Describing and Verifying Monitoring Capabilities for SLA-Driven Service Based Systems

Marco Comuzzi and George Spanoudakis
Department of Computing, City University London
Northampton Square, EC1V 0HB London, UK
{sbbd286, G.Spanoudakis}@soi.city.ac.uk

Abstract. Monitoring the operation of Service Based Systems (SBS) to ensure compliance with a set of service level agreements (SLAs) for example cannot always rely on a pre-specified monitoring infrastructure, where all the information and components required for monitoring are a priori known and available. This because new services with unknown monitoring infrastructures and capabilities may be dynamically assembled to an SBS. To address the need for dynamic configuration of SBS monitoring infrastructures, this paper proposes a model for describing the monitoring capabilities of different services of an SBS and discusses the process for verifying the monitorability of required properties based on these capabilities.

1 Introduction

Service Based Systems (SBSs), i.e., systems created by orchestrating loosely coupled autonomous and remotely deployable services, represent a typical kind of dynamic software systems for which runtime monitoring is required in addition to static verification and testing [1,2]. Monitoring is needed to ensure that the terms under which services are meant to operate at runtime (aka *Service Level Agreement* terms) are preserved. The need for establishing clear and machine-readable SLAs between service providers and consumers has been widely recognised in industry and academia as well as the need for monitoring mechanisms. Existing monitoring mechanisms either require the instrumentation of the orchestration process of an SBS or its services to perform monitoring (intrusive monitoring [2]) or the existence of mechanisms for capturing runtime information about service execution (e.g. service calls) which is passed to an external entity performing the monitoring (non-intrusive monitoring [1,3,4,5]).

Current intrusive and non intrusive monitoring mechanisms, however, assume a pre-defined infrastructure in which the SBS and the information that will be available from its components for monitoring are always known in advance. This assumption is not always valid, however, since the set of the services deployed by an SBS or even its orchestration process may change at runtime. If, for example, an SBS needs to monitor the response time of one of its constituent services S, it will require information regarding the timestamp of the calls of the operations of S and the timestamp of the responses of these operations. When S is dynamically substituted by another service, to continue monitoring the same response time property, the

monitoring infrastructure of the SBS will need to ensure that the substitute service of S, say S', has the ability to provide the required information. SBS will also need to initiate the provision of this information as soon as S' is bound to the SBS. Also, some of the existing monitoring approaches make even more restrictive assumptions, including, for example, the assumption that monitoring is performed solely on the basis of information collected from the side where the process that orchestrates the services of the SBS is executed (i.e., the *SBS orchestration process*).

To address this limitation, in this paper we describe an extension of the monitoring framework that was originally introduced in [3,5] which enables it to deal with monitoring scenarios where dynamic changes in an SBS affect the availability and provision of monitoring information and, therefore, the monitorability of the properties required from it. The key element of this extension is the introduction of a model enabling the expression of the *monitoring capabilities* of the services of an SBS. Based on the specification of the monitoring capabilities of the constituent services, the monitoring framework can assess whether the properties that need to be monitored for an SBS can indeed be checked and identifies the components (monitors and event captors) that will be used for monitoring. The check of the ability to monitor certain properties is based on a process that we also outline in the paper.

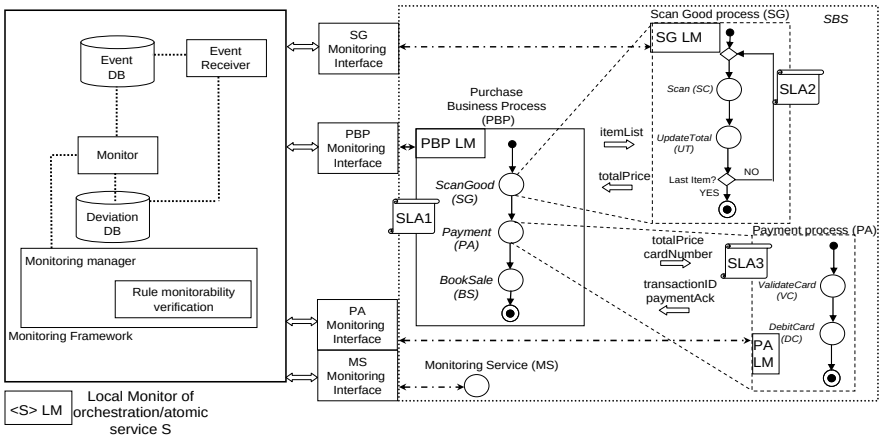

Fig. 1. The Monitoring Framework

A typical scenario for dynamic monitoring is reported in Fig. 1. The left side of the figure shows the runtime monitoring framework developed by Mahbub and Spanoudakis [3, 5] that we extend in this paper. The central component in this architecture is a reasoning engine, called *Monitor*. This engine checks whether the monitoring rules that have been specified for an SBS are satisfied against the events which are sent to the framework by its services. The framework provides a component, called *Event Receiver*, which is invoked by the services of an SBS to notify the events needed for monitoring. In the extended version of the framework that is presented in this paper, the monitoring of specific properties (rules) may be also delegated by the framework to local monitors which are attached to the services

of an SBS or a subset of them. Therefore, rule violations represent a particular type of event, which are produced by the SBS, received by the Event Receiver, and directly stored in the Deviation DB.

The right side of Fig. 1 shows an SBS and the architectural connection of the monitoring framework with the local monitors of the services in the SBS. The example is based on a retail solution for managing purchases in a supermarket. This solution is realized by a *Purchase Business Process* (PBP) that is implemented as the sequential orchestration of three different services, namely *ScanGoods* (SC), *Payment* (PA), and *BookSale* (BS). These services are possibly offered by different organizations or service providers and implemented as atomic services or workflows. The SBS also includes a monitoring service (MS). MS is not required to fulfill the requirements expressed by the users of the SBS, but it exposes operations used for monitoring purposes. In the example SBS of Fig. 1, we also assume that the provisioning of each orchestration or atomic service is regulated by a set of SLAs specifying functional and QoS properties of the SBS that should be guaranteed and monitored (service guarantee terms).

Using the above example, in the following we introduce the scheme for describing monitoring capabilities (Sect. 2), overview the process of checking monitorability in an SBS (Sect. 3) and present some concluding remarks about our ongoing work (Sect. 4).

2 Describing Monitoring Capabilities

A monitoring capability is composed of a set of *MonitoringNotifications* and *OperationTypeCapabilities*. Monitoring notifications express the ability of services to notify to external clients (monitors) runtime information concerning their execution. This information is expressed in the form of events. Operation type capabilities refer to operations that may be invoked by a monitor on a specific service for monitoring purposes. A monitor may, for instance, invoke operations in an external service to access specific monitoring-related computations (e.g. the computation of the statistical profile of the timestamps of service operation calls). The description of monitoring notifications can reference a *ServiceLandscapeModel*. This model defines in a precise and univocal way the elements of the SBS that the notifications and defined SLA properties may refer to. The structure of different types of monitoring notifications and operation type capabilities are discussed in the following.

A monitoring notification may be of two different subtypes: *MonitoringResultEventType* or *InteractionEventType*. A *MonitoringResultEventType* notification represents monitoring results which are generated by a monitor that is attached to a service in the SBS and are sent to external clients. *InteractionEventType* notifications are events captured at the interface of a workflow engine or a service container (e.g. service calls and responses with related I/O parameters).

An *Interaction Event* (IE) had the following form:

IE=(NotificationID, Sender, Source, Status, Operation, ID, Receiver, Timestamp)

where:

- *NotificationID* and *ID* uniquely identify the type of the notification and the specific event, respectively;
- *Sender, Source, and Receiver* identify the *End Point Reference* (EPR) of the monitoring interface that sent the notification event, the captor of the notification event (), and the receiver of the notification event (i.e., the monitor), respectively
- *Operation* specifies the signature of the operation to which the event refers;
- *Status* specifies whether the event refers to a service call or response, in terms of type (invocation or response) and input or output parameters; and
- *Timestamp* is the time when the event has been captured at the source.

A *MonitoringResultEvent* (MRE) has the following form:

MRE=(ID, NotificationID, Sender, Receiver, Agreement, GuaranteeTerm, Source, Timestamp)

where

- *ID*, *NotificationID*, *Sender*, *Receiver*, *Source* and *Timestamp* are as in the case of IE events;
- *Agreement* is one of the SLAs regulating the SBS provisioning to the user; and
- *GuaranteeTerm* is the guarantee term within the agreement that MRE refers to (i.e., the property that has been monitored).

3 The monitorability assessment process

In this section we briefly describe the process followed by the monitoring framework to assess the monitorability of functional and non-functional properties of an SBS, given the monitoring capabilities declared by its services and the SLAs established with its users. Initially, the SLAs established with the SBS users are deployed. The process of properties monitorability assessment must be executed before the start of SBS provisioning and it involves the external monitoring framework and the generic SBS services, i.e. orchestrations or atomic services.

The deployment of the SLAs triggers two parallel sub-processes executed by the monitoring framework (A) and at each SBS service (B), respectively.

- *Sub-process A*: Initially, the monitoring framework extracts the guarantee terms from the SLAs related to SBS provisioning and translates the guarantee terms of these SLAs into the local language used by the monitor to generate the *monitoring properties* (the monitoring engine described in [5] requires these properties to be expressed as Event Calculus (EC) monitoring rules – see examples in Table 1). Subsequently, the framework checks the generated monitoring properties against the statically defined capabilities of the SBS services, i.e., the Interaction event types. By matching the monitoring properties with the static monitoring capabilities, the monitoring framework can assess which of these properties (and, therefore, guarantee terms) it can directly monitor given the known event notifications and operation type capabilities.
- *Sub-process B*: The generic SBS service, via its local monitor, starts with extracting the terms from the SLA. The second step is then to translate such terms in the monitoring properties language adopted by the service local monitor. The

local monitor assesses the set of terms that can be locally monitored, i.e., for which the external monitoring framework can delegate monitoring, and dynamically produces the MonitoringResultEventType capabilities to be attached to the static capability document.

Examples of monitoring rules expressed in EC and descriptions of capabilities of the SBS elements are shown in Table 1. Rule R1 in the table expresses a functional property stating that a sale can be booked into the warehouse (interaction event type *ie1*) only after having received a positive confirmation (*paymentAck*) from the invocation of the payment service (*ie2*). Rule R2 shows how to exploit an operation type capability. In particular, the rule states that the size of the *transactionID* returned by the *validateCard* operation of the VC service in PA should always be less than 100kB. The size of *transactionID* is evaluated by the external monitoring service MS, whose invocation by the monitor is specified by assumption A4.

(R1)	Agreement:SLA1; GuaranteeTerm:property1 (∀ t1:time; paymentAck1, itemList:String) Happens(ie1, t1, R(t1, t1)) ⇒ (∃ t2:time) Happens(ie2, t2, R(0, t1)) ∧ paymentAck2 = OK ∧ paymentAck1= paymentAck2 ∧ transactionID1=transactionID2 ie1=(wID1, PBP, PBP:BS, REQ-B, bookSale(paymentAck1, itemList, transactionID1)) ie2=(wID2, PBP, PBP:PA, RES-A, commitPayment(paymentAck2, transactionID2))
(R2)	Agreement:SLA2, GuaranteeTerm:property2 (∀ t1:time) Happens(ie4, t1, R(t1,t1)) ⇒ call:MS:getSize(TransactionId) < 100 ie4=(wID4, PA, PA:VC, RES-A, validateCard(transactionID, totalPrice))
(P1)	Name: MonitoringCapabilityPBP, ID: P1, Sevice: PurchaseBusinessProcess InteractionEventType ie1=(wID1, PBP, PBP:BS, REQ-A, bookSale(paymentAck, itemList)) ie2=(wID2, PBP, PBP:PA, RES-B, commitPayment(paymentAck, transactionID))
(P2)	Name: MonitoringCapabilityPA, ID: P2, Service: PaymentProcess WorkflowEventType ie3=(wID3, PA, PA:VC, REQ-B, validateCard(cardNumber, TotalPrice)) ie4=(wID4, PA, PA:VC, RES-A, validateCard(transactionID, totalPrice)
(P3)	Name: MonitoringCapabilityMS, ID: P3, Service: MonitoringService OperationTypeCapability call:MS:getSize(var: VarTYpe):(transactionID)

Table 1 - Examples of monitoring rules (R/A) in Event Calculus and monitoring policies (P)

Table 1 also shows examples of monitoring capabilities exposed at the monitoring interface by the services PBP (P1), PA (P2), and MS (P3). The capability C1 reports the events required for monitoring rule R1 and can be exposed by PBP. C2 reports the capabilities of PA. These capabilities are required for monitoring the rule R2 (the event *ie4*, in particular). C3 specifies the capabilities of MS. More specifically, MS exposes only one operation type capability (i.e., the operation *getSize(Var: VarType)*) which, given a variable, can return its size (in kB). By comparing rules and capabilities of Table 1, the monitorability assessment process can establish that all the monitoring rules can be monitored according to the specified policies. Thus, in this particular case delegation would not be required.

Once the dynamic capabilities are generated, the monitoring infrastructure can identify the SLA guarantee terms whose monitoring can be delegated, and delegate their monitoring to local service monitors. In general as indicated in Fig. 1, for an SBS there will be three sets of guarantee terms: (i) the set *terms(SLA)* of all the terms which appear in the SLAs regulating the provisioning of the SBS, (ii) the set *terms(MON)* of the terms can be directly monitored by the monitoring infrastructure

through event notifications and operation type capabilities, and the set $terms_S(DEL)$ of the terms whose monitoring can be delegated to a local monitor associated with a service or orchestration S. Generally for an SBS consisting of NS services S_i associated to local monitors *terms(DEL)* will be obtained as $terms(DEL) = \cup_{i=1...NS} terms_{Si}(DEL)$. Also, terms(SLA) $\supset$ [terms(MON) U terms(DEL)], that is, some of the terms in the SLA may not be monitored on the basis of monitoring capabilities available at the SBS. The terms in the set terms(MON)\terms(DEL) can only be monitored by the external monitoring framework on the basis of events and operation type capabilities.

4 Conclusions and Future Work

In this paper, we have presented the extension of a framework for monitoring service based systems (SBS) regulated by SLAs whose aim is to support the dynamic assessment of the monitorability of the different terms in these SLAs by monitors associated with the orchestration process of the SBS or local services. This assessment is based on the monitoring capabilities that the individual services can provide and which can be described in a scheme that we have introduced for this purpose. Currently, we are focusing on the implementation of the proposed scheme and process for assessing rule monitorability as an extension of the EC-based runtime monitor of SBS presented in [3,5]. We are also working on the definition of a metric for *service monitorability*, based on monitoring capabilities, that could be used to characterise the QoS of different elements of an SBS.

Acknowledgements

The research reported in this paper has been supported by the EU Commission as part of the F7 Integrated Project *SLA@SOI* (grant agreement n. 216556).

References

1. F. Barbon, P. Traverso, M. Pistore, M. Trainotti, Run-Time Monitoring of Instances and Classes of Web Service Compositions, Proc. IEEE ICWS 2006.
2. L. Baresi and S. Guinea. Towards Dynamic Monitoring of WS-BPEL Processes, Proc. ICSOC 2005.
3. K.Mahbub, G. Spanoudakis. Monitoring WS Agreements: An Event Calculus Based Approach, Test and Analysis of Service Oriented Systems, (eds) L. Baresi, E. di Nitto, Springer Verlag, 2007.
4. O. Moser, F. Rosenberg, and S. Dustdar, Non-intrusive monitoring and service adaptation for WS-BPEL, Proc. WWW 2008.
5. G.Spanoudakis, K. Mahbub. Non Intrusive Monitoring of Service Based Systems, International Journal of Cooperative Information Systems, 15 (3), pp. 325-358, 2006.