Dependability Evaluation of Web Service-Based Processes[*]

László Gönczy[1], Silvano Chiaradonna[2], Felicita Di Giandomenico[2], András Pataricza[1], Andrea Bondavalli[3], and Tamás Bartha[1]

[1] DMIS, Budapest University of Technology and Economics
Magyar Tudósok krt. 2. H-1117, Budapest, Hungary
gonczy@mit.bme.hu
[2] ISTI-CNR, Via G. Moruzzi 1, I-56124 Pisa, Italy
+39 50 315 2904
{silvano.chiaradonna, felicita.digiandomenico}@isti.cnr.it
[3] DSI - Universita' di Firenze, Viale Morgagni 65, I-50134 Firenze, Italy
+39 55 479 6776
bondavalli@unifi.it

Abstract. As Web service-based system integration recently became the mainstream approach to create composite services, the dependability of such systems becomes more and more crucial. Therefore, extensions of the common service composition techniques are urgently needed in order to cover dependability aspects and a core concept for the dependability estimation of the target composite service. Since Web services-based workflows fit into the class of systems composed of multiple phases, this paper attempts to apply methodologies and tools for dependability analysis of Multiple Phased Systems (MPS) to this emerging category of dependability critical systems. The paper shows how this dependability analysis constitutes a very useful support to the service provider in choosing the most appropriate service alternatives to build up its own composite service.

1 Introduction

Recently, the main paradigm of creating large scale information systems is shifting more and more towards integrating services instead of integrating components as in traditional technologies. Open standards like Web Service Description Language (WSDL) assure system interoperability. This integration and development paradigm is called Service Oriented Architecture (SOA). The top level description of a SOA process describes the main business logic and it is usually very close to the traditional business process models (BPM). Recent development tools provide a quite powerful support for functional service integration but they lack the support of the description and analysis of the non-functional aspects in the system. However, service level

[*] This work was partially supported by the HIDENETS European project (IST-FP6-STREP-26979) and the "Quality of Service and Dependable Computer Networks" Project of the Italian-Hungarian Intergovernmental S&T Cooperation Programme.

integration raises new problems as the service provider is composing its main services from elementary services as building blocks without having a complete control over them. Thus, the result of the main service may be invalidated by simple faults and errors in imported services. Dependability analysis has to focus on creating a system-wide dependability model of the component models and evaluating the impact of the faults in the individual components, including the identification of dependability bottlenecks and the sensitivity analysis of the overall system to the components' dependability characteristics [1].

Based on the observation that Web services-based workflows fit into the class of systems composed of multiple operational phases characterized by potentially different requirements and goals, the paper attempts to apply methodologies and tools for dependability analysis based on the paradigm of Multiple Phased Systems (MPS, [2], [6], [7]) to this emerging category of dependability critical systems. A methodology for transforming workflow description of composite Web services into an MPS description is proposed, and, once such a description is derived, appropriate tools for MPS modeling and evaluation are applied to quantitatively assess specified dependability indicators. Hereby we use the DEEM tool to describe dependability models and to evaluate the indicators.

This paper is organized as follows. Section 2 describes the Web service flows and exposes the need for evaluating dependability indicators and discusses the related work. Section 3 introduces the MPS paradigm and the DEEM tool for dependability analysis. Section 4 discusses the possible ways of combining Web service flows as an implementation-close description of processes running in a distributed environment and MPS as a dependability modeling paradigm. Section 5 describes the model transformations performed in the VIATRA 2 framework [5] which enables a (semi-) automatic transformation of business processes (such as those built of Web services) to formal analytical models (e.g. Deterministic and Stochastic Petri Nets). Section 6 illustrates the methodology by a case study. Section 7 concludes the paper and summarizes further research directions.

2 Dependability Aspects of Web Service Flows

Present BPM tools (e.g. [26]) enable performance analysis/simulation with the restriction that all resources are available. Therefore, no faulty states can be modeled in a consistent way, failure rates and repair times cannot be considered during the analysis and no dependability analysis can be performed on the model. Similarly, there is a lack of error handling, despite the fact that some languages (for instance, Business Process Execution Language [8]) can handle exceptions. A BPEL exception handling routine, however, may contain only compensation actions which try to eliminate the effect of an uncommitted transaction, transaction time-outs, non-atomic operations etc.

The service-based approach to system integration raises the problem of defining Service Level Agreements [9] (SLA) between the provider and user of the main service. In this context, SLAs are used to describe the required quantitative parameters of a service, related to a particular client or class of clients. In general, an SLA contains measurement objectives, their guaranteed values, a measurement methodology and some goals and obligations for the participating parties. An SLA can be attached to all service invocations, described as simple activities in a BPM, although no unified formalization of such documents exists.

As the service level of the system depends on external providers, a standardized description of the QoS parameters of Web services is needed in order to have a consistent view of the QoS at the level of composed services. Several descriptions of the QoS parameters of Web services were proposed, e.g. in [11], [12], and [13], however, no single, standardized description format was generally accepted. Accordingly, in the current paper, no specific syntax will be assumed on the service quality description, but merely only those core concepts which can be found in an arbitrary QoS definition will be referred.

To illustrate the importance of dependability analysis of Service Oriented Architecture, consider a sample process of three simple steps: receiving a request, forwarding it to an external partner ("outsourcing") and then returning the answer to the client. The first and the last activities use internal resources (e.g. a Web server) having known performance and dependability characteristics. The second activity is however deployed on an external system, therefore its resource usage is unknown, it is described only by its Service Level Agreement parameters, for instance, "AverageResponseTime" etc. The provider of the main service has to estimate the guaranteed QoS parameters of his service, for instance the failure rate, which depends on the failure rates of the internal resources and the failure rate of the external service over which he has no control.

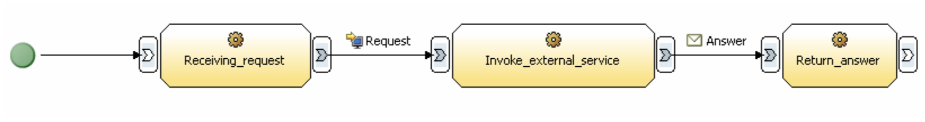


Fig. 1. A sample process with external method invocation

The model based analysis of a process necessitates additional information to the basic functionality of the process, such as failure rates of components, required and guaranteed response times, availability, repair times (for instance, time interval between retrying to invoke a service), etc. Non-functional design patterns such as Recovery Block should also be considered, e.g. in case of a failure, the invocation of the fastest service can be followed by calling a slower but more robust variant. The system dependability model [14] has to be created from rather different engineering models. The external service is described only by a black box model, describing the functional interface, performance and dependability-related quantitative parameters while the internal services have to be extracted from a model indicating both the functionality and the deployment to resources, extended by the non functional parameters. A uniform system-wide model has to be derived from these engineering models for the further analysis.

Available extensions of BPM lack a support of the usage of dependability parameters and fault tolerance patterns in the model. There are, however, numerous evolving standards, specifications and research in this field such as [17]. XML-based description languages such as WS-Reliability[16], WS-BaseFaults [18] can describe the characteristics of a certain endpoint, i.e. a Web service or a port/method of a Web service. These descriptions can be associated with WSDL files [19].

As the actual description language is irrelevant from the analysis point of view, a general description language is adopted among the several emerging languages and technologies such as Web Service Level Agreements (WSLA) which contains language elements for at least a subset of the performance and dependability characteristics. As this language is quite flexible and extensible, we propose to use this for the description of non-functional parameters, as this way the external services and the internal resources could be characterized using the same description, using Service Level Agreements. A typical SLA contains the objectives to be measured, such as the transactional throughput of a web server or the response time of a remote web service, the measurement algorithm (e.g. how to compose an average measure), the fee of the service and the punishments related to violating the requirements. Guaranteed values of SLA parameters can be negotiated with the client. After such a negotiation, a complex process can be composed based on elements having well-defined QoS guarantees. The process of this negotiation is out of the scope of this paper; several ideas are discussed in [21], [22]. In our research the emphasis is on the evaluation of the process models extended with the dependability description. Hereby we suppose a WSLA-like description for the resources and the services.

Evaluation of Web Service compositions has been addressed in the literature by using Petri Net-based techniques ([29], [31]), Timed Automata [30], non-deterministic automata [32] or some kind of pi-calculus [31], [33]. PEPA models are also used to derivate quantitative characteristics of the systems and are the basis of SLA evaluation [28]. However, to our best knowledge, none of these were applied directly on a high-level (for instance, BPM) description to perform quantitative dependability analysis without the need to create a lower level model of the system, only basic verification is fully automatized.

The availability of a versatile and highly efficient tool dealing with dependability analysis of Multiple Phased Systems, combined with the appropriateness to include web-service based processes in the category of MPS systems as shown in the sequel, motivated the choices at the basis of our work.

3 Dependability Modeling: Multiple Phased Systems and DEEM

This paper elaborates on characterizing Web services-based workflows as Multiple-Phased Systems for the purpose of dependability analysis. In this section, a brief overview of MPS is provided, together with a short description of the tool DEEM for the dependability analysis of MPS.

Multiple-Phased Systems (MPS) is a class of systems whose operational life can be partitioned in a set of disjoint periods, called "phases". During each phase, MPS execute tasks, which may be completely different from those performed within other phases. The performance and dependability requirements of MPS (such as throughput, response time, availability, etc.) can be utterly different from one phase to another. The configuration of MPS may change over time, in accordance with performance and dependability requirements of the phase being currently executed, or simply to be more resilient to an hazardous external environment. As the so-called MPS goal may change over time, the sequence of phases of which the MPS execution is composed (the execution of a given workflow) may depend on the state (such as success/failure) of previous phases. Phased Mission Systems (PMS) and Scheduled Maintenance Systems (SMS) are two typical subtypes of MPS. Examples of MPS can be found in various application domains, such as nuclear, aerospace, telecommunications, transportation, electronics, and many other industrial fields. Because of their deployment in several critical application domains, MPS have been widely investigated, and their dependability analysis has been the object of several research studies ([2], [3], [4], [6], [7], the complete expose of the literature can be found in [2]).

Recently, a dependability modeling and evaluation tool, DEEM, specifically tailored for MPS, has been developed at the University of Florence, and ISTI-CNR [4]. DEEM relies upon Deterministic and Stochastic Petri Nets (DSPN) as the modeling formalism, and on Markov Regenerative Processes (MRGP) for the model solution [2]. When compared to existing general-purpose tools based on similar formalisms, DEEM offers advantages on both the modeling side (sub-models neatly model the phase-dependent behaviors of MPS), and on the evaluation side (a specialized algorithm allows a considerable reduction of the solution cost and time).

The rich set of modeling features provides DEEM with a two-level modeling approach in which two logically separate parts are used to represent MPS models. One is the SystemNet (SN), which represents the resource states and the failure/repair behavior of system components for each phase, and the other is the PhaseNet (PhN), which represents the execution of the various phases, as illustrated later in Figure 6. Each net is made dependent on the other one by marking-dependent predicates which modify transition rates, enabling conditions, transition probabilities, multiplicity functions, etc., to model the specific MPS features.

In DEEM, very general dependability measures for the MPS evaluation can be defined by a reward function. Among the measures assessable through such approach are the probability of successful mission completion, the relative impact of each single phase on the overall dependability figures, and the amount of useful work that can be carried out within the mission. The main motivation of using DEEM was that –as it was shown in [2]– it is a versatile and highly efficient tool for dependability modeling and evaluation of MPS systems.

4 Combining BPM as a Modeling Language and MPS as a Dependability Analysis Paradigm

As it was discussed earlier in Sect. 2., BPM models can be extended to capture non-functional parameters of the system. Therefore, an approach is needed which exploits the possibility of modeling multiple states of a resource. The modeling methodology and the evaluation procedure implemented in DEEM allow to describe the flow models of the web service systems and to analyze their dependability attributes, as shown in the next subsections.

4.1 Considering Business Process Flows as MPS

Processes in the SOA context can be considered as Multiple Phased Systems in a natural way. The two layer-representation of Multiple Phased Systems corresponds exactly to the logic of workflow-like integrated component services. The upper layer corresponds to the workflow sequence consisting of the sub-service elements while the detailed model may be used to describe the individual component services. Remind that the possibility of splitting the description into functional and non-functional aspects allows the natural expression of different parameterization of the invocation of the services and data-dependent branching in the main workflow. To illustrate the modeling issues of a business process, a more detailed view of the process in Fig.1. is presented, as shown in Figure 2.

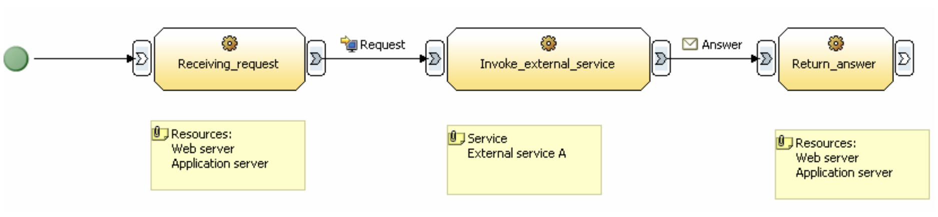


Fig. 2. Sample business process and the underlying resources

The “Receiving request” and “Return answer” activities are executed locally, i.e. on controllable and observable internal resources. Resource parameters can be modeled in a standard way using the General Resource Model UML [20] profile of the OMG. The resource faults and their effects can also be described by using the notations defined in this profile as it was shown in [10], [15]. The “Invoke external service” activity corresponds to a Web service invocation, therefore the quantitative parameters of this activity can be derived from the SLA descriptions of the service as pointed out in Sect. 2.

The dynamics of the business process flow can be treated as a Multiple Phased System in the following way. The phases (partitions of system operation) are the tasks of the BPM, unless consecutive tasks use the same internal resource. This way the context of the operation (the environment of the mission) will be different for each phase. The resource parameters –appearing in SystemNet if modeled in DEEM– such as failure rates, repair times, number of identical resources (e.g. the number of possible retransmissions of a request or the number of Web servers) may depend on the operation context, thus on the actual phase. The mission goal can change over the time and the execution of the process (i.e. the mission goal) may depend on the result of previous phases and the system state. For instance, if validating a credit card does not terminate within a predefined timeout, then a flight ticket reservation cannot be confirmed, but is saved as a conditional reservation.

The chosen method to evaluate the dependability of the web service systems is describing the model in BPM and transforming it to a MPS model, since the basic description language –in which the process is built– is easy to use, a wide range of tools are available, and an implementation skeleton (i.e. the workflow control description) can be generated directly from the model after passing the dependability analysis. Moreover, BPM activities can be converted into phases of a MPS in a quite natural way while the opposite direction (i.e. generating the skeleton of a control flow from a MPS model) raises several questions as many activities can be described in one phase, if their dependability parameters are the same. Using model transformations instead of describing the model in a meta-language brings the benefit of easy implementation of additional transformations (for instance, model checking based on qualitative properties of the mode) as the model is stored in the format of the graph transformation tool. The transformation tool also enables the generation of practically any type of output (which can be further the basis of a runtime validation).

The basic BPM model should be extended with some information about the required behavior of components (basic activities), such as maximum response time, maximum number of timeouts in a given time interval, guaranteed rate of good answers for a prefixed number of requests, availability, etc. All these characteristics can be derived from the characteristics of the resources and the software implementation (if known) in the case of internal services (those we have control over) or from the Service Level Agreements in the case of external service. The business process model can be transformed to a MPS according to the following rules:

- The different activities will be different phases in PhaseNet with different goals, dependability metrics and resources.
- The performance and dependability characteristics of resources and services will determine the SystemNet parameters such as transition rate, initial marking, etc.
- The dependencies between PhaseNet and SystemNet are given by the task-resource bindings and SLAs of the (both internal and external) services.
- The measurements of a MPS analysis are determined by the "business measures" of the BPM, i.e. the QoS parameters of the main service.

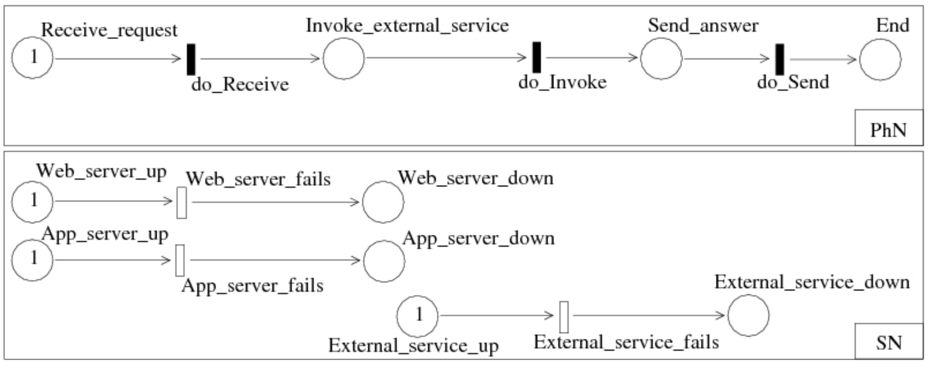


Fig. 3. The sample process seen as a MPS system and modeled in DEEM

Figure 3 shows the sample process as a Multiple Phased System. Note that the parameters of the DSPN are derived from the BPM parameters based on the resource descriptions and SLAs. The expected durations of the timed transitions of the PhaseNet correspond to the estimated execution time for activities of the business process. For instance, the expected duration of the transition "do_Receive" results from the average execution time of "Receive request" activity while the expected duration of the transition “do_Invoke” is derived from the average response time of the external service, described in the corresponding SLA. Error manifestation is expected to happen at the invocation of an operation using a resource. Resource faults inducing errors are modeled by the timed transitions of the SystemNet while the enabling conditions of these transitions model the resource allocation.

For instance, the transition “Web server fails” is enabled during the phases which correspond to tasks using the web server while the transition rate corresponds to the expected failure rate of a server during a typical transaction. These parameters are represented in the IBM WBI as the “description” of the resources (since the definition of failure rates of resources is not supported by the present BPM tool). For external services, such as “External service” in this example, the failure rate is derived from the Service Level Agreement, also using a textual field in the modeling tool. The main difference between the internal resource usage and the external service invocation is that in the former case, multiple resources can be used simultaneously and a fault in any of them prohibits the proper service while in the latter case only a single service and –at most– one resource, namely the application server is used.

5 Model Transformations with VIATRA2

As it was mentioned in Sect. 4., the dependability indicators of business processes can be evaluated if they are transformed into a MPS model. The model transformation-based analysis of business process descriptions consists of the steps shown in Figure 4.

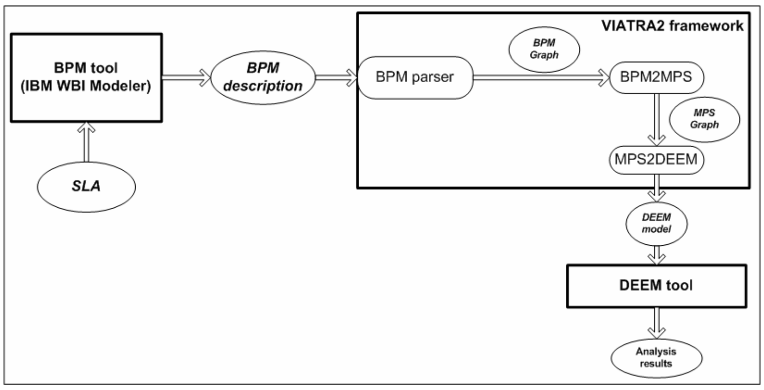


Fig. 4. Transformation of business processes

Using model transformations for the analysis of high-level system models is part of the Model Driven Architecture (MDA) concept. The motivation for using model transformations in the VIATRA (Visual Automated Model Transformations) framework was the *extensibility* of the transformation engine by additional parsers and plugins which enable the decoupling of the format of the source model and the target analysis platform.

First, the engineering model of the functionality –enriched by dependability parameters– is taken to be analyzed by formal methods. This model will be generated by a BPM tool [26] and transformed by the VIATRA2 framework into a DEEM model. To build a mathematical model from the high level business description in an automated way, the BPM is parsed into a graph representation ("BPM Graph"). In our case, this will be the inner representation language of the VIATRA2 tool, which is a public domain model transformation framework, developed at BUTE [23]. It is now part of the Eclipse GMT project [5]. In the VIATRA2, graph pattern matching [24] is controlled by Abstract State Machines [25].

Then, graph transformations are performed upon this parsed model in order to generate a graph which represents the relevant elements of the system in the target paradigm ("MPS Graph"). In this case, the target paradigm is MPS and the target model representation format is that of the DEEM tool. However, as it will be discussed later, the transformation itself was implemented in two steps. Once the model can be read by the target analysis tool, a precise analysis method (in this case, the Markov Regenerative Process-based dependability evaluation) can be performed.

The transformations were implemented with the VIATRA2 model transformation framework. The automatic generation of a DEEM model consists of three basic steps:

1. Importing the XML files which contain the description of different aspects of the BPM model ("BPM description"), such as the basic process model and the actual values of the variables which determine the runtime behavior of the system, e.g. the probabilities of paths to be followed after decisions. This step was implemented using the built-in BPM parser component of the VIATRA2 framework, which creates an inner graph representation –in the VPML language of the tool– of the business process ("BPM Graph").
2. Transforming the graph representation of the BPM structures and concepts into a graph representation of a multiple phased system. The model transformation ("bpm2mps") itself is implemented in this step. The metamodel of a general MPS description contains the elements of a DSPN-based representation of MPS, such as the places, the transitions and the arcs of the SystemNet and the PhaseNet ("MPS Graph"). The transformation is described by precondition patterns matching to the concepts of the BPM metamodel and the corresponding postcondition patterns give the equivalent Petri Net structures, describing a Phase Mission System still in the graph representation language. As this transformation is based on a generic MPS metamodel, the analysis tool can be replaced by another Petri Net based tool without any change in this transformation.
3. Code generation: once a graph representation of the MPS is available, the text file in the DEEM format can be generated by a simple transformation ("mps2deem"). This transformation is designed to take a graph, in which the elements are stored in a tree structure and references between them describe the logical connections, and generate a text file ("DEEM model"). The main reason of the separation of model transformation and the code generation is twofold; first, this way the changes in the tool representation format (or even the replacement of the DEEM by another analysis tool) can be easily tracked and do not interfere with algorithm of the transformation of the main concepts. Therefore, the transformations are maintainable. Second, since the DEEM representation is a flat format, the whole graph tree is needed for the code generation, and therefore this step cannot be started before the entire model transformation is finished.

In order to analyze business processes with DEEM ("Analysis results"), the two transformations ("bpm2mps" and "mps2deem") were implemented in the VIATRA2 framework. At the moment there are some limitations on BPM elements due to the BPM parser of the framework, but anyway they do not affect the essence of the methodology.

6 Case Study

Consider an insurance company with a database containing client data (e.g. previous accidents) wanting to provide a premium calculator service which receives client data and returns an estimated insurance fee for the given person. Consider that this company wants to interact with other companies to complete its knowledge about a client's insurance record. This interaction is done via Web service interface; the partners provide similar premium calculator services.

The clients of such an application are employees of the company, individual brokers, other companies, registered users, etc. The company implements this functionality as a Web service to support communication between heterogeneous, loosely-coupled systems. The company wants to assure QoS parameters for the clients. Therefore, its own resources and services have to match several expectations just as the external services.

There are different types of clients with different QoS requirements against the premium calculator service. For instance, a client with a "Golden value" contract (another insurance company) may have different expectations against the system than a registered user accessing the service from a home PC.

Requests which mean a big risk (a calculation for an insurance of big amount or for a client with missing personal data) have to be checked by other partners to eliminate the chance of failure or cheating. On the other hand, different partner companies offer their calculator services (which are external activities in the process flow) for different prices.

The measures of interest are –among others– the following:

- The probability that a client request fails (for different types of clients).
- Performability metrics which show the cost of dependability, i.e. which external services to invoke at given QoS parameters and price. Requests of different client types, of course, can be forwarded to different external partners in order to assure the required QoS at a reasonable price.

Finally, sensitivity analysis is required to evaluate the effect of component failure rates on above measures.

6.1 The Example Model in BPM

This section describes the high-level BPM model of the example. The concrete modeling tool is IBM WBI Modeler which, on the one hand, supports the modeling of resources (in this case, the quantitative parameters of the services) and, on the other hand, has a BPEL export feature.

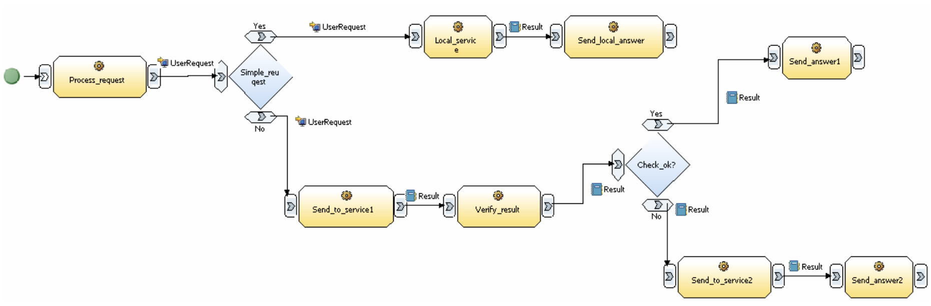


Fig. 5. The example in IBM WBI modeler

The rounded rectangles represent the internal and external activities. Parameters of the resources and the services are stored in the model repository but are not visualized. To comply with the BPEL standard, internal activities can also be accessed via a SOAP interface, but their QoS parameters depend on resources with known characteristics.

6.2 The Example Model as an MPS

This section describes the MPS model of the example business flow, showing the general method of transforming a BPM to a MPS. The system can be considered as a MPS in the following way.

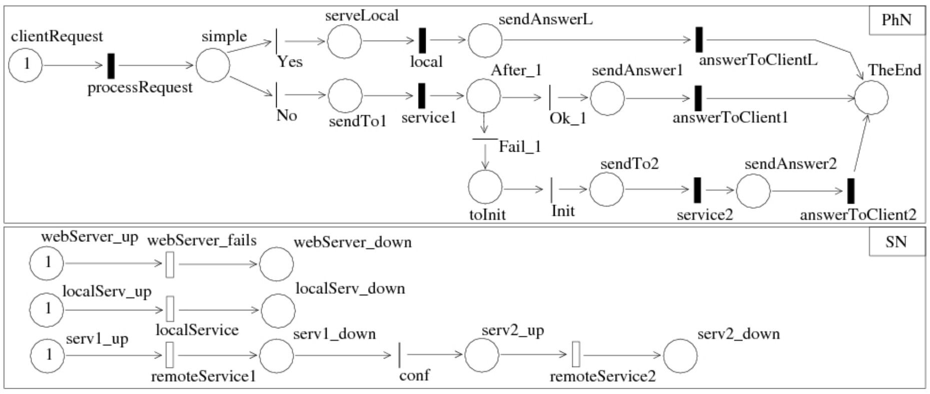


Fig. 6. The MPS model of the example

The Web server is modeled in the SystemNet (the lower partition of Fig. 6) which can be reconfigured according to that actual phase. The Web server can fail with a failure rate which is the parameter of the "webServer_fails" transition. This motivates the usage of the same topology (places and transitions) for representing all resources; the transition rates may change over time, according to the actual phase.

The first phase of the system is receiving a message from the client. This corresponds to the first activity of the business process. The second phase is upon a decision; whether the request can be served locally (in this case, no remote web service is invoked) or it needs to be sent to a remote partner. In the latter case, the answer is verified against some basic requirements, for instance, the presence and the consistency of all required data fields are checked.

If additional information is needed, the request is sent to a backup service, provided by another partner, and built upon another database. The first activity of the workflow ("Process request") is the only element of a sequence of internal service invocations. Therefore, it is the first phase of the system. The length of the phase will be represented by a timed transition (processRequest in Fig. 6) with a transition duration determined by the length of the task in the business process. During this phase, the system can fail if the internal resource, in this case the Web server, crashes.

The next phase is selected according to the user type; the simple calculations are served locally while the difficult calculations, e.g. those of users with missing data or big value of insurance, are sent to external partners. In this version, the "Recovery Block" pattern is implemented in a service oriented environment, which could be called "Recovery Block-like Service Invocations". This means the invocation of a primary service, and if the answer is not acceptable –for instance, some user record is empty– then the invocation of a backup service.

The parameters of external service invocations (modeled by phase "service1" and "service2") are determined by the SLAs. The failure rate of the services comes from the UpTimeRatio parameter from the SLA (which is represented in the BPM tool as a resource parameter of the services which represent the remote partners). The possible reconfiguration of the system, i.e. the resending of the request to the "backup service", is represented by the transition "conf" in the SystemNet.

6.3 Dependability Analysis Results

To illustrate how dependability analysis constitutes a support to the provider of the composed service, we answer the questions "What is the probability of the failure of a client request?" and "Which is the most appropriate external service provider from the set of available providers?".

In a real scenario, the parameters of the services and resources are described by Service Level Agreements. As our aim is to present a methodology for the evaluation of dependability indicators, we used some sample values based on a measurement performed against public domain web services, such as the web service interface of google [27]. Hereby we suppose ten available service alternatives for Service1, which have their parameters described in Table 1 (considering the same response time). Please remember that these do not have realistic meanings, but have been chosen just to illustrate the possibilities of such an analysis. Due to the space problems, we do not include a table with the fix parameters used for the evaluations (e.g., the costs in the reward measure, the duration of the phases, etc.).

Table 1. Parameters of services in the example

Service alternatives	1.	2.	3.	4.	5.	6.	7.	8.	9.	10.
Failure rate	0.010	0.012	0.017	0.020	0.025	0.030	0.040	0.042	0.048	0.070
Service price	14.0	13.5	13.3	13.0	12.8	12.6	12.5	12.2	12.0	11.5

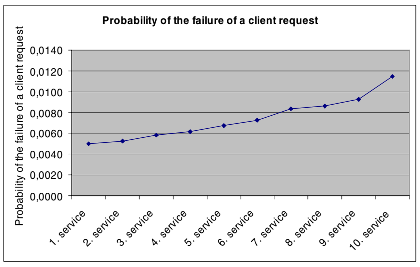
The aim of this analysis is at determining the impact of the price and dependability characteristics of an external web service on the probability of the failure of a client request and on the income of the composite service. This income is the fee that a client pays for the service minus the sum of the prices of the invoked local and external services. As the client receives a compensation for every failed request, this value has to be considered as a penalty. The measures of interest are defined in DEEM as

```
probServiceFail = IF ( MARK(webServer_down)=1 OR
MARK(localServ_down)=1 OR MARK(serv1_down)=1 OR
MARK(serv2_down)=1 ) THEN (1) ELSE (0) //failure prob

ServiceReward = [VAR(ClientFee) -
VAR(serv1Price)*FUN(serv1Succ) -
VAR(serv2Price)*FUN(serv2Succ)-
VAR(localPrice)*FUN(localServSucc)]*

(1-FUN(serviceFail)) -
FUN(serviceFail)*VAR(servicePenalty) //service reward
```

The results of the analysis is shown in Fig. 7.



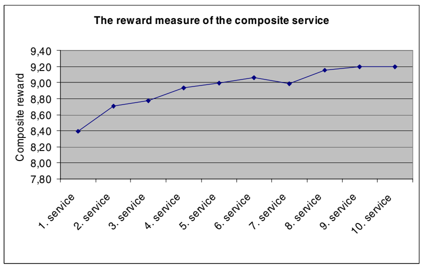


Fig. 7. Results of the dependability analysis

Based on dependability aspects only, 1.service is obviously the best choice as the number of failed requests has a minimum for this service. However, if the services are evaluated against the performability measure, then 10.service seems to be optimal as it has the highest reward value. If both aspects are considered, 9.service should be chosen instead, as it has almost as good performability measure as that of 10.service with a significantly lower probability of failure of a client request.

7 Conclusions

We presented a methodology for transforming higher level models of Service Oriented Architectures into a formal description in order to perform dependability analysis. Based on the observation that Web services-based workflows fit within the Multiple Phased Systems, model transformations were implemented in the VIATRA2 framework to perform precise mathematical analysis. Business process descriptions extended with quantitative parameters taken from SLAs were transformed, by using semi-automated transformations, into a precise mathematical model, a formal description of a Multiple Phased System, which can be solved by the dependability evaluation tool DEEM.

The current research direction is to extend the proposed methodology to analyze high level models described in BPEL and XML-based Web service description languages to provide a dependability analysis for a wider toolset. This way, a really platform (and vendor) independent analysis framework can be established. SLA-driven synthesis of web service compositions is another important research direction.

References

1. M. Martinello. Availability modeling and evaluation of web-based services –A pragmatic approach. PhD Thesis. LAAS-CNRS, 2005.
2. I. Mura, A. Bondavalli, X. Zang, and K. S. Trivedi, "Dependability modelling and evaluation of phased mission systems: a DSPN approach," in IEEE DCCA-7 - 7th IFIP Int. Conference on Dependable Computing for Critical Applications, San Jose, CA, USA, 1999, pp. 299–318.
3. I. Mura and A. Bondavalli, "Markov regenerative stochastic Petri nets to model and evaluate phased mission systems dependability," IEEE Transactions on Computers, vol. 50, no. 12, pp. 1337–1351, 2001.
4. A. Bondavalli, S. Chiaradonna, F. Di Giandomenico, and I. Mura. Dependability modeling and evaluation of ultiple-phased systems, using DEEM. IEEE Transactions on Reliability, 53(4):509-522, 2004.
5. The VIATRA2 Model Transformation Framework, Generative Model Transformer Project, The Eclipse Foundation. http://eclipse.org/gmt/
6. M. Smotherman and K. Zemoudeh, "A non-homogeneous Markov model for phased-mission reliability analysis," IEEE Transactions on Reliability, vol. 38, no. 5, pp. 585–590, 1989.
7. M. Alam and U. M. Al-Saggaf, "Quantitative reliability evaluation of repairable phased-mission systems using Markov approach," IEEE Transactions on Reliability, vol. R-35, no. 5, pp. 498–503, 1986.
8. Specification: Business Process Execution Language for Web Services Version 1.1, May 2003. http://www-128.ibm.com/developerworks/library/ws-bpel/
9. D. Menasce, V. A. F. Almeida. Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice Hall, 2001.
10. I. Majzik, A. Pataricza, and A. Bondavalli. Stochastic dependability analysis of system architecture based on UML models. In R. De Lemos, C. Gacek, and A. Romanovsky, editors, Architecting Dependable Systems, LNCS 2677, pp. 219-244. Springer-Verlag, Berlin, Heidelberg, New York, 2003.

11. Web Service Level Agreements Project. http://www.research.ibm.com/wsla/
12. Web Services Flow Language (WSFL 1.0) - Appendix C: Endpoint Property Extensibility Elements. IBM Software Group, 2001.
13. 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
14. A. Avizienis, J.C. Laprie, B. Randell, C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing IEEE Transactions on Dependable and Secure Computing, Vol.1, N.1, pp.11-33, 2004
15. A. Pataricza: From the General Resource Model to a General Fault Modeling Paradigm? Workshop on Crititcal Systems Development with UML at UML 2002, Dresden, Germany
16. WS–Reliability. OASIS Standard
17. http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf
18. A. Graziano, S. Russo, V. Vecchio, P. Foster,Metadata models for QoS-aware information management systems. In Proc. of SEKE 2002, Ischia, Italy, 2002.
19. Web Services Base Faults. Oasis.
20. http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-BaseFaults-1.2-draft-02.pdf
21. Web Service Description Language 1.1. W3C.org. http://www.w3.org/TR/wsdl
22. OMG Group, General Resource Model (GRM), URL: http://www.omg.com
23. L. Zeng, B. Benatallah, M. Dumas. Quality Driven Web Services Composition. In Proceedings of WWW2003, May 20-24, 2003, Budapest, Hungary.
24. S. Ran. A model for web services discovery with QoS. ACM SIGecom Exchanges, Vol. 4., N.1, pp. 1-10. ACM Press, 2003.
25. D. Varró, G. Varró, A. Pataricza, "Designing the Automatic Transformation of Visual Languages," Science of Computer Programming, 44:205-227 2002.
26. H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg (eds.). Handbook on Graph Grammars and Computing by Graph Transformation, vol. 2: Applications, Languages and Tools, World Scientific, 1999.
27. E. Börger and R. Stark. Abstract State Machines. A method for High-Level System Design and Analysis. Springer-Verlag, 2003.
28. IBM Corporation. WebSphere Business Integrator 5.1
29. http://www-06.ibm.com/software/integration/
30. Google Web API (beta). http://www.google.com/apis/index.html
31. J.T. Bradley, N.J. Dingle, S.T. Gilmore, W.J. Knottenbelt. Derivation of Passage-time Densities in PEPA Models using ipc: the Imperial PEPA Compiler. In Proc. of MASCOTS'03, Orlando, USA, 2003, pp. 344.351.
32. C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hoftstede WofBPEL: A Tool for Automated Analysis of BPEL Processes. In. Proc. of ICSOC 2005, Amsterdam, The Netherlands. 2005. pp. 484-489.
33. R. Kazhamiakin, P. Pandya, M. Pistore. Modelling and Analysis of Time-related Properties in Web Service Compositions. In Proc. of WESC'05, Amsterdam, The Netherlands, 2005.
34. W. M.P. van der Aalst, M. Dumas, A.H.M ter Hofstede, N. Russell, P. Wohed, H. M. W. Verbeek. Life After BPEL?. In Proc. of WS-FM, Versailles, France. 2005. pp 35-50.
35. J. Koehler, G. Tirenni, S. Kumaran, From Business Process Model to Consistent Implementation: A Case for Formal Verification Methods, EDOC, Lausanne, Switzerland, 2002, pp. 96-106.
36. R. Milner. Communicating and Mobile Systems: The Pi-Calculus. Cambridge
37. University Press, Cambridge, UK, 1999.