Adaptation of Service-Based Applications Based on Process Quality Factor Analysis

Raman Kazhamiakin[1], Branimir Wetzstein[2], Dimka Karastoyanova[2], Marco Pistore[1], and Frank Leymann[2]

[1] FBK-Irst, via Sommarive 18, 38100 Trento, Italy
{raman,pistore}@fbk.eu
[2] Institute of Architecture of Application Systems, University of Stuttgart, Germany
{karastoyanova,leymann,wetzstein}@iaas.uni-stuttgart.de

Abstract. When service-based applications implement business processes, it is important to monitor their performance in terms of Key Performance Indicators (KPIs). If monitoring results show that the KPIs do not reach target values, the influential factors have to be analyzed and corresponding adaptation actions have to be taken. In this paper we present a novel adaptation approach for service-based applications (SBAs) based on a process quality factor analysis. This approach uses decision trees for showing the dependencies of KPIs on process quality factors from different functional levels of an SBA. We extend the monitoring and analysis approach and show how the analysis results may be used to come up with an adaptation strategy leading to an SBA that satisfies KPI values.

1 Introduction

In recent years extensive attention has been paid to devising and improving the concepts and infrastructures for service-based applications (SBAs) [1]. SBAs can be viewed in terms of three functional layers, namely (i) business processes, (ii) service compositions that implement these business processes, and (iii) services and service infrastructure. A major concern for enterprises is to ensure the quality of their SBA-based business processes. Thereby, process quality goals are specified in terms of Key Performance Indicators (e.g., order fulfillment time), i.e. key process metrics that contain target values which are to be achieved in a certain period. KPIs of business processes that are implemented in terms of SBAs are typically monitored using business activity monitoring technology. If monitoring results show that KPIs do not meet target values, further *process quality factor analysis* is needed to find out which of the lower level process metrics (e.g., duration of process activities, type and amount of ordered products etc.) or QoS metrics (e.g., availability of IT infrastructure) mostly influence KPI target violations.

After influential factors of KPI violations are identified, the goal is to perform *process adaptation* in order to prevent KPI violations for future process instances or even for the running instances. Thereby, several challenges arise. Firstly, one has to choose appropriate adaptation actions for each influential factor identified (e.g., selection of a faster delivery service in order to decrease deliver time). Secondly, one has to take into account that an adaptation action can have positive effect on one metric

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 395–404, 2010.

but negative effect on others (e.g., a faster delivery time normally involves higher costs). Thus, when adapting the process one has to choose a set of adaptation actions which improve the influential factors as shown in the analysis and take into account their effects.

In order to deal with this adaptation problem, we extend previous work described in [2] which uses decision trees for process quality factor analysis. Based on this work, in this paper, we show how to extract a set of adaptation requirements from the decision tree and find an adaptation strategy consisting of a set of adaptation actions which takes into account both positive and negative effects of adaptation actions on metrics. We discuss limitations of our approach so far and show several possibilities for extending the approach in future work.

2 Background and Motivation

In this section we present the motivation for our approach and a scenario which we use in the following sections for explaining our concepts based on examples. This scenario has already been used in [2] for experimental evaluation of process quality factor analysis. The scenario consists of a purchase order process implemented by a reseller which offers products to its customers and interacts with its suppliers, a banking service, and a shipment service for processing the order. The customer sends a purchase order request with details about the required products and needed amounts to the reseller. The latter checks whether all products are available in the warehouse. If some products are not in stock, they are ordered from suppliers. When all products are in place, the warehouse packages the products and hands them over to the shipment service, which delivers the order to the customer, and finally notifies the reseller about the shipment. For measuring the performance of its business process, the reseller defines a set of Key Performance Indicators (KPIs). A typical KPI for the reseller in our scenario is order fulfillment lead time [3], which measures the number of days from order receipt to the delivery of the ordered products at the customer. A KPI is a key metric (with either technical or business meaning) with target values which are to be achieved in a certain analysis period (e.g., order fulfillment lead time < 5 days). After specifying a set of KPIs with target values, they have to be measured based on executed process instances. If the measurement shows an unsatisfying result, i.e. the KPI targets are violated, the reseller wants to improve its process, for instance, by using process adaptation.

Due to the fact that KPIs are complex characteristics that rely upon a wide range of factors originating from different functional levels, adaptation of underlying SBAs is not a straightforward approach. In our scenario the KPI may be influenced by many factors (which have to be measured both on process level (process performance metrics) and service infrastructure level (QoS metrics): duration of sub-processes and activities, response time and availability of used services, ordered products and their properties such as number of ordered items, product type and size, cost of delivery service, availability of IT infrastructure etc. All those factors and a combination of those can lead to late delivery. Thus, the first step needed is to perform a process quality factor analysis and find out the influential factors for KPI violations.

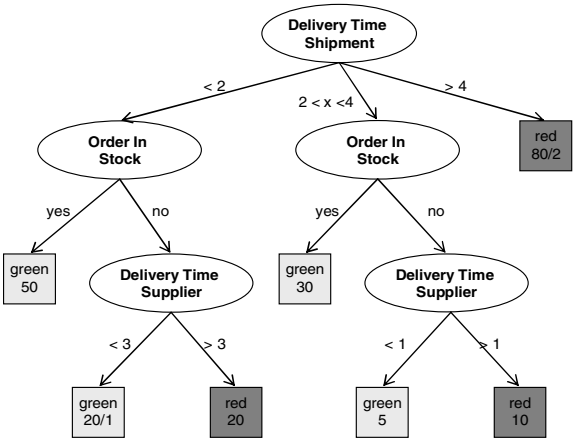

Fig. 1. An Example Dependency Tree

Our approach to quality factor analysis is based on machine learning techniques, more specifically decision tree algorithms [4]. The approach, its assumptions and reasons for using machine learning techniques, have been described in detail in [2]. In the following we will give only a general overview based on an example. The result of this analysis is a decision tree as shown in Fig. 1, called a *dependency tree* as it shows the main quality factors the KPI *depends* on. The tree is generated automatically for a KPI selected by the user. The leaves of the tree show the classification of the KPI, i.e. whether it is satisfied ("green") or violated ("red") in relation to its target values, and the number of process instances which led to this path. Note that the classification (satisfied or violated) could be extended towards more than two nominal values or even numerical value ranges (regression trees); this is part of our future work. The other nodes of the tree are the main influential factors (metrics) and the branches contain conditions on those metrics.

In order to improve those factors, different adaptation actions may be considered, for example, replacing a service either dynamically or using a predefined set of services; renegotiating the Service Level Agreements (SLAs) with the corresponding service provider; outsourcing a subprocess or replacing it with a service from an external provider. On infrastructure level, possible adaptations are replacement of IT components with faster ones, clustering for improving availability, upgrading hardware components etc. For a particular situation, different adaptation actions and their combinations may be necessary for improving the same KPI; consistency and non-contradicting actions with respect to the KPI (and perhaps other KPIs) needs to be ensured. This is because an adaptation action can positively affect one influential factor but negatively others. Assume, for example, the selection of a new better performing service which leads to a better response time but negatively affects the cost metric. We call the collection of the adaptation actions that, being enacted in combination, achieve the desired outcome an adaptation strategy.

3 Overview of the Approach

In this work we present an approach that allows for adapting service-based applications in order to prevent the violation of KPIs. The overall process is represented in Fig. 2. This approach consists of the following four phases:

- *Quality modeling for analysis and adaptation.* At design time the metrics model and the adaptation actions model are created. In the metrics model, the user specifies the application KPIs, and the quality metrics representing the potential influential factors of KPIs. In the adaptation actions model, the user specifies the available adaptation actions per metric and the effect of those actions on application metrics specified in the metrics model. In particular, this model allows for defining whether an action contributes positively or negatively to a certain quality factor, i.e., whether it improves the value of a metric.

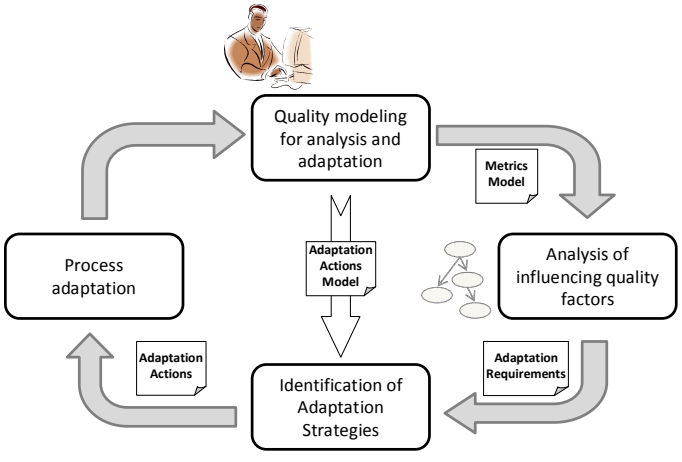

Fig. 2. Quality factor analysis and adaptation

- *Analysis of influential quality factors.* In the second phase, based on the metrics model, the monitoring of KPIs and potential influential quality metrics is performed across the instances of the application; the information is continuously aggregated and updated. Then the metrics related to previous executions of a given application are analyzed in order to identify the reasons, i.e., the influential factors, which lead to the undesired values of the specified KPIs. A dependency tree is generated. In the next step, one identifies those tree paths of application metrics (and their value ranges) that correspond to the "bad" values of the KPI and thus fail the underlying business goal. The result of the analysis characterizes thus those factors of the application that should be improved, i.e., that are subject of adaptation, and how they should be improved (their values) (Section 4.2).
- *Identification and selection of an adaptation strategy.* In the next phase the approach aims to combine, and enact concrete adaptation actions that address the identified requirements as part of a coherent adaptation strategy (Section 4.3). This phase relies on the adaptation action model, where the effect of those

actions on different application metrics is described. It takes into account that an adaptation action contributes positively or negatively to a certain quality factor, i.e., whether it improves the value of a metric. After identification of a set of alternative adaptation strategies, one strategy is selected based on certain criteria.

- *Process adaptation.* The selected adaptation strategy is used for adaptation of the process model or process instance by executing all contained adaptation actions. After adaptation, the existing KPIs and metric definitions might have to be adapted thus closing the cycle.

4 Identification of Adaptation Requirements and Strategies

Once the influential factors are highlighted, it is necessary to identify those elements of the application that should be improved, i.e., to identify *adaptation requirements*. Based on those requirements and on the model of adaptation actions associated to the quality properties of the application, possible *adaptation strategies* are identified and triggered.

4.1 Model of Adaptation Actions

In order to adapt different elements of the application at all the levels of the application stack, i.e., to enable a holistic adaptation strategy, it is necessary to provide a generalized model of possible adaptation actions. This model should relate different adaptation mechanisms to the quality properties of the application, which are subject of adaptation. In other words, the model should characterize the available adaptation actions to the model of metrics described above. More precisely, the definition of adaptation actions in our approach consists of the following elements:

- *Adaptation mechanism*. This part of the definition characterizes the machinery or a technique used to realize the specified adaptation action. For example, the adaptation action "substitute a service with another service" may be realized by a composed mechanism, which consists of service discovery and binding [5]. Other actions may include (but are not limited to): re-negotiation of quality parameters of the services used in the service composition, re-composition of the underlying service composition or a part of the process, replacement of a subprocess with another subprocess or with a single service (process outsourcing, [6]), or infrastructural reconfiguration.
- *Adaptation effect. T*o relate the adaptation action to the system we characterize the former in terms of the effects the action causes on one or another application quality metric. We say that the action has a *positive effect* on the metric if the value of the latter is improved as a re*sult of the appli*cation of the action. We say that the action has a *negative effect* on the metric if the value of the latter is worsened as a result of the application of the action. Otherwise, we say that the *action does no*t affect the metric.

We remark that for the presented approach we abstract away the details on how the adaptation actions are re*alized and enac*ted. The core part of the model of the adaptation action is how the action relates to the application metrics.

An adaptation action *a is defined as a pair* $\langle M^+, M^- \rangle$ *where*

- $M^+ \subseteq M$ is a set of metrics, on which the action has a direct positive effect;
- $M^- \subseteq M$ is a set of metrics, on which the action has a direct negative effect.

4.2 Identification of Adaptation Requirements

After the dependency tree is generated (as discussed in Section 2), the next step is to identify the adaptation requirements for the application in order to improve the performance of the application. This activity relies on the analysis of the dependency trees of the relevant KPIs of the application. Intuitively, this analysis may be described as follows.

- As a first step it is necessary to understand, which of the violations of the KPI (i.e., which of the "red" blocks) should be prevented. For example, it may be the case that all the possible violations should be avoided. In this case it is necessary to find an adaptation strategy (consisting of possibly several adaptation actions) preventing all of those situations. It is also possible to prevent violations only in selected situations, e.g., the most frequent ones. In this case, the other violation cases are ignored and excluded from the tree. We remark that this decision may be done by the business analysts or even automatically, based on some predefined criteria (e.g., for the cases where number of violations exceeds 10%).
- Second, it is necessary to associate the violations with the influential factors that might help avoiding the violations. This is done by identifying all the metrics in the nodes (and their sub-ranges) on the path from the "red" node to the root of the tree. If some of these metrics is improved such that there are no violations, then the adaptation will be successful. In the above dependency tree in order to improve for the central "red" node, it is enough to improve the metric "Delivery Time Supplier" to the value of < 1. On the other hand, in order to improve for the rightmost "red" node it is not enough to improve the value of the "Delivery Time Shipper" to the value of < 4. Indeed, the other violations are still possible and the other metrics should be improved as well. If a running instance is to be adapted and some of the metric values are already known for this instance, then obviously some paths of the tree leading to "red" nodes might be irrelevant for that instance and can be excluded from further consideration.
- The final step takes into account the need to consider all the selected "red" nodes together in order to merge the appropriate actions into a complete set of adaptation requirements.

In order to realize this approach, we rely on the algorithms provide by the decision procedure for the Satisfiability Modulo Theories (SMT [7]). In this problem, the goal is to find solutions for a set of logical constraints (formulas) with respect to combinations of background theories, such as the theory of real or integer numbers, Boolean arithmetic, and even complex data structures. Below we show how the problem of finding adaptation requirements may be expressed in terms of SMT problem, and how the adaptation requirements may be then extracted.

Our goal is to avoid all the paths in the tree that lead to the "red" leaf nodes. That is, the combination of the "metric-range" pairs on the path should not occur. This

combination may be represented as a conjunction of expressions over those metrics, i.e., for a path with n nodes we built an expression $(\mu_1,r_1) \wedge \ldots \wedge (\mu_n,r_n)$, where (μ_i,r_i) represents an expression over the i^{th} metric on the path. For instance, for the central "red" path in the example tree the expression would be as follows:

(2<"Del. Time Shipment"<4) ∧ ("Order in Stock"=yes) ∧ ("Del.Time Supplier">1)

If in case of running instance adaptation, some of these metrics values are known, then the expression can be simplified. If, e.g., "Order in Stock" is false, then the expression is already known to be false and thus this "red" path is already avoided for this instance. If "Order in Stock" is true, then it can simply be removed from the expression, thus simplifying later analysis.

In order to avoid those paths, our goal is to make all those expressions false, i.e., for m paths, we have to find possible assignments of metric values such that the following formula becomes true[1]:

$$((\mu_{11},\neg r_{11}) \vee \ldots \vee (\mu_{n1},\neg r_{n1})) \wedge \ldots \wedge ((\mu_{1m},\neg r_{1m}) \vee \ldots \vee (\mu_{nm},\neg r_{nm}))$$

It is easy to see that if the formula is satisfied, then neither "red" node is reachable. The result of the analysis is represented as a set of alternatives, each of which contains the list of metrics that should be adapted and their expected ranges. In order to carry out the analysis task we use the MathSAT tool [8], which implements the SMT decision procedure.

More precisely, we define the resulting adaptation requirements as $R = \{A_1,\ldots,A_n\}$, where $A_i = \{ (\mu_{1i},r_{1i}),\ldots,(\mu_{mi},r_{mi})\}$ is a set of metric-range pairs that should be achieved in order to address the adaptation needs.

We present the approach using the dependency tree depicted in Fig. 1 (we assume that the goal in the example is to avoid any of possible violations). For the metrics "Delivery Time Shipment" (Sh), "Delivery Time Supplier" (Su), and "Order in Stock" (O), and for the three paths to "red" nodes we construct the following three constraints:

- Sh < 2 ∧ O=no ∧ Su > 3; Sh > 2 ∧ Sh < 4 ∧ O=no ∧ Su > 1; Sh > 4

Based on these clauses, we need to satisfy the following formula:

- (Sh > 2 ∨ O=yes ∨ Su < 3) ∧ (Sh < 2 ∨ Sh > 4 ∨ O=yes ∨ Su < 1) ∧ (Sh < 4).

The result of the analysis provided by the tool represents the following alternatives:

- (2 < Sh < 4) and (Su < 1), (Sh < 2) and (Su < 3), and (O=yes) and (Sh < 4)

That is, to avoid violations of the KPI it is necessary to improve the metric "Delivery Time Supplier" to the value < 1 and the metric "Delivery Time Shipment" to the value from 2 to 4, or alternatively improve the metric "Delivery Time Supplier" to the value < 3 and the metric "Delivery Time Shipment" to the value < 2, or "Order in Stock" to become true and the "Delivery Time Shipment" to the value in range < 4. Note that as the "Order in Stock" metric is not adaptable, the third alternative is not relevant for adaptation of process models (future process instances); however it could be used for adaptation of running process instances where O=yes.

[1] Here the notation $\neg r_{ij}$ stands for complement of the specified range.

4.3 Identification of Adaptation Strategies

After the adaptation requirements are identified, the next step is to associate possible adaptation strategies which should lead to KPI fulfillment, i.e. the sets of adaptation actions that adapt the corresponding influential factors. As described in Section 4.1, for adaptable metrics a set of possible adaptation actions has been specified. The first step is thus to come up with alternative adaptation strategies, and in a second step to select one of those strategies in an optimal way.

```
1   let S = ∅ // set of resulting strategies
2   for each A_i ∈ R
3    S = S ∪ strategies(A_i)

4   function strategies(A)
5    let S_A = {∅} // set of strategies for A, initially contains an empty set
6    for each (μ,r) ∈ A
7     let S' = S_A // temporary set of partial strategies built on previous steps
8     S_A = ∅
9     let act={a | μ∈M⁺(a)∧ forall(μ',r')∈A,¬(μ'∈ M⁻(a))}
10    if act=∅ return ∅ // the whole alternative cannot be achieved
11    for each a ∈ act // built a Cartesian product of actions
12     for each s ∈ S'
13      S_A = S_A ∪ {s ∪ {a}}
14    return S_A
```

Fig. 3. Strategy selection algorithm

The algorithm for identifying adaptation strategies is represented in Fig. 3. The set of strategies contains the strategies for all the alternatives. For every alternative the following procedure is applied (lines 4-10). For each of the metric to be adapted, we select the set of actions that improve it without negatively affecting other metrics to be adapted (line 9). If this set is empty for some metric, the alternative cannot be satisfied and an empty result is returned. Otherwise, a Cartesian product of those actions with the actions for other metrics is created (lines 11-13). The resulting set of strategies is returned. For the sake of simplicity we omit here formal proofs of the algorithm correctness.

It is easy to see that any of the strategies extracted in this way will satisfy the identified adaptation requirements. However, the effect of different adaptation strategies on the SBA is not the same. This is because adaptation strategies depending on contained adaptation actions differ in their negative effects on certain applications metrics.

To order the strategies we adopt a heuristic, in which the strategy with less negative effects is more preferable. All the strategies are then ordered according to this number: the lower this number is the more the adaptation strategy is preferred. Note that even if the two actions in the strategy negatively affect the same metric, the effect is counted twice as it may have stronger impact. However, other approaches and heuristics for the selection of an optimal adaptation strategy may be thought of. Some of them are discussed in the conclusions section.

5 Related Work

The field of QoS-aware adaptable SBAs has only recently been given attention, which is also reflected in the scarce amount of related literature. There are no approaches, to the best of our knowledge, that enable adaptation of SBAs based on quality characteristics yet in an integrated manner across all layers, based on monitoring and analysis of KPIs and the corresponding influential factors.

There are several existing works in the context of QoS-aware service compositions [9, 11] which describe how to create service compositions which conform to global and local QoS constraints taking into account process structure when aggregating QoS values of atomic services. These approaches can be used for QoS-based adaptation by replanning the service composition during monitoring [12]. Our approach is different in that we not only take into account QoS but also quality characteristics from other SBA layers and perform analysis based on their dependencies. We do not (yet) exploit information on process structure during dependency analysis, as the approach described in [10], but use decision tree algorithms instead.

Closely related to our approach is iBOM [13] which is a platform for monitoring and analysis of business processes based on machine learning techniques. It focuses on similar analysis mechanisms as in our approach, but does not deal with adaptation of SBAs by extracting adaptation requirements from the decision trees and automatically deriving adaptation strategies, but uses simulation and what-if analysis techniques instead.

Work on service composition adaptation is available and the existing approaches that do not focus on QoS-awareness of SBAs have been classified. The available approaches are mechanisms for service composition adaptation can similarly be borrowed in the approach presented in this paper as adaptation mechanisms on the service composition level [5, 14].

6 Conclusions and Future Work

In this paper we have presented a novel adaptation approach for SBAs based on quality factor analysis. We have extended previous work on quality factor analysis by showing how the resulting dependency tree can be used for adaptation purposes. In particular we have shown how to model adaptation actions and associate them with quality metrics, how to extract adaptation requirements from the dependency tree and come up with an adaptation strategy.

Our future work involves extending the approach in several ways. First, it is possible to define global SBA constraints as a metric that should not be negatively affected by any of the adaptation actions. If some action may violate such a constraint, it should be excluded. Second, if it is possible to capture the effect of the adaptation action onto the metric with a higher precision (e.g., instead of simple positive/negative contribution give a numerical value or even specify the effect of the action on the metric value), then the analysis should give precedence to the actions with better effect. Finally, it is possible also to exploit the relation between the metric and the number of KPI violations. This would allow also for ordering the requirements: the more violations are associated with the metric value, the more important it is. The adaptation actions, therefore, should be selected accordingly.

Acknowledgements. The research leading to these results has received funding from the European Community's 7th Framework Programme under the Network of Excellence S-Cube Grant Agreement no. 215483.

References

1. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer 11 (2007)
2. Wetzstein, B., Leitner, P., Rosenberg, F., Brandic, I., Dustdar, S., Leymann, F.: Monitoring and Analyzing Influential Factors of Business Process Performance. In: Proceedings of EDOC 2009, Auckland, New Zealand (2009)
3. Council, S.: Supply Chain Operations Reference Model Version 7.0 (2005)
4. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
5. Karastoyanova, D., Houspanossian, A., Cilia, M., Leymann, F., Buchmann, A.: Extending BPEL for Run Time Adaptability. In: Proceedings of EDOC 2005, Enschede, The Netherlands (2005)
6. Danylevych, O., Karastoyanova, D., Leymann, F.: Optimal Stratification of Transactions. In: Proceedings of ICIW 2009, Venice, Italy (2009)
7. Tinelli, C.: A DPLL-based Calculus for Ground Satisfiability Modulo Theories. In: Flesca, S., Greco, S., Leone, N., Ianni, G. (eds.) JELIA 2002. LNCS (LNAI), vol. 2424, pp. 308–319. Springer, Heidelberg (2002)
8. Bozzano, M., Bruttomesso, R., Cimatti, A., Junttila, T., van Rossum, P., Schulz, S., Sebastiani, R.: An Incremental and Layered Procedure for the Satisfiability of Linear Arithmetic Logic. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 317–333. Springer, Heidelberg (2005)
9. Zeng, L., Benatallah, B., Dumas, M., Kalagnamam, J., Chang, H.: QoS-aware Middleware for Web Services Composition. IEEE Trans. on Software Engineering 30(5) (May 2004)
10. Bodenstaff, L., Wombacher, A., Reichert, M., Jaeger, M.C.: Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: Proceedings of SCC 2008, Washington, DC, USA (2008)
11. Jaeger, M.C., Muhl, G., Golze, S.: QoS-aware Composition of Web Services: An evaluation of selection algorithms. In: Proceedings of COOPIS 2005, Cyprus (2005)
12. Canfora, G., di Penta, M., Esposito, R., Villani, M.L.: QoS-Aware Replanning of Composite Web Services. In: Proceedings of ICWS 2005, Orlando, USA (2005)
13. Castellanos, M., Casati, F., Shan, M.C., Dayal, U.: iBOM: A Platform for Intelligent Business Operation Management. In: Proceedings of the 21st International Conference on Data Engineering (ICDE 2005), pp. 1084–1095 (2005)
14. Karastoyanova, D., Leymann, F., Buchmann, A.: An Approach to Parameterizing Web Service Flows. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 533–538. Springer, Heidelberg (2005)