Open access • Book Chapter • DOI:10.4018/978-1-61692-857-5.CH011

MEDUSA: Middleware for End-User Composition of Ubiquitous Applications

Oleg Davidyuk, Nikolaos Georgantas, Valérie Issarny, Jukka Riekki

Institutions: University of Oulu

Published on: 01 Jan 2011

Topics: Ubiquitous robot and Middleware (distributed applications)

MEDUSA: Middleware for End-User Composition of Ubiquitous Applications

Oleg Davidyuk, Nikolaos Georgantas, Valérie Issarny, Jukka Riekki

Davidyuk, O., Georgantas, N., Issarny, V. & Riekki, J. (2009). MEDUSA: Middleware for End-User Composition of Ubiquitous Applications. In: Mastrogiovanni, F. & Chong, N.Y. (Eds.), Handbook of Research on Ambient Intelligence and Smart Environments: Trends and Perspectives. IGI Global, to appear.

MEDUSA: Middleware for End-User Composition of Ubiquitous Applications

Oleg Davidyuk[1,2], Nikolaos Georgantas[1], Valérie Issarny[1] and Jukka Riekki[2]

[1]ARLES Research Team, INRIA Paris-Rocquencourt,
Domain de Voluceau, Le Chesnay, 78153, France
{Firstname.Lastname}@inria.fr

[2]Dept. of Electrical and Information Engineering and Infotech Oulu,
University of Oulu, P.O. Box 4500, 90014, Finland
{Firstname.Lastname}@ee.oulu.fi

Abstract. Activity-oriented computing (AOC) is a paradigm promoting the run-time realization of applications by composing ubiquitous services in the user's surroundings according to abstract specifications of user activities. The paradigm is particularly well-suited for enacting ubiquitous applications. However, there is still a need for end-users to create and control the ubiquitous applications because they are better aware of their own needs and activities than any existing context-aware system could ever be. In this chapter, we give an overview of state of the art ubiquitous application composition, present the architecture of the MEDUSA middleware and demonstrate its realization, which is based on existing open-source solutions. On the basis of our discussion on state of the art ubiquitous application composition, we argue that current implementations of the AOC paradigm are lacking in end-user support. Our solution, the MEDUSA middleware, allows end-users to explicitly compose applications from networked services, while building on an activity-oriented computing infrastructure to dynamically realize the composition.

1. Introduction

Ubiquitous computing as envisioned by Mark Weiser (1991) emphasizes the interaction of users with smart spaces composed of humans and multiple networked computing devices. This vision has been empowered by continuing progress in various relevant fields of research, such as wireless communication, mobile computing, mobile sensing and human-computer interaction (HCI). Firstly, wireless communication standards (e.g., WiFi and Bluetooth) have enabled users to access smart spaces using their mobile terminals and portable devices. Secondly, efficient low-powered CPUs make mobile terminals capable of running software platforms which support advanced interoperability between different devices. In addition, mobile sensing technologies, like RFID and GPS, make ubiquitous applications context-aware and they also enable alternative HCI interfaces based on, e.g., physical contact (Nokia, 2009). However, despite all these technical developments, the user-centric nature of ubiquitous computing should not be neglected.

An important new trend in ubiquitous computing research is activity-oriented computing (AOC) (Masuoka, Parsia & Labrou, 2003; Ben Mokhtar, Georgantas & Issarny, 2007; Sousa, Schmerl, Steenkiste & Garlan, 2008a). This paradigm adopts a user-centric perspective and assumes that smart spaces are aware of user needs and activities and reactively, or even proactively, satisfy user demands by composing and deploying the appropriate services and resources. User activities consist of the users' everyday tasks and they can be abstractly described in terms of (i) the situation (context) where the tasks take place, (ii) the system functionalities required for accomplishing the activities, and (iii) user preferences relating to

QoS, privacy, security or other non-functional requirements. The system then dynamically realizes the activities by composing the applications according to the activity descriptions.

Under the AOC paradigm, the research community concentrates on introducing autonomic systems, which are self-manageable, self-configurable and adaptive, and therefore do not require user involvement. However, as pointed out by Hardian, Indulska & Henricksen (2008) and confirmed through user evaluation and usability tests (Davidyuk, Sánches, Duran & Riekki, 2008a; Vastenburg, Keyson & de Ridder, 2007), involving users in application control is essential for ensuring the user acceptance of autonomous products, especially in home or office automation domains. The AOC approach also has another drawback which limits its applicability. Application composition in AOC systems is performed by matching user activity descriptions with the networked services and resources discovered in the vicinity of the user according to the chosen criteria. Although some prototypes allow end-users to adjust the criteria at run-time (Davidyuk, Selek, Duran & Riekki, 2008b; Sousa et al, 2008a), the composition is limited to predefined activity descriptions (i.e. templates), which are complex structures and are not supposed to be understood or modified by end-users. Thus, users are neither able to customize the existing applications nor able to create their own smart space applications.

To address the aforementioned limitations of AOC systems, we present MEDUSA, a middleware solution which enables user-driven application composition in smart spaces. MEDUSA enables end-users to create simple applications from available ubiquitous resources discovered in the vicinity of the users. The applications are composed on the users' handheld devices. MEDUSA employs an end-user approach to programming applying it to the Ambient Intelligence (AmI) and ubiquitous computing domains (Kawsar, Nakajima & Fujinami, 2008; Mavrommati & Darzentas, 2007; Mavrommati & Kameas, 2003; Sousa, Schmerl, Steenkiste & Garlan, 2008b). The design of the MEDUSA middleware builds on our years of experience in developing middleware interoperability solutions, e.g., Amigo (2009) and PLASTIC (2009) as well as RFID-based user interfaces for smart spaces (Davidyuk et al, 2008a; Davidyuk, Sanches, Duran & Riekki, 2009; Riekki, 2007).

This chapter describes the design rationale for the MEDUSA middleware architecture and it organized as follows. Section 2 surveys the background of ubiquitous application composition focusing on developments in AOC research, thus providing an overview of the key aspects relating to the specification of ubiquitous applications, the middleware support of dynamic application realization and end-user involvement. Section 3 then introduces MEDUSA, which empowers end-users to participate in application development and further leverages existing open-source middleware solutions for the actual realization of application composition. Finally, Section 4 concludes with a summary of the contribution of this study and our plans for future research.

2. Related Work

Before presenting related work, we first introduce the notion of application composition, which here refers to the process of building applications by assembling services offered by computing devices embedded in the environment.

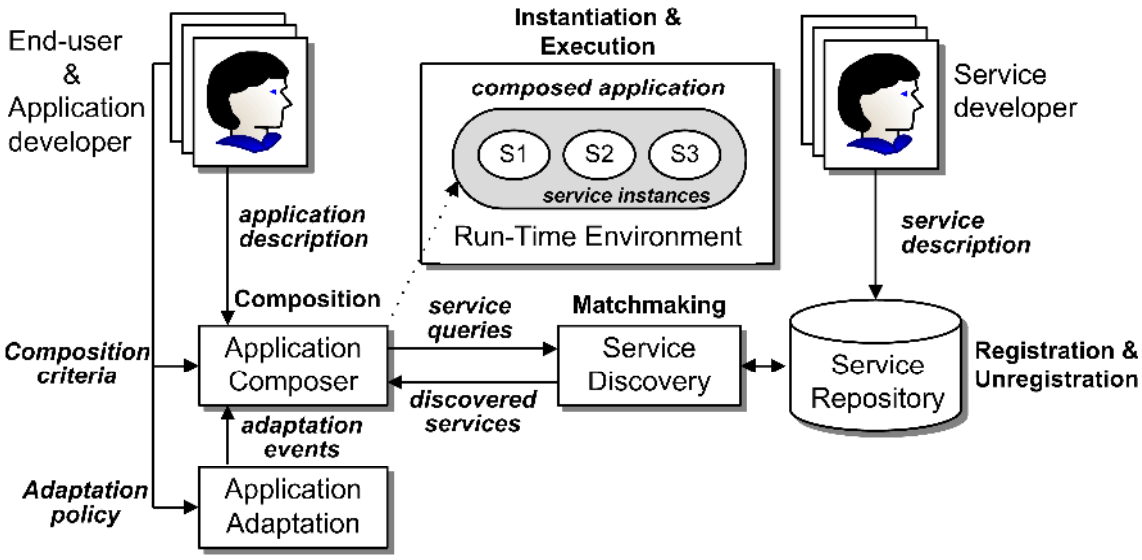

Fig. 1. Generic application composition process.

As shown in Figure 1, the application composition process generally involves two separate actors, the end-user and the service provider. The role of the service provider is to develop and publish services which provide some functionalities through a well-defined interface. The end-user's role is to create applications and utilize (i.e., interact with) applications. Some authors further separate the roles of application users and application developers, thus assuming that applications are developed and used by different persons (Masuoka et al, 2003; Sousa, Poladian, Garlan, Schmerl & Shaw, 2006; Sousa et al, 2008a). Others do not consider the end-user role at all and suggest a view in which only application developers are involved in the application composition process (Paluska, Pham, Saif, Chau, Terman & Ward, 2008).

The application composition process relies on descriptions of applications and services. An application description specifies the abstract services the application is composed of and the relationships between them (i.e. control and data flows). The application composer is responsible for decision-making in the application composition process. The application composer uses application descriptions and possible composition criteria (preferences, fidelity constraints, costs and so on) provided by the end-user as inputs and chooses available services which satisfy the given criteria. The selection of services is supported by a service discovery protocol, which is responsible for the matchmaking functionality (i.e., for matching service discovery requests against the service descriptions stored in the service registry). Since the discovered set of services may potentially contain redundant instances, the application composer optimizes (i.e., further reduces) the service set and produces an application configuration (i.e., application composition plan), which satisfies the criteria given earlier. Different application configurations may be produced depending on the criteria and the situation in the environment. After this, the application is instantiated and executed by the runtime environment. During execution, the application can be adapted (i.e. recomposed) according to user defined adaptation policies. These policies are formal rules which trigger predefined actions when the application or user context changes.

To date, several studies supporting ubiquitous application composition have been presented. They focus on service provisioning issues (Chantzara, Anagnostou & Sykas, 2006; Nakano, Takemoto, Yamato & Sunaga, 2006; Takemoto, Oh-ishi, Iwata, Yamato, Tanaka, Shinno, Tokumoto & Shimamoto, 2004), context-aware adaptation (Preuveneers & Berbers, 2005a; Hesselman, Tokmakoff, Pawar & Iacob, 2006; Jianqi & Lalanda, 2008; Bottaro, Gerodolle & Lalanda, 2007; Bottaro, Bourcier, Escofier & Lalanda, 2007; Handte Herrmann, Schiele & Becker, 2007; Rouvoy, Eliassen, Floch, Hallsteinsen & Stav, 2008; Rouvoy, Barone, Ding, Eliassen, Hallsteinsen, Lorenzo, Mamelli & Scholz, 2009), service validation and trust

(Bertolino, De Angelis, Frantzen & Polini, 2008; Bertolino, De Angelis & Polini, 2009; Buford, Kumar & Perkins, 2006), service communication path optimization (Kalasapur, Kumar & Shirazi, 2005), automatic application code generation (Nakazawa, Yura & Tokuda, 2004) and distributed user interface deployment (Rigole, Vandervelpen, Luyten, Berbers, Vandewoude & Coninx, 2005). Several design styles for developing adaptive ubiquitous applications through composition have also been suggested (Paluska et al, 2008; Saif, Pham, Paluska, Waterman, Terman & Ward, 2003; Sousa et al, 2008b).

Some of the most promising research deals with the activity-oriented computing approach (Masuoka et al, 2003; Ben Mokhtar et al 2007; Sousa et al, 2008a). These solutions focus on decoupling the end-user tasks from their system-level realization, i.e., they let the end-users concentrate on the tasks or activities they need, rather than ask them to specify how they want these tasks to be performed by the system. User tasks are simple everyday activities (e.g., in the home or office environment), which can be achieved by compositing an application. AOC solutions take a user-centric approach to application composition by suggesting that users explicitly provide descriptions to the system via dedicated task composition and control interfaces (Davidyuk et al, 2008a; Messer, Kunjithapatham, Sheshagiri, Song, Kumar, Nguyen & Yi, 2006; Sousa et al, 2006). Other approaches assume that the descriptions are provided to the system implicitly through user context recognition facilities (Ranganathan & Campbell, 2004) or that user task descriptions are developed by application developers (Beauche & Poizat, 2008; Ben Mokhtar et al, 2007).

Next, we discuss related work in the context of three key issues, namely, the specification language, middleware support and end-user involvement.

2.1. Specification Language

The specification language is the cornerstone of the application composition process as it essentially serves the following purposes: (i) it enables service providers to advertise the properties of their services, both the functional and the non-functional properties; (ii) it enables the discovery of networked services by matching the service specifications with query requests sent by consumers; and (iii) it helps in arranging the discovered services in order of priority according to their non-functional properties and optimization criteria.

The solution presented by Paluska et al (2008) suggests a design style called “goal-oriented programming” and a proprietary scripting language for describing composite applications. The solution is centered around application developers who create applications by describing the goals and the techniques corresponding to these goals. The goals are abstract decision points which determine the application's structure and behavior by describing the functionalities required by certain parts of the application. Each goal is described in terms of quality parameters, which are used to evaluate whether or not the goal has been reached. The techniques are specified as programming scripts which describe ways of achieving goals, e.g., by using certain hardware instances. The scripts introduced by the techniques do not directly implement application functionalities. Instead, what they provide is rather an abstraction of the existing component modules and ubiquitous devices required by the application. The structure of resulting application resembles trees; this is due to the planning algorithm which is used to optimize the set of techniques.

Most of the work on dynamic composition uses proprietary XML-based specification languages bound to particular application domains (e.g., office automation, web services and mobile services). The structure of these specifications has two features: it is fixed (i.e., adding a new property requires redesigning the language) and it resembles a tree because XML is used. An XML-based specification is used, for instance, by Sousa et al (2006; 2008a) for describing user activities. In their paper, Sousa et al (2008a) present an example activity that describes the services and the service properties required to review a movie clip. Their example consists of two services, namely “play Video” and “edit Text”. The properties of these services are “material” (i.e., required files) and “service state” (i.e., video playback position, video

dimensions etc.). Sousa's specification supports two types of attributes: numeric (integer) values and enumerations (i.e., sets of integer or string values). Although XML-based languages allow the tailoring of specifications to certain problems (or application domains), they themselves do not contain any interpretation of the concepts (i.e., semantic meaning of attributes, values, etc), instead they leave this task to the system or the application. This means that the interpretation of XML-based specifications depends on the application or system logic, which can potentially create ambiguity if two different systems interpret the same specification differently.

In addition to XML-based models, multiple service specification standards exist, such as Web Services Description Language (WSDL)[1] and YAWL[2]. Although they have been used successfully in many existing systems, none of these languages has been accepted as a global standard in the application composition domain. Therefore, systems using different description languages may be incompatible with each other due to the diversity of their service and application descriptions. This is also known as semantic heterogeneity (Halevy, 2005) and it occurs because service description languages in general support different concepts (i.e., the service descriptions vary content-wise) and they may specify the same concepts in different ways. For example, service behavior can be modeled using multiple techniques, such as process modeling (e.g., BPEL[3]) and conversations (e.g., WSCL[4]).

Another alternative to the XML-based approach to specifications is the ontology-based approach which also solves the problem of semantic heterogeneity. This approach models applications, services, their properties and possible relationships between them using a common theory (also called the “upper ontology”), thus enabling the participating parties to reason and match service concepts, even if their descriptions do not comply in syntax.

Several ontology-based languages have been suggested for service descriptions, such as Web Service Ontology (OWL-S)[5] and Web Services Modeling Framework (WSMF)[6]. These languages have been used frequently, especially in dynamic application composition systems (Hesselman et al, 2006; Lee, Chun & Geller, 2004; Ben Mokhtar et al, 2007; Preuveneers & Berbers, 2005a; Preuveneers & Berbers, 2005b; Ranganathan & Campbell, 2004). For example, iCOCOA (Ben Mokhtar et al, 2007) uses an OWL-S based semantic language for specifying user activities, services and their properties. iCOCOA particularly focuses on dynamic service properties and models service behavior using workflows. Thus, each service is modeled as a set of service operations which are interconnected with control and data relationships. In addition, iCOCOA also describes service QoS properties using qualitative and quantitative attributes.

Another OWL-S based specification language is used by CODAMOS middleware (Preuveneers & Berbers, 2005a; Preuveneers & Berbers, 2005b). The main focus of the CODAMOS middleware is the hierarchical composition of service-based mobile systems, in which each service can consist of a sequence (i.e., a hierarchy) of other services. The CODAMOS specification defines the functional and non-functional properties of the services and relationships between them, i.e., connectors. The connectors link services and provide communication channels within the composed structures of the services. The non-functional properties of the services include contracts specifying user requirements and context and service control interfaces.

The language used for the application and service specification has a significant impact on middleware support in dynamic application composition. This point is discussed further in the next section.

[1] http://www.w3.org/TR/wsdl

[2] http://www.yawl-system.com/

[3] http://docs.oasis-open.org/wsbpel/

[4] http://www.w3.org/TR/wscl10/

[5] http://www.daml.org/services/owl-s/1.0/

[6] http://www.wsmo.org

2.1. Middleware Support

We distinguish between two important middleware functionalities in the dynamic composition of ubiquitous applications: (i) the application composer, which realizes the application by implementing a matching or a planning algorithm to choose the necessary services and (ii) the functionality that enables the interoperability of the service discovery, the service descriptions and the service communication.

Application composer. The application composer implements an algorithm to select service instances which realize the application. The algorithm performs either a matching or a planning function. Matching algorithms select appropriate services simply by matching the attributes of the services. In contrast, planning algorithms perform optimization and select the set of services which best satisfies a certain criteria. Planning algorithms are usually applied to systems in which finding a solution requires significant time and computing resources.

Application composition by matching is used in the iCOCOA (Ben Mokhtar et al, 2007), InterPlay (Messer et al, 2006), PCOM (Handte et al, 2005), CASE (Hesselman et al, 2006), USON (Takemoto et al, 2004), Galaxy (Nakazawa et al, 2004) and SesCo (Kalaspur et al, 2005) projects. For example, iCOCOA suggests an application composition engine that uses semantic reasoning as well as a QoS attribute- and a conversation-based matching algorithm. The algorithm dynamically integrates the available service instances into the application according to the service behavior and application QoS constraints.

Several solutions use planning algorithms for application composition (Beauche & Poizat, 2008; Chantzara et al, 2006; Preuveneers & Berbers, 2005a; Ranganathan & Campbell, 2004; Rouvoy et al, 2009; Sousa et al, 2006; Sousa et al 2008a). For example, Sousa et al (2006, 2008a) and Ranganathan & Campbell (2004) use similar planning approaches to compose applications and particularly to address fault-tolerance issues. Their application composition engines take the goal description of an abstract user into account, in addition to the user's current context and preferences, in order to find a sequence of actions which leads to the best realization of the user activity. The resulting sequence (or plan) has to be executed by the application framework to ensure that none of the executions fail because of resource unavailability. This is done by dynamically monitoring the execution of the plan and the resources.

The composition mechanism of MUSIC (Rouvoy et al, 2008; Rouvoy et al, 2009) uses a utility-based planning algorithm, which relies on the normalized utility function determined by the required properties of the application and its current execution context. The utility function defines the relevancy of the QoS properties and reflects the application state (i.e. deployed or running), which affects the way particular QoS properties are estimated. This planning algorithm is also capable of negotiating QoS values directly with service providers during planning.

Another project, CODAMOS (Preuveneers & Berbers, 2005a), uses the Protégé reasoning tool to take the capacities of the client devices into consideration during planning. The algorithm estimates the resource capacities of the client devices and composes the applications accordingly. The algorithm uses the backtracking approach to optimization, i.e., it cuts down on the user preferences if it does not find any suitable solutions which fit the required device set. CODAMOS is a particularly interesting project, as their algorithm optimizes the structure and the functionality of the application according to the available devices, instead of optimizing the set of services to meet the application QoS requirements.

Middleware interoperability. According to Ben Mokhtar (2007), dynamic application composition requires two types of middleware interoperability, that is, among service discovery and among service communication protocols. As discussed in 2.1, the former relates to overcoming semantic heterogeneity and can be addressed through the semantic description of services and composite applications. The latter requires adequate mapping among heterogeneous middleware protocols.

A number of solutions have been proposed recently to address the interoperability issues of the service discovery and communication functionalities. For example, MUSDAC (Raverdy, Issarny, Chibout & de la Chapelle, 2006) and ubiSOAP (Caporuscio, Raverdy & Issarny, 2009) use auxiliary service components to translate messages sent between different service discovery networks. An instance of such a service is added to each service discovery network enabling the clients to use multiple protocols at the same time. Siebert, Cao, Zhou, Wang & Raychoudhury (2007) and Bromberg & Issarny (2005), on the other hand, suggest a universal adaptor approach which implements both client and server-side functionalities for discovering services and for mapping the primitives used by the universal adaptor with the primitives used by various service discovery systems.

Similarly, the ANSO architecture (Bottaro et al, 2007; Jianqi & Lalanda, 2008) suggests using adaptive adaptors to target service discovery heterogeneity issues. ANSO uses the UPnP service discovery protocol and provides explicit mappings of other protocols for UPnP in order to integrate sensors, computing devices and web services into one coherent and manageable network. Unlike other solutions, the ANSO service discovery engine generates a separate proxy component for each service instance which needs to use some other protocol than UPnP. Each time the service providers register a new service, ANSO dynamically generates a proxy using Java reflection (i.e. the bytecode generation technique) which implements the service discovery protocol required by the service instance. The proxies also provide access to the service functionalities.

Application composition also needs another kind of interoperability which is related to service communication protocols. Service communication interoperability is required because service instances may use incompatible communication protocols, such as Bluetooth and Wi-Fi. This kind of interoperability has been addressed in particular by the AmIi (Georgantas, Issarny, Ben Mokhtar, Bromberg, Bianco, Thomson, Ravedy, Urbeita & Cardoso, 2009) and ubiSOAP (Caporuscio et al, 2009) solutions. Although AmIi is a middleware solution which focuses on semantic interoperability, it also provides interoperability among heterogeneous RPC-protocols. For this reason AmIi introduces a proprietary AmIi-COM communication mechanism, which is based on runtime protocol translation. Similar to AmIi, the ubiSOAP middleware addresses communication interoperability. It implements a custom SOAP protocol to enable service communication in wireless networks. We discuss these two solutions in Section 3.3.

2.3. End-User Involvement

In ubiquitous computing, end-users rarely play an active role. This is best demonstrated by research in context-aware application composition in which users do not explicitly interact with the system (i.e., through a user interface), but rather influence the decisions made by the system passively, i.e., through sensors which autonomously capture the users' preferences, their behavior, current needs and other parameters. For this reason, end-user involvement in ubiquitous computing is very limited and often non-existing.

However, end-user involvement has been more extensively studied in AOC research. The AOC systems assume that application composition is performed on the basis of predefined activity templates that are developed by application programmers, thus restricting the role of the end-users in composing applications (or activities) to simply matching the templates. This is simultaneously, both a major drawback and a contradiction in the AOC approach, because, on one side, the AOC promises to support end-user activities, but on the other side it restricts the user choice to activity templates that are predefined by the system. Thus, studies on end-user involvement in AOC have mainly focused on interfaces used to customize user activity templates.

For example, Sousa et al (2006, 2008a) present a set of user interfaces for customizing activity templates and specifying end-user preferences. These preferences define the constraints and requirements that are taken into account by the application composer, which chooses the service instances for the corresponding user activity. The example shows a user activity

template, which includes the following tasks: editing text, browsing the Web and editing a spreadsheet. Users can associate each of these single tasks with specific material (filename or address). Sousa's user interfaces also support multiple dimensions of application QoS requirements and the particular value of each dimension is represented by the slider position. The user chooses the QoS dimension (e.g., latency) and then adjusts the position of the slider to define the value intervals “bad”, “moderate”, “good” and “excellent”.

Fig. 2. Physical user interface for providing user preferences used by Davidyuk et al (2008a).

In our previous study, we suggested another approach to collect user preferences based on physical interfaces (Davidyuk et al, 2008a). In this approach, users specify their preferences by touching the appropriate RFID tags. An example of such an interface, the interface of a ubiquitous movie player application, is shown in Figure 2. This interface allows the users to choose the quality of the video they want to play with the application, which can be “very low”, “low”, “medium” or “high”. For example, in order to choose the maximum quality, users need to touch (i.e. read the tag using a mobile terminal) the corresponding RFID tag labeled “high”. Although we find that Sousa's interfaces are more flexible in terms of the value ranges of the captured preferences, RFID-based interfaces demonstrate a higher usability and also require less learning effort.

The InterPlay middleware (Messer et al, 2006) provides several user interfaces for querying device and content information and for obtaining the status of user activities at home. This interface set also includes a work interface for activity composition similar to the one in Sousa's study, which uses a verb-subject-device template. In order to compose an activity, users perform three steps in this interface: (i) they choose an action from the “play”, “print” and “show” options, then (ii) they choose a material (i.e., content type) from the “movie”, “music” and “photo” options and finally (iii) they choose the target device they want to use to watch the content. However, the template only allows the user to choose one device instance per user activity. If a user needs to compose an activity from two device instances, then this kind of template will not support it.

2.4. Summary

Our review of dynamic application composition solutions can be summarized by stating that in the related work the XML-based service specification languages are primarily used. These languages are easier to design than, e.g. ontology-based alternatives, but they neither allow reasoning nor encoding interpretation (i.e., meaning) of specification concepts. Thus, XML-based specifications may cause semantic heterogeneity (i.e., ambiguity) issues. Therefore, we consider the ontology-based specification approach as the most promising solution for supporting application composition. Such an approach does not require global agreement among service providers and consumers on a specification standard, thus different legacy specification standards can be supported by mapping them to a common ontology. In addition, this approach

offers greater flexibility and expressiveness compared to XML-based specification languages. Still, there is one argument against using ontologies in specifications: they increase the overall latencies, because the ontologies need to be processed.

Solution	Specifi-cation	Composer	Interope-rability	End-User Involvement
Paluska et al (2008)	Script-based	Planning	-	-
Sousa et al (2006, 2008a)	XML	Planning	-	Yes
iCOCOA (Ben Mokhtar et al, 2007)	OWL-S	Matching	Semantic	-
Gaia (Ranganathan & Campbell, 2004)	DAML	Planning	Semantic	-
PerSo (Beauche & Poizat, 2008)	YAWL	Planning	-	-
InterPlay (Messer et al, 2006)	RDF	Matching	-	Yes
PCOM (Handte et al, 2007)	XML	Matching	-	-
CODAMOS (Preuveneers & Berbers, 2005a)	OWL-S	Planning	Semantic	-
ANSO (Bottaro et al, 2007a)	XML	-	Semantic	-
MUSIC (Rouvoy et al, 2009)	XML	Planning	-	-
CASE (Hesselman et al, 2006)	OWL-S	Matching	Semantic	-
USON (Takemoto et al, 2004)	XML	Matching	-	-
Galaxy (Nakazawa et al, 2004)	XML	Matching	-	-
SesCo (Kalaspur et al, 2005)	XML	Matching	-	-
IST-Context (Chantzara et al, 2006)	XML	Planning	-	-
DRACO (Rigole et al, 2005)	XML	-	-	-

Table 1. Comparison of related work.

As can be seen from Table 1, only a few of the discussed ubiquitous application composition solutions deal with middleware interoperability. Supporting interoperability is the cornerstone functionality, which allows an application composition system to utilize services that are specified in different service description languages and use various communication protocols. Since the large number of existing (and well-established) service discovery protocols, service description languages and service communication protocols make the adoption of one unique solution a rather unrealistic scenario, then supporting middleware interoperability is an essential requirement for making an application composition system truly ubiquitous.

End-user involvement has been studied in the context of ubiquitous application composition by Sousa et al (2006, 2008a) and Messer et al (2006). However, these two approaches assume that application composition is performed on the basis of predefined activity templates (i.e., application descriptions), which are developed by professional programmers. As a result, end-users are not able to create their own applications and activities according to their own needs. Ideally, applications should be created by the end-users themselves, as they are the ones with in-depth knowledge of their own needs and activities. Involving users in the process of creating applications would result in a better understanding of how the applications should be created and what services should be used. In addition, it would give the users a feeling of having more control over the environment, which is an important factor in ensuring user acceptability of prototypes as demonstrated by Davidyuk et al (2008a).

3. MEDUSA Middleware

In this section we discuss our earlier research on application composition and explain how our findings motivated the development of the MEDUSA middleware.

In our previous work on ubiquitous application composition we have developed two system prototypes which were used in building our example applications (Davidyuk et al, 2008a; Davidyuk et al, 2008b; Davidyuk et al 2009). These prototypes include a proprietary service discovery protocol, an application composition algorithm and support composite multimedia applications using the application deployment and messaging facilities of the REACHES framework (Sánchez, Cortés & Riekki, 2007). The prototypes use a mobile terminal as a remote control unit which allows the users to create audio/video playlists by touching physical objects associated with certain multimedia files (Davidyuk et al, 2008a; Davidyuk et al, 2009). The applications in both prototypes have fixed structures, i.e. composition is performed using predefined application templates.

We also conducted a user study to evaluate the feasibility of the first prototype, which relied on an autonomic algorithm to compose applications. We reported this study in Davidyuk et al (2008a). The end-user involvement in that prototype was very limited and focused on application-related issues, such as choosing multimedia content and specifying preferences over it. As the result, we observed that the user acceptance of the autonomous application composition was very low, because the users were bound by the decisions made by the algorithm. Thus, a key finding was that end-user control in application composition is necessary.

We used the results from the feasibility test as a base for designing our second prototype, CADEAU (Davidyuk et al, 2009) which focuses on end-user control in application composition. Unlike the first prototype, CADEAU allows the user to choose service instances (more specifically, services that represent real devices) manually by touching or interactively. In other words, the users are able to choose the most appropriate means of interaction according to their needs and the situation at hand. However, the prototype restricts application composition to simply matching the predefined application descriptions with the services discovered according to a given criteria. Therefore, the matching approach to composition forces the end-users to rely on applications designed by application developers, instead of giving them the possibility to create their own applications in a do-it-yourself fashion.

The main goals of the MEDUSA middleware are related to providing end-user support for the creation and customization of applications and for controlling the composition process according to user needs. To achieve these goals, MEDUSA utilizes a composition tool for encoding user intent into applications and a set of control interfaces. These interfaces are based on our previous work (Davidyuk et al, 2009). Another important issue addressed by MEDUSA is interoperability between heterogeneous devices, networks and platforms. Achieving interoperability is a prerequisite for building an open application composition system, as the services constituting an application need to be able to discover each other, exchange information and indeed interpret this information meaningfully. This is especially important if the environment, in which the application composition system operates, consists of services implemented and deployed by independent providers (Ben Mokhtar, Raverdy, Urbeita & Speicys Cardoso, 2008).

The following section introduces the overall architecture of the MEDUSA middleware (Section 3.1) paying special attention to end-user support (Section 3.2) and interoperability (Section 3.3).

3.1. MEDUSA Middleware Architecture

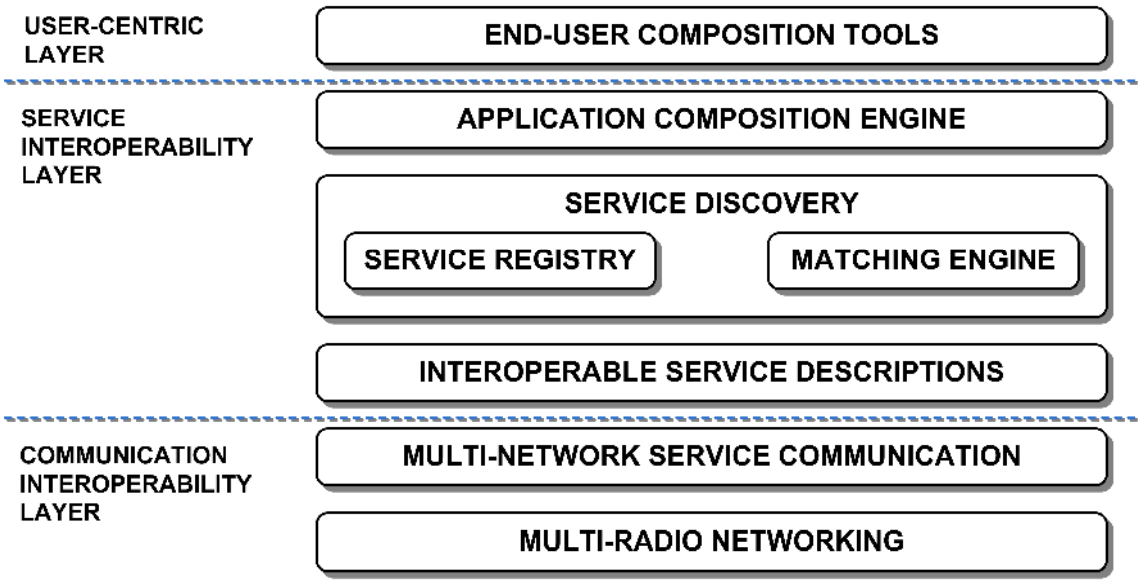

Fig. 3. MEDUSA conceptual architecture.

The architecture of MEDUSA is decomposed into the following layers; including end-user support and communication management in ubiquitous computing environment (see Figure 3):

1) The communication interoperability layer uses a common network interface to both integrate and hide the underlying multi-radio networking technologies. The layer contains two entities, namely multi-radio networking and multi-protocol network management (Caporuscio et al, 2009). The first one effectively manages the nodes' multi-radio facilities using a cross-network addressing scheme and provides point-to-point and multicast communication primitives. Whereas, the second entity is responsible for multi-protocol network management, communication mobility and multi-network routing. In other words, it handles the mobility of the nodes to the upper layers in a transparent manner and also enables messages to be routed across nodes physically located in different networks. The realization of this layer is further discussed in Section 3.3.

2) The key role of the service interoperability layer is to enable semantic and syntactic interoperability between different service providers and consumers without imposing them to use a specific standard. Interoperability is achieved using a common service description model, which specifies the mapping function between the service concepts and the functionalities provided by the platform nodes (Ben Mokhtar, 2007). In addition to this, the layer is also responsible for the service discovery, which stores descriptions of available services and performs matchmaking (i.e. searches for service descriptions in the repository matching the service query). The MEDUSA service discovery supports various legacy service discovery protocols through pluggins. This enables interoperability as presented by Georgantas et al (2009). The application composition engine employs multiple composition algorithms to produce application configurations, which are optimized using QoS requirements and user-defined criteria (Davidyuk et al, 2008b). We describe the realization of this layer in Section 3.3.

3) The user-centric layer provides functionality for creating and customizing applications and interfaces. The functionality can be used to control the composition of applications at run-time, and it can be used by end-users to encode their intents into the applications before they are composed by the application composition engine. In addition,

end-users can utilize this functionality to control the composition process and to adapt applications at run-time by providing information on their preferences and by choosing the service instances that constitute the application. The MEDUSA end-user support is explained in further detail in the following section.

3.2. End-User Support

The functionality of the user-centric layer is provided by the end-user application composition tool and the application control interfaces. The composition tool helps users to arrange services into applications according to their own needs. We assume that each ubiquitous environment provides a set of cards which are associated with the service instances available in the environment. These cards can be issued, e.g., by the administrator who is responsible for maintaining and installing the actual services. Thus, each service is represented with a square paper card with a graphical icon on one side and an RFID tag attached the other side. A sticker with an RFID tag used in our prototype and an example set of cards representing a file server, a display, a remote controller, and an audio service are shown in Figure 4. Each of these RFID tags contains a web link to its service description which can be read by touching the card with an RFID-equipped mobile phone, as shown in Figure 4.B. Users can arrange the cards into different structures or sequences (an example sequence is shown in Figure 4.C), which are then read by touching them with a remote controller. In addition to sequences, other structures are also supported, however, they require connection cards to combine different parts of application structures. The main advantage of using a physical RFID-based interface for application composition is that it enables user cooperation and collaboration in designing ubiquitous applications, which would be difficult to realize using a traditional desktop-based user interface.

Once the application structure is provided to the system by touching service cards, end-users have to specify control and data dependencies between the services they have chosen through the mobile phone UI. This step can be supported using a mobile phone-based assistant, e.g. the office assistant in MS Word, which will guide the users through the application composition process. So, for example, if the users designed an application which is not complete structure-wise, the assistant would suggest how to complete the application using e.g., information from a database with existing application descriptions. Similar approach is used by Wisner & Kalofonos (2007) for programming smart homes.

Fig. 4. Sticker with RFID tag (A), RFID reader (B), and set of service cards (C).

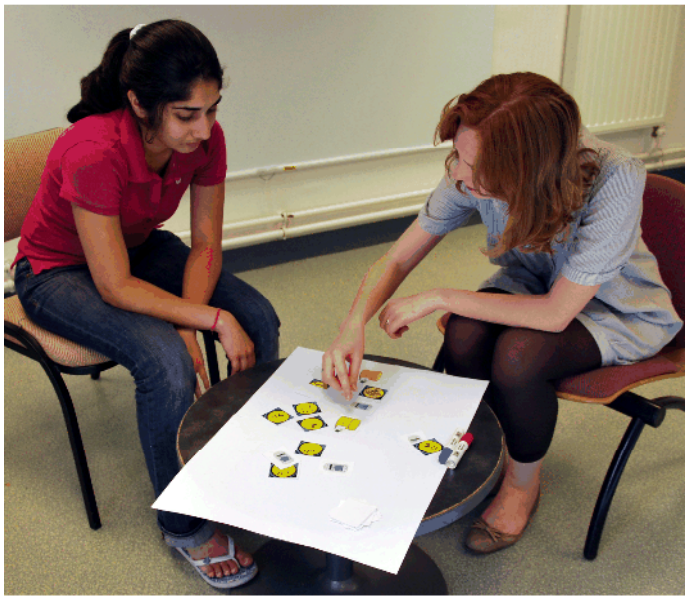

Fig. 5. Two MEDUSA users jointly developing an application.

We have tested our approach with a prototype composition tool and an initial set of cards. Figure 5 shows test subjects jointly implementing an application using a mobile phone. Two users participated in the preliminary experiment which lasted 1.5 hours. The users were asked to design abstract applications using the given service cards. In addition, they were allowed to use service cards they thought up themselves, if necessary. Altogether six applications were designed for multiple domains and several additional services were suggested during the experiment. The application domains included home, office, hospital and learning environments. We learned three lessons from this experiment. First, the graphical designs of the tags (i.e. icons) have to be self-explanatory and very intuitive for the users, and match their technical background. Secondly, we found that our set of service cards for application composition should be further extended. Thirdly, our composition tool needs a mechanism to motivate users to build applications, because motivation is necessary to balance the effort needed to learn to use the tool.

When the users develop applications using the composition tool, the applications only exists as abstract descriptions which have to be realized by service instances. In other words, the services constituting an application have to be connected at run-time to the service instances available in the environment. This task is performed by the application composer and it can be controlled by users through a set of user interfaces, which have two functions: (i) they allow users to choose among the possible application configurations (i.e. they are able to map the application descriptions to the service instances) suggested by the application composition engine; (ii) they permit users to directly choose service instances which are physically available in the environment. MEDUSA offers four different control interfaces, which enable different degrees of user involvement in controlling the application composition.

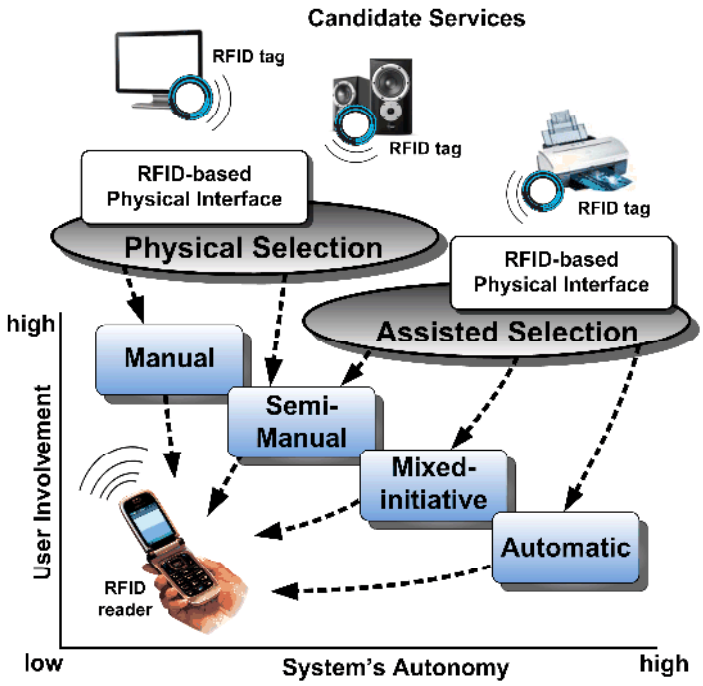

Fig. 6. MEDUSA end-user control interfaces.

The MEDUSA middleware supports the following interfaces: manual, semi-manual, mixed-initiative and automatic. The interfaces are shown in Figure 6 where they are arranged according to the level of user involvement and system autonomy they provide.

The manual interface assumes that the users have full control over application composition. Thus, the application composition engine is not utilized at all. The users can choose service instances by touching them with their mobile phone, as shown in Figure 7 (left). The interface is based on RFID technology and each service instance has an RFID tag attached to it. Each occasion when the tag is touched it uniquely identifies the service instance associated with the tag. Non-visual resources, i.e., abstract services or services located in a hard to reach places (e.g., on the ceiling) are represented with RFID-based control panels. Figure 7 (right) shows an example of such a control panel for a multimedia projector service. The application is started once the user has chosen all the necessary service instances.

Fig. 7. User selecting a service instance (left), and example service control panel (right).

The semi-manual interface does not require users to choose manually all the services. The users can select some services by touching while the application composition engine will complete the application configuration by assigning the missing services automatically. The advantage of this interface is that in the fact only a part of the application configuration will be

realized by the system. Thus, the users can choose the most important services in their opinion, and leave the less important decisions to be made automatically by the system.

The mixed-initiative interface restricts user control to the options suggested by the application composer. However, the users can always switch to another control interface if they are not satisfied with the options suggested by the system. The options are dynamically produced and ordered according to user defined criteria into a list of application configurations starting with the most attractive one. The list also shows the service instances required by each application. Before the application starts, the user can browse through the application configuration list and identify the services used by clicking the phone's soft buttons. This feature is essential if users need to validate the configuration of the application before starting it.

The autonomic interface uses the application composer to start and run an application configuration without distracting the user. This interface assumes that the user does not want to control the application composition.

3.3. MEDUSA Middleware Interoperability

The MEDUSA communication interoperability layer is realized using ubiSOAP (Caporuscio et al, 2009). The ubiSOAP communication middleware is specifically designed for resource-limited portable ubiquitous devices which can be interconnected through multiple wireless links (i.e. Bluetooth, Wi-Fi, GPRS, etc). This feature is essential for MEDUSA, because the front-end functionality of the middleware is intended for mobile phones. In addition, ubiSOAP adopts Web Service standards as a baseline for implementing ubiquitous services, extends the standard SOAP protocol with group messaging connectivity and supports node mobility. ubiSOAP has been implemented into two SOAP engines, Axis2[7] and CSOAP[8], which demonstrates that ubiSOAP allows legacy applications to communicate using the standard SOAP protocol. This feature guarantees the compatibility of MEDUSA with thousands of existing online services.

Figure 8 shows the two-layered architecture of ubiSOAP (for details please refer to Caporuscio et al (2009). The lower layer is the ubiSOAP connectivity layer which selects the network based on user policies, as users may require the utilization of a certain type of network for personal reasons. This layer also identifies and addresses the applications in the networking environment. The upper layer is the ubiSOAP communication layer which extends the use of the standard SOAP protocol for messaging between the participating services by introducing SOAP multi-network multi-radio point-to-point transport and group (multicast) transport protocols.

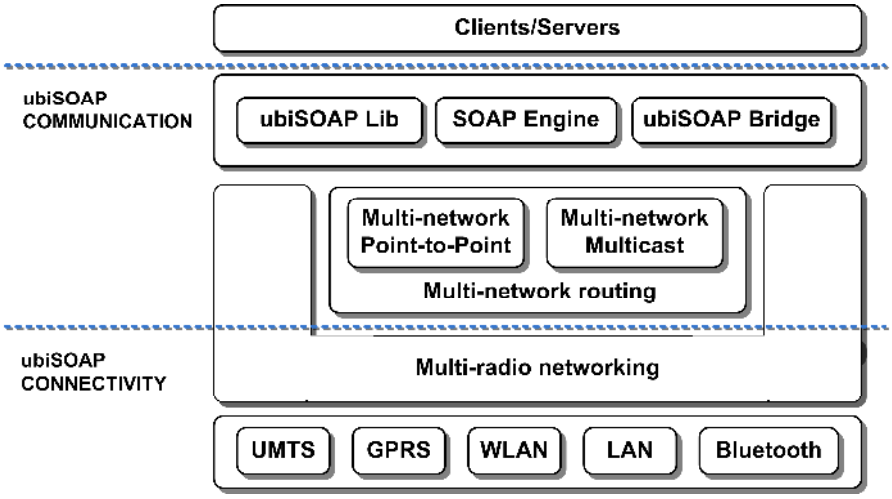

Fig. 8. ubiSOAP architecture. Adapted from Caporuscio et al (2009).

[7] available from http://ws.apache.org/axis2/

[8] available from http://www-rocq.inria.fr/arles/download/ozone/

The upper MEDUSA service interoperability layer is realized by adopting the AmIi (Ambient Intelligence interoperability) service description model (Georgantas et al, 2009) and the interoperable ambient service discovery (Ben Mokhtar, 2007). AmIi is a semantic-based solution which achieves conceptual interoperability between heterogeneous service platforms. This type of interoperability is required from service providers wanting to register services using different service discovery protocols, which in turn use different service description languages. AmIi relies on the interoperable service description model and the multi-protocol service discovery engine. The former enables mapping between the heterogeneous service description languages, thus providing conformance on both a syntactical and a semantic level. As a result, the service descriptions, which were originally written in languages such as UPnP or WSDL, can be translated into corresponding interoperable service descriptions. As shown in Figure 9, the AmIi service description model captures both functional (i.e. interface-related) and non-functional (e.g., QoS) service properties in addition to conveying information on service behavior and service grounding. The latter specifies how the service can be accessed using a legacy communication protocol. The service behavior is modeled using a workflow language e.g. BPEL. However, the model also supports other alternatives. The functional and non-functional properties are either specified with a reference to an existing ontology or they may contain embedded semantic descriptions.

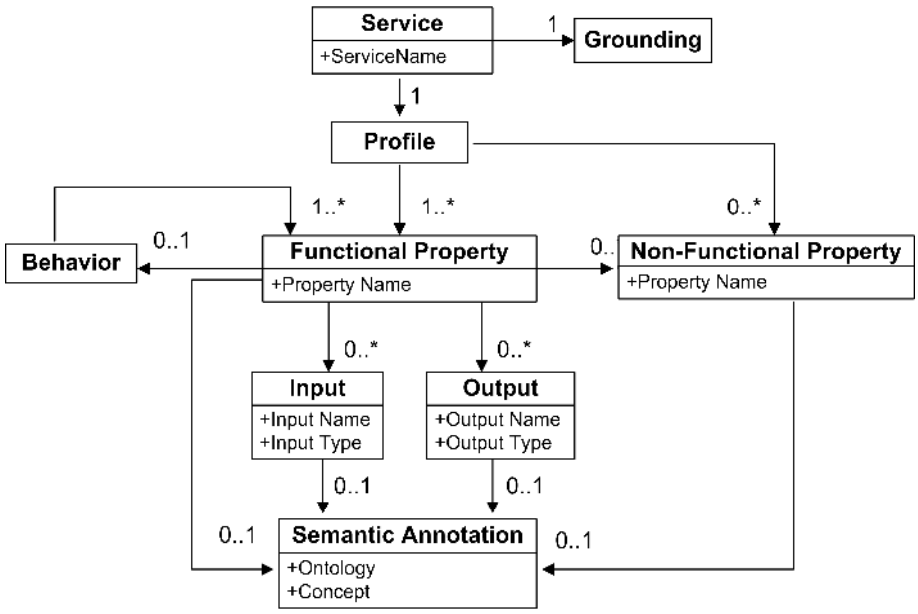

Fig. 9. AmIi Service Description Model (AMSD), adapted from Ben Mokhtar (2007).

The interoperable service discovery function provided by AmIi is achieved using distributed service repositories which support legacy service discovery protocols. This means that each repository runs a set of plugins (i.e., proxies) associated with various service discovery protocols. The legacy service discovery protocols are then able to exchange service descriptions with the repositories and answer query requests. However, such a mechanism requires the translation of all the service descriptions into one interoperable format, which may potentially increase the latency of the discovery protocol. Therefore, the interoperable ambient service discovery also supports a plugin for registering and querying the service descriptions directly in the interoperable format.

The application composition engine in MEDUSA is realized using the composition algorithms initially introduced in Davidyuk et al (2008b). These optimization algorithms are based on the theories of evolutionary and genetic computing and they are used to optimize application configurations (i.e. set of services that constitute the application). The algorithms perform the optimization on the basis of user specified criteria (the nearest, the fastest or the

cheapest option) and user preferences (fidelity and QoS requirements). For example, an application configuration can be optimized in order to minimize the overall application bandwidth consumption and to maximize the QoS properties of interest. These algorithms are generic and support (i) customizable criteria which may include multiple simultaneous optimization goals and (ii) various application QoS property types which can be added and removed from the service descriptions as needed at run-time.

4. Conclusions

The vision of activity-oriented computing advocates the development of systems which are explicitly aware of user needs and activities. These activities consist of the users' everyday tasks (e.g., in a home or office environment), which are abstractly described in terms of the situation in which the activities take place, the required system functionalities for accomplishing the activities, and the user preferences such as fidelity, privacy, or using other non-functional requirements. These activities are dynamically realized by the system composing the applications according to the users' activity descriptions. Although existing AOC systems are autonomous and rarely involve users in the actual application composition, we argue that end-user support in such systems should not be neglected.

To address this issue, we present MEDUSA, our middleware solution for end-user application composition with a two-fold goal: (i) enabling end-user application development and controlling the composition of applications at run-time; and (ii) solving issues relating to the interoperability of service communication and service discovery, thus promising to integrate thousands of already existing services. We discussed the middleware architecture, its functionality and also presented realizations of MEDUSA utilizing the open-source solutions ubiSOAP and AmIi. MEDUSA end-user support helps avoid predefined activity descriptions, which are used in most of the related work. The end-user composition tool supports users who do not have advanced technical skills. The tool relies on mobile terminals and RFID-based physical interfaces. These technologies were found very promising in terms of usability and user acceptance (Davidyuk et al, 2008a).

One lesson learned from this work is that end-users have a need to create their own services, thus the role of the end-user has to expand into the role of a service developer. Our plan is to achieve this change iteratively, in a few steps. For example, at first our system will allow an exporting functionality which enables user-composed applications to be exported as services, so that these applications can be used to compose other applications.

In the future we are planning on experimenting with the middleware and the end-user composition tool. In addition to conducting performance measurements, we will evaluate our solution with a series of user experiments in order to assess factors relating to user acceptance and usability. It would also be interesting to study what kinds of applications the end-users usually compose and what kinds of services they typically need in different situations.

We are also planning on making the service cards from thick cardboard or pieces of wood, thus the cards would resemble pieces of a jigsaw puzzle. By making different cutout interfaces we will restrict the ways in which the services can be combined together. This is necessary because some services may not be permitted to combine directly, but they can be combined using an auxiliary service in between.

Acknowledgements

This work has been funded by Tekes (National Technology Agency of Finland) under UBICOM technology program, GETA (Finnish Graduate School in Electronics, Telecommunications and Automations) and Nokia Foundation.

The authors would like to thank personnel of ARLES team in INRIA, especially Animesh Pathak, Pushpendra Singh, Elena Kuznetsova and Sneha Godbole for their valuable comments and ideas regarding the functionality of the MEDUSA middleware and the content of this chapter.

Keywords

Service-oriented and ubiquitous computing, ambient intelligence, end-user and interaction design, application composition.

References

Amigo. (2009). Project web site. Retrieved on November 16, 2009, from http://www.hitech-projects.com/euprojects/amigo

Beauche S. & Poizat P. (2008). Automated Service Composition with Adaptive Planning. In Bouguettaya, A., Krueger, I. & Margaria, T. (Eds.), *Proceedings of the 6th International Conference on Service-Oriented Computing* (pp. 530-537). Berlin, Heidelberg: Springer-Verlag.

Bertolino, A., De Angelis, G., Frantzen, L. & Polini, A. (2008). The PLASTIC Framework and Tools for Testing Service-Oriented Applications. *In Software Engineering: International Summer Schools, ISSSE 2006-2008* (pp. 106-139).

Bertolino, A., De Angelis, G. & Polini, A. (2009). Online Validation of Service Oriented Systems in the European Project TAS3. *Proceedings of the ICSE Workshop on Principles of Engineering Service Oriented Systems, PESOS* (pp. 107-110). Washington DC: IEEE Computer Society.

Bottaro, A., Bourcier, J., Escofier, C. & Lalanda, P. (2007a). Context-Aware Service Composition in a Home Control Gateway. *Proceedings of the IEEE International Conference on Pervasive Services.* Washington DC: IEEE Computer Society.

Bottaro, A., Gerodolle, A. & Lalanda, P. (2007b). Pervasive Service Composition in the Home Network. *Proceedings of the 21th International Conference on Advanced Networking and Applications AINA'07* (pp. 596-603), Washington, DC: IEEE Computer Society.

Bromberg, Y.-D. & Issarny, V. (2005). INDISS: Interoperable Discovery System for Networked Services. In Alonso, G. (Ed.) *Proceedings of the of ACM/IFIP/USENIX 5th International Conference on Middleware Middleware'05* (pp. 164-183), New York, NY: Springer-Verlag New York.

Buford, J., Kumar, R., & Perkins, G. (2006). Composition Trust Bindings in Pervasive Computing Service Composition. *Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications Workshops,* Washington DC: IEEE Computer Society.

Caporuscio, M., Raverdy, P.-G. & Issarny, V. (2009). *ubi*SOAP: A Service Oriented Middleware for Ubiquitous Networking. *Journal of Transactions on Service Computing.* To appear.

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.

Davidyuk, O., Sánchez, I., Duran, J. I. & Riekki, J. (2008a). Autonomic Composition of Ubiquitous Multimedia Applications in REACHES. *Proceedings of the 7th International ACM Conference on Mobile and Ubiquitous Multimedia, MUM'08* (pp. 105-108). New York, NY: ACM.

Davidyuk, O., Sánchez, I., Duran, J. I. & Riekki, J. (2009). CADEAU: Collecting and Delivering Multimedia Information in Ubiquitous Environments. In Kamei K. (Ed.) *Adjunct Proceedings of the 7th International Conference on Pervasive Computing PERVASIVE'09* (pp. 283-287).

Davidyuk, O., Selek, I., Duran, J. I. & Riekki, J. (2008b). Algorithms for Composing Pervasive Applications, *International Journal of Software Engineering and Its Applications*, 2 (2), 71-94.

Georgantas, N., Issarny, V., Ben Mokhtar, S., Bromberg, Y.-D., Bianco, S., Thomson, G., Raverdy, P.-G., Urbieta, A. & Speicys Cardoso, R. (2009). Middleware Architecture for Ambient Intelligence in the Networked Home. In Nakashima, H., Augusto, J. C. & Aghajan, H. (Eds.), *Handbook of Ambient Intelligence and Smart Environments*. Springer. To appear.

Halevy, A. (2005). Why Your Data Wont Mix. *ACM Queue* 3(8), 50-58.

Handte, M., Herrmann, K., Schiele, G. & Becker, C. (2007). Supporting Pluggable Configuration Algorithms in PCOM. *Proceedings of International Workshop on Pervasive Computing and Communications* (pp. 472-476).

Hardian, B., Indulska, J. & Henricksen, K. (2008). Exposing Contextual Information for Balancing Software Autonomy and User Control in Context-Aware Systems, *Proceedings of the Workshop on Context-Aware Pervasive Communities: Infrastructures, Services and Applications CAPC'08.* (Sydney, May, 2008).

Hesselman, C., Tokmakoff, A., Pawar, P. & Iacob, 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: Springer-Verlag.

Jianqi, Y. & Lalanda, P. (2008). Integrating UPnP in a Development Environment for Service-Oriented Applications. *Proceedings of the IEEE International Conference on Industrial Technology ICIT'08,* (pp. 1-5).

Lee, Y., Chun, S. & Geller, J. (2004). Web-Based Semantic Pervasive Computing Services. *Proceedings of the IEEE Intelligent Informatics Bulletin*, 4(2).

Kalasapur, S., Kumar, M. & Shirazi, B.A. (2005). Personalized Service Composition for Ubiquitous Multimedia Delivery. *Proceedings of the 6th IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks WoWMoM'05* (pp. 258-263), Washington, DC: IEEE Computer Society.

Kawsar, F., Nakajima, T. and Fujinami, K. (2008). Deploy Spontaneously: Supporting End-Users in Building and Enhancing a Smart Home. *Proceedings of the 6th International Conference on Ubiquitous Computing, UbiComp'08* (pp. 282-291). Vol. 344. New York, NY: ACM.

Masuoka, R., Parsia, B. & Labrou, Y. (2003). Task Computing - the Semantic Web meets Pervasive Computing. In Fensel D. et al (Eds.) *Proceedings of the 2nd International Semantic Web Conference ISWC'03* (pp. 866-881), Lecture Notes In Computer Science, Vol. 2870. Berlin, Heidelberg: Springer-Verlag.

Mavrommati, I. & Darzentas, J. (2007). End User Tools for Ambient Intelligence Environments: An Overview. In Jacko J. (Ed.) *Human-Computer Interaction Part II* (pp. 864-872). Springer-Verlag Berlin Heidelberg.

Mavrommati, I. & Kameas, A. (2003). End-User Programming Tools in Ubiquitous Computing Applications. In Stephanidis C. (Ed.), *Proceedings of International Conference on Human-Computer Interaction* (pp. 864-872). London, UK: Lawrence Erlbaum Associates.

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.

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

Ben Mokhtar, S., Georgantas, N. & Issarny, V. (2007). COCOA: COnversation-based Service Composition in PervAsive Computing Environments with QoS Support. Journal of Systems and Software, 80(12), 1941-1955.

Ben Mokhtar, S., Raverdy P.-G., Urbieta, A. & Speicys Cardoso, R. (2008). Interoperable Semantic and Syntactic Service Matching for Ambient Computing Environments. *Proceedings of the 1st International Workshop on Ad-hoc Ambient Computing*, France.

Nakano, Y., Takemoto, M., Yamato, Y. & Sunaga, H. (2006). Effective Web-Service Creation Mechanism for Ubiquitous Service Oriented Architecture. *Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services CEC/EEE'06* (p. 85). Washington DC: IEEE Computer Society.

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).

Nokia. (2009). *Point&Find API for mobile phones.* Retrieved on November 16, 2009, from http://pointandfind.nokia.com

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

PLASTIC. (2009). *Project's web site*. Retrieved on November 16, 2009, from http://www-c.inria.fr/plastic/the-plastic-middleware

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

Preuveneers, D. & Berbers, Y. (2005b). Semantic and Syntactic Modeling of Component-Based Services for Context-Aware Pervasive Systems Using OWL-S. *Proceedings of the 1st International Workshop on Managing Context Information in Mobile and Pervasive Environments* (pp. 30-39).

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.

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.

Raverdy, P.-G., Issarny, V., Chibout, R. & de La Chapelle, A. (2006). A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments. *Proceedings the 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services MOBIQUITOUS'06* (pp. 1-9), Washington DC: IEEE Computer Society.

Riekki, J. (2007). RFID and Smart Spaces. *International Journal of Internet Protocol Technology*, 2(3-4), 143-152.

Rigole, P., Vandervelpen, C., Luyten, K., Berbers, Y., Vandewoude, Y. & Coninx, K. (2005). A Component-Based Infrastructure for Pervasive User Interaction. *Proceedings of the International Conference on Software Techniques for Embedded and Pervasive Systems* (pp. 1-16).

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.

Rouvoy, R., Eliassen, F., Floch, J., Hallsteinsen, S. & Stav, E. (2008). Composing Components and Services Using a Planning-Based Adaptation Middleware. *Proceedings of the 7th Symposium on Software Composition SC'08* (pp. 52-67).

Saif U., Pham, H., Paluska, J.M., Waterman, J., Terman, C. & Ward, S. (2003). A Case for Goal-oriented Programming Semantics. *Proceedings of the System Support for Ubiquitous Computing Workshop at Ubicomp'03*.

Sánchez, I., Cortés, M. & Riekki, J. (2007). Controlling Multimedia Players using NFC Enabled mobile phones. *Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia MUM'07*, Vol. 284. (pp.118-124). New York, NY: ACM.

Siebert, J., Cao, J., Zhou, Y., Wang, M. & Raychoudhury, V. (2007). Universal Adaptor: A Novel Approach to Supporting Multi-Protocol Service Discovery in Pervasive Computing. In Kuo T.-W. et al (Eds.), *Proceedings of the International Conference on Embedded and Ubiquitous Computing EUC'07* (pp. 683-693), Lecture Notes In Computer Science, Vol. 4808. Berlin, Heidelberg: Springer-Verlag.

Sousa, J. P., Poladian, V., Garlan, D., Schmerl, B. & Shaw, M. (2006). *Task-Based Adaptation for Ubiquitous Computing*, (Tech. Rep.). Pittsburgh, PA: Carnegie Mellon University, School of Computer Science.

Sousa, J. P., Schmerl, B., Steenkiste, P. & Garlan, D. (2008a). Activity-Oriented Computing, In Mostefaoui, S., Maamar, Z. & Giaglis, G. M. (Eds.), *Advances in Ubiquitous Computing: Future Paradigms and Directions* (pp. 280-315). Hershey, PA: IGI Publishing.

Sousa, J. P., Schmerl, B., Steenkiste, P. & Garlan, D. (2008b). uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. *Proceedings of the 7th IEEE/IFIP Conference on Software Architecture* (pp. 72-80). Washington, DC: IEEE Computer Society.

Vastenburg, M., Keyson, D. & de Ridder, H. (2007). Measuring User Experiences of Prototypical Autonomous Products in a Simulated Home Environment, *International Journal of Human Computer Interaction,* (2), 998-1007.

Weiser, M. (1991). The Computer for the 21st Century. *Scientific American,* 265(3), 94-104.

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.

Yang, Y., Mahon, F., Williams. M.H. & Pfeifer, T. (2006). Context-Aware Dynamic Personalized Service Re-composition in a Pervasive Service Environment. In Ma J. et al (Eds.) *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing UIC'06* (pp. 724-735). Berlin, Heidelberg: Springer-Verlag.

Authors

Oleg Davidyuk received his MSc degree in Information Technology from Lappeenranta University of Technology, Finland, in 2004. He is currently working towards his PhD degree in MediaTeam Oulu Research Group in University of Oulu, Finland. His research interests include application and service composition, user interaction design, middleware and ubiquitous computing. Oleg's publications can be found at www.mediateam.oulu.fi/publications/?search=davidyuk.

Nikolaos Georgantas received his Ph.D. in 2001 in Electrical and Computer Engineering from the National Technical University of Athens. He is currently a researcher at INRIA with the ARLES research project-team. His research interests relate to distributed systems, middleware, ubiquitous computing systems and service and network architectures for telecommunication systems. He is or has been involved in a number of European projects and several industrial collaborations.

Valérie Issarny got her PhD and "Habilitation à diriger des recherches" in computer science from the University of Rennes I, France, in 1991 and 1997 respectively. She currently holds a "Directeur de recherche" position at INRIA. Since 2002, she is the head of the INRIA ARLES research project-team at INRIA-Rocquencourt. Her research interests relate to distributed systems, software engineering, mobile wireless systems, middleware and ubiquitous computing. Further information about Valérie's research interests and her publications can be obtained from http://www-rocq.inria.fr/arles/members/issarny.html.

Jukka Riekki is professor at the University of Oulu, in the Department of Electrical and Information Engineering. He leads together with his colleague the Intelligent Systems Group. His main research interests are in context-aware systems serving people in their everyday environment. Currently he studies in several projects physical user interfaces, context recognition, and service composition. In these projects he cooperates with research groups from China, Japan, and Sweden. He is a member of IEEE.