A Survey of Dynamic Service Composition Approaches for Ambient Systems

Aitor Urbieta, Guillermo Barrutieta
Computer Science Department
Mondragon Unibertsitatea
20500 Mondragon, Spain
[aurbieta,gbarrutieta]@eps.mondragon.edu

Jorge Parra, Aitor Uribarren
Software Technologies Area
Ikerlan
20500 Mondragon, Spain
[jparra,auribarren]@ikerlan.es

ABSTRACT

Ambient systems are populated by autonomous devices interconnected to one another that supply a variety of functionality that is eventually used by the users. One of the biggest challenges in this research area is to establish automatic mechanisms to compose, in a dynamic way, services on demand. The objective of the composition is to satisfy user needs combining the existing services, when there is not a service in the environment that can perform it. The aim of this research work is, firstly, to have a look at and analyze the state of the art regarding service composition in ubiquitous environments and, secondly, to propose a set of properties and characteristics that can be applied to this type of systems to allow their classification and ultimately their scientific comparison.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures

Keywords

service composition, ubiquitous computing, pervasive computing, pervasive services

1. INTRODUCTION

An ambient system is an ubiquitous environment that establishes a mechanism to provide the users with all the functionality of the devices, components and local and distributed software applications in a flexible, integrated and almost transparent way for the end-user [39]. All this implies that the computational devices are integrated, embedded and distributed in the physical media and surroundings around us. It is, therefore, a non-intrusive distribution of all these devices with the goal of offering user-centered functions. Nevertheless, these environments cannot be seen as a set of nodes and computational independent components. They have to be seen as the medium that allow the users to complete a set of simple or complex tasks using the available services [39]. In order to do that, the devices must be conscious of the existence of the rest of the devices and they must be able to establish coalitions among them with a limited human intervention if at all [14]. Therefore, the environments should be as autonomous as possible, requiring a minimum high level human intervention [27]. The ubiquitous computing tries to get away from the paradigm of the user in front of a PC as the unique interaction mechanism with software components and tries to promote the use of mobile and embedded devices [1]. All this will lead us to change many of the current application design patterns to adapt to the new challenges opened by this type of, ever increasing number of, devices like dynamicity, mobility, limitation of resources, connectivity, etc.

Nowadays, we are already surrounded by many devices. We are referring to, not only PCs, PDAs or mobile phones as computational nodes, but to a wide variety of devices capable of computational processing nearby and around us [8]. For instance: cars, TVs, DVD players, multimedia portable devices, video game players, domestic appliances, etc. Although we have the necessary infrastructure to achieve the goal proposed by Weiser, the reality is that we are far from it. The main reason that explains this paradox is that the majority of the devices are isolated. They offer a specific and concrete function but lack a real capacity to collaborate among themselves [37].

In the scientific literature, we can find many examples of ambient systems, also identified as ubiquitous environments, pervasive environments, intelligent or active spaces, etc. Many projects have been developed to achieve "intelligent homes" [40, 19, 24, 13] but we still have a long way to go. We have no doubt that the most extended and accepted paradigm for the development of ubiquitous environments is the Service Oriented Architecture (SOA). The functionality offered by the computational nodes is achieved by a service and the applications can be developed combining these services. This solution meets perfectly the requirements of an ubiquitous system.

One of the main features of ambient applications, that sets them apart from the traditional programming techniques, is their capability to adapt to a changing context [3]. They must be sensitive to changes that occur in the environment continuously, such as the location and state of users and devices. The services offered to the user must adapt dynamically to the changes in context depending on the information obtained from the user as well as from the environment itself. This context adaptation offers the user the most ad-

SOMITAS 2008, February 14, Quebec, Canada

DOI 10.4108/ICST.AMBISYS2008.2842

equate services and in the most suitable way. In this way, the interaction mode between users and applications change completely: it is not the user who adapts to the restrictions of use imposed by the applications, it is the other way around.

As we have already said, the main limitation that exists today to make real the ambient environments is the little collaboration capability among the devices. Users often try to execute tasks that the devices present around the users are not capable of doing in isolation. If these devices are not capable of offering the expected or desired result, it might be possible that this result might be obtained by the collaboration among different devices [16, 30]. One of the strong points of the Service Oriented Architecture, and a really useful one in the case, is the capability to compose simple services in order to obtain more complex ones. SOA offers the necessary infrastructure to do so. The composition of services is therefore a key factor that we must take into account. The composition of services can be defined as the possibility of combining functionalities offered by different services to obtain a new functionality. Different techniques have been proposed for the service composition such as AI planning [31] or Workflows [33]. In a ubiquitous environment, this composition presents a clear detail that sets it apart from the composition of static environments: the dynamicity in the presence of devices and in the context that require dynamic techniques of re-composition to adapt to the changes occurring in the environment. In order to provide end-users with the most adequate functionality it becomes necessary to discover the most appropriate services for their contextual situation and compose them in the most suitable way. The main problem that we have to face in order to compose services might be the variety of terms used to describe each service [30]. Therefore we must come up with an agreement to represent the concepts that take part in each service in order to combine them in an adequate way.

The current article is structured as follows: Section 2 establishes the key features for service composition in ubiquitous environments; in Section 3, a set of properties and characteristics for grouping, classifying and comparing the proposed ubiquitous service composition techniques are described; this is followed by Section 4, where the main characteristics of the approaches studied are described; finally, in Section 5 with the conclusion obtained from the study, a set of new areas of research, according to the authors, are outlined.

2. RELEVANT ASPECTS FOR UBIQUITOUS COMPUTING ENVIRONMENTS

Next, we describe the essential aspects for Service Composition in Ubiquitous Computing Environments.

2.1 Context-awareness

Ubiquitous computing is closely related to the environment. The main reason for this relation is due to the heterogeneity and ubiquity of the entities that are in communication in those types of environments. Both aspects require instant discovery and composition of the services offered in the situation where the user is, tasks that the user is doing, etc. That is to say, adaptation is necessary depending on context or situation.

The concept of context-awareness [29] has increasingly gained importance in the area of distributed systems since the 90's. Since it seemed to be a promising solution for a lot of problems which have been implied by the usage of mobile terminals in ever-changing environments [34]. For this reason, it is necessary to define and manage a context model in a machine processable way. A well defined model is the key to being able to develop context-aware applications, but the wide range of context information and its properties are different. That is why different approaches have been proposed in the last years as described by Strang and Linnhoff-Popien [34]. In this article, six different types of context model approaches are defined based on the data structure for the management of contextual information. One of the conclusions of this comparative is that the ontology based model is one of the best models because it allows us to represent the knowledge with high expressiveness, as well as making knowledge distribution and reasoning over the context easier. That is why we consider that the ontology based model allows a better service selection and therefore an improved service discovery and composition process.

2.2 Quality of Service (QoS)

As with the context, the Quality of Service (QoS) is decisive for user experience enrichness in ubiquitous computing environments. During the discovery and composition process, not only are functional requirements taken into account but also the non-functional requirements such as disponibility, latency, confidentiality, integrity, processing cost, necessary bandwidth, etc. Lately, more approaches take into account the Quality of Service as a necessary attribute to have in mind on application developing for Ubiquitous Computing environments [21, 20].

2.3 Service Oriented Architectures (SOA)

In ubiquitous computing paradigm described by Weiser [39] physical environments are described as full of computational resources that behave in a non-intrusive way with the user, facilitating user interaction and access to the resources and information, everywhere and everytime. In these environments the interaction between the user and the system is intuitive, pleasant and natural. One of the main features of this kind of environment is the mobility of computational resources, that makes these environments highly open and distributed, which need ad-hoc deployment and execution, that is to say, situation customized deployments, integrating available software and hardware resources at suitable place and time. This dynamicity is only possible if the devices are considered as autonomous and independent resources inside the system. Thus SOA paradigm is particularly suitable for this kind of environment since, in this architectural model, applications and devices that support them are abstracted as loosely-coupled services that can be integrated in big systems. Most of SOA implementations use Web Services.

2.3.1 Semantic Web Services

Semantic Web Services [18] set up a new research area that comes from the synergy between both Web technologies with the faster adoption and with more expectations in later years: Web Services and Semantic Web [2]. Although traditional Web Services allow communication among different platforms and operating systems, they only share syntactical information without any machine-understandable semantics: only the data structure exchanged amongst service

clients and providers without specification of the meaning of these messages. Thus, dynamic service composition or discovery, are difficult tasks in traditional service infrastructures and human intervention has always been required. The Semantic Web Services paradigm tries to keep the intervention of end-users to the minimum, automating as much as possible the discovery, composition, invocation and interoperability of Web Services.

2.3.2 *Service discovery and composition*

Bearing in mind that the ontologies allow the semantic description of any domain of knowledge, adding domain services to Web Services produce the Semantic Web Services. These ones allow the processes to express and describe which activities are able to do in such a way that is possible to discover services in a concrete environment and to establish a collaboration among them to develop high level tasks with higher added value [18] in a dynamical way. That is to say, given a specific task and some Web Services which are able to execute the given task combining them in an appropriate manner, the Semantic Web Services technologies allow us to compose automatically the execution flow and the inputs and outputs of the Web Services, to execute in a collaborative way the given task, without human intervention. That is why they are considered as an intelligent infrastructure for advanced systems deployment. During recent years some efforts are being made by the scientific community at the Semantic Web Services specification area. These research works produced some service description languages, like: OWL-S, WSDL-S, WSMO, SWSL and SAWSDL.

The advantages that the context and quality of service aware semantic discovery bring are the key to ubiquitous environments. They allow to perform syntactical and semantical attributes based service discovery and not only in syntactical ones (not semantically annotated) as it is the case of simple Web Services. Bearing in mind these attributes, it is possible to achieve a more suitable collaboration of the devices in the environment.

3. CRITERIA FOR THE EVALUATION OF THE APPROACHES

In the following lines the four groups of features that have been defined to classify the existing approaches are described, as well as the results of the comparatives:

3.1 Language expressiveness

This group contains the features related to the expressiveness of the language used to describe the published services as well as the user tasks (see Table 1):

Service modelling language (mod): Language employed to describe functional and non-functional features of the services.

Semantic-aware (sem): It indicates if the language allows to express semantically the functional and non-functional attributes of the services for a later use at discovery and composition process.

User task description method (task): This criterion indicates the way that the required task or goal is defined. For example as an operation, as a conversation, as a goal to achieve, etc.

Published services description method (pub): This factor indicates how the published services are described. For example based on operations, conversations, etc.

3.2 Composition model

In this group, the features related to the method used for the creation of the composition or plan (synthesis) are described (see Table 2):

Composition creation technique (tech): Used technique to fulfill the required objective/goal. For example AI Planning techniques, Workflow techniques, Data-Mining, etc.

Composition language (lang): The language used to describe and execute the service composition that has been previously created using the composition technique. For example, scripts, BPEL4WS, etc.

Context-aware (cont): It indicates if the composition process takes into account context information (i.e. user location, user profile, etc.) during the creation of the plan.

QoS-aware (QoS): It indicates if the composition takes into account quality information (i.e. latency, memory usage, etc.) during the creation of the plan.

User intervention (user): It indicates if the user participates during the process of composition (i.e. selecting services).

3.3 Execution model

This group describes the features related to the mechanisms employed for the execution of the plan (see Table 3):

Orchestration/Choreography (comp): This criterion specifies whether the composite services are executed in a centralized (orchestration) or a distributed (choreography) manner.

Replanning during the execution (repla): This criterion specifies if the execution technique supports replanning (i.e. due to context changes) during the execution of the composition.

Dynamic/Static service binding (bind): This criterion specifies if the services used in the composition are defined in a static or a dynamic way.

3.4 Execution environment

The features related with the context in which the approach is executed are grouped here (see Table 4).

Infrastructure (infra): The infrastructure that is used in the approach is being specified by this criterion. It can be based on an existing architecture or on a new one.

Service discovery method (disc): Technology or protocol used for the the discovery of published services.

Device that supports the execution (device): It indicates where the composition is executed.

4. OVERVIEW OF THE COMPARED APPROACHES

Based on the results of the comparison, a short overview of the approaches is described in the next lines, emphasizing the most important features of each one.

Proposed in [17, 32], Task Computing faces service composition towards user desires, despite adapting the user to the services of the environment. Thus, the system allows us to achieve an environment that is completely focused on final users using DAML-S service descriptions. Ni [23] also tries to abstract users from the complexity of composition by OSOA, facing like many authors [27, 38, 26, 25, 23] AI Planning techniques in order to solve service composition problems. The proposal described in [13, 40] by the Nokia

Table 1: Results of the comparative of the Language expressiveness group

Approach	mod	sem	task	pub
[13, 40]	WSDL, UPnP	Syntactic	Workflow	Simple Services
[5]	OWL-S	Semantic	Composite FSM	FSM adaptive behaviour
[9]	WSDL	Syntactic	Workflow	Simple Services
[38]	WSDL	Syntactic	Context information + Goal	Simple Services
[3]	Key-Value pairs	Syntactic	Workflow	Simple Services
[26]	OWL-SC	Semantic	Simple Service	Simple Services
[4]	Key-Value pairs	Syntactic	Workflow	Simple Services
[10]	OWL-S	Semantic	Workflow	Simple Services
[19]	OWL-S	Semantic	Workflow	Simple Services
[25]	OWL-S	Semantic	Workflow	Simple Services
[12]	UPnP	Syntactic	Workflow	Simple Services
[24]	OWL-S	Semantic	Workflow (3rd party service providers)	Simple Services
[15]	Not specified	Semantic	Simple Service	Simple Services
[16]	WSDL	Syntactic	Workflow (State Chart Diagrams)	Simple Services
[1]	WSDL	Syntactic	Message Sequence Charts	Simple Services
[22]	WSDL	Syntactic	Simple Services	Simple Services
[30]	OWL-S	Semantic	Workflow	Simple Services
[21, 20]	COCOA-L (OWL-S)	Semantic	Workflow	Workflow
[7, 6]	DAML-S	Semantic	Workflow (Description-level Service Flow)	Simple Services
[17, 32]	OWL-S	Semantic	Workflow	Simple Services
[23]	OWL-S	Semantic	Goal (desired state)	Simple Services
[27]	DAML+OIL	Syntactic	Goal	Simple Services
[28]	WSDL	Syntactic	Goal (converted to a Workflow)	Simple Services
[36, 35]	OWL-S	Semantic	Goal (converted to a abstract plan)	Simple Services

Table 2: Results of the comparative of the Composition model group

Approach	tech	lang	cont	QoS	user
[13, 40]	AI Planning-Depth First Search 1	Scripts	No	No	Yes
[5]	AI Planning-Finite State Machines	OWL-S Process	No	No	No
[9]	ECF+Mobile Agents	Executable Choreography Language	Yes	No	No
[38]	AI Planning-HTN-SHOP2	BPEL4WS	Yes	No	No
[3]	Workflow (Service Selection)	Not specified	Yes	No	No
[26]	AI Planning-HTN	Directed Acyclic Graph (DAG)	Yes	Yes	No
[4]	Workflow (Service Selection)	XML	Yes	Yes	No
[10]	Workflow (Service Selection)	Not specified	Yes	No	Yes
[19]	Workflow (Service Selection)	XML	No	Yes	No
[25]	AI Planning-HTN+LCW	Not specified	Yes	No	No
[12]	Workflow	Functional Task Description (FTD)	Yes	Yes	Yes
[24]	Workflow (Service Selection)	OWL-S Process	Yes	No	No
[15]	Data-Mining	Not specified	Yes	No	No
[16]	Agent Conversation based	State Chart Diagrams	Yes	Yes	No
[1]	Workflow (Service Selection)	BPEL4WS	Yes	Yes	No
[22]	Ubiquitous Coordination Model	Not specified	Yes	Yes	No
[30]	AI Planning-Backward chaining-STRIPS	OWL-S Process	Yes	No	No
[21, 20]	Workflow-Finite State Automatas	COCOA-L (OWL-S)	Yes	Yes	Yes
[7, 6]	Workflow (Service Selection)	Description-level Ser. Flow (DAML-S)	No	No	No
[17, 32]	Workflow (Service Selection)	Not specified	Yes	No	Yes
[23]	AI Planning-STRIPS	Not specified	Yes	No	Yes
[27]	AI Planning-Blackbox planner-STRIPS	Lisp	Yes	No	Yes
[28]	Workflow (Service Selection)	BPEL4WS	Yes	No	Yes
[36, 35]	Workflow+AI Planning-CSP (validation)	OWL-S Process	Yes	No	No

Table 3: Results of the comparative of the Execution model group

Approach	comp	repla	bind
[13, 40]	Orchestration	No	Dynamic
[5]	Orchestration	Yes	Dynamic
[9]	Choreography	No	Dynamic
[38]	Orchestration	No	Dynamic
[3]	Orchestration	No	Dynamic
[26]	Orchestration	No	Dynamic
[4]	Orchestration	Yes	Dynamic
[10]	Orchestration	No	Dynamic
[19]	Orchestration	No	Dynamic
[25]	Orchestration	No	Dynamic
[12]	Orchestration	No	Dynamic
[24]	Orchestration	No	Dynamic
[15]	Orchestration	No	Dynamic
[16]	Choreography (Agent based)	No	Dynamic
[1]	Orchestration	Yes	Dynamic
[22]	Orchestration	No	Dynamic
[30]	Orchestration	No	Dynamic
[21, 20]	Orchestration	No	Dynamic
[7, 6]	Distributed Orchestration	Yes	Dynamic
[17, 32]	Orchestration	No	Dynamic
[23]	Orchestration	No	Dynamic
[27]	Orchestration	Yes	Dynamic
[28]	Orchestration	No	Dynamic
[36, 35]	Orchestration	No	Dynamic

Table 4: Results of the comparative of the Execution environment group

Approach	infra	disc	device
[13, 40]	End-User Programming Framework (EUP)	UPnP, WS, Jini	Mobile Phones
[5]	Not specified	Not specified	Not specified
[9]	Executable Choreography Framework (SOAP)	Not specified	Embedded Systems
[38]	SOAP	UDDI	Not specified
[3]	IPOJO over OSGI	OSGI	Gateway (Home)
[26]	SOAP	Not specified	Not specified
[4]	Smart Space Middleware over OSGI	OSGI	Gateway
[10]	Jini	Reggie (Jini lookup service)	Not specified
[19]	Home Appliances Integration Unit (HAIU)	Not specified	Home entertainment syst.
[25]	OntoPlan	Not specified	Not specified
[12]	UPnP	UPnP	Resource Constrained devi.
[24]	OSGI (UPnP & Jini)	OSGI	Gateway (Home)
[15]	Service Provisioning Middleware (COSEP)	Not specified	Not specified
[16]	SOAP	UDDI	Not specified
[1]	SOAP	Not specified	Not specified
[22]	CB-Sec Framework	LW-UDDI	Embedded PCs
[30]	Multi-agent architecture and SOAP	Not specified	Not specified
[21, 20]	SOAP	Not specified	Not specified
[7, 6]	Architecture developed for the approach	Group Based Service Disc.	Mobile Devices
[17, 32]	Task Computing Environment (SOAP)	UPnP	PDA
[23]	SOAP	WS-Discovery, UPnP	Mobile Devices
[27]	GAIA	Discovery module of GAIA	Not specified
[28]	SOAP	UDDI, Multicast DNS	iPaq
[36, 35]	SOAP	Not specified	Not specified

Research Center Cambridge, suggests an architecture where the decision of final device selection to use at the composition is in users' hands. This architecture uses DSF1 composition algorithm to find the possible combinations in the selected environment and is considered as a semi-automatic composition approach because it needs user's intervention during the process of composition.

In [12] the desired objective is defined as end-to-end, eg."link (video-source) with (display)" and the system tries to find the most suitable services to link both ends using a simple search algorithm. Thus, policies and resource availability are taken into account in order to choose the most adequate solution. For Vukovic and Robinson [38] the service composition problem can be viewed as a planning problem using HTN with SHOP2. In this case, the composition is defined using BPEL4WS; this description is transformed to SHOP2 objectives; and the planner resolves the plan to fulfill these objectives. Finally, once the new plan is resolved, it is translated into BPEL4WS again. Changes in the environment (appearing and disappearing services and mobile users) where the composition is being carried out may require online replanning of the composition, facilitating the adaptation of the application to contextual changes, as it is being made in [5, 4, 1, 7, 6, 27].

Qui et al. [26] propose OWL-SC, an extension of OWL-S that supports context information and complete service descriptions. Furthermore Qui et al. propose to use HTN as a planning technique but they introduce the concept of "Plan Library" to store the plans and not to recalculate again the same plan. Ontoplan [25] also uses HTN, but it is combined with Local Closed World reasoning (LCW) to avoid redundancies in the composition.

The use of workflows is proposed by many authors like Ranganathan and McFaddin [28], Bellur and Narendra [1], etc. For instance, Ben Mokhtar et al. [21, 20] propose the COCOA system, where user tasks are defined as OWL-S processes and the final composition is obtained using a matchmaking algorithm (using Finite State Machines (FSM)) that creates the composition using the fragments of the advertised service conversations. Vallee et al. [35, 36] and Maamar et al. [16] propose to combine multi-agent techniques with Semantic Web Services to provide dynamic composition of context-aware services in order to achieve more adequately the user desired task. In [15] a data-mining based approach is used, which considers context information and the historical usage of the services to define new service compositions.

The use of ontologies for service representation/description is proposed by many authors [10, 19, 30]. These articles propose to use semantical concepts to annotate the attributes (inputs, outputs, preconditions and effects) of services in order to enhance the matchmaking and compositions processes. Others, like Lee et al. [4] use OSGi as service description language so that a composition expressed as a graph and represented using XML is built like a new OSGi service, using existing services. OSGi is also considered in [3] and [24], although in the last one the composition process is made by 3rd party experts instead of final users.

In most of the previously mentioned works the execution of the composition is centralized. There is a central coordinator that controls the flow of the composition (orchestration). However, Chakraborty et al. [7, 6] propose decentralized protocols (distributed orchestration) to compose services, using DAML-S as service description language and [16] proposes to deploy the execution of the composition using agent conversations (choreography) to deploy the control of the execution and invoke the correct services.

There are some works that propose the use of semantic descriptions for service composition algorithms, where it is worth pointing out the following: Carey et al. [5] propose to extend AI based planning method, criticized by its need of replanning in order to adapt to contextual changes. To improve this lack(shortfall), the authors propose to use Finite State Machines (FSM) to model the adaptive behaviour (non-functional behaviour) that should have the resultant composition. These FSMs are in charge of adapting to new context changes without replanning. The Executable Choreography Language (ECL) described in [9] allows injection of service control flow modifications using the Executable Choreography Framework (ECF) in a non-invasive way, that is to say, without manual changes of the original service flow defined by the user.

5. CONCLUSIONS

An analysis of the above mentioned ubiquitous environment systems and different methods for composition shows the following conclusions:

Workflow systems or methods are mainly used in cases where the petitioner has a well-defined process model, but where a program needs to find the atomic services in an automatic way to fill the process. On the other hand, planning AI techniques are used when the petitioner does not have a process model but a set of preferences and restrictions and makes a plan on the basis of his/her preferences and restrictions. It is difficult to say which is the best method of the two for an intelligent environment because it could vary according to the type of system to be developed and the cases to be used.

The composition is centrally implemented in almost every approach, which means that composition is carried out by a single device instead of a distributed composition made by a number of collaborating devices. A distributed and collaborative composition model, with a dynamic set of devices and services, resembles, in a more appropriate way, the ubiquitous computing environment envisioned by Weiser [39].

Apart from being centralized, some of the systems use devices with advanced computing capabilities for composition like, for instance, PDAs and PCs. But the majority of devices located in ubiquitous environments do not have high processing capabilities. Therefore, considering the use of a distributed composition approach with limited resource devices is not completely realistic yet, because most required technologies and standards are still too heavy to be deployed in such devices.

Many systems require a very strong and necessary participation of final users in the composition process; however, a pervasive environment should try to minimize this. This kind of environment should be able to assist the users with the declaration of desired tasks or goals. These tasks, injected into the environment, should force the devices to be aware of user needs and work together to satisfy users' needs.

BPEL4WS and OWL-S are the main languages used in the studied proposals. However, works related to ubiquitous computing composition have not been made with WSMO, SWSF, WSDL-S or SAWSDL due to the fact that these ones are more recent proposals and OWL-S and BPEL4WS are

more matured technologies that already have a set of developing tools and APIs. There is then an open way for the use of the new Semantic Web Services proposals in intelligent and ubiquitous computing environments or their possible adaptation to resource constrained devices such as embedded systems.

5.1 Areas requiring a deeper technical study

Current devices, applications and systems do not have the same level of reactivity or context awareness described in the areas proposed by Lassila [14], Weiser [39] or the ISTAG [11] because they are either simple device systems which behave in a basic way like opening doors when a person is detected around, switching on the light when a person is coming, redirecting a call to the closest telephone, etc.; or they are just centralized systems with a central computer and devices located around it just receive working commands [37].

In order to achieve a more intelligent, advanced and impromptu behaviour with the ability to reason and learn independently, intelligent environment devices must have some degree of intelligence and must be light, small and consume low energy, so they can be embedded in every day gadgets. This kind of intelligent environments should be built only in embedded systems, instead of in systems requiring big devices like PCs, which create artificial and unreal scenarios, very far away from the idea of ubiquitous computing. But these two requirements, (small devices and high processing capabilities) are not easy to handle. Normally, the smaller the device is, the less "intelligent" it can be. Therefore, it becomes necessary to find either the right balance between the two variables or just choose one of them.

A happy solution should be investigated, based on a light (small and with low energy consumption) system but still intelligent enough to be able to reason over user context, perceived inputs and the desired objectives, and capable of communicating and coordinating with other devices located in the environment to perform more complex tasks. It becomes necessary to set up a value chain that provides intelligence to the environment, so it acts with reactivity and context-awareness based on user preferences. This could be the resultant value chain: Sensorization and user profiles -> Context-awareness (based on ontologies) -> Reasoning (based on rules) -> Composition (based on strategies) -> Operation -> Devices and gadgets. Initial learning is parallel to the value chain and will determine the initial rules and further adaptation, that is to say, learning will allow us to establish the content of the rules, the user profiles, the user behaviours, etc.

Attempts to apply the above technologies to embedded platforms to get more reactive and user awareness environments were made but results were uneven. Thus, the adaptation of Semantic Web technologies for embedded platforms in intelligent environments is still a challenge.

The key concepts applied to the embedded semantic web services technologies for the interoperability and automatic composition of business processes can be completely applied to ambient systems. This would lead to an automatic collaboration of ambient services to get a more sophisticated behaviour of the environment for user benefit. An infrastructure for *Ambient Semantic Web Services* would facilitate the design and development of these scenarios according to user needs, giving them a more advanced model to interact with the environment. Therefore, the research and study of this field is a technological challenge to deal with which may require the following:

- Adaptation of current composition execution approaches (orchestration / choreography) to create and carry out distributed business processes in ubiquitous computing environments (ambient systems) where devices have restrictions in relation to their processing and communication capability.

- Development of new specifications for distributed composition of ambient services to meet some specific requirements in these scenarios.

6. ACKNOWLEDGMENTS

This work was partly supported by a grant from the Basque Government's SAIOTEK program. The authors would also like to thank the ARLES team (*http://www-rocq.inria.fr/arles*) for their cooperation in this work.

7. REFERENCES

[1] U. Bellur, N. C. Narendra, I. I. T. KReSIT, and I. Mumbai. Towards service orientation in pervasive computing systems. *International Conference on Information Technology: Coding and Computing*, 2:289–295, 2005.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web, 2001.

[3] A. Bottaro, J. Bourcier, C. Escoffier, and P. Lalanda. Autonomic context-aware service composition. *2nd IEEE International Conference on Pervasive Services*, 2007.

[4] W. L. C. Lee. S. Ko, S. Lee and A. Helal. Context-aware service composition for mobile network environments. In *4th International Conference on Ubiquitous Intelligence and Computing (UIC2007)*, 2007.

[5] K. Carey, D. Lewis, S. Higel, and V. Wade. Adaptive composite service plans for ubiquitous computing. In *2nd International Workshop on Managing Ubiquitous Communications and Services (MUCS 2004)*, December 2004.

[6] D. Chakraborty. Service discovery and composition in pervasive environments, 2004.

[7] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Service composition for mobile environments. *Journal on Mobile Networking and Applications, Special Issue on Mobile Services*, 10(4):435–451, January 2005.

[8] S. D. Cotofana, S. Wong, and S. Vassiliadis. Embedded processors: Characteristics and trends. pages 17–24, May 2001.

[9] T. Cottenier and T. Elrad. Adaptive embedded services for pervasive computing. In *Workshop on Building Software for Pervasive Computing - ACM SIGPLAN conf. on Object-Oriented Programming, Systems, Languages, and Applications*, 2005.

[10] C. Hesselman, A. Tokmakoff, P. Pawar, and S. Iacob. Discovery and composition of services for context-aware systems. volume 4272, 2006.

[11] ISTAG. Scenarios for ambient intelligence in 2010. *Technical Report, EU Commission*, 2001.

[12] G. Kaefer, R. Schmid, G. Prochart, and R. Weiss. Framework for dynamic resource-constrained service composition for mobile ad hoc networks. *UBICOMP, Workshop on System Support for Ubiquitous Computing*, 2006.

[13] D. N. Kalofonos and F. D. Reynolds. Task-driven end-user programming of smart spaces using mobile devices. Technical Report NRC-TR-2006-001, Nokia, 2006.

[14] O. Lassila and M. Adler. Semantic gadgets: Ubiquitous computing meets the semantic web. In *Spinning the Semantic Web*, pages 363–376, 2003.

[15] S. Y. Lee, J. Y. Lee, and B. I. Lee. Service composition techniques using data mining for ubiquitous computing environments. *International Journal of Computer Science and Network Security*, 6(9):110–117, 2006.

[16] Z. Maamar, S. K. Mostefaoui, and H. Yahyaoui. Toward an agent-based and context-oriented approach for web services composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5):686–697, 2005.

[17] R. Masuoka, B. Parsia, and Y. Labrou. Task computing - the semantic web meets pervasive computing. pages 866–881, 2003.

[18] S. A. McIlraith, T. C. Zeng, and H. Zeng. Semantic web services. *IEEE Intelligent Systems and Their Applications*, 16(2):46–53, 2001.

[19] A. Mingkhwan, P. Fergus, O. Abuelma'Atti, M. Merabti, B. Askwith, and M. B. Hanneghan. Dynamic service composition in home appliance networks. *Multimedia Tools and Applications*, 29(3):257–284, 2006.

[20] S. B. Mokhtar, D. Fournier, N. Georgantas, and V. Issarny. Context-aware service composition in pervasive computing environments. In *RISE*, pages 129–144, 2005. crossref: DBLP:conf/rise/2005.

[21] S. B. Mokhtar, N. Georgantas, and V. Issarny. Cocoa: Conversation-based service composition in pervasive computing environments. *Proceedings of the IEEE International Conference on Pervasive Services*, 2006.

[22] S. K. Mostefaoui, A. Tafat-Bouzid, and B. Hirsbrunner. Using context information for service discovery and composition. *Proceedings of the Fifth International Conference on Information Integration and Web-based Applications and Services*, 3:15–17, 2003.

[23] Q. Ni. Service composition in ontology enabled service oriented architecture for pervasive computing. In *Workshop on Ubiquitous Computing and e-Research*, 2005.

[24] H. Pourreza and P. Graham. On the fly service composition for local interaction environments. In *IEEE International Conference on Pervasive Computing and Communications Workshops*, page 393. IEEE Computer Society, 2006.

[25] A. Qasem, J. Heflin, and H. Muñoz-Avila. Efficient source discovery and service composition for ubiquitous computing environments. 2004.

[26] L. Qiu, Z. Shi, and F. Lin. Context optimization of ai planning for services composition. In *ICEBE '06: Proceedings of the IEEE International Conference on e-Business Engineering*, pages 610–617, 2006.

[27] A. Ranganathan and R. H. Campbell. Autonomic pervasive computing based on planning. *International Conference on Autonomic Computing*, pages 80–87, 2004.

[28] A. Ranganathan and S. McFaddin. Using workflows to coordinate web services in pervasive computing environments. *Proceedings of the IEEE International Conference on Web Services*, pages 288–295, 2004.

[29] B. N. Schilit and M. M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.

[30] M. Sheshagiri, N. M. Sadeh, and F. Gandon. Using semantic web services for context-aware mobile applications. *Second International Conference on Mobile Systems (MobiSys 2004), Applications, and Services - Workshop on Context Awareness*, 2004.

[31] E. Sirin and B. Parsia. Planning for semantic web services. *Semantic Web Services Workshop at 3rd International Semantic Web Conference*, 2004.

[32] Z. Song, Y. Labrou, and R. Masuoka. Dynamic service discovery and management in task computing. *Mobiquitous*, 00:310–318, 2004.

[33] B. Srivastava and J. Koehler. Planning with workflows-an emerging paradigm for web service composition. *Workshop on Planning and Scheduling for Web and Grid Services*, 2004.

[34] T. Strang and C. Linnhoff-Popien. A context modeling survey. 09 2004.

[35] M. Vallee, F. Ramparany, and L. Vercouter. Dynamic service composition in ambient intelligence environments: a multi-agent approach. In *First Workshop on YR-SOC*, 04 2005.

[36] M. Vallee, F. Ramparany, and L. Vercouter. Flexible composition of smart device services. In *International Conference on Pervasive Systems and Computing (PSC05)*, pages 165–171. CSREA Press, 2005.

[37] J. I. Vazquez, D. López de Ipiña, and I. Sedano. Soam: A web-powered architecture for designing and deploying pervasive semantic devices. *International Journal of Web Information Systems*, 2007.

[38] M. Vukovic and P. Robinson. Adaptive, planning based, web service composition for context awareness. *2nd International Conference on Pervasive Computing*, 2004.

[39] M. Weiser. The computer for the 21st century. *Scientific American*, 256(3):94–104, 1991.

[40] P. Wisner and D. N. Kalofonos. A framework for end-user programming of smart homes using mobile devices. 2007.