Service Intelligence and Service Science: Evolutionary Technologies and Challenges

Ho-fung Leung
*The Chinese University of Hong Kong, China*

Dickson K.W. Chiu
*Dickson Computer Systems, Hong Kong*

Patrick C.K. Hung
*University of Ontario Institute of Technology, Canada*
Chapter 9
Specification of Context for Management of Service-Oriented Systems with WS-Policy4MASC

Vladimir Tosic
NICTA, Australia & The University of Western Ontario, Canada & The University of New South Wales, Australia

Rasangi Pumudu Karunaratne
The University of New South Wales, Australia

Qinghua Lu
NICTA, Australia & The University of New South Wales, Australia

ABSTRACT

Specification of monitored context properties and their influence on operation of service-oriented systems and on management activities is a prerequisite for context-sensitive operation. We researched context specification for a management system performing various management activities and potentially used by mobile service-oriented systems. Due to the similarities between processing and use of context properties and processing and use of quality of service (QoS) metrics, we decided to model context properties analogously to QoS metrics. We built our solutions for specification of context properties and related management activities into two languages: the Web Service Offerings Language (WSOL) and WS-Policy4MASC, the latter of which is the focus of this book chapter. WS-Policy4MASC is a powerful extension of the industrial standard Web Services Policy Framework (WS-Policy) with constructs for specification of information necessary for run-time policy-driven management. The presented constructs related to context increase usefulness of WS-Policy4MASC for management of mobile service-oriented systems.

DOI: 10.4018/978-1-61520-819-7.ch009
INTRODUCTION

IT system management is the process of monitoring and control to ensure regular operation, maximize quality of service (QoS), discover and fix problems, accommodate change, account consumed resources, bill consumers, enforce security, and minimize operational costs. It is necessary to achieve dependable IT systems. Monitoring determines the state of a system, e.g., by measuring or calculating various QoS metrics, determining presence of faults, evaluating satisfaction of requirements and guarantees, and calculating monetary amounts to be paid. Here, QoS is a group of measures of how well (e.g., how quickly, how reliably) a system performs its operations. A QoS metric is a particular measure of QoS. Some examples of QoS metrics are response time, throughput, and availability. On the other hand, control puts the system into the desired state, by performing run-time adaptation (e.g., re-configuration) of a system to ensure its regular operation, in spite of external changes or internal run-time problems (e.g., faults, performance degradations). For example, control of a service-oriented system includes its re-configuration, re-negotiation of contracts between the composed services and between the system and other parties, and re-composition of services. Formal and precise specification of management information is necessary for successful management activities.

One frequent-approach to IT system management is based on policies. A policy formally specifies a collection of high-level, implementation-independent, operation and management goals and/or rules in a human-readable form. Policies are enacted during runtime by middleware that measures or calculates monitored information and executes control actions. A service level agreement (SLA) is another format for specification of this information. It is a special type of contract (a binding and enforceable formal agreement between two or more parties) that specifies QoS (and often price/penalty) information. It can be used as an alternative or a complement to policies. A class of service (a.k.a. service offering) is a predefined SLA that can be used by multiple consumers (i.e., it is not custom-made). While information specified in policies and SLAs is similar in content, SLAs require two or more parties (while policies can be specified for one party only) and, traditionally, architecture of management middleware is different.

Another issue relevant for our research is that the use of mobile service-oriented systems is rapidly increasing. In such systems, services and/or consumers execute in mobile devices, e.g., laptops, personal digital assistants (PDAs), or mobile/cell phones. While at first the term “mobile Web service” was used to denote systems where only consumers were mobile and provider services stationary, the number of mobile provider services is growing. Mobile service-oriented systems support ad hoc integration of diverse software running in mobile devices with other software running on the Internet, primarily through the use of Web service industrial standards. Example application areas are mobile business, fleet management (e.g., truck tracking), and disaster relief.

Management of mobile service-oriented systems has to deal with issues that are not very prominent in management of non-mobile systems. These specific management issues include:

i. context-sensitive operation and management;
ii. relatively frequent disturbances and changes of communication-level QoS;
iii. possibility of relatively frequent disconnection during execution; and
iv. limited resources (e.g., scarce run-time memory, relatively low processing power, limited battery lifetime, and slow wireless links).

In this book chapter, we explore the management issues related to context-sensitivity (i.e., the influence of context on operation and management) of service-oriented systems. There are very
different definitions of the term “context” in the literature and we will discuss this in the following section. For now, it suffices to say that we define context as information about external run-time circumstances (e.g., geographic location, events that come from outside, preferences of system’s consumers) that characterize the situation of the managed system and influence its operation, but are outside its direct control. Changes in external context (e.g., geographic location of mobile systems) significantly influence not only systems’ operation, but management activities. For example, when a truck with a truck-tracking mobile Web service enters USA from Canada, dynamic (run-time) adaptation should be automatically initiated to reconfigure the Web service to use the wireless network for the USA instead of the wireless network for Canada and to reconfigure the Web service operation returning speed of the truck so that results are provided in miles/hour (units used in the USA) instead of in kilometers/hour (units used in Canada).

To enable context-sensitive behavior of a mobile service-oriented system, a management system should support specifying, publishing, monitoring, storing, processing, analyzing, communicating, updating, and using context information. It should also enable using context for monitoring and control activities. All this requires specification of context management information that addresses at least the following four sets of questions:

1. How to formally specify monitored context properties (attributes) in a way useful for management activities?

2. How to formally specify when, where, how, for which parties (e.g., provider, consumer) and by which parties (e.g., provider, consumer, some independent/third party) the context monitoring is performed, as well as when and how values of context properties are transferred between various parties?

3. How to specify various ways in which context properties influence operation of service-oriented systems (particularly provider services), monitoring of service-oriented systems, and dynamic adaptation of service compositions?

4. How can specification of context properties be used for discovery and selection of appropriate service-oriented systems and their QoS?

The past solutions for management of service-oriented systems and for context-sensitivity of mobile service-oriented systems did not fully address these issues. Therefore, in this book chapter, we discuss how appropriate specification of context management information can be added to the existing languages used in management of service-oriented systems and their compositions. We present our conceptual solutions for modeling context and its impact on management activities in a general, language-independent way. These conceptual solutions can be added to different languages for management of service-oriented systems. We implemented an early version of these conceptual solutions as extensions to the Web Service Offerings Language (WSOL) and the corresponding Web Services Offerings Infrastructure (WSOI) management middleware. These implementations were summarized in (Tosic, Lutfiyya, & Tang, 2006a; Tosic, Lutfiyya, & Tang, 2006b). WSOL is based on classes of service and has a limited support for dynamic (run-time) adaptation. On the other hand, our new language WS-Policy4MASC (Tosic, Erradi, & Maheshwari, 2007; Tosic, 2010) is based on policies. As elaborated in (Tosic, Erradi, & Maheshwari, 2007; Tosic, 2010), the crucial advantage of WS-Policy4MASC is much stronger support for dynamic adaptation of service compositions, so there is a broader possible impact of context in WS-Policy4MASC. For a recent WS-Policy4MASC version 0.9 (Karunarathne, 2008), we adapted, extended and improved our conceptual solutions for modeling
context and related management activities. In this book chapter we will focus on explaining and illustrating our recent implementations in WS-Policy4MASC and not on our (preliminary) solutions in WSOL.

This book chapter is organized as follows. In the “Background” section, we discuss different definitions of the term “context” and survey the main related work. Then, in the section “Overview of the WS-Policy4MASC Policy Language”, we present the main concepts and characteristics of our WS-Policy4MASC language. In the section “Our Approach to Modeling Context”, we present and illustrate in the implementation-independent way the main concepts in our solution for modeling context for management of diverse service-oriented systems. The following section “WS-Policy4MASC Implementation of Context Modeling and Its Evaluation” discusses how we implemented the previously presented implementation-independent context modeling solutions in the WS-Policy4MASC language. The examples presented in this section are specific to WS-Policy4MASC. At the end of this section, we discuss how we evaluated (validated) our solutions. In the section “Future Trends” we outline possibilities for further research in this area, while in the section “Conclusion” we summarize the main contributions of our work. After the “References” section, we repeat the key definitions in the section “Key Terms and Definitions”.

**BACKGROUND**

In the mobile computing literature, there are several different definitions of the term “context”. One of the most often cited definitions states “Context is any information that can be used to characterize the situation of an entity.” (Dey, 2001, p. 5) Similarly, Chen and Kotz (2000, p. 3) provide the definition that “context is the set of environmental states and settings that either determines an application’s behavior or in which an application event occurs and is interesting to the user.” They identify four categories of context: computing context (e.g., network connectivity, nearby resources), user context (e.g., user’s profile), physical context (e.g., temperature), and time context. They also note the importance of context history. Furthermore, they differentiate between active context that determines behavior of the system and passive context that is relevant but not critical to the system. A somewhat narrower definition from Spanoudakis, Mahbub, and Zisman (2007, pp. 235) states that context is “any type of information regarding an SBS [service-based system] that can change dynamically”. The authors discuss that apart from information such as clock time, geographic location, physical environment, and presence of other processes that run on the same platform, this definition also includes information about SLAs that can change dynamically (i.e., during run-time). Moving more towards the SLA-type of information, Herssens, Faulkner, and Jureta (2008, p. 363) state that context is “any information about the interaction between users and a web service, for which an SLA is specified”. They identify five categories of context: user, provider, resource, environment, and Web service. From the session/transaction coordination viewpoint, Little (2007, pp. 441) explains context through the sentence “In order to correlate the work of multiple Web Services within the same activity, it is necessary to propagate additional information called the context to each participating service.” From the viewpoint of Web service composition, (Rong, Liu, & Liang, 2008) defines context as a collection of collaborating/partner (e.g., provider or consumer) Web services.

While such broad and diverging definitions of context are common, we believe that they are not completely appropriate and useful for the practice of IT system management. In the broadest definitions of context, such as (Dey, 2001), virtually any and every information that is relevant for IT system management can be viewed as context (e.g., because it can be used to characterize a situation
of the managed system). With such broad definitions, it is also almost impossible to answer the question: What is not context, but is still relevant for system’s operation or management? In the IT systems management literature, the broad term “management information” is already used to denote information that is relevant for IT system management. If all management information is context, then there is no benefit of introducing the new term “context”.

We studied many of the existing literature definitions and informal interpretations of the term “context” and concluded that they cover several groups of concepts that have considerably different nature and that must be differentiated for successful system management activities:

i. Situational circumstances that are external to the managed system and over which the managed system does not have direct control and, thus, is not responsible for. Some examples are: time/weekday/date, geographic location (if the managed system is carried and does not control its movement), competitors’ offers (e.g., whether a competitor offers a similar service at lower prices), external collaborating/partner (e.g., provider, consumer) services, properties and preferences of these partners (although the managed system might choose which partners to accept and limit their preferences to some options, it cannot directly modify these preferences), on-line presence of the partners.

ii. Historical circumstances in execution of the managed system over which the managed system had some control in the past and, thus, is at least partially responsible for. Some examples are: the current state of the managed system (it is a consequence of past actions by the managed system), session states that the managed system has with individual partners, and past values of QoS metrics that the managed system was responsible for.

iii. Current and future circumstances over which the managed system has some control and, thus, is at least partially responsible for. Some examples are: future values of QoS metrics for which the managed system is responsible, functional and QoS guarantees that the managed system promises (e.g., in SLAs or policies), prices that the managed system sets for its use, on-line presence of the managed system.

The facts whether (or not) the managed system is responsible for some circumstances and whether (or not) it can do anything about them in the future are crucial from the management viewpoint and directly determine the managed system’s behavior related to these circumstances. Also note that information about the concepts from the second and third group is already maintained in many management systems. Contrary, information about only a few concepts from the first group (e.g., about date/time) is used in the traditional management systems, while the rest is rarely maintained. Importance of some of the concepts from the first group (e.g., geographic location) is significantly higher in mobile systems compared to non-mobile systems.

Therefore, we decided to provide two definitions of the term “context” – one with a broad meaning and the other with a narrow meaning. In the definition with the broad meaning, context is (any and every) information that characterizes situation of the (managed) system. This definition is consistent with the broadest definitions, e.g., Dey (2001), that we found in the literature. On the other hand, our definition with the narrow meaning states that context is information about external run-time circumstances that characterize situation of the (managed) system and influence its operation (i.e., execution, behavior), but are outside its direct control. It includes only the first group of concepts from the above classification. This definition is consistent with the definition of context in linguistics: “the words before and
after a word or passage in a piece of writing that contribute to its meaning” (Farlex, 2009). Using this narrow definition, context of a service-oriented system does not include a description of the system’s implementation, input values provided by a consumer, changes to its state caused by execution, or values of QoS metrics. For example, current clock time and country of location are context for a mobile banking Web service, but account balance is not (it is part of Web service’s state). Similarly, response time of code that implements a Web service is not context (it is determined by internal processes), but network delay time for Web service messages over the Internet is context (it is external and not under direct control).

Online presence of external collaborators/partners is context, while on-line presence of the managed system is not context (the managed system can change it at least in some situations; if the managed system cannot change its on-line presence, then the underlying cause, such as lack of network connectivity, is context). In this book chapter, we will use the narrow definition of context and will explicitly note when we refer to the broad definition. We also define the term “context property” as an attribute (aspect) of context. In the above examples, current clock time, country of location, network delay time, presence of external collaborators/partners, and network connectivity are five different context properties.

Many recent research projects and papers are related to context modeling, processing, and/or management in mobile computing. They explored the concept of context in various circumstances and from various viewpoints. These related works (see Table 1) used various definitions of context, oftentimes with a different meaning compared to our narrow definition (but within our broad definition). We will review here only several past works that can be viewed as most closely related to our topic of context modeling for management of service-oriented systems or our narrow definition of the term “context”.

Probably the closest related work to our research is (Herssens, Faulkner, & Jureta, 2008). This paper presents an SLA management approach to ensure that after a change of context, SLAs are adapted automatically and with minimal human intervention (i.e., autonomically). These authors are aware of (and reference) some of our past publications on WSOL and WSOI. An issue with their approach is that SLA adaptation cannot be done at will, because there are many potential consistency issues. Although their approach tries

---

**Table 1. The meaning of the term “context” in various papers**

<table>
<thead>
<tr>
<th>Paper</th>
<th>External Situational Circumstances outside Direct Control</th>
<th>Past Circumstances under Some Control</th>
<th>Current &amp; Future Circumstances under Some Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>time</td>
<td>location</td>
<td>external partners</td>
</tr>
<tr>
<td>(Dey, 2001)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Herssens et al., 2008)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Little, 2007)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Amundsen &amp; Eliassen, 2008)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Spanoudakis et al., 2007)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Chen et al., 2006)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Rong et al., 2008)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our narrow definition</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
to capture dependencies among various management information items (including context properties and QoS metrics), it seems that they have treated these complicated dependencies in a somewhat simplistic manner. Furthermore, their definition of context is significantly different from our definition with the narrow meaning, although there are some overlaps. Our improved context modeling solutions in WS-Policy4MASC provides support for adaptation that is not present in related work, including (Herssens, Faulkner, & Jureta, 2008).

Several specialized formats for specification of context were developed (e.g., see Strang & Linhoff-Popien, 2004; Chen, Yang, & Zhang, 2006; Baldauf, Dustdar, & Rosenberg, 2007; Bolchini, Curino, Quintarelli, Schreiber, & Tanca, 2007), but they are different from the developed models for specification of other management information (e.g., QoS) and cannot be viewed as directly applicable in management systems for various management activities and various types of service-oriented systems. For example, context models and representations developed by the Semantic Web community, such as (Forstadius, Lassila, & Seppanen, 2005), are not directly useful for management because they do not describe in detail management activities to be performed and are not accompanied by appropriate management infrastructures. Another example is (Sheng & Benatallah, 2005), which describes Context-tUML – a UML extension (profile) for modeling context-aware Web services. Their model includes modeling of context properties (attributes), their retrieval from context sources, binding of context to context-aware objects, and triggering of adaptation after context changes. Thus, their model addresses many (albeit not all) aspects relevant for use of context in management systems. The problem is that this modeling is done in a way that is not similar to the modeling of other management information, so its integration into general management systems would be very difficult. In the section “Our Approach to Modeling Context”, we will explain that it is important to have a context model that is consistent with the models for specification of other management information.

(Amundsen & Eliassen, 2008) presents a model that combines resource QoS metrics and context properties, developed for the dynamic mobile middleware called QuAMobile. This model does not consider specifics of service-oriented systems and does not address all issues in context management. However, note the authors’ claim that design of self-managed mobile computing systems has to consider both context information and QoS metrics. This viewpoint agrees with our approach, which was first published in (Tosic, Lutfiyya, & Tang, 2006a). Our work deals with the specifics of management of service-oriented systems (e.g., adaptation of Web service compositions).

A somewhat different type of related work is (Wibisono, Zaslavsky, & Ling, 2008), which presents a context service framework to facilitate context management in mobile environments. Their work focuses on dealing with quality of context (the topic that our work does not address), but does not address the issues we focus on. We see compatibilities and possible integrations between our work and their work.

Little (2007) presents the Web Service Context (WS-Context) as a specification defining a shared context structure, which relates to activities between Web services. The title of this work makes it sound like a crucial related work to our research. However, their definition of context actually means “session/transaction history” and is, thus, substantially different from our definition of context with the narrow meaning. The author claims that WS-Context is the only Web service specification that describes a generic context (i.e., session/transaction) management mechanism. In fact, many other Web service technologies with session management were published previously, such as the early version of the WSOI management middleware described in (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2005).
Note that the main languages in management of service-oriented systems do not contain context-related concepts. Notable examples of these languages are the Web Service Level Agreement (WSLA) language (Keller & Ludwig, 2003), WS-Agreement (Ludwig, Dan, & Kearney), and the Web Services Policy Framework – WS-Policy (W3C Web Services Policy Working Group, 2007). As argued in (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2005), WSLA has higher run-time overhead than WSOL. Since run-time overhead is an important issue in mobile (and, particularly, embedded) systems, we have not proceeded with extension of WSLA. (Another argument against WSLA extension is that WSLA is no longer supported.) Since WS-Agreement is only a general framework for specification of contracts, without direct support for detailed specification of QoS-related information, we decided that its extension would be too complicated. As will be explained in the next section, our WS-Policy4MASC is an extension of WS-Policy (a general framework for specification of policies), which makes it attractive to the large number of current and potential future users of WS-Policy.

The general conclusion of our literature survey is that these past projects and papers did not completely explore context specification and management for mobile service-oriented systems from the viewpoint of a management system for various management activities and various types of service-oriented systems (including non-mobile). Similarly, a model and format for specifying context properties through extensions of existing management information models and formats for service-oriented systems would be beneficial, but was not developed prior to our research project.

OVERVIEW OF THE WS-POLICY4MASC LANGUAGE

WS-Policy4MASC (Tosic, Erradi, & Maheshwari, 2007) is a powerful language for formal specification of management policies for service-oriented systems. It has been used in the Manageable and Adaptable Service Compositions (MASC) middleware for management of service-oriented systems (particularly service compositions), as described in (Erradi, Tosic, & Maheshwari, 2007). The design and prototype implementation of the MASC middleware were based on the Microsoft .NET 3.5 platform. However, WS-Policy4MASC is a general-purpose language for description of management information and its use is not limited to the MASC middleware or the Microsoft.NET platform or even the orchestration type of Web service compositions. WS-Policy4MASC extends the Web Services Policy Framework – WS-Policy (W3C Web Services Policy Working Group, 2007), an industrial specification standardized by the World Wide Web Consortium (W3C).

WS-Policy is a general framework for specifying various Web service properties in a way that complements the widely-used industrial standards the Web Services Description Language (WSDL) and the Web Services Business Process Execution Language (WSBPEL). It defines an extensible container to hold domain-specific policy assertions. In the WS-Policy model, a policy is defined as a collection of policy alternatives, each of which is a collection of policy assertions. WS-PolicyAttachment defines a generic mechanism that associates a policy with subjects to which the policy applies, such as WSDL or WSBPEL constructs. Various policy subjects are possible, such as process, flow, links, service, endpoint, operation, message, or message part. A policy scope is a set of policy subjects to which a policy may apply. WS-Policy has many good features, such as simplicity, extensibility, and flexibility (e.g., policies can be specified inside and outside WSDL and WSBPEL files). Nevertheless, it must be noted that WS-Policy is only a general framework, while the details of the specification of particular categories of policies will be defined in specialized languages – domain-dependent extensions of WS-Policy. Currently, only standard
extensions for security, reliable messaging, and a few other management areas that were not the focus of our project had been published. It is not clear whether and when some standard specialized languages for the specification of QoS, context properties, business value metrics (at least prices and penalties), and other management information will be developed. Some unification and standardization of common elements (e.g., expressions) of various WS-Policy languages would reduce the overhead of supporting this framework. Further, WS-Policy does not detail where, when, and how policies are monitored and evaluated. Since many policies have to be monitored and controlled during run-time, WS-Policy needs better support for management applications, including explicit specification of such management information. Consequently, we had to develop a new domain-independent WS-Policy extension, which we named WS-Policy4MASC.

WS-Policy4MASC extends WS-Policy by defining new types of WS-Policy policy assertions. These are not domain-specific policy assertions because they can be used for representing functional constraints, QoS requirements and guarantees, adaptation (control) actions, security capabilities, business value metrics (e.g., prices), and other information. Compared to an a posteriori combination of several single-domain policy solutions, a true integration of different domains (as done in WS-Policy4MASC and a few older works, e.g., WSOL) leads to better specification of inter-domain dependencies, increased interoperability between domains, and lower total run-time overhead (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2005). WS-Policy4MASC mandates no changes to WS-Policy constructs, so they can be used with WS-Policy4MASC in the same way as for other WS-Policy policy assertions. Using standard WS-PolicyAttachment mechanisms, WS-Policy4MASC policy assertions can be attached to WSDL constructs describing individual Web services or their parts (e.g., service, endpoint, operation, message constructs) and/or WSBPEL or Microsoft’s Extensible Application Markup Language (XAML) constructs describing Web service compositions or their parts (e.g., process, sub-process, activity). Analogously to WS-Policy, the syntax of WS-Policy4MASC is based on the Extensible Markup Language (XML) and defined using XML Schema.

Figure 1 shows the main concepts and relationships in WS-Policy4MASC version 0.9 (Karunaratne, 2008). The main difference from WS-Policy4MASC version 0.8 (Tosic, Erradi, & Maheshwari, 2007) is the addition of probability policy assertions (and the support for specification of context, which is not shown in Figure 1, but is discussed later in this book chapter). WS-Policy4MASC constructs can be classified into “real” policy assertions and auxiliary constructs. The “real” policy assertions describe the operation and management goals and/or rules, while the auxiliary constructs specify additional details necessary for use of the “real” policy assertions (for technical reasons, the auxiliary constructs are specified as WS-Policy policy assertions, but cannot be used in management without the “real” policy assertions).

In WS-Policy4MASC version 0.9 there are 5 types of “real” policy assertions (all inheriting from the abstract construct “MASCPolicyAssertion”) relevant for run-time management (particularly, dynamic adaptation) of service-oriented systems:

1. Goal policy assertions specify requirements (e.g., 128 bit security must be used) and guarantees (e.g., response time of a particular activity has to be less than 1 second) to be met in desired normal operation. In the MASC middleware, they guide monitoring activities, primarily evaluation of various monitored conditions. Specification of Web service requirements/guarantees for monitoring activities was addressed by a number of past languages, such as WSLA (Keller & Ludwig, 2003) and WSOL.
While such a concept was necessary in WS-Policy4MASC to support monitoring activities, WS-Policy4MASC solutions in this area offer only minor advantages, summarized in (Tosic, Erradi, & Maheshwari, 2007).

2. Action policy assertions specify diverse actions to be taken if certain conditions are met (e.g., some goal policy assertions were not satisfied). For example, these actions can be removal, addition, replacement, skipping, or retrying of a sub-process (or individual activity) or process termination. In MASC, they guide adaptation and other control actions (and a few aspects of monitoring, such as monitored data transfer). One of the distinctive characteristics of WS-Policy4MASC, compared to the related work, is built-in support for a diverse range of common service composition and business process adaptation actions, presented in (Tosic, Erradi, & Maheshwari, 2007). These supported actions address various frequent adaptation needs, including (but not limited to) customization (e.g., versioning), corrective adaptation for fault management, and optimization for QoS (performance) and business value management. For example, assume that there is a known type of system faults that is relatively easy to detect with functional pre- or post-conditions and for which there is a known set of corrective actions (execution of these actions is the system’s dynamic adaptation). To address such situations with WS-Policy4MASC, these pre-/post-conditions are specified in a goal policy assertion and the corrective actions are specified in an action policy assertion. The triggering event for the action policy assertion is non-satisfaction of the goal policy assertion and the action policy assertion can specify additional filtering conditions that narrow possible causes to the target system fault. That is, when the pre-/post-conditions in the goal policy assertion are not met, an event is raised. The
raising of this event leads to evaluation of the filtering conditions. If they are satisfied, this indicates that the probable reason is the system fault we want to handle, so the corrective actions in the action policy assertion are executed. Simple performance problems are addressed in an analogous manner (in such cases, goal policy assertions describe limits of acceptable performance). More complicated situations (e.g., requiring correlation of events and/or complex causal chains of events and actions, possibly with uncertainty whether some situations will occur) can also be specified in WS-Policy4MASC.

3. Utility policy assertions specify expressions for calculating monetary amounts quantifying diverse business value metrics assigned to particular run-time situations (e.g., non-satisfaction of some goal policy assertion, execution of some action, another event). They can be used by MASC for accounting/billing and, indirectly, for selection between alternative action policy assertions. Specification of diverse (both financial and non-financial) business value metric types in utility policy assertions is one of the most important contributions of WS-Policy4MASC and is discussed in detail in (Tosic, 2010) and, to a lesser degree, in (Tosic, Erradi, & Maheshwari, 2007).

4. Probability policy assertions specify probabilities that particular situations will occur. As explained in (Tosic, 2010), they can be used for specification of risks and trust in various parties.

5. Meta-policy assertions specify which action policy assertions are mutually conflicting (i.e., their conditions are satisfied at the same time, but their actions must not be executed together) and which business value driven conflict resolution strategies should be used to decide which action policy assertion to execute. Meta-policy assertion specification of diverse business value maximization strategies for policy conflict resolution is another significant original contribution of WS-Policy4MASC. Meta-policy assertions guide business value driven management activities in MASC, which are elaborated in (Tosic, 2010) and, to a lesser degree, in (Tosic, Erradi, & Maheshwari, 2007). For example, (Tosic, 2010) presents an algorithm that calculates which of the conflicting action policy assertions (representing different dynamic adaptation options) leads to the highest business value (taking into consideration company’s business strategy that determines which business value metrics are relevant for the company).

In addition to these 5 new types of “real” policy assertions, WS-Policy4MASC enables specification of additional detailed information that is necessary for run-time policy-driven management (particularly, adaptation) of service-oriented systems and overcomes some other limitations of WS-Policy (e.g., imprecise semantics of policy assertions’ effects on policy subjects). Some of this information (e.g., which party performs evaluation/execution of a policy assertion, which party is responsible for meeting a goal) is specified in attributes of the above-mentioned “real” policy assertions. Much more information is specified in additional auxiliary WS-Policy4MASC constructs, specifying ontological meaning, monitored data items (e.g., monitored QoS metrics), states, state transitions, schedules, events, scopes, and various types of expression (Boolean, arithmetic, arithmetic-with-unit, string, time/date/duration). Many of these auxiliary constructs are not shown in Figure 1, to prevent overcrowding.

The most important among the auxiliary constructs is the “When” construct that specifies when something (e.g., evaluation of a goal policy assertion, execution of actions in an action policy assertion, or calculation and billing of monetary amounts specified in utility policy assertions) should happen in the MASC middleware. It
contains information about one or more states in which this occurs, one or more events (e.g., Web service operation executed) that can each independently trigger this occurrence, and an optional filtering Boolean condition to be satisfied. WS-Policy4MASC has built-in constructs for specification of a wide range of events (and, as mentioned above, adaptation actions) common in management of service-oriented systems and business processes they implement. Further, the language is extensible, so new types of events (and actions) can be defined easily.

The auxiliary construct “OntologicalMeaning” provides some support for specification of semantics of monitored data items. The actual definitions of ontological concepts are in external (reusable and extensible) ontologies. Thus, the construct “OntologicalMeaning” specifies the namespace of the used ontology and the name of the ontological concept within this ontology (e.g., “measurementOntology:Availability” or “currencyOntology:USDollar”), as well as identification of the language in which this ontology is defined. These external ontologies can be specified in any of the currently used ontology languages, such as OWL, but we provided a very simple ontology schema that is the default ontology language. If a management party does not understand the used ontology language, it can still perform simple syntax matching of namespaces and names.

While the MASC middleware (Erradi, Tosic, & Maheshwari, 2007) is based on the orchestration type of Web service compositions (where there is a central entity directing invocation of all other services), WS-Policy4MASC can also be used in choreographies (where there is no such central entity). A distinction should be made between private policies of individual choreography participants (or groups of orchestrated participants) and mutually agreed public policies that form a contract (SLA) between choreographed parties. Private policies for an independent participant are stored in the policy repositories of that participant, who then processes them internally. Private policies for a group of orchestrated participants are usually stored in the policy repository of their orchestrator (but more distributed arrangements are also possible). Contractual public policies between choreographed parties are stored in policy repositories of all relevant parties (and only those parties). This relevance is determined by the content of several attributes (such as “Management-Party”, “ResponsibleParty”, “BeneficiaryParty”, “PayingParty”, and several others) describing roles of particular parties. Events specified in contractual public policies have to be exchanged between all relevant parties in the contract. This can all be specified (explicitly or implicitly) with the current version of WS-Policy4MASC. Further, it is possible to specify control actions common in Web service choreographies (e.g., contract re-negotiation), because some of these actions (relevant also for orchestrations) are already built into the WS-Policy4MASC language grammar (schema) and because WS-Policy4MASC is extensible. However, because such WS-Policy4MASC extensions can be specified in different ways, interoperability would be improved if the most common missing actions are also standardized in the WS-Policy4MASC grammar. This is an item for future work.

It is important to note that WS-Policy4MASC addresses context in the broader sense of this term with a number of constructs. Out of the three categories of context identified earlier in this book chapter, WS-Policy4MASC has very strong support of the last two. Most importantly, many WS-Policy4MASC constructs specify current and future circumstances over which the managed system has control (i.e., the third group of context concepts from our classification). For example, definition of QoS metrics is done in the “MonitoredDataItem” construct, functional and QoS guarantees are specified in goal policy assertions, while prices are specified
in utility policy assertions. Related to this, there are special actions (“MonitoredDataCollection” and “MonitoredDataTransfer”) specifying monitoring and transfer of QoS metrics and special events (e.g., “MonitoredDataItemUpdated” for notification when values of QoS metrics change, “GoalPolicyAssertionSatisfied” for notification when the guarantees are met or not met). Furthermore, the historical circumstances in execution of the managed system (i.e., the second group of context concepts from our classification) can also be specified. In particular, the current state is specified in the “State” construct, while the filtering Boolean expression in the “When” construct can be used for description of how past values of QoS metrics influence which WS-Policy4MASC policy assertions are relevant in particular circumstances. Some of the support for this group of context concepts is in the MASC middleware, instead of the WS-Policy4MASC language. Notably, process instance identification (ID) is used in MASC for session/transaction management. Contrary to the last two groups of context constructs from our classification, there was not enough support in WS-Policy4MASC version 0.8 for the first group of concepts, i.e., the situational circumstances that are external to the managed system. This is the group of concepts that belongs to our narrow definition of context. The strongest WS-Policy4MASC version 0.8 support here is for time/date/duration expressions. While it is syntactically possible to specify external context properties (such as location) in “MonitoredDataItem” constructs, the semantics of this is not completely appropriate (as will be discussed later in this book chapter). Therefore, we decided to build into WS-Policy4MASC version 0.9 (Karunaratne, 2008) comprehensive support for context monitoring and context-sensitive service-oriented system management, which we will explain in the remainder of this book chapter.

**OUR APPROACH TO MODELING CONTEXT**

We have noted in the “Introduction” section that to enable context-sensitive operation and management, at least four sets of questions about specification of context information should be answered. We will discuss our approach to modeling context through answers to these questions.

1. How to formally specify monitored context properties in a way useful for management activities? Our approach to this question has four key aspects, discussed in the following paragraphs:
   a. support for well-defined, detailed, and precise specifications,
   b. modeling of context properties analogously to QoS metrics,
   c. extension of existing management information formats for Web services and corresponding management middleware, and
   d. outsourcing of definitions of context property types into external ontologies.

First, it is crucial to understand that to perform management activities with minimal human intervention, the information about these activities must be unambiguous, detailed, precise, and in a well-defined format that can be interpreted by software. Without this, management cannot succeed in the long-run. For example, if the information is not detailed and precise (e.g., there is no information where monitoring is done), the management system might rely on some implicit assumptions and conventions (e.g., that all monitoring is done on provider side), but after a change of circumstances (e.g., there is a new need for some consumer-side monitoring), the management system is no longer appropriate. The saying: “You cannot
control what you cannot monitor and you cannot 
monitor what you cannot describe” is popular in 
the IT system management community. Therefore, 
our research always aims at such details and pre-
cision that enable flexible management (but are 
not incurring too high overhead).

Second, during our study of various formats 
for specification of context information in the past 
published literature and also our examination of 
how context impacts management activities, we 
noticed significant similarities in both semantics 
and syntax of context properties and other manage-
ment information, particularly QoS metrics. Both 
QoS metrics and context properties are groups 
of diverse attributes monitored during run-time 
that have to be described in detail for successful 
management activities. Further, both context 
properties and QoS metrics are measured/calcu-
lated, aggregated, transferred, processed, stored, 
and accounted in a similar manner. Additionally, 
context properties often have the data type contain-
ing numerical value and associated measurement 
unit, which is the most common format for QoS 
metrics and other management information. Using 
different formats to describe similar information 
can lead to problems in interoperability and in-
creases run-time overhead, which is an important 
issue in mobile devices. Therefore, we decided 
to model context properties analogously to QoS 
metrics. However, it is important to note that 
there are also differences between QoS metrics 
and context properties, which justify the need for 
separate specification of context properties. Most 
importantly, the managed system is responsible 
for values of QoS metrics, while this is not the 
case for context properties. These differences 
can affect specification and calculation of prices 
and monetary penalties. Also, whenever value 
of a context property changes, there is a need to 
reevaluate which context-dependent management 
activities are relevant in the new context, while 
a change in value of a QoS metric usually has 
a less immediate effect. Thus, specification of 
context properties is similar, but not identical to 
the specification of QoS metrics.

Third (but closely related to the previous 
point), we decided to extend existing manage-
ment information formats/languages (first WSOL 
and then WS-Policy4MASC) and corresponding 
management middleware (WSOI and MASC, 
respectively), instead of developing a completely 
ew format/language or extending the existing 
works on context management. This is because 
there have been more academic works and indus-
trial products on Web service management than on 
context management for Web services. The cor-
responding expertise, experience, languages, and 
tools (including, but not limited to our own work 
related to WSOI/WSOI and WS-Policy4MASC) 
could be reused with our approach. Prior to our 
study of addition of context, WSOL and WS-
Policy4MASC already addressed for QoS metrics 
questions that are similar to those we are discuss-
ing here for context properties. In addition, WSOI 
and MASC middleware contained appropriate 
support to measure these QoS metrics, transfer 
their values, evaluate relevant QoS constraints, 
and adapt service-oriented system compositions. 
These existing QoS management infrastructure 
solutions could be extended for processing of 
context information relatively easily.

Fourth, instead of building a limited set of 
context property types into the specification lan-
guage or allowing definition of context property 
types together with (i.e., in the same files as) other 
management-related definitions, we decided to 
outsource definitions of context property types into 
external ontologies. While we provide some simple 
formats for such ontological definitions, these 
external ontologies can be, in principle, defined 
in any ontological language. This is consistent 
with the way how we define QoS metric types. 
The main benefit of this solution is the balance 
between flexibility/extensibility and complexity 
of the specification language for management 
information. Various existing ontologies that de-
fine different types of context properties could be
reused to some extent (having in mind the differences between definitions of the term “context”).

2. How to formally specify when, where, how, for which parties (e.g., provider, consumer) and by which parties (e.g., provider, consumer, some independent/third party) the context monitoring is performed, as well as when and how values of context properties are transferred between various parties?

As mentioned above, our approach is based on the recognition that specification of management information (including context information) must be unambiguous, detailed, precise, and in a well-defined format that can be interpreted by software. Therefore, our languages (WSOL and WS-Policy4MASC) enable specification of all the necessary details, usually in XML attributes of special actions for monitoring and data transfer. (When complexity of a particular description is so high that it cannot be specified within an XML attribute, XML elements are used.) For example, information about the party that performs monitoring and information about the party for which context is monitored is specified in such attributes (in a general case, these two parties need not be the same, e.g., context could be monitored for a provider, but monitored by an independent third party). To reduce complexity of specifications, the most common situations are specified as default values of XML attributes and elements. When it comes to definition of monitoring activities, our languages support both measurement and calculation (including estimation) of context properties. For example, geographic longitude and latitude are context properties that could be measured by a Global Positioning System (GPS) device attached to the mobile Web service, while country of location could be determined by providing geographic longitude and latitude to a conversion Web service. To specify when monitoring (as well as transfer of management information and all other management activities) happens, our languages define various events (WS-Policy4MASC has much richer support in this regard than WSOL) and also enable definition of complex schedules. Our languages also support various mechanisms for transfer of management information. The default is piggybacking into SOAP headers, but special push and pull operations can also be specified, as discussed in (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2005).

3. How to specify various ways in which context properties influence operation of service-oriented systems (particularly provider services), monitoring of service-oriented systems, and dynamic adaptation of service compositions?

In different contexts, a service-oriented system might require different inputs or initial conditions, provide different results, provide different QoS, bill different prices and monetary penalties, use different management third parties, or perform different adaptation actions (such as replacing some activities or services in a service composition with appropriate alternatives). In our work (WS-Policy4MASC is much more powerful than WSOL in this regard), there are two groups of solutions, discussed in the following paragraphs:

a. values of context properties can be used in expressions to specify their influences, and
b. event notifications about context changes can be used to trigger adaptation activities.

Our languages enable detailed and powerful specification of various expressions. These expressions can manipulate information in different data types (Boolean, arithmetic, arithmetic-with-unit, string, time) and can have different purpose (e.g., description of conditions to be met, description of monetary prices/penalties to be paid and other utilities, description of probability of something occurring, filtering/limiting applicability of other constructs). Our work enables specification of ex-
pressions that limit provided service functionality only to particular values of context properties. In this way, behavior of the service-oriented system can differ in different contexts. For example, a post-condition (modeled in WS-Policy4MASC as a goal policy assertion) of the truck-tracking Web service could state that a Web service operation returns speed of the truck in kilometers/hour in Canada or in miles/hour in the USA. Analogously, expressions enable limiting monitoring and control (including adaptation) activities only to particular values of context properties. For example, it is possible to specify for the truck-tracking Web service that monitoring of availability performed by a particular independent third party is performed only in Canada (e.g., because this third party is hosted by a wireless network provider that operates only in Canada). In such a case, constructs that limit acceptable values for the monitored availability and that prescribe billing of prices or penalties related to this availability or specify what to do when these availability guarantees are not met are all relevant only in Canada. An important special case of such influence of context is when different adaptation actions are performed in different contexts. It is also possible to limit the expected values of context properties in expressions. This helps in fault detection. For example, if the truck-tracking Web service is attached to a truck that is supposed to drive only in Canada and the USA and suddenly the context property for location of this truck-tracking Web service states that the location is in the middle of the Indian Ocean, something is wrong (e.g., the GPS device is faulty) and many of the regular management activities will be meaningless before the cause of this fault is determined and corrected. In some (relatively rare) situations, it might be useful to only check in a particular expression whether a context property is monitored (and then do something based on this Boolean value) without limiting the value of the context property, e.g., because the details of context-sensitivity are described in another, related, construct. Our work also supports these situations.

A crucial aspect of our work (particularly in WS-Policy4MASC) is specification of adaptation actions to be executed when context changes. The expressions discussed above are passive in determining context changes (in the sense that they do not discover context changes until they are evaluated). Therefore, our work also enables active event notification about context changes. When such a notification arrives, the management middleware undertakes the required management operations. These event notifications can be raised by the management middleware itself or they can be external. An example internal management middleware event for the truck-tracking Web service is the change of country from Canada to the USA. It is determined by evaluating expressions specified in our languages and when it happens, it automatically triggers execution of other management actions (e.g., described in WS-Policy4MASC action policy assertions), such as reconfiguring the Web service to use wireless network for the USA instead of the wireless network for Canada, to switch payments from CAD$ to US$, and/or to switch from the metric to the imperial (US) system of measures. An example of an external event is notification that temperature of the freezer in a truck that transports perishable goods (e.g., food) is above some limit prescribed by safety standards. When this external event is raised, the format of the information that the truck-tracking Web service sends to its consumers changes, to highlight the urgency of the situation.

4. How can specification of context properties be used for discovery and selection of appropriate service-oriented systems and their QoS?

In our approach, SLAs or policies are advertised in an extended Universal Description, Discovery, and Integration (UDDI) directory, as additions to the Web Services Description
Language (WSDL) functional specifications. Since values of QoS metrics and context properties change much more often than functional characteristics, SLAs/policies are published not in WSDL extensions but as separate UDDI tModels. In addition to storage of supplementary information, the extended UDDI directory also contains improved operations for publication and search. After a set of services satisfying required functional characteristics is determined during a search, SLAs/policies (including values of context properties) are used for narrowing the selection. Further details about this solution are outside of the scope of this book chapter, but note that a prototype extension of UDDI for WS-Policy4MASC policies was designed, implemented, and tested in (Liu, 2008) and will be presented in another publication. (Unfortunately, there was no analogous work for WSOL.)

Compared to the approaches that specify context properties separately from the other management information, our approach has higher complexity (and this is its main weakness). Rich and powerful management information specification languages (e.g., our WSOL and WS-Policy4MASC) are complex, both in terms of syntax and semantics. This complexity has design-time and run-time aspects. During design-time, the power and generality of the language can lead to verbosity. The complexity of language processing during run-time is related to (and, to some extent, causes) the overall complexity of the corresponding management middleware (e.g., our WSOI and MASC). The use of such management middleware can sometimes incur overhead (e.g., additional use of memory, processing, battery and other resources; higher response times) that might not be acceptable, particularly in mobile service-oriented systems with limited resources. While our solutions might not be appropriate in some cases when only simple specification of context with no (or very simple) other management aspects is needed and resources are scarce, in most other situations the benefits of our approach significantly outweigh the problems caused by higher complexity.

WS-POLICY4MASC IMPLEMENTATION OF CONTEXT MODELING AND ITS EVALUATION

To enable specification of the conceptual solutions presented in the previous section, WS-Policy4MASC has a number of constructs, specified through XML elements and attributes in the language grammar. We will particularly emphasize the new constructs added in WS-Policy4MASC version 0.9 that directly support specification of context-related information. (Karunaratne, 2008) contains the precise XML Schema and illustrative examples for these new WS-Policy4MASC constructs, while this book chapter will provide summary, explanations, and one example highlighting many of our solutions to general readership. The example that we will present is related to management of the truck tracking mobile Web service mentioned earlier. Figures 2 through 7 show some aspects of WS-Policy4MASC support for context-sensitivity in this example and we will discuss them in the following paragraphs. Please note that the complete WS-Policy4MASC file for this example contains some additional information (e.g., namespace definitions), that we omitted for brevity. Furthermore, this WS-Policy4MASC file is accompanied by a WS-PolicyAttachment file, as explained and illustrated in (Tosic, Erradi, & Maheshwari, 2007). The omitted aspects are not crucial for understanding the WS-Policy4MASC modeling of context-related concepts.

The most important feature of the WS-Policy4MASC 0.9 support for context is the new construct “MonitoredContextProperty” for specification of monitored context properties. This includes not only directly monitored context properties (e.g., geographic latitude and longitude), but also complex context properties (e.g., country of location) calculated from other context
properties. Since our approach is to specify context properties analogously to other management information (particularly QoS metrics), this new construct has somewhat similar syntax to the construct “MonitoredDataItem” shown in Figure 1, but the semantics are different. Note that “MonitoredContextProperty” can be used wherever “MonitoredDataItem” can be used (this required modifications of some other WS-Policy4MASC constructs, as summarized later in this section). Among other attributes and sub-elements, the “MonitoredContextProperty” element contains definition of the data type and, if applicable, unit type of the context property, as well as a reference to an external ontological definition of the monitored context property. The party (e.g., provider, consumer) or other scope (e.g., particular operation) for which the context property is monitored is determined by the WS-PolicyAttachment scope to which this particular “MonitoredContextProperty” construct is attached.

Figure 2 shows example WS-Policy4MASC definitions of two context properties. “CurrentLongitude” is defined as a real (i.e., decimal, float) number associated with the unit “Degree-Decimal-Minus180toPlus180” (defined in the ontology determined by the namespace “measurementOntology”). “CurrentCountry” is defined as a string associated with no unit and representing the ontological concept “CountryOfLocation” (defined in the ontology determined by the namespace “contextOntology”).

To specify the details necessary for monitoring and inter-party transfer of context properties, WS-Policy4MASC version 0.9 extended the existing constructs “MonitoredDataCollection” and “MonitoredDataTransfer”, so that they could be used not only for QoS metrics, but also context properties. Both of these constructs are actions that can be specified in action policy assertions. The “MonitoredDataCollection” action references a context property that is monitored and specifies how its measurement or calculation is performed, where this measurement is performed (the default is within the party that performs monitoring), and whether the monitored values are transferred using piggybacking in SOAP headers (which is the default mode of transfer of monitored values). If push/pull operations for transfer of monitored values are used in addition to (or instead of) SOAP headers, then the “MonitoredDataTransfer” action should also be specified. The latter construct references a set of context properties to be transferred together and specifies to which party they are transferred, the type of data transfer

Figure 2. The truck-tracking Example: Definitions of monitored context properties for current longitude and current country

```xml
< masc-cp:MonitoredContextProperty MASCID="CurrentLongitude"
PropertyType="xs:decimal" PropertyUnit="measurementOntology:Degree-Decimal-
Minus180toPlus180">
< masc-om:OntologicalMeaning OntologicalDefinition="contextOntology:GeographicLongitude-
WestIsNegative"/>
</masc-cp:MonitoredContextProperty>
...
<!-- Definition of geographic latitude is similar to the definition of geographic longitude, so it is
omitted for brevity -->
...

< masc-cp:MonitoredContextProperty MASCID="CurrentCountry"
PropertyType="xs:string" PropertyUnit="masc-cn:MASC_NO_UNIT">
< masc-om:OntologicalMeaning OntologicalDefinition="contextOntology:CountryOfLocation"/>
</masc-cp:MonitoredContextProperty>
```
(e.g., push or pull), and the particular operation (in a particular endpoint/port of a particular Web service) used for this data transfer. The information on which party (e.g., orchestrator, provider, or consumer) performs monitoring or from which data is transferred is specified in an attribute of the action policy assertion that contains the “MonitoredDataCollection” and “MonitoredDataTransfer” actions. The information about the conditions that have to be satisfied for monitoring or transfer of context property values are specified in the “When” construct referenced by the action policy assertion. These conditions list states in which the actions could be executed, events (possibly periodic events generated from complex schedules) that could trigger execution of these actions, and optional filtering Boolean conditions that make that the actions are executed only if these conditions are satisfied. In the same action policy assertion, multiple “MonitoredDataCollection” and “MonitoredDataTransfer” actions can be specified, but then they all have to have the same execution conditions and to share the same party that is responsible for management activities.

Figure 3 shows example WS-Policy4MASC constructs for collection (in this case, calculation) and transfer of the context property “CurrentCountry”. The “MonitoredDataCollection” construct “LocationDataCalculation” states that “CurrentCountry” is calculated as the result of providing the context properties “CurrentLongitude” and “CurrentLatitude” to the operation “CountryFromLongitudeAndLatitude” (specified in the external WSDL file determined by the namespace “conversion-wsdl”) and that it is not passed in SOAP headers. (Monitoring of “CurrentLongitude” and “CurrentLatitude” is not defined in Figure 3, but these values could be obtained from a GPS device.) The “MonitoredDataTransfer” construct “LocationDataTransfer”
states that the context property “CurrentCountry” is transferred to the orchestrator (“MASC_WSORCHESTRATOR” is a special constant) using a predefined push-type operation. Since these actions are contained in the action group “LocationData-CollectionAndTransfer” (shown in Figure 3), which is used in the action policy assertion “MonitorCurrentCountry” (shown in Figure 5) for which management party is the provider (“MASC_WSPROVIDER”), this means that calculation of “CurrentCountry” is performed by the provider and that the provider transfers (i.e., pushes) the calculated value of “CurrentCountry” (to the orchestrator, as mentioned above).

To specify influence of context properties on operation of the managed service-oriented systems, monitoring activities, and dynamic adaptation activities, we had to only slightly modify specification of expressions and to add one new event type. WS-Policy4MASC 0.9 contains a very powerful and detailed specification of Boolean, arithmetic, arithmetic-with-unit, string, and time expressions. It is based on the WSOL specification of expressions, summarized in (Tosic, Pagurek, Patel, Esfandiari, & Ma, 2005). As shown in Figure 1, WS-Policy4MASC Boolean expressions are specified within filtering Boolean conditions of the “When” construct or within goal policy assertions. Arithmetic-with-unit expressions are specified within business value metrics that are within utility policy assertions and could be used for specification of various limits for policy conflict resolution strategies in meta-policy assertions, while arithmetic expressions are specified within probability policy assertions. Additionally (and this is not shown in Figure 1), Boolean expressions can contain comparisons of the other types of expressions (e.g., a comparison of a monitored context property with the arithmetic-with-unit data type with a constant or expression with the arithmetic-with-unit data type). Particularly important is the use of context properties in Boolean expressions within filtering conditions that are part of the “When” construct, because they limit applicability of WS-Policy4MASC policy assertions only to a particular context. For example, use of such a filtering condition within a “When” construct for an action policy assertion leads to execution of actions (e.g., adaptations) in different contexts. To support such use of context properties, we enabled that monitored context properties can be referenced as parameters in all types of expressions (since different context properties could have different data types) and added the Boolean operator “IsMonitoredContextProperty” checking whether a context property is monitored. To support notification about changes in context properties, we added the new event type “MonitoredContextPropertyChanged”, which references a particular monitored context property. Using this event type, it is possible to use a change in a monitored context property to trigger evaluation of a goal policy assertion, execution of an action policy assertion, and calculation of a utility (or probability) policy assertion.

Figure 4 shows definition of the “MonitoredContextPropertyChanged” event notifying about updates of “CurrentLongitude”. The same figure also defines the “When” construct that is satisfied when geographic longitude or geographic latitude is updated. This “When” construct does not have a filtering condition, which means that it is executed whenever one of the specified events is triggered (and the MASC system is in the executing state).

Figure 5 shows how WS-Policy4MASC enables specification that a change of the context property “CurrentLongitude” or “CurrentLatitude” triggering the “When” construct “LongitudeOrLatitudeUpdated” (defined in Figure 4) results in re-calculation and transfer of the context property “CurrentCountry” through the action group “LocationData-CollectionAndTransfer” (defined in Figure 3).

Figure 6 shows how the context property “CurrentCountry” is used within the Boolean expression “IsCurrentCountryUSA” that specifies a filtering condition of the “When” construct “CountryUpdatedToUSA”. In this Boolean
expression, “CurrentCountry” is compared for equality with the string constant “USA”. Only if this condition is satisfied (and the value for the “CurrentCountry” context property was updated and the MASC system is in the executing state), the “When” construct is satisfied. This means that in any country other than the USA, this “When” construct cannot be satisfied.

Figure 7 shows how the context-sensitive “When” construct “CountryUpdatedToUSA” (defined in Figure 6) limits evaluation of the goal policy assertion “MaxSpeedInUSA” only to the USA. In this case, the Boolean expression “UpperLimitOfValidCurrentSpeed” contained by the goal policy assertion “MaxSpeedInUSA” limits the current speed of the tracked truck to 65.00 miles/hour (this speed is returned by an operation of the truck-trucking mobile Web service). However, the main point in Figure 7 is the reference to the “When” construct “CountryUpdatedToUSA” with the above-mentioned filtering condition that limits applicability to the USA. There could be an analogous Boolean expression, “When” construct, and goal policy assertion limiting truck’s speed to 100 kilometers/hour when the truck is in Canada. Then, change of value in the context property “CurrentCountry” automatically affects which of these two goal policy assertions is evaluated.

To verify and validate our conceptual solutions related to modeling of context for management of service-oriented systems, we built these concepts into our languages (first WSOL and then WS-Policy4MASC) and middleware (WSOI and only to some extent MASC). The implementation, verification, and validation of our
Specification of Context for Management of Service-Oriented Systems with WS-Policy4MASC

Figure 6. The truck-tracking example: Definition of the “when” construct that defines the situation when current country is updated to the string “USA”

```
<masc-se:When MSCID="CountryUpdatedToUSA">
  <masc-se:AllowedStates>
    <masc-se:StateRef To="tns:MSC выполненний"/>
  </masc-se:AllowedStates>
  <masc-se:PossibleTriggerEvents>
    <masc-se:EventRef To="tns:CountryUpdated"/>
  </masc-se:PossibleTriggerEvents>
</masc-se:When>
```

Figure 7. The truck-tracking example: Definition of the boolean expression and the referencing goal policy assertion to limit value of the monitored data item for current speed in the USA

```
<!-- Definition the monitored data item “CurrentSpeed” is omitted for brevity. In essence, current speed is returned by an operation of the truck-trucking mobile Web service and uses measurement units of the current country of location (e.g., the imperial system for the USA, the metric system for Canada), -->
...
  <masc-ex:BooleanExpression MSCID="UpperLimitOfValidCurrentSpeed">
    <masc-ex:ArithmeticWithComparator Type="LessOrEqual"/>
    <masc-ex:MonitoredDataItemRef To="tns:CurrentSpeed"/>
    <masc-ex:ArithmeticWithConstant>
      <masc-ex:ArithmeticValue>65.00</masc-ex:ArithmeticValue>
      <masc-ex:UnitOntologicalType>measurementOntology:MilesPerHour</masc-ex:UnitOntologicalType>
    </masc-ex:ArithmeticWithConstant>
  </masc-ex:BooleanExpression>
...
  <masc-gp:GoalPolicyAssertion MSCID="MaxSpeedInUSA" ResponsibleParty="masc-cn:MSC_WSPOD" ManagementParty="masc-cn:MSC_WSPOD"/>
  <masc-se:WhenRef To="tns:CountryUpdatedToUSA"/>
  <masc-ex:BooleanExpressionRef To="tns:UpperLimitOfValidCurrentSpeed"/>
</masc-gp:GoalPolicyAssertion>
```
MASC code can be reused to deal with context. The most important modification required from the MASC middleware extension is to update the management information model to store, process, and communicate context information. While we have not yet implemented it fully (due to the unfortunate lack of C#/.NET programmers in our current team), we thoroughly checked the detailed design that there are no issues with implementability. To validate usefulness of the proposed WS-Policy4MASC improvements, we explored several hypothetical case studies (e.g., the truck tracking Web service) for which we developed example WS-Policy4MASC files and checked their syntax and semantic correctness. We rejected several potential improvements that required too complex modifications of related service-oriented system middleware tools or for which we could not find in our case studies convincing examples of usefulness. The fact that we used a similar approach to verification and validation of implementation of some of these conceptual solutions in WSOL/WSOI strengthens our conclusions that these conceptual solutions are feasible/implementable and useful.

FUTURE TRENDS

The area of context-sensitive operation and context management for service-oriented systems executing in mobile devices is an important research trend. As the number and diversity of mobile devices (e.g., mobile/cell phones, PDAs, laptops) grows rapidly, the number of mobile consumer and provider Web services increases, along with diversity of context properties relevant for their operation. Therefore, the importance of this research area will continue to increase in the near future. Due to the huge diversity of definitions of the term “context”, this is a very broad research area. We believe that it is important to more clearly characterize the area and its research problems by agreeing on a definition of the term “context”. (Otherwise, this term will become an empty marketing buzzword, used at will for promotion of whatever is put for sale.) Note that definition of the term “context” directly impacts definition/specification/description of individual context properties (attributes), which directly impacts what can be done in and achieved by monitoring and control activities. In this book chapter, we provided our broad definition and narrow definition of the term “context”, appropriate from the viewpoint of IT system management. However, we invite the readers to discuss and improve our proposal, so that we together can find a terminological solution appropriate for the broader community.

An important research sub-area is how to integrate context specification and management into (Web) service management systems that deal with various management activities and various types of service-oriented systems (some of which are mobile, but the majority of which are non-mobile). This sub-area is the focus of this research chapter. While we made important progress towards addressing the identified challenges, many open issues remain, both for the broader community and for our research program, as outlined below.

The main challenges for the broader research community are in the area of control of service-oriented systems, taking into consideration various context properties. Dynamic (run-time) adaptation to changes in context is a particularly important and fruitful topic for future research. Ideally, such dynamic adaptation would be with minimal human involvement, because knowledgeable humans are not always available, might react too slowly, or are too expensive. More precisely, humans would only set high-level goals and rules (e.g., in the form of policies) and not detailed programs how to react to each particular situation, because many situations might not be predictable when the software is written. This is the ideal of self-management of IT systems, also popularized through the autonomic computing initiative (Kephart & Chess, 2003). However, it is important that the manage-
ment system optimizes business value metrics (e.g., profit, customer satisfaction) in addition to technical QoS metrics (e.g., response time, availability). This is because business value metrics are more important to business users than technical QoS metrics. Unfortunately, the past practice has shown that mapping between technical and business models and metrics is difficult. For example, higher availability need not lead to increases in business profits. The goal of business-driven IT management (BDIM) (Bartolini, Sahai, & Sauve, 2007) is to determine such mappings and leverage them to make run-time IT system management decisions that maximize business value. For example, it tries to quantify impact on business profits of increased/decreased availability. Our work on WS-Policy4MASC and the corresponding MASC middleware contains unique support for autonomic business-driven management of service-oriented systems (Tosic, Erradi, & Maheshwari, 2007). However, many additional adaptation algorithms for dynamic adaptation to various changes (including, but not limited, changes in context) can be developed. These additional algorithms can use and, if needed, extend WS-Policy4MASC for specification of diverse management policies and other management information (including information related to context).

Another set of challenges for the broader research community are in the area of service selection and composition, taking into consideration context information. There are many open issues here. One of them is that historical information about QoS metrics from past contexts might not be fully appropriate for the new context of the composition. For example, historical average and maximum response time of a service located in Toronto, Canada that had up to 1000 concurrent customer invocations from Canada might not be relevant when this service is composed with services located in Sydney, Australia and is expected to receive up to 5000 concurrent invocations from Australia. Another open issue is that lightweight mashups of services (the popularity of which rapidly increases) might lack contractual composition information that is very useful (if not necessary) for management, leading to additional management challenges. Finally, we note the open issue of better integration of ontologies, Semantic Web technologies, and service intelligence research results into IT system management, both for service selection and composition and for control of service-oriented systems.

The long-term objective of our own research program is a powerful management system for service-oriented systems executing in diverse environments, ranging from embedded systems to mobile devices to desktop computers to specialized servers to cloud/utility/grid computing systems. In the short term, we are particularly focused on providing innovative solutions for autonomic business-driven management of service-oriented systems and business processes they implement. The work presented in this book chapter is an important step towards our goal. However, we have been building upon the presented results (and WS-Policy4MASC in general) by developing new algorithms for advanced (Web) service management and business process management, particularly in the area of dynamic adaptation. We also want to perform additional implementation and evaluation of our solutions. In this respect, we are searching for industrial partners interested in application of our research results on real-life scenarios.

**CONCLUSION**

Many recent research projects and papers (particularly in mobile computing) are related to context modeling, processing, and/or management. They explored the concept of context in various circumstances and from various viewpoints. However, they used many different definitions of the term “context”. We found that these definitions cover several groups of concepts that have considerably different nature and that must be differentiated for
successful IT system management (monitoring and control) activities. Therefore, we decided to provide two definitions of the term “context”. In the broad sense, context is (any and every) information that characterizes situation of the (managed) system. On the other hand, in the narrow sense, context is information about external run-time circumstances that characterize situation of the (managed) system and influence its operation (i.e., execution, behavior), but are outside its direct control. In this book chapter, we have focused on context in the narrow sense. Context-sensitivity means taking into consideration the influence of context on operation and management of the managed system. It is particularly important for mobile systems, because location and characteristics of the surrounding environment are context properties that can considerably influence operation and management. Context-sensitivity is also important for operation and management of mobile (and also non-mobile) service-oriented systems, but this area is not yet explored thoroughly. In different contexts, a service-oriented system might require different inputs or initial conditions, provide different results, provide different QoS, bill different prices and monetary penalties, use different management third parties, or perform different adaptation actions.

Specification of monitored context properties and their influence on operation of service-oriented systems and on management activities is a prerequisite for context-sensitivity. In spite of many recent related works, context specification for a management system performing various management activities and used by various types of service-oriented systems was not addressed adequately prior to our research.

Our approach to providing context-sensitive operation and management of service-oriented systems is based on extending existing management information specification languages and corresponding management middleware. This supports compatibility between management of mobile service-oriented systems and management of the other service-oriented systems. Due to the similarities between processing and use of context properties and processing and use of QoS metrics, we decided to model context properties analogously to QoS metrics. However, due to the semantic differences between context properties and QoS metrics (e.g., the fact that the managed system is not responsible for values of context properties, but is at least partially responsible for values of QoS metrics), context properties are modeled as a distinct concept. While our specifications of monitored context properties are well-defined, detailed, and precise (so that they can be used for management activities), their meaning is defined in external reusable ontologies. To formally specify when, where, how, for which parties and by which parties the context monitoring is performed, as well as when and how values of context properties are transferred between various parties, our approach provides special actions for monitoring (measurement or calculation) and transfer of values of context properties. The necessary details are captured in attributes (or sub-elements) of these actions. To formally specify various ways in which context properties influence operation, monitoring, and control (particularly dynamic adaptation) of service-oriented systems, we enabled that values of context properties can be used in various expressions and provided events that notify about context changes and, subsequently, trigger adaptation activities.

While these are the major strengths of our approach, its main weakness is additional complexity (compared to the approaches that specify context properties separately from the other management information). The generality, expressive power, and richness of management information specification languages cause some complexity. This complexity has design-time and run-time aspects. During design-time, there is a need to specify various aspects and there can be some verbosity. (Our WS-Policy4MASC example in Figure 2 through Figure 7 shows some verbosity.
While most constructs in WS-Policy4MASC are optional, there are some constructs, such as the “When” construct, that are usually necessary for specification of context and its influences.) The complexity of language processing during runtime is related to (and, to some extent, causes) the overall complexity of the corresponding management middleware. The use of such middleware can incur overhead (e.g., additional use of memory, processing, battery and other resources; higher response times) that might not be acceptable, particularly in mobile service-oriented systems with limited resources. Thus, our solutions might not be appropriate in cases when only simple specification of context with no (or very simple) other management aspects is needed and the resources are scarce. Nevertheless, in most situations the strengths of our approach significantly outweigh its weaknesses.

We implemented our solutions for specification of context properties and related management activities in the WS-Policy4MASC language, which extends the industrial standard WS-Policy with powerful constructs for formal specification of information necessary for run-time management (e.g., dynamic adaptation) of service-oriented systems. These WS-Policy4MASC constructs include goal policy assertions, action policy assertions, utility policy assertions, probability policy assertions, meta-policy assertions, as well as a number of auxiliary constructs specifying ontological meaning, monitored QoS metrics, states, state transitions, schedules, events, scopes, and various types of expression (Boolean, arithmetic, arithmetic-with-unit, string, time/date/duration).

To enable specification of the conceptual solutions for modeling of context and related information, the recent WS-Policy4MASC version 0.9 added or extended several constructs, such as the new “IsMonitoredContextProperty”, and the new event type “MonitoredContextPropertyChanged”. Particularly important is the use of context properties in Boolean expressions within filtering conditions that are part of the “When” construct, because they limit applicability of WS-Policy4MASC policy assertions only to a particular context. Also important is the use of the event type “MonitoredContextPropertyChanged” to trigger evaluation of goal policy assertions, execution of action policy assertions (e.g., performing dynamic adaptation), and calculation of utility (or probability) policy assertions. All these new or extended constructs increase usefulness of WS-Policy4MASC for management of mobile service-oriented systems. To verify feasibility and implementability of the new WS-Policy4MASC constructs, we checked their syntactical correctness and thoroughly studied required modifications of the MASC middleware that uses WS-Policy4MASC for management of service-oriented systems. To validate usefulness of the proposed WS-Policy4MASC improvements, we explored several hypothetical case studies (e.g., the truck tracking mobile Web service). We plan additional implementation and evaluation of our solutions, hopefully on real-life scenarios.

ACKNOWLEDGMENT

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program. The initial stages of the research project on specification of context for management of service-oriented systems (but not the WS-Policy4MASC specifics) were performed while Vladimir Tosic was an NSERC (Natural Sciences and Engineering Research Council of Canada) post-doctoral fellow at the University of Western Ontario. Hanan Lutfiyya and Yazhe Tang contributed to these stages. Earlier versions (up to 0.8) of WS-Policy4MASC were developed within
a research project sponsored by the Australian Research Council (ARC) and Microsoft Australia. Abdelkarim Erradi and Piyush Maheshwari contributed to this early work on WS-Policy4MASC.

REFERENCES


KEY TERMS AND DEFINITIONS

System Management: The process of monitoring and control of a system to ensure regular operation of the system and to handle run-time problems (e.g., faults, performance degradations).

Monitoring: The process of determining the state of a system, e.g., by measuring or calculating various quality of service (QoS) metrics, determining presence of faults, evaluating satisfaction of requirements and guarantees, and calculating monetary amounts to be paid.

Control: The process of run-time modification (e.g., re-configuration) of a system to ensure its regular operation, in spite of external changes or internal run-time problems (e.g., faults, performance degradations).

Quality of Service (QoS): A group of measures of how well (e.g., how quickly, how reliably) a system performs its operations.

QoS Metric: A particular measure of QoS, such as response time, throughput, or availability.

Context (Broad Meaning): Information that characterizes situation of a system, such as geographic location, past interactions with other systems, and past availability.

Context (Narrow Meaning): Information about external run-time circumstances (e.g., geographic location of a mobile system, events that come from outside, preferences of system’s consumers) that characterize situation of a system and influence its operation, but are outside its direct control.

Context Property: A particular attribute/aspect of context, such as country of location or network latency between the monitored system and external systems.

Policy: A collection of high-level, implementation-independent goals and/or rules that describe what should be achieved by management (e.g., specify limits for response time).