Composition of context aware mobile services using a semantic context model

João Paulo Sousa
Departamento de Informática e Comunicações
Instituto Politécnico de Bragança
Bragança, Portugal
jpaulo@ipb.pt

Eurico Carrapatoso
Faculdade de Engenharias/INESC Porto
Universidade do Porto
Porto, Portugal
emc@fe.up.pt

Benjamin Fonseca
CITAB/Universidade de Trás-os-Montes e Alto Douro
Vila Real, Portugal
benjaf@utad.pt

Maria da Graça Campos Pimentel
Departamento de Ciências de Computação
Universidade de São Paulo
São Carlos-SP, Brazil
mgp@icmc.usp.br

Renato de Freitas Bulcão-Neto
Innolution Sistemas de Informática
Ribeirão Preto-SP, Brazil
rbulcao@innolution.com.br

Abstract— **Context-awareness has been regarded as an important feature for mobile services. However, only a few services are sensible to context and the features that are context-aware are still limited. Composition of Web services has received much interest in business-to-business or enterprise application, but not so much interest in business-to-consumer applications. This paper presents iCas, a novel architecture that enables the creation of context-aware services on the fly, and discusses its main components. We compare our approach with similar systems and point out the main differences and advantages. To explore context-awareness to support service composition, iCas uses SeCoM, a semantic model to represent context. The main parts of this model are explained as well the advantages of using a semantic model to represent context. We also describe the use of our approach in an university campus to provide pedagogical features and assist the socio-pedagogical interaction of various types of users.**

Keywords: Context-aware, Services composition, Semantic Web, Web Services

I. INTRODUCTION

It is predictable that in the near future the network mobile environment will be characterized by interaction between services and that those services will be provided to users dynamically and transparently. In this scenario, the use of captured contextual information related to issues such as location, current activities, objects in the neighbourhood and device features plays a crucial role in the simplification of the interaction between humans and the digital world.

Often users only assume the role of consumers of services provided by third parties. For those users a set of useful services and information is provided, but they are aimed at a general market, leaving aside users that would like to take advantage of more personalized services. This paper proposes and describes a service oriented open infrastructure for a mobile network environment. We call this architecture iCas and it allows a user to receive in his mobile device (e.g. PDA, netbook, notebook) context-aware information (e.g. location, time, neighborhood, user profile) and have a set of useful services that are sensitive to his current context. The user can also compose services dynamically in real time to create a new highly personalized envirementwith more features and use or share it as many times as he wants [1].

The remainder of this paper is structured as follows: section 2 discusses related work, section 3 presents some definitions of context, and section 4 introduces the SeCoM semantic model to describe and to provide reasoning about context. Section 5 discusses the several approaches to composing Web Services and the main innovations of our proposal, followed by the description of the OWL-S ontology to support semantic Web Services. Section 6 presents the iCas, a Service Oriented Architecture (SOA) and describes the details of each of itscomponent. Section 7 presentsa scenario for using iCas, a university campus, where iCas will be used to allow users to compose in a had-hoc way new services for enhancing everyday campus life. Section 8 describes the first performance evaluation. Finally, we provide some conclusions and suggestions future work, in section 9.

II. RELATED WORK

A number of context-aware systems has been developed to demonstrate the usefulness of context-aware technology, such as ParcTab [2], which was one of the first systems to offer a general context-aware framework and ContextToolkit [3], which presents a modular context-aware framework with reusable components. Which allows programmers to build more easily interactive context-aware systems based on sensors. These systems donot have an open context model because often the context is described in an object-oriented basis and so the information is strongly coupled tothe programming model.

More recently several studies appeared to support context-aware composition of services, one more generic and others dedicated to mobile environments [4][5][6][7][8].

In [4] the authors present a distributed architecture and associated protocols for service composition in mobile environments. This study emphasizes some factors that allow the composition of services in ad-hoc networks such as mobility, dynamic changing service topology, device heterogeneity, fault tolerance and reliability.

In [5] the authors propose a framework for dynamic composition of context-aware mobile services. The main features are service adaptation to devices and networks, and service adaptation to the user preferences and user location. However the study does not specify which approach is used to compose new services.

SOCAM [6] presents a middleware architecture for rapidly building context-aware services. It provides support for discovering, acquiring, interpreting and accessing context information. It also presents one of the first ontologies that define the main classes of context: person, location, activity and computer entity. Nevertheless, this architecture does not allow the composition of services. MyCampus [7] is a semantic web environment that uses agents that are able to find context information to improve users' campus life. The MyCampus architecture is composed by eWallets (static knowledge containers), which support automated discovery and access to the context. The users can subscribe task-specific agents to assist them in different context tasks using the semantic information in eWallets. These agents are able to discover, execute and compose automatic semantic Web services using the Semantic Markup for Web Services (OWL-S) [9].

In [8] the authors present CACS a framework that enables context-aware composition of Web Services. This framework supports capability matches and goal-driven composition services flow. The CACS architecture uses software agents to discover, compose, select, and automatically execute Web Services using OWL-S.

In [4][5][7][8] we saw that these systems do nothave an open model to describe context, which causes some limitations on sharing context knowledge and context reasoning with external systems. The studies[4][5][8] present architectures that support the automatic composition of services. The user makes a request to the architecture, most of the times to a software agent, whichcollects context information and tries to find the most suitable service, which agrees with the request description. If the agent doesnot find the service or it doesnot exist, then the software agent decomposes the request into multiple sub-goals in order to find the matching services.

In all the cases that use automatic composition, it is a hard task to maintain the details about the rules of services' invocation. These approaches also do not have an open model to describe context, which causes some limitations regarding the sharing of context knowledge and context reasoning with external systems.

III. CONTEXTUAL INFORMATION

The development of an architecture that uses context information requires the perception of the meaning of context and how it can be used. A phenomenon that is observed when someone is asked about what context is that most of the people understand what it is, but they feel that it is hard to explain. For this reason many timescontext definitions are done by enumeration of examples or by choosing synonyms for the context.

The term context was introduced for the first time in [10], referring it location, people, hosts and accessible devices nearby, as well as changes to such things. On [11], the authors define context as location, people in the neighborhood of the user, time and temperature, among others. In [12]context is defined as being the user location,

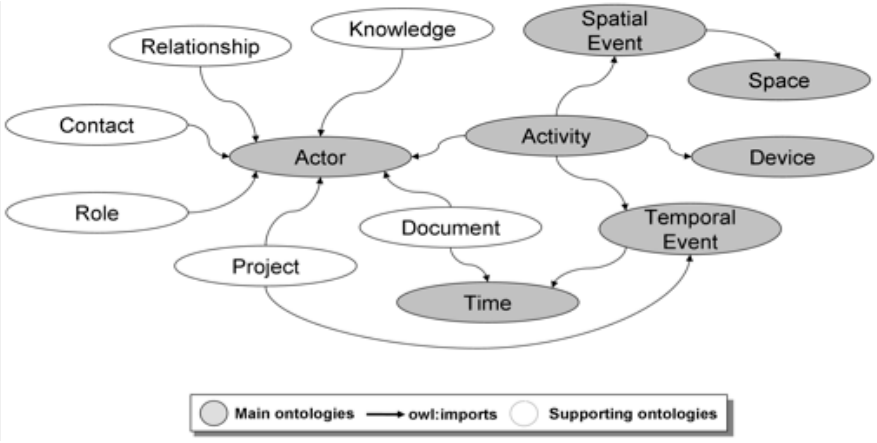

Figure. 1 An overview of the SeCoM model [16].

environment, identity and time information. In [13] the authors have the following interpretation of context: "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the applications themselves". The authors in [14] present another understanding of context. They define it as everything that affects the computation except the explicit input and output data.

There are more context definitions, some described by examples, others described generically and some other in a more explicit way. After we made the review about the meaning of context, we understood context as all the information captured in a non-explicit way and used to create dynamic rules that change the way that services and information are provided to an actor. An actor can be a human or a software agent.

IV. THE USE OF A SEMANTIC MODEL

Contextual information models based on ontologies have been explored in several architectures that support context-aware services (e.g. [6][15][16]). These models allow the cooperation among objects and the discovering, acquisition, inference, and distribution of contextual information. An ontology is defined by R. Gruber as an explicit and formal specification of a conceptualisation of a domain of interest [17]. Ontologies consist of concepts (known as classes), relations (properties), instances and axioms, and on the computing context.Ontologies provide a shared understanding between applications of a domain, typically the common sense about that domain.

To describe the context, we decided to use the semantic model SeCoM (Semantic Context Model), presented in [16].

The use of a semantic model brings about several advantages:

- the possibility of having a high degree of expressiveness and formalism to represent concepts and relations in a context-aware scenario; it allows reasoning about context;
- the use of a semantic information context model, based on Semantic Web standards, makes the exchange, reuse and sharing of context information between context-aware applications easier;
- it decouples the information context model from the programming model, unlike some architectures presented in the section II.

SeCoM is composed of six main ontologies: Actor, Activity, Space, Spatial Event, Temporal Event, Device, Time, and six support ontologies, Contact, Relationship, Role, Project, Document, Knowledge. Fig. 1 shows the SeCoM ontologies and their relationships.

A. The SeCoM Model: An Overview

Considering context modelling, we have developed the Semantic Context Model (SeCoM) [16, 18], which represents the semantics of context information through a set of semantic web ontologies. From the perspective of a context information model, the following is the list of SeCoM's main characteristics:

- it is an effort towards a domain-independent model for context-aware computing ;
- it models classical types of context information such as who (identity), where (location), when (time), what (event and activity) and how (device) [19];
- it is semantic-oriented with high level of expressiveness and formality borrowed from the Description Logics (DL) [20], which is a mature knowledge representation technique representing a subset of first-order logic;
- it is based on ontologies as formalism of context information representation, which is, in turn, based on DL expressiveness and decidability;
- it is a modular model, where each type of context information is represented in a particular ontology to facilitate both its reuse and extension;
- it reuses concepts from general consensusandstandardized Semantic Web ontologies;
- it allows inference of new facts from previous context information due to its ontological semantics;
- it uses Semantic Web standards for representing the structural, semantic and logic views of context information such as Resource Description Framework (RDF) [21] and Web Ontology Language(OWL) [22];
- it is a two-layered context information model, whichfacilitates the task of an application developer to reuse and/or extend the most general concepts of SeCoM.

B. The SeCoM model: A Detailed View

The main ontologies composing the SeCoM context information model are briefly presented next. Further information on the SeCoM model found elsewhere [16, 18, 23].

1) ACTOR ontology: it models the profile of entities performing actions in an ubiquitous computing environment such as people, groups and organizations.

2) TIME ontology: it models temporal information in terms of time instants and time intervals (two or more not null time instants), relations between time instants and intervals (temporal mereology), relations between time intervals (mainly based on Allen's Temporal Algebra [24]), and calendar and clock information (time duration, day of week, month of year, etc.).

3) TEMPORAL EVENT ontology: it models events with temporal extensions such as instant or interval events. It is an extension of the Time ontology because temporal events are defined assubclasses of the class *time:TemporalThing*. In other words, it is able to represent temporal methology between instant and interval events, and temporal relations between interval events. In addition, this ontology also represents information about periodic temporal events such

as the frequency of an event, or even the time duration between occurrences of an event.

4) SPACE ontology: it describes the whereabouts of actors. It models virtual and real-world indoor (e.g. *Room*) and outdoor (e.g. *Street*) places, mereological (e.g. *spatiallyContains*) and spatial (e.g. *isSpatiallyConnectedTo*) relations between places, geographic coordinates (e.g. *latitude*) and *directions* (e.g. *north*) and administrative regions (e.g. *City*).

5) SPATIAL EVENT ontology: it models events with spatial extensions called spatial events, which are subclasses of *spl:SpatialThing* defined in the Space ontology. Spatial events can be represented by two main disjoint subclasses: physical events, which are those occuring in a physical location (e.g. entrance in a meeting room), and virtual events, which include those occuring in a virtual location (e.g. entrance in a chat room). In general, both physical and virtual spatial events inherit all properties, relations and axioms from the classes *spc:PhysicalLocation* and *spc:VirtualLocation*, respectively.

6) DEVICE ontology: it describes computational devices that can be used in ubiquitous computing interactions. The main concern is to represent those devices in terms of their hardware and software platforms, mereological relations between their components, and mobile computing aspects needed for context-aware computing. In general, it models information about storage and battery capacity, multimedia support, wireless and wired network connectivity, operating systems and browsers supported, virtual machines installed, among others.

7) ACTIVITY ontology: it describes activities as sets of spatiotemporal events including the corresponding actors and devices involved in. Thus, this ontology directly imports the Actor, Spatial Event, Temporal Event and Device ontologies, as depicted in Fig. 2. Being modeled as spatiotemporal events, activities reuse the same attributes and relations of both spatial and temporal events. In other words, it is possible to interrelate activities in terms of mereological and spatial relations between their physical/virtual locations, or even in terms of temporal relations between their corresponding time instants and intervals. Besides, it also models activities as of two disjoint types: impromptu and scheduled. The former represents activities occuring in an informal manner (e.g. cocktail meetings), whereas the latter represents activities planned in terms of time and space (e.g. lectures at a conference room). The following is an RDF excerpt of a Computer Science Conference activity represented as a scheduled activity occuring at the "DVR-001" Da Vinci room, which is located on the Conference floor at a university. CS conference started at 10 am on March 7, 2009, and it took two hours long. Activities' participants are described by means of the property actvy:hasParticipant. The actvy: prefix is used to represent the XML namespace for the Activity ontology. In terms of temporal and spatial reasoning, a reasoner could infer that this computing conference still took place at 11 am on the Conference floor.

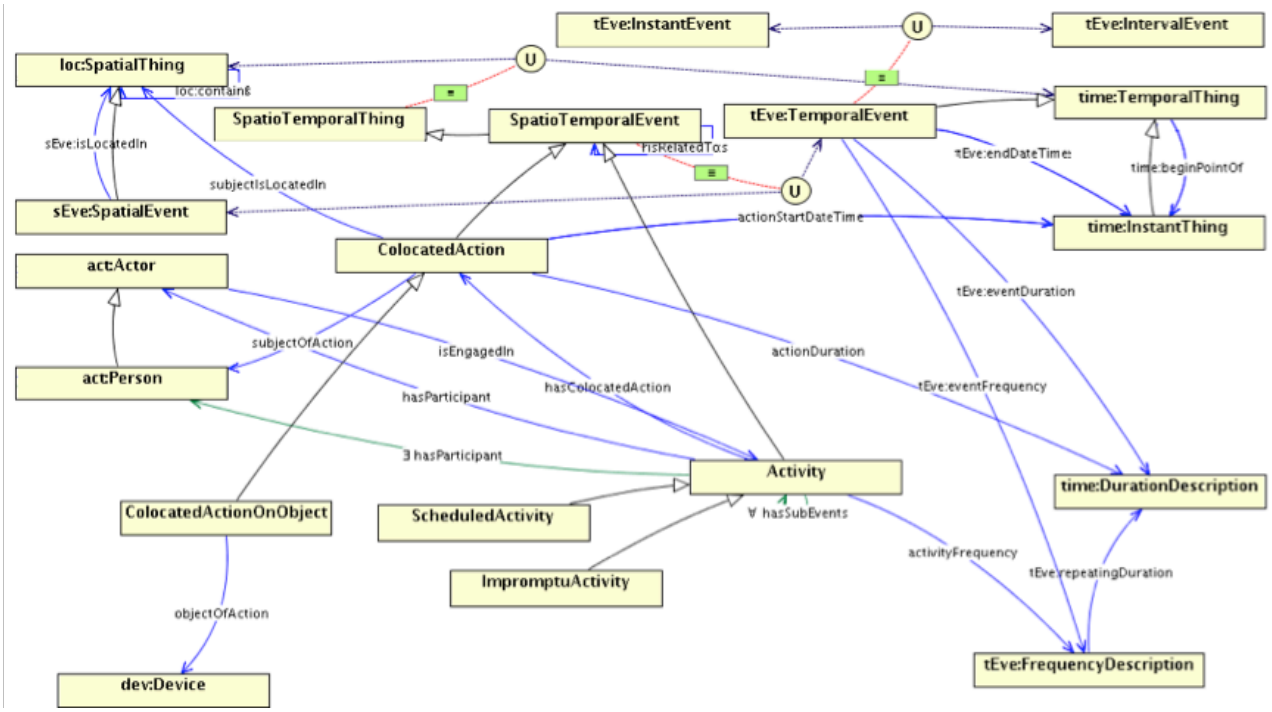

Figure. 2 The Activity ontology.

```
<actvy:CSConference rdf:ID="cmeeting19">
<rdf:type rdf:resource=
"&actvy;#ScheduledActivity"/>
<actvy:hasParticipant rdf:resource="#person19"/>
<sEve:isLocatedIn rdf:resource="#room82"/>
<time:beginPointOf rdf:resource="#bpo67"/>
<time:intervalDurationDescriptionDataType
       rdf:datatype="&xsd;#duration">PT2H
</time:intervalDurationDescriptionDataType>
</actvy:CSConference>
<act:Person rdf:ID="person19">
<act:hasName>Claus Ana</act:hasName>
</act:Person>
<spc:DaVinciRoom rdf:ID="room82">
<rdf:type rdf:resource="&spc;#Room"/>
<spc:placeName>DVR-001</spc:placeName>
<spc:isSpatiallyPartOf rdf:resource="#floor4"/>
</spc:DaVinciRoom>
<spc:ConferenceFloor rdf:ID="floor4">
<rdf:type rdf:resource="&spc;#Floor"/>
<spc:placeName>Conference floor
</spc:placeName>
<spc:isSpatiallyPartOf df:resource="#ipb"/>
</spc:ConferenceFloor>
<time:InstantThing rdf:ID="bpo67">
<time:instantCalendarClockDataType
        rdf:datatype="&xsd;#dateTime">
        2007-03-07T10:00
</time:instantCalendarClockDataType>
</time:InstantThing>
```

V. WEB SERVICES COMPOSITION

The composition of services allows developers and users to create new services or applications, based on a Service Oriented Architecture (SOA) that supports description, discovery and communication. One of the most used SOA technologies is Web Services, due to the advantages already known to the scientific community [25][26][27].

Web Services have often been used for the composition of services. Nowadays there are six approaches to the Web Services composition [28]: WSBPEL [29], Semantic Markup for Web Services (OWL-S) [30], Web Components [31], Algebraic Process Composition [32], Petri Nets [33] and Model Checking and Finite-States Machines [34]. The previous approaches intended to solve the problems found in services composition such as syntax and semantic verification, resource reservation, QoS or deadlocks. In [28] and [35] the authors compare several solutions, based on characteristics such as automatic composition, composition verification, scalability, goal satisfaction, connectivity and non-functional properties.

When the purpose is to implement the composition of mobile services, we have to consider some concerns such as the complexity of the services to be built. For this purpose, wemust find a compromise between simplicity in service creation and flexibility:a more flexible service requires more complex rules and probably specific technical knowledge. In this case the simplicity offered to end users is lost.

To achieve this goal, we chose to compose services in an interactive way: the user gradually generates the composition with ad-hoc forward or backward selection of services. Using this approach for composing Web services requires that they understand their features and how they interact together. The Web Services Definition Language (WSDL) [36] specifies a standard way to describe the interfaces of a Web Service at the syntactic level. However, WSDL does not support the semantic description of services. OWL-S has appeared to fulfill this limitation and uses the OWL language to describe Web Services. OWL-S provides Web services with a set of markup language constructs for describing the

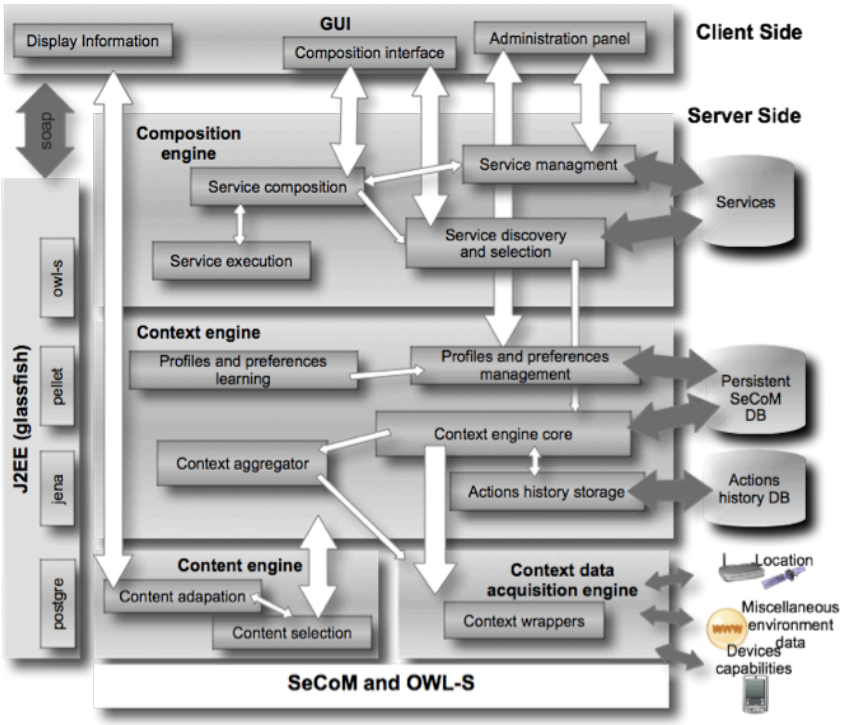

Figure. 3 Overview of iCas architecture.

properties and capabilities in anunambiguous interpretable form to the software agents. OWL-S is a framework that enables automatic discovery and matchmaking tasks, and composition and execution of Web Services.

OWL-S consists of the following classes: ServiceProfile - specifies how the services are announced to the world; ServiceModel - specifies how to interact with the service; ServiceGrounding - specifies the details of how an agent can access the service.

VI. PROPOSED ARCHITECTURE AND IMPLEMENTATION

To support the composition of context-aware services on the fly and provide context-aware information to the users, we propose a Service Oriented Architecture (SOA) based on ontologies. We divide the architecture into four essential engines to explore the potential of context, showed in Fig. 3.

When a user choosesthe service composition IDE, the service discovery component gets the preferences, parameters and interests. With this information and the OWL-S services descriptions, the service discovery and selection selects the services from the service repository to perform a context-based selection, and then delivers it as a list to the IDE.

When a user starts a composition, maybe he knows clearly which tasks he wants to achieve with the composition or perhaps he starts to compose, choosing compatible services that can suggest the creation of a new service. In either situation the service composition is an ongoing process, where the user can add or remove services interactively.

Each time a service is selected to be part of the composition, the service discovery and selection module searches for services (Fig. 4) using data collected from the context engine core and returns further possibilities based on the current context and user policies. The search and selection is only possible due to the OWL-S service description, which allows creating relationships with other ontologies that can describe details about a service type and its features.

The search is performed using the description of the ServiceProfile class, which contains what the services can do, and specifies the input/output types, preconditions and effects. The first selection of services is carriedusing the ServiceProfile hierarchies, which choose the services from a particular category. Then a matching is performed, selecting the services whose input is syntactically compatible with the output of the current service.

Finally a scoring is carried out using the weights of the evaluation parameters defined in the ServiceProfile and a particular evaluation policy, which depends on the service category.

The ongoing user composition is supported by the service composition function, which generates a workflow of services calls. Fig 5 shows an overview of the interactions between the components from the several engines and the GUI , when a composition is accomplished.

By the time that a user finishes the composition, the entity service composition has created a composite service that contains a workflow. This workflow is a composite service that has the three key descriptions of an OWL-S service: service profile, grounding and model, as mentioned in the end of section V. This newly composed service can be saved, executed or used in another service composition task. To store the service, the service composer uses the service management component, and to execute the service it calls the service execution component.

The service management component deals with the services stored in the services container, providing operations such as adding, removing and sharing services using the policies properties. The service container only stores the OWL-S description of the service (service profile,

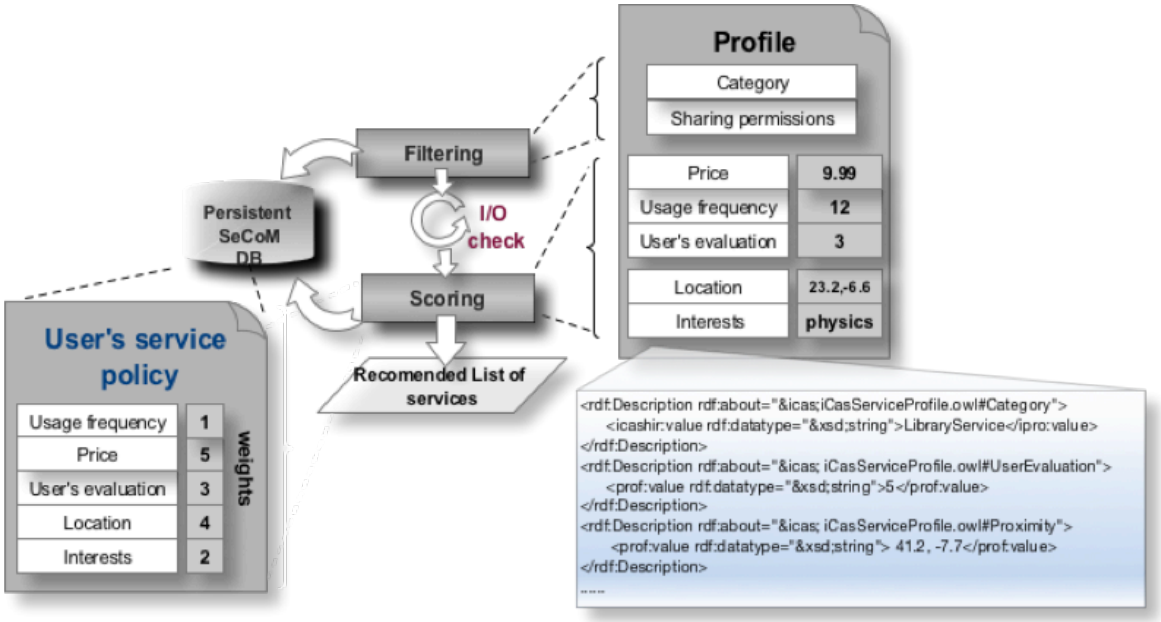

Figure. 4 Service selection mechanism.

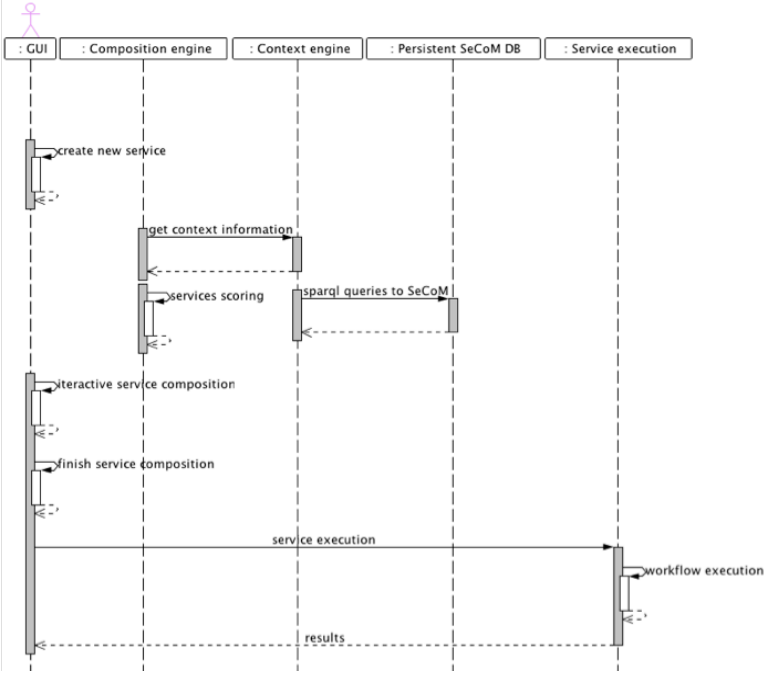

Figure. 5 Composition sequence diagram.

model and grounding). The service functionality is still provided by a third party (e.g. e-learning platform Web service).

The service execution module, using the OWL-S API, provides an execution engine to invoke atomic processes described by WSDL or Universal Plug and Play (UPnP) [37] groundings, and composite processes that use control constructs sequences, unordered, and split. All execution processes that depend on conditional statements, such as if-then-else and repeat-until, are not supported by the API. When the service execution promotes a composition, it follows a workflow to call each individual service and exchange data between them, according to the flow defined by the user.

The context engine is responsible for managing all related context data and for reasoning about context. All context information is stored in a permanent OWL ontology storage system. The context engine core uses the Jena API to store the RDF models of SeCoM using a Postgre DB. This engine is also responsible for extracting knowledge from the SeCoM ontology, using RDF Query Language and Protocol (SPARQL) [38] queries and for making inferences to derive additional statements that are not described explicitly in the SeCoM model. The following code is a SPARQL query to the persistent ontologies, to get all the events related with the Computing subject and their location.

```
PREFIX rdf: <http://w3.org/1999/02/rdf-syntax-ns#>
PREFIX acti:<http://icas.ipb.pt/activity.owl#>
PREFIX spac:<http://icas.ipb.pt/spatial.owl#>
SELECT DISTINCT ?event ?subjectIsLocatedIn ?hasName
 WHERE {
  ?subjectIsLocatedIn spac:hasName ?hasName
  ?hasColocatAction acti:subIsLocatIn ?subIsLocatIn
  ?event acti:hasColocatAction ?hasColocatAction
  ?event a acti:ScheduledActivity
  ?event acti:hasSummary ?hasSummary
  ?event acti:validationStatus true
  FILTER regex(?hasSummary, "Computing")
 }
```

Using OWL's capabilities also enables to make inferences using the Pellet reasoner, (e.g. "if a user is located in the library, he is in university campus", or if a user has interests in "ontologies", and because ontologies has a transitive properties with "semantic web" and this one also related with "context-awareness", hence the user is also interested in "context-awareness").

The context aggregators keep in memory (non-persistent) highly changing dynamic data that is captured from various sources related to an entity (e.g. user, object). For each entity an instance is created that relates that entity with data from the sources (e.g. user's location and data sensor). This component moves the computational charge caused by the frequent data updates into the persistent ontology.

The profiles and preferences management component is responsible for managing the explicit user profile and interests information. Using the administration panel this component allows the user or administrator to manage explicit context such as insert, update and remove profile parameters and user preferences.

The actions history storage component captures each action performed by the context engine core and stores it in the actions history DB. The main actions are search, insert, update and remove, and they are stored in the following format: Action + target Triplet (e.g. update: Bob isMemberOf the Sciences Students Group).

The profile and preferences learning component can change preferences and profile data using historic information of user actions (e.g. if a student queries many times a particular book in the library services, the theme category of that book is added to the hasInterestesIn property of the knowledge ontology). The profile and preferences

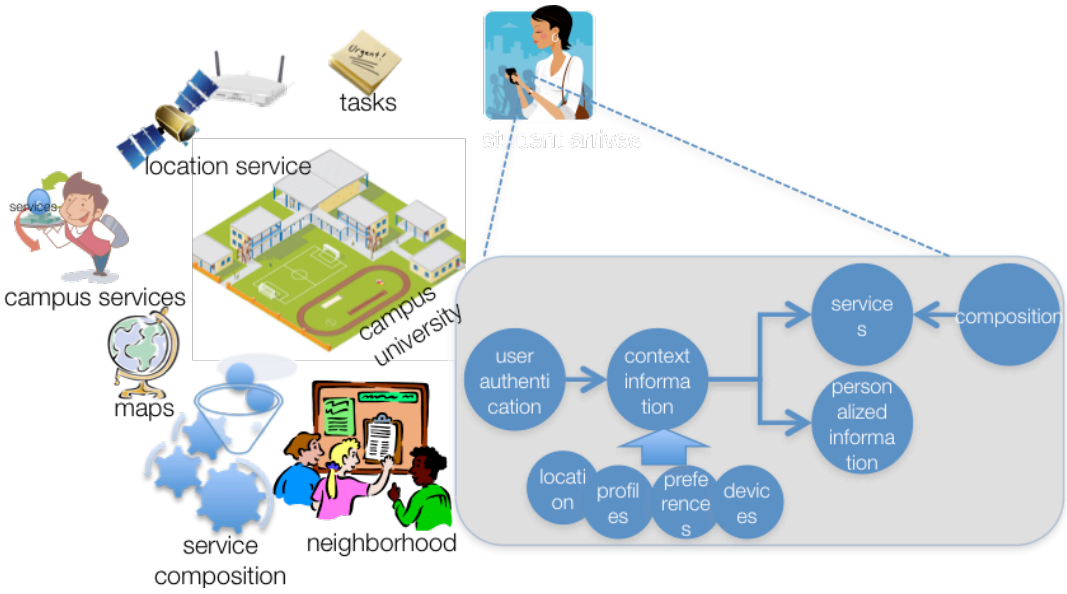

Figure. 6 iCas usage scenario in a university campus.

learning is an independent component. It searches for particular actions stored in the actions history DB, and counts the number of times that an action appears and, accordingly, changes specific parameters defined to be learned. Although this is not an optimal approach, a good solution can only be achieved with a large-scale utilization of iCas architecture and the collecting of user feedback. In the future this mechanism may also evolve to an AI algorithm, searching for patterns in the database.

The context data acquisition engine collects data from several sources, such as location devices, sensors and external services, and prepares the data to be used by the content engine and context engine (e.g. convert units values from a data sensor, or transform the coordinates user's location to a referential location (room 2.1)).

The content engine is composed by two components: the content selection is a timer function that periodically selects the user interests information from the context engine and delivers it to the content adaptation module for transformation. To be able to consult information in arbitrary devices, the information content must be provided in a device-independent way. iCas provides the context information as RSS feeds that are adapted by the content adaptation component. To do that this component adapts the information to the user's device features, using XHTML Modularization [39].

The iCas system is implemented integrally in Java (JDK 1.6.0). The iCas middleware architecture is composed of:

- Composition engine and context information system: Glassfish v2, JAX-WS 2.1, JAXB 2.1, Jena 2.5.4 and OWL-S 1.1.
- Context, profiles and preferences management DB: Postgre 8.2.8.
- Actions history storage management DB: Postgre 8.2.8.
- Ontologies models: SeCoM and OWL-S.

All four engines are implemented in the Glassfish v2 application server, which provides the functions to the GUI client through HTTP, as Web Services. This configuration was chosen to support the ad-hoc composition of services in mobile devices, bringing the reasoner's computational requirements to the server side.

VII. EXAMPLE OF APPLICATION

We have chosen a university campus as a scenario for using iCas (Fig. 6). This architecture aims the support students and teachers in their campus life, helping them to keep updated and improve their social and pedagogical interaction.

When a student arrives at the campus and connects his mobile device to the wireless network he will have to authenticate. This authentication is used to identify the user in a WiFi campus system and in the iCas architecture.

The campus university already has a location system based on the wireless network, which is used to locate the users inside the campus. Besides the service location, the campus also has other services that can provide useful information integrated to the iCas system. Some of the most important services are: an e-learning platform that provides news about lessons, classes contents and others pedagogical information; library services and administrative services.

To implement a scenario we developed an iCas Client application. Fig. 7, shows the iCas client adapted from the Web Service Composer application [40], under the terms of the GNU Lesser General Public License. The main features of iCas consist of providing context-aware information and the dynamic composition of services. For this purpose the user's GUI client has four panels: informative, services composition, maps and administration.

In the information panel the user can access campus information based on his context (e.g., activities, events,

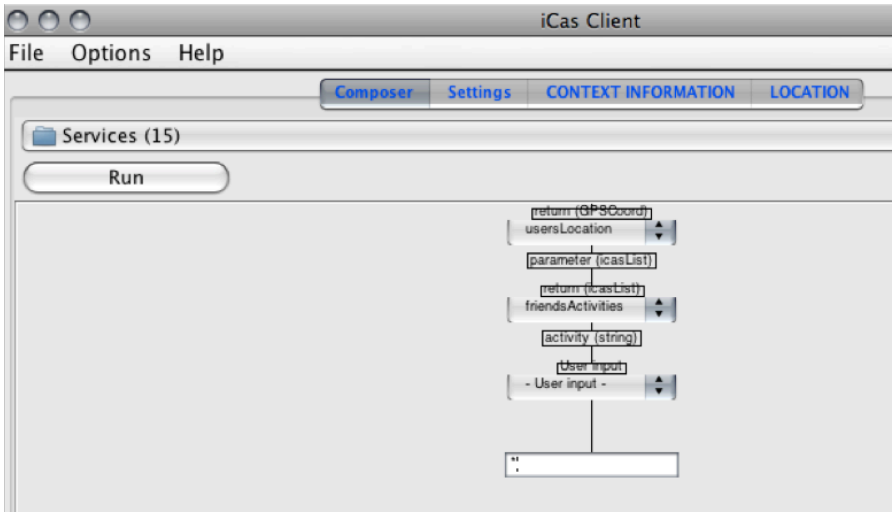

Figure. 7 iCas client prototype – composition panel

news). To compose services in an ad-hoc away the user can use the services composition panel. If the user uses any service that has location output format, information will appear on the maps panel. Any task related with administration, such as changing user profile data and other explicit information, has to be done in the administration panel.

A typical example of this usage scenario is the Friends' Awareness Location Service, in which the user combines the following set of services to get information about the activities of friends that are located in the campus: User's Activities – information gathered by analyzing the user's profile and Users Location Service – provides users locations based on the information gathered on the campus location system aforementioned.

Fig. 8 shows the previous composition built in the composer panel, with Friends' Activities service and Users Location service selected.

When a user starts to compose a new service he selects the composition panel and a list of the available services is presented to him, sorted by the service selection mechanism shown in Fig 5, and described in section VI. During the search for available services he sees two services that might fit his needs: the Friends' Activities service and Users Location service. So, he starts to compose the services and chooses first the Friends' Activities service but when he tries to select the next service to join, he realizes that the User's Location doesnot appear in the list of available services. This happens because itoccurred an incompatible matching betweenthe output of the User's Location (GPSCoord) and the input of Friend's Activities (Activity). So he starts again, selecting first the User's Location service. Next he can find the Friend's Activities service in the list of available service.

The service is available to join for composition, because

Figure. 8 iCas client prototype – maps panel

Table 1 Times to insert context usingthe Context Engine Core.

Number of entries	Add People (ms)	Add Places (ms)	Add Devices (ms)	Add Schedule Activities (ms)	Sum (ms)	Add All (ms)	Upate User Location (ms)
500	2688	3451	2562	4804	13506	19905	9560
1000	5251	6960	5668	10694	28573	34897	35036
2000	9822	14014	11755	20237	55827	66283	41036
5000	24555	34071	26363	51972	136961	154992	97773
10000	52232	71400	57125	110720	291477	300574	130370

now the output and input parameters are syntactically compatible. Next the user writes a wildcard in the input box to know all the activities of his friends. The output (users activities and location) of the service composition is presented in Fig. 8. In the end the user also can save the new composed service to use next time or share it with other users.

VIII. Performance Evaluation

The implementation presented in the section VI is ongoing work. To get the first performance evaluation, we tested some components that we consider critical to evaluate the performance of iCas architecture.

A. Testing Scenario

As seen in the previous section a limited client prototype was implemented which despite being tested by some users it was not ready for a survey-based evaluation. The difficulties in simulating real conditions for the user context, and the composition of services based on the current user context, lead us to evaluate the performance of that components that present more challenges or even problems.

In our test scenario we used two computers connected to the campus wireless network (IEEE 802.11g).

Computer 1 (C1) is an Intel Core 2 Duo 7400 (2.4Ghz) 3GB DDR2 with OS X 10.5.5, and runs the iCas architecture middleware described in Section VI.

The Glassfish, that runs third party Web Services, is installed in computer 2 (C2), an Intel Core 2 Duo T8600 4GB DDR2 with Linux (kernel 2.6.24) as its operating system. Some of the third party services installed in this machine are services provided by the library, and e-learning platform.

B. Context Engine Core test

In this test we intended to get the first performance results from the following main components that are exposed to computationally and I/O intensive processes: context engine core (inserting data and querying for derived contexts), service composition and service execution. We excluded services discovery and selections because the selection is highly dependent on the context engine core.

Table 1 presents the results performed in C1. For each result three measures were made and the table shows the average time in milliseconds (ms) of theses measures.

The graphic in Fig. 9 shows the average time consumed by the Context Engine Core to execute one query, which saves context information into the persistent ontology database. It is possible to observe that the Context Engine performs well in terms of the data volume to store and the variation is gradual and linear. During these tests the persistent ontology database has reached 1GB in disk space.

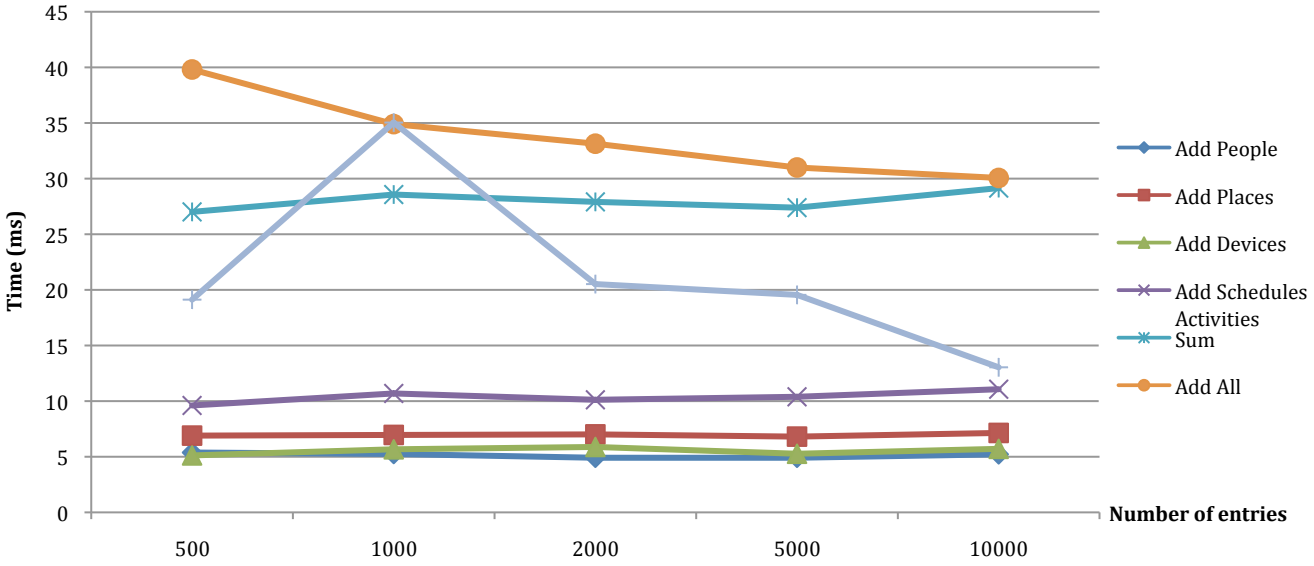

Figure 9 Times to insert context using the Context Engine Core.

It can be observed that the Context Engine Core is able to support intensive loads and that the use of persistent ontologies s not a problem, but it seems to be a good option. Nevertheless, this performance could be improved either by optimizing the DB engine parameters or by using a faster computer to host iCas and the database management system.

To test the reasoning component we executedtwo types of SPARQL queries:

- The first one was a simple query that returns the interests of a specific person and the time average to execute this query was 10ms.
- The second was a more complex query described in the section VI. This query, returns all the events related with a subject and where they are happening. The average time to execute this query was 80ms.

Finally an inference using the Pellet reasoner was executed to explore the resources of OWL language, more specifically the transitive property, already explained in Section VI. In this example the user location was inferred and the average time do to this operation was 304ms.

C. *Service Composition and Execution test*

Table 2 presents the results of testing the Service Selection Mechanism,described in Section VI. The second column shows the time to load the services descriptions and to check its consistency for different numbers of services, specified in column 1. It should be noted that this delay only occurs when iCas is initialized and the services are loaded, which not demand a quick response of this operation as occurs on the services selection process.

When a new service is added or removed to the services repository only that service ontology is added or unloaded, which is a fast operation. The third column shows the time consumed to select the services to deliver to the user.

Table 2 Times of the Selection Mechanism

Number of Services	Time to load and check the consistency (sec)	Services Selection (ms)
10	8,9	25,0
20	31,0	45,0
38	80,7	97,0

The graphic of the Fig. 10 shows the average time needed to load each service. Fig 11 illustrates the average time required to make a service selection.

In Fig. 10, it's possible to observe that the time required to select the services islow which enables to give a quick response to the users' requests.

Observing both figures, it's also possible to realise that the consumption of time per service, required for loading and checking it, and to the services selection, has a minimal increment as the number of services to use increases.

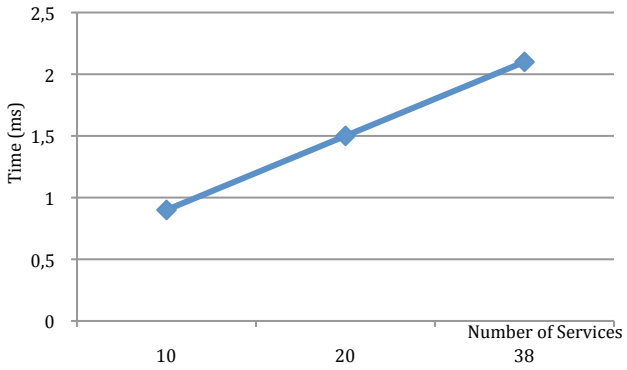

Figure 10 Load and check services process.

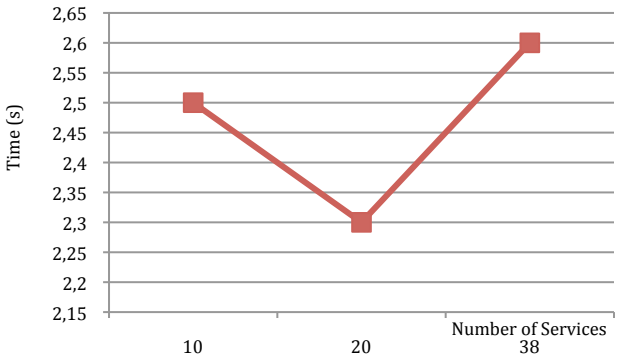

Figure 11 Service selection process.

To test the service composition and service execution we ran a client in C1, which launched a number of threads. Each thread intended to simulate a user that orders a service composition and its execution. Table 3 shows the test results of the Service Composition and Service Execution components.The test consisted in the variation of two parameters: the number of services used in a composition and the number of requests to perform the composition and its execution.

Each thread is responsible to make a unique request and to wait for the response.

The composition of services was the result of services joined in pipeline. The services that were part of this composition were provided by the application server running in the C2 machine, and had an execution time of 20ms. Our intention was to figure out how thesecomponents performed

with differentloads of service composition and execution. The maximum number of services used in a

Table 3 Times of Services Composition

	1 thread (ms)	100 threads (ms)	100 threads (time per thread (ms))	500 threads (ms)	500 threads (time per thread (ms))
2	287,8	26843	268,4	133058	266,1
4	410,5	27339	273,3	135900	271,8
8	528,8	28511	285,1	141471	282,9
10	642,0	30619	306,1		
12	780,0	33702	337,0		
14	820,0	X			
16	927,8	NT			

composition was 16, joint sequentially and the time consumedto execute this composition was 927ms.

To test limit conditions, we used this last composition(16 services)for a load of 100 requests and this component wasnot able to respond and it halted. Analyzing the time consumption of each thread to execute the composition of 12 services, it was 337ms, less than the time a unique thread took to execute the same composition (780ms). It's also possible to see that before the iCas frizzed the time to make and execute a service increased linear and gradually with the increase of the number of services used to make a composition.This problem willbe analyzed in future.

Using a composition of 8 services, this component was able to compose and executed requests made by 500 threads with the average time less than 300ms. In the future we also intend to test parallel compositions and the mix of pipelined and parallel workflow composition.

IX. CONCLUSION

In this paper we have presented iCas, a service-oriented architecture that uses an ontological context model to provide personal and contextual information and to support the composition of context-aware services. The two major contributions of our work are the joint use of a semantic context model (SeCoM), to describe and explore the expression of contextual information, along with the support of dynamic composition, of context-aware services by the user.

A prototype of the iCas platform has been implemented and functional tests have been conducted. Some experimental setups for services composition have been made using the iCas client prototype.

We also present the first performance evaluation in which we tested some of the main components of iCas, and found that the results of having a central server architecture to provide the had-hoc composition of services were encouraging.

A. *Limitations*

The current iCas implementation has some limitations. One is the granularity of services, i.e., which level of granularity the services should have to provide the best services to the user's needs. A fine-grained service addresses small units of functionality or exchange small amounts of data. Consequently, it will be more complicated to the user to build a service and to the architecture to orchestrate more services. Otherwise the coarse-grained services encapsulate more functionality reducing the number of services to make a composition, but they also hide the high level of functionality under one single interface and usually exchange more complex data, which might be harder to deal with.

Another problem is the transformation of standard web services into OWL-S services. There are tools to perform this task, but they have very limited functionality regarding service inputs, outputs and the range of these parameters, which are described by the service profile. If a service has complex datatypes (ex. structures, data collections), these tools are not able to perform that transformation. Some of these complex datatypes have to be described by the user, using the OWL and the service parameters can also be transformed using XSLT transformations, which are very susceptible to syntax errors.

For now, it is not possible to provide execution processing that depends on conditional statements, such as if-then-else and repeat-until, because they are not supported by the API. The API authors already announced the intention to include such functionalities in future versions.

Until now we have not tested the service composition in devices with limited resources, and the client prototype uses the standard Java Virtual Machine and Web Services.

There are also other limitations and challenges related with services compositions and the issues discussed in [28, 41], such as composition correctness, services dynamic availability and services trust,

B. *Future Work*

In the future we intend to finish the implementation of iCas and test it in a real scenario on a university campus. In this scenario we intend to determine how the context-aware mobile technologies can be used to assist pedagogical features and the socio-pedagogical interaction of various types of users.

REFERENCES

[1] J. P. Sousa, E. Carrpatoso, and B. Fonseca, "A Service-Oriented Middleware for Composing Context Aware Mobile Services," in Internet and Web Applications and Services, International Conference on, Venice, Italy, 2009, pp. 357-362.

[2] R. Want, B. Schilit, N. Adams, R. Gold, K. Petersen, D. Goldberg, J. Ellis, and M. Weiser, "The Parctab Ubiquitous Computing Experiment," Mobile Computing, pp. 45-101, 1996.

[3] D. Salber, A. Dey, and G. Abowd, "The Context Toolkit: Aiding the Development of Context-Enabled Applications," 1999, pp. 434-441.

[4] D. Chakraborty, A. Joshi, T. Finin, and Y. Yeshadoi, "Service Composition for Mobile Environments," Mobile Networks and Applications, vol. 10, 2005.

[5] S. Panagiotakis and A. Alonistioti, "Context-Aware Composition of Mobile Services," IT Professional, vol. 08, pp. 38-43, 2006.

[6] T. Gu, H. Pung, and D. Zhang, "A service-oriented middleware for building context-aware services," Journal of Network and Computer Applications, vol. 28, pp. 1-18, /01// 2005.

[7] M. Sheshagir, N. Sade, and F. Gandon, "Using Semantic Web Services for Context-Aware Mobile Applications," in MobiSys 2004 Workshop on Context Awareness, Boston, 2004.

[8] L. Nan, Y. Junwei, L. Min, and S. Yang, "Towards Context-Aware Composition of Web Services," in Fifth International Conference on Grid and Cooperative Computing, Washington, DC, USA, 2006, pp. 494–499.

[9] W3C, "OWL-S: Semantic Markup for Web Services," 2004.

[10] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on, 1994, pp. 85-90.

[11] P. J. Brown, "The Stick-e Document: a Framework for Creating Context-aware Applications," in Proceedings of EP'96, Palo Alto, 1996, pp. 259-272.

[12] N. S. Ryan, J. Pascoe, and D. R. Morse, "Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant," in Computer Applications in Archaeology, Oxford, 1998.

[13] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, London, UK, 1999, pp. 304-307.

[14] H. Lieberman and T. Selker, "Out of context: computer systems that adapt to, and learn from, context," IBM Syst. J., vol. 39, pp. 617-632, 2000.

[15] H. Chen, F. Perich, T. Finin, and A. Joshi, "SOUPA: standard ontology for ubiquitous and pervasive applications," in Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on, 2004, pp. 258–267.

[16] R. F. Bulcão Neto and M. G. C. Pimentel, "Toward a Domain-Independent Semantic Model for Context-Aware Computing," in 3rd Latin American Web Congress (LA-Web'05), Argentina, 2005, pp. 61-70.

[17] T. R. Gruber, "A translation approach to portable ontology specifications," Knowl. Acquis. %@ 1042-8143, vol. 5, pp. 199-220, 1993.

[18] R. F. Bulcão Neto and M. G. C. Pimentel, "Performance evaluation of inference services for ubiquitous computing," in XII Braziliam Symposium on Multimedia and Web Systems, Brazil, 2006, pp. 27-34.

[19] G. D. Abowd, E. D. Mynatt, and T. Rodden, "The human experience," IEEE Pervasive Computing, vol. 1, pp. 48-57, 2002.

[20] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, The Description logic handbook: Theory, implementation, and applications: Cambridge University Press, 2003.

[21] D. Brickley and R. V. Guha, "RDF Vocabulary Description Language 1.0: RDF Schema," W3C, 2004.

[22] G. Schreiber and M. Dean, "OWL: Web Ontology Language Reference. W3C Recommendation," 2004.

[23] R. F. Bulcão Neto, A. A. Macedo, J. A. Camacho-Guerrero, and M. G. C. Pimentel, "Configurable semantic services leveraging applications context-aware," in Proceedings of the 11th Brazilian Symposium on Multimedia and the web, Brazil, 2005, pp. 1-9.

[24] J. Allen, "Maintaining knowledge about temporal intervals," Communications of the ACM, vol. 26, pp. 832–843, 1983.

[25] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services - Concepts, Architectures and Applications," 2003.

[26] H. K. Cheng, Q. C. Tang, and J. L. Zhao, "Web Services and Service-Oriented Application Provisioning: An Analytical Study of Application Service Strategies," Engineering Management, IEEE Transactions on, vol. 53, pp. 520-533, 2006.

[27] M. P. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on, pp. 3-12, 2003.

[28] N. Milanovic and M. Malek, "Current Solutions for Web Service Composition," IEEE Internet Computing, vol. 8, pp. 51-59, 2004.

[29] Oasis, "UDDI v3.0 Ratified as OASIS Standard," 2005.

[30] A. Ankolekar, "DAML-S: Web Service Description for the Semantic Web," 2002.

[31] J. Yang and M. Papazoglou, "Web Component: A Substrate for Web Service Reuse and Composition," in CAiSE '02: Proceedings of the 14th International Conference on Advanced Information Systems Engineering, London, UK, 2002, pp. 21-36.

[32] R. Milner, F. L. Bauer, W. Brauer, and H. Schwichtenberg, "The polyadic pi-calculus: a tutorial," in Logic and Algebra of Specification: Springer-Verlag, 1993, pp. 203-246.

[33] R. Hamadi and B. Benatallah, "A Petri net-based model for web service composition," in ADC '03: Proceedings of the fourteenth Australasian database conference, Darlinghurst, Australia, Australia, 2003, pp. 191-200.

[34] T. Bultan, X. Fu, R. Hull, and J. Su, "Conversation specification: a new approach to design and analysis of e-service composition," in WWW '03: Proceedings of the 12th international conference on World Wide Web, New York, NY, USA, 2003, pp. 403-410.

[35] B. Srivastava and J. Koehler, "Web service composition - current solutions and open problems," in ICAPS 2003 Workshop on Planning for Web Services, 2003.

[36] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1. W3C Note," World Wide Web Consortium (W3C), 2001.

[37] UPnP.org, "UPnP Forum - Standards," in Standards, 2009.

[38] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C, 2008.

[39] W3C, "XHTML™ Modularization 1.1, W3C Proposed Recommendation," 2008.

[40] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic Composition of Web Services using Semantic Descriptions," in Proc of Web Services: Modeling; Architecture and infrastructure workshop in conjunction with ICEIS2003, 2003.

[41] M. Bourimi, F. Kühnel, and D. e. D. I. Abou-Tai, "Tailoring collaboration according privacy needs in real-identity collaborative systems," in CRIWG 2009 - 15th Collaboration Researchers' International Workshop on Groupware Peso da Régua, Douro, Portugal, 2009.