A Survey of Service Composition in Ambient Intelligence Environments

Thanos G. Stavropoulos[1,2] • Dimitris Vrakas[1] • Ioannis Vlahavas[1,2]

[1]*Aristotle University of Thessaloniki*

[2]*International Hellenic University*

Abstract This article presents a comparative review of systems performing service composition in Ambient Intelligence Environments. Such environments should comply to ubiquitous or pervasive computing guidelines by sensing the user needs or wishes and offering intuitive human-computer interaction and a comfortable non-intrusive experience. To achieve this goal service orientation is widely used and tightly linked with AmI systems. Some of these employ the Web Service technology, which involves well-defined web technologies and standards that facilitate interoperable machine to machine interaction. Other systems regard services of different technologies (e.g. UPnP, OSGi etc) or generally as abstractions of various actions. Service operations are sometimes implemented as software based functions or actions over hardware equipment (e.g. UPnP players). However, a single service satisfies an atomic only user need, so services need to be composed (i.e. combined), in order to provide the usually requested complex tasks. Since manual service composition is obviously a hassle for the user, ambient systems struggle to automate this process by applying various methods. The approaches that have been adopted during the last years vary widely in many aspects, like domain of application, modeling of services, composition method, knowledge representation and interfaces. This work presents a comparative view of these approaches revealing similarities and differences, while providing additional information.

Keywords web services, service composition, ambient intelligence, ubiquitous computing

1 Introduction

Ubiquitous or pervasive computing (UbiComp, PerComp) is one of the technological paradigms of the future also referred to as the third wave of computing. As Mark Weiser coined the term, such systems are able to perceive user needs and interface with them in an intuitive way (Weiser 1999). The computer fades in the background and interfaces with the user through everyday life physical objects (e.g. wearable devices, electronic appliances). As a result, users do not have to enter the computer's environment but computers fit the user space instead, without requiring his full attention.

The vision of Ambient Intelligence (AmI) slightly extends these ideas by incorporating intrinsic intelligence in pervasive systems. Human-computer interaction then, becomes even more hassle-free and non-intrusive by automations.

Another paradigm, tightly linked with AmI systems, is Service Oriented Computing (SOC). Service orientation is in essence the practice of using abstractions of data and functions into

services which users or applications can consume in a universal way. The corresponding setting derived from SOC is the Service Oriented Architecture (SOA) which further entails additional well-defined web standards. These technologies greatly contribute to the reuse of existing implementations and remote collaboration between different enterprises. Most importantly, they offer the required abstractions for high-level sophisticated AmI systems. SOA has become so interconnected with AmI systems that it is almost considered as a requirement. Meanwhile, as devices and sensors required in AmI become more compact and woven into everyday-life objects, services again come in handy as they expose their data and functions, forming the so-called Internet of Things. Thus, a large amount of information and functions become available and the need for efficient discovery, selection and composition of those services arises. Obviously, manual browsing and selecting from a large service registry is a huge hassle for users that contradicts the AmI vision. In addition to that, users most of the time, require complex tasks instead of the simple ones offered by atomic services. In other words, they need to combine two or more atomic services together in order to form sequence of services (composite service) that achieves sophisticated tasks.

Meanwhile, the Semantic Web technologies have emerged and greatly contributed to automating these tasks. Although primarily designed for the Web, ontologies can aid in semantically describing services and contribute in realizing AmI systems. Semantic annotations for services render their descriptions machine interpretable and enable the automatic discovery and selection even through reasoning.

AmI systems venture to automate the composition process to enhance user experience. This task essentially disintegrates to the tasks of automatic discovery and selection of services. Many known systems employ different technologies and methodologies resulting in end-products with widely different aspects. Most of the systems indeed use semantic annotations for services but even so, different modeling languages result in systems with different aspects. Technologies have yet to converge and standardize. There are more than one service and composite service description languages, different ontologies have been designed resulting in different vocabularies, thwarting true semantic interoperability. As technologies try to converge, it is interesting to observe which technologies and methodologies are employed by current AmI systems and what kind of results are presented by them.

The work presented in this article extends the review paper by Urbietta (2008). The review is extended in both terms of investigated aspects and number of systems.

The rest of the article is organized as follows: The next section reviews Service Composition in detail. The third section presents a comparative review of AmI systems, also detailing their aspects. Finally, conclusions from the comparisons are drawn and presented in the final section.

2 Overview of Service Composition

The service oriented approach is so interconnected with AmI systems that it can be considered an essential element. Services offer universal and remote access, extensibility, collaboration and reuse of existing functionality. Particularly, the idea of considering data and functions as services is well suited for AmI. Users and high-level applications need to be deprived of the hassle to manage low-level functions and data. Services offer the required abstraction so that users can transparently accomplish at least atomic tasks by calling (consuming) a service. In other cases, a middleware is formed (Park 2011) that can also be service-based. Another notion is Task-oriented computing, where users are presented with solutions to desired tasks rather than the methods (services) to manually accomplish tasks and the actual workflow can be transparent to the user. A similar notion is the one of Activity Oriented Computing where software systems are built and configured based on models of user activities (Sousa 2006).

However, user needs can rarely be satisfied by atomic tasks only. In practice, services are combined, either in a serial manner or even asynchronously (their workflows are woven together). Users manually browse available services in repositories, discover desired services and feed them with the required inputs. They also have to manually control the dataflow among the various services. This hassle increases along with the number of available services.

AmI systems strive to automate this process by automatic service discovery, selection, matching, composition and execution of the composite service. Known systems propose methodologies for all or focus on some of these stages. As the proposed methodologies range and make use of different standards the resulting systems can share a lot of aspects in some stages and widely differ in others.

The rest of this section details the most dominant implementation of Services which is the Web Service technology, Semantic Web Services or simply the use of Semantic annotations and ontologies and finally presents the problem of Service Composition.

2.1 Web Services

Web Services are a specialization of the notion of Services that entails well-defined web standards. The term generally refers to any given web portal offering some service, but strictly speaking, it refers to specific implementations incorporating standard web protocols (i.e. WSDL,

SOAP, UDDI), most of which are XML-based. In SOA, there has to be provisioning by a Service Broker, clients or Service Consumers do not directly browse servers or Service Providers but only after the Broker intermediates between them. On the other hand, generic Services include Web Services and more technologies e.g. UPnP[1]. Generic services most of the time are not meant, by nature, to operate over Web e.g. UPnP targets home use and functions in a local network of UPnP devices. AmI Systems mostly regard Web Services, benefiting from universal standards in descriptions and less often from remote function.

Atomic web services are the building blocks for the composition process. Each service carries out an atomic task, and is much similar to a Remote Procedure Call (RPC). There are many frameworks that help realize web services like JAX-WS[2], WCF[3], OSGi[4] etc.

Fortunately, technology convergence has emerged as the W3C standardized the Web Service Description Language or WSDL[5]. WSDL is an XML language that syntactically defines web services, by type-defining their operations, along with their inputs and outputs, and bindings to their implementations. In WSDL 2.0, the interfaces of a service (portTypes in WSDL 1.1) are initially abstractly defined, having many operations of typed inputs and outputs. The concrete section or grounding of the service defines one or more bindings for each interface so that the service can actually be called and executed (i.e. consumed). Bindings inherently support HTTP, SOAP and Java but can be extended to any given implementation. Finally, the service has endpoints that assign URIs to each binding so that implementations can be sought.

Services in SOA are typically published on a suitable meta-data registry, the so-called Service Broker. Such centralized registries are UDDI and Jini. UDDI registries are XML-based and widely used, also as means for Business to Business communication. They provide clients with the WSDL descriptions of services, and hence with the means to call the desired service. Jini[6] is a service-oriented programming model that extends Java technology. It offers a flat service registry, the so-called Jini Lookup Service, which enables service discovery by matching Java data types and attributes of services. Other service discovery mechanisms, that consider dynamicity, are passive e.g. use multicast, where clients instantly get notified of new or obsolete services. Typically, service browsing is a manual process which introduces a hassle for the user that scales badly. Additionaly (syntactic only) descriptions naturally make sense to humans only so this

[1] Universal Plug and Play Forum (UPnP) - http://www.upnp.org/
[2] Java API for XML Web Services (JAX-WS), JSR 224 - http://www.jcp.org/en/jsr/detail?id=224
[3] Windows Communication Foundation (WCF) - http://msdn.microsoft.com/en-us/netframework/aa663324
[4] Open Services Gateway initiative (OSGi) - http://www.osgi.org/
[5] Web Service Description Language (WSDL) - http://www.w3.org/TR/wsdl
[6] Jini - http://www.jini.org/

process cannot be automated. Semantic annotations can tackle this problem by rendering descriptions machine-readable.

2.2 Semantic Web Services

Semantic Web technologies have emerged to reform and organize the vast Web infrastructure. The Web contains a massive amount of unorganized data scattered around web pages that has to be discovered manually, traditionally through the keyword-search approach. In the absence of other means, information can only be sought and appreciated by human users. Semantic Web technologies strive to provide a uniform machine-interpretable representation for this data in order to automate the process of discovery and interpretation. XML-based languages, like RDF[7] and OWL[8], enable the design of taxonomies of concepts for this data to achieve that goal. As a result, semantically annotated data on the Web can be sought, parsed and reasoned upon by software agents in reference to one or more ontologies. Mishra (2011) provides a survey of reasoners for that purpose.

Web services are now a vital part of the Web itself and widely used by common web users as well as the industry. Much ongoing work tries to define standards to apply Semantic Web technologies on web services i.e. semantically annotate their descriptions. The so-called semantic web services provide semantic descriptions for many of their aspects (e.g. operations, inputs, outputs and even preconditions and results) in reference to some ontology, as opposed to syntactic descriptions e.g. WSDL. The main advantage of this methodology is the high level of automation in web service discovery, invocation and composition using semantic matching.

An initial attempt at a standardized technology was DAML-S[9], that later evolved into OWL-S[9]. OWL-S is an upper ontology for services that disambiguates common aspects and is focused to achieve the aforementioned goals. As the OWL-S definition states, the semantic web focuses on providing automations for knowledge discovery, while OWL-S focuses on getting things done i.e. services.

OWL-S contains three main classes that can be related to a service: Service Profile, Service Process Model and Service Grounding. The Service Profile is a (sometimes abstract) description of what the service accomplishes, its inputs and outputs. However, services usually accomplish tasks by taking some actions that have effects or some preconditions. OWL-S also defines preconditions and results (i.e. effects) for services. This quadruple of data is referred to as IOPR (Inputs,

[7] Resource Description Framework (RDF) - http://www.w3.org/RDF/
[8] Web Ontology Language (OWL) - http://www.w3.org/2004/OWL/
[9] DAML-S and OWL-S - http://www.daml.org/services/

Outputs, Preconditions and Results) or IOPE (for Effects). The Process Model details the workflow of the service's operation so that consumers of the service can analyze it, monitor its execution or more interestingly coordinate its workflow with that of another service (i.e. intertwine the execution of more than one services e.g. in Thomson 2008). The Process Model can also be used to describe composite services. Finally, the Service Grounding offers a concrete form of the service so that it can actually be invoked – instantiated. OWL-S inherently offers a WSDL grounding/mapping, but can be extended to support any other technology as well. All in all, OWL-S and WSDL are much alike: the Service Grounding term is similar to WSDL's binding: Input and Output descriptions are similar to WSDL's message (WSDL 1.1) and type descriptions and processes to operations. However, even if OWL-S and WSDL overlap, they are actually complementary, since WSDL offers syntactic only descriptions and OWL-S doesn't contain a binding to actually invoke a service unless there is Grounding to e.g. WSDL. OWL-S conforms to the SOA architecture where services are primarily advertised by a broker in a service registry and discovered then by clients-consumers. It is compatible with any known service registry (e.g. UDDI) but clients and service developers have to share a common ontology reference for efficient matching of concepts. Another approach to semantic annotations of services is SAWSDL[10], which directly extends WSDL, without the need of a mapping.

2.3 Web Service Composition

Service Composition entails many problems mainly due to dynamicity and vague service descriptions. Since the World Wide Web is dynamic by nature, web services cannot be treated as static objects. They are created, altered and destroyed in a dynamic manner and therefore the composer needs to adapt and find alternatives or substitutes to missing services. The same goes for smart environments as service providers enter and leave it. In addition, service descriptions are mostly fully comprehensive by humans only, although semantic annotations contribute in this area. After all, a composite service solution can only be evaluated by the user who originally needs it.

An inherent property of service composition is service adaptation. Adaptation refers to the ability of dynamically adjusting to available services only, providing the user with less optimal solutions instead of no solution at all. In other words, when services are missing from the environment, a settlement in user requirements can be met and less preferable services can be used. This functionality is mainly required in dynamic systems (where services come and go), and enhances the reliability of the system. Adaptation and composition can happen in a way

[10] Semantic Annotations for WSDL (SAWSDL) - http://www.w3.org/2002/ws/sawsdl/

transparent to the user or not. In the latter case, the user is presented with a set of alternative solutions – suggestions to choose from.

Approaches to composition mainly employ A.I. Planning or service selection/matching. There are also some deviations or variations e.g. entailing Multi-Agent systems.

Planning inherently fits the service composition problem, as it aims to find a series of actions to transform an initial state to a state containing goal predicates. Although not explicitly mentioned in its definition, OWL-S descriptions are ideal for the application of Planning techniques. The descriptions of IOPR or IOPE, can serve directly in Planning by regarding services as actions. In other words, the problem of finding a series of services with well-defined IOPRs for a composite service is transformed into the problem of finding a series of actions with the same IOPRs that reaches a goal state. The advantage of this method is the reuse of a library of known planning algorithms.

Matching or simple service selection is the brute force approach of iteratively checking each service trying to match with the required functionality. Finding a series of services for the composite service includes matching atomic services at each step. Naturally, syntactic matching across standard service descriptions is a broad matching criterion and results in many meaningless composite services that finally can be accepted or turned down by the user. On the other hand, semantic descriptions enable semantic discovery and matching of the services that greatly improves and automates the process.

3 Comparison of Systems

This section overviews the most important systems that perform service composition in Ambient Intelligence environments. In order to present the systems in a uniform way that facilitates comparison among them and drawing of significant conclusions, all systems are presented following five common views/aspects that are further decomposed in a larger number of features that are also used as comparison criteria. Namely these views are: the environmental setting of the application, the formulation of the composition problem, the use of semantics, the composition process and finally end-product features. Each of these views is discussed in detail in the next sections and an indicative number of systems are presented in tables 1 to 5 respectively.

3.1 Environmental Setting

The first view that was investigated, namely environmental setting, concerns the environment in which the systems are embedded or applied. In other words, we investigate the nature of the application domain, the types of devices (sensors and actuators) that are present in it

and the exact role of the services in the system. Table 1 presents an indicative number of systems and their environmental settings. The application domain, the role of services and the devices used are presented in columns 2,3 and 4 respectively.

System	Application Domain	Role Of Services	Devices
Paluska 2008	Entertainment, Communication	Video Streaming	Multimedia, Mobile Phones
Sousa 2006	PC applications	Video Streaming, Web Browsing Text Editing	PCs
Mokhtar 2007	Entertainment, Information	Video/Audio Streaming	Multimedia
Thomson 2008	Entertainment	Video/Audio Streaming, User Control	Multimedia (UPnP)
Ranganathan 2004	Smart Environments	Video/Audio Streaming, Presentations, Videoconferencing, Messaging	Multimedia, Biometric, Wearable's, e.t.c.
Beauches 2008	e-Shopping	Online Payments, e-Book Searching, e-Book Downloading	PCs
Messer 2006	Entertainment	Video/Audio Streaming Image Handling	Multimedia (UPnP)
Preuveneers 2005	Communications	Video/Audio Streaming Transcoding	Multimedia
Bottaro 2007	Communication	Text Display (TV), Text to speech Lamp Adjustment (Color, Brightness)	PCs, Multimedia, Lighting
Iacob 2008	Information	Localization (GPS, GSM)	GPS, Mobile phones
Hesselman 2006	Entertainment, Information	Video/Audio Streaming, Transcoding	Car multimedia, Mobile phone
Yokohata 2006	Entertainment	Video/Audio Streaming	Multimedia (UPnP), Mobile phones
Lagesse 2010	Traffic Control	Video/Audio Streaming Transcoding	Traffic Cameras, PDAs, Mobile Phones, PCs,
Davidyuk 2010	Smart Environments	Video/Audio Streaming File Hosting	Multimedia, RFID, Mobile Phones
Vukovic 2007	Entertainment Information	Restaurant Searching, Routing Translation, Speech Synthesizing	GPS, Mobile phones
Lee 2007	e-Shopping	Product Browser, Localization (RFID) Shopping List from Fridge	RFID, PDAs, Embedded PCs
Kaefer 2006	Entertainment	Video/Audio Streaming Transcoding	Multimedia Mobile phones
Lee 2006	Smart Environments	Video/Audio Streaming Switching on/off Lamp	Multimedia Lighting
Maamar 2005	Travel	Flight, Hotel, Car Booking, Attraction Search	-
Bellur 2005	Health	Medical Exams Sharing (X-rays, MRIs)	PDAs, Embedded PCs
Mostefaoui 2003	AAL	Video/Audio Streaming	Embedded PCs
Sheshagiri 2004	e-Shopping	Localization (IP), Online Orders	PDAs
Masuoka 2003	Smart Environments	Information Sharing, File Hosting, Presentation, Routing	PDAs
Paolucci 2009	Travel	Flight, Hotel Booking	-
Ibrahim 2009a	Smart Environments	Image management, File hosting/compressing	PCs
Robinson 2004	Smart Environments	Heating/Telephone Control, Scheduling, Localization, File Printing Location	PCs, Peripherals, Home Appliances
Santofimia 2011	Smart Environments	Intruder Identification (fingerprints, iris, face)	Biometric

***Table 1**. Environmental Setting*

3.1.1 Domain of Application

This field shows the main field of application of the corresponding AmI system. Some systems retain their generality and do not define a particular domain of application. Indicative systems of an abstract domain of application are (Carey 2004), (Qiu 2006) and (Chakraborty 2005). Some systems only indicate a domain as an example for reference. Some regard their systems domain-independent but implement services for a particular domain as a showcase. After all, some of the proposed algorithms can be just as effective for other problems except service composition. Paluska (2008) for example applies his planning approach in non-pervasive problems as well e.g. crisis management, recipes and hardware design. The methods themselves are indeed domain-independent, so any system could possibly be adapted by developing the proper middleware. However, there is no telling if a system would perform just as well, in all domains.

Domains of application mainly range from Multimedia (or generally Entertainment), Smart Rooms/Offices (i.e. Smart Environments) and transportation to news services. Other examples include Aspect Oriented Programming (Cottenier 2005).

Entertainment primarily concerns Multimedia and gathers the most interest. Systems on this domain mostly employ UPnP/DLNA technologies. DLNA compliant TVs and Set-Top-boxes offer services for streaming audio and/or video to them (act as Media Renderers) or controlling their playback and pulling audio/video streams (act as Media Players). Devices where the media is stored act as Media Servers like PCs or NAS (Networked-attached storage) which is basically a networked hard drive. Then, the same or different devices like Smartphones or PCs can control the flow of media streams from storage to renderers (act as Media Controllers). Naturally, these systems do not necessarily employ Web Services, but rather compose DLNA functions (Messer 2006). Preuveneers 2005 regards services that aid peer to peer broadband communications in forms of speech, audio and video in any combination. Thus, this approach has elements that can be met in systems of the Multimedia and Communication domains. Davidyuk 2010 system mostly forms multimedia applications as well. Iacob 2008 considers Multimedia and location awareness only as a motivating scenario for design.

Smart Environments include Smart Homes and Smart Offices/Meeting Rooms etc. Smart Offices include streaming presentations from laptops or Smartphones to a dynamically selected projector, transferring data and contacts between collaborators and remote printing (e.g. in Masuoka 2003). Informational services can be employed in a number of cases where informative data like the weather, news, local points of interest, local viewings etc. are required. Ranganathan 2004 enables videoconferencing, messaging and presentations (in addition to playing music or

video files). Thus, the Smart Office domain shares elements with the Multimedia domain (and UPnP devices can serve both domains).

3.1.2 Role of Services

This criterion is tightly linked to the previous one, as the domain of application more or less dictates the nature of available services. Such a list is useful for previewing what kind of services a system provides and has been experimented with. Services have syntactically and sometimes semantically, well-defined input and output data. In practice, multimedia services are often PlayMovie(File Movie), PlayAudio, Print(File document), etc. Communication services can be Send message to somebody, start videoconference, Display Presentation, Play Music, Play Video (Ranganathan 2004).

Preuveneers (2005) uses a component-driven approach where services have many components that in turn have ports. Other than that, these concepts are very similar to standard WSDL ones: Services have Components, Components have Ports and Ports exchange Messages of specific types. Finally, the services demonstrated are a Communication Service, that incorporates a Video Filter (used to adjust frame rate and size), an Audio Encoder/Decoder for speech, a Video Encoder and a Controller component that (de-)multiplexes text, speech and video and sends/receives the combined data stream.

Iacob (2008) develops a prototype that concerns Context (location and time) awareness. Services getUserLocation, getUserCell and getGeoLocation are shown on the prototype run.

Santofimia (2011) targets a Smart Room, but in its implementation focuses on intruder detection. Services considered in this approach are consequently intruder identification by fingerprints or biometric data.

3.1.3 Devices

Devices used in an AmI system greatly affect user experience in the system. They indicate the level of integration, diversity, mobility and thus user comfort in the system. Naturally, smartphones gather the most interest, being compact computational devices. Other than that, UPnP Multimedia is widely used along with Printers, RFID tags and more ambient devices e.g. touchscreens, lighting and sensors.

Davidyuk (2010) incorporates RFID tags and readers for user input and multimedia devices offering audio and video services. Ranganathan (2004) presents a system that is implemented on top of the GAIA pervasive infrastructure. GAIA incorporates a huge variety of devices. In categories, GAIA input devices include touch screens and microphones, authentication devices

like fingerprint sensors and smart card readers, handheld devices and wearable devices like smart watches and smart rings. Output devices include large plasma screens and video walls.

A system can employ a range of devices, not only for offering services and interfaces. A mobile device can also perform the composition itself and relieve the server from computational load. Specifically, a mobile device associated with its user can enter the environment and independently compose available services. In that case, the computational load of the composition procedure is transferred from the main system to individual devices. Hence, systems with many users and/or many requests can be salvaged from large loads of composition requests. Davidyuk (2010) uses a mobile phone not only for user input (RFID reader) but also for performing the composition. Other examples of systems explicitly performing composition on mobiles are **(**Yokohata 2006**),** (Davidyuk 2010**),** on PDA **(**Masuoka 2003**),** on a gateway/embedded PC (Lee 2007). Mostefaoui (2003) presents a system that runs the composing service on embedded PCs on shopping carts to aid disabled people. Finally, Messer (2006) has developed a Java digital TV prototype that can compose other UPnP services in range.

3.2 Problem Formulation

The second aspect of the comparative presentation concerns representational issues. More specifically it is about how the problem of service composition is modeled and presented in a formal way. The modeling of the problem involves the representation of the available services, the user-defined goals and finally the solution to the problem (i.e. the composite service). The values for these criteria for a number of systems are presented in Table 2.

3.2.1 Service Representation

Modeling Language of atomic services is one of the most indicative aspects of a system. Languages can be classified as the ones for syntactic descriptions and the ones for semantic descriptions. Most of recent AmI systems have settled in using the OWL-S language. However, the minimal upper OWL-S ontology leaves out domain specific objects, and also Groundings for other service technologies like UPnP. Consequently many authors have developed OWL-S extensions i.e. more varied and domain specific ontologies. Such examples are Amigo-S in (Thomson 2008), OWL-SC in (Qiu 2006), COCOA-L in (Mokhtar 2006, 2007). The rest of the systems employing OWL-S, separately define a suitable ontology and reference it in service descriptions. Ibrahim (2009a) defines a custom service model but not as an OWL-S extension.

System	Service Descriptions	Goal Description	Solution Representation
Paluska 2008	Script language	Simple Service	Script
Mokhtar 2007	OWL-S (ext.)	Workflow	OWL-S
Thomson 2008	OWL-S (ext. – AMIGO-S)	Workflow	BPEL
Ranganathan 2004	DAML+OIL	Simple Service	Lisp
Beauches 2008	YAWL	Workflow	YAWL
Messer 2006	XML + OWL/RDF	Simple Service	XML
Preuveneers 2005	OWL-S	Composite Service	Internal Representation
Iacob 2008	OWL-S	Composite Service	Ecore metamodel in Eclipse
Hesselman 2006	OWL-S	Workflow	Internal Representation
Vukovic 2007	OWL-S	Simple Service	BPEL4WS
Kalofonos 2006	WSDL, UPnP	Workflow	Scripts
Carey 2004	OWL-S	Composite FSM	OWL-S Process
Cottenier 2005	WSDL	Workflow	Executable Choreography Language
Vukovic 2004	WSDL	Context information + Goal	BPEL4WS
Bottaro 2007	Key-Value pairs	Workflow	Internal Representation
Qiu 2006	OWL-S (ext. OWL-SC)	Simple Service	DirectedAcyclic Graph (DAG)
Lee 2007	Key-Value pairs	Workflow	XML
Mingkhwan 2006	OWL-S	Workflow	XML
Qasem 2004	OWL-S	Workflow	Internal Representation
Kaefer 2006	UPnP	Workflow	Functional Task Description (FTD)
Pourezza 2006	OWL-S	Workflow	OWL-S Process
Maamar 2005	WDSL	Workflow	State Chart Diagrams
Bellur 2005	WDSL	Message Sequence Charts	BPEL4WS
Mostefaoui 2003	WDSL	Simple Services	Internal Representation
Sheshagiri 2004	OWL-S	Workflow	OWL-S Process
Mokhtar 2006	OWL-S (ext. COCOA-L)	Workflow	COCOA-L (OWL-S)
Chakraborty 2004	DAML-S	Workflow	Description-level Ser. Flow (DAML-S)
Masuoka 2003	OWL-S	Workflow	Internal Representation
Ni 2005	OWL-S	Predicate logic	Internal Representation
Vallée 2005	DAML-S	Predicate logic	Internal Representation
Paolucci 2009	BPEL	Simple Service	Plan Solution
Ibrahim 2009a	Custom	Composite Service	Custom
Robinson 2004	Script	Composite Service	Script
Santofimia 2011	DOBS Middleware	JADEX Goals	Internal Representation

Table 2. *Problem Formulation*

Davidyuk (2010) also uses such an extension, named AmIi, for service descriptions, matching and discovery. The AmIi service description model enables Profile and Grounding descriptions (like OWL-S). Each service can have a behaviour description, functional properties (inputs and outputs), non-functional properties especially useful for providing QoS and semantic annotations referenced by an ontology. This model provides mapping to legacy services and standard languages like WSDL or UPnP, enabling interoperability with the majority of existing

services. Other than that, older systems use the OWL-S predecessor, DAML-S, for semantic annotations. Even plain DAML+OIL has been used (which of course is not meant for services). WSDL is the only W3C recommendation in that field and still holds a standard place among developer preferences. After all, OWL-S descriptions have to map to WSDL services most of the time to gain actual functionality.

3.2.2 Goal Description

Systems that do not automatically sense user needs, have to receive some user goal description in order to provide a solution for it. Users mainly enter an abstract goal, or sentence (e.g. PlayVideo) that can also be regarded as a Simple Service. The system and the user have to share a goal vocabulary or use templates in order for the goal to be interpreted. In planning-based systems, the goal is usually converted in predicates, which are then used to find planning goal states. Ni (2005) skips this process as users directly enter predicates. Otherwise, users enter a detailed workflow of the desired functionality or a Composite Service whose atomic constructs have to be selected and instantiated.

3.2.3 Solution Representation

The final product is a composite service that is naturally consumable by the users. Usually composite services are formed in a universal representation format so that they can be re-discovered and executed in the future (this saves re-composition time). Such frameworks for representing compositions are YAWL[11], BPEL4WS (or just BPEL)[12], the OWL-S Process model and WS-CDL (Choreography Description Language)[13]. However some systems (Preuveneers 2005, Hesselman 2006, Bottaro 2007, Qasem 2004, Mostefaoui 2003, Masuoka 2003, Ni 2005, Santofimia 2011) do not explicitly export and save the resulting composite service in a universal reusable format but rather execute it on-the-fly. Equivalently, the service remains internally represented and consumed. It cannot be discovered or reused in the future.

3.3 Semantics

The next field of interest concerns the presence of semantics for the services. The findings of the review concerning this view are presented in Table 3. The criteria used for the comparison present whether the systems embody semantic annotations apart from syntactic ones, the language used for the representation of the ontology and which features the ontology includes. These

[11] YAWL - http://yawlfoundation.org/
[12] BPEL - http://www.oasis-open.org/committees/wsbpel/
[13] WS-CDL - http://www.w3.org/TR/ws-cdl-10/

features are organized in categories, such as quality of service, context, functionality and underlying hardware details.

System	Semantic Annotations	Ontology Language	QoS	Context	Functionality	Hardware
Paluska 2008	Syntactic	×	×	×	×	×
Sousa 2006	Syntactic	×	×	×	×	×
Mokhtar 2007	Semantic	OWL	✓	-	-	✓
Thomson 2008	Semantic	OWL	-	✓	✓	✓
Ranganathan 2004	Semantic	DAML+OIL	-	✓		✓
Beauches 2008	Semantic	Not Specified	-	-	✓	-
Messer 2006	Semantic	OWL/RDF	-	-	✓	✓
Preuveneers 2005	Semantic	OWL	-	✓	✓	✓
Bottaro 2007	Semantic	OWL	-	✓	-	-
Rouvoy 2009	Syntactic	×	×	×	×	×
Takemoto 2004	Syntactic	×	×	×	×	×
Iacob 2008	Semantic	OWL	-	✓	-	✓
Hesselman 2006	Semantic	OWL	-	✓	✓	-
Chantzara 2006	Syntactic	XML	-	✓	-	-
Davidyuk 2010	Semantic	XML	✓	-	✓	-
Vukovic 2007	Semantic	Not Specified	-	-	✓	-
Qiu 2006	Semantic	OWL	-	✓	-	-
Lee 2006	Semantic	Not Specified	-			
Bellur 2005	Semantic	OWL	✓	✓	✓	✓
Maffioletti 2006	Syntactic	OWL	-	✓	✓	✓
Sheshagiri 2004	Semantic	OWL	-	✓	-	-
Chakraborty 2005	Semantic	DAML	-	-	✓	-
Masuoka 2003	Semantic	OWL	-	-	✓	-
Ni 2005	Semantic	OWL	-	-	✓	-
Vallée 2005	Semantic	OWL	-	✓	✓	-
Ibrahim 2009a	Semantic	OWL	✓	-	✓	-
Santofimia 2011	Semantic	OWL	-	✓	✓	✓

***Table 3**. Semantics*

3.3.1 Semantic Annotations

Service descriptions can either have or do not have semantic annotations. Examples of systems with semantic annotations are the ones presented in (Davidyuk 2010), (Preuveneers 2005) and (Iacob 2008). What seems initially unorthodox is that two systems do not semantically annotate services but they do design ontologies. Namely, in (Chantzara 2006) and (Maffioletti 2006) the systems use XML service descriptions and consume their ontologies internally.

3.3.2 Ontology Contents

Some systems employ ontologies to define taxonomies and relationships between concepts in their domain of interest. After that, service descriptions are annotated and reference these ontologies. Ontologies usually disambiguate context parameters like location, time and environmental conditions. Some ontologies also strive to describe QoS properties of services e.g. latency, response time, CPU load etc. That would result in QoS optimized compositions useful for cases of limited resources (e.g. mobile use on-the-go).

A main goal for ontologies is describing the functionality of services, namely their inputs and outputs. Additionally, services can also have preconditions and effects that also need to be type-defined. After all, OWL-S is such an ontology that sets the basis for defining IOPEs. Finally, given the vast variety of AmI hardware, many ontologies also define concepts like Platforms, Devices, Resources and taxonomies between them.

This section presents the contents i.e. taxonomies and concepts included in ontologies for AmI systems. Ontology languages are also listed. They range from OWL to DAML and XML. This information corresponds to the last five columns of Table 3, namely Ontology Language, QoS, Context, Functionality and Hardware. The various concepts met in these ontologies are categorized in those that regard QoS parameters, Context, Functionality and Hardware. QoS related concepts can be Throughput, Latency etc. Context includes time, location and setting concepts. Functionality includes actions and IOPEs and Hardware concepts describe devices and resources.

Specifically, the ontology infrastructure of the GAIA system (Ranganathan 2004) is described in detail in (Ranganathan 2003). These ontologies mainly define either context information or entities in the environment. Context is represented in a predicate form that inherently suits the planning component (World states in Planning are sets of facts – predicates). E.g. the predicate Location (Chris, in, Room 2401) declares knowledge about a person's location. Other context-related predicates can be classified in: physical context (location and time), environmental context (weather, light and sound levels), informational context (stock quotes, sports scores), personal context (health, mood, schedule, activity), social context (group activity, social relationships, whom one is in a room with), application context (email, websites visited) and system context (network traffic, status of printers). Ontologies are used to type-check arguments of these predicates (e.g. Chris and Room 2401). On the other hand entity-related ontologies define taxonomies and relations between devices, services, applications and users.

GAIA (Ranganathan 2004) incorporates an ontology server that enables incremental addition of new ontologies. Classes and properties are then merged with the existing ones.

Preuveneers (2004) defines an ontology in OWL that along with a context management system, is able to adapt services based on context. Concepts defined in this ontology revolve around the four main concepts of User, Platform, Service and Environment. At a glance, Environment has location, time and environmental condition data. A user has mood, profile, role and tasks (to complete) that include activities and use services in turn. A platform provides hardware that relates to resources (power, memory, cpu, storage and network) and i/o devices, and software that provides services. Software can be an operating system, a virtual machine, a middleware or a rendering engine. The three main concepts are interconnected in many ways: a service requires a platform, a platform has an environment. The ontology for services is in fact OWL-S (that provides service profile, model and grounding) and is interlinked with the rest of the ontologies as tasks use services and software provides services.

Iacob et. al. (2008) also include the concept of context in their ontology. In fact, this ontology demonstrated, is only a fragment of an example domain ontology required by their prototype. Concepts include a User, associated with a GSMCell, a GeoLocation, his Home, his Office, and a Schedule. Finally Context comprises of GeoLocation, GSMCell and Schedule. So, indeed, context ontologies mainly define time and location terms.

Santofimia (2011) proposes a general simplistic semantic model for universal use across AmI applications. This only includes basic concepts that cannot be left out in AmI which namely are “Service”, “Device”, “Event”, “Action”, “Object” and “Context”. Furthermore, as a showcase, they map this model to an OWL ontology, adding more domain-dependent concepts and relationships suited for their intrusion-detection implementation, like “Announce” and “Hazard”.

All in all, true semantic interoperability across systems can only be achieved by using a common vocabulary. Apparently, most of the systems share some perspective on concepts: most of them define context, environmental parameters, hardware and functionality, but in many different ways. Reuse and convergence have yet to emerge. Only Sheshagiri (2004) and Bottaro (2007) reference existing vocabularies.

3.4 Composition Process

The fourth view of the comparative presentation is the composition of services itself and deals with several issues regarding how the composition is actually realized and what type of information it takes into account. More specifically, Table 4 presents an indicative selection of systems and analyzes them in terms of composition method, details concerning the specific

technique used and parameters that the composition takes into account, such as context awareness, quality of service, user involvement and user preferences.

System	Method	Technique	Context	QoS	User Involvement	User Prefs
Paluska 2008	Planning	HTN	✓	✓	×	×
Sousa 2006	Matching	Knapsack Optimization	×	✓	Manual	✓
Mokhtar 2007	Matching	Capabilities Matching	✓	✓	×	×
Thomson 2008	Matching	Capabilities Matching	✓	✓	×	×
Ranganathan 2004	Planning	BLACKBOX	✓	×	×	✓
Beauches 2008	Planning	GraphHTN	×	×	×	×
Messer 2006	Matching	Capabilities Matching + Ranking	✓	✓	Manual	✓
Preuveneers 2005	Matching	Capabilities Matching + Constraint solving	✓	✓	×	✓
Bottaro 2007	Matching	Capabilities Matching + Ranking	✓	×	×	✓
Iacob 2008	Matching	Capabilities Matching + Ranking	✓	✓	×	✓
Yokohata 2006	Matching	Capabilities Matching	✓	×	Manual	×
Nakazawa 2004	Matching	Capabilities Matching	×	×	×	×
Lagesse 2010	Matching	Capabilities Matching + Constraint solving	×	✓	×	✓
Davidyuk 2010	Planning	Genetic Algorithm	×	✓	Ranging	✓
Wisner 2007	Planning	State Space Search	×	×	Manual	×
Vukovic 2007	Planning	TLPLAN	✓	×	×	×
Qiu 2006	Planning	HTN	✓	✓	×	✓
Qasem 2004	Planning	HTN	✓	×	×	×
Maamar 2005	Agents	Context Based Negotiations	✓	✓	×	×
Sheshagiri 2004	Planning	STRIPS	✓	×	×	✓
Mabrouk 2009	Matching	Capabilities Matching + Ranking	×	✓	×	×
Masuoka 2003	Matching	Manual	✓	×	Manual	✓
Ni 2005	Planning	State Space Search	✓	×	Choose Solution	×
Vallée 2005	Agents	Semantic matching	✓	×	Choose Solution	✓
Paolucci 2009	Planning	BDD		×	×	×
Salomie 2008	Fluent Calculus	Fluent Calculus	×	×	×	×
Ibrahim 2009a	Matching	Pair-wise Capabilities Matching	×	✓	×	×
Robinson 2004	Matching	Capabilities Matching	✓	×	×	×
Santofimia 2011	Planning	HTN	✓	×	×	×

***Table 4**. Composition Process*

3.4.1 Composition Methods and Techniques

Some methodologies could be considered as framework dependent and hence restrictive. In other words, some techniques require specific representation form like predicates, and a

transformation is required beforehand. Other than that, methodologies can be applied to any domain and range from planning to service matching/selection and slight variations.

Ranganathan (2004) employs blackbox STRIPS planning that also supports retryable actions. This system's implementation in general favors the planning methodology. Common planning aspects are widely used, like the world representation in predicate form. However, this representation is a requirement for the system's planning component and thus the component could be considered as restrictive (Davidyuk 2008).

The planning subsystem in (Davidyuk 2010), receives a series of abstract user subgoals that have to be grounded – mapped to actual implementations/devices. The technique involved is based on evolutionary and genetic computing optimization and also takes into account multiple user criteria (nearest, fastest or cheapest solution) and user preferences (fidelity and QoS).

In general, most of the service matching systems rely on service discovery subsystem to match the required capabilities. If applicable, matching required QoS parameters or preferences is also a requirement. A utility function is also used sometimes to rank service candidates. E.g. Iacob (2008) employs a matching approach where goal services are iteratively decomposed until a matching solution is found for each component. Each match is evaluated against a utility measure of semantic similarity.

Multi-Agent systems as autonomous entities can enhance an AmI system in a number of ways. Popular approaches are exchanging data about context or playing an active role in composition by performing matching or planning. They can also simply autonomously call the composition subsystem (relieving the user of having to do so).

(Vallée 2005) employs a multi-agent-system approach that combines context management and service matching. There are three types of Agents: Assistant, Composition and Service Agents. Assistant Agents receive the goal task either explicitly from the user or infer it from context-based rules. Composition Agents put together possible solutions comprised of a set of requirements on services and their relationships. Service Agents fulfill these requirements by selecting services based on semantic matching service descriptions and evaluate them, based on current context. Agents of each type are able to negotiate with one another. The user can finally take decision on the resulting alternatives. Maamar 2005 also employs autonomous agents that collectively perform service composition. The different types of agents are associated with composite services, atomic services and service instances and negotiate based on their knowledge of the current service state.

BDI Agents in (Santofimia 2011) and (Santofimia 2008) interact with each other to exchange information and finally call plans to fulfill their goals. The planning infrastructure is

HTN-based. The MAS which is implemented in JADEX, discovers Services and refers to the knowledge base and semantic model over the ZeroC ICE-based middleware.

3.4.2 Context-awareness

Context-awareness refers to the ability of a system or method to perceive and take into account the current environment or user state. As a result, outputs are case sensitive: the same inputs can bring about different results according to the user's situation. Apparently, this property is most critical for AmI systems that pursue high-level of automation, smart sensing and reacting to user context.

The most popular approach for context awareness is designing an ontology, where concepts regarding context and system entities can be defined and inter-related. E.g. a location or temperature can be assigned to a user or a room respectively. Systems that follow this methodology include (Ranganathan 2004), (Preuveneers 2005). Context can also be the product of reasoning (e.g. absolute coordinates can point out a certain location or room) which is the case in (Preuveneers 2005)**.** Context is also part of an ontology in GAIA and has been described in the corresponding section. An approach to consider context in composition is presented in (Vallée 2005) which evaluates each selected service based on current context.

Except the view of context as a set of statements about a person a place or an object (Abowd 1999), another is regarding context as a whole world representation including static and dynamic facts. This approach is followed by Santofimia 2011, where a knowledge base and a semantic model are used to model a world state.

3.4.3 Quality of Service – QoS

Quality of Service considers added value parameters like service latency, response time or costs in general. Systems that take these parameters into account, offer optimum solutions. A typical approach to consider QoS is using an ontology that defines QoS concepts. Then, service QoS parameters can be registered and reviewed at discovery-time as service meta-data. Davidyuk (2010) indeed includes QoS metadata – non-functional properties of services in an upper ontology. Thus each service has known non-functional or QoS properties, that serve as selection criteria.

3.4.4 User Preferences

Some systems also take decisions taking user preferences into account. This parameter differs from context as preferences are not case sensitive, but rather characterize a specific user. E.g. a user can generally prefer mobile services. Another user could prefer black and white printing to color. Preferences could also be considered as profiles that can be interchanged.

Preuveneers (2005) considers user preferences and namely High, Medium or Low Quality in audio or video, only if the current context allows it and resources are available.

3.4.5 User involvement in Composition

AmI systems aim automatic service composition i.e. minimizing user involvement. Traditional non-AmI applications require full user involvement. However, sometimes users do get partially involved. As the transition between fully autonomous systems and fully controlled systems is ongoing, many systems still require partial user involvement either to compensate for the lack of inference of user goals or to simply supervise the process and manually select the best composite solution. Indeed, a usual form of user participation is presenting the user with a variety of solutions, along with their rankings (if a utility function is available), so that he is able to choose based on his intent. In (Vallée 2005) users take decisions on the final alternative solutions presented. After all, it is argued whether AmI systems can make accurate predictions of the user intent.

Davidyuk (2010) focuses on the user-centric proportion of AmI applications and lets users manually craft applications (i.e. composite services) arranging abstractions of service implementations. The system provides four interchangeable interface levels of user involvement that range from fully manual to fully automatic. As user involvement rises, system autonomy decreases. In manual mode, the user directly selects service instances/devices while the composition component is disabled. Moving on, while on the semi-manual mode, the system's composition component can complete the application by assigning the services the user has left out. In the mixed-initiative mode, the system presents a range of composite services ordered by user defined criteria. Users then are able to browse and validate which service instances will be used each time, prior to execution. Finally, in the fully autonomic mode the system autonomously executes composite services.

On the other hand, high level of user involvement can be met in (Sousa 2006), (Messer 2006), (Yokohata 2006) and (Wisner 2007). E.g. in (Messer 2006) users are aided to enter the goal services (pseudo sentences in the form of Play Video file x). If there is no user involvement, the system's function is totally transparent to the user. In other words, user involvement is the opposite of system transparency. Transparency is considered as a criterion on the survey of Ibrahim (2009b).

3.5 End-Product Features

The final view of the reviewed systems concerns several features of the system performing service composition as a whole. More specifically, we are interested in security issues and how they are resolved in the systems, the type of interfaces offered to the user, whether the systems have been evaluated and finally if they are implemented as stand-alone or multi-agent systems. Table 5 presents an indicative number of such systems and analyzes their performance in the aforementioned features.

3.5.1 Security

New paradigms like the Service Oriented Architecture, Task Oriented Computing etc. have yet time to mature and standardize. There are many modeling languages, many frameworks and many methodologies. Each one has its own aspects, and security issues are not always inherently addressed. Systems need to proactively take measures to prevent malicious use that would result in loss of data, privacy breaches and malfunctions.

An approach to security by using ontologies is introduced by GAIA (Ranganathan 2004). GAIA incorporates access control policies and ontologies that define user roles and thus privileges. However, the Semantic Web technologies used to define these ontologies have no inherent authorization mechanisms. These mechanisms have to be engineered. Maamar (2005) has indeed developed an Access Control mechanism for preventing malicious of services and thus misusing resources. Khosrowshahi (2009) considers consumer privacy by adopting a model that always maintains data locally.

3.5.2 User Interface

An AmI system supposedly has a physical, intuitive interface so that users do not have to enter its monolithic environment (e.g. sitting in front of the desktop computer). This review focuses on the composition component, but also at the system as a whole. Naturally, the UI greatly signifies an AmI system. User interfaces can be bidirectional: they allow input and/or output to and from the system. Input interfaces especially, solely depend on the advancement of Human-Computer interfaces. There are visions of virtual reality, motion input (i.e. gestures) or even input by pads attached to the skull.

Ranganathan (2004) uses a windows forms GUI application to enable the user input the goal task. Users check desired complex tasks and choose desired parameters. They are also able to view the planning process' solution i.e. grounded actions as the GAIA infrastructure executes the service.

System	Security	UI	Evaluation	M.A.S.
Paluska 2008	×	×	×	×
Sousa 2006	×	GUI	Experimental	×
Mokhtar 2007	Security Parameters	×	Experimental	×
Ranganathan 2004	Access Control	GUI, Feedback	×	×
Messer 2006	×	TV GUI Compose Tool	Experimental	×
Bottaro 2007	×	PDA GUI	Experimental, Comparison	×
Yokohata 2006	×	GUI	Experimental	×
Lagesse 2010	Trust by Learning	×	Experimental, Comparison	×
Davidyuk 2010	×	Mobile GUI + Physical UI (cards)	User	×
Wisner 2007	×	GUI Compose Tool	×	×
Cottenier 2005	×	×	×	Migrating Agents
Vukovic 2004	×	GUI	×	×
Vukovic 2007	×	×	Experimental	×
Lee 2007	×	PDA GUI	×	×
Khosrowshahi 2009	Data maintained locally	PDA GUI	×	×
Lee 2006	×	×	Simulation	Service Agents
Maamar 2005	Access Control	GUI Composite Editor	×	Service Agents
Mostefaoui 2003	×	×	×	Context Agents
Sheshagiri 2004	Privacy Parameter	×	×	Task Specific Agents
Mabrouk 2009	Security and Privacy Parameters	×	Experimental	×
Chakraborty 2005	×	×	Simulation, Comparison	×
Vallée 2005	×	×	×	Composition Agents
Paolucci 2009	×	×	Experimental	×
Santofimia 2011	×	×	Experimental	Manager, Preceptor Agents

***Table 5**. End-Product Features*

Davidyuk (2010) demonstrates an innovative and intuitive mixture of a physical and a computer graphical interface. Specifically, it employs RFID tags and a mobile phone that not only reads them but also presents a GUI for user input. The tags are placed on custom-made cards associated with atomic services or data e.g. video/audio files. Unreachable elements like projectors and loudspeakers have their cards placed on a reachable control panel in the room. Scanning a series of cards forms a workflow of the requested composite service. However, as the level of user involvement is configurable (as mentioned before), the user has to enter or view some information

(except in the fully automatic mode). A mobile phone GUI enables them to view selected service instances prior to execution for validation purposes. In the mixed-initiative mode they can choose between suggested compositions.

Output interfaces currently remain conventional. Users solely came aware of the effects their requests have e.g. see their documents printed, audio or video is played on target devices. This methodology certainly lacks feedback. Feedback is valued not only in case of faults, where the user simply needs to know what went wrong but also for tracking down the reason why the composite service contains the specific atomic services (e.g. could be due to QoS parameters or unavailability of alternatives). Feedback also adds to the system's feel of responsiveness: users remain uncertain whether the system functions or not until they see the effects. If no solution is found, users indefinitely remain uncertain.

Traditional output on displays is certainly an acceptable alternative as an output interface. Some systems indeed provide graphical tools (GUIs) to present users with results. The composite service solution,, alternative ones or general information can be shown.

In Vallée 2005 an interface is implied as users not only enter goals but also make the final decision between alternatives.

Finally, a GUI can be bidirectional, which complies with the traditional desktop computer paradigm. Users enter data in an application and are presented with the results. More flexible systems include mobile applications for input/output which enhances the system with a more pervasive feel (Bottaro 2007, Davidyuk 2010, Lee 2007, Khosrowshahi 2009)

3.5.3 Evaluation

Empirical results are a valuable piece of information, when reviewing systems. Every kind of such results shows actual performance and therefore measures usability and effectiveness. Evaluations can take many forms. Some systems showcase composition runtimes in different settings and present extensive results in charts. For example, (Ibrahim 2009a) measures composition times while increasing the number of available atomic services each time. This evaluation is carried out once for plain services (syntactic descriptions) and once with semantic annotations. Naturally, composition times in the second case range high above the ones in the first case. Similarly Vukovic (2007) presents scalability performance. Mabrouk (2009) presents execution times for various workloads, number of services and QoS parameters. Chakraborty (2005) confirmed over simulation that their setting outperforms centralized composition.

Davidyuk (2010) carried out a user test to evaluate its RFID-card interface. A single pair of users experimented with an initial set of cards for 1.5 hours, designing six composite services

(applications). Interesting conclusions out of the experiment suggest that the smart card designs (i.e. icons) should be self-explanatory and intuitive and there should be a motivating mechanism for the system.

3.5.4 Multi-Agent Systems

Multi-Agent systems are an active area of research that many systems can benefit from. All autonomous systems can be regarded as software agents. This criterion specifically presents systems that employ a multi-agent system where agents negotiate, reason and act to aid composition in any way.

Vallée 2005 employs a MAS of three different types of Agents that negotiate, to aid the composition process. Hesselman (2006) includes Context-Agents in his approach. However these agents are not considered as MAS, as these are simply software agents that provide data (i.e. context information) and do not negotiate or interact.

Being autonomous entities, BDI Agents can perceive the world and act towards accomplishing their goals. That eliminates the need for user involvement as in (Santofimia 2011), where Agents evaluate perceptions of the world and autonomously take actions. In this particular implementation, the Agents perceive intruder presence, and authenticate him using different techniques.

Acknowledgments

This project is funded by Operational Program Education and Lifelong Learning, OPS 200056 (International Hellenic University, Thessaloniki, Greece).

Conclusion

This article presented a review of systems in Ambient Intelligence environments, mainly focusing on service composition aspects. AmI systems rely more and more on service composition to provide intelligent automations to their users. As service orientation becomes a standard in AmI, methodologies to develop such an infrastructure have surfaced and slightly converged. AmI systems also focus on relevant but different domains of smart spaces and employ various sets of devices. Spread of UPnP multimedia devices has led to wide use of AmI home multimedia systems. Smart Offices, Meeting rooms, Teleconference and Health have also been domains of interest. Meanwhile, service description languages have settled on the standardized WSDL format. However, as WSDL can provide syntactic interoperability only, the use of Ontologies has

been issued upon. Many languages have emerged to semantically annotate Web Service descriptions, referencing OWL ontologies. Although it is not a recommendation yet, OWL-S is the dominant language in recent AmI systems. However, not only technology has to converge in standard languages but also, a universal agreed-upon AmI vocabulary of concepts has to be met.

Context awareness is a key element in AmI. Ontologies for context resources appear to be the dominant approach towards it. Other than that, concepts include QoS parameters, type-definition of inputs, outputs, preconditions and effects of services, Platforms, Devices etc.

At the next stage, services are discovered and composed. The two main methods of composition, planning and matching have been presented and described. Planning uses existing progress and out-of-the-box algorithms for the task while matching is brute-force in nature. Semantic annotations enhance discovery and can benefit both planning and matching. Finally, a couple of other methods include the use of MAS, where agents cooperate to form a goal service. The methods can and need to consider some parameters, namely QoS, context-awareness and user preferences. Also sometimes the user is able to fully or partially control composition e.g. by selecting one of the proposed solutions.

Finally, AmI systems present different additional properties. Some of them consider security issues by applying access control or requiring trust-related parameters in services. Security is one of the key issues to be addressed in the future. A few systems also offer mobile, desktop or even physical UIs. Test runs, experimental results or comparisons of the systems can be found as means for evaluation. Another interesting topic of AI, Multi-Agent Systems, has also been employed in a few AmI systems, aiding in context or data handling or even composition itself.

References

Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P., 1999, Towards a better understanding of context and context-awareness. In: HUC,pp. 304–307.

Beauche S.,Poizat P., Automated Service Composition with Adaptive Planning. ICSOC 2008: 530-537

Bouguettaya, A., Krueger, I. & Margaria, T. (Eds.), Proceedings of the 6th International Conference on Service-Oriented Computing (pp. 530-537). Berlin, Heidelberg: Springer-Verlag.

Bellur U., Narendra N. C. Towards service orientation in pervasive computing systems. International Conference on Information Technology: Coding and Computing, 2:289-295, 2005.

Bertoli P., Kazhamiakin R., Paolucci M., Pistore M., Raik H., Wagner M.: Continuous Orchestration of Web Services via Planning. ICAPS 2009

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

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

Chakraborty D. Service discovery and composition in pervasive environments, THESIS, 2004.

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

Chantzara M., Anagnostou M. & Sykas E. (2006). Designing a Quality-Aware Discovery Mechanism for Acquiring Context Information. Proceedings of the 20th International Conference on Advanced Information Networking and Applications, 1(6), AINA'06. Washington DC: IEEE Computer Society.

Cottenier T. and Elrad T.: 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.

Davidyuk O., Georgantas N., Issarny V., Riekki J. Dans: MEDUSA: Middleware for End-User Composition of Ubiquitous Applications, Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives, IGI Global (Ed.) (2010)

Hesselman C., Tokmakoff A., Pawar P. & Iacobs S., (2006). Discovery and Composition of Services for Context-Aware Systems. Proceedings of the 1st IEEE European Conference on Smart Sensing and Context (pp. 67-81). Berlin: SpringerVerlag.

Iacob S. M., Almeida J. P. A., Iacob M.E.: Optimized dynamic semantic composition of services. SAC 2008: 2286-2292

Ibrahim N., Le Mouël F., Frénot S., MySIM: A Spontaneous Service Integration Middleware for Pervasive Environments, ACM International Conference on Pervasive Services (ICPS), july 2009, London, England.

Ibrahim N., Le Mouël F. (2009) A Survey on Service Composition Middleware in Pervasive Environments, 1-12. In International Journal of Computer Science Issues (IJCSI)

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

Kalofonos D.N. and Reynolds F.D. , "Task-Driven End-User Programming of Smart Spaces Using Mobile Devices". Published in Nokia Research Center Technical Report (NRC-TR-2006-001)

Khosrowshahi B. S., Graham P.: Component placement and location for a dynamic software composition system. C3S2E 2009:127-130

Lagesse B., Kumar M., Wright M.: ReSCo: A middleware component for Reliable Service Composition in pervasive systems. PerCom Workshops 2010: 486-491

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

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

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

Mabrouk N. B., Beauche S., Kuznetsova E., Georgantas N., Issarny V.: QoS-Aware Service Composition in Dynamic Service Oriented Environments. Middleware 2009: 123-142

Maffioletti S.: UBIDEV A Homogeneous Service Framework for Pervasive Computing Environments THESIS, 2006

Masuoka R., Parsia B., Labrou Y.: Task Computing - The Semantic Web Meets Pervasive Computing. International Semantic Web Conference 2003: 866-881

Messer A., Kunjithapatham A., Sheshagiri M., Song H., Kumar P., Nguyen P. & Yi K.H. (2006). InterPlay: A Middleware for Seamless Device Integration and Task Orchestration in a Networked Home. Proceedings of the Annual IEEE International Conference on Pervasive Computing PerCom'06 (pp. 296-307), Washington DC: IEEE Computer Society.

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

Mishra R. B. and Kumar S.: Semantic web reasoners and languages, Artificial Intelligence Review, 2011, Volume 35, Number 4, Pages 339-368

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

Mokhtar S. B., (2007). Semantic Middleware for Service-Oriented Pervasive Computing. Doctoral dissertation, University of Paris 6, Paris, France.

Mostefaoui S. K., Tafat-Bouzid A., and Hirsbrunner B. 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.

Nakazawa J., Yura J. & Tokuda H. (2004). Galaxy: a Service Shaping Approach for Addressing the Hidden Service Problem. Proceedings of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (pp. 35-39).

Ni Q., Sloman M. An Ontology-enabled Service Oriented Architecture for Pervasive Computing. In Proceedings of ITCC (2)'2005. pp.797-798

Paluska J. M., Pham H., Saif U., Chau G., Terman C., Ward S.: Structured decomposition of adaptive applications. Pervasive and Mobile Computing 4(6): 791-806 (2008)

Park J.H. and Kang J. H.: Intelligent service processing in common USN middleware Artificial Intelligence Review, 2011, Volume 35, Number 1, Pages 37-51

Pourreza H. and Graham P. 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.

Preuveneers D., Van den Bergh J., Wagelaar D., Georges A., Rigole P., Clerckx T., Berbers Y., Coninx K., Jonckers V., De Bosschere K.: Towards an Extensible Context Ontology for Ambient Intelligence. EUSAI 2004: 148-159

Preuveneers D. & Berbers Y. (2005). Automated Context-Driven Composition of Pervasive Services to Alleviate Non-Functional Concerns. International Journal of Computing and Information Sciences, 3(2), 19-28.

Qasem A., Heflin J., and Mu~noz-Avila H. Efficient source discovery and service composition for ubiquitous computing environments. 2004.

Qiu L., Shi Z., and Lin F. 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.

Ranganathan A., McGrath R. E., Campbell R.H., Mickunas M.D. Ontologies in a Pervasive Computing Environment. In Workshop on Ontologies and Distributed Systems (part of the 18'th International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico, Aug 9 2003

Ranganathan A. & Campbell R. H. (2004). Pervasive Autonomic Computing Based on Planning. Proceedings of the IEEE International Conference on Autonomic Computing ICAC'04 (pp. 80-87), Washington, DC: IEEE Computer Society.

Robinson J., Wakeman I., Owen T.: Scooby: middleware for service composition in pervasive computing. Middleware for Pervasive and Ad-hoc Computing 2004: 161-166

Rouvoy R., Barone P., Ding Y., Eliassen F., Hallsteinsen S., Lorenzo J., Mamelli A. & Scholz U. (2009). MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In Cheng, B. H. et al (Eds.) Software Engineering For Self-Adaptive Systems (pp. 164-182), Lecture Notes In Computer Science, Vol. 5525. Berlin, Heidelberg: Springer-Verlag.

Salomie I., Chifu V. R., Harsa I.: Towards automated web service composition with fluent calculus and domain ontologies. iiWAS 2008: 201-207

Santofimia M. J., Moya F., Villanueva F. J., Villa D. and Lopez J. C.: An agent-based approach towards automatic service composition in ambient intelligence Artificial Intelligence Review, 2008, Volume 29, Numbers 3-4, Pages 265-276

Santofimia M. J., Fahlman S. E., del Toro X., Moya F. and Lopez H. J.: A semantic model for actions and events in ambient intelligence, Engineering Applications of Artificial Intelligence (June 2011)

Sheshagiri M., Sadeh N. M. and Gandon F. 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.

Sousa J. P., Poladian V., Garlan D., Schmerl B. and Shaw M.: Task-Based Adaptation for Ubiquitous Computing. In IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems, Vol. 36(3), May 2006

Takemoto M., Oh-ishi T., Iwata T., Yamato Y., Tanaka Y., Shinno K., Tokumoto S. & Shimamoto N. (2004). A Service-Composition and Service-Emergence Framework for Ubiquitous-Computing Environments. Proceedings of International Symposium on Applications and the Internet, SAINT'04-W (pp. 313-318). Washington DC: IEEE Computer Society.

Thomson G., Bianco S., Mokhtar S. B., Georgantas N. and Issarny V.: Amigo Aware Services, Communications in Computer and Information Science, 1, Volume 11, Constructing Ambient Intelligence, Part 7, Pages 385-390

Urbieta A., Barrutieta G., Parra J., Uribarren A.: A survey of dynamic service composition approaches for ambient systems, Proceedings of the 2008 Ambi-Sys workshop on Software Organisation and MonIToring of Ambient Systems

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

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

Vukovic M., Kotsovinos E., Robinson P.: An architecture for rapid, on-demand service composition. Service Oriented Computing and Applications 1(4): 197-212 (2007)

Weiser M., The computer for the 21st century, ACM SIGMOBILE Mobile Computing and Communications Review, v.3 n.3, p.3-11, July 1999 [doi>10.1145/329124.329126]

Wisner P., Kalofonos D.N. (2007). A Framework for End-User Programming of Smart Homes Using Mobile Devices. Proceedings of the 4th IEEE Consumer Communications and Networking Conference CCNC'07 (pp. 716-721), Washington DC: IEEE Computer Society.

Yokohata Y., Yamato Y., Takemoto M., Sunaga H.: Service Composition Architecture for Programmability and Flexibility in Ubiquitous Communication Networks. SAINT Workshops 2006: 142-145