Journal of Software Engineering and Applications, **2011, 1, 1-8**
doi:10.4236/jsea.2011.41001 Published Online January 2011 (http://www.scirp.org/journal/jsea)

Security Engineering of SOA Applications Via Reliability Patterns

Luigi Coppolino, Luigi Romano, Valerio Vianello

Centro Direzionale di Napoli, Dipartimento per le Tecnologie, Università degli Studi di Napoli "Parthenope", Napoli, Italy
Email:{luigi.coppolino, lrom, valerio.vianello}@uniparthenope.it

Received November 30th, 2010; revised December 20th, 2010; accepted December 30th, 2010.

ABSTRACT

Providing reliable compositions of Web Services is a challenging issue since the workflow architect often has only a limited control over the reliability of the composed services. The architect can instead achieve reliability by properly planning the workflow architecture. To this end he must be able to evaluate and compare the reliability of multiple architectural solutions. In this paper we present a useful tool which allows to conduct reliability analysis on planned workflows, as well as to compare the reliability of alternative solutions in a what-if analysis. The tool is implemented as a plug-in for the widely adopted Active BPEL Designer and exploits the concept of reliability pattern to evaluate the reliability formula of the workflow. The effectiveness of the approach and the operation of the tool are demonstrated with respect to a case study of a business security infrastructure realized by orchestrating simple security services.

***Keywords*:** *Reliability of Security Services, Reliability Patterns, Workflow Systems, SOA Applications, Web Services Technology*

1. Rationale and Contribution

Web Services can be combined in complex composite services achieving new functionalities [1]. Composition may aggregate services developed and exposed within a certain organization. More interestingly, the composed Web Services can be the result of an orchestration of services exposed by different organizations. Benefits of composition have been long discussed in the past few years highlighting and demonstrating the advantages coming from the achievement of new functionality by composing autonomous services [2]. Nevertheless it is widely accepted that web applications are easy to fail as confirmed by a U.S. Government study [3] and for Web Services the situation is also worst since a number of application layers are built on top of classical web servers. As pointed out in [4], failures are inevitable in the modern Internet-Connected environments and, when dealing with composite services, assuming the failure of any individual Web Service will cause the failure of the composite service, even if all the other Web Services are reliable, one unreliable Web Service could decrease the overall reliability to a very low level. This evidence related to the reliability of the composite Web Service rises up a doubt with respect to the actual adoption of this distributed model of developing complex services [5]. Since the reliability engineer designing the workflow has no chance to modify the simple services at all, especially while dealing with services composed across organization boundaries, the only way to ensure the reliability of the composite service is increasing the reliability of the workflow by appropriately planning its architecture, *i.e.* properly adopting diversity and redundancy. This requires the development of appropriate methodologies for a quick and early evaluation of the composite service reliability and the development of tools which can be easily adopted to compare multiple architectural choices for the orchestration of a service. In this paper, we propose a formal approach that allows a workflow architect to perform reliability analysis of a SOA-based service. The approach exploits the concept of reliability patterns to derive an aggregate reliability function and it is suited for a wide class of workflow processes. The approach is implemented in a tool (more precisely, as a plug-in for Active BPEL Designer). The tool allows the system architect to evaluate the impact—in terms of reliability—of possible workflow alternatives, as early as in the first steps of the design. The effectiveness of the approach and the operation of the tool are demonstrated with respect to a case study of a business security infrastructure realized by orchestrating simple security services. The

rest of the paper is organized as follows. Section 2 provides an overview of the related work. Section 3 presents the concept of workflow patterns, while in Section 4 reliability patterns are derived and their reliability formulas evaluated. Section 5 discusses the assumptions and limitations of the model. Section 6 presents a typical case study of a business security infrastructure. In Section 7 it is described the implementation of the plug-in for ActiveBPEL and its operation is demonstrated with respect to the case study at hand. Finally, Section 8 concludes the paper with final remarks.

2. Related Work

Web Services based systems are typically composed by orchestrating a number of simpler services (generally Web Services themselves) in a common workflow. In such a case it is widely accepted that the reliability function of the workflow must be derived based on the reliability functions of individual tasks in the workflow. A mature work in this field is [6], where the authors propose a set of workflow patterns with related reliability expressions. A workflow engine named METEOR, which allows combining such patterns to build a more complex workflow, is also presented. Based on the reliability expression of the elementary patterns METEOR permits to derive the aggregate reliability expression of the composed workflow. The main limitation of this approach lies in the possibility of getting the reliability expression only for those workflows that can be obtained by composing the patterns described in [6]. To overcome such limitation, we start from results presented in [7] where the authors present, by extending results reported in [8], a set of 43 workflow patterns whose combinations can provide pretty every workflow. Starting from this set of patterns, we identify the combinations of them that are meaningful from a reliability point of view and derive for them a reliability expression. We refer such combinations as "reliability patterns". Since virtually any workflow can be obtained by combining the workflow patterns in [7], "reliability patterns" can also be applied to obtain the reliability formula for any workflow. We demonstrate our approach to the patterns defined in [8], similarly it is possible to derive new reliability patterns from the remaining patterns defined in [7]. By doing so we verify that, not only all the patterns defined in [6] are also obtained as reliability patterns, but new patterns, not considered in [6], such as the "Multi-Merge Parallel", are also identified and considered for reliability evaluation. Since our approach can be applied to retrieve the reliability expression of virtually any workflow, it can be applied to already existing workflow designing tools, instead of needing the design of new ones, as it was for METEOR. This is verified by applying the concept of reliability patterns to a popular commercial workflow designing tool, namely Active BPEL Designer [9], and enabling an early evaluation of reliability formulas for designed workflows. Even more interestingly, the proposed plug-in can be easy adapted to any WS-BPEL [10] compliant designer. Finally it is worth noting that the estimation of such formulas is not intended at exactly measuring the reliability of a composed service, but at allowing a what-if analysis of alternative architectural solutions at design time. This means that the simplicity of reliability expressions should be preferred to their precision.

3. Workflow Patterns

E. Gamma *et al.* defined a pattern as "The abstraction from a concrete form which keeps recurring in specific non arbitrary contexts" [11]. When dealing with Web Services composition it is worth considering workflow patterns that are defined in [12] as "An abstract description of a recurrent class of interactions based on activation dependencies". Workflow patterns can be considered from multiple perspectives, namely a control-flow perspective, a data perspective, a resource perspective, an operational perspective. In particular control-flow perspective refers the execution order of a set of activities. With respect to the control-flow perspective of workflow, W. M. P van der Aalst *et al.* identifies a set of twenty basic control-flow patterns (in the following referred as patterns), which can be combined to generate virtually any control flow. While analyzing the patterns provided in [8], some observations are due: 1) Since we are only interested in the reliability of the workflow from an architectural point of view, not all the patterns are relevant for our purposes. As an example, the pattern Cancel Case relates to the workflow management system and is therefore not relevant for the composition process. 2) From a reliability point of view some patterns are equivalent. As an example the Multiple Instances pattern provides the same reliability of a Parallel Split or of a Multiple Choice depending on the necessity of completing or not all the activated instances. 3) Combinations of patterns are often needed—in order to address reliability—instead of individual patterns. This is the case of the Parallel Split, for which deriving reliability requires knowing if the following task is a Synchronization or a Multi-Merge. 4) Finally, not all pattern combinations yield valid workflows, as one example the sequence of a XOR-Split and an AND-Join is not allowed since it refers to a scenario, where only one in a set of tasks is activated but the end of all of them is waited before the workflow can terminate. In the next sections, we first describe an algorithm which derives the aggregate reliability function through a workflow graph reduction, then we discuss the derived reliability patterns and their reliability formulas, finally

we present an example showing how the algorithm works.

4. Workflow Reliability Patterns

Starting from the resulting presented in [8] and considering the definitions provided in Section 3, we define the concept of reliability pattern as: "An elementary combination of patterns which itself behaves as a pattern from a reliability perspective". The previous definition yields that 1) for a reliability pattern, a reliability formula can be defined starting from the reliability formula of each activity in the pattern, 2) for any subset of patterns in the reliability pattern, a related reliability formula cannot be defined. As an example the sequence of an AND-SPLIT pattern and a MULTI-MERGE JOIN pattern matches an m-out-of-n reliability structure so it can be uniquely characterized from a reliability point of view. On the other hand it is not possible to characterize the reliability of neither the AND-SPLIT pattern nor the MULTI-MERGE JOIN pattern if they are considered separately. In the following sections, first is defined a redu- ction algorithm exploiting the concept of reliability patterns to characterize the reliability of a workflow, then reliability patterns are identified and their reliability formulas are obtained.

4.1. Reduction Algorithm

When dealing with a workflow, we are assuming that Web Services are composed in an orchestration. We assume a workflow described as

$$W = (t,a,fr,fp,fc) \quad (1)$$

Where: t is a set of tasks (each represented by a circle); a is a set of transitions (each represented by an arrow); fr is a function which associates to every task ti in t its reliability function; fp is a function which associates to every transition aij (connecting the task i to the task j) a probability pij, representing the probability that once task ti terminates task tj is activated. In other words pij represents the probability of activation of the transition aij. Every time the task ti is unambiguously identified, the index "i" will be omitted and pij substituted with pj; and fc is a function which for every task ti in t associates a value ci in [0, 1] representing the probability that a failure of task ti does not lead to a failure of the workflow. Hence ci represents a coverage factor, and can be expressed as:

$$ci = \sum_{g \in G} \phi(g) P(g) \quad (2)$$

Where:

- g is a failure mode for the task i
- G is the fault dictionary for the task i
- $\phi(g) = 1$ if the failure g can be tolerated, 0 otherwise
- P(g) is the occurrence probability of the failure g

This implies that the reliability for the single task is increased by a factor representing the probability that the component will fail without leading to a workflow failure, that is:

$$R_i = R_i' + \left((1 - R_i')\right) c_i \quad (3)$$

Where R_i' represents the reliability of the task ti and represents the reliability of the task ti as perceived by the workflow engine. The latter equals the former when c is zero, i.e. the workflow cannot tolerate a fault in one of the orchestrated services. If c equals 1 the formula returns 1 meaning that the component is optional from a reliability point of view. In the next two sub-sections we will always use the term reliability with reference to the meaning it assumes in (3). A start task and an end task must be identified into the set of the tasks. The start task does not have any incoming transition and represents the invocation of the orchestrated service by an external client. The end task does not have any outgoing transition and represents the end of the orchestration. Once the graph representing the Web Services orchestration is defined, the reduction algorithm is performed by going backward through the graph (from the end task to the start one) and each time an individual reliability pattern is found its component tasks are collapsed in a single task whose reliability is defined by the pattern reliability formula. The process is than iterated until the whole workflow is collapsed in a single task whose reliability depends on the reliability of the individual tasks, the probabilities pij and the coverage factors c_i.

4.2. Reliability Patterns

The authors described the identified reliability patterns in a previous work [13], in **Table 1** are reported the formulas of these patterns.

5. Assumptions and Limits of the Model

The main hypothesis underling the analysis proposed in the previous section, and of course the proposed approach, is the independence of events Ai = 'time to first failure of activity i' ∈ t and Aj = 'time to first failure of activity j' ∈ t , for each I ≠ j. This means, for example, that if two services are offered by the same provider it is assumed that they are deployed on physically independent servers. A further simplification in this approach lies in the absence from the model of the communication channel reliability. Actually the communication channel may itself introduce faults, as an example by dropping packets, or modifying them or just delaying their delivery beyond timeout expiration. Anyway such a kind of behavior can be embedded into the model of the single service. Finally it is worth noting that the obtained model provides the reliability of the services orchestration without considering the reliability of the service that performs the orches-

Table 1. Reliability pattern expressions.

Reliability Pattern Name	Reliability Pattern Expression
Sequence	$R = R_A x R_B$
Synchronizing Parallel	$R = R_A R_B \sum_{i_1=0}^{2} \ldots \sum_{i_N=0}^{2} u\left(\sum_{j=1}^{n} \partial(i_j - 1) - k\right) \times$ $\times \prod_{j=1}^{n} \left[\partial(i_j - 1) R_j p_j + \partial(i_j)(1 - R_j) p_j + \partial(i_j - 2)(1 - p_j)\right]$ Where u(n) = 1 if n ≥ 0, 0 otherwise ∂ (n) = 1 if n = 0, 0 otherwise And $\sum_{j=1}^{n} \partial(i_j - 1) > k \Leftrightarrow \sum_{j=1}^{n} \partial(i_j - 1) - k > 0$
Multi-merge Parallel	$R = R_A \sum_{i_1}^{2} \ldots \sum_{i_n=0}^{2} u\left(\sum_{j=1}^{n} \partial(i_j - 1) - k\right) \times$ $\times \prod_{j=1}^{n} \left[\partial(i_j - 1) R_j p_j R_B + \partial(i_j)(1 - R_j) p_j + \partial(i_j - 2)(1 - p_j)\right]$
XOR Parallel	$R = R_A R_B \sum_{i=1}^{n} p_{Ai} R_i$ Where … $\sum_i p_{Ai} = 1$
Loop	$R_l = \frac{R(1-p)}{1-pR}$ Where p is the probability of running the loop.
Discriminator Parallel	$R = R_A R_B \sum_{i=1}^{n} p_{iB} R_i$ Where $\sum_{i=1}^{n} p_i = 1$

tration. As an example let us consider a service which by means of an orchestration engine (e.g. BPEL) coordinate the invocation of other services by fo- llowing a predefined workflow. In this case the reliability of the orchestration service, of the server hosting such a service and of the orchestration engine, should be modeled and in case of hypothesis of independence it should be multiplied by the reliability of the entire workflow.

6. Case Study

This section considers a realistic case study to show how the proposed approach can be applied in the field of reliable workflow development. The case study considers a company with four branches each with its IT department [14]. The four branches are federated in a trusted domain [15]. So that a user which logs at one of the federated entities, in order to access a service, obtains a SAML [16] token which can be used at a later time to log in at any of the other federated entities without the need of being authenticated again. In the presented scenario the user, holding the SAML token, sends a request to access a service provided by the company (**Figure 1**). The request is intercepted by the Access Manager of one federated entity which picks the SAML token up and tries to validate it. Two alternatives are possible:

1) The SAML token was actually released by that entity (branch 1 of the company): in this case, the Access Manager requests the Identity Manager to retrieve the appropriate authorization profile for the user holding the token. The Identity Manager will do it by means of the Directory Server which provides an abstraction of the data repositories in the company.

2) The SAML token is not recognized as a valid token by the branch 1 of the company: the Access Manager charges the Federation Manager in its domain with managing the SAML token. The Federation Manager will ask other Federation Managers in the same trusted domain to check for the SAML token. Each of the Federation Manager will provide its Access Manager with a copy of the token. These in turn will repeat the same steps of the validation procedure as operated by the Access Manager in the branch 1. Finally the required service will be accessed with the authorization profile provided by the federated entity which actually issued the SAML token.

Assuming, that each of such functionalities is provided as a service, the whole procedure can be described by the workflow graph of **Figure 2**. A workflow architect can study the workflow in order to make a reliability prediction of the orchestrated service. Further the designer can study the workflow even to modify the architecture of the service itself; if, for example, the four federated entities are distributed in Europe but two of them are both in Italy, the workflow architect could compare the reliability of an architecture where the four entities are seen as four branches of the company, with an architecture where the two entities in Italy are connected through a virtual dedicated LAN resulting in a company with only three branches for a reliability point of view. The workflow architect can take the best decision based on a trade-off analysis in terms of total reliability of the service versus implementation cost for the chosen solution. While, to evaluate the cost of an architectural solution could be a simple task, to compute the total reliability function of a complex workflow is not straightforward. In the next section we present a useful tool that can help the workflow architect performing a reliability prediction analysis.

7. Implementation

In this section we first show the main capability of the proposed reliability prediction tool and then we have demonstrated the usage of the tool with respect to the case study presented in the previous section.

7.1. The Plug-in

The proposed algorithm was developed as a plug-in for ActiveBPEL Designer in [9] which is a widely used workflow designing tool. Once installed the plugin allows the workflow designer to perform a reliability analysis for a BPEL Web Services orchestration. More precisely

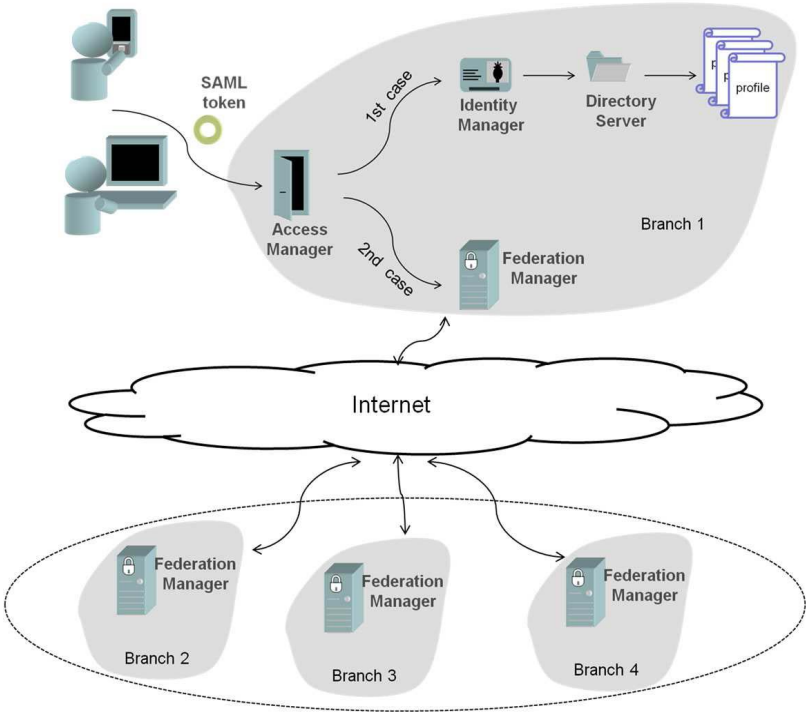


Figure 1. The case study scenario: A company with four branches federated in a trusted domain.

it allows to:

- Retrieve the reliability function for the workflow under design;
- Evaluate the reliability of the workflow at a specific point in time, that is the probability that the workflow will not fail until the specified time;
- Plot the resulting reliability function with respect to the time;
- Obtain usual reliability metrics, such as the MTTF (Mean Time To Failure), for the analyzed workflow.

In order to perform the above described analysis the designer has to provide the workflow with the information required by the model (such as the symbolic expression of the reliability function of each activity, and the transition probabilities). Such information is directly embedded in the BPEL description of the workflow by exploiting the standard WS-BPEL [10] extensibility. This allows the tool to remain compliant with any WS-BPEL orchestration despite the specific editor adopted for its design. Moreover the plug-in extends the Active BPEL Designer interface to simplify the provisioning of reliability related data. Then, in order to compute the total workflow reliability function, the plug-in uses an XSLT style-sheet to convert the workflow BPEL/XML based representation to an internal representation that only includes the workflow dependability attributes and the recognized reliability patterns. After this transformation has been done, the plug-in calls a class that calculates the global reliability function as described in the reduction algorithm. The desired analysis estimates are then obtained by evaluating the retrieved reliability function. Symbolic operations are made possible by the adoption of the MathEclipse plug-in [17].

7.2. Experiment

With respect to the case study workflow depicted in the previous section in **Figure 2**, **Figure 3** shows the steps made by the reduction algorithm in order to obtain the workflow reliability function. Each step is represented by a numbered box in which the patterns recognized by the algorithm are highlighted. In the first and third steps sequence patterns are recognized and reduced (dotted boxes). In the second and fifth steps XOR parallel patterns are matched and reduced (dashed boxes). In the fourth step an AND-SPLIT configuration is found (dot-dashed boxes). Even though pencil and paper calculation is possible, this is for sure an error prone process, as well as a time consuming one, since the most complex is the workflow graph the toughest is to evaluate the reliability function by hand. As an alternative the reliability function for the workflow can be automatically evaluated by using the proposed plug-in. **Figure 4** shows the workflow as it could be implemented with the Active BPEL Designer tool. To make possible the evaluation of the reliability estimates it was required specifying the reliability function of each activity as well as the transi-

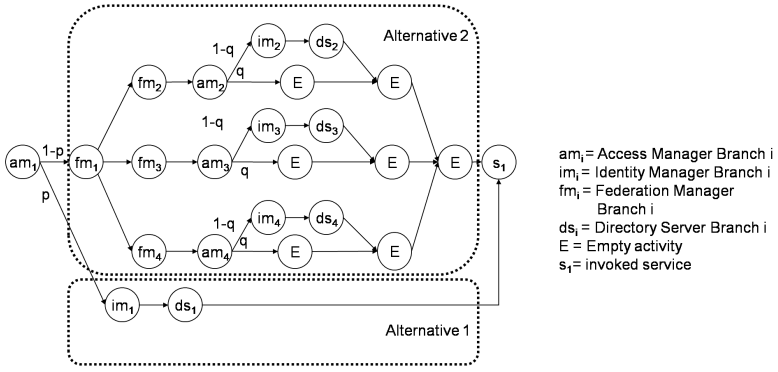


Figure 2. The workflow graph of the case study scenario.

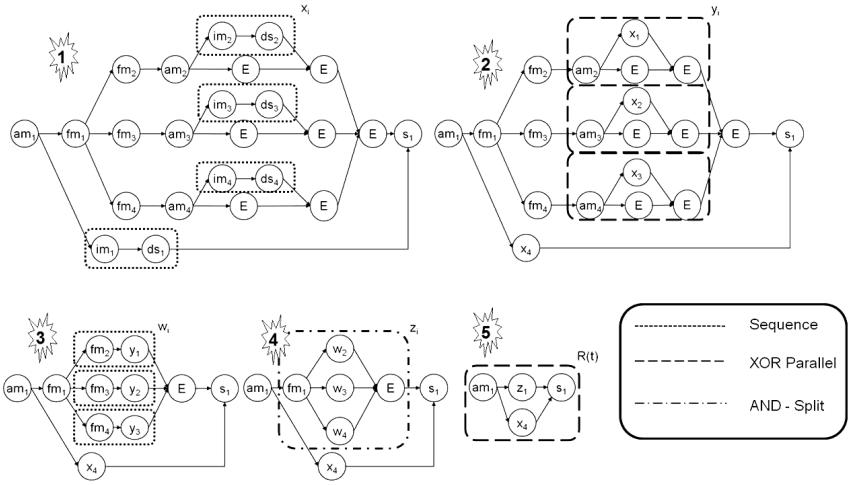


Figure 3. The workflow of the case study scenario and the related reduction process.

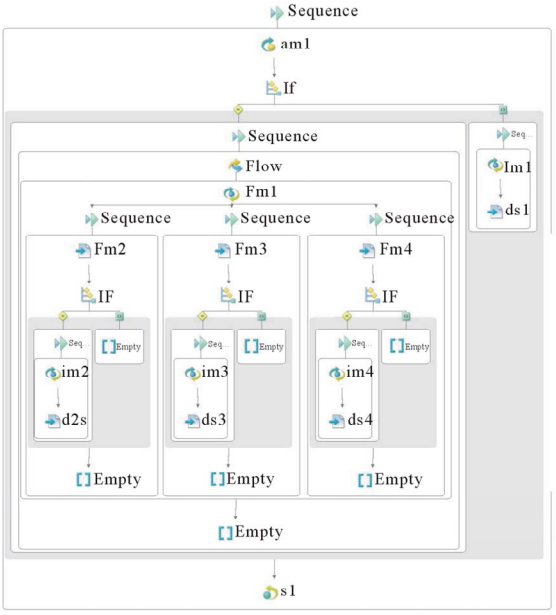
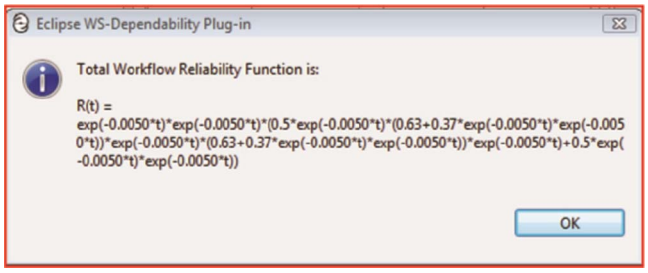
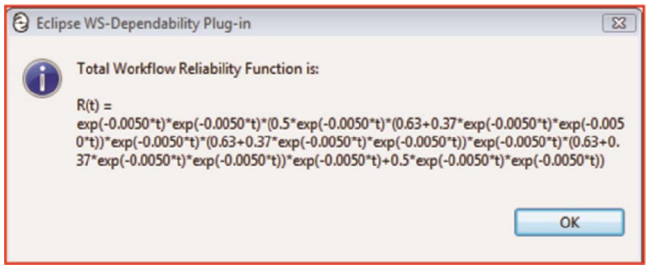


Figure 4. The schematic of the case study workflow in Active BPEL Designer.



(a) Company with four federated entities



(b) Company with three federated entities

Figure 5. Total workflow reliability function.

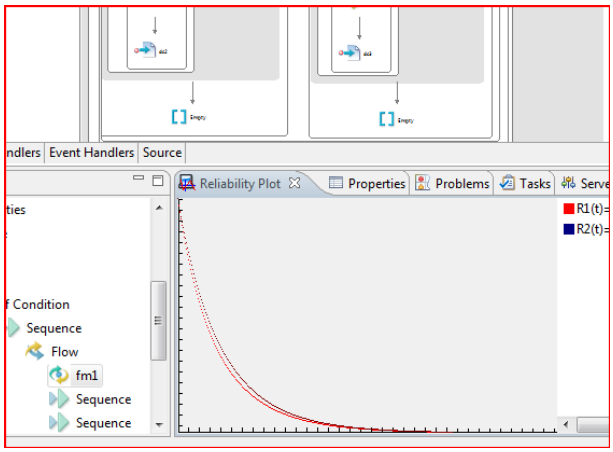


Figure 6. A screenshot of the ActiveBPEL workspace showing the plot reliability function.

tion probabilities. In our case study, for the sake of simplicity, we assumed all the non-empty activities to have reliability functions exponentially distributed with the same failure rate value. We explicitly note that evaluating the failure rate of Web Services is beyond the scope of this paper, we instead use a realistic value of 0.005 failures/second as resulting from the experiments conducted in [18]. The transition probabilities were assumed to be p = 50% and q = 0.37%, and the value of k for the Synchronizing Parallel pattern is always equals to the number of branches of the pattern. Once fixed those values the tool can infer the desired reliability evaluations. For example using the plug-in we can easily compare the reliability of the two architectural solutions presented in the previous section. **Figure 5** depicts the final reliability expressions in the two cases and **Figure 6** illustrates a screenshot which shows the workspace with the charts of R1(t) (4 branches) and R2(t) (3 branches), obtained running the plot reliability function command.

8. Conclusions

In this paper we have proposed a formal approach that allows a workflow architect to perform reliability analysis of a SOA-based service. The approach exploits the concept of reliability patterns to evaluate the reliability function of a wide class of workflows. We have integrated our reduction process into an orchestration engine so to provide a useful plug-in, which can be used to perform a reliability analysis for a planned workflow, as well as to compare the reliability of alternative solutions in a what-if analysis.

9. Acknowledgements

This work has been partially supported by the Italian Ministry for Education, University, and Research (MIUR) in the framework of the Project of National Research Interest (PRIN) DOTS-LCCI: Dependable Off-The-Shelf based middleware systems for Large-scale Complex Critical Infrastructures, by the Italian Ministry of Industry in the framework of funding INDUSTRIA 2015 (SISTEMA project). This work received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement No. 225553 (INSPIRE Project).

REFERENCES

[1] L. Cherbakov, G. Galambos, R. Harishankar, S. Kalyana, and G. Rackham, "Impact of Service Orientation at the Business Level," *IBM Systems Journal*, Vol. 44, No. 4, 2005, pp. 653-668. doi:10.1147/sj.444.0653

[2] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, May 2004, pp. 311-327. doi: 10.1109/TSE.2004.11

[3] BIGSF, "Government Web Application Integrity," The Business Internet Group, San Francisco, 2003.

[4] S. Bhiri, O. Perrin and C. Godart, "Ensuring Required Failure Atomicity of Composite Web Services," *Proceedings of the International World Wide Web Conference* (*WWW'05)*, Chiba, 2005, pp. 138-147.

[5] "Will Reliability Kill the Web Service Composition?" 2010. http://www.cs.rutgers.edu/rmartin/teaching/spring04/cs553/papers/009.pdf

[6] J. Cardoso, A. Sheth, J. Miller, J. Arnold and K. Kochut, "Quality of Service for Workflows and Web Service Processes," *Web Semantics*: *Science, Services and Agents on the World Wide Web*, Vol. 1, No. 3, April 2004, pp. 281-308. doi:10.1016/j.websem.2004.03.001

[7] N. Russell, A. H. ter Hofstede, W. M. van der Aalst and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View," BPM Center Report BPM-06-22, BPMcenter.org, 2006.

[8] W. M. P van der Aalst, A. H. M. ter Hofstede, B. Kie-Puszewski and A. P. Barros, "Workflow Patterns," *Distributed and Parallel Database*, Vol. 14, No. 3, 2003, pp. 5-51.

[9] Y. Qian, Y. Xu, Z. Wang, G. Pu, H. Zhu and C. Cai, "Tool Support for BPEL Verification in ActiveBPEL Engine," *Proceeding of 18th Australian Software Engineering Conference* (*ASWEC* 2007), Australia, 2007, pp. 90-100. doi:10.1109/ASWEC.2007.50

[10] "WS-BPEL Specification," http://www.oasis-open.org/committees/tchome.php?wgabbrev=sbpel(lastaccessed30/11/2010)OASISSecurityAssertionMarkupLanguage(SAML), http://www.oasis−open.org/committees/tchome.php?wgabbrev =security(last accessed30/11/2010)

[11] E. Gamma, R. Helm, R. Johnson and J. Vlissides, "Design Pattern—Elements of Reusable Object-Oriented Software," Addison-Wesley, Menlo Park, California 1995.

[12] S. Bhiri, O. Perrin and C. Godart, "Extending Workflow Patterns with Transactional Dependencies to Define Reliable Composite Web Services," *Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services*, Guadeloupe, 2006.

[13] L. Coppolino, L. Romano, N. Mazzocca and S. Salvi, "Web Services Workflow Reliability Estimation Through Reliability Patterns," *Proceedings of the* 3*rd International Conference on Security and Privacy in Communications Networks and the Workshops,* (*SecureComm*'07), 2007, pp.107-115.

[14] F. Campanile, L. Coppolino, S. Giordano and L. Romano, "A Business Process Monitor for a Mobile Phone Recharging System," *Journal of System Architecture*, Vol. 54, No. 9, 2008, pp. 843-848. doi:10.1016/j.sysarc.2008.02.005

[15] "The WS-Federation Specification," 2010, http://msdn2.microsoft.com/enus/library/ms951236.aspx

[16] "OASIS Security Assertion Markup Language (SAML)," http://www.oasis-open.org/committees/tc_home.php?wg_abbrev = security (last accessed 30/11/2010)

[17] "MathEclipse Project," 2010, http://sourceforge.net/projects/matheclipse/

[18] P. Chan, M. Lyu and M. Malek, "Making Services Fault Tolerant," *Lecture Notes in Computer Science*, Vol. 4328, pp. 43-61, 2006. doi:10.1007/11955498_4

