A MANAGEABILITY INFRASTRUCTURE FOR THE MONITORING OF WEB SERVICE COMPOSITIONS

Christof Momm[1], Christian Mayerl[1], Christoph Rathfelder[1], Sebastian Abeck[1]

[1] Universität Karlsruhe (TH), Institute of Telematics, C&M IT Research, Zirkel 2, 76131 Karlsruhe, Germany
{momm, mayerl, crathfel, abeck}@cm-tm.uka.de

Abstract. The management of web service composition, where the employed atomic web services as well as the compositions themselves are offered on basis of Service Level Agreements (SLA), implies new requirements for the management infrastructure. In this paper we introduce the conceptual design and implementation for a standard-based and flexible manageability infrastructure offering comprehensive management information for an SLA-driven management of web service compositions. Our solution thereby is based on well-understood methodologies and standards from the area of application and web service management, in particular the WBEM standards.

Keywords: Web Service Composition Management; Manageability Design; Instrumentation; WBEM; CIM

1 Introduction

Today, companies require IT support that is tightly aligned with their business processes and highly adaptive in case of changes. These requirements can be met by employing a Service-Oriented Architecture (SOA) [1]. In SOA, functionality required for business processes is provided by atomic web services (WS) or by web service compositions (WSC), which implement fully automated and reusable parts of business processes [2]. Each service is characterized by the fact that it is operated by a service provider and the terms of use are contractually fixed by means of Service Level Agreements (SLA). Thereby, several independent providers may exist.
For an SLA-driven service provisioning, a provider needs an appropriate SLA management framework [3]. As a prerequisite, the operated WS and WSC together with their implementing components must be manageable, meaning they facilitate manageability capabilities for monitoring and controlling non-functional properties of services. The capabilities are offered through a harmonized manageability interface. This paper focuses on manageability capabilities for the monitoring of WSCs, which generally impose very specific requirements on the management infrastructure [4]

Following [5], the design of manageability capabilities for WSCs comprises three steps. First, the required management information has to be identified and formalized as a management information model. Then, the respective manageability interface is derived. This step is followed by the conceptual design of the manageability infrastructure that effectively realizes the manageability interface. More precisely, the infrastructure comprises management agents that account for retrieving the management information specified by the information model. This is accomplished by communicating with an appropriate instrumentation, which represents a non-functional extension to the implementation that offers a light-weight interface to the management agents, which allows for the retrieval of the relevant information. In accordance to this approach, in this paper we address the following questions:

- Which management information specific to WSC are required?
- How must a manageability interface be designed to meet these specific requirements?
- How should the implementing manageability infrastructure for WSC be designed?
- How can a WSC be instrumented in an adequate way?

The paper is structured as follows: Section 2 introduces the requirements for a WSC manageability interface by means of a motivating example. Section 3 presents the design of the manageability infrastructure based on WBEM, including the information model and the required instrumentation. Section 4 provides implementation experiences of the design. An evaluation of different instrumentation variants in regard of the produced management overhead is provided in section 5. A comparison of our approach in contrast to related research works is given in section 6.

2 Requirements for a WSC Manageability Interface

In this section, we illustrate some requirements for a WSC manageability interface in regard of the relevant management information. The requirements are motivated following a sample WSC taken from a scenario within the field of higher education.

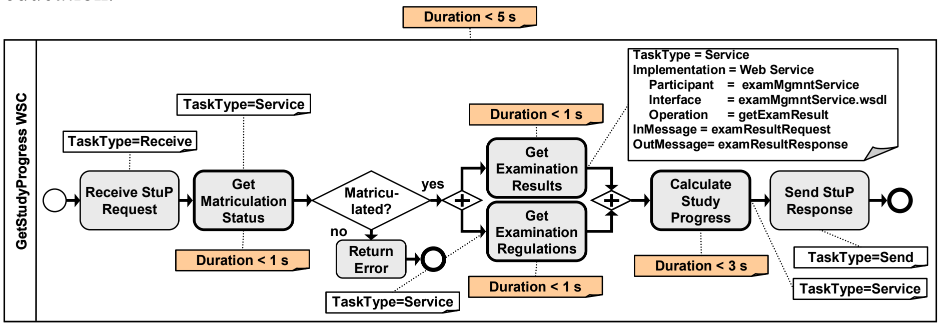

Fig. 1. Sample WSC with Quality Constraints

The WSC depicted on **Fig. 1** implements a small fully executable process, which provides students with information about the progress of their studies. For this purpose, several existing WS are employed within several service tasks. According to the WSC definition, after a request has been received, the matriculation status is retrieved through a service task. If the student is matriculated, his examination results as well as the associated examination regulation are collected. These documents are then both passed on to a WS that calculates the actual current study progress. Finally, the study progress is returned to the requesting student.

As this WSC is provided through a web portal, the service Duration of a single WSC instance must be less than 5 seconds. It becomes apparent, that this overall response time directly depends on the execution or processing times of the included WS. These are manifested in the Duration of a service task execution. Note that due to communication and engine processing overhead the service task execution time might be significantly higher. So the WSC management information should **(a)** include a sufficient management abstraction of the internal WSC logic [6]. This is important when it comes to the analysis of an SLA violation, which in our case would occur if the Duration of the WSC was greater than 5 seconds. To determine the exact reason of such a violation the provider needs information about the durations of the single service tasks. Unfortunately, at this point it is still not clear, whether the included WS or for instance the composition engine caused the violation. In consequence, **(b)** the provider needs information about the actual execution time of the included WS. This information somehow has to be integrated. **(c)** It has to be possible to retrieve management information about single executed instances of a WSC or WS respectively. This allows a flexible derivation of SLA-related metrics and the analysis/prediction of SLA violations can be performed in a more accurate way. Also, this feature represents an essential requirement for a business activity monitoring [7]. Another strongly desired feature of SLA management solutions is the automated and proactive initiation of adequate countermeasures in a timely manner, for instance in case of (predicted) SLA violations. For this purpose, the manageability interface **(d)** has to facilitate an active provision of management events. The same holds for information that is intended for a "real-time" business activity monitoring.

3 Design of a Manageability Infrastructure

This section introduces the design of the WSC manageability infrastructure, which we decided to build on the set of management and internet technologies proposed by the Web-Based Enterprise Management (WBEM) standard for enterprise computing environments. The reasons for that are:

- As this standard is already widely used for the management of distributed applications a seamless integration of our solution into existing management environments is rendered possible.
- WBEM has been recently been extended by the WS-Management standard [8], which allows a management through web services and therefore provides a standardized access to the management information
- The Common Information Model (CIM) [9], as the core of WBEM, allows the modeling of managed elements for the internal composition logic, single instances and the relations between the various managed elements

An apparent option might be the employment of the Web Services Distributed Management (WSDM) standard [10]. But as WSDM does not include an information model, it could only be used as a substitute for WS-Management. A mapping from CIM to WSDM is proposed in [11]. At this time, it is hard to come to decision, which one would be better. However, the design presented in the following would work with both standards.

3.1 WSC Information Model

The first step in designing manageability capabilities represents the specification of a management information model. As illustrated before, a WSC consists of different WSC elements, e.g. service tasks or gateways. Additionally a service task uses a WS. Therefore, the information model includes managed elements (ME) for describing the WSC as a whole, the different internal WSC elements and the included WS. For each ME, we distinguish between information about each executed WSC instance and information related to the general definition of the WSC, like configuration settings.
In the following we will show how these concepts are specified in CIM. Thereby, we faced the problem that CIM so far does not support the specific semantics of WSCs. Nevertheless, the CIM metrics model [12] offers a suitable foundation for modeling the required management information. More precisely, the UnitOfWork concept serves well as the basis for all execution-related WSC MEs. The definitional MEs on the other hand can be modeled by means of a corresponding UnitOfWorkDefiniton. The hierarchical internal structure of compositions can be represented through the SubUnitOfWork association. **Fig. 2** shows the complete WSC information model based on these concepts.

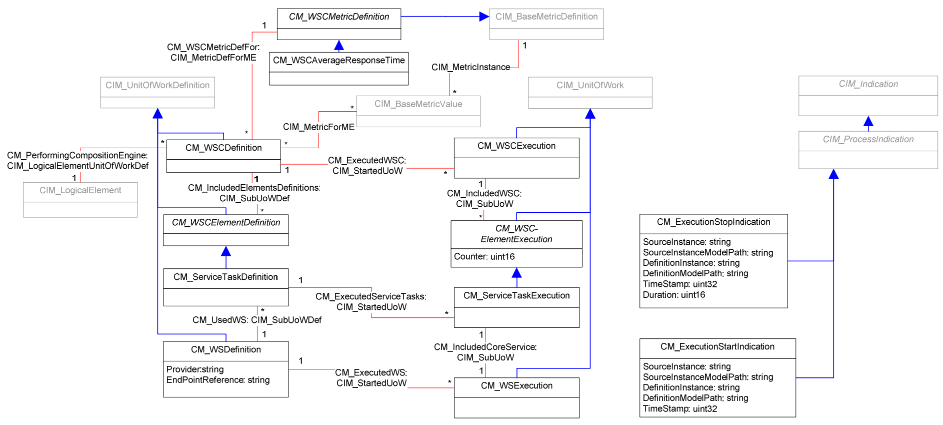

Fig. 2. CIM-based WSC Information Model

All CIM classes corresponding to the execution of the WSC or embedded WSC elements inherit from CIM_UnitOfWork, whereas the corresponding configuration-related classes inherit from CIM_UnitOfWorkDefinition. The ME for a WSC element generally represented by the abstract class CM_WSCElementDefinition or CM_WSCElementExecution respectively. Our information model introduces classes for a service task as one example for a concrete WSC element. The service task ME is associated with a WS ME providing information about the employed WS. The corresponding definition class (CM_WSDefinition) may for instance be used for changing the provider or endpoint reference at runtime whereas the execution class (CM_WSExecution) provides information about the particular WS processing time. The relationships between the definition classes and the execution classes are modeled through custom associations inherited from CIM_StartUnitOfWork.

The association CM_WSCMetricDef can be used to extend the CM_WSCDefinition by different metrics described through the abstract class CM_WSCMetricDefinition derived from CIM_BaseMetricDefinition, e.g. CM_WSCAverageResponseTime. The calculated value of the metric is represented by the class CIM_BaseMetricValue.

To integrate management information about the engine that executes the WSC, we define the association CM_PerformingCompositionEngine extending the class CIM_LogicalElementUnitOfWorkDef. A CM_WSCDefinition may be linked with an ME for the composition execution engine, as one example for a (vertical) integration of the underlying application management.

So far, this information model does not cover actively provided indications about MEs. Unfortunately, the definition of such indications is not well supported by CIM. More precisely, there is no suitable class in CIM for defining the conditions that lead to the triggering of an indication [13]. Nevertheless, it is possible to define the indications required for WSCs themselves on basis of the predefined CIM_ProcessIndication. For demonstrating our approach, we introduce two basic indications signaling the start and completion of an arbitrary execution element

3.2 General WSC Instrumentation Alternatives

The presented WSC information model introduces the managed elements for a WSC. Now, a suitable approach for gathering the necessary runtime information (i.e. instrumentation) is needed. In the following we will discuss several approaches for solving this problem.

A very common approach is the employment of the compositions engines' audit trail, which holds the complete execution data of a composition instance. However, neither a standardized audit data specification nor a standardized interface exists. Also, we observed problems concerning the appropriateness of the information available from the audit trail, for instance in terms of timeliness. In consequence, we argue against using the engines audit trail interface and propose to rather extend the compositions by adequate sensors, as also promoted by [14]. These sensors actively deliver the required management information to the responsible management agent. Thus, this approach is also referred to as an active instrumentation [15]. The monitoring of a WSC element of type "ServiceTask" for instance would require two additional sensors, one before and one after the respective activity [16]. Unfortunately, the additionally required sensors cause a management overhead that must not be ignored. As an example, if each service task should be monitored this would result in a triplication of the overall required service tasks. Hence, this aspect should be given a careful consideration when specifying the management requirements.

3.3 WSC Manageability Infrastructure Components

Having defined the WSC information model and discussed general concepts for the WSC instrumentation, we now present the design of the according WSC manageability infrastructure allowing a uniform access to the management information (**Fig. 3**). As already pointed up, the design is based on the principles proposed by the WBEM standards. Hence, the required management agents are realized as WBEM-compliant providers, as far as applicable. Thereby, we distinguish several different types of providers.

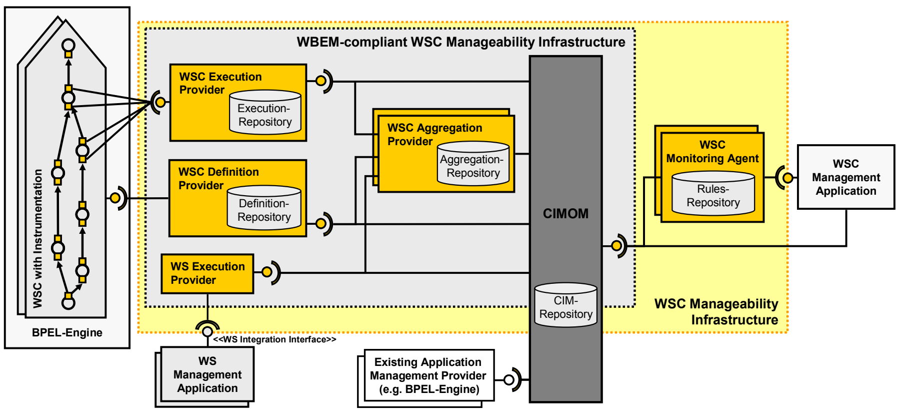

Fig. 3. WBEM-based WSC Manageability Infrastructure Design

The **basic providers** account for facilitating access to elementary MEs along with the associated indications. Elementary MEs only comprise parameters or indications which may be directly retrieved from the instrumentation. In order to make the MEs persistent, each basic provider possesses a management repository. This repository is of increased significance in case an active instrumentation based on sensors is used. Only then, the execution information is still available after the completion of the corresponding WSC element.

For the WSC manageability infrastructure we identified three basic providers. The WSC Execution Provider accounts for the provision of all execution-related MEs of the WSC information model along the respective associations and indications. As we decided on an active instrumentation it provides an interface for receiving the required management events from the WSC with Instrumentation. The basic WSC Definition Provider on the other hand is responsible for providing access to every definition MO, including the CM_WSDefinition. In case the reallocation of the included WS should be supported, a further WSC instrumentation, as indicated by the additional interface, would be required. However, the realization of such an interface is part of our future work. The integration of execution-related management information concerning included WS is handled by the WS Execution Provider, which uses a WS Integration Interface.

In addition, we introduce **aggregation provider**, which account for management metrics that can be derived from MEs already offered by existing basic or aggregation providers (e.g. mean values). They may also aggregate (basic) indications to more complex indications, for instance fault indications concerning the WSC elements to a general WSC fault indication. In our case, there is a WSC Aggregation Provider that handles the calculation of the metric holding the average response time of a WSC.

All provider components are made accessible through a CIM Object Manager (CIMOM), which may also integrate existing providers (like one for the composition engine). This would allow an integrated management of these resources.

For implementing management agents that account for more complex management functions, like the monitoring of critical SLA parameters, standard WBEM providers used so far are not suitable. This is because CIM does not allow the specification of reconfigurable rules for providers [13]. Therefore, we decided to place these kinds of management agents outside the WBEM-compliant manageability infrastructure. As one example, we introduce the WSC Monitoring Agent for the determination SLA violations concerning the execution of the WSC. This agent relies on rather complex, reconfigurable rules, which are stored in a local rules repository. Aggregation providers in contrast only require very simple rules that are not reconfigurable but rather hard-coded.

3.3 Integration of WS Management Information

The integration of management information about the included WS is accomplished by the IntegrationProvider, which requires a unique identifier of both the WS instance as well as the invoking WSC instance to associate the CoreServiceExecution with the corresponding ServiceTaskExecution element [17]. Thereby, the concrete WS instance identifier along with the required management information may not be determined until the service is actually requested. If the identifier of the invoking WSC instance is not available at that time, a subsequent correlation of the information is rendered impossible. So the WSC instance identifier has to be communicated within the service call. For the desired solution we identified the key requirement that the implementations of the employed WS must remain unchanged. Hence, we arrived at the conclusion, that the necessary WSC identifier should be included into the SOAP header of each relevant WS invocation. By attaching an appropriate message interceptor into the execution environment for the WS, the existing implementation does not have to be modified.

A comprehensive protocol for exchanging the measurements between two independent parties is specified in [17]. If the WSCs along with the included WS are operated by a single provider, a simple integration interface that offers access to the relevant measurements by specifying the instance identifiers is sufficient.

4 Implementation Experiences

In this section, we present the implementation of the previously sketched design. For the implementation of the managed resources, namely BPEL-based WSCs that include basic WS, we decided on the Oracle BPEL Process Manger (Oracle BPEL-PM) along with WS implementations based on Apache Tomcat/AXIS.

4.1 Active Instrumentation Alternatives for a WSC Management

As argued above, an active instrumentation of the WSC is desired. For the implementation of an active instrumentation several feasible options exist. One solution that complies with the existing BPEL standard represents the employment of standard service tasks for communicating with a management agent implemented as a WS. Unfortunately, SOAP calls are very expensive in regard of the communication overhead. The employed Oracle BPEL-PM provides several more efficient options. With the supported Web Service Invocation Framework (WSIF) for instance, an EJB instead of a SOAP binding may be used. In doing so, the BPEL processes remain standard-conform, but supposedly the communication overhead is reduced. However, not every BPEL engine supports the WSIF. Furthermore, Oracle-specific sensor activities may be used. Activity sensors for instance monitor the state of an activity and as one option send JMS messages in case of state changes. The sensors are defined within separate XML files, so the BPEL processes remain untouched. The usage of the latter solution is presumably the best option in regard of the additional management overhead. Unfortunately, this solution definitely is not applicable with other composition engines. To provide clarity of the actual management overhead caused by the different approaches, we implemented all of them and performed and evaluation, which is presented in section 5.

4.2 Java-based Manageability Infrastructure

Since the Oracle BPEL-PM runs on a JEE application server, we decided on using Enterprise Java Beans (EJB3) for implementing the core of our manageability infrastructure. For a CIMOM we employed the Java-based WBEMServices.

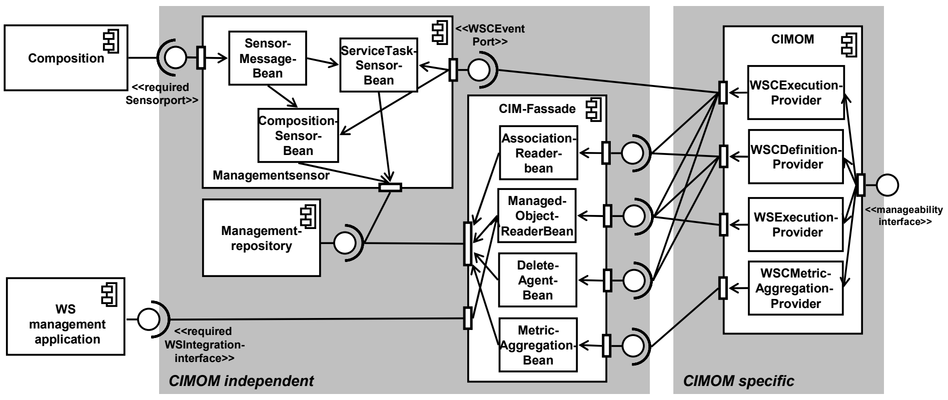

Fig. 4. EJB-based Implementation of the Manageability Infrastructure

As the interface between the CIMOM and associated CIM provider is not standardized, provider implemented for a specific CIMOM implementation typically cannot be used with other CIMOMs without modification [18]. Therefore, we draw a distinction between a CIMOM-specific and CIMOM-independent part.

The CIMOM specific part comprises three different provider classes that correspond to the basic providers introduced within section 3.3. They are completed by the WSCMetricAggregationProvider, which is responsible for providing aggregated metrics, in our case the average response time. These classes implement the specific interfaces defined by the WBEMServices framework and perform the required data mapping between the CIMOM independent part and the specific CIMOM.

The CIMOM independent part on the other hand is fully based on EJB. Entity beans are used for the different managed elements. They are subsumed by the ManagementRepository. The communication with the active instrumentation is handled by a message-driven bean and specific session beans that account for updating the managed elements and generating the respective CIM indication. Access to the managed elements (entity beans) is offered by the CIM-Facade, which comprises several session beans. The interfaces of these session beans match the standardized interfaces supported by a CIMOM. This allows an easy migration to another CIMOM implementation [18].

4.3 Integration of WS Management Information

The Oracle BPEL-PM so far does not allow the integration of the WSC instance identifier into the SOAP header. So we decided to use a proxy *WS'* for each included WS which is invoked instead. This proxy offers a modified interface, where each request message *R* is changed to *R'* by adding an additional message part holding the WSC instance identifier. The identifier in *R'* is set within the BPEL process definition by using standard BPEL activity "assign". The proxy *WS'* extracts the identifier from *R'* and places it in the SOAP header of the request message *R*. Within the WS each request and response is intercepted through a custom message handler. Thereby, the execution time is calculated and stored within a log file along with the WSC and the WS identifier. Accordingly, the WSIntegrationInterface operates on this log file.

5 Test and Evaluation

In this section we provide a qualitative evaluation for the different active instrumentation approaches explained in section 4.1 based on empirical performance measurements.

Our test scenario comprises an implementation of the WSC introduced in section 2 along with the four required WS. Each of the four included service tasks is extended by two sensors. So the instrumented BPEL process executes three times more service tasks than the original one. Each test setting was stressed with 1000 calls performed by 1, 5, 10 and 20 parallel threads. Between each call there is a fixed delay of 3 seconds. So the requests follow a uniform distribution. As a qualitative indicator we decided on the duration of the whole WSC measured in milliseconds.
The test scenario was deployed on an Oracle BPEL-PM on an AMD 64 3200+ with 2 GB RAM. The included WS are deployed on a separate system (Intel Core Duo 1.83 GHz, 2 GB RAM) using Apache Tomcat 5.0 and Axis 1.1. This system is also used for providing the WSC manageability infrastructure. **Fig. 5** summarizes the evaluation

results. For each instrumentation alternative the average duration in relation to the parallel threads as well as the standard deviation and the average management overhead are depicted.

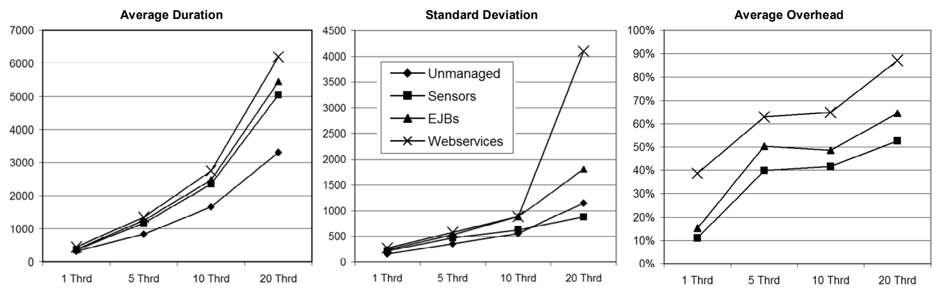

Fig. 5. Evaluation results for different instrumentation alternatives

As the results show, the management overhead regarding the duration in any case is rather high. The employment of Oracle-specific sensors thereby is the best option. An instrumentation purely based on standard web services on the other hand at least causes 30 percentage points more overhead than the sensors and with 20 threads requesting in parallel the standard deviation is unacceptable. The cause of this are most likely cueing effects in the BPEL engine in case of too many WS invocations The EJB-binding represents a good trade-off between the two options.

One explanation for the generally high overhead is that the included WS operations require very little processing times, because a small test database instead of the productive database was used. The sensors on the other hand per se require little processing time. If the response times of the employed WS are artificially extended to a realistic value, as also done in [18], the management overhead in terms of percentage in this case is halved. However, a further optimization is still desirable. For instance, we established a very fine-grained monitoring, which might not be necessary in all cases. A mechanism for dynamically adjusting the degree of monitoring, as presented in [19], would reduce the overhead significantly while the management requirements may still be met. Furthermore, it should be taken into account, that we examined a short-running BPEL process in our scenario. If long-running BPEL processes are monitored, the overhead in most cases becomes negligible.

6 Related Work

The presented manageability infrastructure is designed for being used within a SLA management infrastructure. In literature, two major frameworks for a SLA-based management of web services and web service compositions have been presented. These frameworks rely on an instrumentation of the managed resources and a manageability infrastructure. In the following, we'll discuss these approaches in comparison to our solution, which could be integrated into these frameworks.

In [3] an extensive framework for SLA-driven management on basis of Web Service Level Agreements (WSLA) is presented. However, this solution mainly focuses on the management of atomic WS and does not adequately support WSC. In [20] the approach is extended by a WBEM-based monitoring infrastructure. Thereby, we could particularly yield benefits from the presented information model on basis of CIM and CIM metrics. WSCs are supported by tracking the in- and outgoing message flow and employing event correlation, which does not allow a monitoring of internal the composition logic, like for instance gateways.

In [17] a competing framework that explicitly supports the SLA-based monitoring of composite services is presented. The manageability infrastructure is founded on an information model based on the ITU-T model. A monitoring of the internal composition logic is rendered possible. However, the solution represents a very proprietary approach as it relies on the Hewlett-Packard (HP) Process Manager and the manageability interface is not built on standards. This complicates the integration with other management applications.

An approach that strongly relates to our work is presented in [13]. This work so far is not grounded in one of the SLA management frameworks we introduced before. This contribution presents extensions to WBEM/CIM for an active monitoring of WSC, including a rule-based monitoring. However, the required manageability infrastructure is kept at a minimum as this aspect is not the major concern of this paper. For instance, the information model is limited to the CIM_UnitOfWork class. In summary, the specific semantics of WSCs and the resulting instrumentation issues are not fully regarded.

7 Conclusion & Outlook

In this paper, the conceptual design of an extensible manageability infrastructure tailored to the needs of an SLA-aware WSC management has been proposed. By building the solution on the widely-accepted WBEM standards a seamless integration of the underlying application, system and network management is now rendered possible. The specific WSC information model comprises all aspects required for a comprehensive SLA-driven management and by introducing the definitional MEs, it is applicable to all feasible kinds of functional WSC definitions. Finally, the support for an event-based monitoring enables a proactive management using autonomous agents. Also, the requirements for a business-related activity monitoring are met.

Concerning our current research, we are focusing on a methodology for an automated generation of the manageability infrastructure, in particular the instrumentation. In particular, we are working on the design of adequate meta- models that allow for a seamless integration of non-functional aspects into the development process of WSC. In this context, the realization of automated transformations operating on suitable UML profiles for describing the functional and non-functional behavior are of special concern. In addition, our management infrastructure is enhanced in order to support the monitoring of business process related key performance indicators (KPIs).

References

1. Ali Arsanjani: Service-Oriented Modeling and Architecture, IBM developer works, 2004.
2. Frank Leymann, Dieter Roller, M.-T. Schmidt: Web Services and business process management. In: IBM Systems Journal (2002) 41, S. 198-211, 2002.
3. A. Dan, D. Davis, R. Kearney, A. Keller, R. King, D. Kübler, H. Ludwig, M. Polan, M. Spreitzer, A. Youssef: Web services on demand: WSLA-driven automated management. IBM Systems Journal, Vol. 43 (1), 2004.
4. Tosic,V., Esfandiari, B.: Towards a System for Web Service Composition Management, In Proc. of the 2005 IEEE International Conference on Web Services (ICWS 2005), Orlando, USA, 2005.
5. O. Mehl, M. Becker, A. Köppel, P. Paul, D. Zimmermann, S. Abeck: A Management-aware Software Development Process Using Design Patterns, Integrated Network Management (IM2003), IFIP Conference Proceedings, Kluwer, 2003.
6. Christof Momm, Christoph Rathfelder, Sebastian Abeck: Towards a Manageability Infrastructure for a Management of Process-Based Service Compositions, C&M Research Report, Karlsruhe, 2006.
7. Jun-Jang Jeng, Josef Schiefer, and Henry Chang, An Agent-based Architecture for Analyzing Business Processes of Real-Time Enterprises, Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03), 2003.
8. Distributed Management Task Force (DMTF): WS-Management - CIM Binding, Version 1.0.0b, http://www.dmtf.org/standards/published_documents/DSP0227.pdf, 2006.
9. Distributed Management Task Force (DMTF): CIM Specification, Version 2.2, DMTF, http://www.dmtf.org/standards/cim/cim_spec_v22, 1999.
10. OASIS: Web Services Distributed Management (WSDM). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsdm
11. IBM: Proposal for a CIM mapping to WSDM, 2005. http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-wsdm/ws-wsdm.pdf
12. Distributed Management Task Force (DMTF): Metrics Model, DMTF http://www.dmtf.org/standards/documents/CIM/DSP0141.pdf, 2003.
13. Arun Kumar, Neeran Karnik, Ravindranath C.K.: Moving from Data Modeling to Process Modeling in CIM, IEEE 2005.
14. Carolyn McGregor, Josef Schiefer: A Web-Service based framework for analyzing and measuring business performance, Information Systems and e-Business Management, Springer-Verlag 2004.
15. Oliver Mehl: Modellgetriebene Entwicklung managementfähiger Anwendungssysteme, Universität Karlsruhe (TH), C&M (Prof. Abeck), 2006.
16. Christof Momm, Robert Malec, Sebastian Abeck: Towards a Model-driven Development of Monitored Processes, 8. Internationale Tagung Wirtschaftsinformatik (WI2007), Karlsruhe, Februar 2007.
17. Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A., Casati, F.: Automated SLA Monitoring for Web Services. In Proc. of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 2002), Montreal, Canada, Lecture Notes in Computer Science, No. 2506. Springer-Verlag (2002) 28-41, 2002.
18. Markus Debusmann, Marc Schmidt, Markus Schmid, Reinhold Kroeger: Unified Service Level Monitoring using CIM, Proceedings of the Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03)
19. L. Baresi, and S. Guinea. Towards Dynamic Monitoring of WS-BPEL Processes: In Proceedings of the 3rd International Conference on Service Oriented Computing, 2005.
20. Markus Debusmann, Alexander Keller: SLA-Driven Management of Distributed Systems Using the Common Information Model, Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management (IM 2003), 2003.