Quality-Driven Design of Web Service Business Processes

Tarek Zernadji, Chouki Tibermacine, Foudil Cherif

▶ **To cite this version:**

Tarek Zernadji, Chouki Tibermacine, Foudil Cherif. Quality-Driven Design of Web Service Business Processes. WETICE: Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprise, Jun 2014, Parma, Italy. pp.110-112, 10.1109/WETICE.2014.47 . lirmm-02124407

HAL Id: lirmm-02124407

https://hal-lirmm.ccsd.cnrs.fr/lirmm-02124407

Submitted on 9 May 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Quality-driven Design of Web Service Business Processes

Tarek Zernadji
Computer Science Department
University of Biskra, Algeria
zernadji@yahoo.fr

Chouki Tibermacine
LIRMM, CNRS
and Montpellier II University, France
tibermacin@lirmm.fr

Foudil Cherif
Computer Science Department
University of Biskra, Algeria
foud_cherif@yahoo.fr

***Abstract*—Web service business processes are a kind of service compositions considered as one of the most frequent form of service-oriented software architectures. In this paper, we present a method that helps software architects in the design of such service architectures. This method assists the architects in answering them by proposing some patterns achieving quality requirements. We consider in our work that quality can be achieved through patterns, which are specified with checkable/processable languages. Besides this, the method that we propose simulates the application of these patterns and notifies the architect with its consequences on the other implemented qualities.**

I. Introduction: Problem Statement

Service-orientation provides an architectural style for building service-based software systems. One possible and quite frequent form of service oriented architectures is Web service orchestration. BPEL processes are one of the most largely used technologies for modeling these orchestrations and make them executable.

It was argued that quality requirements drive the design process [1]. So, given the fact that the shape of a software architecture is determined by the desire of achieving certain quality requirements by certain architectural design decisions, it is useful to think of assisting the architects in making these decisions during the architectural design process. We focused our work on service-oriented architectures (SOA), and we considered SOA patterns as the kind of architectural design decisions the architect can make.

In this paper, we propose a process that provides a systematic assistance to architects during the design of Web service orchestrations. The architect should identify quality attribute from the quality requirements specification. Then, given the identified quality attribute some patterns are proposed (Section II-A) to the architect for helping her/him to take design decisions. We argue in this work that for achieving quality attributes, an architect can have as a design decision the selection of an SOA pattern [2]. Our process is thus based on a documentation of design decisions as SOA patterns and their rationale as the quality attributes they implement. This process aims more precisely at helping an architect in choosing the well suited pattern to apply on her/his architecture. It uses a set of evaluation criteria and a quality impact analysis for that purpose (Section II-C). The architect is then assisted in a semi-automatic way to apply the selected pattern (Section II-B) thanks to reusable and customizable scripts defined using a scripting language for Web service orchestrations, named WS-BScript, which is introduced in this paper. The process ends by asking the architect to document (Section II-F) the eventual new design decisions made into her/his architecture so that future designs can be assisted in the same way. Before concluding and presenting some perspectives to our work, we make an overview of the related work (Section III).

II. Proposed Approach

The process that we propose in our work deal with Web service orchestrations design. Through this process the architect is assisted to: i) make concrete changes leading to a refined service orchestration, and ii) perform this with minimal negative effects on existing qualities. The process steps are detailed in the following sections.

A. Pattern Selection

We believe that at the design phase some quality attributes are correlated with functional requirements, hence, they have to be processed at the same time with them. For example, if the architect uses the process we propose, in order to integrate the reliability quality attribute, he may replicate some service partners. Thus, she/he should look for similar service partners that satisfy the same functional requirement. Consequently, those service partners allow to achieve the reliability quality attribute.

So, the architect should identify first the quality attribute she/he wants to implement. This leads to analyze the SOA patterns catalog automatically and results with a collection of patterns, or with no pattern(s). The pattern's specification includes a:

- Pattern Name: The identifier of the pattern and a simple textual description of its role;
- Architectural Script: The set of parameterized actions that indicate the way the pattern can be applied on the architecture. Actions are formalized using a scripting language for Web service orchestration reconfiguration named *"WS-BScript"*[1];
- Architectural Constraints: The list of parameterized constraints that enable to check if the orchestration is compliant with the pattern;
- Quality Attribute: The ISO 9126 quality characteristic that is implemented by the pattern.

[1] https://sites.google.com/site/wsbscript

B. Pattern Application

The selected SOA patterns are applied on a targeted Web service orchestration by means of some scripts, which specify simple architectural changes expressed with a Web service orchestration scripting language called "*WS-BScript*". WS-BScript[2] is a lightweight DSL that enables the patterns catalog administrator, whose responsibility is to feed the pattern catalog, to specify primitive changes making possible the reconfiguration of Web service orchestrations.

In this step of the process, the architect will apply one or several predefined[3] scripts on her/his orchestration. For this end, the architect has to configure the scripts she/he wants to apply by initializing their parameters first and then by customizing them on the fly (through `ask` actions).

C. Quality Impact Analysis

Two key elements are used in the Quality Impact Analysis step of the process: i) a Multi-Criteria Decision Making (MCDM) method, named "WSM" [3] (Weighted Sum Model), to evaluate a number of SOA pattern alternatives and determine the one that best satisfies the architect for a quality requirement, and ii) the solicitation of a quality-oriented assistance service that helps in diagnosing the consequences of any applied pattern on the other implemented qualities.

For the first element, the MCDM problem we want to solve can be expressed as following: "what is the pattern that impacts the less the most important quality attributes, having the best degree of satisfaction for the targeted quality attribute, and is the most suitable to the architect preferences (context suitability, e.g., price, etc.)?"

We have formulated the MCDM problem as: i) alternatives are some selected patterns to classify; ii) decision criteria as a weighted aggregation of:

1) Criticality of the impacted quality attribute;
2) Satisfaction degree of a pattern for a quality attribute;
3) Context-Suitability of the pattern.

For our evaluation purpose using the "WSM" method, we chose to normalize the aforementioned criteria according to the scale proposed in [4]. The later gives eleven scores ranging from 0.045.. to 0.955 and their corresponding linguistic terms from *"Exceptionally low"*.. to *"Exceptionally high"*. This normalization allows us to deal with a single-dimensional case (all the units are the same) of the MCDM problem which fits well the use of "WSM" method. We note here that the patterns in the catalog are previously documented by the architect according to the model proposed in [5]. This model introduces some fine-grained information namely, the criticality degree (a_{iC1}) of a quality attribute, the formalization degree, and the satisfaction degree (a_{iC2}). The documentation is enriched with a context-suitability degree (a_{iC3}), which is specified and documented at design time because it depends on the pattern's suitability to a given situation and to the orchestration. This degree cannot be reused in different orchestrations. It can however be reused in the future design of the same orchestration.

The second element of the quality-related impact analysis step is an assistance service which indicates the related qualities that may be altered when applying a pattern. It is mainly based on the evaluation of some OCL-like constraint that navigate in a metamodel of BPEL. An excerpt of a constraint checking the Service Facade Pattern [6] is given below:

```
1  context AppealedAssessmentsSystem : Process inv :
2  let fst : Activity =
3  if not self.activity->first()->oclIsTypeOf(Sequence) --
       not a sequence
4  then self.activity->first()
5  else if self.activity->first().oclIsTypeOf(Sequence)
6    then self.activity->first().oclAsType(Sequence).
         activity->first()
7    else null
8    endif
9  endif
10 in
11 if fst <> null
12 then fst.oclIsTypeOf(Receive)
13 else false
14 endif
15   -- and ... (check then, that there is no Receive
         activity which is not related to an Invoke
         activity that precedes it)
16   )
17 else true
18 endif)
```

The constraint excerpt checks that the first activity in the BPEL process is a `Receive` (see Line 12). The Service Facade pattern imposes the existence of a single Facade service which receives invocations from client services. All other receive activities must match with previously invoked "server" services.

D. New Patterns Definition

It is on the responsibility of the architect to validate its choice of a specific pattern or to reject it. If the architect is not satisfied with any of the proposed patterns, then she/he can define new patterns (specialization of existing patterns, for example), which she/he is asked to document according to the proposed specification. They will be considered as new reusable architecture design decisions that could potentially be applied on some architecture descriptions in the future.

E. Pattern Cancellation

Applying a selected pattern may lead to unwanted conflicts between quality attributes. Therefore, the architect may want to eliminate the applied pattern that causes conflicts. In this case, the process goes through the pattern cancellation step where an elimination of the concerned pattern is performed. This is done by automatically applying the pattern's opposite architectural actions, hence avoiding to the architect the burden of doing it manually or specifying the cancellation script. The generation of a cancellation script is handled automatically[4] (by the "*WS-BScript*" toolset).

[2]The complete specification can be found here: https://sites.google.com/site/wsbscript/ws-bscript-specification

[3]The patterns scripts are already specified in the patterns catalog by the catalog administrator, the architect has just to apply them.

[4]A complete list of these rules can be found in: https://sites.google.com/site/wsbscript/ws-bscript-cancellation-rules

F. Documentation of the New Architecture

In this step, the chosen pattern is applied to the orchestration and added in the architecture decision documentation as a new design decision. This documentation contains all design decisions (SOA patterns) that was made to build the architecture. In addition, the architect has to complete a part of this documentation, namely the criticality degree of the quality attribute that the pattern implements, the satisfaction degree of the pattern for the quality attribute, the formalization degree of the pattern, and also the related qualities of the quality attribute. This information is necessary for the quality-driven design process especially in the quality impact analysis step.

III. Related Work

Several approaches have been proposed in the literature to address quality requirements integration in software architectures. In [1] the authors use reusable design decisions namely attribute primitives. We use SOA patterns and we give support to the architect to choose among several possible alternatives of a design decision the one that satisfies the best a given quality goal. Besides this, we help the architect in applying the selected design decision in a semi-automatic fashion, and we give her/him assistance to make impact analysis.

In [7] and [8] the authors use a Patterns catalog to document patterns as design decisions. However, their work differs in the way pattern selection and validation is performed. Indeed, in [7] they use questions to help architects in choosing and validating patterns, whereas, we use an MCDM method in a complementary way with a quality-related impact analysis to select and validate patterns. Additionally, our process offers a support to integrate patterns in a semi-automatic way.

In [9], similarly to our work they mapped some quality attributes addressed by SOA patterns [6] to quality attributes from the ISO 9126 quality model. Their work is complementary to our work and could be helpful to the architect especially while building the patterns catalog.

[10] and [9] investigated a quantitative evaluation of the impact of some architectural patterns. In our work, we identify automatically the impact through the solicitation of a quality-oriented assistance service that helps in diagnosing the consequences of any applied pattern on the other implemented qualities.

In [11] the authors presented an approach to Web service (WS) modeling, discovery and selection. They use an Intentional Service Model (ISM) which they enhance with quality aspects to configure the WS discovery and selection process. The selected services satisfy some quality requirements. In our work quality requirements are goals to be achieved in the service orchestration and contribute in its construction. Their work is then complementary to our work.

In [12] they proposed an approach to WS composition that satisfy quality requirements. The result of the composition in their work is a sequence of invocations to services that satisfy dynamic quality attributes achieved at runtime (e.g., response time), while in our work we produce service orchestrations which embody more complex BPEL modeling elements. In addition, in our work we are interested more to static quality attributes (e.g., portability) integrated at design time.

IV. Conclusion and Future Work

In this paper, we presented a quality-driven method to Web service business processes design which uses SOA patterns. We argue that catalogs, such as [6], of design patterns can be documented in a (more or less) structured, automatically checkable and semi-automatically processable way. This method assists architects during the design of Web service orchestrations by suggesting to them the "most" appropriate patterns: i) that respects the more the integrated quality attribute (the pattern that gives the best scores for the evaluation criteria), and ii) that affects the less the other quality requirements already satisfied and documented in the software architecture (through the use of the quality impact analysis). We deal in our work with service-oriented software architectures, which are Web service business processes.

As perspectives to our work, we would like to enhance the organization of the catalog of patterns. Instead of a flat organization, we want to define a hierarchical one, built using some classification techniques like FCA (Formal Concept Analysis). In this way, we can easily look for substitutable patterns which can be proposed together to the architect in the process. Besides this, we plan to integrate in the proposed process an impact analysis activity on the business logic aspect. We can thus also evaluate the impact on the existing functionality implemented in the software architecture.

References

[1] L. Bass, F. Bachmann, and M. Klein, "Quality attribute design primitives and the attribute driven design method," in *Proc. of PFE-4 2001*. Bilbao, Spain: Springer-Verlag, October 2001.

[2] T. Zernadji, C. Tibermacine, and F. Cherif, "Processing the evolution of quality requirements of web service orchestrations: a pattern-based approach," in *Proc. of WICSA'14*. IEEE CS.

[3] P. C. Fishburn, "Additive utilities with incomplete product sets: Application to priorities and assignments," *Operations Research*, vol. 15, no. 3, 1967.

[4] S.-J. J. Chen and C. L. Hwang, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*. Secaucus, NJ, USA: Springer-Verlag, 1992.

[5] C. Tibermacine and T. Zernadji, "Supervising the evolution of web service orchestrations using quality requirements," in *Proc. of ECSA'11*. Essen, Germany: Springer-Verlag, September 2011, pp. 1–16.

[6] T. Erl, *SOA Design Patterns*. Prentice Hall, 2009.

[7] Z. Durdik and R. Reussner, "Position paper: approach for architectural design and modelling with documented design decisions (admd3)," in *Proc. of QoSA '12*, New York, NY, USA, 2012, pp. 49–54.

[8] T. M. Ton That, S. Sadou, and F. Oquendo, "Using Architectural Patterns to Define Architectural Decisions," in *Proc. of WICSA/ECSA'12*, Helsinki, Finland, Aug. 2012, pp. 196–200.

[9] M. Galster and P. Avgeriou, "Qualitative analysis of the impact of soa patterns on quality attributes," in *Proc of QSIC'12*. IEEE, 2012, pp. 167–170.

[10] N. B. Harrison and P. Avgeriou, "Leveraging architecture patterns to satisfy quality attributes," in *Proc. of ECSA'07*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 263–270.

[11] M. Driss, N. Moha, Y. Jamoussi, J.-M. Jzquel, and H. H. B. Ghzala, "A requirement-centric approach to web service modeling, discovery, and selection," in *Proc. of ICSOC'10*. Springer-Verlag, 2010, pp. 258–272.

[12] Z. Azmeh, M. Driss, F. Hamoui, M. Huchard, N. Moha, and C. Tibermacine, "Selection of composable web services driven by user requirements," in *Proc. of ICWS'11*. Washington DC: IEEE CS, July 2011.

View publication stats