

On Modeling and Maximizing Business Value for Autonomic Service-Oriented Systems

Vladimir Tasic^{1,2,3}

¹ NICTA* and ² University of New South Wales, Sydney, New South Wales, Australia

³ University of Western Ontario, London, Ontario, Canada

vladat@computer.org

Abstract. The existing Web service management solutions are almost exclusively focused on optimization of technical QoS metrics (e.g., availability). However, financial (e.g., profit) and other business value metrics (e.g., customer satisfaction) are more important than technical QoS metrics. We discuss the challenges of business value modeling and maximization in autonomic (e.g., self-healing) Web service management systems. In particular, we point out the need to model business strategies (e.g., maximizing customer satisfaction) and diverse types of business values (not only financial ones). To illustrate how some of these challenges can be addressed, we present the modeling of diverse business values and business strategies in our language WS-Policy4MASC and the corresponding algorithms for business value maximization in our middle-ware MASC (Manageable and Adaptable Service Compositions).

Keywords: Web service, business value, Web service management, policy-driven management, business-driven IT management, autonomic computing

1 Introduction

Management of IT (information technology) systems, including Web service systems, is the process of their monitoring and control to ensure correct operation and achieve maximal benefits. Management activities include: discovering and fixing problems (e.g., faults, performance problems), maximizing quality of service (QoS), accommodating change, accounting of consumed resources, billing of consumers, and many others. QoS is a broad concept that encompasses various extra-functional properties (e.g., performance, availability) that describe how well a system performs its functions. For example, performance is often expressed in QoS metrics, such as average response time (and/or throughput) in the last 24 hours. Monitoring of Web service systems includes measurement or calculation of QoS metrics, evaluation of requirements and guarantees to discover problems (if any), calculation of prices/penalties to be paid, and accounting. Control to meet guarantees and adapt to changes includes re-

* 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 Centres of Excellence program.

configuration of an individual Web service, re-negotiation of contracts between Web services, and re-composition of Web services. In the vision of autonomic computing, IT systems manage themselves using configurable policies, with minimal human intervention. For example, such self-management to fix problems is referred to as self-healing. A policy formally specifies a collection of high-level, implementation-independent, operation and management goals and/or rules in a human-readable form.

An important question for operation of autonomic systems is: When an autonomic (e.g., self-healing) system is confronted with a situation in which several groups of control actions can be taken, how does it decide the course of action? The answer to this question influences not only control activities, but also monitoring activities (because different decision making strategies require different information), as well as modeling of policies that guide management activities. We argue that such decision making should be driven by maximization of business value. The focus of this paper is a discussion of challenges to achieve such decision making. We also summarize our own research to illustrate how some of these challenges can be addressed.

Business value is a broad concept that refers to any measure of worth of a business entity [1]. It includes financial aspects, such as prices, monetary penalties, earned income, costs, profit, and return on investment (ROI). However, it also includes other aspects important for business operations, such as number of customers, market share, and customer satisfaction. Since it describes how well a (business) system performs its function, it can be viewed as a special case of QoS (in the broad sense), although QoS traditionally refers to technical QoS metrics (e.g., response time, availability). Instead of the term ‘business value’ some authors use the terms ‘key performance indicators (KPIs)’, ‘business performance metrics’, ‘business QoS’, or ‘quality of business (QoBiz)’. There are substantial similarities between these concepts, so they can be considered nearly synonymous (discussion of their subtle differences is outside the scope of this paper). Since the primary goal of every business is to produce business value for its stakeholders (notably owners, shareholders), we find the term ‘business value’ the most appropriate. It is already widely used in the business community.

The past Web service management solutions were mostly focused on functional correctness and optimization of technical QoS metrics. However, business value metrics are more important to 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 [2]. For example, higher availability need not lead to increases in business profits. The goal of business-driven IT management (BDIM) [2, 3] 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. Business-driven autonomic computing was identified in [2] (pp. 23-24) as an area of open research challenges.

Some of the past Web service management solutions (cf. [4]) provided a simple treatment of financial business values (e.g., prices), but without considering other business values (e.g., customer satisfaction, market share). While financial business values are important for all companies, they are not the only measures of business worth. Human managers use many *business strategies* that differ (among many other aspects) in how they prioritize business values in different time frames (e.g., long-

term vs. short-term). Business strategies are a major differentiator of companies in a market, so many diverse business strategies will exist in the Web service market.

The following example motivates our research and illustrates the importance of the relationship between a business strategy, business values, and Web service control (self-healing) actions. There are 2 companies that use the same Web service composition, but one has business strategy “exceptional customer satisfaction”, while the other has the strategy of maximizing short-term profit from contracts. Suddenly, a performance problem (QoS reduction) with one Web service in the composition occurs, so the overall customer satisfaction suffers. There is a replacement Web service, but it is very expensive and leads to high financial losses in the short term, although it keeps customer satisfaction very high. If only financial business values are modeled and processed, an autonomic Web service management system will choose the option of “doing nothing”, under which customer satisfaction suffers, but short term losses are not big. While this is appropriate for the second company, it is not appropriate for the first one. For the first company, it is appropriate to choose the expensive temporary replacement and experience “short term pain” (the financial losses) for “long term gain” (reputation about the “exceptional customer satisfaction”).

This paper is organized in the following way. This section introduced the area of our work. In the following section, we point out the main achievements in the past related work. In Section 3, we discuss several additional challenges in business value modeling and maximization that motivate our research and influence our solutions. Then, in Section 4, we overview our own solutions for business value modeling and maximization, built into our language WS-Policy4MASC and middleware MASC (Manageable and Adaptable Service Compositions). In the final section, we summarize conclusions and outline items for future work.

2 Related Work

A prerequisite for performing IT system management activities is existence of a machine processable and precise format for specification of the monitored requirements, guarantees, capabilities, and control actions. In the last several years, a number of languages were published to address some aspects of this need for Web service QoS management. For example, [4] provided a systematic analysis of the Web Service Level Agreement (WSLA), the Web Service Offerings Language (WSOL), the Web Services Agreement Specification (WS-Agreement), the Web Services Policy Framework (WS-Policy), DAML-QoS, and several other languages. In the meantime, many other languages appeared. Many of these languages provide detailed models of QoS for Web services (notable exceptions are WS-Agreement and WS-Policy, but they allow development of QoS extensions). Some of them support describing control actions (this support is stronger for control of business processes implemented with Web services than for control of individual Web services). Most of the languages enable simple specification of various prices (e.g., subscription, pay-per-use) and some also allow for monetary penalties for not meeting guarantees. These are financial business values. It is implicitly assumed that the business strategy is to maximize profit, calculated as a sum of prices and penalties. However, these languages did not

provide support for detailed specification of business values (particularly non-financial ones) and business strategies that maximize them.

Many of the above-mentioned languages are accompanied by Web service management infrastructures. There are also additional research projects and industrial products offering such management frameworks. The overview in [4] concluded that QoS of Web services was well-researched, but that (in spite of some support) further work on control actions (including self-healing actions) was needed. In the meantime, frameworks with improved support for control actions, including elements of self-healing, appeared. Examples are Dynamo [5] and JOpera [6]. Another recent stream of research studies how to diagnose which services caused problems in a complex composition. However, much remains to be researched and achieved in the area of autonomic service-oriented computing, particularly on self-healing. One of the main under-explored challenges is support for business value driven self-healing. Some Web service management systems (particularly commercial ones) contain accounting/billing subsystems for monitoring of financial business values. They use contractual financial obligations (specified in policies or service level agreements – SLAs) to calculate monetary amounts to be paid. While such subsystems are certainly useful, significant additional work is needed on monitoring of non-financial business values and, particularly, on business-driven decision making for control actions.

Business value is much more than prices and penalties modeled in the current Web service QoS models. It has been shown in business literature [7] that modeling of non-financial business values (in addition to financial ones) is very beneficial for long-term strategic management of companies. The balanced scorecard (BSC) [7] is a well-known approach that goes beyond financial metrics to represent well-being of a company. It contains multiple company-specific business values organized along 4 standard perspectives: financial, customer, internal business processes, and learning & growth. Unfortunately, it is aimed at humans and is not completely suitable for direct use in autonomic IT system management software (e.g., due to unclear mapping from high-level business management to run-time IT system management activities). Although the concept of a balanced scorecard is not directly reusable in this context, it influenced several business-driven IT management works (including our own).

The Management by Business Objectives (MBO) [8] is an application of the balanced scorecard ideas for business-driven IT management. Its information model contains objectives, key performance indicators (KPIs), and perspectives. Importance weights are assigned to objectives and perspectives. Aline is the corresponding engine that calculates alignment between different actions and objectives, defined as likelihood of meeting the objectives. While we praise MBO achievements, we note that in practice it can be difficult to determine weights that are precise enough. Furthermore, this work did not provide detailed modeling of business values (KPIs).

This paper is related to our previous publications on the WS-Policy4MASC language [1], the MASC middleware [9], and UML (Unified Modeling Language) profiles for WS-Policy4MASC. It complements these past papers (particularly [1]) by putting the focus on challenges in business value modeling and maximization, while the WS-Policy4MASC language implementation is only an illustration of possible solutions. In addition, some of the presented modeling solutions (e.g., specification of probability of events) and the algorithm for selection of the option with highest business value were not published previously.

3 Some Challenges in Business Value Modeling and Maximization

In this section, we discuss several challenges in business value modeling and maximization that motivate our research and influence our solutions. (There are also other challenges, but we had to make a selection due to the limited space in this paper.) While the challenges related to modeling of business strategies maximizing business values are the main motivation for our research, we start by discussing modeling of business values in order to facilitate understandability of the discussion.

1. Modeling of diverse business values – both financial and non-financial. To maximize a business value (e.g., customer satisfaction), it has to be formally modeled. There are many different business values and their characteristics require different considerations, discussed next. It should be possible to specify more than one business value (e.g., with different characteristics or different probabilities of occurrence) for one situation (e.g., replacement of a Web service). It is also necessary to specify for which party (e.g., provider Web service, consumer) business value is calculated, since different parties will likely get different business values from the same situation.

1.1. Explicit description of characteristics of business values. Just specifying names of business values is not sufficient for autonomic run-time processing. Description of business value characteristics is needed to decide which business values to maximize. Examples of such characteristics are: whether a business value is financial or not, whether it is contractually agreed upon to or it is a prediction of future events, and whether it is a benefit or cost. One approach is to build into a modeling (specification) language several business value types with implicitly stated characteristics (e.g., profit is financial, customer satisfaction is non-financial). This is not flexible and scalable as the number of business values increases. A much better approach is developing ontologies of business values (and business strategies), describing in detail various characteristics in an explicit machine processable format, such as a Semantic Web language. A drawback of using ontologies is that, depending on the level of formality and precision, their overhead can be high. Instead of full, formal ontologies in powerful Semantic Web languages, limited (but precise) descriptions in simpler formats might suffice in some cases. Whether or not Semantic Web languages are used, it is necessary to systematically examine which characteristics of business values are relevant in various business strategies, build these characteristics into models, and support run-time monitoring of business values. Several interrelated issues without clear-cut answers should be considered for this modeling. Some of these issues are listed next.

1.2. Monetization of non-financial business values. One such issue is whether to monetize non-financial business values, i.e., represent them with financial (monetary) estimates. An example is stating that the estimated value of 80% customer satisfaction is AU\$1000 per day. Monetization is useful because it makes processing of business value information significantly easier. For example, if a decision is influenced by previously agreed contractual agreements, impact on customer satisfaction, and impact on market share, then monetization helps express all 3 factors in the same monetary units and perform calculations to determine the total (estimated) business value of different options. Monetization requires modeling information about used currencies, e.g., in ontologies. Irrespective of whether ontologies are used, there is a need for a link to up-to-date information about currency conversion. While monetization simplifies processing, it also has caveats and is not always appropriate. Most importantly, it

is very difficult to obtain precise estimates due to the randomness of involved processes. Point estimation (e.g., “\$1000 per day” in the above example) is simple for processing, but does not capture important information about confidence in different estimates. Range estimation (e.g., “from \$800 to \$1200 per day with uniform probability distribution”) is more precise, but it is more difficult to process and introduces additional problems (e.g., determining probability distribution). Irrespective of which estimation type is used, estimates should be based on different inputs, such as expert opinions, simulations, historical information, context in various historical situations, and anticipated developments. Historical information about financial outcomes can be obtained from the accounting/billing subsystems present in many management infrastructures. On the contrary, obtaining historical information and monitoring of non-financial business values depends on the business value type and remains a challenge. (Business intelligence tools can help in this regard, but are not enough.) Additionally, determining which past situations have context similar to anticipated future situations can be very difficult. Furthermore, if more than one non-financial factor contributed to a past situation, differentiating between their impacts can be very difficult. For example, if both customer satisfaction and market share (somewhat, but not completely, dependent on customer satisfaction) contributed to past financial outcomes, it might be difficult to estimate how customer satisfaction (on its own) impacts profit.

1.3. Modeling of possible (but not certain) business values. There is another type of uncertainty, which affects both financial and non-financial business values. It is about probability that future events will happen. Namely, future events of relevance for business value modeling can be of (at least) two types: contractual agreements and possible future events. For the latter category (but under some circumstances also the former category), the probability of happening is not 100% and the business value model should enable specifying this probability. This probability can be used for representing risks and trust. For example, a relationship could exist that “when Web service X is replaced with a Web service that is 20% faster but 10% more expensive, there is 80% probability that average customer satisfaction will raise 5% relative to its current value, but there is also 20% probability that average customer satisfaction will fall 2% relative to its current value”. In many cases, a probability distribution function should be specified instead of one probability. Determining probabilities (and, particularly, probability distributions) is complicated. Mechanisms that could be used are similar to the above-mentioned mechanisms for monetary estimation of non-financial business values. For non-financial business values, determining probability of future events is additional to determining monetary estimates (if they are used). When both monetization estimates and occurrence probabilities are determined using historical information, distinguishing between them might also be complicated.

1.4. Specifying time value of money. Time value of money affects modeling of both financial and monetized non-financial business values. Its example is that getting a \$100 payment today is of higher worth than getting 10 monthly payments of \$10, due to inflation and interest rates. Modeling of business values for autonomic business-driven management should accommodate this phenomenon by keeping information about time when payments are to be made and interest/inflation rates, and by applying financial accounting formulas for calculating net present value of future payments.

II. Modeling of business strategies that maximize business values. One of the main challenges for business-driven management of Web services (and other IT systems) is

how to model diverse business strategies. In this paper we focus on aspects related to maximization of business values (e.g., “maximize profit”, “exceptional customer satisfaction”). We examine scenarios when there are several options and business values associated with each option are summarized and compared to determine which option (only 1) produces the highest business value. In such scenarios, business strategies differ in how the comparison is made, which business values are considered in the comparison, and several other characteristics. While we do not consider here all possibilities (e.g., combining multiple options), we identify the main issues.

II.1. Explicit description of characteristics of business strategies. This issue is analogous to Challenge I.1. For autonomic run-time processing, it is not enough to specify only names of business strategies or build into a modeling language a few business strategies with implicitly stated characteristics. Developing ontologies of business strategies is better, but their overhead should be acceptable. Some of the business strategy characteristics and issues to be considered are discussed next.

II.2. Limiting temporal scope of relevant business values. We previously mentioned that we study how autonomic Web service management systems should choose between several available options. Here, an option is a group of actions to be taken (e.g., to correct the problem). Each of these actions is associated with zero, one, or more business values. These business values should be considered when options are compared. However, after the actions are undertaken, they cause further events that can have additional business values, these new events can cause further events (also associated with business values), and so on. For example, if a replacement of a Web service costs AU\$10 but has 5% higher response time than the original Web service, this higher response time could raise overall Web service composition response time over a contractually guaranteed limit, causing payment of a penalty of AU\$50. The business strategy model should specify how far in the future (in terms of elapsed time and/or chain of events) is relevant for comparison of options. Of course, the simplest approach is to consider only immediate consequences (AU\$10 in the previous example), but this is not always appropriate. Calculation of business values associated with future events can be complicated, particularly when these events have various probabilities of occurrence. Some of the approaches that could be considered in this regard are discrete event simulation, dynamic programming and other operations research methods, and planning techniques from artificial intelligence.

II.3. Calculating the overall business value of an option. When the overall business value of an option is calculated, one of the decisions is whether to use monetization to ease summation and comparison. Another decision is what to do with monetary amounts in different currencies. It is usually appropriate to convert them into a single currency at run-time, using up-to-date currency conversion information. Future payments are challenging in this regard, due to the uncertainty of future currency conversion rates. In most cases it is appropriate to convert all future monetary amounts into net present value using standard formulas from financial accounting (but addressing fluctuation of currency conversion rates). After these preparatory activities, all business values associated with a particular option should be summed. However, even when monetization is used, business strategy might state that some types of business value should be summed separately. For example, it can be appropriate to distinguish between contractually agreed financial business values and possible non-financial business values. This means that in many situations the overall business value is actu-

ally a set of summary business values with various characteristics. A business strategy might specify that some business value types (e.g., all that are not contractually agreed) are not of interest for calculating the overall business value. It could also specify that different business value types should be associated with different weights (e.g., that business values that are not contractually agreed should be multiplied with 0.3). The experience from many areas showed that while weights are a powerful general mechanism for combining amounts from different categories, they have a drawback that in practice it is often difficult to determine weights that are precise enough. Therefore, requiring specification of weights should be used with caution.

II.4. Comparing overall business values of various options. After summary values for each option are determined, they should be compared to determine which option provides the highest business value. However, a business strategy might specify constraints that eliminate some options. A common constraint is cost limit. For example, there might be an option that brings profit of AU\$500,000 after 1 year but requires paying total cost of AU\$25,000 in the present. If there is a cost limit of AU\$10,000 (due to limited money presently available), this option should be eliminated. A business value strategy could specify that only some business value types are to be used for comparison. It could also specify priorities of business values and conditions under which lower priority business value types should be considered. For example, it could state that contractually agreed summary business values for all options should be compared first and then, if the difference between 2 (or more) best options is smaller than the AU\$10 limit, all other business values associated with these “short-listed” options should be added to determine the best option.

4 Our Solutions in WS-Policy4MASC and the MASC Middleware

WS-Policy4MASC [1] is a powerful general language for formal specification of various policies for management of Web services and their compositions. Its main advantage over similar languages (e.g., [5]) is detailed specification of functional and non-functional business values and business strategies for their maximization. It is a domain-independent extension of the Web Services Policy Framework (WS-Policy) [10], an industrial standard by W3C (World Wide Web Consortium). In WS-Policy, a policy is a collection of policy alternatives, each of which is a collection of policy assertions. WS-PolicyAttachment defines generic mechanisms that associate a policy with subjects to which it applies, e.g., WSDL (Web Services Description Language) or WSBPEL (Web Services Business Process Execution Language) constructs. WS-Policy4MASC extends WS-Policy by defining 5 new types of policy assertions:

1. *Goal policy assertions* specify requirements and guarantees to be met in desired operation (e.g., response time of an activity has to be less than 1 second).
2. *Action policy assertions* specify actions to take if certain conditions are met (e.g., a goal policy assertion was not satisfied). Example actions are removal, addition, replacement, skipping, and retrying of a sub-process or an activity (Web service).
3. *Utility policy assertions* specify expressions for calculating financial and monetized non-financial business values assigned to particular situations (e.g., non-satisfaction of a goal policy assertion, execution of an action, or another event).

4. *Probability policy assertions* specify probabilities that particular situations will occur. They can be used for specification of trust (in various parties) and risks.
5. *Meta-policy assertions* specify which action policy assertions are conflicting and which business value maximization conflict resolution strategies should be used.

These 5 policy assertion types can be used as any other policy assertion in WS-Policy. Apart from them, WS-Policy4MASC defines a number of supporting constructs, including definitions of: times when policy assertions should be evaluated (or executed, calculated), events, states, schedules, expressions; as well as references to external ontologies defining semantics (e.g., of QoS metrics). These supporting constructs specify details necessary for actual run-time Web service management activities and overcome some of the limitations of WS-Policy (e.g., imprecise semantics of policy assertions' effects on policy subjects). The precise definitions, meanings (semantics), and examples of these policy assertion types and supporting constructs were given in our past publications, e.g. [1] (This paper introduces some improvements, e.g., probability policy assertions.) We focus here on utility and meta-policy assertions, as well as the policy conflict resolution algorithm developed for the MASC middleware.

A *utility policy assertion* specifies situations in which the policy assertion should be applied, the management party (performing calculation and accounting), the beneficiary party (for which business values are calculated), and the list of one or more business values. Each *business value* contains an arithmetic-with-unit expression and a set of attributes and elements for business value characteristics (cf. Challenge I.1):

- *Arithmetic-with-unit expression*: This element can contain variables (e.g., for relations such as $\text{PriceB}=0.8*\text{PriceA}$, pay-per-volume prices) and function calls. It is evaluated at run-time to produce one number with a currency unit. The sign ('+' or '-') of this number determines whether the business value is a benefit or cost for the beneficiary party. Currency units can be associated with ontological definitions and linked to currency conversion services. All non-financial business values are represented with monetized point estimates (cf. Challenge I.2). We chose this for simplicity. Furthermore, it is also assumed that all future monetary values are already converted into their net present value (so Challenge I.4 is not addressed).
- *Attribute IsTangible*: Whether a business value is financial (WS-Policy4MASC uses 'tangible') or non-financial ('intangible') is specified in this Boolean attribute.
- *Element IntangibleUtilities*: This optional array element enables specifying which aspects (e.g., "customer satisfaction") a non-financial business value represents. However, this information is currently used only for documentation purposes. It is not yet used in our business value maximization algorithms, to avoid difficulties in monetization that were discussed at the end of Challenge I.2.
- *Attribute PayingParty*: This optional attribute specifies the paying party.
- *Attribute IsAgreed*: If paying party is specified, this optional Boolean attribute specifies whether the business value is a contractual agreement or only a possibility (cf. Challenge I.4). Absence of a paying party implies that the business value is a possibility (so the value of IsAgreed is irrelevant) and that the payer cannot be pinpointed. Note that the probability that an event will happen is specified in a probability policy assertion. For simplicity, a probability policy assertion specifies an arithmetic expression that results in a single number in the interval [0, 1] (e.g., 0.8).
- *Element RepeatPeriod*: This optional arithmetic-with-unit element is used for recurring payments (e.g., subscription prices). By default, there is no recurrence.

1. For each conflicting action policy assertion (given in the meta-policy assertion), determine the set of relevant utility policy assertions, across all executed actions
2. For each business value in each relevant utility policy assertion:
 - a. calculate monetary amount and convert it to a common currency
 - b. if the meta-policy assertion uses probabilities of occurrence, calculate corresponding probability (given in a probability policy assertions) and multiply the monetary amount with this probability
3. For each conflicting action policy assertion, sum the results for all business values in all relevant utility policy assertions
4. Eliminate summary results that do not satisfy constraints (e.g., cost limit), if any is specified in the meta-policy assertion
5. Compare these summary results using rules from the meta-policy assertion and select the action policy assertion with the highest sum

Fig. 1. Algorithm for Policy Conflict Resolution to Maximize Business Value

The above attributes classify business value types along 3 mutually orthogonal dimensions: i) financial or non-financial, ii) agreed or non-agreed, iii) benefit or cost. A business value type is a combination of 1 characteristic from each of these dimensions, so there are 8 business value types (e.g., ‘non-financial non-agreed benefit’).

An example is the business value where the arithmetic-with-unit expression contains the constant “+AU\$20” (the ‘+’ sign means that this is a benefit), `IsTangle` is “false”, `IntangibleUtilities` contains only “customer satisfaction”, `PayingParty` is empty (nil), `IsAgreed` is “false”, and `RepeatPeriod` is “1day”.

Policy conflicts arise when there are several valid policy assertions that individually satisfy conditions for application in a particular situation, but for some reason these policy assertions should not be applied at the same time. In the example from Section 1, “use the expensive replacement Web Service” and “do nothing” are two conflicting options. Randomly choosing an option leads to non-determinism. Using implicit priorities (e.g., order in the policy repository) is better, but it is not flexible. Using explicit priorities is somewhat more flexible, but it opens issues of defining and assigning priority levels and what happens when priorities are equal. Most importantly, none of the mentioned approaches maximizes business value. The most flexible approach suggested in the past literature is to use meta-policies (policies about policies). We adopted this approach, but focused on maximization of business value.

A `WS-Policy4MASC meta-policy assertion` references at least two conflicting action policy assertions and one policy conflict resolution strategy. It also specifies the management party (calculating and making decisions) and the party for which business value is examined (e.g., the provider). Elements and attributes of the policy conflict resolution strategy describe business strategy characteristics (cf. Challenge II.1).

Our *algorithm for policy conflict resolution* to maximize business value is outlined in Fig. 1. It results in selection of only 1 “best” conflicting action policy assertion, but supports different conflict resolution strategies. Here, “best” means “highest business value, while meeting all constraints”. The strategies perform the algorithm steps differently, depending on which combinations of above-mentioned business value types (e.g., ‘non-financial non-agreed’) they use to determine the “best” option.

STEP 1 determines relevant utility policy assertions. For simplicity, we adopted that only utility policy assertions directly associated with the actions listed in the conflicted action policy assertions are deemed relevant (cf. Challenge II.2). That is, our current implementation of the algorithm does not consider utility policy assertions associated with events triggered after these actions are executed. However, if some of

the relevant utility policy assertions contain recurring payments, these payments will be used, within the future period (e.g., 3 days) specified in the arithmetic-with-unit element *TimeLimit*. This period starts when the meta-policy assertion is evaluated.

STEP 2 calculates monetary amount for each business value in each relevant utility policy assertion. The calculation first evaluates the arithmetic-with-unit expression and converts the result into the common currency (the *CommonCurrency* element). If Boolean attribute *UseProbabilities* is “true”, then this monetary amount is multiplied by corresponding probability of occurrence (cf. Challenge I.3). (Matching between utility and probability policy assertions is performed to find the corresponding ones.)

STEP 3 performs summing of all business values relevant for each conflicting action policy assertion. However, this is not a simple sum! The summary result for one action policy assertion is actually a set of 8 numbers – 1 number for each business value type (cf. Challenge II.3). That is, all ‘non-financial non-agreed benefit’ business values are added together, but separately from business values of any other type (e.g., ‘financial non-agreed benefit’). All amounts are in the common currency.

STEP 4 eliminates action policy assertions that do not satisfy constraints. In our current implementation of the algorithm only the cost limit constraint is used (cf. Challenge II.4). The element *CostLimit* contains an arithmetic-with-unit expression that produces the cost limit (it can be +infinity), converted into the common currency. For each action policy assertion, all 4 cost business values (i.e., ‘financial agreed cost’, ‘financial non-agreed cost’, etc.) in the summary result are added. If these costs are greater than the cost limit, this action policy assertion is eliminated. If all action policy assertions are eliminated, the system notifies human administrators.

STEP 5 compares the remaining action policy assertions (cf. Challenges II.3 and II.4). All benefits and costs of the same other characteristics (e.g., ‘financial agreed’) are added. The Boolean attributes *UseTangible*, *UseIntangible*, *UseAgreed*, *UseNon-agreed* determine what business value types will be used in the first phase of the comparison. (Some combinations are not allowed.) For example, if *UseTangible* and *UseAgreed* are “true”, while *UseIntangible* and *UseNonagreed* are “false”, only ‘financial agreed’ benefits and cost are used. Then, the action policy assertions are ranked according to this sum. Next, tiebreaking limits in the arithmetic-with-unit elements *TiebreakerLimitForTangibility* and *TiebreakerLimitForAgreability* are calculated (in the common currency). Which tiebreaking is used first is determined by the Boolean attribute *TangibilityBeforeAgreability* (“true” means tangibility, i.e., financial/non-financial). All action policy assertions for which the distance to the top ranked one is less or equal the first tiebreaking limit are “short-listed” and if the “short list” contains more than 1 entry, additional business values (e.g., financial/non-financial, if tangibility is the first tiebreaker) are added to them. Then, the second tiebreaker is applied. If after the second tiebreaker, two or more action policy assertions have the same business value, explicit (and, if needed, implicit) priorities are applied.

For example, “customer satisfaction” can be modeled with non-financial, non-agreed, benefits and costs. To maximize customer satisfaction over 1 year, we set *UseIntangible* and *UseNonagreed* to “true”, *UseTangible* and *UseAgreed* to “false”, and *TimeLimit* to “1year”. To assure that total costs do not go over AU\$1000, we set *CostLimit* to this amount. To consider all agreed payments before financial non-agreed payments and to set this (first) tiebreaking limit to AU\$100, we set *TangibilityBeforeAgreability* to “false” and *TiebreakerLimitForAgreability* to “AU\$100”.

5 Conclusion

We strongly argue for business-driven autonomic management of Web services and business processes they implement. Such management should build upon, but go beyond, the past work on QoS management (including self-healing) based on technical metrics. In this paper, we discussed some challenges in the modeling of business values and corresponding business strategies. Many other challenges remain. We hope that the future work by the broader community will address these challenges.

Our own modeling solutions in WS-Policy4MASC and the corresponding algorithms offer original support for specifying both financial and non-financial business values and for business strategies maximizing them. We examined their usefulness on a set of realistic scenarios (e.g., a stock trading Web service composition containing about 10 Web services). We evaluated their feasibility by implementing them in the MASC Web service management middleware [9]. A MASC prototype was implemented in C# and .NET 3.5. Operation of most modules in MASC is configured through WS-Policy4MASC policies that describe what (e.g., QoS metrics) to monitor, which control actions to take to reconfigure (self-heal) the composition, and which business value types to maximize when there are several possible courses of action.

While we made some progress towards addressing the identified challenges, many open issues remain, both for our project and for the broader community. We are extending our solutions with modeling and processing of standard errors for estimates, and probability distributions. We plan work on discrete event simulation of a chain of actions, algorithms using the content of the attribute `IntangibleUtilities`, and algorithms using weights to combine amounts from different business value types.

References

1. Tosic, V., Erradi, A., Maheshwari, P.: WS-Policy4MASC – A WS-Policy Extension Used in the MASC Middleware. In: SCC'07, pp. 458--465. IEEE (2007)
2. Bartolini, C., Sahai, A., Sauve, J.P. (eds.): Proceedings of the Second IEEE/IFIP Workshop on Business-Driven IT Management, BDIM'07. IEEE (2007)
3. Casati, F., Shan, E., Dayal, U., Shan, M.-C.: Business-Oriented Management of Web Services'. CACM. October 2003, 55--60. ACM (2003)
4. Tosic, V., Hung, P.C.K.: Quality of Service (QoS) Specification and Management for XML Web Services. Tutorial at ICWS/SCC 2005. IEEE-CS (2005)
5. Baresi, L., Guinea, S., Plebani, P.: Policies and Aspects for the Supervision of BPEL Processes. In: CAiSE 2007. LNCS, vol. 4495, pp. 340--395. Springer (2007)
6. Pautasso, C., Heinis, T., Alonso, G.: Autonomic execution of Web service compositions. In: ICWS 2005, pp. 435-442. IEEE (2007)
7. Kaplan, R.S., Norton, D.P.: Using the Balanced Scorecard as a Strategic Management System. Harvard Business Review, HBS Press, Jan.-Feb. 1996, pp. 75-85.
8. Bartolini, C., Salle, M., Trastour, D.: IT Service Management Driven by Business Objectives: An Application to Incident Management. In: NOMS 2006, pp. 45--55. IEEE (2006)
9. Erradi, A., Maheshwari, P., Tosic, V.: MASC – .NET-based middleware for adaptive composite Web services. In: ICWS'07, pp. 99-108. IEEE (2007)
10. W3C Web Services Policy Working Group: Web Services Policy 1.5 – Framework. W3C Recommendation, 04 Sept. 2007, <http://www.w3.org/TR/ws-policy/>