GSO: Designing a Well-Founded Service Ontology to Support Dynamic Service Discovery and Composition

Luiz Olavo Bonino da Silva Santos*, Giancarlo Guizzardi†, Renata Silva Souza Guizzardi†,
Eduardo Gonçalves da Silva*, Luís Ferreira Pires* and Marten van Sinderen‡
*Software Engineering Group
University of Twente, Enschede, The Netherlands
Email: (l.o.bonino, e.m.g.silva, l.ferreirapires)@ewi.utwente.nl
†Department of Computer Science
UFES, Vitória, Brazil
Email: (gguizzardi, rguizzardi)@inf.ufes.br
‡Information Systems Group
University of Twente, Enschede, The Netherlands
Email: m.j.vansinderen@ewi.utwente.nl

***Abstract*—A pragmatic and straightforward approach to semantic service discovery is to match inputs and outputs of user requests with the input and output requirements of registered service descriptions. This approach can be extended by using pre-conditions, effects and semantic annotations (meta-data) in an attempt to increase discovery accuracy. While on one hand these additions help improve discovery accuracy, on the other hand complexity is added as service users need to add more information elements to their service requests. In this paper we present an approach that aims at facilitating the representation of service requests by service users, without loss of accuracy. We introduce a Goal-Based Service Framework (GSF) that uses the concept of goal as an abstraction to represent service requests. This paper presents the core concepts and relations of the Goal-Based Service Ontology (GSO), which is a fundamental component of the GSF, and discusses how the framework supports semantic service discovery and composition. GSO provides a set of primitives and relations between goals, tasks and services. These primitives allow a user to represent its goals, and a supporting platform to discover or compose services that fulfil them.**

***Keywords*-Service-Oriented Computing; ontology; service discovery; service composition;**

I. Introduction

Service-Oriented Computing (SOC) has been gaining momentum in recent years with an increase in industry adoption and research efforts. In industry, SOC has been seen as an approach to integrate legacy and new systems with an standardized set of protocols and interfaces in a distributed manner. Among the research efforts we can include the pursuit of supplying semantics to service descriptions, message exchanges and service requests. The addition of semantics aims at supporting semantic interoperability for heterogeneous systems. Ontologies are being used in the realm of SOC for providing this semantic richness [1], [2], [3].

Even when semantically enriched, service client's requirements are expressed in terms of inputs, outputs, preconditions and effects, also known as IOPE. End-users, i.e., human service clients, may have difficulties to express such requirements as they would have to deal with technical issues such as the request's language, and the type, format and coding of the IOPE. Additionaly, if we consider the use of services in a Pervasive Computing environment where the end-users would have to deal with a significant number of services, computing devices, sensors, interfaces and actuators, the problem of expressing service requirements and providing the service inputs increases. To tackle application scenarios where end-users are not technology literate and are immersed in an environment populated by a plethora of services, computing devices, sensors, etc., we propose the use of goals to express what the end-user wants accomplished by services. The use of goals aims at raising to a higher abstraction level the definition of service client's requirements and, therefore facilitating its use by end-users.

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 service client'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) [1], GoalMorph [4] and the approach presented by Zhang et al in [5]. However, besides not agreeing on a goal definition, these initiatives either do not clarify how goals are gathered and modeled ([4] and [5]) or mix the goal definition with the service that should fulfill it ([1]).

In this paper we present an ontology-based approach to support dynamic service discovery and composition. Our

978-1-4244-5564-5/09/$26.00 ©2009 IEEE

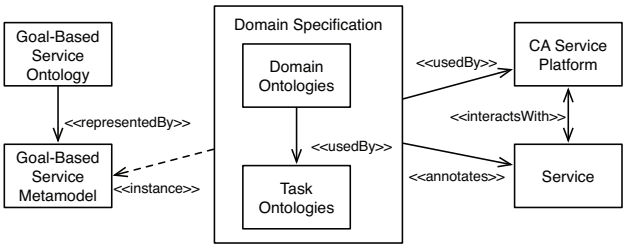

Figure 1. Main components of the Goal-Based Service Framework

Goal-Based Service Ontology (GSO) defines a language that supports domain specialists to define domain specifications. These domain specifications are composed by a set of ontologies providing a common knowledge about an specific domain and are used to semantically annotate services and the exchanged messages between service clients, service providers, context providers and the supporting platform. Therefore, this paper aims at addressing the following research questions: *(i)* how to provide a more intuitive way for (non-technical) service clients express service requests?; and *(ii)* how to provide dynamic service discovery and composition from these service requests?

This paper is further structured as follows. Section II gives an overview of the architecture of the Goal-Based Service Framework. Section III discusses the main concepts and relations of the proposed Goal-Based Service Ontology and explains the rationale behind the definition of these concepts. Section IV presents the supporting service platform and the matching and composition algorithm. Section V presents an example usage scenario of GSF in the Home Health Care domain. Section VI presents some final considerations and identifies topics for future work.

II. Goal-Based Service Framework (GSF)

In our work we consider a scenario of Pervasive Computing associated with SOC technologies and concepts. In this scenario we have human agents surrounded by and interacting with a plethora of computational devices and services. This motivates the need of a platform support to tackle with the issues of service discovery and composition in an unobtrusive way.

Our framework to support dynamic service discovery and composition is based on goal modeling and assumes that the involved stakeholders (service clients, service providers, context providers, supporting platform) share the same conceptual models, i.e., the same set of domain ontologies. This requirement is necessary because the approach relies on the availability of domain-specific ontologies. Figure 1 depicts the main elements that comprise our framework. The elements of this Goal-Based Service Framework (GSF) are described as follows:

- *Goal-Based Service Ontology (GSO)*. This ontology defines domain- independent concepts such as service, service client, service provider, goal, task and their relations, among others. This domain independency is however limited to domains and applications within the scope of the aforementioned scenario of Pervasive and Service-Oriented Computing.
- *Goal-Based Service Metamodel (GSM)*. Generated from Goal-Based Service Ontology, this metamodel represents the concepts defined in GSO and defines the language used by domain specialists to create domain specifications.
- *Domain Specification*. GSF can be used in different application domains such as Health Care, Ambient Intelligence, etc. For each of these application domains a domain specialist defines a domain specification, namely the concepts and relations relevant to the domain, goals that users can have, valid tasks in the application, etc. GSM, representing GSO concepts, provides a modeling language that enables domain specialists to define domain specifications allowing a shared knowledge about particular domains. A domain specification is composed of: *(i)* a domain ontology including domain-specific concepts, the relations among these concepts and valid goals that users of that domain can have; and *(ii)* a task ontology which uses the concepts defined in the domain ontology and provides domain-specific definitions of valid tasks and how they can be related to user's goals fulfillment.
- *Context-Aware Service platform*. The context-aware 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 mechanisms for service discovery, composition, invocation and monitoring, among others. Moreover, the context-aware components of our supporting platform provide user's contextual information that is used *(i)* to select which of the tasks that support a given goal will be used in the service discovery and composition procedures and, *(ii)* as input data for the discovered services. The context information gathering reduces the need of direct user input and, thus, reduces also the need of user's interaction supporting a more autonomic behavior of the platform.

A normal deployment of GSF consists in the GSO, GSM and the CA Service Platform. A second step is the addition of domain specifications by domain specialists. Service providers can start to semantically annotate their services and service descriptions based on the concepts present on these domain specifications. The service descriptions are added to the CA Service Platform by the service providers.

A set of domain specifications are being developed in the scope of this work for the purpose of validating the

use framework as a whole and to check the suitability of GSO and GSM to specify domains. In this paper we present excerpts of these domain specifications as usage examples of GSO.

III. Goal-Based Service Ontology

The Goal-Based Service Ontology (GSO) includes concepts and relationships that (represented by the Goal-Based Service Metamodel) allows domain specialists to define their goal-based service-oriented domain models. Clarity and an appropriate formalization of semantics are important requirements for ontologies. These requirements are especially relevant in Service-Oriented Computing (SOC) to enable complex tasks involving multiple agents. The focus of GSO is to provide ontologically sound concepts relating concepts of SOC such as Service Provider, Service Client and Service, with concepts pertinent to our goal-based approach, such as Goal and Task. Nevertheless, these concepts are not sufficient for a complete domain specification. More domain-independent concepts and relations are necessary to characterize a domain such as Description, Agent, Intention, Material Relation, among others. In order to provide these concepts and relations and at the same time supply formalization and semantic clarity we use a foundational ontology namelly Unified Foundational Ontology (UFO) [6] which is based on formal principles derived from linguistics, philosophy and mathematics. A foundational ontology is an axiomatic system of domain-independent real-world categories [6], [7]. A foundational ontology provides a conceptual modeling language to be used to express other ontologies (such as GSO). In our work GSO builds upon some of the concepts and relations of UFO and adds SOC and goal (and task) related concepts and relations.

A. Goal definition

The concept of goal has several different definitions depending on the domain the term is used, e.g., Philosophy, Sports, Economy, among others. Narrowing down to the Computer Science domain, a variety of definitions of the goal concept can also be found. In the Artificial Intelligence (AI) realm, goal is defined as a "description of a world state that is expected to be realized" [8]. Among the several definitions for goal in the agent-oriented computing community, in [9] goal is defined as a "state with highest utility and an agent must choose the course of action to reach that goal" and in [10] goal is defined as a "final state that the agent tries to achieve by moving from its initial state through a defined and finite sequence of intermediary states".

In the community of Semantic Web, WSMO [1] defines the concept of Goal as "representations of an objective for which fulfillment is sought through the execution of a Web service" and that "goals can be descriptions of Web services that would potentially satisfy the user desires". From these definitions we have that WSMO ties goal closely to Web services, i.e., a commitment is done already in the ontological level w.r.t. the specific technology to realize services. An example of this close tie between a WSMO goal and Web services is in WSMO's goal description model which includes the interface of the Web service the user would like to interact with. In our work we consider Web services as one possible technological solution for implementing services and do not limit our approach to this specific technology.

For the purposes of this framework, we adopt and extend the goal definition presented in [11] and define goal as the *propositional content of a service client's intention*. In other words, a service client (an intentional agent) has an intentional moment of the type ***Intention***. We use moment in its ontological sense of being an individual that can only exists in other individual, i.e., moments are existentially dependent of other individuals. Here, the term moment has no relation with the intuitive notion of time instant in natural language. For instance, moments such as color, intention and commitment can only exist if a bearer of those moments exists, namely, a colored thing, an intentional agent and a committed individual, respectively. Every intentional moment has a type and a propositional content. The propositional content is an abstract representation of a class of situations referred by that intention.

The other types of intentional moments are ***Belief*** and ***Desire***. Belief is defined in UFO as any knowledge an agent has about the world. Examples include my belief that the Moon orbits the Earth and, my belief that Paris is the capital of France. Desire and intention express a will of an intentional agent towards a state of affairs in reality. The difference between intentions and desires is that by intending something, an intentional agent commits at pursuing it, i.e., an intention is a desire plus an internal commitment. Therefore, a service client commits to pursue the fulfilment of his goals. This definition allows that many alternative situations can satisfy (in the logical sense) the goal. For instance, if one has a goal of having a vehicle to go to work, a situation where he has a car satisfies the goal as well as (if no other constraints are defined in the goal) a situation where he has a bicycle.

B. Tasks, goals and services

Figure 2 presents a model of types depicting the *Goal* concept of GSO and how it is related to UFO concepts (grayed boxes). In GSO we added that a Goal is *owned* by a Service Client Type. This ownership relation defines a meta-commitment making that individual instances of the Service Client Type have a goal of certain kind, i.e., let S be a service type a g a goal, we have that S owns g iff for every instace x of S there is an intention I which is an intrinsic property of x (inheres in x) and g is the propositional content of I. Therefore, goals can be used

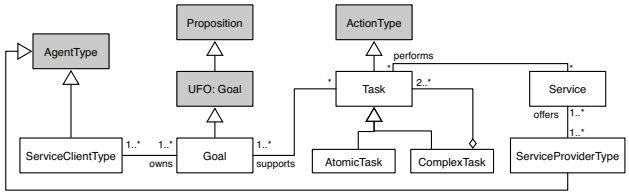

Figure 2. Goal definition

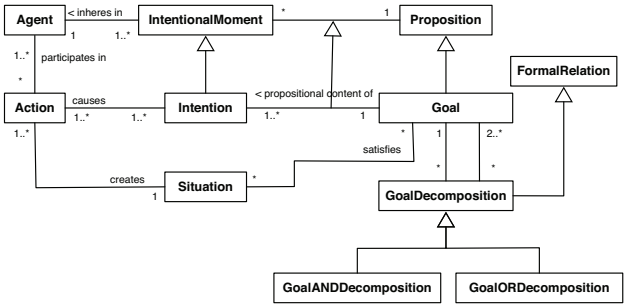

Figure 3. Goal satisfaction and composition

to characterize Service Client Types during the domain specification phase. For instance, a domain specialist can determine that in a Supply Chain domain *Customer* (a Service Client Type) is characterized as having the goals of *HavingRawMaterialWhenNeeded* and *OptimizeInventory* while *Supplier* (another Service Client Type) is characterized as having the goal *SupplyRawMaterialWhenRequested*.

Task in GSO is a specialization of the UFO concept of Action Type. An Action in UFO is an intentional event, i.e., an event performed by one or more agents in order to accomplish a goal. In figure 2, the relation *Performs* between *Service* and *Task* (again, an Action Type) represents that instances of *Task* are executed by the Service Provider when the associated service is executed, i.e., a service is executed when the task it is committed to perform is executed. Finally, the relation *Supports* between *Task* and *Goal* represents that a successful execution of that task satisfies that goal, i.e., the situation derived from the outcome of a task execution matches a situation that satisfies the goal.

As depicted in Figure 3 (a model of instances), a Goal can be structured in two different ways, namely, in a decomposition structure (GoalANDDecomposition) and in a specialization structure (GoalORDecomposition). These two structures have different implications on goal fulfillment. In the decomposition structure, the fulfillment of the high-level goal is accomplished with the fulfillment of all the sub-goals. For instance, a high-level goal GetMedicalTreatment (from the Health Care domain) is fulfilled when its sub-goals GetMedicalConsult and GetMedicinePrescription have been fulfilled. Conversely, in the specialization structure, the fulfillment of a sub-goal implies the fulfillment of the high-level goal. For instance, the same hypothetical high-level goal GetMedicalTreatment is fulfilled when either one of the sub-goals GetHomeMedicalTreatment and GetHospitalizedMedicalTreatment is fulfilled.

Figure 3 also shows the causal chain of goal satisfaction. An intention (of which a goal is its propositional content) causes an action (an instance of a Task) to be performed, i.e., since the agent is committed to the goal satisfaction, he acts accordingly to pursue its satisfaction. The action creates a situation that satisfies the goal. The use of situations to satisfy goals opens the possibility of using a Fuzzy mechanism to assess partial satisfaction (if necessary) of goals. Depending on the domain being specified using GSO, the domain specialists can define different goal satisfaction degrees.

In GSO, the ownership relation entitles the owner agent, i.e., a particular agent instantiating an specific Service Client Type, to delegate the fulfillment of the goal to another agent. Moreover, by delegating a goal to an agent, the delegatee commits to the fulfillment of that goal. Therefore, the delegation relationship implies also a commitment between the delegator and the delegatee in relation to a goal. In GSO, this delegation relationship occurs when a service client delegates the fulfillment of a goal to a service provider. In the scope of this paper we focus on the open delegation [11] of a goal. In an open goal delegation, a service client delegates the satisfaction of a goal to a service provider but does not prescribe any specific way of reaching this satisfaction. In other words, the service client only wants the goal satisfied without caring about how it is going to be satisfied. In contrast, in a close delegation the service provider should satisfy the service clients goal by means of a specific task. The relations of ownership, (close and open) delegations and satisfaction relations in GSO are also reflected in common goal-based requirements engineering languages such as i* and Tropos [12].

Although the supporting service platform (CASP) intermediates the discovery, composition and invocation of services on behalf of the service client, we consider that the goal delegation occurs between the service client and the service provider and not between the service client and CASP. Having the goal delegation established from the service client to the service provider allows CASP to determine and enforce trust relationships (when necessary) between service clients and providers, i.e., when required, the platform can only allow service discovery, composition and execution to services whose providers are trusted by the service client.

C. Service

GSF and (more specifically) CASP are targeted to support the discovery and composition of computational services. However, at the ontological level we also consider services at the social level. This separation between social and

computational services allows us to cope with situations where a computational service can be related to a social service and contribute to the fulfillment of a client goal. For instance, in a traveling scenario, John has the goal of spending his holiday in Paris. One social service that fulfills this goal is the air transportation service by means of flying John from Rome to Paris. In the case John decides to use a computational platform (such as GSF) to have its goal fulfilled, an electronic flight ticket booking service is one computational service related to the mentioned social service that can fulfill his goal.

In GSO we define service as *a temporal entity related to the commitment (a service agreement) that a Service Provider will perform a task (a type of action) on behalf of a Service Client whose outcome satisfies a Service Client's goal*. This definition of service is based on the analysis of social services presented in [13].

Our definition encompasses some of the main characteristics of service as defined in the Marketing and Economics fields, namely, intangibility (as being a temporal entity) and the inseparability of production and consumption. As opposed to a product, when a service is delivered (the equivalent to the product's production) its outcome, which may satisfy the client's goal, is immediately perceived by the service client (the consumer). In [14], the authors state that the service's value "*is always uniquely and phenomenologically determined by the beneficiary*". In our framework this statement remains valid as the service client (the beneficiary) determines the service's value by the fulfillment of his goal, i.e., the added-value of a service from the service client's perspective is perceived when the service fulfills a client's goal.

In our definition two related but distinct aspects can be considered, the service execution and the service agreement. Both have time-limited lifespan but represent different concepts. While the former represents that actual execution and consequent service provisioning, the later represents the conditions and validity for the service provisioning. For instance, when a bank's client cashes out money from an ATM, the money withdraw service execution lasts for the time that the operation to request and receive the amount of money lasts. However, the agreement for the money withdraw service is valid for as long as the client has an account in his bank. This makes explicit that a service is not only an execution of a certain task (or process) but also encompasses a set of meta-commitments, e.g., commitments to commit to execute actions of a certain type under certain conditions [15].

Tying a service with a client's goal makes explicit the added-value of the service, and the analysis of the purpose of a service and its selection; namely, a service is selected because of its role on fulfilling a client's goal. Moreover, the relation between a goal and a service supports dynamic service discovery by comparing situations that could satisfy

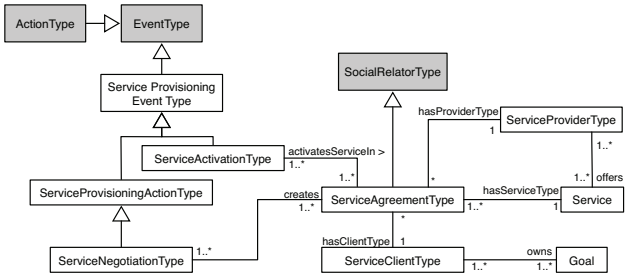

Figure 4. Service negotiation and activation

a goal with the situations generated by services' outcomes. In other words, it is possible to discover which services fulfill a goal by verifying if the situation generated by the service's outcome is equivalent to a situation that can satisfy a goal.

Figure 4 depicts the relations between Service, Service Client Type and Service Provider Type. The *Service Provision Event Type* represents types of events that can participate in service provision such as *Service Negotiation Type* and *Service Activation Type*. When a service client discovers and selects a service, a negotiation takes place to determine the conditions and constraints for the service provisioning. A successful negotiation creates a service agreement type which is a social relator binding the service client and service provider and can be potentially composed of a set of commitments and claims, e.g., the commitment of providing the service under certain conditions and for an specified cost. This social relator (the *Service Agreement Type*) can be described in a contract (not depicted in the figure) which is a normative description [11].

Figure 5 depicts the service components, namely *Input*, *Output*, *Pre-Condition* and *Effect*. In this figure we also make explicit the aforementioned distinction between a Service (that can be a social service) and a Computational Service. In GSO a service performs a *Service Task*. This Service Task is what a Service Provider commits to perform on behalf of the Service Client. In GSO we distinguish the *Service Provider* which is the agent responsible for the service from the *Service Executor* which is the agent that actually performs the *Service Task*.

In some situations, Service Executor and Service Provider can refer to the same agent individual but in other situations can be a different agent individual that has been delegated the service execution. For instance, one can hire a cleaning company to provide a cleaning service. However, this cleaning company can hire free lancers to actually clean the client's offices, i.e., the actual executor of the service is not the same (or a member of the) entity that has been hired for the service but a third-party. For simplicity, in this paper we assume that the Service Provider is the one performing the Service Task.

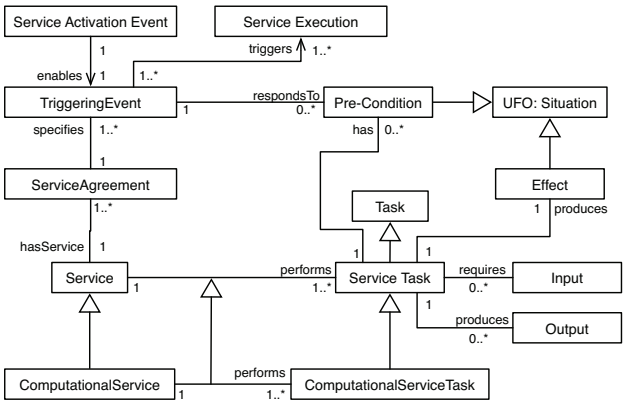

Figure 5. Service IOPE

The *Service ActivationEvent* enables the *Triggering Event* associated with the service. We defined the Service Activation to cope with situations where a service is contracted (i.e., negotiated, agreed and committed) but the actual execution of the service occurs in a different point in time. For instance, when a client opens an account in a bank and receives his account card, the service of withdrawing money in authorized ATMs is enabled but it is actually executed only when the client requests the money withdraw at the ATM.

The *Pre-Condition* represents a situation that should occur prior to the service execution. The *Triggering Event* responds to pre-conditions, i.e., when a certain situation represented by the service task's pre-condition occurs, the Triggering Event triggers the Service Execution. For instance, in a wake-up service the pre-condition is defined by the wake-up time. When the wake-up time is reached, the execution of the wake-up service is triggered. However, some services do not have pre-conditions defined, namely, can be executed regardless any situation. In these cases, the Triggering Event triggers the service execution immediately after being enabled by the Service Activation Event.

Input represents the information required by the service task for its execution. For instance, a flight booking service requires the information about the origin and destination of the flight. *Output* represents the information produced by the execution of a service task such as the reservation code or the e-ticket number for the flight booking service. While every service task execution produces an effect this is not true for inputs and outputs. Services such as TV broadcast or money transfer do not require an input or produce an output, respectively.

Figure 6 depicts how a service is described. In GSO we consider that different parts of a service have different descriptions. The *Service Profile* provides an overview of the service for advertisement purposes. It describes what the service does, its requirements and conditions. Also, service-

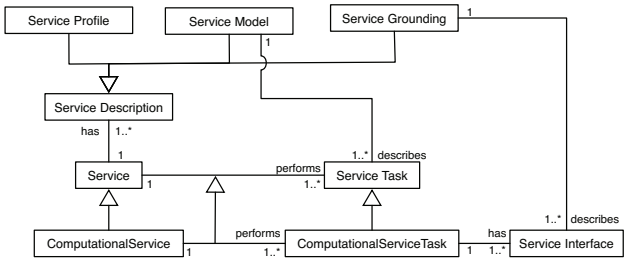

Figure 6. Service negotiation and activation

level agreement parameters can be included and used in the service negotiation. The *Service Model* describes the Service Task and provides information about the activities involved in the Service Task execution. The Service Model is used to assess the service's behavior, i.e., what set of activities the service performs. The Service Model can be used for service monitoring and orchestration (not discussed in this paper). Moreover, the Service Model can be described in different granularity levels allowing a more superficial or more in-depth view of the Service Task. The *Service Grounding* describes the *Service Interface* of the Computational Service Task. The Service Interface defines the technology-specific information necessary to invoke the service, namely, the communication protocol, parameters' types, URI, etc. GSO does not commit to any particular language to describe services, such as WSDL, OWL-S or SAWSDL. Any of these languages could be used to instantiate the concepts defined in GSO.

IV. Context-Aware Service Platform

The Context-Aware Service Platform aims at supporting non-technical service clients (end-users) in finding a suitable service, i.e., a service that fulfills requested goals. The CASP also provides support to service providers on the process of service publication. A service provider offers their services to the clients through the CASP by registering the service descriptions to the platform. The service descriptions are semantically annotated (e.g., its IOPEs, behavior description, quality properties, etc.) using the concepts defined in the domain specifications ontologies. These semantic annotations are used by the platform for the service discovery, composition and invocation.

Figure 7 depicts the main architectural components of the CASP. Domain Specialists are responsible for providing domain specifications to the platform. Domain specifications are used to provide shared semantics to the terms used amongst the stakeholders and the platform, e.g., Service Providers semantically annotate their service descriptions using the concepts defined in the domain specifications. The domain specifications are created using the modeling language defined by the Goal-Based Service Metamodel (GSM). A GSM-based editor provides to domain special-

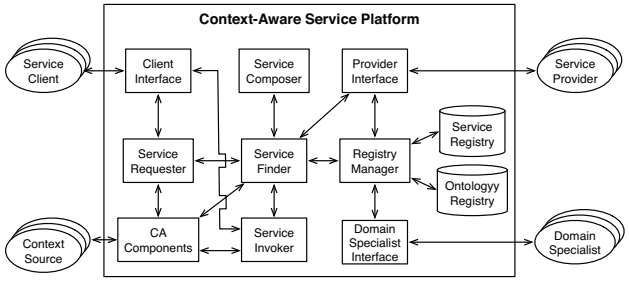

Figure 7. Context-aware service platform

ists the support to define domain specification. A Domain Specialist, through the Domain Specialist Interface, can submit or update a domain specification that is stored in the Ontology Registry.

The Provider Interface offers to Service Providers the facilities for registering and maintaining their service descriptions. The Provider Interface allows Service Providers to access the available domain specifications stored in the Ontology Registry to semantically annotate their service descriptions.

The service discovery algorithm of CASP tries to establish that a service s performs a task t that supports a goal g. Therefore, this algorithm tries to assess an indirect goal fulfillment chain (service-task-goal). However, when a service is registered to CASP, the service provider can provide to the platform a direct (service-goal) or indirect goal fulfillment information (service-task-goal). The direct goal fulfillment information indicates to the platform which goal(s) the service fulfills. The indirect goal fulfillment information indicates which tasks defined in the domain specifications the service performs. Since these tasks have already been defined as supporting goals, knowing that an specific service supports a task and that the task supports a goal, gives the platform the information that the service fulfills the goal.

When the direct goal fulfillment information is provided to the platform by the service provider, CASP tries to reconstruct the service-task-goal chain by matching the service task against the tasks that have been defined in the domain specification as supporting the goal. In other words, when the service s is indicated as fulfilling the goal g, the platform tries to match service task st (specified in the related service description) against the set of tasks that support goal g (defined in the domain specification). When there is a match, a link is established by the platform informing that the service performs the related task(s). If there is not a match, the platform automatically updates the domain specification by adding the service task of the given service as a task that supports the goal.

Regardless of having provided by the service provider the direct or indirect goal fulfillment information, for each registered or updated service the platform tries to assess which (additionally) task(s) defined in the domain specifications the service performs and, consequently, which goals it fulfills.

This automatic update associated with the support for modifications and extensions to be performed by domain specialists creates the possibility of an expressive growth of the domain specifications. For instance, it is desirable to avoid the insertion of duplicate goals, or to avoid that goals are inserted in incorrect hierarchies. To help cope with the increasing numbers and complexity of goals and tasks, techniques and approaches of the Knowledge Management (KM) field can be used. In our work we use an adapted version of KARe [16]. KARe uses taxonomies to classify documents. Each node of the hierarchy is represented by the node description (name), and also by the most relevant terms of all documents classified by the taxonomy. In our adjusted version of KARe, when the service provider inserts a new service, the platform informs the closest goals based on the service's description, i.e., the goals that a more likely to be fulfilled by this service. Then, the service provider select the goal(s) that the service is actually designed to fulfill.

The platform allows a service client to express his service request as a goal to be fulfilled. The client-defined goal is then matched against the set of goals defined in the domain specifications. The platform tries to determine the tasks in the task ontology that support the goal. Provided with this set of tasks, the platform creates an internal service request. The service request contains the properties that define the different tasks, consisting on a set of inputs, outputs, preconditions, effects, goals, and non-functional properties. Not all these properties have to be always present in a service request. In our framework the service request is further optimized, by means of the context information available regarding the service client. This optimization consists of filtering the previously created service request inputs and preconditions, considering only the inputs and preconditions that can be delivered by context sources of the service client. This allows shielding the user from directly supplying to the service, by delegating the gathering of the required information to the platform, through the available context sources. When the available context information is not enough to supply all of the required inputs, the platform requests the information to the service client.

Provided with this service request, the Finder component is invoked to discover (and possibly compose) services that match the specified service request. The first action performed by the Service Finder component is to discover services that match the service request's goals. The component queries the Registry Manager for all the services that have exact semantic match with the service request goals. Furthermore, services with goals semantically related to the service request goals, namely goals that are subsumed by the requested goals, given the domain goal ontology, are also retrieved. This allows to increase the set of retrieved and

meaningful services, i.e., increase the probability of find a service that fulfills the user request. The set of discovered services is organized in a matrix, where input/preconditions and output/effect semantic concepts define the rows and columns of the matrix. In case none of the retrieved services fully match the user service request, a further step is taken in the Service Composer component and the set of retrieve services are composed aiming at creating a composite service that fully delivers all the user services request goals. To reduce the need for service composition, once a composite service is created, Service Composer creates a description for it and stores in the Service Registry. In this way if that same service request is resubmitted Service Finder can discovers this new composite service and do not need to compose is again.

The process of service composition is performed by a graph-based algorithm for automatic service composition [17]. In the graph a node represents a service and an edge represents an output-input semantic relation. The algorithm starts by creating a graph with services that provide the service request outputs and effects. Then, in each iteration the algorithm matches the inputs of the graph's services with the outputs of the services from the set of discovered services organized in the matrix. The process continues until the service request's inputs, preconditions and goals are fulfilled by the composite service. Whenever multiple services provide a matching output for a graph service input, new graphs are created representing an alternative service composition. Input-output matching, and goal matching, are performed using the domain ontologies. This allows exact, plugin and subsume semantic matchings. Given that alternative service compositions can be created, a selection phase takes place after the composition phase. In this phase the user service request, his preferences and possibly his context are used to select the most appropriate service composition. Afterwards the selected service composition is transformed into an executable format, e.g., BPEL, and deployed in an execution engine, so it can be executed to deliver the requested service to the user.

Service clients use the Client Interface component to submit their goals to the platform. These goals can be defined either by choosing and customizing goals previously defined in the domain specifications or by creating new goals. New goals are specified in terms of situations that can satisfy the goals, i.e., the service client defines the parameters configuring a situation that fulfills his goal. For instance, in a Smart Home domain, a householder (a service client in this domain) has the goal of having ambient comfort customized to his preferences. In this example, the householder can submit his goal to the platform by specifying a situation where the lights are adjusted to his favorite color and intensity, and the temperature is set to 22 degrees Celsius whenever he enters a room of his house. Figure 8 depicts an example of the goal selection and customization GUI.

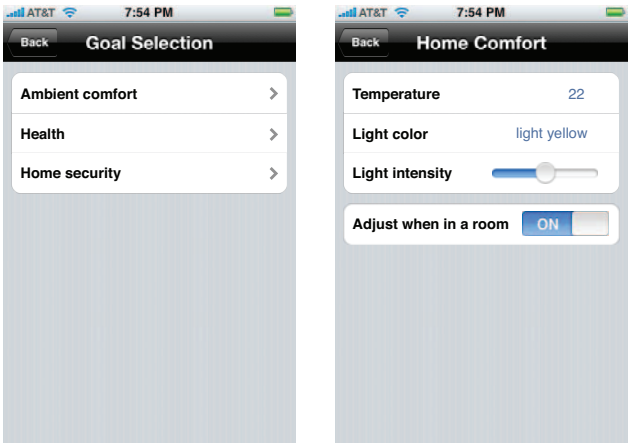

Figure 8. Goal selection and customization

V. Example Scenario

In this section we present an example scenario using GSF in the area of Home Health Care aiming at illustrating the feasibility and applicability of our approach. In this example we model the domain using GSO/GSM. The scenario is described as follows:

"*John is a remote patient that receives health treatment at home. His house is equipped with several sensors that provide contextual information about his health condition such as weight, heart beat rate, blood pressure and glucose level. Moreover, movement sensors allow the determination of the householders' location and to assess whether their are in a responsive condition or not (e.g., asleep, fainted, etc). The main goal of John is to remain healthy. The house is equipped with the Context-Aware Service Platform, the Home Health domain has been specified and this domain specification is available to the platform. Several health-related services are available to the platform.*"

Figure 9 shows a fragment of the Home Health care domain specification. In this figure a *Remote Patient* which is a type of service client owns the two goals *Have Medical Attention* and *Keep Remote Patient Healthy*. The *Have Medical Attention* goal is supported by two tasks, namely, *Calls Doctor* and *Calls Ambulance*. Here we have an example of a goal being supported by two distinct tasks. The *Keep Healthy* goal is supported by the *Monitors Health Condition* complex task. This complex task is composed by the sub-tasks *Monitors Blood Pressure*, *Monitors Weight* and *Monitors Heart Beat*.

Figure 10 shows an UML object model of the instantiation of our illustrative domain specification. In this object model, *John* becomes a Remote Patient (a type of service client) when he pursues the fulfillments of his goals through services. Since *Keep Remote Patient Healthy* is a proposition, we have that *Keep John Healthy* represents a binding between an instance of *Remote Patient* and a generic

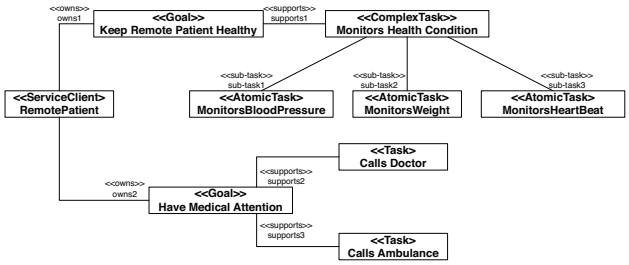

Figure 9. Domain specification fragment

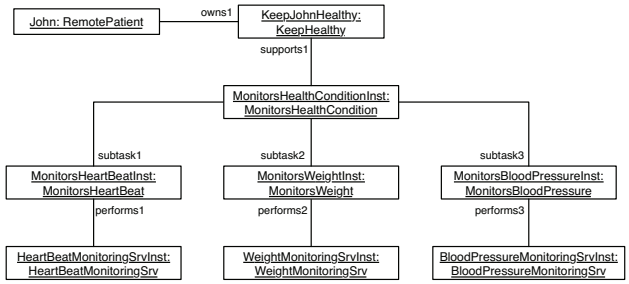

Figure 10. John's instance model

proposition. However, for the sake of simplicity, we use a uniform representation for genuine instantiation and instance binding in a generic proposition.

Having John's goal, the GSF's Context-Aware Service Platform searches for instances of tasks that support John's goal *Keep John Healthy*. The supporting platform found that the complex task instance *Monitors Health Condition Inst* and its sub–classes *Monitors Weight Inst*, *Monitors Blood Pressure Inst* and *Monitors Heart Beat Inst* support John's goal. Having found the supporting tasks, the platform proceeds to search for services performing these tasks. In Figure 10 the platform found the services *Weight Monitoring Srv*, *Blood Pressure Monitoring Srv* and *Heart Beat Monitoring Srv* that perform the tasks *Monitors Weight Inst*, *Monitors Blood Pressure Inst* and *Monitors Heart Beat Inst*, respectively.

The Context-Aware Service Platform, acting on behalf of the service client negotiates a service agreement. In this example, this agreement stipulates the frequency of the monitoring activities and the threshold for emergency warnings in the case of abnormal health indicators' values, e.g., a blood pressure measurement above 200/160 or below 90/40.

VI. Conclusions and Future Work

This paper presented the Goal-Based Service Ontology that aims at providing the means for domain specialists define domain models. GSO is part of a framework (the Goal-Based Service Framework) for goal-based dynamic service discovery and composition. This framework is primarily target on application on scenarios where the service clients are end-users without technological training and scenarios where the service clients require reduced interaction with the services. For this purpose we propose the use of goal to express the service clients' requirements and the use of context-awareness to gather information to be used as inputs for the services. In this manner the service clients have a higher level of abstraction way of expressing what they want to be accomplished by the services (by using goals) and a reduced need to interact with the services (by using context information).

Moreover, we presented and discussed the ontological foundations of the main terms defined in the framework, i.e., goal, task, service client, service provider and service platform. This ontological foundation provides a solid underlying conceptualization and supports the semantic definition of the terms used throughout our framework.

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. Examples of suitable domains for our framework are Ambient Intelligence (AmI), health care and mobile pervasive applications where users are not able to interact often with the computational devices. Additionally, the CA Service Platform has been presented together with an overview of the service discovery and composition algorithm. An example scenario has been discussed to illustrate the usage of the framework and the feasibility of the approach.

Currently, the first version of GSO/GSM has been designed together with the Home Health Care domain specification. Also, a prototype of the platform has been implemented and tested with a limited amount of services and concepts of the ontologies.

As future work we have: *(i)* definition of techniques, guidelines and tool support for client's goal specification and domain specification based on GSO; *(ii)* use of model transformation techniques for automatic transformation of goals and tasks models into service requests; *(iii)* definition of more domain specifications to assess the appropriateness of GSO in more domains; *(iv)* test the platform with more complex domains and larger amount of services; *(iv)* definition of evaluation criteria for the framework and; *(v)* comprehensive evaluation of the framework based on the defined criteria.

Acknowledgment

The present work is partly funded by the Freeband Communication project A-Muse (http://a-muse.freeband.nl), sponsored by the Dutch government under contract BSIK 03025 and by a CNPq (Brazilian National Research Council) Productivity Grant (second author).

REFERENCES

[1] J. de Bruijn, C. Bussler, J. Domingue, D. Fensel, M. Hepp, M. Kifer, B. König-Ries, J. Kopecky, R. Lara, E. Oren, A. Polleres, J. Scicluna, and M. Stollberg, "Web service modeling ontology (wsmo)," October 2006. [Online]. Available: http://www.wsmo.org/TR/d2/v1.3/20061021/

[2] D. Martin, M. Burstein, J. H. O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. (2004, November) Owl-s: Semantic markup for web services. W3C. [Online]. Available: http://www.w3.org/Submission/OWL-S/

[3] "Service-oriented architecture ontology," The Open Group, http://www.opengroup.org/projects/soa-ontology/uploads/40/16940/soa-ontology-200-draft.pdf, Draft Technical Standard 2.0, July 2008.

[4] M. Vukovic and P. Robinson, "Goalmorph: Partial goal satisfaction for flexible service composition," in *Proc. International Conference on Next Generation Web Services Practices NWeSP 2005*, 22–26 Aug. 2005, p. 6pp.

[5] K. Zhang, Q. Li, and Q. Sui, "A goal-driven approach of service composition for pervasive computing," in *Proc. 1st International Symposium on Pervasive Computing and Applications*, 3–5 Aug. 2006, pp. 593–598.

[6] G. Guizzardi, "Ontological foundations for structural conceptual models," Ph.D. dissertation, University of Twente, 2005.

[7] L. Schneider, "Designing foundational ontologies: the object-centered highlevel reference ontology ochre as a case study," in *in Conceptual Modeling – 2003, 22nd International Conference on Conceptual Modeling, I.-Y*, ser. Lecture Notes in Computer Science, I.-Y. Song, S. Liddle, T. Ling, and P. Scheuermann, Eds., vol. 2813/2003. Springer Berlin / Heidelberg, October 14 2003, pp. 91–104. [Online]. Available: http://www.springerlink.com/content/5x3x7twxgkb5q03d

[8] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2002.

[9] M. N. Moghadasi, A. T. Haghighat, and S. S. Ghidary, "Evaluating markov decision process as a model for decision making under uncertainty environment," in *Proceedings of the 2007 International Conference on Machine Learning and Cybernetics*, vol. vol. 5, 19-22 Aug 2007, pp. p.p. 2446–2450.

[10] J. S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, ser. Artificial Intelligence series. MIT Press, July 1994.

[11] G. Guizzardi, R. Falbo, and R. S. S. Guizzardi, "Grounding software domain ontologies in the unified foundational ontology (ufo): The case of the ode software process ontology," in *1th Iberoamerican Workshop on Requirements Engineering and Software Environments (IDEAS'2008)*, Recife, Brazil, 2008.

[12] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004. [Online]. Available: http://citeseer.ist.psu.edu/bresciani02tropos.html

[13] R. Ferrario and N. Guarino, "Towards an ontological foundations for services science," in *Proceedings of Future Internet Symposium 2008*, D. Fensel and P. Traverso, Eds. Springer Verlag, 2008.

[14] R. F. Lusch and S. L. Vargo, *The service-dominant logic of marketing: dialog, debate, and directions*. Armonk, N.Y.: M.E. Sharpe, 2006. [Online]. Available: http://www.loc.gov/catdir/toc/ecip0518/2005024992.html

[15] R. Silva Souza Guizzardi, "Agent-oriented constructivist knowledge management," Ph.D. dissertation, University of Twente, February 2006.

[16] R. S. S. Guizzardi, P. G. Ludermir, and D. Sona, *Agent and Web Service Technologies in Virtual Enterprises*. Idea Group Publishing, July 2007, ch. A Recommender Agent to Support Knowledge Sharing in Virtual Enterprises, pp. 115–134.

[17] E. Silva, J. Martinez Lopez, L. Ferreira Pires, and M. van Sinderen, "Defining and prototyping a life-cycle for dynamic service composition," in *Proceedings of the 2nd International Workshop on Architectures, Concepts and Technologies for Service Oriented Computing (ACT4SOC 2008)*, July 2008.