On Integrating Trust into Business-Driven Management of Web Services and Their Compositions

1Chern Har Yew, 1,2Vladimir Tosic, 1Hanan Lutfiyya
1Department of Computer Science, The University of Western Ontario, London, Ontario Canada
2NICTA and The University of South Wales, Australia
chyew@csd.uwo.ca, vladat@computer.org, hanan@csd.uwo.ca

Abstract—This paper discusses why trust is an essential aspect for business-driven IT management, as well as how it impacts management of web services and their compositions. We summarize two ongoing projects which will enable the use of trust for business-driven management of web services.

Index Terms—Trust, Policy, Business-Driven Management, Web Services

I. INTRODUCTION

The goal of business-driven IT management (BDIM) [1] is to determine mappings between technical and business performance metrics and leverage them to make run-time IT system and/or service management (monitoring, control) decisions that maximize business value. While trust has been examined from various perspectives in computing (e.g. [3]), it has not yet been extensively studied from the perspective of its impact on business-driven IT management of service-oriented computing systems.

Trust is “the quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context” [3]. We can think of trust as an attitude by one entity about the dependability/capabilities of another entity. Trust is an important part of any social interaction. For example, we may choose to interact with people whom we consider to be trustworthy. When interacting with a person whom is not well trusted, we may employ different strategies, such as taking a more cautious approach. Just as we take our trust in a person into account when making interaction decisions, the concept of trust can also be used in service-oriented computing to improve decision making ability.

To be able to utilize trust for business-driven decisions, it is necessary to consider its properties, including the following (but also others): (1) Trust is a qualitative aspect that can be quantified in several different ways. The simplest, but most limited, way is binary. Another way is to have several discrete levels of trust, e.g., 0.0 (no trust), 0.25 (low trust), 0.5 (medium trust), 0.75 (high trust), 1.0 (complete trust); (2) Trust is subjective, i.e., trust is personal and should always be calculated from the trustor's point of view. As a result, A's trust for C may be different from B's trust for C; (3) Trust is not symmetric. If A trusts B, this does not mean that B trusts A; (4) Trust is not transitive. If A trusts B and B trusts C, this does not mean that A trusts C; (5) Trust is dynamic. The level of trust that a trustor has for a trustee may change over time based on the trustor’s experiences. As a result, any evidence used to perform trust calculation should be discounted based on time by giving outdated data less weight than evidence gathered in the present; (6) Trust is context-dependent. A trustor’s trust in a trustee may depend on the context in which the interaction takes place. For example, if A trusts B to fix his car, this does not mean that A trusts B to fix his computer. Trust in one context does not directly translate into trust into a different context.

Just as we take our trust in a person into account when making interaction decisions, the concept of trust can also be used in service-oriented computing to improve decision making ability. In Section 2 we discuss how this could be done. In Sections 3, 4, 5 we summarize our work in two ongoing projects in which we use trust for business-driven management of web services. The final section summarizes our conclusions.

II. USE OF TRUST IN BUSINESS-DRIVEN MANAGEMENT OF WEB SERVICES

Service-oriented computing is a computing paradigm that utilizes services as the basic construct in the development of distributed applications. Services can be distributed over the network. Applications are composed from the services independently of the hardware and operating system platforms. The emergence of standards for web services and web service compositions facilitates the development of large-scale applications developed in heterogeneous environments with multiple administrative domains.

The most obvious application of trust for business-driven management of web services is for selection of a service. There may be multiple instances of a service that can be selected and bound at run-time by a consumer (client). Furthermore, if a service instance does not satisfy the consumer, it can be replaced during run-time by another, similar, service. The metrics most often used in service selection are cost and performance that the service provider promises. This is often not sufficient for effective service
selection. A provider may find that it is in its interest to violate or prematurely terminate its contract with a consumer. Therefore, a service consumer should be able to model its trust in a service provider. There is some work (such as [5]) that models trust for web services and uses this trust information for web service selection. There is also work that it in the sense that it is based on perceptions of some community (as opposed to an individual trustor). Unfortunately, the way trust or reputation is calculated in the existing work does not fully correspond to the properties of trust identified in Section 1. Another issue is that the rules used for web service selection are not linked to business strategy. Consequently, we find that further work on using trust for business-driven selection of web services is needed.

Furthermore, there are many other situations in business-driven management of web services and, particularly, management of web service compositions in which information about trust could (or even should) be used in decision making. Assume a situation when a provider of a web service does not reply within the guaranteed response time. The consumer could continue waiting for the result or use a replacement web service. In analogous situations from real life (e.g., waiting for a late train), humans tend to use trust in making decisions – the higher the trust, the higher the probability that they will decide to wait. Thus, it would be beneficial to integrate into systems for business-driven management of web services rules specifying that if the consumer has high trust in the provider then the consumer is more likely to be patient if the provider is not behaving as expected. Another example is from the area of business processes implemented with web service compositions. Assume that a web service implementing one optional activity is down and there is no appropriate single replacement web service but the same results can be achieved with a composition of additional three web services. The decision whether to simply skip this activity (since it is optional) or use the replacement composition of three web services will depend on the combined trust in these three web services. Prior to our work, such issues have not been examined by the service-oriented computing or business-driven IT management communities.

III. ADDING TRUST TO WS-POLICY4MASC

WS-Policy4MASC is our XML (Extensible Markup Language) format for formal specification of monitoring and control policies for web service systems [3,6]. It is used in the Manageable and Adaptable Service Compositions (MASC) middleware [3]. Due to limited space, we will briefly summarize WS-Policy4MASC, while the details (including precise syntax and semantics) are available in the references. WS-Policy4MASC extends the Web Services Policy Framework (WS-Policy, standardized by W3C) by defining new types of policy assertion (goal, action, utility, and meta-policy) and additional necessary information (e.g., events and schedules). 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 one second). Action policy assertions specify diverse actions (e.g., replacement of a block of activities in a web service composition with another block of activities) to be taken if certain conditions are met (e.g., some goal policy assertions were not satisfied). Utility policy assertions specify expressions for calculating monetary amounts assigned to particular situations (e.g., non-satisfaction of some goal policy assertion, execution of some action, another event). These monetary amounts quantify not only financial, but also other business values (e.g., customer satisfaction). The represented type of business value is described through a combination of four mutually orthogonal properties (benefit or cost, tangible or intangible, agreed or possible, absolute or relative). A meta-policy assertion references at least two conflicting action policy assertions (i.e., in some situation any of them could be applied, but not more than one at a time) and one policy conflict resolution strategy to be used. These conflict resolution strategies maximize various combinations of business values (e.g., only agreed intangible benefits), representing various business strategies (e.g., “exceptional customer satisfaction”). For each of the conflicting action policy assertions, the algorithms built into the MASC middleware [3] sum all relevant business values associated with execution of this action (the relevant types of business value are determined from the used conflict
resolution strategy) and then compare these sums to determine which action policy assertion results in a higher business value.

The rich WS-Policy4MASC specification of business values in utility policy assertions and of connections between business values/strategies and management decisions in meta-policy assertions are the basis for business-driven management of web services (and their compositions) implemented in the MASC middleware. For example, choice between different web services can be represented with conflicting action policy assertions. Then, service selection is based on a business strategy (e.g., maximize revenue or maximize customer satisfaction) represented in meta-policy assertions. However, the version of WS-Policy4MASC described in [3,6] does not model trust. Therefore, we have been extending the WS-Policy4MASC and the MASC middleware with modeling of trust and use of trust information in decision making.

First, we added into WS-Policy4MASC a new (5th) type of policy assertion. Trust policy assertions specify trust that particular situations will happen (any situation for which a utility policy assertion can be specified, such as meeting a response time guarantee in a goal policy assertion). Fig. 1 extends the figure from [6] with TrustPolicyAssertion and its relationships. This new construct inherits from MASCPolicyAssertion the XML elements for specification of a unique ID (identifier), responsible management party (calculating trust), and activity status. In addition, it adds specification of a trustee, trustor, and the arithmetic expression for trust calculation. Trust is quantified as a continuous value in the interval [0.0, 1.0]. The numeric values for trust could be determined using the Trust Calculator described in the next section. The situation to which trust applies is described through a When construct (listing applicable states, triggering events, and filtering conditions). In this way, WS-Policy4MASC now enables assignment of trust (and utility) to various situations (not only selection of Web services), meaning that our work goes beyond past work. Note that, as for all WS-Policy4MASC policy assertions, association with specifications of the involved parties (Web services) is done through the standard WS-PolicyAttachment.

Second, we have added into meta-policy assertions a new optional Boolean attribute for specifying whether trust should be taken into consideration for conflict resolution.

Third, we have modified the MAS algorithms for summation of relevant business values during business-driven policy conflict resolution (see [3]). Now, if trust is considered, a utility value is multiplied by the corresponding trust value (for the same situation and when trustor in the trust policy assertion is the same as beneficiary party specified in the utility policy assertion). For example, if selection of web service A brings to B possible tangible absolute benefits of $10/day, but B only trust A 0.8, then the resulting utility (used for comparison with the other conflicting actions) for B is $8/day. If more than one trust value is associated with the same situation (when there are different ways to get into the same situation), their minimum is used. In this way, we enable use of trust in business-driven management of Web services and their compositions, thus going beyond past work.

Fourth, we are currently experimenting with the MASC middleware implementation of the algorithms and the WS-Policy4MASC extensions. While we do not yet use the Trust Calculator described below, this is an item for our future work.

IV. CALCULATING TRUST AND DESIGN

In order to support decision making based on trust, we have designed and implemented a service, called the Trust Calculator, which determines trust. There are different types of evidence that can be used to calculate trust. These include: (i) Any direct experiences that the trustor has with the trustee. Every time a trustor interacts with a trustee, evidence can be collected with regards to the interaction on a scale of one to five. The evidence could also be objective in nature. For example by verifying that the interaction is in compliance with the negotiated SLA; (ii) A recommender may provide recommendations on whom a trustor should trust. A recommender may lie or exaggerate its recommendations. As a result, when using recommendations in trust calculation, it is important for the trustor to be able to use recommendations from different sources; (iii) Reputation is a social quantity [5] that reflects the general consensus of members in a social network about a trustee. Unlike recommendations, the reputation of a trustee is usually shared within a social network [5] and as a result cannot be personalized. Even so, reputation is still important especially in cases when a trustor is new to an interaction environment. In such cases, reputation can be used as a starting point for making interaction decisions until some direct experiences and recommendations have been accumulated.

Any entity that provides a rating of a service is called a rating provider. It is the responsibility of a rating provider to gather evidence about a service’s trustworthiness so as to calculate a rating for the service. There are many different types of rating providers. For example, a service consumer can own a rating provider that is responsible for rating a service based on the consumer’s direct experiences. A service consumer can contact a rating provider owned by a recommender to obtain recommendations. A reputation system is also a rating provider that rates a service based on the consensus of members in a social network. By having all the rating providers implement a common retrieval API, this simplifies how a Trust Calculator can obtain ratings from the different types of rating providers.

V. DESIGN OF TRUST CALCULATOR

To support trust calculation we have design the Trust Calculator service. Our work differs from others (e.g., [1]) in that it provides trust calculation based on the properties specified in section 1 and it allows trust calculation using evidence from different sources of evidence (as opposed to just a single type of evidence such as reputation) as desired by the consumer. The Trust Calculator consists of a Trust