A Case Study on Optimizing Web Service Monitoring Configurations

Garth Heward[1], Jun Han[1], Ingo Müller[1], Jean-Guy Schneider[1], and Steve Versteeg[2]

[1] Swinburne University of Technology, Hawthorn, Victoria, Australia
[2] CA Labs, Melbourne, Victoria, Australia
{gheward,jhan,imueller,jschneider}@swin.edu.au,
Steve.Versteeg@ca.com

Abstract. Whilst monitoring web services provides benefits in terms of demonstrating that Service Level Agreements (SLAs) have been met, monitoring comes with a cost on the QoS of delivered services. We present the application of our method for optimizing the configuration of a suite of web service monitors in order to minimise the QoS impacts of monitoring on a web service provider. We discuss the actions required for optimization, and benefits that were achieved. Through this case study, we highlight the possible benefits of optimization to a web service provider, and give details on how we achieve optimization.

Keywords: Web Services, Monitoring, Monitoring Optimization.

1 Introduction

WS (Web Service) providers give quality guarantees for their services, so that consumers are assured of the expected Quality of Service (QoS) of services before and during service consumption. To demonstrate that they have met these guarantees, providers monitor their services. However, achieving this monitoring goal introduces a problem, since monitoring can impact the delivered quality of web services. This impact has been demonstrated to be as much as 40% (for response time) for a single monitor[1]. In order to reduce these impacts, we have developed a method for optimizing a suite of WS monitors [2]. This optimization balances the benefits of monitoring, in terms of monitoring coverage and the costs of monitoring, in terms of QoS impacts.

In this paper, we present an application of our optimization technique on the IT system of a WS provider, in the form of a case study. This case study is designed to demonstrate both the information and procedures required for optimization, and the possible benefits for a WS provider. After presenting an overview of our optimization technique, we describe in detail its application to a realistic WS system. This includes how we determine an optimal configuration for the case study, and the effect that this configuration has on delivered QoS. To achieve this, we measure the effectiveness of the optimization by testing the case study system before and after optimization is applied.

In Section 2, our case study system to which optimization will be applied is described. In Section 3, our technique for optimization is described. In Section 4, the application of optimization onto the case study is described. In Section 5, a series of experiments applying optimization to the case study system are described and their results are provided and discussed.

2 iTravel Scenario

iTravel is a company that offers web services to travel agencies for booking flights and hotels. iTravel has web services flight_info and hotel_info, used to lookup flight and hotel information, and services book_flight and book_hotel, used to book flights and hotels. iTravel has SLAs with its consumers, giving guarantees for response time, reliability and compliance to privacy regulations. If iTravel is unable to demonstrate that they have met these requirements they must refund affected consumers a percentage of their monthly account fees. iTravel must also meet their corporate governance requirements. These state that iTravel is accountable for the security compliance and privacy compliance for their customers' information. iTravel is liable for large fines if they cannot demonstrate that these requirements have been met.

To demonstrate that the SLA and corporate governance requirements have been met, iTravel has a suite of monitors. iTravel has progressively added monitors into its system, each meeting a new monitoring requirement. iTravel now has a suite of monitors that can measure response time and reliability, and meet security and privacy compliance requirements by detecting security and privacy violations. In recent months, however, iTravel have not met their SLA obligations for response time or reliability, and were required to refund the account fees of affected consumers. Investigation revealed that WS-monitors were imposing a large performance burden on the WS system and the monitors in fact had been doing some overlapping monitoring. Furthermore, the use of some monitors had caused services to become unreliable due to non-response. Therefore, iTravel would like to have their monitors optimally configured so that they only monitor when it is valuable to do so (when requirements demand it), and only the least costly (in terms of QoS) monitor is selected for each monitoring requirement.

2.1 iTravel System

The iTravel system consists of three subsystems, one for each of Flights, Hotels, and Administration. Each subsystem is hosted on its own, dedicated hardware. This case study focuses on the Flights Subsystem, shown in Figure 1. As part of the normal application system, web services are shown in light grey, rounded boxes in the figure; the WS consumers (there may be many) are shown in the light grey, square box; and WS proxies are shown in boxes with dark grey backgrounds. The monitors are shown in numbered, square white boxes, including two Probe type monitors (flight_info Probe and book_flight Probe), two Interceptor type monitors (book_flight Interceptor and hotel_info Interceptor), and one

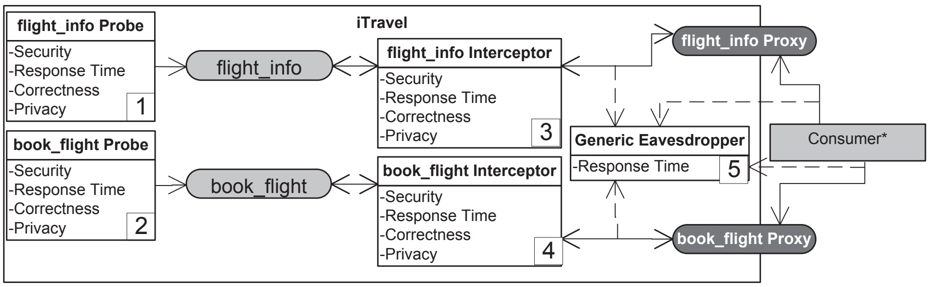

Fig. 1. iTravel Flight Subsystem

Eavesdropper monitor (Generic Eavesdropper). Solid lines represent service invocation/ communications, and dashed lines show points of message interception.

The aspects of the services being monitored concern security, response time, privacy, and reliability. Security monitoring supports the assessment of the compliance with security regulations by continuously producing well-defined and traceable evidence (hereafter simplified to 'Security') by checking messages for known weaknesses such as SQL injection and verifying that actions such as authentication have been performed. Response time monitoring measures the round-trip time for a service request, on the service provider's side. Privacy monitoring supports the assessment of the compliance with privacy compliance regulations (hereafter simplified to 'Privacy') by performing actions such as verifying and cleansing outgoing personal information. Monitoring for reliability checks that requests receive a response, and that the response is properly formatted.

The probe monitors (flight_info Probe and book_flight Probe) are capable of meeting requirements for monitoring response time, reliability, security and privacy of the web services by invoking the services and pretending to be service consumers. The probes are hosted on the same server as the web services.

The interceptor monitors (flight_info Interceptor and book_flight Interceptor) measure the same properties, but do so by intercepting the service requests of consumers, rather than generating their own requests as probes do. The interceptors have the capability to forward messages after analysing them, and therefore interceptors can filter or modify communications and perform functions such as firewalling. The interceptor monitors are hosted on their own, dedicated servers.

The eavesdropper monitor (Generic Eavesdropper) acts similarly to the interceptors; however, it cannot modify or stop a message from being transmitted, as it is only a passive listener. The Generic Eavesdropper is hosted on its own, dedicated server.

3 Overview of Monitoring Optimization

The optimization problem is to find a monitoring configuration that gives the maximum value in terms of QoS impact (costs) and monitoring coverage (benefits). In this section, we present an overview of our optimization approach in

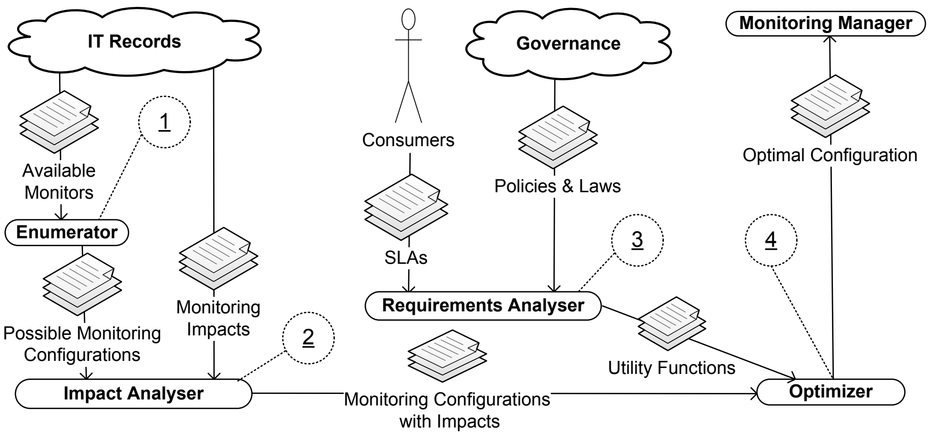

Fig. 2. Optimization Framework

order to describe what actions are performed to calculate an optimal configuration. The full details of this optimization approach can be found in [2,3].

Figure 2 shows a framework illustrating our approach to solving this monitoring optimization problem, annotated with the ordering of activities of the optimization process. In step **1**, the set of deployed monitors is identified based on IT records, and then the Enumerator generates all the possible monitoring configurations for this set of monitors. In step **2**, the performance and quality impacts of each monitor are identified from IT Records or benchmarking, and the Impact Analyser derives the total performance impact of each possible monitoring configuration. In step **3**, the monitoring requirements and associated penalties for not meeting them are identified from analysing SLAs, policies and laws, and the Requirements Analyser transforms requirements into a set of utility functions that define the benefit gained from monitoring a quality whilst a specific level of performance is being achieved. In step **4**, the Optimizer uses the set of utility functions from the Requirements Analyser to score all monitoring configurations with impacts from the Impact Analyser, and the monitoring configuration that yields the highest utility is selected and applied to the monitoring system.

4 iTravel Optimization

In order to obtain the best monitoring coverage whilst delivering acceptable QoS, we have applied optimization as described in Section 3 to the iTravel system as described in Section 2.

4.1 iTravel Baseline QoS

The response time for both the flight_info and book_flight services in the iTravel scenario has been measured under controlled (lab) conditions to be 2.5 seconds. Security and Privacy have been measured to be 100%, and as such are considered

100% as long as there is at least one monitor observing each of them at a sampling rate of 100% - otherwise they are 0%. Analysis of historical executions has been performed to determine that reliability is 98% under ideal conditions of no monitoring for the flight_info service, and 99% for the book_flight service.

4.2 iTravel Requirements

iTravel is bound by SLA and corporate governance requirements. The SLA requirements are from a single SLA that all service consumers have with iTravel, and the corporate governance requirements come from iTravel's business and legal obligations. The requirements on the flight subsystem are:

SLA1. flight_info Response Time will be $\leq$ 10s, penalty=33% account fee
SLA2. book_flight Response Time will be $\leq$ 10s, penalty=66% account fee
SLA3. flight_info will be $\geq$95% Reliable, penalty=100% account fee
SLA4. book_flight will be $\geq$98% Reliable, penalty=100% account fee
SLA5. flight_info will meet Privacy requirements, penalty=100% account fee
SLA6. book_flight will meet Privacy requirements, penalty=100% account fee
CG1. flight_info will meet Security requirements, penalty=$10,000
CG2. book_flight will meet Security requirements, penalty=$10,000
CG3. flight_info will meet Privacy requirements, penalty=$5,000
CG4. book_flight will meet Privacy requirements, penalty=$5,000

For example, SLA1 states that iTravel must deliver a response time of under 10 seconds for the flight_info service, otherwise the consumer must be refunded 33% of their monthly account fee (which is $100).

4.3 iTravel Monitoring Impacts

We divide the types of monitoring impacts into monitoring instance impacts (iMI) and monitoring overhead impacts (iMO). The monitoring instance impacts are those that occur each time a monitor is used. For example, there may be an impact on response time whenever a probe monitor is used to test a service, and the more times the monitor is used, the higher the impact will be. Monitoring overhead impacts differ in that they are a fixed impact that occurs whenever a monitor is enabled. For example, using an interceptor-based monitor may increase response time by a fixed amount (due to extra transmission time), regardless of how many events that monitor observes.

We have measured the monitoring instance impacts and monitoring overhead impacts[1] for response time and reliability in the iTravel system. These impacts are an average percentage increase of response time and decrease of reliability, compared with the response time and reliability of a service without monitoring. For example, using the book_flight Probe will decrease reliability by 4% and increase response time by 2% for the book_flight service, plus an additional 1% on response time for each property measured (security, response time, privacy, or reliability). Additionally, each quality measured with the book_flight Probe impacts

[1] Available at `http://www.ict.swin.edu.au/personal/gheward/`

the response time of the flight_info service by 0.5%. The interceptor monitors have a fixed impact regardless of how many qualities of service are monitored, e.g. the impact of monitoring security, response time, reliability and privacy of flight_info using the flight_info Interceptor is the same as the impact of monitoring only security. Note that some monitors have an impact on the service that they *aren't* monitoring. For example, the flight_info Probe reduces the response time of book_flight. These impacts occur due to monitors and services sharing the same, limited set of resources.

4.4 iTravel Requirements Analysis

The Requirements Analyser transforms the iTravel requirements iR (described in Section 4.2), into a set of utility functions iU[2] describing the utility to be gained for meeting these requirements. In iTravel's case, utility is directly and linearly computed from fines for non-compliance, however the set of utility functions may be modified or extended to add custom requirements or model some non-linear relationship. iTravel's requirements (section 4.2) are transformed into the set of iTravel utility functions, in format (service, quality type, min. quality level, min. sampling rate)$\mapsto$utility:

(flight_info, Response Time, 0.25, 1) $\mapsto$ 0.165 (3300),
(book_flight, Response Time, 0.25, 1) $\mapsto$ 0.33 (6600),
(flight_info, Reliability, 0.95, 1) $\mapsto$ 0.5 (10,000),
(book_flight, Reliability, 0.98, 1) $\mapsto$ 0.5 (10,000),
(flight_info, Privacy, 1, 1) $\mapsto$ 1 (20,000),
(book_flight, Privacy, 1, 1) $\mapsto$ 1 (20,000),
(flight_info, Security, 1, 1) $\mapsto$ 0.25 (5,000),
(book_flight, Security, 1, 1) $\mapsto$ 0.5 (5,000).

This set describes the utility received from meeting each requirement. Utilities have been normalised to values between 0 and 1. Actual monetary values are shown in brackets, e.g. 1 (20,000) to clarify the derivation of the utility levels. All utility functions are defined incrementally, so that the total utility of a system is the sum of each individual utility function's value.

4.5 iTravel Optimization

The sets of available monitors, their impacts and their overheads were used to calculate the set of possible monitoring configurations with impacts for iTravel, $iMCI$. $iMCI$ maps each possible monitoring configuration to a set of net monitoring impacts. This information on monitoring requirements and impacts, along with the baseline response times and reliability for the iTravel web services has been used to perform optimization of iTravel's WS monitoring system.

The utility u of each configuration in $iMCI$ was calculated, and the configuration in $iMCI$ that yields the highest utility was selected and applied to the system. For iTravel, this configuration was the flight_info Interceptor and book_flight

[2] Available at `http://www.ict.swin.edu.au/personal/gheward/`

Interceptor both being set to monitor every quality of their respective target services, and every other monitor being disabled. The impact of this configuration is the sum of the impacts for each service (10% for Response Time on both services, 2% for reliability on flight_info, and 1% for reliability on book_flight). Therefore, the resulting response times of each service were $(2.5/(1 - 0.10) \approx 2.8)$, under no load, the reliability for flight_info was $(0.98 - 0.2 = 0.96)$, and the reliability for flight_info was $(0.99 - 0.01 = 0.98)$.

5 iTravel Optimization Evaluation

The iTravel case study has been implemented to verify our optimization technique and measure the possible QoS benefits of monitoring optimization. We now present a set of experiments and results to demonstrate the benefits of applying monitoring optimization to the iTravel system.

There always exists one or more optimal monitoring configurations (yielding the highest utility) in terms of monitoring coverage and QoS provided. These tests are designed to discover these optimal monitoring configurations in the iTravel system, and compare the utility and actual QoS of an optimal configuration to the utility and actual QoS of the corresponding maximum (un-optimized) monitoring configuration, in which all monitors run at sampling rates of 100%.

5.1 Experiment Configuration

The baseline QoS and monitoring impacts of the iTravel system were used to develop a simulation system that allows for the QoS of any particular monitoring configuration to be estimated through a series of simulated executions over a fixed period. For these tests, that fixed period was 6,000 seconds. This figure was selected through analysis of test results, since the results of all tests stabilised before 6,000 seconds. The simulations use response time measured through real service invocations to determine what the performance will be for a given monitoring configuration at any particular load level. Additionally, the reliability of all invocations is determined after each simulation.

As discussed above, a set of utility functions representing the iTravel requirements has been generated. These were used along with measured monitoring impacts to perform optimization on the iTravel monitoring system. The optimal monitoring configuration for iTravel was enabling both Interceptor monitors at 100% for all qualities of service, and disabling all other monitors.

Note that, since we are presenting a simple scenario, it is possible to select this optimal configuration manually by examining the system requirements and impacts. However, we intend for these optimizations to be applied to larger and more complex scenarios for which optimization would be too difficult to achieve without automation. Furthermore, optimization may be applied frequently at run-time, as parameters such as requirements and load levels change. In this case, it would not be effective to have a human constantly checking the system for changes and re-optimizing as they occur.

5.2 Results

We report the resulting average response times, reliability (percent of correctly returned invocations), and net utilities for each simulation. Response time and reliability results demonstrate that the performance and reliability of the iTravel system has increased by applying an optimal monitoring configuration. Utility results demonstrate that we have met iTravel's monitoring requirements. The total utility for the optimized configuration was higher than that of the maximum configuration in each test, as only the minimum valuable monitoring level was met. This allowed for a QoS increase, thus minimising the chance of a penalty to consumers for poor QoS.

Figure 3 shows response time versus load level for the iTravel scenario with unmonitored, maximum, and optimal monitoring configurations. The load levels on the horizontal axis represent the average number of active client requests, whilst the vertical axis gives the average response time over both services. The dotted line indicates average response times under maximum monitoring, the dashed line indicates the average response times under optimal monitoring, and the solid line represents the average response times under no monitoring. Since the response times for each service were almost identical, the results in Figure 3 represent the performance of both the book_flight and flight_info services.

Optimization reduced the average response time of the iTravel scenario by approximately 65% from maximum monitoring (from 33 to 12 seconds on average), and reduced the average response time impact of monitoring by over 50%(the optimal monitoring configuration was on average 11% slower than no monitoring, versus the maximum monitoring configuration's 68%).

The horizontal, solid line on Figure 3 shows the 10-second boundary, for which penalties apply in the iTravel system. The optimized configuration stays under this boundary for three times longer than the maximum monitoring configuration, i.e. the system with optimized monitoring dealt with three times as much load before a penalty for slow response times would have been paid.

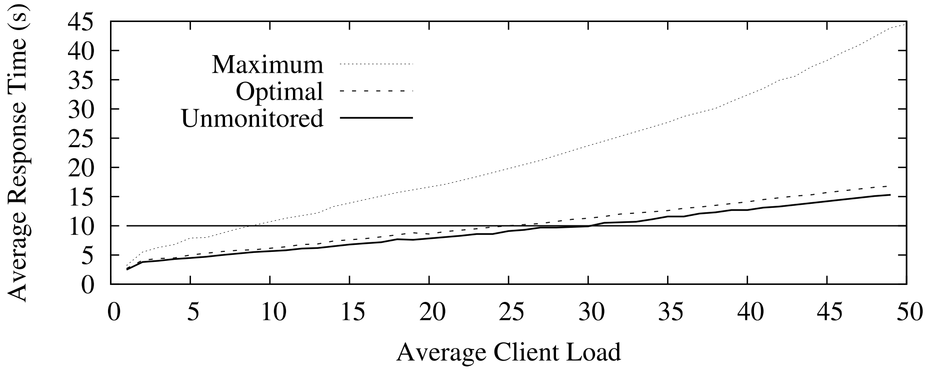

Fig. 3. iTravel Optimization Results

In terms of reliability, the unmonitored system maintained levels of 98% for flight_info and 99% for book_flight, whilst the optimal monitoring configuration yielded levels of 96% for flight_info and 98% for book_flight, and the maximum monitoring configuration yielded 92% for flight_info and 95% for book_flight.

The utility provided by the unmonitored solution was 0, since no requirements would have been able to be verified. Conversely, the maximum and optimal monitoring configurations both included complete monitoring coverage. Of the optimal and maximum monitoring configurations, only the optimal monitoring configuration yielded reliability levels that met requirements, so the utilities for reliability were received (0.5 for each service) under that configuration. For the optimal configuration, the utility at or below response time of 10 seconds was 4.245 (the highest achievable utility), and the utility above 10 seconds was 3.75. For the maximum configuration, the utility at or below response time of 10 seconds was 2.75, and the utility above 10 seconds response time was 2.25.

iTravel is a special case, since all of the utility functions have been directly derived from requirements with fines for non-compliance. For this reason, we can revert the expected utility values back to monetary terms, for comparison of each monitoring configuration. For this case study, 1 util is worth $20,000. Therefore, the expected benefits of the optimal configuration over the default, maximum configuration are $(4.245 - 2.75) \times \$20,000 = \$29,900$ when the response time is under 10 seconds, and $(3.75 - 2.25) \times \$10,000 = \$30,000$ when the response time is over 10 seconds.

The test results demonstrate that both the utility and QoS of the iTravel case study system were increased by optimizing the configuration of WS monitors. Although the results cannot be generalised to all WS systems as they depend on system properties, benefits can be substantial, even for a simple system. These benefits depend on the amount of overlapping functionality of a monitoring system, the level of impacts of monitors, the benefits of monitoring, and the benefits of achieving performance goals. As such, the benefit of optimization will increase as a WS system becomes more complex, with more overlapping monitors to choose from, more system elements to monitor, and a greater number of monitoring and performance requirements to meet.

6 Related Work

There are various methods for optimizing WS-based systems through selective execution or load balancing [4,5]. Whilst highlighting the need to optimize the performance of WS systems, none of these techniques relate to the management of a WS monitoring system.

Ranganathan and Dan present a Web Services management system to monitor and reallocate local system resources for services based on comparing their current QoS to their SLAs[6]. Whilst performing system-level administration, this method does not re-configure the web service monitoring system.

Baresi and Guinea present a method for dynamic monitoring of WS-BPEL processes, which uses high level monitoring rules [7]. These monitoring rules are used to control the monitoring of each WS-BPEL process. The rules are created with an equivalent of debug levels (one to five), which allows for optimization in terms of performance versus monitoring trade-offs at run-time. The monitoring level must be set for each service as a unit. Rather than assigning a monitoring resolution (debug level) to each individual service in the system, we assign a monitoring resolution to each quality of each service, directly reflecting requirements from SLAs. Furthermore, we enhance the optimization by selecting those monitors which will most efficiently monitor each service.

Overall, there have been numerous efforts for optimizing the QoS of web services and web service compositions, which consider the selection of services in order to optimize QoS. We have discovered no other work that optimizes a web services monitoring system by trading off between monitoring costs and benefits, directly translated from SLAs and other requirements for monitoring.

7 Conclusions and Future Work

We have described the optimization of a suite of WS monitors for a case study, and demonstrated the possible benefits of optimizing a WS monitoring configuration. This optimization uses sets of monitoring requirements and impacts of WS monitors to configure those monitors in a way that yields an optimal mix of monitoring coverage and monitoring impacts.

We will extend our optimization work in three directions. First, we will develop a heuristic approach to optimization to ensure scalability of the technique. This new heuristic technique will allow for faster optimization of a large system (hundreds of services and monitors), so that the system can be re-optimized either overnight, or continuously. Second, the framework and implementation will be extended so that run-time properties of the system being monitored are fed back into the monitoring optimization framework, for re-optimization at run-time. This will allow for the system to maintain an optimal configuration, even with changes to aspects such as requirements, system load, available monitors, or their impacts. Third, we will develop a method for using techniques to predict the future system state, and optimize a suite of monitors according to this predicted future state, rather than the current system state. This will include attempting to continuously verify or update the monitoring impacts on the system.

In the future, the complexity of this case study will be extended to allow for more services and monitors. Additionally, we will modify the case study to allow for run-time optimization and automated monitoring configuration, in order to demonstrate the possible benefits of these approaches.

Acknowledgements. This work is supported by the Australian Research Council in collaboration with CA Labs.

References

1. Heward, G., Müller, I., Han, J., Schneider, J.-G., Versteeg, S.: Assessing the performance impact of service monitoring. In: Australian Software Engineering Conference (ASWEC 2010), pp. 192–201 (2010)
2. Heward, G., Han, J., Müller, I., Schneider, J.G., Versteeg, S.: Optimizing the configuration of web service monitors. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 587–595. Springer, Heidelberg (2010)
3. Heward, G., Han, J., Müller, I., Schneider, J.G., Versteeg, S.: Optimizing the configuration of web service monitoring configurations. Technical Report, Swinburne University of Technology (2010)
4. Ludwig, H., Dan, A., Kearney, R.: Cremona: An architecture and library for creation and monitoring of ws-agreements. In: International Conference on Service Oriented Computing (ICSOC 2004), pp. 65–74 (2004)
5. Herssens, C., Jureta, I.J., Faulkner, S.: Dealing with quality tradeoffs during service selection. In: International Conference on Autonomic Computing (ICAC 2008), pp. 77–86 (2008)
6. Ranganathan, K., Dan, A.: Proactive management of service instance pools for meeting service level agreements. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 296–309. Springer, Heidelberg (2005)
7. Baresi, L., Guinea, S.: Towards dynamic monitoring of ws-bpel processes. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 269–282. Springer, Heidelberg (2005)