Towards Proactive Cross-Layer Service Adaptation*

Chrysostomos Zeginis, Konstantina Konsolaki,
Kyriakos Kritikos, and Dimitris Plexousakis

Information Systems Laboratory, ICS-FORTH, Heraklion, Greece
{zegchris,konsolak,kritikos,dp}@ics.forth.gr

Abstract. Service-Based Applications (SBAs) enable the automation of business processes. Therefore it is crucial to monitor their non-functional properties and take adaptation actions when QoS violations occur, across all functional layers. In this paper we propose a framework for the proactive cross-layer adaptation of SBAs. We exploit a cross-layer monitoring mechanism to detect a wide range of events, based on which we can both reactively and proactively adapt the system. In particular, the detection of event patterns help us to prevent future faults and failures from happening, by firing specific, dynamically derived rules, that map event patterns to suitable adaptation strategies. Our framework is validated using a traffic management scenario.

Keywords: monitoring, proactive adaptation, cross-layer, pattern, rules.

1 Introduction

Web Services evolution provides testimony of a move towards combining existing and new applications in order to provide more complex SBAs. An SBA is constituted by three main layers [4]: The Business Process and Management (BPM) layer provides the business process along with the activities, the involved roles and the Key Performance Indicators (KPIs), that define and measure progress toward organizational goals. The Service Composition and Coordination (SCC) layer refines this process by combining suitable Web services accompanied by the corresponding SLAs and the Service Infrastructure (SI) layer provides the underlying infrastructure. While Web services evolve in a very dynamic and vulnerable environment, they need to be monitored to detect violations of their functional and non-functional properties. Moreover, whenever a violation is detected, it is crucial to adapt the SBA accordingly. Web service monitoring and adaptation should take place across all the SBA layer, since these layers are closely related with many dependencies among them.

In this paper, we analyze our approach towards proactive cross-layer service adaptation. Firstly, Section 2 summarizes the related work. Section 3 presents

* This work constitutes preparatory foreground work on the FP7 IP PaaSage (Model-based Cloud Platform Upperware).

X.S. Wang et al. (Eds.): WISE 2012, LNCS 7651, pp. 704–711, 2012.

an updated version of ECMAF [10] which enables the efficient derivation and management of rules that map event patterns to suitable adaptation strategies, i.e. workflows of adaptation actions. Then, we exemplify the framework's functionality. The proposed monitoring engine captures monitored events from all the SBA layers, while the adaptation mechanism exploits the monitored events to derive new event patterns and the corresponding rules, in the way described in Section 4. Section 5 validates the framework using a specific case study. Finally, some concluding remarks and future work directions are presented in Section 6.

2 Related Work

During the past years some approaches towards cross-layer monitoring and adaptation have been proposed. CLAM [11] is a cross-layer adaptation manager, which identifies the application capabilities affected by the adaptation actions and an adaptation strategy that solves the adaptation problem. In [9] the authors propose a methodology for the dynamic and flexible adaptation of multi-layer applications using adaptation templates and taxonomies of adaptation mismatches. Guinea et al. [3] present an integrated approach for monitoring and adapting multi-layered SBAs, based on a variant of MAPE control loops [5]. Gjørven et al. [2] propose an approach towards cross-layer self-adaptation, which exploits mechanisms across two layers, Service Interface and Application, related to SCC and BPM layers respectively. In [8] the authors present an AOP-based approach towards runtime adaptation of service compositions for preventing SLA violations. Finally, in [6], the authors present the design and implementation of an experimental facility for cross-layer adaptation. The aforementioned approaches are compared in Table 1 according to a set of criteria, that are explained in [10].

Table 1. Comparison of Cross-layer Monitoring and Adaptation approaches

Approaches	**Cross-layer**	**Dynamicity**		**Intrusiveness**		**Timeliness**		**Type of properties**	
		Mon.	Adapt.	Mon.	Adapt.	Mon.	Adapt.	Kind	Scope
Zengin et al. [11]	✓	–	✓	–	∼	∼	R	NF-F	I-C
Popescu et al. [9]	✓	–	✓	–	✓	Im	R	F	I
Guinea et al. [3]	✓	✓	✓	✓	∼	Im	R	NF-F	I-C
Gjørven et al. [2]	✓	∼	✓	∼	–	∼	R	F	I
Leitner et al. [8]	✓	✓	✓	✓	✓	Im	R-P	NF	I
Jiang et al. [6]	✓	–	✓	–	∼	Im	R	NF	I-C
ECMAF	✓	✓	✓	–	–	Im	R-P	NF-F	I-C

✓: Satisfaction, –: Unsatisfaction, ∼: Uncertainty, Im: Immediate, R: Reactive, P: Proactive, F: Functional, NF: Non-Functional, I: Instance, C: Class

From the comparison of the related cross-layer approaches we conclude that none of them satisfies all the criteria. As far as dynamicity is concerned, most of these approaches perform dynamic adaptation unlike dynamic monitoring. In addition, most approaches are non-intrusive regarding monitoring and adaptation, while all of them are reactive rather than proactive. Finally, as far as the

type of properties is concerned, they mostly focus on non-functional properties, while the scope of most of them is instance-based.

The main strengths of our proposed framework are the ability to handle both functional and non-functional properties, its proactive adaptation capabilities, by exploiting detection of warning event patterns and the efficient rule management, and its extensibility, as it can integrate new monitoring and adaptation techniques with the existing ones, while preserving its functionality and integrity.

3 The ECMAF Framework

The proposed framework (ECMAF) (Fig. 1), presented in [10], has been slightly changed and partially implemented with the purpose of providing an event-based approach towards cross-layer service monitoring and adaptation. Its main architectural differences with the previous one are the detailed view of the Monitor Manager, the addition of the Rule Manager and the integration of the reasoner in the Event Pattern Detector. Moreover, we have extended the framework's functionality, so as to efficiently address both reactive and proactive adaptation.

We adopt three types of events in our approach. **Successful events** carry information about successful invocations and normal state of the system components. **Warning events** indicate that a component (device, service, software, etc) of our system does not perform normally and a monitoring property has exceeded the warning threshold, which has been defined by the service requester as part of the SLA (e.g. service execution time surpasses warning threshold of 100ms). These events may lead to failing events, helping us to proactively adapt our system. **Failing events** indicate that a failure appeared during the SBA

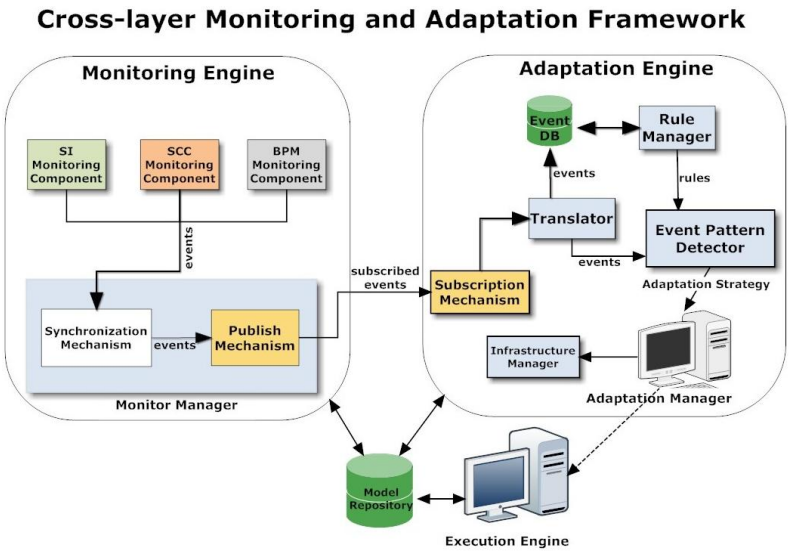

Fig. 1. Architecture of ECMAF

execution. These events are detected by the monitoring engine when a monitoring property is violated. Moreover, failing events, as well as warning events, can capture functional properties, as for example the service input type.

Concerning monitoring of the BPM and the SCC layers, we are using the Astro monitoring tool [1]. As Astro exploits the ActiveBPEL engine, it provides the ability to monitor service compositions implemented in BPEL. It supports both instance and class monitors, as well as the monitoring of functional and non-functional properties. The framework detects violations at the SCC and the BPM layers by comparing the monitoring property values with the predefined thresholds, and records them at a log file. The SI monitoring is performed by the Nagios (`http://www.nagios.org/`) monitoring tool, which offers complete monitoring and alerting for servers, switches, disks and other types of infrastructural components. Both monitoring tools also provide success and warning events. For the definition of the monitored properties we exploit the OWL-Q non-functional service description language [7], which has been extended accordingly so as to allow the definition of functional properties.

As far as adaptation is concerned, our approach relies on detecting patterns of monitored events, in order to prevent future failing events. Initially, the subscribed monitored events are collected by the Siena subscription mechanism (`http://www.inf.usi.ch/carzaniga/siena/`), which, in turn, passes them to the *Translator* and are finally stored in a MySQL *Event Database*. The *Rule Manager* regularly consults the database to derive new event patterns and the corresponding mapping to suitable adaptation strategies, in the form of new rules, which are then sent to the Event Pattern Detector (EPD). This procedure is explained in Section 4. The *Translator*, a Java-based program, transforms the monitored events to the respective format of the *Event Pattern Detector*. For event processing and comprehensive pattern detection, we use Esper (`http://esper.codehaus.org/`), which can detect an event pattern while new monitored events are delivered to the EPD. Then, EPD detects the event pattern that corresponds to the body of a specific rule produced by the Rule Manager and in this way selects the appropriate adaptation strategy that corresponds to the rule head and forward it to the Adaptation Manager. Each adaptation strategy is mapped to a BPEL file which determines which adaptation actions is performed by which component as well as the ordering of the adaptation actions.

Finally, the *Adaptation Manager* is responsible to provide the appropriate BPEL file, which realizes the adaptation strategy, by defining the responsible component for each adaptation action and the ordering of these actions. Complex adaptation strategies are produced from simpler ones in a bottom-up manner by merging the corresponding BPEL files accordingly. The *Infrastructure Manager* is able to treat malfunctions regarding the SI layer, such as memory reallocation, server switching and other. This tool is planned to be a separate one with infrastructure adaptation capabilities. The *Execution Engine*, which is planned to be an extension of the existing Astro BPEL engine with adaptation capabilities, is called to apply the adaptation strategies.

4 Rule Derivation

In order to define the patterns of monitored events for the executed SBA, the Rule Manager is regularly querying the Event Database to discover the event patterns that lead to failing events. A number of SBA instance invocations have to be considered, so as to find these event sequences that are repeated for many times and have always the same SBA failing result. Then, the Rule Manager marks this sequence as an event pattern for this SBA. Unique identifiers, such as business process IDs, service IDs and infrastructure IDs, are exploited for event correlation during pattern derivation.

After defining an application-specific event pattern, that may contain events from all three layers, the Rule Manager maps this pattern to an adaptation strategy in the following way. Some predefined simple rules, mapping events to specific adaptation strategies, are produced manually by the application manager and are passed to the Rule Manager. Each monitored event can be mapped to more than one adaptation strategy, but we keep a priority of these strategies, defined by the application manager, to express their suitability. The Rule Manager extracts only the rule with the highest priority for each monitored event. Simple events are mapped to one or more adaptation strategies, while the strategies of more complicated event patterns (containing more that one event) are produced from the individual strategies of the events that participate in the pattern. We assume that each unique adaptation action has a unique name, in order to facilitate the strategy matching. Finally, the Rule Manager extracts a rule that is passed to the Event Pattern Detector. The following techniques, exploiting the strategy sets that have been mapped to each individual event of the pattern, are used to examine which is the best strategy combination.($S(e)$ symbolizes the set of strategies assigned to the monitored event e):

Case 1: The optimum solution derives from the intersection of the adaptation strategies. If the intersection is non-empty, then the event pattern is associated to all the strategies of the intersection but only one of them is selected to be the most suitable one, according to strategies priority. In particular, the rationale is to multiply the priorities and take the intersection with the highest combined priority. Big products mean high priority. **Case 2:** The worst case is the situation, where there is no intersection among the strategy sets. In this case we choose the sets union, i.e. the adaptation strategy with the highest priority for each one of the monitored events. Moreover, the other strategy combinations are stored and assigned a priority. **Case 3:** In any other case, the overlapping between the sets is considered to decide the adaptation strategies. For instance, if S(e1) is overlapped with S(e2) and S(e2) is overlapped with S(e3), then the adaptation strategy will comprise two adaptation strategies with the highest priorities from the following set $(S(e1) \cap S(e2)) \cup (S(e2) \cap S(e3))$.

5 Case Study

Our work aims at locating the warning event and taking adaptation actions in order to prevent future failing events, as well as to reactively adapt the SBA

when proactive adaptation fails to detect the failure. The application domain considered in this paper concerns a traffic management system, which is analyzed in detail in [10] (Fig. 2). At the SCC layer, the oval shapes represent the third-party composite services, the rectangles the internal composite services, while the dashed ones the manual activities. We suppose that the domain expert specifies in a separate document his goals for the quality of the system using a set of KPIs using OWL-Q. In addition, she defines adaptation strategies that can be taken in order to adapt the business process as well as the initial rules. The corresponding SLAs of the web services are stored in the model repository. We assume that the following manual rules have been imported in the Rule Manager:

- *available memory <100MB ⇒memory realloc.* (*R.* 5.1, *Pr.* 1)
- *elapsed time of assessment service >500ms ⇒memory realloc.* (*R.* 5.2, *Pr.* 1)
- *execution time of assessment service >1sec ⇒DCS migration* (*R.* 5.3, *Pr.* 1)
- *process duration >10min ⇒recomposition* (*R.* 5.4, *Pr.* 1)
- *available memory < 100MB + elapsed time of assessment service >500ms ⇒ memory realloc.* (derived from Rules 5.1, 5.2) (*R.* 5.5, *Pr.* 1)

At run time, we continually collect monitored events from the Astro and the Nagios monitoring tools. Successful events indicate that the system is running normally and the adaptation manager does not have to take any actions. After a certain period of time, the monitoring engine detects that the assessment service has currently low available memory (below warning threshold of 100MB). At the same time we notice that this service is not running optimally and the Astro monitoring tool detects a warning event indicating that its elapsed time since its invocation has exceeded the warning threshold of 500ms. These events are passed directly to the Monitor Manager and delivers them to the Adaptation Engine. This event sequence appeared many times in the past SBA execution history leading to the occurrence of the failing event, so the Rule Manager has derived a corresponding event pattern and mapped this pattern to the most suitable adaptation strategy according to the predefined manual rules. Specifically, the exported rule 5.5 stems from the intersection of the adaptation strategies of the

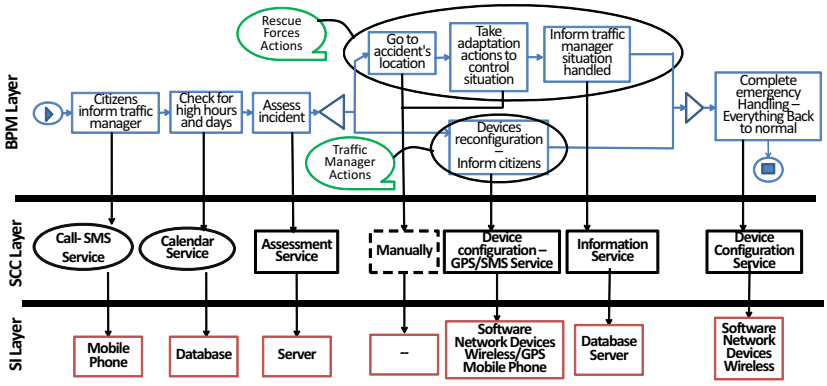

Fig. 2. Critical traffic conditions - Traffic management scenario

two warning events. The strategy's suitability lies on the fact that by executing this service with better memory allocation, the probability that the SLA guarantee corresponding to the service execution time is not violated becomes very high. However, the service execution time is finally violated. Consequently, the respective event is detected and Rule 5.3 is fired, aiming at addressing a future KPI violation, i.e. that the process duration surpasses the 10 minutes threshold, by migrating the Device Configuration Service (DCS) to another more powerful machine to compensate for the additional time spent by the assessment service. This strategy is performed in parallel with the manual activities, which consume most of the process time, so it does not introduce any additional delay.

6 Conclusions and Future Work

To sum up, in this paper we have presented an extended framework, that can efficiently deal with both reactive and proactive cross-layer of adaptation of SBAs. Its main contribution is the pattern-based handling of monitored events in order to perform adaptation. On the one hand, it is both the detection of warning event patterns and the efficient rule management that enables the proactive SBA adaptation, and on the other hand, the efficient rules management reinforces its reactive adaptation capabilities. Furthermore, initial validation of our approach has been presented, based on a traffic management scenario. As future work, we will finalize the framework by implementing adaptation enactment mechanisms, that will experimentally evaluated. In addition, the framework will be enriched with new capabilities to capture the state of the different applications.

References

1. Barbon, F., Traverso, P., Pistore, M., Trainotti, M.: Run-time Monitoring of Instances and Classes of Web Service Compositions. In: ICWS, pp. 63–71. IEEE (2006)
2. Gjørven, E., Rouvoy, R., Eliassen, F.: Cross-layer self-adaptation of service-oriented architectures. In: MW4SOC, pp. 37–42. ACM (2008)
3. Guinea, S., Kecskemeti, G., Marconi, A., Wetzstein, B.: Multi-layered Monitoring and Adaptation. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 359–373. Springer, Heidelberg (2011)
4. Hielscher, J., Metzger, A., Kazhamiakin, R.: Taxonomy of adaptation principles and mechanisms. S-Cube Project Deliverable (2009)
5. Horn, P.: Autonomic Computing: IBM's Perspective on the State of Information Technology. Tech. rep. (2001)
6. Jiang, S., Hallsteinsen, S., Lie, A.: An experimental facility for cross-layer adaptation of service oriented distributed systems. In: NIK, pp. 97–108 (2011)
7. Kritikos, K., Plexousakis, D.: Semantic QoS Metric Matching. In: ECOWS. IEEE, Zurich (2006)

8. Leitner, P., Wetzstein, B., Karastoyanova, D., Hummer, W., Dustdar, S., Leymann, F.: Preventing SLA Violations in Service Compositions Using Aspect-Based Fragment Substitution. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 365–380. Springer, Heidelberg (2010)
9. Popescu, R., Staikopoulos, A., Liu, P., Brogi, A., Clarke, S.: Taxonomy-driven Adaptation of Multi-Layer Applications using Templates. In: SASO (October 2010)
10. Zeginis, C., Konsolaki, K., Kritikos, K., Plexousakis, D.: ECMAF: An Event-Based Cross-Layer Service Monitoring and Adaptation Framework. In: NFPSLA-SOC. Springer (2011)
11. Zengin, A., Marconi, A., Pistore, M.: CLAM: Cross-layer Adaptation Manager for Service-Based Applications. In: QASBA 2011, pp. 21–27. ACM (2011)