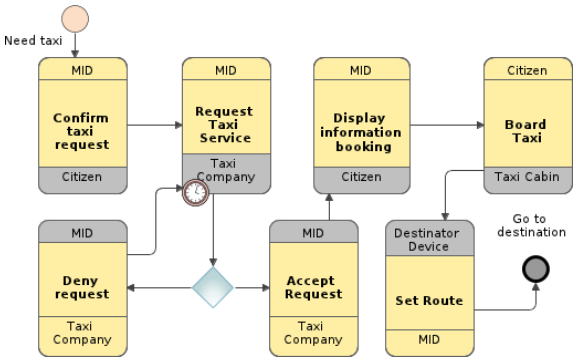

Figure 1: Taxi request example.

the MID tries to forward the request to a taxi company. At this point, the taxi company can either accept or deny the request; in the latter case, the MID will try to forward the request to another taxi company. As soon as a taxi is found, the MID will provide the details to the citizen, and when he or she is on the taxi the route information will be sent to the taxi navigation system.

This simple example hides some issues, not apparent in the diagram, that should be taken into account when deploying the system. For example, the MID participant will expect the taxi company to respond in a rather fast time. A taxi company should have a minimum number of available taxis to be contacted. There might be some requirements on the route set, secure storage of requests, and so on. These requirements are additional to the system's functionality, but the diagram only displays the process flow.

Figure 2 and Figure 3 show the quality requirements annotations in Q4BPMN applied to the referred choreography.

The *Confirm taxi request* task is a direct interaction between the MID and its owner, so it does not have any specific requirement regarding performances, security, or dependability. *Request taxi service*, on the other hand, is surrounded by the following contraints. First of all, the request should occur in a limited time, say 5000 milliseconds: the citizen might be in an urgent need of transportation, and several companies might deny the request before he/she can find one which accepts it. However, a matter of seconds is unlikely to be a problem, so this will be a soft constraint.

The communication between the MID and the taxi company should be encrypted, to protect both the citizen from unauthorized access to personal information, and the company from other companies intercepting the call and "stealing" the customer.

The taxi company known to the MID is supposed to have a minimum number of taxis among its assets.

[2]http://www.choreos.eu/

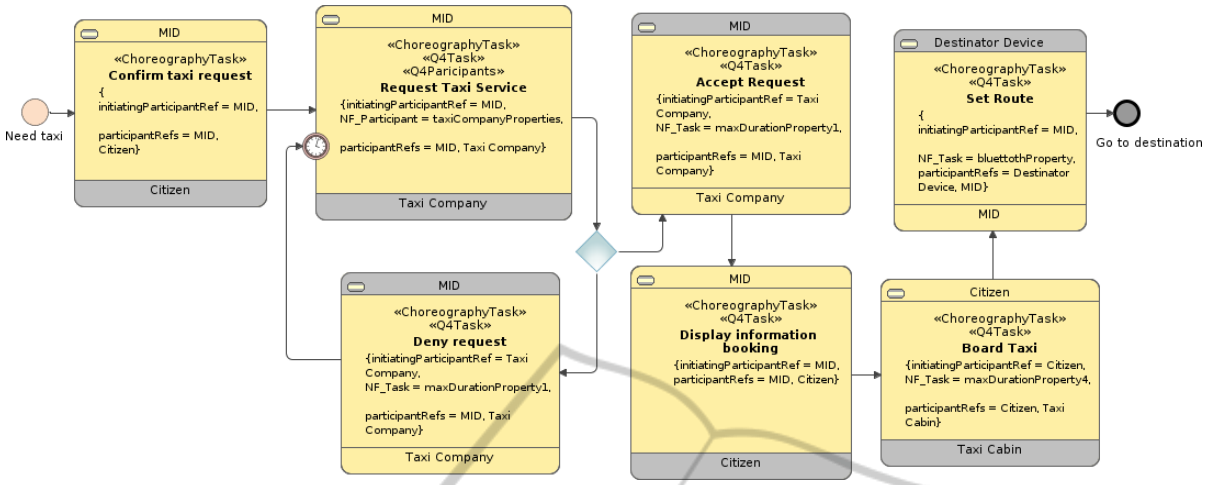

Figure 2: Q4BPMN: Annotated choreography diagram.

For example, the company will be contacted if it has a minimum of 20 functioning taxis at any time, and possibly at least double that (these would be two separate requirements, one hard and one soft).

Also the *Deny request* task is supposed to happen within a limited amount of time, such as 5000 milliseconds, so that the MID can keep on contacting other companies; but again, this will not be a hard requirement. The same holds for *Accept request*.

Once the taxi company has accepted the request and a car is going to pick up the customer, he or she will need to be informed about the booking information. For the aforementioned reasons, this communication must be encrypted in a secure way.

The *Board taxi* task must have a temporal requirement which represents the minimum time the taxi car must wait, in case of the passenger not being present, before being considered available again for another request. Since this requirement defines a behaviour that must be kept by a participant, it is a participant requirement, not a task requirement. The maximum delay time will be 15 minutes; after this, the request is considered no longer valid.

Set route involves an interaction between the MID and the taxi car's navigation device. This involves a communication that should (soft requirement) occur on a protocol known by both devices (Bluetooth in our example), therefore the MID and the taxi car must be equipped with an interface compatible with that protocol. This implies that the communication will be secure based on Bluetooth's pairing procedure, so there will not be a separate requirement for encryption.

In the annotated example, quality requirements are provided also for the choreography as a whole. Specifically, we assumed that a MID cannot fail more than 4 times per 1000 operations; while the dependability requirement for the Destinator Device is set to 5 failures over 1000 operations. With respect to the whole choreography, this example assumes that global execution must complete in 20 minutes, and that the choreography is considered dependabile if it fails no more than once every 100 enactments. Even though Figure 3 shows the specifications of such quality requirements, due to space limitations, images detailing their annotation have been omitted.

5 CONCLUSIONS

This work supports the adoption of non-functional annotations within a BPMN choreography diagram. The paper discusses how all stakeholders can benefit form such annotations : the choreography designers, the service providers interested in playing a role, the monitors, and so on. Then, it details some types of non-functional annotations, retrieving them from previous literature, and provides a possible implementation of the approach.

There is currently no graphical notation which allows to easily distinguish the annotations from the tasks. An ongoing collaboration with NoMagic aims at improving the implementation and the frontend to annotate the diagram, possibly adopting the MagicDraw Open API (No Magic, Inc., 2011).

A useful improvement to this work would be to leverage MDE techniques to automatically derive different quality models starting from BPMN models of the system. Thus, existing analysis methodologies concerning performance, reliability and security, for example (Balsamo et al., 2004; Becker et al., 2009; Woodside et al., 2005; ACM, 2011), in turn may be readily applied "as is".

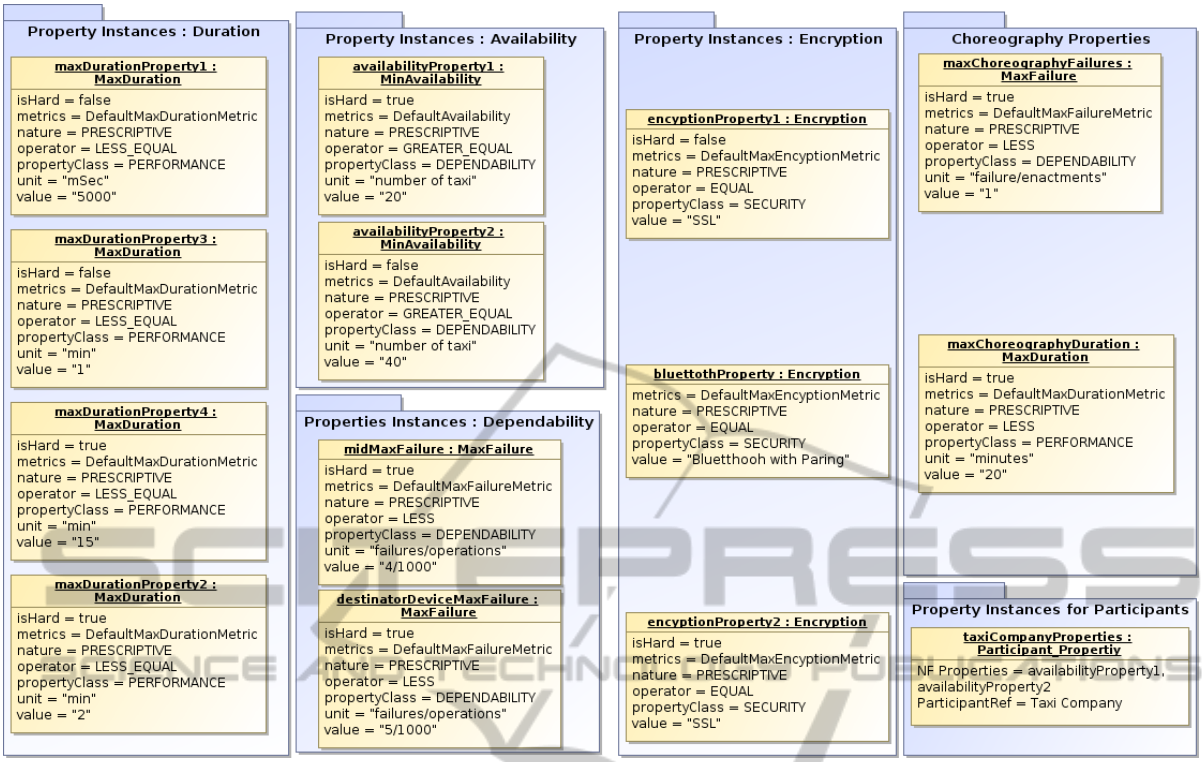

Figure 3: Q4BPMN: Specifications of the quality requirements.

ACKNOWLEDGEMENTS

This paper is supported by the the European Project FP7 IP 257178: CHOReOS, and partially by the European Project FP7 NoE 256980: NESSoS.

REFERENCES

ACM (1998-2011). Proc. of ICPE/WOSP series.

Balsamo, S., Di Marco, A., Inverardi, P., and Simeoni, M. (2004). Model-based performance prediction in software development: A survey. *IEEE TSE*, 30(5):295–310.

Bartolini, C., Bertolino, A., De Angelis, G., and Lipari, G. (2006). A uml profile and a methodology for real–time systems design. In *Proc. of €µ-SEAA*, pages 108–115. IEEE.

Becker, S., Koziolek, H., and Reussner, R. (2009). The Palladio component model for model-driven performance prediction. *Journal of Systems and Software*, 82(1):3–22.

Bertolino, A., Calabrò, A., Lonetti, F., Di Marco, A., and Sabetta, A. (2011). Towards a model-driven infrastructure for runtime monitoring. In *Proc. of SERENE*, pages 130–144. Springer.

Bocciarelli, P. and D'Ambrogio, A. (2011). A BPMN extension for modeling non functional properties of business processes. In *Proc. of TMS-DEVS*, pages 160–168. SCS.

Di Marco, A., Pompilio, C., Bertolino, A., Calabrò, A., Lonetti, F., and Sabetta, A. (2011). Yet another metamodel to specify non-functional properties. In *Proc. of QASBA*, pages 9–16. ACM.

France, R. B. and Rumpe, B. (2007). Model-driven development of complex software: A research roadmap. In Briand, L. C. and Wolf, A. L., editors, *Proc. of ICSE-FOSE*, pages 37–54.

Koziolek, H. (2010). Performance evaluation of component-based software systems: A survey. *Perform. Eval.*, 67(8):634–658.

No Magic, Inc. (2011). OPEN API v. 17.0.1 user guide.

Object Management Group (2010). Business Process Model and Notation (BPMN) Version 2.0.

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, 36(10):46–52.

Woodside, M., Petriu, D. C., Petriu, D. B., Shen, H., Israr, T., and Merseguer, J. (2005). Performance by unified model analysis (PUMA). In *Proc. of WOSP*, pages 1–12. ACM.