Dependability Analysis of Web Service-based Business Processes by Model Transformations

László Gönczy[1]

[1] DMIS, Budapest University of Technology and Economics
Magyar Tudósok krt. 2. H-1117, Budapest, Hungary
gonczy@mit.bme.hu
http://www.mit.bme.hu/~gonczy/

Abstract. A typical composite Web service is built of basic Web services, both internal and external, over which the integrator does not have a complete control. The service-based integration of enterprise systems raises the need for the analysis of non-functional characteristics of a composite Web service. Such an analysis should cover *dependability* (for instance, the reliability of the main process, sensitivity analysis on component reliability, etc.) and *performability* aspects (e.g. determining the optimal process structure for a given level of reliability) as well. The paper discusses how graph transformations can be used to adapt formal methodologies for analyzing SOA models. This way a flexible and extendable analysis framework can be developed for Web service-based business processes.

1 Introduction

SOA-based integration of heterogeneous enterprise systems has a growing popularity, but this raises new problems in assuring the dependability of these systems. The provider of the main service has to guarantee predefined non-functional (mainly dependability related) parameters for his clients. The provider has to consider similar non-functional parameters of external Web services involved in the operation of his main service to be able to calculate and plan the dependability parameters. Building a workflow of composite Web service-based therefore requires describing Service Level Agreement parameters for each service component of the system.

Although specifications for describing QoS parameters exist (such as WSEL [1] or WSOL [2]), the current flow composition languages (e.g. BPEL [3]) and the available design tools ([4], [6] etc.) do not support a standardized extension of process models. Although there were attempts to extend process descriptions (e.g. the specification of WSFL [1] referred to WSEL as a possible way of defining QoS for Web services), currently available implementations do not allow the definition of reliability. Obviously, a dependability analysis needs to be performed on the model additionally to the extension with dependability descriptions of each basic Web service invoked at the main service. The available workflow design tools do not support such analysis and extension.

The current research aims at defining non-functional parameters –such as a dependability description– for process models, and then adapting formal methods to determine dependability characteristics of the system (e.g. the sensitivity of system reliability on the attributes of a given component). Dependability design patterns such as N-Version programming and Recovery Block can also be considered. The process description is based on evolving industrial technologies such as BPEL [3] or WSLA [5].

One possible way of the definition of the requirements against a service and the guaranteed QoS parameters is called Service Level Agreements. Therefore, a language is needed for capturing the following attributes:

- The guaranteed performance parameters of a service, such as *response time* and *throughput*. Both of these can mean a guaranteed minimum or an average value. In the latter case, the exact algorithm of the average calculation also has to be defined.
- The dependability parameters of a service, such as reliability and availability.
- The definition of different levels of the parameters for different groups of users.

Since the main purpose of the current research is not the exact definition of an SLA description language (moreover, the industry is not likely to accept a pure academic suggestion for such an important area), in the following we consider a general SLA description language with the required capabilities. There were attempts to define such a language (WSOL [2], WSEL [1], WSLA [5]), among which the WSLA specification of IBM is very close to fulfill the requirements listed above, although it has not become widespread yet because of the lack of reliable, industrial tool support. The exact process and documentation of SLA negotiation are also out of the scope of this paper just as the exact measurement of the concrete parameters. However, the process of the dependability analysis does not depend on these as the change in the syntax of parameter definition do not cause a change in the transformation algorithm, only the parser has to be changed according to the schema of the related XML language.

1.1 Towards Dependable Web services

The most common dependability parameters −which can be used to describe the non-functional requirements of virtually any kind of service, independently from the nature of the service− are reliability and availability [7]. A mathematical formalism, into which these concepts fit in a natural way, is necessary for the purpose of model analysis. Then the analysis methodology should be able to derive global parameters for the main service model from the components' parameters (given by SLAs).

Using design patterns that are proven in the field of reliability can enhance the dependability of the main service. Such patterns can be, for instance, the N-Version Programming and the Recovery Block scheme [8]. In the case of SOA, the basic variants can be Web services of different platforms, vendors, etc. The applicability of a particular design pattern may, of course, be influenced by the nature of the problem (e.g. in the case of e-banking, multiple execution of a given transaction is undesirable, therefore the N-Version pattern cannot be applied for such a system). The parameters

of a combined service −which is built according to a design pattern− depend on the parameters of the basic variants and the decision algorithm implemented by the pattern.

The task of the dependability analysis is to evaluate a given process model against some requirements (such as reliability). On the other hand, if the available composition possibilities are known, a performability analysis helps to decide which composition is optimal in a particular situation. The criteria of such an analysis are based on the expected QoS level (namely dependability) and the cost of the composed Web service. The analysis methodology should be able to model the effect of including such patterns in the model. The inclusion of such patterns can be based on the result of a dependability analysis, i.e. the bottlenecks of the system can be eliminated and the probability of a system failure can be reduced by replacing a service invocation by a design pattern. For instance, critical tasks can be replaced by a Recovery Block schema. The analysis can be used for two basic purposes; determine the optimal composition for given requirements and determine the guaranteed parameters for a given composition.

2 Frameworks of Formal Analysis

This section discusses possible methodologies which can be used to answer questions against the non-functional properties of the model. The representation formats of the tools implementing these methodologies will be the target languages of the model transformation. This way the integration of an additional tool requires only a transformation from the graph representation of model transformation framework to the representation of the tool. Two possible methodologies are Phased Mission Systems [10] (using GSPN descriptions) and P-graphs [11].

2.1 Phased Mission Systems for Dependability Analysis

Phased Mission Systems (PMS) are a class of systems where the operational life of the system can be considered as a sequence of different phases. The goal of each phase is different, the resources have different characteristics (for instance, different failure rates) and the decisions on system operations can depend on resource states. PMS models can be described as a special subclass of General Stochastic Petri Nets and evaluated according to Markov Regenerative Processes solving algorithm [10].

A Web service-based workflow can be treated as a PMS in a natural way with the following considerations:

- Invocations of services (both external and internal Web service implementations) are phases in the PMS model. The length of a phase depends on the performance metrics of the SLA.
- Parameters determined by the SLAs of the services (such as reliability) are the parameters of resources where the resource configuration during a given phase represents the dependability characteristics of the related Web service.

– The mission profile can be chosen dynamically, ie. the decision about the next phase depends on the system state (whether the result of an invoked service has been proper or erroneous.)

Measures of interest are "Probability that a client request will fail" or "System sensitivity on the availability of a given Web service". These measures correspond to the marking of the Petri Net describing the resource state.

PMS modeling is also able to estimate the effect of implementing design patterns. If the system is particularly sensitive on one Web service, it can be substituted by a Recovery Block which consist of more (typically 2 or 3) variants and an adjudicator. This assumes that there is an available description of semantically equivalent services (in a special UDDI or in the form of an ontology).

2.2 P-Graphs for Cost Optimization

P-Graph description [11] is a formal methodology for describing workflows in chemical industry. A P-graphs is a directed graph in which a node represents a system component with an interface (i.e. inputs and outputs) and a cost, with a similar structure to Petri Nets The accelerated branch and bound algorithm can find the optimal solution for a given problem, which means the optimal structure of system components.

Web service flows can be considered as P-graphs according to the following mappings:

– The Web services are components of the P-Graph.
– The possible structures of Web service flows are built by hand or derived automatically on the basis of a domain-specific ontology.
– The P-Graph solver algorithm can find an optimal solution considering mandatory elements (which must be part of the solution) and prohibited elements (which must not be part of the process).

If the Web services and the main service are given with their SLA parameters, P-Graph analysis can answer questions like "What is the cheapest solution, considering that the request of a certain user must be served by Service X. in Step Y.?".

3 Performing Model Transformations on Composite Web Services

As the current SOA modeling tools (such as [3]), [6] do not support the automated formal analysis of a process model, the integration of formal methods is needed to evaluate the model against non-functional requirements. This integration raises the problem of the evolution of the modeling and analysis tools. The proposed approach is based on model transformation which enables the reuse of model validation and evaluation algorithms available in formal methodology tools.

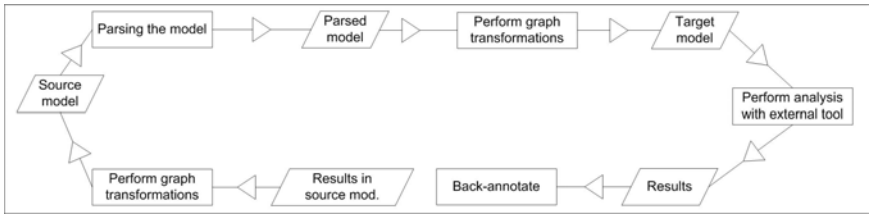

Fig. 1. Evaluating processes with model transformation.

The process of model transformation contains the following steps:

- Parsing the model to the representation format of the model transformation tool.
- The graph representation is the basis of the tool-specific output (e.g., a Petri Net). The output is generated by graph transformations, which take subgraphs of the model and turn them into model elements in the target language (e.g. places, transitions and their rates, guard conditions, etc. in a Petri Net).
- The analysis is performed by the external formal analysis tool (e.g. a Petri Net modeling tool). This can be done automatically, if the tool has an API or the analysis can be started from command prompt. Otherwise, manual start of the analysis is needed.
- The results of the analysis have to be back-annotated to the source model. For instance, the average marking of a place in the Petri Net means the estimated probability of a failure or the average throughput of a transition is the expected throughput of one functionality of the system.
- According to the analysis results, the model can be extended with design patterns for dependability. This can be done within the graph transformation tool.

If the source model is described in a "de facto" standard format (for instance, BPEL), then the analysis features can easily be extended with the integration of new modeling paradigms. This means transformations to new target languages (new implementations for the "Generate output" step).

This process needs a model transformation framework which can be extended to parse the source model (e.g. from a BPEL description), perform graph transformations on the model and generate some output during the transformation. The VIATRA2 framework [9], developed at the BUTE, has a plugin-based architecture which can currently parse UML and BPM models, and a BPEL parser is also under development for the framework. Parsers for additional description languages (for example, WSLA) need only the XSD schema of the language. The tool can perform graph transformations (based on pattern matching) and the execution of these transformations (ie. determining which rule to be used) is controlled by an ASM representation [12].

4 Conclusions

As the current workflow modeling and integration software are not able to capture important non-functional parameters of the system, like dependability, adapting for-

mal methods for standard process description (e.g. BPEL) is needed. With a model transformation framework, such as VIATRA2, the extended process model can be transformed into formal model representation formats such as GSPN, and then be analysed by methodology specific tools such as DEEM.

The presented approach is flexible, as new source models (ie. new process description languages) and new target models (ie. new formal analysis tools) can be seamlessly integrated. The presented ongoing work will be part of a PhD thesis in the field of "Reliable Web services".

Acknowledgements

I would like to say thanks to the group of Prof. Andrea Bondavalli (Andrea Bondavalli, Felicita Di Giandomenico, Silvano Chiaradonna) at the CNR-ISTI research department in Pisa, Italy for the help they provided in the field of Phased Mission Systems.

References

1. Web Services Flow Language (WSFL 1.0) - Appendix C: Endpoint Property Extensibility Elements. IBM Software Group, 2001
   http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
2. V. Tosic, B. Paguerk, K.Patel. WSOL – A Language for the Formal Specification of Various Constraints and Classes of Service for Web Services. Research Report, Carleton University, 2002.
   http://www.sce.carleton.ca/netmanage/papers/TosicEtAlResRepNov2002.pdf
3. Business Process Execution Language for Web Services Version 1.1. 2003
   ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf
4. Business Process Execution Language for Web Services Java Run Time. IBM alphaworks.
   http://alphaworks.ibm.com/tech/bpws4j
5. Web Service Level Agreements Project. http://www.research.ibm.com/wsla/
6. Oracle BPEL Process Manager. Oracle Technology Network
   http://www.oracle.com/technology/products/ias/bpel/index.html
7. J.C. C. Laprie, A. Avizienis, H. Kopetz. Dependability: Basic Concepts and Terminology. Springer-Verlag New York, 1992.
8. A. Avizienis and J. C. Laprie. Dependable computing: from concepts to design diversity. In Proc. IEEE, 74(5):629–638, May 1986.
9. D. Varró, G. Varró, A. Pataricza, "Designing the Automatic Transformation of Visual Languages," Science of Computer Programming, 44:205-227 2002.
10. A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and I. Mura, "Dependability modeling and evaluation of multiple-phased systems, using DEEM," IEEE Transactions on Reliability, 2004.
11. Friedler, F., L. T. Fan, and B. Imreh, "Process Network Synthesis: Problem Definition", Networks, 28(2), 119-124
12. E. Börger and R. Stark. Abstract State Machines. A method for High-Level System Design and Analysis. Springer-Verlag, 2003.