Service Adaptation Recommender in the Event Marketplace: Conceptual View

Yiannis Verginadis[1], Ioannis Patiniotakis[1],
Nikos Papageorgiou[1], and Roland Stuehmer[2]

[1] Institute of Communications and Computer Systems,
National Technical University of Athens,
{jverg,ipatini,npapag}@mail.ntua.gr
[2] FZI Forschungszentrum Informatik, Karlsruhe, Germany
stuehmer@fzi.de

Abstract. In this paper, we present the conceptual architecture of a highly scalable federated platform for processing vast number of events coming from distributed sources, in order to detect interesting situations that lead to service adaptation recommendations. The approach presents an Event Marketplace, a platform for mediating between event providers and (complex) event consumers in very large and heterogeneous environments which is enriched with timely reaction capabilities, through situational-driven service adaptations. We focus on an approach for detecting real-time interesting situations that lead to service adaptations.

Keywords: Event Marketplace, Distributed Systems, Service Adaptation.

1 Introduction

Adaptation as one of the basic phenomena of biology is the evolutionary process whereby a population becomes better suited to its habitat [1], [2]. The notion of adaptation has been extensively used, especially nowadays, in the computer science domain. It is considered as one of the most desired functionalities of today's highly dynamic, distributed and ubiquitous environments in the service-oriented setting. The need for highly flexible services that can be orchestrated in order to provide certain behaviors and at the same are able to react and adapt inside their "habitat" (i.e. change based on the context and events that formulate the imprinting of a highly dynamic service environment), is considered to be a desired but difficult to achieve fact.

The recent expansion of the "everywhere" deployment of the wireless sensors networks (part of the Internet of Things) introduces the information from stationary or moving objects in the global service adaptation task. Let us consider the following scenario as an example of ubiquitous interaction between services resulting in a more personalized and adapted service execution:

"Paul is a businessman who has been flying from Paris to New York. He used the entertainment service on board, but hasn't finished watching the movie before the landing. Two hours later he is entering his room in the downtown hotel he booked earlier and wow: the room entertainment service is ready to play the movie Paul was

R. García-Castro et al. (Eds.): ESWC 2011 Workshops, LNCS 7117, pp. 194–201, 2012.

watching in the plane – of course only the unfinished part." Such a scenario involves the ubiquitous interaction between services, the processing of dispersed events and the actual adaptation of service execution in a way that satisfies the customer. All these constitute multidimensional problems. Let us name just a few challenges. First of all, such an interaction cannot be modeled in the design time, since it is not possible to predict all interesting interactions in advance. Secondly, a common understanding between different actors in a heterogeneous environment is requested. Next, it is challenging to ensure a proper (on time, complete and relevant) delivery of interesting information in a large scale distributed environment.

In this paper we present the concept and the architecture of a platform that can satisfy these requirements (section 2). The platform uses its cloud-computing nature to be "elastic" and operate in the "pay as you go" mode. We can consider the platform as a kind of the Event marketplace (similar to the notion of the service marketplace) where events coming from different producers can be arbitrary combined by different event consumers. In addition, we focus and give details about a core part of this platform that recommends changes (adaptations) of services' configurations, composition or workflows, in order to overcome problems and achieve higher performance by reacting to real time events (i.e. formulating an interesting situation) at the right time and in the right way (section 3). In section 4, we present the related work of our approach, whereas section 5 contains concluding remarks.

2 Proposed Conceptual Architecture

The initial conceptual architecture for our platform is depicted in Figure 1. In this section we introduce the components and present their main functionalities.

The Distributed Service Bus (DSB) provides the service oriented architecture (SOA) and event driven architecture (EDA) infrastructure for components and end user services. It acts as the basis for service deployments, and processes (BPEL, BPMN), routing synchronous and asynchronous messages from services consumers to service providers. Based on the principles of the system integration paradigm of Enterprise Service Bus the DSB is distributed by nature.

The Governance component allows users to get information about services and events, as well as specifying QoS requirements as service level agreement (SLA) contracts using the WS-Agreement standard [3]. The Governance component extends a standard Service-based governance tool (OW2 Petals Master [4]) by adding governance mechanisms for event-based systems. Its role is to provide ways to govern services and events. It provides standards-based APIs and a graphical user interface.

The Event Cloud provides storage and forwarding of events. The role of the Event Cloud is a unified API for events, real-time or historic. To that end, it contains a peer-to-peer network to store histories of events durably in a distributed fashion. In the same way the list of subscribers is distributed across the peers to notify subscribers if a given new event is stored at any node and a corresponding matching subscription exists in the system. Subscriptions may use a simple set of operators such as conjunctive queries which can be evaluated efficiently on a single peer. More complex queries are executed in the DCEP component.

The DCEP component (Distributed Complex Event Processing) has the role of detecting complex events and reasoning over events by means of event patterns defined in logic rules. To detect complex events, DCEP subscribes to the Event Cloud for any simple event defined in the event patterns at a given point in time. DCEP supports traditional event operators such as sequence, concurrent conjunction, disjunction, negation etc., all operators from Allen's interval algebra [5] (e.g., during, meets, starts, finishes etc.), window operators, filtering, enrichment, projection, translation, and multiplication. Out-of-order event processing is supported (e.g. events that are delayed due to different circumstances such as network anomalies).

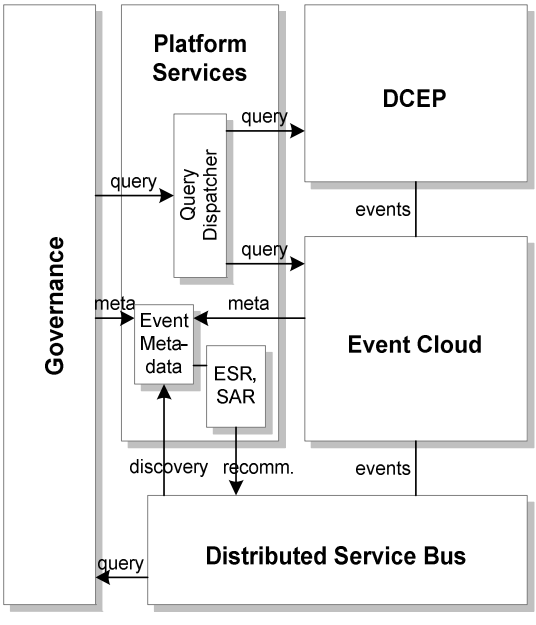

Fig. 1. Conceptual Architecture

The Platform Services component incorporates several functional additions to the platform as a whole. The Query Dispatcher has the role of decomposing and deploying user subscriptions in pieces supported by the Event Cloud and DCEP respectively, taking into account the expressivity supported by the two target components. The Event Metadata component stores information about events, such as source descriptions, event type schemas, etc, to enable the discovery of relevant events for an event consumer and to provide data to the subscription recommender. The ESR and SAR components form the Event Subscription Recommender (ESR) and Service Adaptation Recommender (SAR). ESR will recommend or perform dynamic subscriptions for complex events based on situations detected from semantic-enabled events and the context of subscribed services. Thus, ESR will provide assistance to services that will have the option to be subscribed to specific events at the “right time” without the services having complete knowledge about the supply in the marketplace at a given time. SAR is thoroughly discussed in the next section of this paper.

3 Service Adaptation Recommender (SAR)

The objective of SAR is to suggest service administrators, changes (adaptations) of their services' configurations, composition or workflows, in order to overcome problems or achieve higher performance. Based on recognized situations, SAR will be able to define adaptation pointcuts (points in a service flow that need to be adapted as a reaction to a certain situation) and advices (what to adapt and how based on a number of service adaptation strategies). In this section, we present technical requirements that apply for Service Adaptation Recommender (SAR) software component along with its initial conceptual view (figures 2 and 3).

One of the basic capabilities SAR feature is the situation awareness and detection functionality. This corresponds to the ability to sense situations relevant to service objectives and operation and the ability to track transitions between situations, by processing events and contextual service information. We consider complex events, detected in real time, as a way to signify situations that may require adaptation and we plan to enrich them with contextual information for defining dynamically what modifications are needed. Based on situational awareness module, SAR will be able to make intelligent recommendations for service adaptations (figure 2). It will improve service performance by detecting problems that need to be resolved (e.g. underperforming services, or suboptimal service workflows for the given situation) and by providing adaptation advices to be implemented in the appropriate workflow places (e.g. service tasks that need reordering, alteration or substitution etc.).

We have defined some general requirements that apply here. Firstly, it is important that SAR will be able to register for simple events as well as for complex events that carry combined information (e.g. the last 20 minutes, there is a 5% radiation increase). Both simple and complex events need to carry semantics that will allow further processing and reasoning. The federated middleware of our platform must also guarantee that SAR will receive all events that they have been subscribed to. Failure to deliver an event to SAR may lead to the non-detection of a new situation. For reasoning purposes domain knowledge is needed, which may be comprised of an event ontology, a context and a situation model. Scalability is another general requirement that applies to SAR, as the ability to cope with and extract valuable information from a "burst" of events, is imperative for detecting interesting situations. Finally, since our platform will be federated, it is important to detect global situations across the several service busses (i.e. Distribute Service Bus).

Since, SAR will be the dedicated software component to suggest changes (adaptations) of services' configurations, it needs to leverage domain knowledge for processing contextual information (e.g. preferences) of services and for analyzing service composition information (Fig. 2 - "Service Analyzer"). SAR also needs the ability to comprehend semantics carried by events, reasoning event semantics with service information, in order to detect relevant situations or trigger situational transitions (Fig. 2 - "Situation Awareness Module"). By acquiring all of the above it will detect and define appropriate service adaptation solutions as timely answers to situations that dictate reactions (Fig. 2 – "Adaptation Recommender"). Regarding DSB, SAR needs subscription/unsubscription capabilities from event sources, in response to situation transitions, as well as query capability to event storage for past events and historical service data in order to identify similar past cases ("Collaborative Filter").

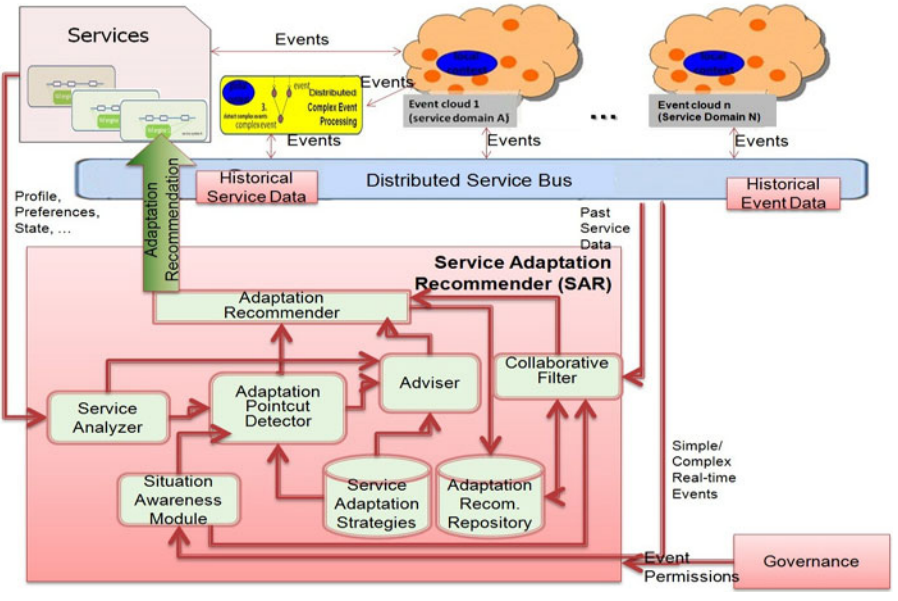

Fig. 2. SAR: Conceptual View

Service Adaptation Recommender (SAR) is considered as a core part of our platform and aims to provide recommender services for reacting at the right time and in the right way. Service Adaptation Recommender (SAR) will suggest changes of services configurations in order to overcome problems or improve service performance by:

- retrieving and analyzing service preferences/context ("Service Analyzer")
- analyzing service composition information ("Service Analyzer")
- deducing the situational model for the service using its semantics, composition, and preferences ("Situation Awareness Module")
- defining where in process flow we might need an adaptation ("Adaptation Pointcut Detector")
- evaluating and defining several alternative solutions of adaptation based on service adaptation strategies ("Adviser")
- identifying and recommending and storing service adaptations as reaction to a current situation ("Adaptation Recommender")
- exploiting collective intelligence by deriving adaptation recommendations based on previous ones in similar situations ("Collaborative Filter")

Service Adaptation Recommender (SAR) will be Java based software component that will depend on the events acquired from our federated platform along with their semantics and the contextual information gathered from the related services. SAR's recommendations may include: removal of a problematic service, replacement of an underperforming service, addition of a new service, alteration of the process flow. Specifically, based on recognized situations, SAR will be able to define adaptation

pointcuts (points in a service flow that need to be adapted) and advices (what to adapt and how).

In figure 3 a number of provided and required interfaces are given for SAR. The provided interfaces include setting and/or retrieving permissions to resources (events) for each service, receiving events from Event Cloud (this is the callback interface for Event Cloud) and sending adaptation recommendations to service administrator. The required interfaces include subscribing to / unsubscribing from events from the Event Cloud, querying Event Cloud for past / stored events and retrieving service preferences.

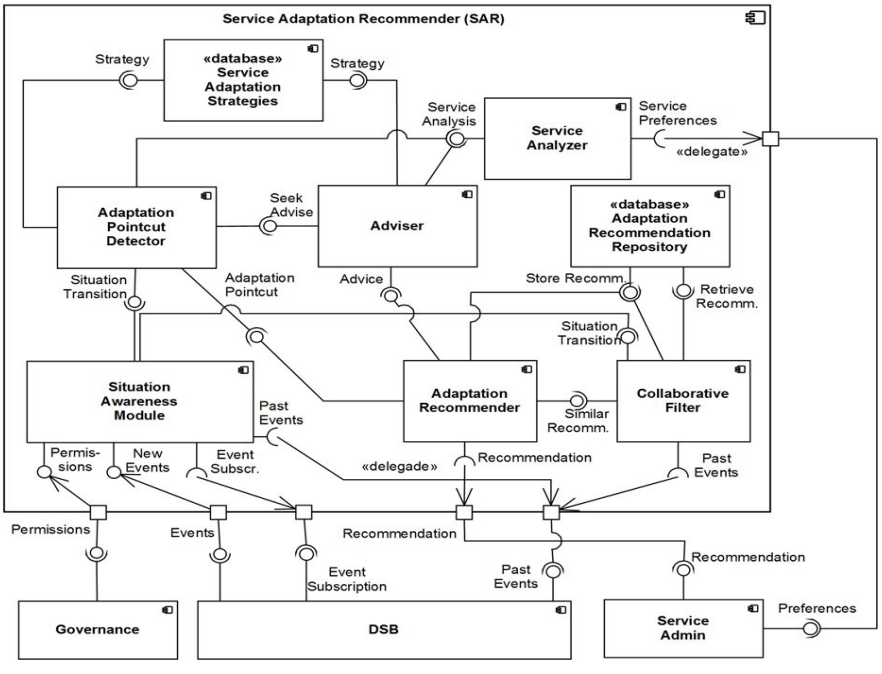

Fig. 3. SAR's Component Diagram

4 Related Work

Recently, there has been a significant paradigm shift towards real-time computing. Previously, queries against databases and data warehouses were concerned with looking at what happened in the past. On the other hand, complex event processing (CEP) is concerned with processing real-time events, i.e., CEP is concerned with what has just happened or what is about to happen in the future. Complex event processing is a very active field of research and is being approached from many angles [6]. Distributed complex event processing approaches circumvent resource limitations by taking advantage of pre-existing (shared) or dedicated network infrastructure. The Padres system [7] is such a distributed approach relying on a pre-existing network of

brokers. A broker may be an end-point for publishers (event sources) and subscribers (event sinks) to access the network. More importantly it is the task of the brokers to match events to the available subscriptions. The set of all interconnected brokers forms an overlay network across the underlying infrastructure. A drawback of Padres is the event pattern language used in its subscriptions. It is limited to handle only key-value maps for events and the set of event operators is very limited, for example, temporal relationships between events cannot be expressed declaratively but must be expressed on a timestamp-arithmetical level.

S-Cube [8] as the most prominent Network of Excellence in service adaptation, points out the evolution and adaptation methods and tools as keys to enable service-based applications (SBAs). Following the S-Cube's terminology, the term adaptation refers to the modification of a specific instance of a system during run-time (e.g. re-execution of a unavailable service or a substitution of a unsuitable service). Nowadays, several efforts that try to cope exactly with this issue point to Aspect-Oriented Programming (AOP), as a novel way to weave alternative actions in business processes at run-time. AOP has been proposed as a technique for improving the separation of concerns in software systems and for adding crosscutting functionalities without changing the business logic of the software. One of the most recognizable approaches for service adaptations using AOP is the work in [9] where the AO4BPEL is introduced. AO4BPEL is an XML-based language that creates a wrapper around the BPEL and has the ability to weave aspects at runtime to business processes. Aspects consist of one or several pointcuts and advices. AO4BPEL is based on XPath [10], which is used to select activity join points (i.e., points corresponding to the execution of activities) and internal join points (i.e., points inside the execution of activities such as the point where the outgoing message of an invoke activity is generated). An advice is the new behaviour to be included at a join point and contains the new code to be executed. SAR will use and extend such AOP based approaches for semantically and dynamically weaving changes in service based systems.

5 Conclusions and Future Research

We presented a novel approach for large scale, context-driven and quality-aware distributed event processing and we focused in the conceptual view of a service adaptation recommender for coping with the challenges of rapidly changing distributed service based systems. We are currently working on implementing the main functionalities of SAR, presented in this paper. The backbone mechanism that will support the core of SAR component will be based on a new modelling and execution framework for situation-aware applications. This framework will be built around the notion of goal-driven and hierarchical Situation-Action-Networks (SANs) which will provide modelling primitives for goals, situations, context, actions and loose mappings of abstract situations to generic action pools. This loose mapping at design time will allow for real situation-driven service adaptation recommendations for SBAs, at run time. The work in SANs actually extends well known research efforts in the planning domain (e.g. Hierarchical Task Networks [11], Behavioural trees [ref], etc.)

This research has been performed in the scope of a research project the vision of which is to develop and validate an elastic and reliable architecture for dynamic and complex, event-driven interaction in large highly distributed and heterogeneous service systems. Such architecture will enable ubiquitous exchange of information between heterogeneous services, providing the possibilities to adapt and personalize their execution, resulting in the so-called situational-driven adaptivity. We plan to test and validate our platform and specially its situational driven reaction capabilities, in a nuclear crisis management scenario and in a smart taxi service system. Both these use cases are characterized by dispersed event sources and are considered to create dynamically changing environments that dictate for distributed event processing and service adaptations.

Acknowledgements. This work has been partially funded by the European Commission under FP7 project PLAY. The authors would like to thank the project team for comments and suggestions.

References

1. Martin, E.A.: The Oxford Dictionary of Science, 6th edn. Oxford University Press (2010); ISBN-13: 9780199561469
2. Williams, G.C.: Adaptation and natural selection: a critique of some current evolutionary thought. Princeton Univ. Press, Princeton (1966)
3. Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web Services Agreement Specification (WS-Agreement) (2004), http://www.gridforum.org/Meetings/GGF11/Documents/draft-ggf-graap-agreement.pdf
4. Petals Master SOA Governance Solution, http://petalsmaster.ow2.org/
5. Allen, J.F.: Maintaining knowledge about temporal intervals. Commun. ACM 26(11), 832–843 (1983)
6. Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
7. Fidler, E., Jacobsen, H.-A., Li, G., Mankovski, S.: The padres distributed publish/subscribe system. In: 8th International Conference on Feature Interactions in Telecommunications and Software Systems, pp. 12–30 (2005)
8. S-Cube, http://www.s-cube-network.eu/
9. Charfi, A., Mezini, M.: Ao4bpel: An aspect-oriented extension to bpel. World Wide Web 10(3), 309–344 (2007)
10. XML Path Language (XPath), http://www.w3.org/TR/xpath/
11. Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Muñoz-Avila, H., Murdock, J.W., Wu, D., Yaman, F.: Applications of SHOP and SHOP2. IEEE Intelligent Systems 20(2) (2005)
12. Lim, C.U., Baumgarten, R., Colton, S.: Evolving Behaviour Trees for the Commercial Game DEFCON. Applications of Evolutionary Computation, 100–110 (2010)