A Trust-Enabling Support for Goal-Based Services

Luiz Olavo Bonino da Silva Santos, Luís Ferreira Pires, Marten van Sinderen
Centre for Telematics and Information Technology, University of Twente, P.O. Box 217, 7500AE, Enschede, the Netherlands
{l.o.bonino,l.ferreirapires,m..j.vansinderen}@ewi.utwente.nl

Abstract

Service-Oriented Computing allows new applications to be developed by using and/or combining services offered by different providers. In several cases a service needs sensitive information from the clients in order to execute. The existence of a trust relationship between the client and the provider determines which restrictions the service has concerning the use of this information by the service. In this paper we present a metamodel for services computing that incorporates trust relationships. The metamodel provides the means for specifying services and service requests including trust requirements and constraints. The proposed metamodel is encompassed in a framework for goal-based service discovery and composition.

1. Introduction

Service-Oriented Computing (SOC) has emerged as a computing paradigm that uses the concept of service as the basic construct for distributed applications. SOC has a promising vision of a world of cooperating services where cooperation relations can be dynamically created to form applications and business processes [1]. In a simple service setting we have a service client that requests the execution of a service to a service provider. Typically, a service needs information from the service client as the input data for its execution. In case of a single request and if only one service provider is considered, it is straightforward for the service client to decide whether the required information is sensitive or not and whether it can be sent to the service. Even if the information is sensitive the service client can decide to send it to the service if a (implicit or explicit) trust relationship exists between the service client and the service provider.

However, in environments where a large number of service clients and service providers are present, it becomes difficult to manually and individually assess the sensitiveness of the information as well as control the trust relationships between service clients and service providers. In such environments, the need for a supporting service platform emerges. A service platform can support service clients in activities such as service discovery, selection, composition and invocation [5]. The support can also be extended to service providers by providing a mechanism for rapid creation, deployment and advertisement of services. The addition of semantics to service descriptions and to messages exchanged between service clients, service providers and the supporting service platform enables complex reasoning tasks [6]. Among these reasoning tasks are the interpretation of service providers’ capabilities and service clients’ requirements.

Consider initially a scenario where the interactions between service clients, service providers and the supporting platform are solely based on the syntax of the exchange messages. In this scenario, activities such as service discovery can be performed by matching the terms contained in the request of a service client with the terms in the description of a service. The terms should have an exact match. This implies that problems can arise if, for instance, the service client maps concept C to term T1 and the service provider maps this same concept C to term T2. This semantic mismatch can be solved or at least minimized by the introduction of semantic annotations. Assuming that the participants share the same conceptual model and that the terms in the exchanged messages are mapped to this model, a semantic interoperability becomes possible.

In this work, we propose to raise the abstraction level of interoperability. At the higher abstraction level, the organizational and social levels are considered and the mapping is established between a client’s goal and the service that fulfills that goal. This paper presents a framework for goal-based dynamic service discovery and composition. The framework includes a precise definition of goal, techniques for goal assessment and modeling, and the rationale for dynamic service discovery and composition. Trust aspects are included in all components of the
framework. The approach to tackle trust issues is the focus of this paper.

This document is further structured as follows. Section 2 presents our goal-based service framework. Section 3 details the proposed metamodel focusing on the trust aspects. Section 4 presents a usage scenario to illustrate the applicability of the framework, and Section 5 gives conclusions and identifies topics for future work.

2. The goal-based service framework

In recent years, goal-based analysis has been used in different areas of Computer Science to identify stakeholders’ objectives, determine requirements for software systems and guide system’s behavior. As a representation of a user’s objectives, goals are used in Service-Oriented Computing to indicate the desired outcome of a service. In the Service-Oriented Computing literature we can find initiatives for service discovery and composition based on goals such as the Web Service Modeling Ontology (WSMO) [2], GoalMorph [3] and the approach presented in [4]. These initiatives have in common the assumption that goal definitions are already available and have been previously identified and modeled. However, these initiatives do not clarify how these goals are gathered and modeled and how the goal descriptions relate to concrete services. Moreover, trust-related issues are treated ad-hoc and mainly by passing to the service client the responsibility to decide to whom information should be provided. We claim that prescriptions of how to model goals and how to define trust relationships are still missing and that they should be addressed when developing a dynamic service discovery and composition framework based on goal gathering and modeling.

Our framework to support dynamic service discovery and composition is based on goal modeling, and assumes that the involved stakeholders (client applications, service providers, supporting platform) are placed in a common and known domain. This requirement is necessary because the approach relies on the availability of domain-specific ontologies. For each domain, valid goals for the stakeholders of that domain are identified together with the tasks required for their fulfillment. Moreover, trust relationships can be defined at the modeling level allowing the supporting platform to enforce these relationships. Figure 1 depicts the main elements that comprise our framework:

- Service platform. A service platform supports the interaction between service providers and service clients. From the service provider’s perspective, the platform supports the publication of service descriptions. From the service client’s perspective, the platform provides facilities for service discovery, composition, invocation and monitoring, among others. Different platform implementations can offer different sets of facilities. In this work we assume that the platform offers at least service discovery and composition.

- Goal-based service metamodel. This metamodel defines domain-independent concepts such as service, stakeholder, organization, goal and task, and their relations. These definitions are further used and specialized in the domain and task ontologies.

- Domain ontologies. These ontologies define domain-specific concepts and the relations among these concepts. Domain ontologies should be available for the service platform as well as for the service providers and clients.

- Task ontologies. Associated with domain ontologies, task ontologies provide domain-specific definitions of the tasks valid in a domain.

For the purposes of this framework, we define goal as a description of an agent’s intended state of affairs. This definition encompasses two aspects: (i) the agent has a setting of the world that he/she wants to be reached, and (ii) the agent is committed to the fulfillment of the goal and will act accordingly to have it fulfilled. The latter aspect is also referred to as the agent’s intentionality. A task is defined here as the means to fulfill a goal, i.e., it defines a behavior that transforms the world from a setting A into a setting B. When the state of affairs derived by task T’s outcome matches the description of the state of affairs of goal G, we say that task T supports goal G.

Figure 1. Main components of the goal-based services framework.
Figure 2: Relations between goals and tasks

Figure 2 depicts the relations between goals, tasks and the related agents. In Figure 2 we derive two other relations:

1. If Agent_B executes Task_A and Task_A supports Goal_A, it implies that Goal_A is fulfilled by Agent_B.
2. If Agent_A has Goal_A and, for some reason it cannot fulfill the goal itself, a delegation is performed. This delegation implies trust relationships and commitments that are further detailed in next sections.

3. The service metamodel

We related our definition of goal and task from Section 2 with concepts of the Service-Oriented Computing realm. The notion of agent, as used in Figure 2 is refined by the notion of service client and service provider. Figure 3 depicts the relations between goals, tasks, services, service clients and service providers.

In our approach we consider a service the concrete realization of a task. In other words, a task is an abstract (in the sense that it does not have a direct implementation) definition of activities whose outcome matches the state of affairs defined by a goal. This separation between a definition of activities and the actual implementation of these activities allows the division of administrative domains of tasks and services. While services are typically defined by service providers, tasks can be defined by the service client, domain specialists or by the providers of the supporting service platform. In the scope of our work as depicted in Figure 1, we assume that task definition is mainly done by domain specialists by means of task ontologies associated with domain ontologies. In this scenario the following steps are followed:

1. Service providers define their services and make them available to the supporting service platform;
2. Domain specialists define domain and task ontologies including goals valid in a domain;
3. The supporting service platform tries to match the services with the tasks defined in the task ontologies;
4. Service clients define their goals and request the fulfillments of these goals to the supporting service platform;
5. The supporting service platform matches client-defined goals with the domain ontology goals.

After these steps, we have a relation between a client-defined goal, its counterpart domain-defined goal, a task supporting the goal and a service that performs the task. Then, a service is executed and its outcome fulfills the service client’s goal.

These steps are only possible when a trust relationship exists between service client and service providers. This relationship can be direct, i.e., a service client A trusts a service provider B; or it can be indirect (by transitivity), i.e., the service client A trusts the supporting service platform and, indirectly, trusts the service providers associated with the service platform. In these two scenarios we have binary relationships relating service clients to service providers or to the service platform. However, more complex scenarios emerge when we consider that the trust relationship has a context. Considering the real-world example where John trusts Peter to borrow his car but does not trust him his personal account’s password. In this example, the trust relationship includes a conditional element (borrowing of John’s car) that limits its existence.

In our framework, the conditional element of a trust relationship is the goal. By including a goal in the trust relationship between a service provider and a service client we define trust as a ternary relationship as depicted in Figure 4. The trust, ownership and delegation relations in our framework are based on and extend the relations presented in the Secure Tropos modeling framework [7] and are defined as follows:
Ownership. Represents the fact that the service client is the legitimate owner of the goal. The owner has full authority over the owned goal and can grant its fulfillment to another agent (the service platform or a service provider);

Trust. Represents the belief of one agent (the service client) that another agent (the service platform or a service provider) will not misuse the goal it has been granted. The former actor is called truster, the latter is called trustee and the object around which the trust is centered is called trustum;

Delegation. Represents a formal event on which an agent delegates to another agent the permission to fulfill a goal. The former agent is called delegater, the latter agent is called delegatee and the object of the delegation is called delegatum.

Regarding trust, the role of the supporting platform is to only allow the establishment of the delegation of a goal from a service client to a service provider if a trust relationship exists between them. Ultimately, the service platform intermediates the discovery and selection of services according to the client’s goals. Therefore, the delegation relationship is complete when the service platform discovers, selects and invokes a service that matches the requirements of the client’s goal.

The trust relationship can be indicated in two alternative ways: (i) the service client explicitly informs to the supporting platform the service provider that it trusts or; (ii) the service client informs to the supporting platform the conditions for trusted service providers. In the former case, the platform only selects services from providers that have trust relationships with the requesting service client. In the latter case, the service platform receives the conditions from the client and matches these conditions to the information concerning the providers. For example, a service client can define that it only trusts providers that are located in its city. In this example the service platform only selects the services from providers located in the same city of the service client.

4. Example scenario

Domotics has been considered as the domain for our example scenario. In the example, John (a service client) has a goal of getting customized comfort settings whenever he arrives at home. Figure 5 depicts the ownership relation between a service client (John) and its owned goal (GetCustomizedComfort). This relationship is automatically assessed by the supporting platform when the service client informs the platform the goal it wants to be fulfilled.

![Figure 5. Service client's goal](image)

Among the several goals defined in the domotics domain ontology, in this paper we consider one to customize the comfort settings of the house according to the user. This goal specifies which comfort settings the user wants, e.g., ambiance illumination, temperature, etc. Figure 6 shows the model defined by domain specialists in which the domain-defined goal GetCustomizedComfort is supported by the task SetLights. In this model, a task decomposition structure is also presented. The model shows that the task SetLights can be composed by the sub-tasks SetLightColor and SetLightIntensity. This
decomposition model allows the platform to search for services that can perform the activities defined in the SetLights task. If no service could be found to perform the activities of the SetLights task, the platform searches for services that can perform the activities defined in the sub-tasks SetLightColor and SetLightIntensity.

Figure 7 shows the trust relationships defined by the service client and informed to the service platform. In this example, John defines that he trusts the service providers ServProv_A and ServProv_B. Based on this information the service platform proceeds to search for services provided by the service providers trusted by John and that perform the activities defined in the tasks depicted in Figure 6.

![Figure 7. Service client's trust relationships](image)

The platform finds the services SetLightColor and SetLightIntensity that performs the activities defined in sub-tasks SetLightColor and SetLightIntensity, respectively.

5. Conclusions and future work

In this paper we present a metamodel for goal-based semantic services that enables the definition and enforcement of trust relationships. This metamodel is part of a framework that supports goal-based semantic services. Besides the metamodel, the framework is composed by domain and task ontologies and a service supporting platform.

The metamodel defines goals as the description of a service client’s intended state of affairs. The domain and task ontologies define valid goals for a specific domain and task that support the fulfillment of these goals. Finally, services are offered to the platform by service providers and are also annotated according to the domain ontologies. When the fulfillment of a goal is requested by a service client, the supporting platform matches the goal to the domain-defined goals and retrieves the tasks defined on the task ontologies that support the goal. A task is an abstract description of activities that can, as its outcome, fulfill a goal. The service is defined here as the concrete realization of activities prescribed by a service provider.

The distinction between the abstract description of activities (the task) and the concrete realization of activities (the service) has been shown useful in our framework to support dynamic service discovery.

The framework assumes the previous existence of domain and task ontologies defined by domain specialists. This assumption makes the framework suitable for environments where the domain is clear and well known.

The mechanism for trust enforcement starts with the service client defining its trusted service providers or defining criteria for potential service providers. Then the supporting service platform searches for services from trusted service providers or for service providers that comply with the informed trust criteria. Preliminary evaluation shows that the definition of trust criteria offers more flexibility. In this way the service client does not need to inform the platform every time it trusts a new service provider. The platform uses the trust criteria to automatically infer whether a service provider can be trusted by the service client or not.

The next steps of our research on this trust-enabling and enforcement framework are: (i) the definition of more complex domain and tasks ontologies; (ii) the complete implementation of the supporting platform;
(iii) the definition of evaluation criteria for the framework and; (iv) the comprehensive evaluation of the framework against the defined criteria.

Acknowledgements
The present work is partly funded by the Freeband Communication project A-Muse (http://a-muse.freeband.nl). A-Muse is sponsored by the Dutch government under contract BSIK 03025. The work on the definition of the goal concept is being conducted in cooperation with Dr. Giancarlo Guizzardi, Universidade Federal do Espirito Santo, Brazil.

6. References