Autonomic Business-Driven Dynamic Adaptation of Service-Oriented Systems and the WS-Policy4MASC Support for Such Adaptation

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

ABSTRACT
When a need for dynamic adaptation of an information technology (IT) system arises, often several alternative approaches can be taken. Maximization of technical quality of service (QoS) metrics (e.g., throughput, availability) need not maximize business value metrics (e.g., profit, customer satisfaction). The goal of autonomic business-driven IT system management (BDIM) is to ensure that operation and adaptation of IT systems maximizes business value metrics, with minimal human intervention. The author presents how his WS-Policy4MASC language for specification of management policies for service-oriented systems supports autonomic BDIM. WS-Policy4MASC extends WS-Policy with new types of policy assertions: goal, action, probability, utility, and meta-policy assertions. Its main distinctive characteristics are description of diverse business value metrics and specification of policy conflict resolution strategies for business value maximization according to various business strategies. The author’s decision making algorithms use this additional WS-Policy4MASC information to choose the adaptation approach best from the business viewpoint.

Keywords: business-driven IT management, key performance indicator (KPI), autonomic computing, dynamic adaptation, policy-driven management, policy specification, policy conflict resolution, service-oriented computing, Web service, Web service management

INTRODUCTION
In the modern world, technical and business changes are frequent and increasingly common. Additionally, the modern globalized economy increasingly supports and often requires various business relationships between diverse companies. These circumstances place important requirements on enterprise information technology (IT) systems: the ability to seamlessly interconnect with IT systems of diverse business partners irrespective of the implementation of these IT systems and the ability to handle various technical and business changes (e.g., temporary computer/network failures and establishment of new business alliances).

Service-oriented computing (SOC) was developed to answer these challenges. In service-oriented systems (Papazoglou & Georgakopulos, 2003), such as Web services and their compositions, parts of internal IT (e.g., software) systems are exposed as implementation-independent services, which are then composed in a loosely-coupled manner, possibly dynamically, i.e., during run-time (instead of during software/system design). However, just implementing and composing service-oriented systems is not enough to fully address diverse
technical and business changes that can affect these systems. To discover and address changes, IT systems have to be managed (Sloman, 1995).

Management of IT systems, including service-oriented systems, is the process of their monitoring and control to ensure correct operation, enforce security, discover and fix problems (such as faults or performance degradations), maximize quality of service (QoS), accommodate changes, and achieve maximal business benefits. Monitoring determines the state of the managed system, e.g., by measuring or calculating QoS metrics, determining presence of problems, evaluating satisfaction of requirements and guarantees, accounting of consumed resources, and calculating prices/penalties to be paid. A QoS metric, such as response time or availability, is a measure of how well a system performs its operations. On the other hand, control puts the managed system into the desired state, by performing run-time adaptation of the system to ensure its correct operation, in spite of changes or run-time problems. For example, control of a service-oriented system includes its re-configuration, re-composition of services, and re-negotiation of contracts between the composed services and between the system and other parties. Control actions result in changes of monitored conditions (e.g., QoS metrics), which closes the management loop.

The majority of past IT system management products act as support tools for human system administrators – these products present summaries of monitored information and often automate some simpler control actions, but it is ultimately human responsibility to make more complex decisions about execution of control actions. Since the complexity of modern IT systems is rapidly increasing, human system administrators exhibit difficulties in making optimal decisions. Furthermore, human system administrators are expensive and might not be available at all times. Therefore, minimizing human involvement in IT system management has been a research goal for several decades and was made prominent by the vision of autonomic computing (Kephart & Chess, 2003). In autonomic computing, IT systems are self-managing, i.e., they manage (e.g., adapt) themselves using configurable policies, with minimal human intervention.

A prerequisite for performing IT system management activities is existence of a machine processable and precise format for specification of the monitored values/conditions and the control actions (Keller & Ludwig, 2003; Tosic, Pagurek, Patel, Esfandiari & Ma, 2005). Policies (Sloman, 1994) are a frequent approach to IT system management, not limited to autonomic computing. A policy formally specifies a collection of high-level, implementation-independent, operation and management goals and/or rules in a human-readable form. To improve flexibility, maintainability, reusability, and simplicity of specifications, policy description of monitoring and control aspects is separated from descriptions of the managed IT system. During run-time, a policy-driven management system refines these high-level goals and rules into many low-level, implementation-specific, actions controlling operation and management of the managed system and its components. Another format for specifying what should be achieved by IT system management is a service level agreement – SLA (Keller & Ludwig, 2003). An SLA is a special type of contract (a binding and enforceable formal agreement between two or more parties) that specifies QoS requirements and guarantees and, often, prices and monetary penalties to be paid. It can be used as an alternative or as a complement to policies. 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). Traditionally, SLAs and policies also differ in the formatting of represented information and in the architectures of management infrastructures (middleware).

The past IT system management (including autonomic computing) solutions were mostly focused on functional correctness, security, and optimization of technical QoS metrics (e.g.,
response time, availability). They provide only a simple treatment of financial business value metrics (e.g., prices and monetary penalties), without addressing non-financial business value metrics. A business value metric is any measure of business worth of an item or a situation to a particular business party. (Instead of the term “business value metric” some authors use the similar terms “key performance indicator – KPI”, “business performance metric”, “business QoS”, and “quality of business – QoBiz”. Some business value metrics are financial, such as earned income, costs, profit, and return on investment (ROI). While financial business value metrics are important for all companies, they are not the only measures of business worth. Examples of non-financial business value metrics are the number of customers, market share, and customer satisfaction. Note that business value metrics are relative to business parties, e.g., concrete numerical quantities and possibly types of business value metrics for a consumer and a service provider can be different.

Business value metrics are more important to business users than technical QoS metrics. For example, a business user is usually not very interested in the fact that availability dropped from 99% to 95%, but wants to know how much this change costs her/his business. Unfortunately, the past practice has shown that mapping between technical and business models and metrics is difficult (Bartolini, Sahai & Sauve, 2006; Bartolini, Sahai & Sauve, 2007; Biffl, Aurum, Boehm, Erdogmus & Gruenbacher, 2006). For example, increasing availability need not lead to increases in business profits, because the costs to provide higher availability (e.g., through replication, partitioning, load balancing, or other means) could outweigh the increased income from customers.

The goal of business-driven IT management (BDIM) research community is to determine mappings between technical QoS and business value metrics and leverage them to make run-time IT system management decisions that maximize business value metrics (Bartolini, Sahai & Sauve, 2006; Bartolini, Sahai & Sauve, 2007). For example, it tries to quantify impact on business profits of increased/decreased availability. Integration of autonomic computing and BDIM was identified in (Bartolini, Sahai & Sauve, 2007, pp. 23-24) as an area of open research challenges. One of the limitations of the past BDIM works is that most of them focused on maximizing profit. However, human managers use many business strategies that differ (among many other aspects) in how they prioritize different business value metrics in different time frames (e.g., long-term vs. short-term). As will be illustrated later in the paper, maximization of short-term profit is not always appropriate. Business strategies are a major differentiator of companies in a market, so many diverse business strategies will exist in the market of (Web) services and should be supported by autonomic BDIM solutions.

We present our results on developing autonomic BDIM solutions, based on the WS-Policy4MASC language (Tosic, Erradi & Maheshwari, 2007; Tosic, 2008) for specification of management policies for service-oriented systems. WS-Policy4MASC can be used for monitoring and control activities focused on functional correctness and technical QoS metrics (Erradi, Tosic & Maheshwari, 2007), but its main original contributions are the description of diverse business value metrics (both financial and non-financial) and the specification of policy conflict resolution strategies for maximization of business value metrics according to various business strategies. We developed algorithms that use this additional WS-Policy4MASC information to provide unique support for autonomic business-driven dynamic adaptation of service-oriented systems. This paper complements our previous publications on the WS-Policy4MASC language, particularly (Tosic, Erradi & Maheshwari, 2007) and (Tosic, 2008), by
providing additional explanations of some of the challenges for autonomic BDIM and how WS-Policy4MASC supports autonomic BDIM.

In the next section, we present motivating examples that illustrate both the need for autonomic BDIM software and some of the difficulties in developing such software. Then, we give a brief survey of the major related work. Next, we provide a high-level overview of the main features of WS-Policy4MASC. The main section of the paper contains detailed discussion of WS-Policy4MASC support for autonomic business-driven adaptation of service-oriented systems. In the final section, we summarize conclusions and items for future work.

EXAMPLES OF SOME CHALLENGES FOR AUTONOMIC BUSINESS-DRIVEN IT MANAGEMENT

Figure 1 shows an example situation of a distributed system that requires dynamic adaptation and which we will use to illustrate the need for autonomic business-driven adaptation of service-oriented systems. Party X implements some business process as a composition of 3 (Web) services: A, B, and C. (Note that A-B-C can be, in principle, any distributed computing system, but we will assume a service-oriented system.) Party B has a contractual guarantee (specified in an SLA or a policy agreed with the other parties) to provide 99% availability every day and charge AU$3 per invocation. If it does not meet the availability guarantee in a particular day, it pays the penalty of AU$10. Parties A and C have their own QoS guarantees and prices, which we will abstract. Party X gives an availability guarantee to its own consumers of 98% per day, with unavailability penalty of AU$50 and price of AU$20 per invocation. This service composition works without problems for some time, but then a problem happens with B that prevents it to keep it its guarantee of 99% availability. For example, B could become overloaded due to requests that it receives from some other consumers (not shown in Figure 1) – it is not uncommon that a service is concurrently part of several service compositions. We will assume that the problem is of such nature that B has to be replaced with an alternative service. After a search of all service directories, no exact replacement (in terms of functionality, QoS, and price) for B was found. Instead, 2 services, D and E, with the same functionality as B, but with different QoS and price were found. D guarantees 95% availability (lower than B) with AU$1 penalty but with the price of AU$1 per invocation (cheaper than B), while E guarantees 99.99% availability (higher than B) with $80 penalty with the price of AU$5 per invocation (more expensive than B).
The composing party X now has to make a decision which of these alternatives (D or E) to choose. We would like that software at X could make this decision without direct human involvement, i.e. autonomically. The traditional IT systems management (including Web service management) software would maximize technical metrics, so it would choose E. However, this need not be the best decision from the business viewpoint. The execution of the composition provides some business value to X – for simplicity, we will assume that this is profit. It is possible that the higher price of E would significantly reduce X’s profit from the composition. On the other hand, it is likely that lower availability of D would cause X to pay availability penalty more often, which might lower X’s profit in spite of D’s lower price. The decision of whether to choose D or E also depends on a number of other factors (not shown in Figure 1), such as probabilities that particular parties will meet their availability guarantees.

Note that situations in real service-oriented systems are often much more complicated than the relatively simple example in Figure 1. For example, a party often offers multiple QoS guarantees (e.g., about response time or throughput, in addition to availability), guarantees/prices/penalties might differ between operations of the same service, there can be various payment models (e.g., per invocation, subscription, per information volume), guarantees/prices/penalties can differ based on context (e.g., X could charge consumers from developing countries less), etc. Additionally, service compositions are often more complicated than in Figure 1, both in terms of the number of composed services and in terms of the control structure (e.g., there could be parallel branches or conditional executions). Further, in some circumstances, many different control actions exist, with different consequences. For example, when a (Web) service does not respond on time, the composing party could decide to: wait for the reply a bit longer, resend the request to this service (if it is idempotent), cancel this request, simply skip this request (and ignore its results when/if they come later), replace the faulty service with a known alternative, search for an alternative in a service directory and then use this replacement, etc. Moreover, not only single control actions, but also groups (e.g., sequences) of actions could be appropriate. Consequently, even when only financial business values are considered and the business strategy is to maximize profit, there are many complexities that have to be taken into account.

The situation is further complicated by the fact that businesses often have non-financial business value metrics and business strategies that consider such metrics. Assume that in the example from Figure 1, choosing D gives X a bit higher profit, but significantly lowers
satisfaction of X’s customers. If X has the business strategy of maximizing short-term profits, it should choose D. However, if X has the business strategy “exceptional customer satisfaction”, then it has sense for X to choose E that brings lower profits (or even short-term loses), in order to maintain/enhance X’s reputation and competitive advantage in the marketplace. In the long term, the reputation and the customer satisfaction will influence customer retention and recruitment, and, therefore, future profits. In the existing markets (e.g., the car market), there are many examples of companies with the “exceptional customer satisfaction” business strategy (e.g., high-end car manufacturers) and other business strategies that use non-financial business value metrics. We argue that solutions for autonomic BDIM (for IT systems in general, not only for service-oriented systems) should be able to use non-financial business value metrics and business strategies that consider such metrics.

**RELATED WORK**

Many languages for specification of policies, SLAs, and related IT system management constructs for service-oriented systems have been developed. The most important among these languages are accompanied by management-related tools, e.g., middleware. Some examples are the Dynamo middleware using the combination of the Web Service Constraint Language – WSCoL and the Web Service Recovery Language – WSReL (Baresi, Guinea & Plebani, 2007), the Web Service Level Agreement – WSLA language and middleware (Keller & Ludwig, 2003), the Web Service Offerings Infrastructure – WSOI using the Web Service Offerings Language – WSOL (Tosic, Pagurek, Patel, Esfandiari & Ma, 2005), and the Cremona (CReation and MONitoring of Agreements) toolkit using the Web Services Agreement Specification – WS-Agreement (Ludwig, Dan & Kearney, 2004). There are also additional research projects, such as JOpera (Pautasso, Heinis & Alonso, 2007), and industrial products that provide management infrastructures for service-oriented systems. While significant, these past results exhibit several limitations. They predominantly approached monitoring or QoS-based selection of Web services and did not sufficiently address the more challenging issues of control (adaptation), particularly for composite service-oriented systems. Only a few recent works (notably, Dynamo and JOpera) provide support for autonomic management of service-oriented systems.

Another important limitation of these past works is their focus on technical QoS metrics, with very limited support for simple financial business value metrics (e.g., pay-per-use and subscription prices, simple monetary penalties) in SLAs/policies. Only some of the past management infrastructures (usually, the commercial ones) contain accounting/billing subsystems that monitor/calculate monetary amounts to be paid. While business-driven management of service-oriented systems was mentioned in the past literature (Casati, Shan, Dayal & Shan, 2003; Tosic, Erradi & Maheshwari, 2007; Tosic, 2008), there are still many open research issues, particularly related to the use of diverse business strategies (beyond short-term profit maximization) and non-financial business value metrics.

Business literature provides evidence that modeling of non-financial business value metrics in addition to financial ones is beneficial for long-term strategic management of companies. A widely accepted approach that represents well-being of a company beyond financial metrics is the balanced scorecard – BSC (Kaplan & Norton, 1996). It organizes company-specific business value metrics along 4 standard perspectives: financial, customer, internal business processes, and learning & growth. Unfortunately, there are difficulties in direct reuse of balance scorecards for autonomic BDIM, because they are intended for human use. One of the issues is how to map from the 3 non-financial perspectives (particularly, learning & growth) into run-time BDIM.
activities at the IT level. Nevertheless, the balanced scorecard concepts inspired and significantly influenced several BDIM projects, in addition to our work. A notable example is Management by Business Objectives – MBO (Bartolini, Salle & Trastour, 2006). Its information model contains objectives, business value metrics (the authors use the term “KPI”), and perspectives (inspired by the balanced scorecard). Importance weights are assigned to these objectives and perspectives – this is a limitation, because in practice it can be difficult to determine precise weights. Aline is the corresponding engine that calculates alignment between different actions and objectives, defined as the likelihood of meeting the objectives. In spite of many contributions, this work did not address detailed modeling of business value metrics and business strategies.

(Tosic, 2008) examined some of the issues in modeling of financial and non-financial business value metrics: the need for explicit description of various characteristics of these metrics; advantages and disadvantages of monetization of non-financial business value metrics (e.g., replacing 80% customer satisfaction with the monetary amount AU$300), the need for modeling of possible (but not certain) business value metrics, and accounting for inflation and interest rates using the “time value of money” formulae from economics. The same paper also examined some of the issues in modeling of business strategies that maximize business value metrics: the need for explicit description of various characteristics of business strategies, options for limiting temporal scope of relevant business value metrics, calculation of overall (total, summary) business value metric associated with execution of a control action, and possible complications (e.g., cost limits, tiebreakers) in comparing these overall business value metrics of various control actions.

Many high-quality BDIM papers were published in proceedings of the BDIM workshops, e.g., (Bartolini, Sahai & Sauve, 2006; Bartolini, Sahai & Sauve, 2007), organized annually since 2006. However, important BDIM papers, such as (Salle & Bartolini, 2004), were published before the term “BDIM” was coined and BDIM-related papers are still published in a wide variety of venues (e.g., autonomic computing conferences), sometimes without using the “BDIM” term. Furthermore, BDIM is related to the other research areas that try to improve alignment between business and IT. (Salle, 2004) overviewed relationships between BDIM and IT governance, which is a set of human activities aiming to achieve that IT systems support business objectives and strategies. (Tosic, Suleiman & Lutfiyya, 2007) discussed the need for integrating BDIM with value-based software engineering – VBSE (Biffl, Aurum, Boehm, Ergodmus & Gruenbacher, 2006), which is a software engineering approach that explicitly considers value issues (e.g., value-based prioritization), in order to make the resulting software more useful. Integration of results from various communities will be needed to achieve autonomic BDIM.

OVERVIEW OF THE WS-POLICY4MASC LANGUAGE

WS-Policy4MASC extends the Web Services Policy Framework – WS-Policy (W3C Web Services Policy Working Group, 2007), an industrial standard by the World Wide Web Consortium (W3C). Analogously to the other Web service standards, the syntax of WS-Policy is specified in the Extensible Markup Language (XML) and its grammar is defined in XML Schema. In WS-Policy, 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 it applies. Various policy subjects are possible, such as service-level constructs (e.g., operation, message) in the Web Services Description Language (WSDL) and process-level constructs (e.g., flow, link) in the Web Services Business Process
Execution Language (WSBPEL). 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. However, it is only a general framework, while details of specification of particular types of policy assertions are left for specialized languages. There are several recent academic WS-Policy extensions for specification of QoS, such as WS-QoSPolicy (Rosenberg, Enzi, Michlmayr, Platzer & Dustdar, 2007) and WS-CoL (Baresi, Guinea, Plebani, 2007), but not for business value metrics and many other issues relevant in autonomic BDIM.

Figure 2. The main concepts and relationships in WS-Policy4MASC

WS-Policy4MASC defines 5 new types of WS-Policy policy assertions, as well as a number of auxiliary (supporting) constructs. The most important constructs and relationships in the current version 0.9 of WS-Policy4MASC are shown in Figure 2. The 5 new policy assertion types inherit from the abstract construct “MASCPolicyAssertion” and can be used (e.g., specified in WS-Policy files, attached using WS-PolicyAttachment) in the same way as other WS-Policy policy assertions. The new policy assertion types are:

1. **Goal policy assertions** specify requirements and guarantees (e.g., response time of an activity has to be less than 1 second) to be met in desired normal operation. They guide monitoring activities.

2. **Action policy assertions** specify actions (e.g., replacement/skipping/retrying of a subprocess, process termination) to be taken if certain conditions are met (e.g., some goal policy assertions were not satisfied). They guide control (adaptation) activities. As elaborated in (Tosic, Erradi & Maheshwari, 2007), WS-Policy4MASC has built-in support for specification of a diverse range of common service composition and business process adaptation actions.

3. **Utility policy assertions** specify diverse business value metrics assigned to particular runtime situations (e.g., non-satisfaction of some goal policy assertion, execution of some action, another event). They guide accounting/billing and provide information for business-driven management.

4. **Probability policy assertions** specify probabilities that particular situations will occur. They guide management of various forms of uncertainty (e.g., risks, trust in various parties).
5. Meta-policy assertions specify adaptation alternatives and business strategies for selection between them. They guide business-driven management activities. The utility, probability, and meta-policy assertions will be explained further in the next section.

The above definitions of WS-Policy4MASC policy assertion types are consistent with the (Kephart & Walsh, 2004) discussion of goal, action, and utility policies, as well as the literature on policy conflict resolution with meta-policies (Lupu & Sloman, 1999). We added probability policy assertions after examination of application scenarios that contained uncertainties, risks, and trust issues.

WS-Policy4MASC enables specification of detailed information that is necessary for run-time policy-driven management and that 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 5 policy assertion types. Much more information is specified in additional auxiliary WS-Policy4MASC constructs, specifying ontological meaning, monitored QoS metrics, monitored context properties (e.g., geographic location), states, state transitions, events, schedules, applicability scopes, and various types of expression (Boolean, arithmetic, arithmetic-with-unit, string, time/date/duration). Not all auxiliary constructs are shown in Figure 2. “When” is the most important auxiliary construct. It specifies 1 or more states in which something (e.g., evaluation of a goal policy assertion) can happen, 1 or more events (e.g., Web service operation executed) that can (mutually independently) trigger this, and an optional filtering Boolean condition to be satisfied.

WS-Policy4MASC was used in the Manageable and Adaptable Service Compositions (MASC) middleware for management of service-oriented systems (Erradi, Tosic & Maheshwari, 2007), which is based on the Microsoft .NET 3.5 platform. (WS-Policy4MASC can also be used with different middleware and we are currently developing MiniMASC, a Java re-implementation and extension of the autonomic BDIM aspects of MASC.) We evaluated feasibility of the WS-Policy4MASC solutions by implementing corresponding data structures and algorithms in MASC. Additionally, we examined their expressiveness, effectiveness, and usefulness on 2 realistic case studies (weather report and stock trading) and several smaller examples. Syntax correctness (well-formedness and validity) of XML Schema definitions of WS-Policy4MASC was checked with XML tools.

One of the difficulties in working with WS-Policy4MASC is the need to write long and complicated XML files (WS-Policy files with WS-Policy4MASC policy assertions and WS-PolicyAttachment). Oftentimes, some of the information in policy files is known during design-time (software engineering). To facilitate writing of WS-Policy4MASC files and to help bridge design-time and run-time management issues, we developed the Unified Modeling Language (UML) profiles for WS-Policy4MASC, as well as the Extensible Stylesheet Language Transformations (XSLT) rules that generate WS-Policy4MASC and WS-PolicyAttachment files from information in these UML profiles (Tosic, Suleiman & Lutfiyya, 2007). Furthermore, we defined XSLT rules that annotate UML diagrams with run-time management information collected by the MASC middleware. The latter helps in design-time software adaptation (re-engineering, improvement) by software engineers and complements the autonomic run-time adaptation.

THE WS-POLICY4MASC SUPPORT FOR AUTONOMIC BUSINESS-DRIVEN ADAPTATION
WS-Policy4MASC enables specification of both financial and non-financial business value metrics in utility policy assertions. A utility policy assertion references a “When” construct describing situations in which business value metrics should be calculated, has attributes describing the management party (performing calculation and accounting) and the beneficiary party (e.g., the provider Web service), and lists 1 or more business value metrics. Each business value metric contains an arithmetic-with-unit expression and a set of attributes and sub-elements determining the optional paying party (e.g., a consumer), the optional payment recurrence (e.g., for subscription prices), and the business value metric category. For simplicity, all non-financial business value metrics are monetized (i.e., represented with an estimated monetary amount) and future payments are given in their net present value. The arithmetic-with-unit expression can contain constants, variables, standard mathematical operators, and function calls. Thus, business value metrics can be defined not only as absolute (e.g., “FutureSales = AU$700”), but also as relative (e.g., “FutureSalesToReturningCustomers = 0.7*FutureSales”). This expression is evaluated at run-time to produce a single number with a currency unit. Currency units can be associated with ontological definitions and linked to currency conversion services.

Note that in practice it can be difficult to monetize contributions of individual non-financial aspects. For example, if both customer satisfaction and market share (somewhat, but not completely, dependent on customer satisfaction) contributed to past profit, it can be difficult to estimate how customer satisfaction on its own impacts profit. Therefore, the WS-Policy4MASC approach is to define business value metric categories that are easier to estimate. There are 3 mutually orthogonal dimensions of properties that classify these business value metric categories: i) financial or non-financial, ii) agreed or non-agreed (i.e., only possible), iii) benefit or cost. A business value metric category is a combination of 1 characteristic from each of these 3 dimensions, so there are 8 business value metric categories. For example, “non-financial non-agreed benefit” characterizes customer satisfaction, while “financial non-agreed benefit” characterizes estimates of future sales. In different circumstances (e.g., business strategies), different business value metric categories are relevant for autonomic BDIM decision making (we will explain later how WS-Policy4MASC addresses this). A WS-Policy4MASC business value metric can have an optional array sub-element describing aspects (e.g., “customer satisfaction”) that it represents, but this information is currently only for documentation purposes.

WS-Policy4MASC enables dealing with uncertainty through probability policy assertions. A probability policy assertion references a “When” construct, has attributes describing the management party as well as the trusting party (believing that the “When” condition will happen with the given probability) and the optional trusted party (promising that the “When” condition will happen), and includes an arithmetic expression. The expression is evaluated during run-time and results in a single number (probability) in the interval [0, 1]. For example, a situation that leads to the utility policy assertion with future sales of AU$700 could be assigned probability of 0.7, meaning that there is 70% chance that this situation will happen. It is also possible to assign probabilities that different estimates of the same business value metric will be correct (the sum of these probabilities has to be 1). An example is when satisfaction of some goal policy assertion GPA1 is associated with 2 utility policy assertions UPA1 and UPA2 and there are 2 probability policy assertions: PP1 specifies that there is 0.6 (60%) probability that UPA1 will be the correct estimate, while PP2 specifies that there is 0.4 (40%) probability that UPA2 will be the correct estimate.

One of the goals of BDIM is to model relationships between technical QoS metrics and business value metrics. In WS-Policy4MASC this is done through specification of utility (and
probability) policy assertions for various conditions modeled in “When” constructs. For example, satisfaction/non-satisfaction of a goal policy assertion or execution of an action policy assertion generates an event, which can be used in “When” constructs for utility policy assertions. Technical QoS metrics are specified in “MonitoredDataItem” constructs that can be used within any expression, such as Boolean expressions in goal policy assertions and filtering conditions of “When” constructs, or arithmetic-with-unit expressions in utility policy assertions (e.g., for modeling of pay-per-volume prices). In the motivating example from Figure 1, Party B has a contractual guarantee to provide 99% availability every day, with AU$10 penalty for not meeting this guarantee. This is modeled in WS-Policy4MASC by defining “AvailabilityPerDay” QoS metric as a “MonitoredDataItem”, specifying in the Boolean expression of a goal policy assertion (GPA-B) that “AvailabilityPerDay” has to be >= 99%, specifying in the “When” construct of the same goal policy assertion (GPA-B) that evaluation is performed every day at a particular time (e.g., midnight), defining an event (E1) triggered when this goal policy assertion (GPA-B) is not satisfied, specifying in the arithmetic-with-unit expression of a utility policy assertion (UPA-B) the constant “AU$10”, and specifying in the “When” construct of the same utility policy assertion (UPA-B) that accounting is performed whenever this event (E1) is triggered. Note that the “MonitoredDataItem” construct need not be used only for low-level technical QoS metrics – it can be also used for composite technical QoS metrics (e.g., average availability in a week), as well as for business process metrics that describe quality of business service, such as delivery time of a physical product (e.g., an ordered book) to a customer. Special action “MonitoredDataCollection” describes details of how, where, when, and by which party measurement or calculation of values for “Monitored DataItem” instances is performed. All these features (plus several others) enable definition of complex, multi-level dependencies between technical QoS metrics and business value metrics. While this increases complexity of WS-Policy4MASC, it is appropriate because in practice it is difficult to determine direct mappings between technical QoS metrics and business value metrics and multiple intermediate layers and probability calculations are needed often.

A common issue in specification of business value metrics is how to determine particular numerical quantities, especially for estimates of non-agreed business value metrics (e.g., that future sales will be AU$600) and for monetization of non-financial business value metrics (e.g., that monetary value for customer satisfaction in some situation is AU$200). Unfortunately, there is no simple answer to this issue, which is even more difficult than determining numerical quantities for technical QoS metrics. The process of determining these numerical quantities should be based on different inputs: expert opinion, market research and analysis, economic models, simulations, impact analysis, historical information from the same or similar systems, context in various historical situations, anticipated developments, and others. Historical information can be very useful, but must be used carefully because changes in context can make it inappropriate and it can be difficult to predict future context. Accounting/billing subsystems in management infrastructures can provide historical financial information. Further, business intelligence (BI) and business activity monitoring (BAM) solutions can be useful for determining historical information about both financial and non-financial business value metrics, as well as context. We had to leave integration of our research with solutions for estimation of numerical values for technical QoS and business value metrics for future work. We simply assume that humans who write WS-Policy4MASC files somehow determined appropriate numerical quantities.
As discussed earlier in the paper, business strategy is important for autonomic BDIM solutions and modeling of influence of diverse business strategies on adaptation decisions is a still underexplored research area. WS-Policy4MASC enables specification of business strategies in meta-policy assertions, as a means of deciding which among alternative adaptation approaches to take. Each adaptation approach is modeled as an action policy assertion. When several policy assertions can be applied in the same situation, but cannot or should not be applied together (or at least not at the same time), there is a policy conflict. In the example from Figure 1, “replace B with D” and “replace B with E” are 2 conflicting adaptation options. Our work enables autonomic choice of 1 “best” conflicting action policy assertion.

Meta-policies (policies about policies) are often used in policy-driven management (Lupu & Sloman, 1999) to decide which among conflicting policies to apply. WS-Policy4MASC uses meta-policy assertions to describe business strategy criteria for: a) calculating conflicting option’s overall business value metric that contains some (possibly all) of the above-mentioned 8 business value metric categories, and b) comparing these overall business value metrics to choose 1 “best” option that meets all specified constraints and results in the highest overall business value metric among the eligible options. WS-Policy4MASC also enables specification of priorities of policy assertions, but they are used only if overall business value metrics of several options are equal, because it can be difficult to a priori define appropriate priority levels. Compared with the approach that uses only priorities of the conflicting policies, as adopted in WS-ReL (Baresi, Guinea & Plebani, 2007) and other works, our approach to policy conflict resolution provides not only better alignment of service-oriented systems with business issues, but also more flexible handling of situations when priority of an option is not known beforehand (e.g., because it depends on some run-time information). It is central to the WS-Policy4MASC support for autonomic BDIM.

A WS-Policy4MASC meta-policy assertion references at least 2 conflicting action policy assertions and exactly 1 policy conflict resolution strategy, and has attributes describing the management party (executing our policy conflict resolution algorithm) and the party for which the overall business value metric is maximized (e.g., the provider Web service). The sub-elements and attributes of a policy conflict resolution strategy describe business strategy characteristics. Four among these attributes describe which properties (financial and/or non-financial, agreed and/or non-agreed) of business value metric categories should be used for calculation of the overall business value metric. For example, specifying “only non-financial” along with “both agreed and non-agreed” leads to the use of business value metrics in the categories “non-financial agreed benefit”, “non-financial agreed cost”, “non-financial non-agreed benefit”, and “non-financial non-agreed cost” – these 4 categories are relevant for maximization of customer satisfaction. Further, there is an attribute that determines whether probabilities of occurrence should be taken into consideration. Then, there is a sub-element specifying common currency into which all currencies are converted and an optional currency conversion service. An optional sub-element specifies (as an arithmetic-with-unit expression evaluated at run-time) the length of time during which future recurring payments (e.g., subscriptions) should be considered. Another optional sub-element specifies (as an arithmetic-with-unit expression) cost limits that must not be exceeded. For example, if a policy conflict resolution strategy says that the cost limit is AU$1000 and an adaptation option requires investment with the total cost (for all 4 cost business value metric categories) of AU$1500, this option will be discarded.

Finally, 2 sub-elements and 1 attribute of a policy conflict resolution strategy are used to specify tiebreaking. Tiebreaking is used when overall business value metrics of 2 or more
conflicting options are close enough so that the difference is considered negligible. For example, when one option provides profit of AU$500 and the other option provides profit of AU$499. In such cases, additional, previously excluded business value metric categories are used for calculation of new overall business value metrics. In the above example, if the original calculation of overall business value metrics did not take into consideration non-financial non-agreed benefits and costs, they can be added as tiebreakers. This can result in a situation when the new overall business value metric for the first option is AU$720, while for the second option is AU$750. Consequently, the second option will be chosen (in spite the fact that in the original calculations it was negligibly worse than the first option). WS-Policy4MASC supports up to 2 rounds of tiebreaking. An attribute of a policy conflict resolution strategy specifies whether the first tiebreaker will be additional business value metric categories along the financial/non-financial dimension (e.g., using “both financial and non-financial” instead of “only non-financial”) or along the agreed/non-agreed dimension. Two sub-elements of a policy conflict resolution strategy specify (as arithmetic-with-unit expressions) 2 limits of what should be considered “negligible” difference between overall business value metrics. One of these limits is for the financial/non-financial dimension and the other is for the agreed/non-agreed dimension.
Figure 3. Our algorithm for selection among conflicting action policy assertions
**Inputs:** A set of conflicting action policy assertions and the meta-policy assertion that describes their policy conflict resolution.

**Step 1. (Calculation of the overall business value metric for each conflicting action policy assertion)**

For each given conflicting action policy assertion:

1.1. Determine the set of all relevant utility policy assertions directly associated with execution of any of the actions contained in the current action policy assertion.

1.2. For each relevant utility policy assertion from Step 1.1:

1.2.1. If the policy conflict resolution strategy uses probabilities:

1.2.1.1. Find the probability policy assertion (if any) whose “When” condition is the same as for the currently processed utility policy assertion and calculate this probability (default: 1.0) – this is the probability that preconditions (e.g., correct execution of an action in the current action policy assertion) for the utility policy assertion will be satisfied.

1.2.1.2. Find the probability policy assertion (if any) whose “When” condition contains only the triggering event that the currently processed utility policy assertion is the correct estimate and calculate this probability (default: 1.0) – this is the probability that the estimate in the utility policy assertion will be correct.

1.2.2. For each business value metric in the current utility policy assertion:

1.2.2.1. Calculate the monetary amount and convert it to the common currency.

1.2.2.2. If the policy conflict resolution strategy uses probabilities, then multiply the converted monetary amount from Step 1.2.2.1 with the probabilities from Steps 1.2.1.1 and 1.2.1.2.

1.3. Sum all converted monetary amounts (for all business value metrics in all relevant utility policy assertions) determined in Step 1.2.2.2 (1.2.2.1, if probabilities not used), separately for each of the 8 business value metric categories – these are the 8 category-specific business value metric sums.

1.4. If the policy conflict resolution strategy specifies a cost limit constraint:

1.4.1. Sum all results from Step 1.3 for the 4 “cost” business value metric categories.

1.4.2. If the sum from Step 1.4.1 is higher than the given cost limit,

Then eliminate this action policy assertion.

Else sum all results from Step 1.3 for the business value metric categories relevant for the given policy conflict resolution strategy – this is the overall (summary) business value metric of the currently processed action policy assertion.

**Step 2. (Ordering of action policy assertions that satisfy all given constraints)**

If all conflicting action policy assertions were eliminated,

Then notify humans that no option satisfies all given constraints and exit this algorithm.

Else list the remaining action policy assertions in the descending order from the highest to the lowest result from Step 1.4.2.

**Step 3. (Tiebreaking)**

If the policy conflict resolution strategy specifies the first tiebreaker, then:

3.1. Determine the tiebreaking limit for the first tiebreaker.

3.2. Shorten the list from Step 2 to include only those conflicting action policy assertions for which the difference from the first element of the list (with the highest result from Step 1.4.2) is less or equal to the tiebreaking limit from Step 3.1.

3.3. For each short-listed conflicting action policy assertion, recalculate the overall business value metric (from Step 1.4.2) by summing all category-specific business value metric sums (from Step 1.3) that are relevant for the initial policy conflict resolution strategy (used in the “else” part of Step 1.4.2) or the first tiebreaker.

3.4. Re-order the short-listed action policy assertions in the descending order from the highest to the lowest result from Step 3.3.

3.5. If the policy conflict resolution strategy specifies the second tiebreaker, then repeat Steps 3.1-3.4 for the second tiebreaker, but use the results from Step 3.3 (instead of Step 1.4.2) and the list from Step 3.4 (instead of Step 2).

**Step 4. (Selection of the action policy assertion best from the business viewpoint)**

If 2 or more action policy assertions at the beginning of the list remaining from Step 3.5 have the same results for the overall (summary) business value metric,

Then use priorities of these action policy assertions to select the action policy assertion with the highest priority (if several action policy assertions have the same priority, use the one most recently added to the policy repository).

Else, select the first element of the list remaining from Step 3.5 – it will have the highest overall business value metric satisfying all given constraints and tiebreaking rules.

**Outputs:** The action policy assertion that should be executed.
Our algorithm for selection of 1 “best” option among conflicting action policy assertions is outlined in Figure 3. Here, “best” means “highest overall business value metric, while meeting all constraints” and depends on which combinations of the 8 business value metric categories are used in the calculations. The algorithm is a refinement and improvement of the high-level algorithm presented in (Tosic, 2008). Versions of this algorithm were successfully prototyped, first in the original MASC middleware and recently in the new MiniMASC middleware we are developing. Further details, explanations, and illustrations of this algorithm will be given in our forthcoming publications.

CONCLUSIONS AND FUTURE WORK

Due to the rapidly increasing complexity of service-oriented (and other IT) systems, their management is becoming a serious problem. On one side, there is a need to perform management with minimal human intervention, i.e., autonomically. On the other, there is a need to align management decisions with company’s business strategies and maximize relevant financial and non-financial business value metrics, by using BDIM approaches. Support for autonomic BDIM requires novel modeling of policies that guide management, additional monitoring solutions (e.g., if customer satisfaction is strategically important, it should be monitored), innovative decision making algorithms that maximize business value metrics important for the used business strategies, as well as management tools that provide the corresponding policy-driven monitoring and control. There are numerous challenges, particularly when non-financial business value metrics and corresponding business strategies are considered. In this paper, we mainly discussed the difficult challenges associated with: a) modeling of financial and non-financial business value metrics and their relationships with technical QoS metrics, and b) modeling of business value metric maximization strategies and their impact on adaptation decisions. We presented WS-Policy4MASC solutions to these challenges, as well as decision making algorithms that use this additional information for policy conflict resolution.

WS-Policy4MASC extends a widely-used industrial standard, WS-Policy, with information necessary for run-time management, including the unique support for autonomic BDIM. The specifications of diverse financial and non-financial business value metrics and business strategies that guide business-value driven selection among alternative control (adaptation) actions are the main distinctive characteristics and contributions of WS-Policy4MASC. WS-Policy4MASC also supports other management aspects, such as fault management and maximization of technical QoS metrics. It has built-in constructs for specification of a wide range of adaptations and events common in management of service-oriented systems and business processes they implement. The WS-Policy4MASC language design and the algorithms using this policy information were successfully prototyped in the MASC middleware and the tools for annotation of UML models with WS-Policy4MASC policy assertions and run-time monitored data. In addition to demonstrating feasibility, we examined their correctness, expressiveness, effectiveness, and usefulness on realistic examples.

Some of the common issues for practical adoption of management solutions in general, and particularly autonomic BDIM solutions, are complexity and overhead of management activities. When WS-Policy4MASC policy assertions are written, in addition to writing policy assertions that exist in the traditional QoS-driven management, there is also a need to model business value metrics, determine numerical values for them, and map business strategies into policy conflict resolution strategies. At run-time, there can be an additional overhead of monitoring.

Some of the common issues for practical adoption of management solutions in general, and particularly autonomic BDIM solutions, are complexity and overhead of management activities. When WS-Policy4MASC policy assertions are written, in addition to writing policy assertions that exist in the traditional QoS-driven management, there is also a need to model business value metrics, determine numerical values for them, and map business strategies into policy conflict resolution strategies. At run-time, there can be an additional overhead of monitoring.
supplementary information (e.g., non-financial business value metrics) and executing the business-driven policy conflict resolution algorithm. In some situations, all this complexity and overhead will not provide sufficient benefits, because the traditional approaches maximizing technical QoS metrics will provide good-enough results. On the other hand, when impact of a wrong adaptation decision (e.g., in the example from Figure 1, choosing D when the business needs require choosing E) can have significant (or even catastrophic) negative business consequences, so the complexity and overhead of the presented solutions become worthwhile. Therefore, we treat the presented autonomic BDIM solutions as an optional layer beyond the traditional IT system management approaches. If they are unnecessary and too complicated for particular management situations, they should not be used.

While many challenges remain for research of autonomic BDIM and WS-Policy4MASC can be improved in different ways, several items have priority in our ongoing and future work. Most importantly, we continue with implementation of our solutions in the new MiniMASC middleware and plan additional evaluations, hopefully on real-life scenarios with industrial partners. We also work on facilitating practical adoption of WS-Policy4MASC, e.g., by improving the tools for translation of annotated UML models into WS-Policy4MASC files. For practical adoption of autonomic BDIM solutions, it would be also beneficial to relate the existing mechanisms for specification of business strategy with the mechanisms for specification of policies in autonomic BDIM solutions. Therefore, we have developed and currently evaluate integration of Business Motivation Model (BMM) hierarchies of ends (e.g., goals) and means (e.g., strategies) into WS-Policy4MASC policy assertions. Apart from making adoption by human managers easier, this extension broadens the range of strategies that can be modeled in WS-Policy4MASC. This work will be described in a forthcoming publication. We have also been working on addressing situations when multiple adaptation decisions can be chosen concurrently (e.g., different decisions for different classes of consumer), instead of choosing only 1 among several options. Integration with business intelligence and/or business activity monitoring solutions and design-time analysis of policies to preventively detect and resolve policy conflicts are some of the topics that we plan to study in the future.

ACKNOWLEDGMENTS
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.

Abdelkarim Erradi and Piyush Maheshwari contributed to the development of early versions of WS-Policy4MASC (up to the version 0.8), within the research project “Building Policy-Driven Middleware for QoS-Aware and Adaptive Web Services Composition” sponsored by the Australian Research Council (ARC) and Microsoft Australia. Rasangi Pumudu Karunaratne contributed to the development of WS-Policy4MASC version 0.9. Nahid Ebrahimi Nejad implemented and tested the described version of the algorithm for selection among conflicting action policy assertions.

REFERENCES


Vladimir Tosic ([vladat@computer.org](mailto:vladat@computer.org); [http://www.nicta.com.au/people/tosicv](http://www.nicta.com.au/people/tosicv)) is a Researcher at NICTA in Sydney, Australia; a Visiting Fellow at the University of New South Wales, Sydney,
Australia; and an Adjunct Research Professor at the University of Western Ontario, London, Canada. He previously held several positions in industry and academia in Europe, Canada, and Australia. He completed his Ph.D. degree at Carleton University, Ottawa, Canada. Dr. Tosic received many academic awards, including the 2001 Upsilon Pi Epsilon / IEEE Computer Society Award for Academic Excellence and several Natural Sciences and Engineering Research Council of Canada (NSERC) awards and grants. Most of his peer-reviewed papers were in the area of management of service-oriented systems and his current research is focused on autonomic business-driven management. Additionally, he presented several conference tutorials about these topics, co-organized several international workshops, and serves as an editorial board or program committee member of many international journals, conferences, and workshops.