Towards a Conceptual Framework to Support Dynamic Service Provisioning for Non-Technical Service Clients

Luiz Olavo Bonino da Silva Santos, Vikram Sorathia, Luís Ferreira Pires, Marten J. van Sinderen
Centre for Telematics and Information Technology, University of Twente, The Netherlands
Email: {l.o.bonino, v.s.sorathia, l.ferreirapires, m.j.vansinderen}@ewi.utwente.nl

Abstract— **Recently, paradigms such as Service-Oriented and Pervasive Computing are being combined and applied in scenarios where users are surrounded by a plethora of computing devices and available services. Dealing with a potentially large number of devices and services can become overwhelming to users without appropriate software support. Moreover, in the case of non-technical users, an additional difficulty is to express service requests using technical concepts such as data types, XML documents, etc. In this paper we present a conceptual framework that aims at supporting the service provisioning for non-technical users, and we focus on the design and operation of the framework's software platform. The platform also makes use of computing devices that surround the users to collect contextual information that helps in the tasks of service discovery, selection and composition.**

Index Terms— **service-oriented computing, ontology, semantic web, service provisioning, goal**

I. Introduction

Service-Oriented Computing (SOC) is a paradigm for the design, use and management of distributed system applications in the form of services. The vision of SOC is that services represent distributed pieces of functionality that can be combined (composed, in SOC terms) to generate new (and more added-value) functionality [1]. In an ideal scenario based on this vision, a service client expresses requirements to a software infrastructure, and the infrastructure discovers, selects and invokes services without the need of further human interaction. Non-functional requirements such as cost, trust and privacy, amongst others, should also be stated by the service client and resolved automatically by the infrastructure.

Although completely automatic service provisioning is the ultimate goal of SOC, much work still has to be done to realize this vision. Scenarios with significant numbers of available services, service providers and service clients, may give rise to issues such as: *(i)* how to express service requests in a more intuitive way (suited for non-technical end-users); *(ii)* how to tackle semantic interoperability issues among service requests, service descriptions and the internal interpretation of terms in the service operation that use different conceptual models; and *(iii)* how to support the discovery, selection and invocation of services that fulfill the service client's goals in the least disruptive and invasive manner.

The SOC vision also overlaps with some of the characteristics of Pervasive Computing. In his seminal paper about Pervasive Computing (also known as Ubiquitous Computing), Weiser foresaw that computing, sensing and communication devices would be transparently embedded in our surrounding environment [2]. These computer-enriched environments would grant access to information and services everywhere and anytime. Readily available information can contribute to the realization of the SOC vision specially by allowing a software infrastructure to gather information related to service execution without needing direct user interaction.

In our work we are particularly interested in scenarios where non-technical users are surrounded by computer-enabled devices and sensors, and a large number of services is available. In these scenarios, additional support should be provided to the end-users to help them deal with the (possibly) overwhelming amount of decisions and interactions regarding service provisioning steps, namely, service request specification, service discovery, selection, agreement, composition and invocation.

In this paper we present a conceptual framework to support dynamic service provisioning to non-technical users. Among the components of the framework we focus on the supporting software platform that intermediates the interactions between service clients and service providers. The main benefits of our framework are to allow service clients to express their service requests using concepts closer to their natural perception and to reduce the need of direct user interactions with the services. The paper also discusses the motivation and requirements for this conceptual framework (and consequently for the software platform) and provides an example that illustrates how this framework can be used and how the software platform should operate.

This paper is further structured as follows. Section II presents motivational scenarios for the proposed conceptual framework that we have used to identify the framework's stakeholders and its requirements. Section III

This paper is based on "Service Provisioning Support for Non-Technical Service Clients," by L.O. Bonino da Silva Santos, L. Ferreira Pires, and M. J. van Sinderen, which appeared in the Proceedings of the 1st International Symposium on Middleware and Network Applications (MNA 2010), Las Vegas, USA, April 2010, part of the 7th International Conference on Information Technology : New Generations (ITNG 2010). © 2010 IEEE.

presents the stakeholders we have identified by analyzing the motivational scenarios, and Section IV presents the requirements derived from this analysis. Section V gives an overview of the conceptual framework components and, in particular, presents the architecture of the software service platform and its functional components. Section VI presents an usage scenario to illustrate the operation of the framework and, in particular, to demonstrate how the software platform supports dynamic service provisioning. Finally, Section VII gives our conclusions and identifies topics for future work.

II. Motivational Scenarios

The main objective of our framework is to support dynamic service provisioning. To identify classes of stakeholders involved in service provisioning and their interaction patterns, we have selected a set of motivational scenarios. These scenarios have been selected from the ones identified, investigated and validated in the scope of the Amigo [3], A-Muse [4] and U-Care [5] projects. In our work we have focused in the areas of Ambient Intelligence and Health Care. These areas have been chosen because they provide provide scenarios with evident usefulness and applicability to a wide range of people. Due to space constraints we selected only three scenarios that cover the issues we target in this paper. The following motivational scenarios have been used to identify service provisioning stakeholders and to initially assess requirements for dynamic service provisioning:

Motivational scenario #1: *Customized ambient comfort*

John is a professional who lives in a city house but also owns a beach house and a mountain cabin. He has ambient comfort preferences such as light intensity, light color, temperature and humidity that he expects to be automatically applied on all houses and working environments he uses. Additionally he would like that incoming messages are delivered according to his current activities. For instance, when he is in a meeting, voice messages should be transcribed and delivered in textual form on his smartphone, or when he is driving, text messages should be delivered in an audible form through his car's audio system. This use case was identified in the Amigo project [3].

Motivational scenario #2: *Emergency assistance for epilepsy*

Maria has a chronic epileptic condition. However, she wants to carry on with her life as normally as possible. As a daily routine, every morning she runs in the park near her house. Nowadays it is possible to detect an imminent epileptic seizure based on body signals. Therefore she wants to be warned if her body signals reach a critical point so she can stop running and try to put herself in a resting position. Concurrently, a relative or friend which is closer to her location should be warned of her potentially coming seizure and head on to her whereabouts. The assigned or on-duty caregiver should also receive information about her body signals, her location, which relative or friend has been warned and when and how far he/she is from Maria's location. This information is used by the caregiver to decide to send an ambulance (depending on the severity of the body signals) or to contact the warned relative or friend to provide further assistance instructions and receive extra situation assessment. This use case was identified in the A-Muse project [4].

Motivational scenario #3: *Controlled use of medicine by home-bound elderly patients*

Peter and Sofia are an elderly couple living alone at their home. Peter suffers from high blood pressure and has to take some controlled medicine. His medicine varies in frequency and schedule. Sofia has a mild diabetes that can be normally controlled via her diet and only in exceptional cases she needs to take an insulin shot. Both of them are in the initial phase of senility, presenting occasional memory lapses. Therefore, a mechanism to remind them to take their medicine is in place so they do not need to have to move to a nursing home, which could degrade their quality of life. This use case was identified in the U-Care project [5].

III. Stakeholders

Below we describe the stakeholders' roles involved in service provisioning that we have identified from the use case scenarios presented in Section II:

- *Service Client*. Responsible for requesting services and deal with possible negotiations over the service provisioning terms. For example, a frequent traveler can negotiate with an airline for discounts on a bulk purchase of tickets or a company can get a faster delivery of supplies after negotiating a transport service. With the service contract, commitments are established between the Service Client and the entity providing the service. These commitments define the Service Client's rights and obligations in the scope of the contracted service. Examples of Service Client's obligations are the payment of a specified amount for the service delivery or the availability of some information required for the service execution. Our definition of Service Client is similar to the concepts of service customer in [6], service client in [7], service consumer in [8] and service requester in [9], [10].
- *Service Beneficiary*. This role is played by the primary entity that perceives the benefits of the service delivery. In our work we distinguish a Service Client, which requests and contracts a service, from a Service Beneficiary. These roles may or may not be performed by the same entity. For example, a parent contracts the education services of a school for his child while the direct beneficiary of the service is the child. In our motivational scenario #3, although the health insurance provider have hired the home health care company, the direct beneficiaries of the service are Peter and Sofia. We categorize as performing the role of Service Beneficiary the entities benefiting from a service, which is requested by the Service

Client with the explicit intention of benefiting the Service Beneficiary.
- *Service Provider*. Responsible for the service itself, the Service Provider advertises its offered services and commits with the execution of the activities described in the service advertisement once a service is contracted by the Service Client. Our definition of Service Provider is similar to the concepts of service trustee in [6] and service provider in [8]–[10].
- *Service Executor*. The entity responsible for executing the activities related to the service. Although the individual performing the role of Service Provider is responsible (and liable) for the service w.r.t. the Service Client, the execution of the service task can be delegated to the Service Executor. The distinction between Service Executor and Service Provider allows the clear separation of responsibilities and obligations between these two entities. Our definition of Service Executor is similar to the concept of service producer presented in [6].
- *Context Provider*. Responsible for supplying mechanisms that allow the supporting platform to request and transparently gather contextual information of users. The contextual information is used by the platform to reduce the need for direct user interaction. These mechanisms include information gathered from user's personal information sources such as profiles, calendar events, appointments, travel bookings, etc., or from sensor devices such as location (from motion detectors, GPS, etc.), blood pressure, heart rate and weight, amongst others.

IV. Service provisioning requirements

We have analyzed the motivational scenarios and the stakeholders to come up with a set of functional and non-functional requirements for our proposed conceptual framework to support dynamic service provisioning. The motivational scenarios allowed us to identify the needs for dynamic service provisioning and gave us insights about how the stakeholders could interact with a supporting software platform. However, the supporting software platform should be accompanied with a set of technologies and guidelines, forming our proposed framework, as discussed in Section V. The most relevant requirements for the purpose of this paper are briefly described as follows:

1) *Domain independence*. The motivational scenarios presented in this paper relate to two different domains, namely, ambient intelligence and home health care. Scenarios related to other domains are also being considered in our work. Therefore, the supporting service platform should be able to operate in different domains while keeping the same functional properties and benefits for its users.
2) *Reduced user interaction*. The framework should reduce the need of direct user interaction. Since we are considering Pervasive Computing and Services environments, constant requests for user interaction when devices and services need some information would lead to undesirable disruptions of the users' routine. The supporting platform should make use of context-aware mechanisms to gather the necessary information aiming at reducing user interaction.
3) *Abstract service request*. By targeting our framework's support on non-technical end-users, we impose a restriction on how the users request service provisioning. We claim that non-technical end-users would have difficulties specifying service requests using current computer-based service technologies such as WSDL [11] and WSMO/WSMX [12], [13]. These difficulties relate to mandatory use of computer-related technicalities such as data types, XML formatting, URLs, URIs, ports, among others, to specify a service request and interact with the discovered services [14]. Therefore, to be able to appropriately support non-technical end-users, the framework should provide an intuitive way of requesting services.
4) *Intelligence and interoperability*. To allow reasoning and reduce issues related to semantic interoperability, the interactions between the supporting service platform and its users should be semantically-enriched. Furthermore, the internal operation of the platform is expected to benefit from the provided semantics. For instance, when searching for a service using a set of parameters, the platform can find candidate services whose parameters are not exact matches but are close enough, by applying subsumption [15].
5) *User support*. The platform should support all its users with interfaces, APIs and tooling according to each user's objectives. The service client should be supported according to its technical expertise and based on the domain's needs. For instance, a service client in the home health care domain such as Peter and Sofia from use-case scenario 3, could interact with the supporting platform through their TV set, facilitating the visualization of the interface items. In contrast, supposing Maria (from use-case scenario 2) is technologically savvy, she interacts with the supporting service platform through a web interface on her computer as well as through her smartphone.
   Service Providers require tools to support them in tasks such as service description publishing and maintenance (insertion, update and deletion) and semantic annotation of the service descriptions. Moreover, the supporting platform can provide feedback on service usage, and information about services that have been requested by users but have not been offered by any Service Provider.
   Context Providers require tooling support to manage the registration of their provided contextual information as well as to provide semantic annotations for the provided information.
6) *Modularity*. The platform should be designed to

be modular in order to allow the substitution of particular elements according to evolution of the requirements, changes in the available technologies and specific characteristics of the applications domains. For example, the change of a Service Client's interfaces mentioned in requirement 5 can possibly be achieved by a modular design.

V. Framework Design

Our solution for service provisioning support has been defined in the scope of a conceptual framework. Our framework consists of a set of technologies and techniques to address the problem of dynamic service provisioning. To define the framework architecture and its components we have analyzed the service provisioning support requirements discussed in Section IV. We adopted a bottom-up approach, i.e., we started with the service provisioning software platform that intermediates the interaction between service provisioning stakeholders, and then we moved upwards on the conceptual layers towards the enabling technologies and techniques necessary to support the operation of the software platform, which should comply with the discussed requirements.

Analyzing our requirements, in particular requirement 4, we have identified a new stakeholder, namely the Domain Specialist. The Domain Specialist is responsible for providing knowledge about a given domain, which allows the software platform to take into account the specific characteristics of the domain. In our framework, this domain knowledge is provided to the supporting service platform using domain ontologies. Therefore, the Domain Specialist is responsible for defining domain ontologies and submitting them to the platform. Additionally, the supporting service platform should provide facilities to help Domain Specialists define and manage their domain ontologies.

Figure 1 depicts our Goal-Based Service Framework (GSF) [16] for dynamic service provisioning. The main elements of the GSF are the following:

- *Goal-based Service Ontology (GSO)*. This foundational ontology 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 ontologies. GSO extends the Unified Foundational Ontology (UFO) [17] by adding concepts related to SOC and by relating the concepts of goal, task and service.
- *Domain modeling language*. The Goal-Based Service Metamodel represents the concepts defined in the GSO and defines the language (the Goal-Based Domain Specification Language) used by domain specialists to create domain ontologies.
- *Domain ontologies*. Defined using the Goal-Based Domain Specification Language (GDSL), domain ontologies provide shared knowledge about particular domains. The domain ontologies define domain-specific concepts, the relations among these concepts, the valid goals for that domain, valid tasks in that domain and how they relate to the domain goals, etc.
- *Context-Aware Service platform (CASP)*. This platform supports interactions 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, amongst others.

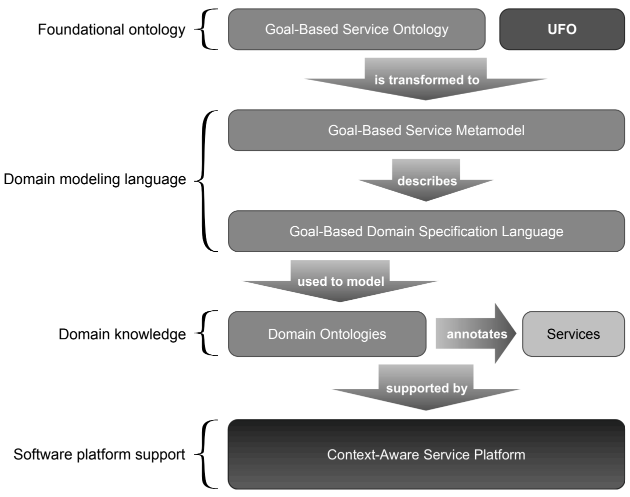

Foundational ontology
Goal-Based Service Ontology
UFO
is transformed to
Domain modeling language
Goal-Based Service Metamodel
describes
Goal-Based Domain Specification Language
used to model
Domain knowledge
Domain Ontologies
annotates
Services
supported by
Software platform support
Context-Aware Service Platform

Figure 1. Framework for service provisioning

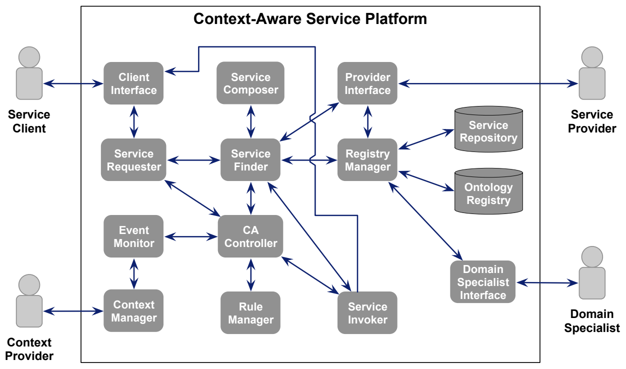
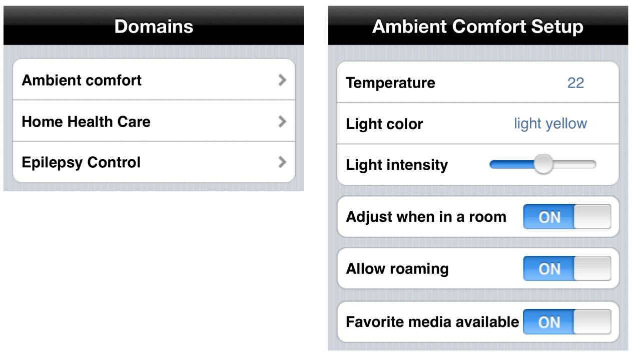


Figure 2. Architectural design of CASP

Figure 2 depicts the architectural design of the CASP. We have separated the platform's components in three main areas and we describe these areas in the sequel.

A. Stakeholders' Interface Components

Figure 2 shows that the CASP supports the interactions with its stakeholders by providing a set of interface components, namely, Client Interface (for Service Clients), Provider Interface (for Service Providers), Domain Specialist Interface (for Domain Specialists) and Context Manager (for Context Providers). These interface components provide APIs that allow GUI applications to interact with the platform. Stakeholders interact with the platform by using either the GUI applications or directly through the APIs.



Domains

- Ambient comfort
- Home Health Care
- Epilepsy Control

Ambient Comfort Setup

Temperature	22
Light color	light yellow
Light intensity	
Adjust when in a room	ON
Allow roaming	ON
Favorite media available	ON

Figure 3. Service Client's GUI application screenshots

The Provider Interface component's API offers methods to retrieve available domain ontologies (to be used to semantically annotate the service descriptions), manage the service descriptions' registration (add, update and delete) and to manage the registration of the Service Provider to the platform. The Service Provider's registration is used to control the ownership of the published services and to give feedback to the providers when a client requests a service that cannot be found in the service registry. In this way the Service Providers can add new services for which clients have expressed an interest.

The Domain Specialist Interface component's API offers methods to manage the registration of Domain Specialists and of domain ontologies (add, update and delete). The CASP includes a GDSL editor to help Domain Specialists define domain ontologies.

The Context Manager component's API offers methods to manage the registration of the Context Providers, to manage the registration of the contextual information they provide and to retrieve available domain ontologies, which are used to semantically annotate the contextual information descriptions.

The Client Interface component's API offers methods that allow Service Clients to submit their service requests, to receive the results of the service execution and to enter information required by the services but that could not be gathered by the platform as contextual information. In the GSF we use the concept of Goal as an abstraction to represent the client's service request. Goal is the propositional content of a service client's intention, i.e., a service client not only wishes something to be accomplished but is committed to its fulfillment [18]. In our approach, we represent a goal by a description of a particular state of affairs that satisfies the goal. For instance, the goal of having your house's ambient comfort set can be satisfied by specifying the room's temperature and lighting settings. Figure 3 depicts our prototype Service Client's GUI application for the Ambient Comfort domain.

Figure 2 also shows that the CASP has two main repositories, namely the Service Registry for storing service descriptions and the Ontology Repository for storing domain ontologies. These two repositories are accessible through the Registry Manager, which provides methods to other components for retrieving, updating and deleting service descriptions and ontologies, while abstracting from the technical details of the repositories' implementations. In our prototype the domain ontologies are specified in OWL [19] and the service descriptions in SAWSDL [20]. The choice for these languages is justified by the availability of tools [21]. In our work we used Fusion Semantic Repository [22] as service registry, WSMO Editor [23] for semantic annotation of service descriptions by service providers and Sesame 2 [24] as ontology repository.

B. Service Provisioning Components

After receiving the Service Client's goal, represented as the state-of-affairs that satisfies the goal, the Client Interface component forwards it to the Service Requester component. The Service Requester is responsible for generating a service request using the CASP's internal format, which contains not only the goal to be fulfilled but also the provided inputs and pre-conditions, and expected effects and post-conditions. We assume that a service fulfills a service client's goal if the state-of-affairs resulting from the outcome of its execution (i.e., its effects) matches the state-of-affairs defined by the service client that represents the goal.

To generate the service request the Service Requester component queries the Context-Aware Controller for client's available contextual information that could be used as inputs for services. For instance, a service that provides current temperature could make use of the client's current location (one of the client's contextual information) as an input parameter. After being generated, the service request is submitted to the Service Finder, which proceeds to discover the candidate service(s). In case no single service fully complies with the service request, a composition is requested to the Service Composer component. To compose the services, the Service Composer uses information present in the domain ontology regarding the processes acceptable in that domain that fulfill the user's goals. The process information can be structured in a hierarchy of processes/sub-processes, giving the Service Composer a template for service composition. For instance, if no service could be discovered to fully book a trip in a travel domain, the Service Composer gather the process information from the domain ontology defining that the book a trip process is composed of "book a flight", "book a hotel" and "book a car" sub-processes. Therefore, the Service Composer can continue to search for services that provide the functionality of these subprocesses.

Once the service's pre-conditions have been met, the service invocation is performed by the Service Invoker. Before invoking the service, the Service Invoker checks whether the contextual information is still valid for the pre-selected services. For instance, a client could have requested services providing information about nearby restaurants before meal times and the platform preselected a service giving this information for the region of the client's residence. However, in the meantime the client may have traveled to another city not covered by the original service. The Service Invoker uses contextual information to detect this situation change and requests a new service discovery to the Service Finder. After service invocation, the Service Invoker submits the service's outputs to the Client Interface to properly inform the service client. The Service Finder, Service Composer and Service Invoker components are extensions of the DynamiCOS semantic service discovery and composition platform discussed in [25].

C. *Context-Aware Components*

The context-aware components are responsible for gathering contextual information and providing it to the other platform components. The platform uses contextual information to *(i)* increase the accuracy and suitability of the selected services, and *(ii)* provide input information for services. The Context-Aware Controller receives a list of requested pieces of information (e.g., John's location or the living room's temperature). The contextual information can be requested for single use, or can be subscribed to if the platform needs continuous updates of that information.

When the request for contextual information is received, the Context-Aware Controller forwards it to the Event Monitor component, which queries the Context Manager for the availability of this information. Context Providers register their contextual information through the Context Manager. This registration contains details such as the provided information and update frequency.

When a contextual information is needed for unique and immediate usage, the Event Monitor queries the Context Manager which, in case the information is available, returns the requested contextual information back to the information's requester. However, some contextual information may be needed under certain conditions, for a certain amount of times, or in some point in the future. In this case, the context-aware components adopt a subscription approach. When subscribed contextual information is requested, the Context-Aware Controller generates an Event-Control-Action (ECA) rule containing the requested information, and when and how frequent this information is required. This rule is managed by the Rule Manager component. The context-aware components that have been built in our prototype are extensions of the Context Management Service and Awareness and Notification Service discussed in [26].

VI. Dynamic service provisioning example

We motivate and illustrate the applicability of our approach with a usage scenario in the health care domain. A health care service transaction may span multiple service providers. A service client may select and access the service of any of the available providers. As depicted in Figure 4, once a health condition is reached, the patient (service client) can decide to consult a General Practitioner (service provider). There can be many GPs available and the patient can select a practitioner by evaluating criteria like geographical proximity, availability of earliest appointment or reputation. During the consultation, the GP may request additional tests. From the pool of medical laboratories, the patient again needs to select a suitable provider based on various criteria. The tests' results are evaluated by the GP, who prescribes the treatment and possibly some medication drugs. Once again, the patient needs to choose a provider from a pool of pharmacists. Additionally, as the patient is receiving the health service(s), some or all the financial aspects are covered by her insurance company, which she has selected much earlier from an insurance service providers pool.

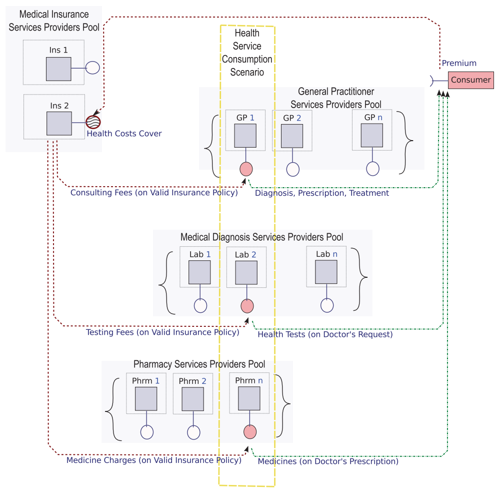

Medical Insurance Services Providers Pool
Ins 1
Ins 2
Health Costs Cover
Health Service Consumption Scenario
Premium
Consumer
General Practitioner Services Providers Pool
GP 1
GP 2
GP n
Consulting Fees (on Valid Insurance Policy)
Diagnosis, Prescription, Treatment
Medical Diagnosis Services Providers Pool
Lab 1
Lab 2
Lab n
Testing Fees (on Valid Insurance Policy)
Health Tests (on Doctor's Request)
Pharmacy Services Providers Pool
Phrm 1
Phrm 2
Phrm n
Medicine Charges (on Valid Insurance Policy)
Medicines (on Doctor's Prescription)

Figure 4. Service provider selection in a typical health care service consumption scenario

Hence, in this typical health care scenario, the actual service provision may include several providers. Each provider can be selected independently, but the services they provide may fulfill specific preconditions. For instance, laboratories typically require a test request from a GP or Medical Expert stating the test specifications. Here, the prescription can be seen as the outcome of a service transaction carried out by the GP or Medical Expert. Similarly, pharmacies often require a drug prescription before they deliver the requested drug.

A. *Domain modeling*

A health care domain ontology is modeled by specializing or instantiating concepts of the GSO. Figure 5 shows an excerpt of the GSO that gives the relation between services and tasks. In GSO, services (commit to) perform tasks. A *Service Task Type* is a sub-category of *Action Type*, which can be instantiated by an *Action* (an individual) creating a *Situation*. This *Situation* (state-of-affairs) is the outcome of a service execution and can be related to what is commonly referred to in SOC as service effect. The service task also requires some inputs and creates outputs. In our use case scenario, medical consultation, medicine delivery and medical test performance are modeled as *Social Services*. The medicine delivery social service requires a medical prescription as input type and its execution creates a situation where the service

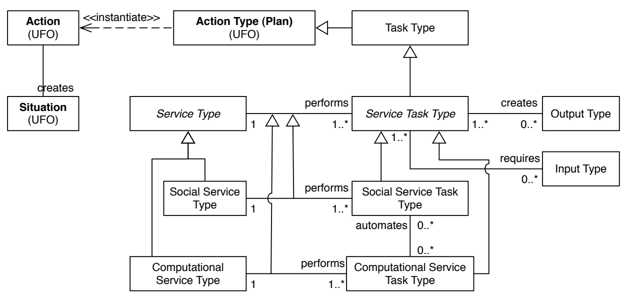


Figure 5. Services and tasks

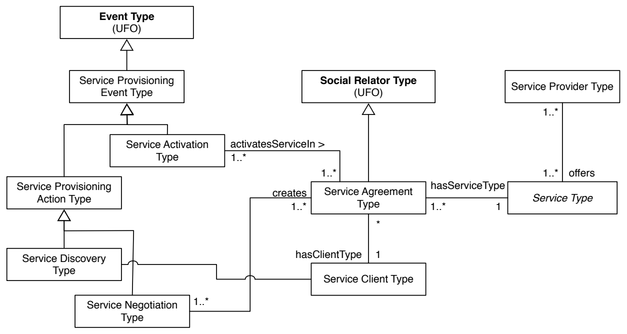


Figure 6. Service provisioning events

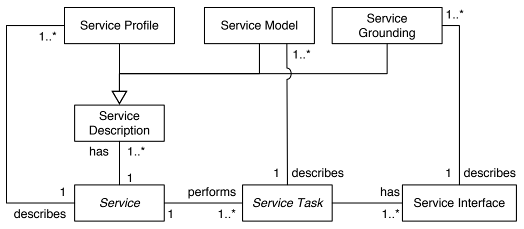


Figure 7. Service description

client has access to the prescribed medicine. Similarly, the medical test performance social service requires the input of a medical request, creates some information as test's result, and its execution creates a situation in which the doctor has access to the results.

Figure 6 shows another excerpt of the GSO that gives the relations between *Service Type*, *Service Client Type* and *Service Provider Type*. Services types can be specified in a given domain by associating them with service provider types that can offer certain kinds of services, and service client types that can request these kinds of services. Service clients and service providers are related by the agreements concerning service provisioning. This relation can be established whenever a service client finds a service whose related task creates a situation that satisfies the service client's goal. The *Service Provision Event Type* represents types of events that can contribute to or enable service provision, such as *Service Negotiation Type*, *Service Discovery Type* and *Service Activation Type*. Due to space limitations, not all events are depicted in Figure 6. 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*. This service agreement is a social relator that binds the service client and service provider. A *Service Agreement Type* can be composed of a set of commitments and claims, e.g., the commitment of providing the service under certain conditions and for a specified cost. This social relator (the *Service Agreement Type*) can be described in a contract (omitted in Figure 6), which is a normative description [27].

In our use case scenario, a patient is a *Service Client Type*, while health insurance companies, pharmacies, general practitioners, medical experts and laboratories are *Service Provider Types*. When closing the deal for a health insurance, the agreements between the health insurance company and the patient are realized in a health insurance contract, which is a *Service Agreement Type* in our health care domain.

Figure 7 depicts the service description concepts. 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 (in a human readable form), its requirements and conditions. Service-level agreement parameters can also 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., the set of activities performed by the service. The Service Model can be used for service monitoring and orchestration (outside the scope of this paper). Moreover, the Service Model can be described at different granularity levels allowing a more superficial or more in-depth view of the *Service Task*. The *Service Grounding* describes the *Service Interface*. The *Service Interface* defines information necessary to invoke the service, i.e., to trigger service execution. In the case of *Computational Services*, the *Service Interface* defines the technology-specific information necessary to invoke the service, namely, the communication protocol, parameters' types and URI. GSO does not commit to any particular language to describe services, such as WSDL, OWL-S or SAWSDL, so that any of these languages could be used to instantiate the concepts defined in GSO.

The functional and non-functional parameters of services can vary depending on the domain. Hence, domain specialists can extend basic service concepts defined in GSO to define domain-specific vocabularies for describing services in their respective domains.

Figure 8 depicts the *Goal* concept of GSO and how it is related to tasks and ultimately to services. In GSO, a Goal is owned by a Service Client Type. This ownership relation defines a meta-commitment making that the individual instances of the Service Client Type have a goal of certain kind. Formally, let S be a service client type and g a goal, we have that S owns g iff for every instance x of S there is an intention I that is an intrinsic property of x (inheres in x) and g is the propositional content of I.

A *Task* in GSO is a specialization of the UFO concept

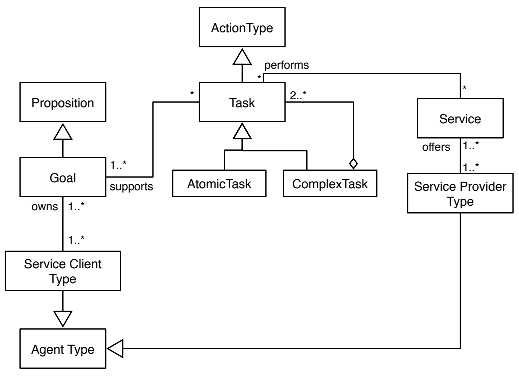


Figure 8. Goal, task and service

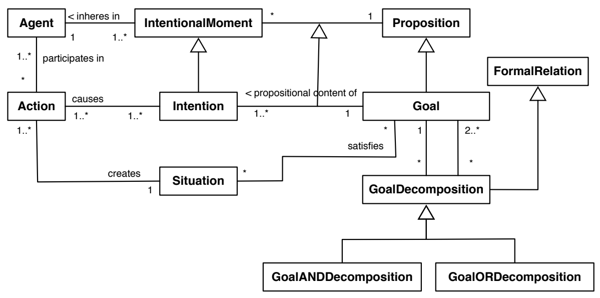


Figure 9. Goal satisfaction and composition

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 8, the relation *performs* between *Service* and *Task* represents that instances of *Task* are executed when the associated service is invoked. Finally, the relation *supports* between task and goal represents that a successful execution of that task satisfies that goal.

Figure 9 shows that 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 for 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 is fulfilled when its sub-goals GetMedicalConsult and GetMedicinePrescription have been fulfilled. Conversely, in the specialization structure, the fulfillment of one sub-goal implies the fulfillment of the high-level goal. Figure 9 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 intentional 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 logic mechanism to assess partial satisfaction (if necessary) of goals. Depending on the domain being specified using GSO, the domain specialists can define different degrees of goal satisfaction.

In our scenarios, we model that a Service Client has a Goal. However, this service client cannot satisfy the goal by his own and delegates the satisfaction for a third party, namely, a Service Provider by means of a Service. The service provider, bound by the agreements established in the scope of the contracted service performs the actions associated with the service, which creates a situation fulfilling the service client's goal.

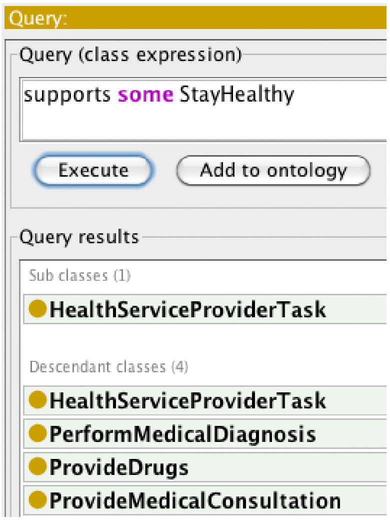
B. Domain registry

Domain ontologies defined by the domain specialists are used as reference ontologies by the Context-Aware Service Platform (CASP). The health domain ontology and other domain ontologies defined in a similar manner are stored by the CASP in an ontology repository so that they can be referred to at runtime for semantic service provisioning. These ontologies are used to annotate service descriptions and to support service requests. In our prototype, we have used Sesame 2 [24], which is an open source framework for storing, inferencing and querying ontologies. The Sesame Server allows one to build a repository that can be accessed through the web-based OpenRDF Workbench or programmatically by using query languages such as SeRQL and SPARQL through the Storage And Inference Layer (SAIL) API. Therefore, the CASP's Ontology Registry has been implemented using the Sesame 2 and the CASP Registry Manager (Figure 2) accesses the ontology registry using the SAIL API.

C. Semantic service selection

Initially, the service client defines to the CASP that his/her main goal in the health care domain is to *stay healthy*. Assuming that the initial state of the service client is that s/he is healthy, the CASP can be configured to collect user's contextual information to identify any situation that may trigger the need for a health service. Remote patient monitoring technology can be used as information source to collect user's health-related information to determine if the health service client is no longer healthy and, therefore, needs medical attention. Whenever the contextual information suggests the need for medical attention, the CASP triggers a query to identify all tasks that can be carried out to satisfy the given goal. All tasks that fully or partially satisfy the *StayHealthy* goal can be determined by the query depicted in Figure 10.

This query is performed on the health domain ontology and returns the service tasks *PerformMedicalDiagnosis*, *ProvideDrugs* and *ProvideMedicalConsultation*, which partially fulfill the *stayHealthy* goal. From the list of identified tasks, another semantic query can determine the tasks that can be performed first. The ordering is given by the task decomposition primitives of GSO, allowing the definition of structures of tasks and sub-tasks. The CASP determines that *ProvideMedicalConsultation* is the first task to be considered and, for performing this task, appropriate service provider types should be identified. A

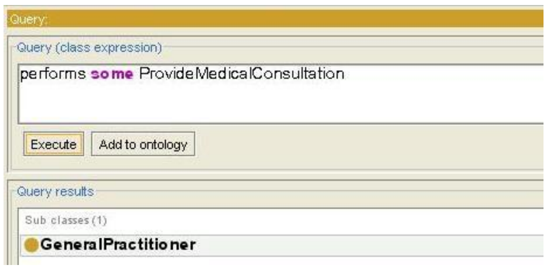


supports some StayHealthy

Sub classes (1)
- HealthServiceProviderTask

Descendant classes (4)
- HealthServiceProviderTask
- PerformMedicalDiagnosis
- ProvideDrugs
- ProvideMedicalConsultation

Figure 10. Semantic query to determine service tasks for the given goal



performs some ProvideMedicalConsultation

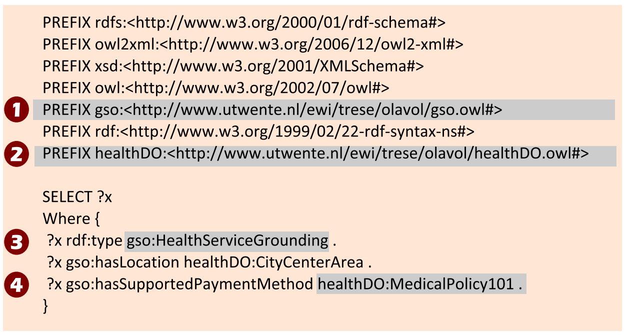
Sub classes (1)
- GeneralPractitioner

Figure 11. Semantic query to determine Service Provider Type for the given Service Task

semantic query can be performed to determine the service provider type *HealthServiceProvider* that can perform *ProvideMedicalConsultation*, as depicted in Figure 11.

There can be multiple instances of service providers offering the same service. The CASP can use the user context information to further refine the selection criteria. The service grounding of the *ProvideMedicalConsultation* service requires input parameters such as *Location* and *SupportedPaymentMethod*, which can be employed as selection criteria. Figure 12 shows a SPARQL query that can be employed to identify the most suitable service provider for this service from the available pool. The query indicates that both the GSO (marked as 1 in Figure 12) and the Domain ontology (marked as 2 in Figure 12) are used. As individual domain ontologies are derived from the GSO's concepts, query of the various service parameters defined in the GSO is allowed. The Service Grounding (marked as 3) defines the domain-independent property *hasSupportedPaymentMethod*, which can be filtered with the domain-specific parameter *MedicalPolicy101* (marked as 4) enabling consistent search strategies that are applicable in various service consumption scenarios.

The successful execution of this query indicates the selection of the first service provider in a health care consumption scenario depicted in Figure 4. Depending upon the outcome of *ProvideMedicalConsultation* task, the CASP can determine the next service task to be



```
  PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
  PREFIX owl2xml:<http://www.w3.org/2006/12/owl2-xml#>
  PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
  PREFIX owl:<http://www.w3.org/2002/07/owl#>
1 PREFIX gso:<http://www.utwente.nl/ewi/trese/olavol/gso.owl#>
  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX healthDO:<http://www.utwente.nl/ewi/trese/olavol/healthDO.owl#>

  SELECT ?x
  Where {
3  ?x rdf:type gso:HealthServiceGrounding .
   ?x gso:hasLocation healthDO:CityCenterArea .
4  ?x gso:hasSupportedPaymentMethod healthDO:MedicalPolicy101 .
  }
```

Figure 12. Semantic query for Service Provider instance selection

executed to fulfill the given goal. The same process can continue until the goal is fulfilled.

VII. Conclusions and Future Work

In this paper we presented our framework for support of dynamic service provisioning. This framework has been designed to target non-technical service client. We claim that these clients have difficulties requesting and interacting with services that use the current technologies related to Service-Oriented Computing (SOC). The SOC-related technologies demand proper knowledge of XML-based document formats, communication protocols, service choreography and orchestration, amongst others. To cope with this issue, our framework uses the concept of goal as an abstraction to express service requirements. This brings the concepts related to service request closer to the user's conceptualization, instead of forcing users to use technical concepts as in traditional approaches.

The concept of goal and its relation to the concepts of task and service are defined in a foundational ontology that is used in our framework as the basis for a domain specification language. This language is used by domain specialists to define domain ontologies, which formalize the knowledge of a given domain and allow the supporting service platform to take into account the specific characteristics of the corresponding domain.

To design of our framework we started by identifying the stakeholders involved in service provisioning using scenarios in the areas of health care and ambient intelligence. After analyzing these scenarios and the identified stakeholders, we came up with a set of functional and non-functional requirements for dynamic service provisioning that our framework should comply with. Then, the architecture design of the framework and the supporting service platform have been presented and discussed.

To demonstrate the applicability of our framework we have provided an use case scenario in the health care domain. Based on this scenario we discussed the modeling of this domain and how the service platform operates to support dynamic service provisioning.

The main benefits of the service platform are the support to service provisioning for non-technical users and the use of contextual information. The use of contextual information is used to both increase the accuracy and suitability of the selected services, and reduce the need of direct user interaction by gathering information that is used as input to services.

In the current stage, the service provisioning and the context-aware components of the CASP have been implemented. The service provisioning components used in our prototype are based on the DynamiCOS framework [25] and the context-aware components are based on the Amigo's ANS [26]. The integration of these two sets of components is underway together with the deployment of the Service Registry, Ontology Repository, Registry Management and Stakeholders' Interface components. Moreover, more complex domains are being modeled using the Goal-Based Domain Specification Language to validate its appropriateness to specify domain ontologies.

REFERENCES

[1] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing research roadmap," European Union Information Society Technologies (IST), Directorate D, Tech. Rep., 2006. [Online]. Available: http://infolab.uvt.nl/pub/papazogloump-2006-96.pdf

[2] M. Weiser, "The computer for the 21st century," *SIGMOBILE Mobile Computing and Communications Review*, vol. 3, no. 3, pp. 3–11, 1999.

[3] "Amigo: Ambient intelligence for the networked home environment," http://www.hitech-projects.com/euprojects/amigo/.

[4] "A-muse: Architectural modeling for service enabling in freeband," http://a-muse.freeband.nl/.

[5] "U-care: User-tailored homecare services platform," http://ucare.ewi.utwente.nl/.

[6] 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.

[7] M. P. Papazoglou and D. Georgakopoulus, "Service-oriented computing," *Communications of ACM*, vol. 46, no. 10, pp. 25–28, October 2003.

[8] K. Laskey, J. A. Estefan, F. G. McCabe, and D. Thornton, "Reference architecture foundation for service oriented architecture version 1.0," Oasis, Committee Draft 02, October 2009. [Online]. Available: http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf

[9] D. Booth, H. Haas, F. G. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard, "Web services architecture," http://www.w3.org/TR/ws-arch/, February 2004.

[10] S. Burbeck, "The tao of e-business services: The evolution of web applications into service-oriented components with web services," Online document, IBM Software Group, October 2000.

[11] "Web services description language (wsdl) version 2.0 part 0: Primer." [Online]. Available: http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626/

[12] 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/

[13] T. Haselwanter, P. Kotinurmi, M. Moran, T. Vitvar, and M. Zaremba, "Wsmx: A semantic service oriented middleware for b2b integration," in *Proceedings of the 4th International Conference on Service Oriented Computing*, A. Dan and W. Lamersdorf, Eds., vol. 4294. Chicago, USA: Springer-Verlag, December 2006, pp. 477–483.

[14] C. Rolland, R. S. Kaabi, and N. Kraïem, "On isoa: Intentional services oriented architecture," in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)*, ser. Lecture Notes in Computer Science, J. Krogstie, A. L. Opdahl, and G. Sindre, Eds., vol. 4495. Springer Verlag, 2007, pp. 158–172.

[15] D. L. McGuinness and A. Borgida, "Explaining subsumption in description logics," in *IJCAI (1)*, 1995, pp. 816–821.

[16] L. O. Bonino da Silva Santos, E. Gonçalves da Silva, L. Ferreira Pires, and M. van Sinderen, "Towards a goal-based service framework for dynamic service discovery and composition," in *Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations*, 2009.

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

[18] L. O. Bonino da Silva Santos, G. Guizzardi, R. Silva Souza Guizzardi, E. Gonçalves da Silva, L. Ferreira Pires, and M. J. van Sinderen, "Gso: Designing a well-founded service ontology to support dynamic service discovery and composition," in *2nd International Workshop on Dynamic and Declarative Business Process (DDBP 2009)*, September 2009.

[19] "Owl 2 web ontology language primer," October 2009. [Online]. Available: http://www.w3.org/TR/2009/REC-owl2-primer-20091027/

[20] J. Farrell and H. Lausen, "Semantic annotations for wsdl and xml schema," August 2007. [Online]. Available: http://www.w3.org/2002/ws/sawsdl/

[21] J. Cardoso, "The semantic web vision: Where are we?" *IEEE Intelligent Systems*, vol. 22, pp. 84–88, 2007.

[22] D. Kourtesis and I. Paraskakis, "Web service discovery in the FUSION semantic registry," in *BIS 2008 - 11th International Conference on Business Information*, ser. Lecture Notes in Business Information Processing, W. Abramowicz and D. Fensel, Eds., vol. 7. Springer, 2008, pp. 285–296.

[23] M. Dimitrov, A. Simov, V. Momtchev, and M. Konstantinov, "WSMO Studio — a semantic web services modelling environment for WSMO," in *ESWC '07: Proceedings of the 4th European conference on The Semantic Web*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 749–758.

[24] J. Broekstra, A. Kampman, and F. van Harmelen, "Sesame: A generic architecture for storing and querying RDF and RDF Schema," in *Proceedings of the first International Semantic Web Conference (ISWC 2002)*, ser. Lecture Notes in Computer Science, I. Horrocks and J. Hendler, Eds., vol. 2342. Sardinia, Italy: Springer Verlag, May 2002, pp. 54–68.

[25] E. Gonçalves da Silva, L. Ferreira Pires, and M. J. van Sinderen, "Supporting dynamic service composition at runtime based on end-user requirements," in *User Generated Services Workshop at the International Conference on Service Oriented Computing (ICSOC 2009)*, November 2009.

[26] L. O. Bonino da Silva Santos, R. Poortinga-van Wijnen, and P. Vink, "A service-oriented middleware for context-aware applications," in *5th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC 2007)*, November 2007.

[27] 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.