iCompose: Context-Aware Physical User Interface for Application Composition

Oleg Davidyuk, Ekaterina Gilman, Ivan Sanchez, Jussi Mäkipeltö, Mikko Pyykkönen, Jukka Riekki

Cent. Eur. J. Comp. Sci., (1)4, 2011, pp. 442-465 • DOI: 10.2478/s13537-011-0031-z
Author version

Central European Journal of Computer Science

iCompose: Context-Aware Physical User Interface for Application Composition

Research Article

Oleg Davidyuk*, Ekaterina Gilman, Iván Sánchez Milara, Jussi Mäkipelto, Mikko Pyykkönen and Jukka Riekki

Dept. of Electrical and Information Engineering, and Infotech Oulu,
P.O. Box 4500, University of Oulu, FIN-90014,
Oulu, Finland

Abstract: Ubiquitous computing environments advocate creating applications by plugging together various resources (mobile devices, displays, augmented everyday objects, and so on) and Web Services to support the user's everyday activities and needs. This approach is referred to as application composition, and such applications are called composite. Due to the dynamic nature of ubiquitous environments, application composition has to be supported at runtime, so that the applications are able to adapt to the situation within the environment and other contexts. Application composition is usually performed by autonomous context-aware mechanisms that provide limited or no control for users. Still, users need to be aware of their environments and be able to control and configure applications when they are composed and executed. Towards this goal we present in this article a context-aware application composition system based on the iCompose interface for composing and controlling applications at runtime. Users compose applications by simply touching the resources in the environment with their mobile devices, while the iCompose interface provides feedback and assists users by suggesting possible further actions. The interface relies on a rule-based reasoner and utilizes various context sources to support users dynamically, according to the situation in which they compose applications. We present a complete implementation of the system and report the results of a user evaluation study conducted with 21 participants. This study assesses the issues of control, usability, feasibility and user acceptance of the iCompose interface for context-aware application composition and the prototype as a whole.

Keywords: Ubiquitous computing • interaction design • physical user interface design • application composition

1. Introduction

Recently, we have been witnessing how our everyday environments are gradually becoming ubiquitous. As was envisioned by Mark Weiser in his pioneering work [72], ubiquitous environments are 'future worlds' that consist of multiple resources which integrate into human activities and support users' everyday lives. Naturally, these environments motivate the development of ubiquitous applications which utilize the available local resources (mobile devices, displays, portable players, augmented everyday objects, and so on) and various Web Services at the same time. The Web Services handle the applications' logic, utilize the local resources, and provide the interfaces through which users can interact and control their environment.

* *E-mail: {Oleg.Davidyuk}@ee.oulu.fi*

The approach of creating ubiquitous applications by using various resources and Web Services as building blocks is also known as *application composition* [18]. This approach emphasizes the following three principles: i) combining these resources and Web Services in many different ways to meet the requirements imposed, among others, by application designers, users and the environment, ii) reusing resources and Web Services as much as possible, and iii) reducing the complexity and the costs of application design and maintenance. However, supporting user needs and everyday activities is the main motivation for using application composition in ubiquitous environments. Application composition aims to support this vision by choosing the appropriate set of resources and services and their configuration as dictated by the users and their needs.

The application composition approach brings a number of benefits for ubiquitous applications and environments. First, by composing services across multiple ubiquitous resources, a collection of resources may be able to provide a seamless application functionality that would otherwise not be available. For example, a group of resource-limited devices can share their individual capabilities and provide the so-called virtual device functionality to overcome their physical limitations [61]. Second, application composition can be used to build an application with multimodal user interfaces by combining and controlling input and output from multiple resources. Mungellini et al [38] use this approach to enable rapid prototype development of applications integrating various interaction technologies (object identification, finger localization, tracking, etc.).

In addition, application composition is advantageous, especially if it is context-aware, i.e. it utilizes context[1] to provide relevant information and adapt the set of resources or behavior of composed applications dynamically. Context-awareness in application composition can be provided at application instantiation time and/or at runtime [9]. The first kind of context-awareness refers to selecting the resources and Web Services which depend on the context in which the application is instantiated. Runtime context-awareness refers to the ability to modify (or recompose) the set of resources and/or Web Services while the application is being executed. Both kinds of context-awareness are necessary to achieve additional flexibility, fault-tolerance [63] or the so-called late binding, which means making use of some resources that are available only at runtime [45]. In particular, late binding enables user mobility in the ubiquitous environment. Nevertheless, the main motivation for context-aware application composition is the requirement to support the user's needs and activities. Users are essentially and ultimately the focus of any ubiquitous environment, which always shapes the computation around user needs and how these needs should be achieved. Context-aware application composition supports this vision by creating and adapting applications according to the user's preferences, his/her location, roles, past behavior and other contexts.

In our previous work, we developed the following prototypes with various degrees of system autonomy in application composition. The first prototype for application composition controlled all processes, including the behavior of applications autonomously [15]. Application composition was performed by the engine that used some preprogrammed composition criteria optimize different application characteristics (e.g. QoS). The prototype utilized the user's fidelity requirements (i.e. context) at the application initialization time - runtime composition and adaptation, however, were not supported. Users were only supposed to trigger the composition process and then interact with the resulting application. The second prototype, CADEAU [16], supported manual and semi-autonomic application composition. The semi-autonomic composition reused the engine of the autonomic prototype; although, application composition was controlled through the mixed-initiative interface which guided users through a sequence of steps that resulted in a composed application. The manual composition allowed users to trigger and fully control application composition through a combination of a physical and a visual user interfaces. Although CADEAU performed application composition according to the user's location, this functionality was only supported at the application instantiation time.

The research presented in this paper has the following two goals: First, we aim to produce a system prototype

[1] *Context, at a coarse grain, refers to a collection of information describing various relevant facts, such as the user's location, his/her preferences, the situation in the environment and other facts [1].*

for context-aware application composition that tackles various technological challenges related to multiuser environments, distributed reasoning, context utilization and decision-making to compose and adapt applications. We present the design and implementation of the prototype to show the technical feasibility of our approach and achieve the first goal. Second, we study the user experience due to the user-centric nature of our ubiquitous system. To do so, we develop a context-aware user interface to control application composition and the runtime environment. We achieve the second goal by evaluating our prototype and the interface in a user study that concentrates on user feeling in control, user acceptance and usability issues.

This article reports our findings regarding the design, implementation and evaluation of the system prototype for context-aware application composition. This prototype includes the interface for context-aware application composition and the example application QuizBlasters. The prototype relies on the user's mobile device as a tool for interacting with the application and controlling application composition. iCompose uses a physical browsing interface [67] to interact with ubiquitous resources and uses an asynchronous interaction style [60] and intelligent assistance paradigm [51]. iCompose comprises two interfaces that supplement each other: the physical and the graphical user interface on the mobile device. The physical interface supports physical browsing, i.e. allows applications to be composed by touching the resources in the environment. The graphical interface and the context reasoner implement the intelligent assistance paradigm by providing feedback, visualizing context and assisting users by suggesting possible further actions that users can perform. This prototype utilizes various contexts in order to provide a natural, less obtrusive and easy-to-use interface. The context reasoner dynamically infers the decisions for the user according to the context in which the application is being composed and executed.

The main contribution of this article is the following: 1) the complete implementation of the system prototype for context-aware application composition, 2) the results of the user study with the prototype, which involved 21 participants, and 3) the guidelines for designing user interfaces to control application composition. To date, the authors are not aware of any other fully implemented prototype for context-aware application composition providing similar functionality.

In Section 2 we review the related work on application composition in ubiquitous environments. Then, we introduce the architecture of our system, the iCompose interface, and the example application QuizBlasters in Section 3. Section 4 presents the principles which form the interaction style of the iCompose interface. Section 5 presents the design and implementation of the system prototypes. The user evaluation study of the most recent prototype is described in Section 6. Next, we discuss the main findings of this work and highlight future research directions in Section 7.

2. Related Work

Due to the increasing popularity of application composition in ubiquitous environments, many prototypes and solutions have been introduced in order to study various issues related to application composition. For instance, Chantzara et al [13] and Takemoto et al [65] studied service provisioning issues, Bertolino et al [5] and Butford et al [10] tackled service validation and trust issues, Kalasapur et al [27] targeted optimization of service communication paths, Nakazawa et al [41] focused on automatic generation of application code, Rigole et al [54, 55] addressed distributed user interface deployment, Mungellini et al [38] applied composition to fast-prototype multimodal applications, Sousa et al, Ben Mokhtar et al, Cang et al and Lindenberg et al [4, 12, 32, 63] introduced specification languages for querying and describing application composition, and finally, Sousa et al [64] and Paluska et al [46] introduced design styles for developing adaptive applications through composition.

A number of authors surveyed application composition approaches in ubiquitous computing. Urbeita et al [66] survey the solutions that use Web Services and Semantic Web. Their survey classifies related work according to composition model, specification language, execution model and execution environment. Another survey [25] focuses exclusively on middleware for service composition. The most recent survey [9] considers application composition from the perspective of the main requirements of ubiquitous computing: context-awareness, managing contingencies, support for heterogeneous devices and end-users.

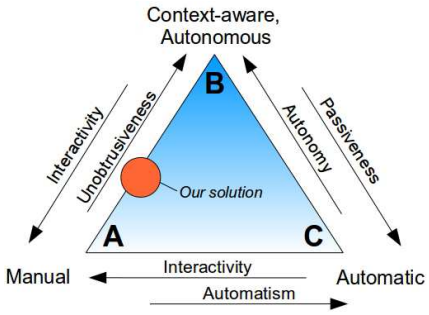


Figure 1: Positioning our solution in a classification of related work.

The work described in this article deals mainly with user control interfaces for context-aware application composition. Therefore, we characterize related work according to the extent of its unobtrusiveness, autonomy and interactivity, as shown in Figure 1. We classify each existing solution according to its two most dominant features, while the position occupied on a side of the triangle denotes the relationship between the features. The corners of the triangle (marked A, B and C) correspond to the following groups: Group A includes the fully interactive composition approaches that perform application composition manually, i.e. by simply executing the commands given by a human operator. They also support the runtime adaptation of composed applications by executing the users' commands. Group B represents the autonomous and context-aware solutions that perform application composition independently from start to finish without any user intervention. These solutions employ various mechanisms for context acquisition, recognition and reasoning in order to determine when and how to compose and adapt applications. These solutions are characterized by unobtrusiveness and assume that interaction between the system and the users is unnecessary. User control, however, can be provided though context utilization mechanisms (i.e. non-interactively). Group C corresponds to those application composition solutions that are automatic and non-interactive. These solutions aim to replace various routine processes such as service discovery, service matching, optimizing application allocation, and so on. Although these solutions are automated as they require no guidance from human users to perform these processes, they remain dependent on users as such solutions are not able to determine when application composition has to be started or finished. Moreover, the support for runtime application composition and adaptation is limited in this group. We discuss these three groups in detail below.

Group A. Related work in this group supports the idea that the system's role is only to provide some means of user interaction (e.g. a physical or graphical user interface), through which users can specify the structure and the functionality of their application. Manual composition assumes support both at the time of application instantiation and runtime, so that users can adapt the composed applications as they need. The majority of existing solutions in this group focus on application composition in the home network domain [14, 22, 42, 43, 50]. Manual application composition in the museum domain has been suggested by Ghiani et al [20]. Several solutions, focusing on the end-user application composition, allow users to install and deploy ubiquitous resources [28, 34] and then to develop simple applications by composing these resources using an editor [34] or by manipulating smart cards associated with the deployed resources [28]. Similarly, composing applications from ubiquitous artifacts (i.e. from real objects manually converted into ubiquitous) is emphasized in the Memodules project [39]. Sanchez et al [**?**] present a conceptual physical interface for composing applications from visual resources by touching NFC tags attached to these resources. A similar approach is presented in our previous work on application composition [16].

Group B. The composition approaches in this category utilize various context-aware mechanisms to realize autonomous application composition and adaptation. We further classify these solutions into i) advanced context-aware systems, and ii) solutions with basic support for context-awareness. The first ones are more elaborate systems utilizing a wide range of context types and providing runtime context provisioning, context monitoring, service discovery, context reasoning and other functionalities. The second ones focus on formal aspects of context-

aware application composition, in particular, on context modeling, specification of context-aware resources and context processing algorithms. Example solutions with advanced context-aware support are presented in [7, 24, 58, 75]. Bottaro et al [7] developed a system for autonomous policy-based application composition and adaptation which targets late-binding and user mobility issues. Late-binding and self-adaptation are the motivation behind MUSIC [58], a middleware for planning-based application composition. Focusing on utilization of the context provided by mobile devices, Hesselman et al [24] presented the composition approach that is tightly coupled with a service discovery and rule-based inference engine to interpret context and match discovered devices. In contrast, the approach of Zhou et al [75] uses the rule-based reasoner to implement application logic and decision-making for application composition. Their solution targets the composition of pervasive Web Services. Examples of composition approaches focusing on formal aspects in context-aware composition are presented in [27, 37, 47, 73, 74]. The solutions of Ben Mokhtar et al [37], Preuveneers & Berbers [47] and Zhang et al [74] address resource-awareness, late-binding and user mobility and perform dynamic application composition and adaptation using a semantic-based matching engine. Concentrating exclusively on user mobility, PICO middleware [27], supports application composition and adaptation through semantic and syntactic matching. In contrast, the Diadalos architecture [73] targets personalization issues in application composition and includes multiple services for collecting and processing this kind of context.

Group C. Systems in this category aim to replace human manual processes in order to minimize user distraction during application composition, configuration, management and other tasks. Although some of these solutions may even utilize user preferences and profiles (e.g. [40]), these systems lack support for context provisioning, monitoring, reasoning and other essential context-aware functionalities. Most research in this category deals with activity-oriented (or task-based) computing[2] These systems assume that application developers (or users themselves) create abstract templates where they specify a set of service categories and their properties (i.e. functionality) which they need to achieve some user's task (activity). Then the system uses automated mechanisms to discover and choose appropriate services whose functionality matches the specified task template. After this, the services chosen are assigned to this task and the system configures the resulting application. Although some solutions provide user interfaces for customizing the predefined task templates to match user needs [33, 36, 63], a typical task-based system rarely supports end-users. Instead, task templates are created by application developers at design time as they are described in a computer-readable form [3, 4, 17, 63]. Application composition is typically carried out by semantically and syntactically matching the original task template according to some specified criteria and a matching (or planning) algorithm. Issues related to semantic and syntactic matching for application composition have been studied, particularly, by Ben Mokhtar et al [4]. Composition based on planning algorithms has been proposed among others by Beauche & Poizat [3], Ranganathan & Campbell [49] and Sousa et al [63], as well as in our previous work [17].

We associate our current work with manual and context-aware composition application, as shown in Figure 1. Indeed, the iCompose interface allows users to choose ubiquitous resources by touching while utilizing various contexts to determine when and how to perform application composition and adaptation. To date, we are not aware of any other prototype for application composition which combines manual and context-aware composition at the same time.

The work presented in this paper is also associated with research on distributed and mobile reasoning approaches. Examples of these approaches were introduced, e.g. in [6, 48, 57, 70]. Viterbro et al [70] suggested a two-tier rule-based reasoning approach for mobile devices which supports client-side reasoning, server-side reasoning and hybrid reasoning scenarios. Przybilski [48] developed a distributed reasoning mechanism which targets mobile devices with limited resources and uses the P2P protocol for discovery and communication. Przybilski's solution focuses on scenarios that require reusing and sharing context information among several mobile devices and a remote server. Bikakis & Antoniou [6] suggested a theoretical model and an algorithm for distributed reasoning

[2] *Both terms are used interchangeably in related work.*

in a P2P system. This approach deals with conflict resolution issues in a multi-context system. Rizou et al [57] introduced an architecture for distributed reasoning in large-scale overlay networks. Their approach relies on the optimization algorithm that takes an abstract reasoning task as input and calculates a placement plan to adapt the reasoning subtasks (operators) to the network conditions. Unlike our reasoning mechanism, this approach does not support reasoning for mobile devices.

Our work is also related to research on physical browsing, as our system relies on touching. Generally, physical browsing refers to an interaction technology for associating physical objects and digital information. In the ubiquitous computing domain, a physical browsing interface is usually based on a mobile device and allows resources to be chosen in the environment by means of scanning, pointing or touching [59, 67]. For instance,the PERCI framework [8] integrates physical browsing interfaces with Web Services and addresses interoperability and multi-tag interaction issues. We do not target these issues in our work. Ailisto et al [2] presented findings from the field studies of two NFC-based applications: a catering service for the elderly and an access control service developed for the city of Oulu. They analyze user experience, security issues and business models for NFC-based applications. A general framework for activating Web Services in ubiquitous spaces by means of a physical user interface was presented by Riekki et al [52, 53]. Although Riekki et al's work mainly addresses graphical design of NFC icons, they also target middleware design, knowledge representation for the data stored on tags and other issues. Their recent work [**?**] introduces the interaction model and a conceptual physical interface for composing applications. Although in our current work we focus on design and user experience issues, some of our solutions (e.g. the visual design of iCompose interface) were inspired by the ideas and concepts presented in [**?**].

At a coarser grain, our research is related to the issue of balancing control between users and the system. This was addressed by Vastenburg et al [69] in their user study that analysed user willingness to delegate control to a proactive smart-home application. Vastenburg et al developed a user interface for the atmosphere control application which has three modes of interactivity: manual, semi-automatic and automatic. However, the behavior of their system was 'wizard-of-oz', that is, the reactions of the system were remotely activated by a human observer during the experiment. Vanderhulst et al [68] presented a Semantic meta-user interface for runtime user control of task-based applications through which users can switch between tasks, manage their execution and perform 'personalized' service discovery. The issue of balancing user control and autonomy was also studied by Hardian et al [23] who suggested ensuring that the users felt in control by explicitly exposing the system's logic and the context used in the application adaptations through a user interface. Design principles for intelligibility and control in context-aware systems were presented and analyzed in [19]. We also studied these issues in our previous work on application composition [16]. In particular, we compared three interaction techniques for application composition and studied the context and the factors which contribute to the users' decision to prefer one interaction technique rather than another.

3. Overview and System Architecture

Our system architecture is intended for ubiquitous environments, which we view as physical spaces populated with multiple resources where multiple applications are being executed. The applications range from single-user to multiuser ones. Applications can be composed of one or many resources and Web Services at the same time. Our architecture supports resources that provide multimedia, computational or other capabilities. These resources are used and controlled by Web Services. The resources are augmented with visual icons and NFC tags placed under the icons. Each tag contains the data which uniquely identifies the resource in question. The same set of resources and tags is capable of serving the needs of multiple applications and users in the environment. In addition, the architecture supports the resources offered by mobile devices which provide user input/output capabilities and various contexts. Users carry these mobile devices which act as interaction tools connecting the users and the environment and allowing them to use these resources in applications.

Our architecture supports context-aware applications that utilize the following types of context: i) user-specific, ii) application-specific, and iii) environmental context. The first type describes the user's personal profile, schedule,

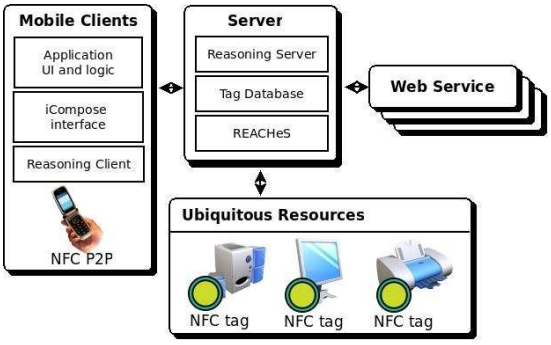


Figure 2: Architecture of the system.

presence, physical location, proximity to nearby users and different preferences. The application-specific context, in its turn, describes the application state and various application-related events that the user generates by touching NFC tags. Finally, the environmental context specifies the state of the environment, i.e. the availability of ubiquitous resources. All these contexts are utilized by the system for application composition and by applications for building their logic.

At a coarse grain, the architecture consists of mobile clients, Web Services, ubiquitous resources and the server interconnecting all three, as shown in Figure 2. The server includes the reasoning server, the NFC tag database and the components of the REACHeS infrastructure [53] that implement the Web Services remote control and communication functionalities. The reasoner supports decision-making processes which utilize various types of context. The tag database stores the data which is necessary to enable the interaction with NFC tags and resource management. This data associates each NFC tag with i) the concrete resource instance, ii) the physical location of the tag, and iii) an optional user command which the system executes when this tag is touched. The Web Services enable the control and communication with resources, implement application logic, provide access to the data used in applications and supply the environmental context. The mobile clients are central to the architecture as they enable the interaction between users, resources and the server and provide the user-specific context. The mobile clients include the iCompose interface, the reasoner client and applications which we explain next.

iCompose interface. This component is the main element through which the user communicates with the system. iCompose consists of numerous other components enabling the traditional graphical user interface (GUI) on the mobile device and the physical interface through which the users interact with the environment. The physical interface relies on the NFC tags in the environment and mobile devices equipped with NFC readers. This component has two responsibilities. First, the GUI visualizes the context and the events related to application composition which provide necessary feedback so that the user is aware of the situation at hand. Second, iCompose offers the means to control application composition. Users either choose commands using the GUI or make choices using the physical interface. The physical interface supports two kind of actions: touching an NFC tag and touching two mobile devices together (as shown in Figure 3).

(a) (b) (c)

Figure 3: User making a handshake (a), interacting with a wall display (b) and a tag (c).

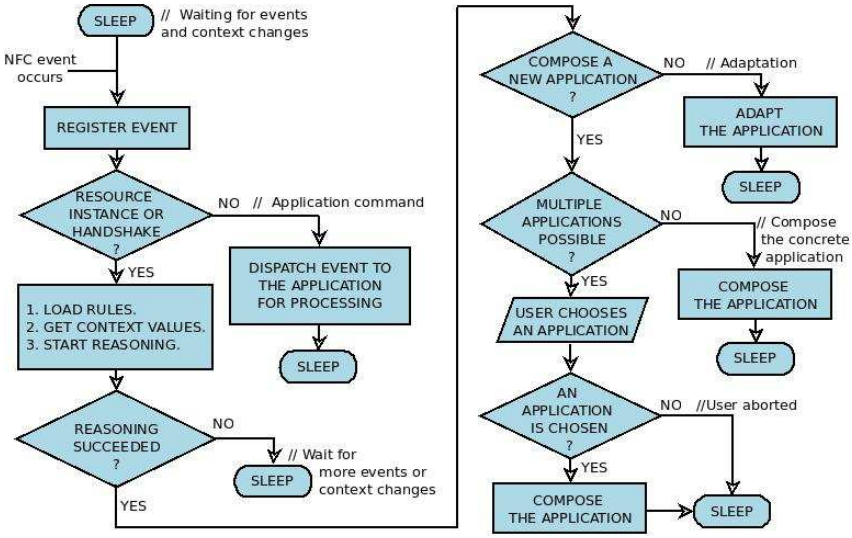


Figure 4: Application composition process.

The latter action is called a handshake; it uses NFC P2P communication protocol. Users touch NFC tags to interact with resources and to issue various commands for applications or the system. Users make handshakes to interact with each other, i.e. to exchange information or issue commands.

The main role of the iCompose interface is to provide user control for the application composition process that triggers composition and adapts the applications according to context and reasoning rules. An instance of this process runs in the background of every mobile client. Users can control this process through the iCompose interface at any time. This process works as follows (see Figure 4): Any of the user's actions, such as touching a tag or making a handshake, wakes up the application composition process and generates an event which is registered and visualized by the iCompose interface. The event identifies either an application-specific command, a resource or a handshake. In the first case, the command is dispatched to the application running in the foreground of the mobile client. Then, this application processes and executes the command as specified by its logic. In the second case, the event triggers the loading of the reasoning rules, obtaining necessary context values and, finally, the reasoning process. This process then tries to determine the required operation (i.e. composition or adaptation) and the concrete application which has to utilize the resources the user has just chosen. If the reasoning process succeeds and the result points to a single application, then, the system adapts or composes and executes the application accordingly. In contrast, if the reasoning result is ambiguous and points to a set of applications, then, the system invokes a user dialog. The user is suggested i) to choose among the alternative applications, or ii) to continue choosing the resources using the physical interface. If the user chooses the concrete application, this application is immediately composed and executed. If the reasoning process fails, it is most likely due to the fact that the available context is insufficient to infer a decision. In this case, the application composition process goes to sleep until the user makes further actions through the physical interface or the system detects context changes. At any time, the user can interrupt the whole composition process. When the reasoner commands composing an application, the chosen resources and Web Services are assigned for the user and the application using the REACHeS resource management. The application starts executing from its main method.

The Reasoner. The role of the reasoner is to support the decision-making processes utilizing various contexts. Although this component is mainly used for the application composition process, the reasoner can also be used by applications to implement their application logic. The reasoner consists of the reasoning server and multiple reasoning clients deployed onto mobile clients. The reasoner supports a distributed reasoning mechanism, which implies that the computation (i.e. reasoning) can take place on the mobile device ('local reasoning') or on the server ('remote reasoning') or on both sides, partially. At the cost of a greater implementation complexity, such

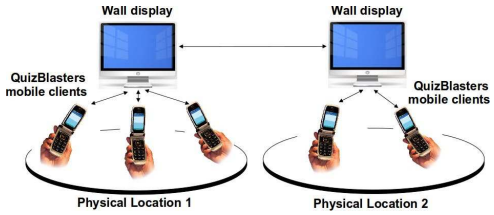


Figure 5: QuizBlasters application composition schema.

a mechanism has two distinct advantages: i) it makes it possible to comply with the users' privacy preferences, and ii) ensures a greater overall performance of the decision-making process. The first advantage implies that the reasoner can process the user's sensitive context information (e.g. his/her physical location) on the mobile device controlled by the user, i.e. using local reasoning. The second one implies that context can be processed at the spot where it is collected. That is, the unnecessary communication overhead is reduced as only the reasoning result will be sent over the network.

The QuizBlasters application. In order to demonstrate the functionality of our system, we present the QuizBlasters learning application which we implemented and later used for the user evaluation study. QuizBlasters combines elements of a treasure-hunt and a multiplayer action game. The application is context-aware and is meant for daily use over a long period of time (e.g. during a semester). It has two modes, the learning mode for a single user (default mode) and the multiplayer action mode called 'Scorch'. The learning mode motivates users to collect and answer as many different learning tasks (i.e. quizzes) as possible by touching the NFC tags placed in the environment. Users need to complete these quizzes in a timely manner in order to make progress and unlock various virtual bonuses. Scorch, in its turn, is the application mode in which the players challenge each other by playing the action game on a wall display. In this game, the players utilize the bonuses and weapons which they earn by completing quizzes in the learning mode. This game resembles the classical Scorched Earth game[3] where multiple players have to destroy each other's avatars using various weapons. In our application, users control their avatars on a wall display using their mobile devices as the remote controllers. In order to play Scorch, users need to compose it from two or more mobile devices and a wall display. Moreover, Scorch can be played in different physical locations: if the players decide to play on the wall displays located in separate rooms, as shown in Figure 5. QuizBlasters utilizes different kinds of context in order to ensure a pleasurable user experience and also to add variability to the gameplay. In addition, this context is used to specify the 'rules of engagement', i.e. when, where and with whom, the players can compose and play Scorch. These rules attempt to enforce a certain behavior and order (e.g. restrict undesirable usage of certain resources, restrict gaming during classes, etc.). QuizBlasters utilizes players' physical location, study performance, daily schedule, proximity to other players, personal friend lists, and so on. NFC tags, NFC-enabled mobile devices and wall displays are the platform for this application.

4. Interaction Model

Our system aims to enable application composition in order to support users' daily needs and activities. We choose mobile devices as the application platform because these devices are truly ubiquitous, can be carried everywhere, and thus, are always available. In contrast to other application composition systems, our system is user-centric and focuses essentially on interaction between users and the environment. Hence, finding a suitable interaction model matching our needs is a major research and development challenge. An interaction model not only acts as the primary guiding force in the user interface design, but also provides the organizing principles for the underlying software design.

[3] *http://en.wikipedia.org/wiki/Scorched_Earth_(video_game)*, *accessed in August 2011*

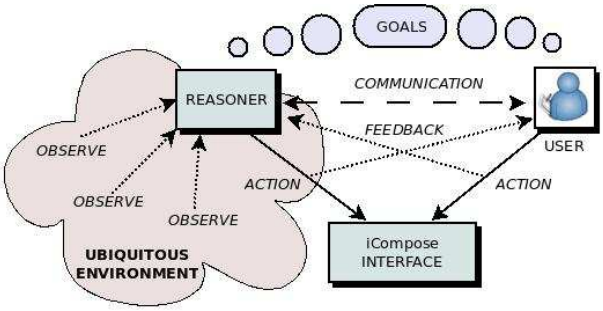


Figure 6: The intelligent assistance interaction paradigm used in iCompose (adapted from [51]).

The main goal of our interaction model is to offer a natural, easy-to-use and flexible user interface which users will trust and feel they have control over, even when composing applications that handle personal and sensitive data. In line with this goal, we specify the following three requirements for the interaction model: i) the primary interaction technique with the ubiquitous resources is physical browsing, ii) the interaction style has to be asynchronous, and iii) the interaction mode uses the so-called intelligent assistance paradigm. We justify these requirements as follows below.

Physical Browsing. As defined by Broll et al [8], physical browsing 'takes advantage of mobile devices that physically interact with tagged objects to facilitate interaction with associated information and services. In general, physical browsing offers the three interaction techniques which are touching (using NFC technology), pointing (using recognition of visual markers) and scanning (using short range wireless technologies, e.g. Bluetooth or wifi). Physical browsing is preferred for selecting resources in ubiquitous environments because of its directness, simplicity and natural characteristics [2, 8, 52, 59, 67].

Another important reason for applying physical browsing is the fact that it paves the way for building user interfaces according to the direct manipulation paradigm [62], which is currently the prevailing design style for HCI. This style is crucial for establishing trust and feeling in control between users and the ubiquitous environment by facilitating the notion of users performing a task or activity themselves, i.e. directly, not through an intermediary like a computer. Further details on how the issues of trust and feeling in control affect user acceptance and usability of ubiquitous systems are presented in our earlier work [15, 16].

Asynchronous Interaction. This interaction style means that the application composition process is blended into users' other daily activities so that users trigger this process spontaneously, whenever it is convenient for them or whenever an opportunity arises. Existing asynchronous interaction examples support such routine daily activities as email, cooperative groupwork (e.g. learning [31, 35]), online conferencing [71], and even pervasive gaming [60]. We argue that application composition will also become one of these routine activities. We also argue that this is the preferred style for interaction in ubiquitous environments where it usually spans across time and space. In other words, resources are physically scattered over the ubiquitous environment; thus, a single point of interaction is often nonexistent. That interaction spans across time due to the fact that, in the course of their daily lives, users simultaneously interact with multiple applications, i.e. switching them between running and paused states on mobile devices and engage multiple resources.

Intelligent Assistance. Intelligent assistance is the interaction paradigm which specifies the relationship between a user and the system collaborating together and sharing the same task and goal, as shown in Figure 6. One distinguishing characteristic of this paradigm is that the user side possesses most of the knowledge and takes the initiative, while the system side assists by negotiating about goals and how to achieve them [51].

The requirement to use the intelligent assistance paradigm is based on the findings of our earlier research on application composition [15, 16], where we compared different interaction paradigms, spanning from automatic to manual composition in order to find a balance between user control and the system autonomy. According to the users, who also participated in our experiments, the most promising interaction paradigm is the one which unites

Figure 7: The low fidelity prototype.

the advantages of both the autonomic and the manual composition. That is, a user can select some resources s(he) wishes to use with the application. The missing resources are assigned with the assistance of the system, which also controls application execution and adaptation. This paradigm allows users to decide which are the most important decisions and make them by themselves, while leaving less important decisions to the system. The advantages of this paradigm are flexibility and the ability to provide users with the feeling of control.

5. Design of the Prototypes

We have designed our system in two iterations. First, we created a low fidelity prototype which we evaluated in a small-scale user experiment. Then, we used the results of this experiment to design and implement the high-fidelity system's prototype, which we evaluated in a large-scale user experience study (reported in Section 6).

5.1. Initial Prototype

In this design iteration, we aimed to study users' usual interaction flows in the multiuser environment and determine the level of detail the iCompose interface needs to reveal to users. The multiuser interaction in ubiquitous environments has much more complex flows than the traditional, single user interaction due to unpredictability in the behavior of a human group. Determining the level of detail was essential for designing the GUI, which must be comprehensible and provide enough detail for users through, e.g. popup messages or visual elements, so that the users are always aware of the situation.

We partly created the low fidelity prototype of paper and carton (as presented in Figure 7) and partly implemented it on Nokia 6313 mobile phones. The implemented functionality included accessing and answering quizzes and joining and playing the multiplayer game. We evaluated this initial prototype in a user experiment involving six IT experts. The findings of this initial experiment led us to formulate the design guidelines for the high-fidelity prototype. The goal of these guidelines is to ensure that the high-fidelity prototype is a pleasurable-to-use software product which creates a desirable feeling of being in control while supporting the user with a sufficient level of information details. These guidelines are the following:

Explicit and timely feedback. The system must always provide some feedback in response to a user's action made through either the physical or the traditional interface. This feedback has to be timely (i.e. to follow the user's action immediately) and explicit (i.e. obvious for the user to notice it). Feedback can be provided in

the form of popup messages, choice dialogues, status updates and notifications through graphical elements. In addition to this, the feedback must also be provided using sound and vibrating alerts for the cases when the user's actions are made through the physical interface or when the user does not anticipate that he or she needs to react in any way (i.e. by making a decision or a choice). We have observed that, in these cases, the user's attention is often concentrated elsewhere than on the mobile phone's GUI.

Providing maximum user control. This issue is the major guideline motivating the design of the interaction style in our system. We have revealed the following three issues related to user control: i) the system's behavior has to be clear and unambiguous for the users, ii) the users should be given the means to control the choices made (or suggested) by the system, and iii) the users should be given the means to achieve their goals also without the system assistance (we called this the 'skip and continue' interaction style). The first issue implies that the system should provide explanations (or details) about the system actions that are going to happen next. We observed that surprises in the system's behavior, both positive and negative ones, reduce the user's feeling of being in control. The second issue entails that the users usually want to choose and confirm explicitly the decisions made (or recommended) by the system on behalf of them. The 'skip and continue' interaction style, in its turn, implies that the user has to be given an option to alter the system's behavior at any point and still achieve his/her goals using some alternative way, i.e. manually.

Privacy issues. These issues came to light since our system utilizes the user's context. However, our users were concerned mainly about privacy in the situations involving other participants, in particular, when registering handshakes and sending messages in the QuizBlasters application. The users requested to have a kind of privacy control page like in Facebook, so that the users themselves can specify the social groups of people who are permitted to interact with them.

We applied these guidelines when designing the iCompose interface and the high fidelity prototype during the next design iteration.

5.2. High Fidelity Prototype

We implemented the high fidelity prototype, including the iCompose interface and the Quizblasters application during the second design iteration as follows:

iCompose and QuizBlasters. The prototype relies on Nokia 6131 NFC-enabled mobile devices as the platform for implementing the mobile client. These devices are capable of reading ISO/IEC 14443-A RFID tags (Mifire 1k type) which we use as resource identifiers. The data is stored in the memory of these tags as NDEF-type messages that may include multiple NDEF records [44]. Each record contains a set of NDEF flags and a variable payload. In our prototype, the payload is an ASCII string which encodes a pair consisting of a parameter name and the corresponding parameter value. The parameters describe the events that are generated by the physical interface. These events are used in the communication protocol between mobile clients and the REACHeS Server (see Figure 8). The events are generated when touching a tag or making a handshake with an NFC-enabled mobile device.

iCompose and the QuizBlasters application are implemented as a set of J2ME Midlets which communicate with the corresponding server components using RESTful HTTP (see Figure 8). We use XML and JSON [26] as the HTTP payload format and kXML2 library [30] for parsing XML messages. The iCompose component and the application are implemented as shown in Figure 9. We implemented each user interface screen and some complex user interface elements as separate classes. Their design followed the Model-View-Controller pattern, which is currently the prevailing design pattern used in application development. Figure 10 shows some user interface screenshots of the iCompose interface and the application.

The Reasoner. The reasoner consists of the reasoning client and the reasoning server. The reasoning client is implemented in J2ME and deployed on mobile phones as the part of the mobile client. The reasoning client is based on the logic programming approach and uses the m-Prolog inference engine optimized for J2ME applications [29]. The class diagram of the reasoning client is shown in Figure 11. The reasoning client is notified about events

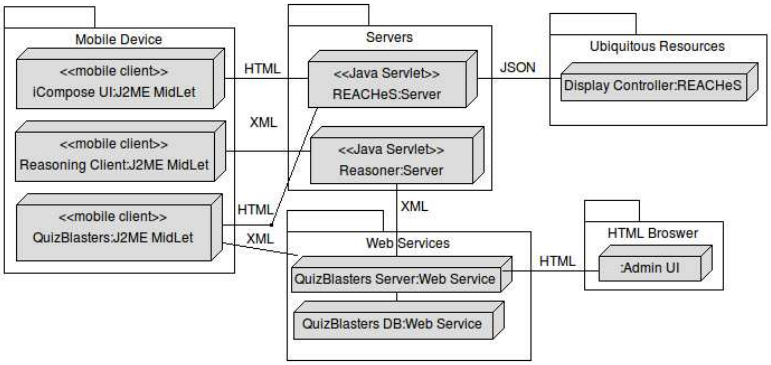


Figure 8: UML Deployment diagram of the system's prototype.

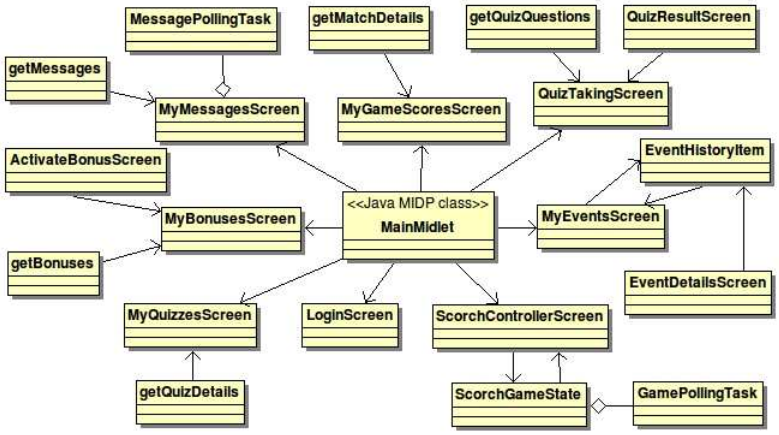


Figure 9: UML Class diagram of the mobile client (simplified).

through the ReasonerListener, which assigns the events to the Queue. The EventHandler processes the events from the Queue in first-in, first-out (FIFO) fashion, and, depending on the event type, the EventHandler invokes the corresponding task (e.g. the DisplayTagEventHandler). Then, this task triggers the InferenceEngine, which first updates the knowledge base with the event details and then infers a decision. The InferenceEngine may request additional context information in order to process a specific event. For example, the ReachesEventsGetter is invoked to obtain the availability status of a display resource if the user attempts to compose or adapt an application using this particular display. The knowledge base consists of Prolog facts and rules which specify the conditions necessary for composing and adapting an application. Optionally, the rules can constitute a part or even entire application logic. For example, Figure 12 shows a snippet of rules corresponding to the QuizBlasters' application logic. The reasoner loads the knowledge base and starts inferring in a backward-chaining manner. In other words, the reasoner determines whether facts exist (i.e. context values) which support the assumed conclusions. The reasoning result is returned to the component invoking the reasoning process. In this example, the result is returned to the QuizBlasters application.

The reasoning server is implemented in Java SE as a RESTful Web Service and deployed on the Glassfish application server. The reasoning server communicates with the reasoning clients using XML messages sent as HTTP GET/POST requests (see Table 1 for details). The reasoning server infers decisions when multiple interacting users attempt to trigger application composition or adaptation (e.g. when new players try to join an existing application instance). In such a situation, the reasoning client of the user who made a handshake or touched a tag triggers the reasoning server to process this event. Then, the reasoning server processes the event by inferring

Figure 10: The main menu (a), event collector (b), event collector during a quiz dialog (c) and Scorch remote controller screen (d).

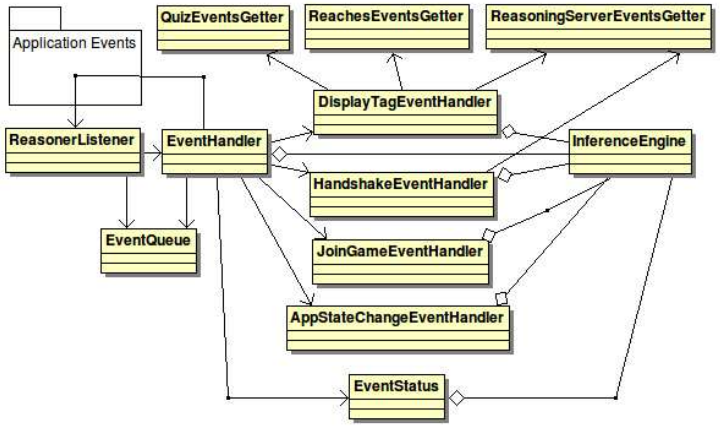


Figure 11: UML Class diagram of the reasoning client (simplified).

a decision and communicates it to the reasoning clients of all the users invovled in this event. The main part of the reasoning server's API is presented in Table 1.

The QuizBlasters server components and the Tag Database are implemented using Ruby on Rails. These components communicate using RESTful HTTP and exchange data which is structured in XML and JSON formats.

Our prototype relies on the REACHeS infrastructure which is responsible for managing ubiquitous resources and supporting communication between resources, Web Services and mobile clients. Due to space limitations, we do not discuss REACHeS in this article, for further details, the reader may refer to [53].

```
result(join_game):-application_activity(idle),
                   display(reserved),application(game),
                   calendar_activity(idle), sameclass(true)
result(wrong_game_class):-application_activity(idle),
                          display(reserved),application(game),
                          calendar_activity(idle),
                          sameclass(false)
result(busy_statistics_message):-application_activity(idle),
                                 display(reserved),
                                 application(statistics)
result(abandon_game):-application_activity(game)
result(abandon_quiz):-application_activity(quiz)
```

Figure 12: Example of the QuizBlasters application rules (m-Prolog).

Request	URI Path Template	Response (HTTP)	Description
POST	.../handshake	Success: 202 Accepted and Location (URL header), Failure: 500 Internal Server Error	Triggers the reasoner server to process a handshake event. Location header contains URL to redirect the client for completion of the request. Next method presents the decrypted URL.
GET	.../handshake/{handshakeId}/userName-/{userName}	Success: 200 OK and XML Message (payload), Failure: 500 Internal Server Error	Retrieves and returns the reasoning result.
GET	.../gameConfirmation/{userName}/{screenId}/{gameId}	Success: 200 OK and XML Message (payload), Failure: 500 Internal Server Error	Updates the application status when a handshake event occurs.

Table 1: The reasoner server's API. Curly brackets denote variables.

Group	Gender		Age		
	M	F	≤25 y.o.	26-30 y.o.	30+ y.o.
(A) Novice users	25%	75%	58%	25%	17%
(B) Advanced users	67%	33%	45%	33%	22%

Table 2: Demography of the user study.

6. User Study and Evaluation

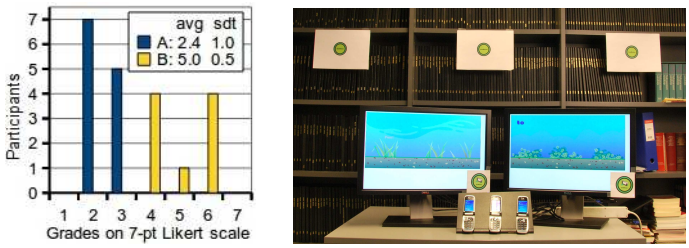
In order to demonstrate the technical feasibility of our system and evaluate the prototype's interaction design, we carried out a user experiment with the fully implemented prototype and the QuizBlasters application. The goal of this user experiment was to make an assessment of the system and the issues related to the application composition process.

Evaluating a system is always a challenge. On the one hand, a user evaluation requires using concrete applications to demonstrate the system's functionality. On the other hand, introducing applications might force test participants to focus on evaluating applications instead. One way to address this problem is to design questions for questionnaires and face-to-face interviews which ask participants not only about the application, but about the system and its features. Next, we explain the methodology and our approach for collecting feedback in detail.

6.1. Methodology of the user study

We invited 21 individuals from the local community of the City of Oulu (Finland) to participate in the user study. They were selected according to their background and experience with mobile technologies so that they formed two focus groups representing novice (non-technical) users and advanced users, correspondingly (see Table 2). The user's personal experience with mobile technologies is shown in Figure 13 (a). The users from the novice group were dominantly young female students studying biology, economics, medicine and social sciences. They had minimal experience with mobile technologies and none of them had any technical background. In particular, 75% of them never or very rarely use mobile phones beyond texting or calling. The second group consisted of users who had advanced skills in mobile technologies, including several IT experts working in the ubicomp domain. These users described themselves as pro-technology enthusiasts who eagerly try new mobile applications and technologies. In addition, 66% of them reported using GPS, Bluetooth and wifi on a daily basis. All the participants in the test tried our prototype for the first time.

The experiment took place at the University of Oulu. The test setup consisted of several displays and NFC tags which were placed on the walls, as shown in Figure 13 (b). Test participants arrived in small groups of two or three persons for a session which lasted approximately 40 minutes. The participants were given a short demonstration of the NFC technology and an overview of the functionality of the prototype. Then, each group had to perform the following two tasks several times. At first, the users had to collect quizzes and bonuses from



(a) (b)

Figure 13: Left: Experience of user groups (A=novice, B=advanced), Right: the user evaluation test setup.

Figure 14: Participants of the user study interacting with the prototype and each other.

NFC tags and answer them. In order to complete the second task, the users had to compose an instance of Scorch using several resources and players. The users also had to utilize various bonuses during the gameplay. The users were encouraged to ask questions, give comments and think out loud during the experiment. After the tasks had been completed, we asked them to fill in an anonymous questionnaire and then discuss their experience with us. The questionnaires were used to collect formal subjective feedback, while the informal interviews allowed the users to express their opinions about the system prototype and suggest ways of improving it.

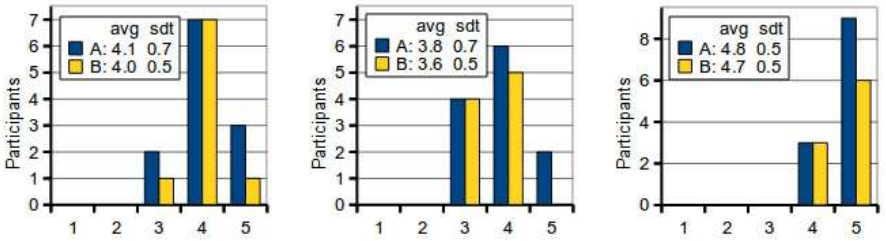
6.2. Results of the user study

Figure 14 shows some snapshots taken during the user evaluation experiment. The prototype demonstrated a stable performance during the experiment and all the participants completed the given tasks successfully. We consider that this together with the faultless deployment of the system confirms the feasibility of our approach from a technical point of view.

As we observed during the experiment, the advanced user group finished the tasks slightly faster than the other users. This did not come as a surprise since we had already noticed that NFC technology causes longer learning curves for inexperienced and novice users in our earlier work [16]. We measured the quantitative results of the questionnaires, in which the test users had to rate a number of statements using a 5-pt Likert scale (1 = totally disagree and 5 = totally agree). We arranged the results of the user study into the three categories as follows:

Usability. Usability was measured using a questionnaire with five propositions, focusing on ease-of-use, intuitiveness, physical effort required and speed. These propositions were, e.g. 'the interface required apparent physical effort to start an instance of the game'. Figures 15, 16a and 16b show the resulting graphs. Although the results given by both groups followed each other hand in hand, we observed that the novice users tended to give slightly higher grades. Another interesting result was the fact that the participants evaluated the physical effort required as 'only slight', despite the need to change position and sometimes even to walk when composing QuizBlasters. This was contrary to our expectations as we had anticipated that the users would find the action of touching tags and making handshakes to be a physically demanding task. Overall, users from both groups gave positive remarks on the idea of connecting a game partner by making a handshake. Called 'human service discovery',

this approach dominates in usability over classical search techniques in ubiquitous environments, as confirmed by Lindenberg et al [32].



(a) (b) (c)

Figure 15: Easiness (a), intuitiveness (b) and physical effort (c) as expressed using the 5-point Likert scale (1=min, 5=max).

Control. Feeling in control (also known as locus of control) was evaluated using the questionnaire with the propositions targeting trust and confidence in system behavior. There was a significant difference in confidence scores between the two user groups, as the confidence scores given by the novice group were on average eleven per cent lower than the scores of the advanced user group (Figure 16c). This might be explained by the fact that the novice users were more conservative and less oriented towards technologies, unlike the group made up of technology enthusiasts. Hence, the novice users expressed a more cautious opinion towards our prototype. The issues of trust did not lead to a significant difference and the results of both user groups were hand in hand (see Figure 16d). We expected that our participants would feel in control due to the direct manipulation paradigm employed in the user interface design. As stated by Shneiderman & Plaisant [62], this paradigm 'allows users gaining confidence and mastery because they are the initiators of action, they feel in control, and they can predict the interface's responses'.

Feasibility and Acceptance. These issues were evaluated both using a questionnaire (i.e. predictability and usefulness) and through interviews following the experiment. The subjective results are shown in Figures 16e and 16f. As can be seen, most test users found the commands and menu choices recommended by the interface predictable. The behavior of the system was described 'as expected', this is supported by the fact that some test users were surprised during the interviews when they were told that the system was actually context-aware. One participant suggested designing the interaction between mobile phones (i.e. handshakes) similarly to bluetooth pairing. In his view, users need to explicitly initiate a handshake for the first time only, so that later, the connection between these two mobile devices will be established automatically, if the same pair of users needs to interact in the same context. Following this approach, some sort of a 'permanent handshake' is established between certain users. Another user suggested a new feature for the event collector interface, so that the interface indicates (i.e. prompts) possible user actions give the current context. That is, the interface can highlight particular NFC tags that the user may need to touch in order to complete the composition/adaptation task. This feature looks feasible, especially if the recommending mechanism relied on the user's context and history of his/her past choices. Surprisingly, perhaps, was the fact that none of test users was concerned about privacy and trust issues linked with the usage of NFC in our prototype. This is probably due to the nature of the application utilized in this experiment: the ubiquitous game. A different application would probably require additional effort to design a privacy protecting mechanism.

7. Discussion and Future Work

This article presented a context-aware physical interface for composing applications in ubiquitous environments which we implemented in combination with a prototype of an application composition system and the Quizblasters

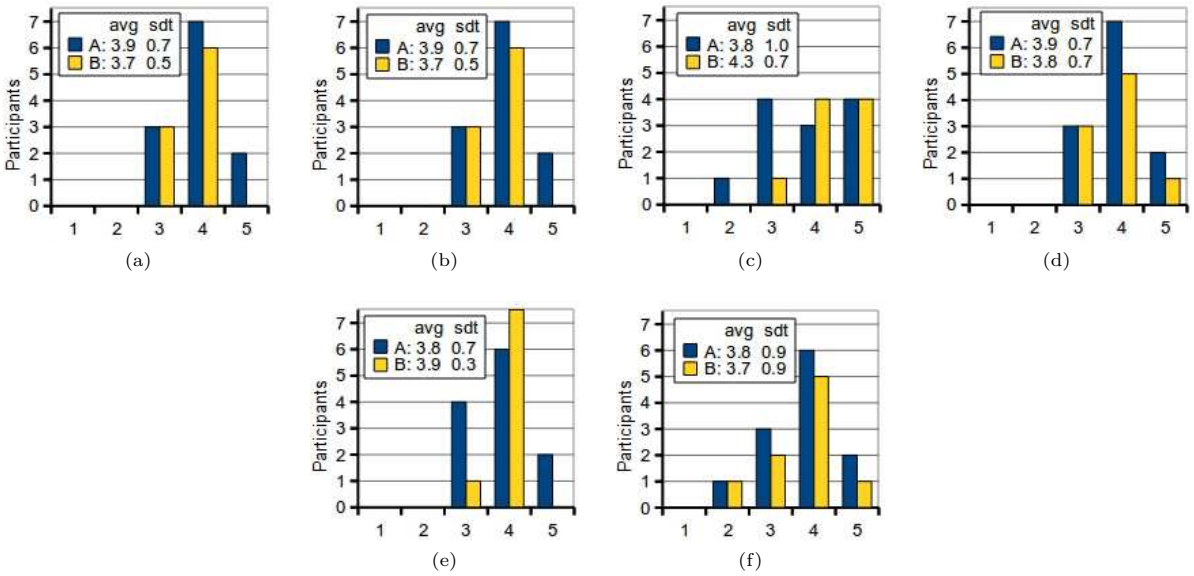


Figure 16: Speed (a), concentration (b), trust (d), confidence (c), predictability (e) and usefulness (f) as expressed using the 5-point Likert scale (1=min, 5=max).

application for demonstration purposes. The iCompose interface provides key features for both controlling application composition and adapting applications at runtime. We reported our findings from the prototype design and implementation, as well as from the prototype evaluation which we carried out with 21 test users. The design and implementation of the prototype demonstrated the technical feasibility of our approach, while the evaluation demonstrated feasibility from the user's point of view.

Technical feasibility issues. Our main goal was to develop a solution supporting application composition for users, hence we focused primarily on designing the iCompose interface and the technologies realizing it. This interface relies on the distributed reasoning mechanism which we implemented for mobile phones. The key challenge here was to design the reasoning protocol that handles reasoning both on multiple mobile clients and the server. This was further complicated due to user mobility and interactivity of reasoning. The mobile clients may disconnect in the middle of reasoning, just as users may simply walk away or even close the application when, e.g. receiving a phone call. The interactivity of reasoning is caused by the fact that the QuizBlasters application required the users to confirm some decisions through the GUI. Therefore, the reasoning protocol had to cope with the uncertainty of the mobile clients' state, which can be either 'disconnected' or 'waiting for user input'. We solved this issue by incorporating a polling mechanism with timeouts in the reasoning protocol.

Another key challenge was related to the interface that must present visually the user's current and past actions and also their semantic meaning using only a limited space on the mobile phone's screen. We resolved this issue by designing the event collector screen. This screen combines two elements: a scrollable list highlighting the recently touched tag icons on top and the textual description of these actions at the bottom. The top part of this screen is always present, so that the user is aware if a tag or a handshake event occurs. The space occupied by the textual description was also utilized for confirmations, notifications and various dialogs (as shown in Figure 10, c). We elaborated on the visual design of this screen while building and evaluating the low-fidelity prototype (i.e. the 'paper prototype'). The final design of this screen is shown in Figure 10 (b).

We developed the key prototype components, the reasoner and the mobile client, first as standalone components and then integrated them into the REACHeS infrastructure which provided the prototype with the ubiquitous

resource management functionality. In contrast, the iCompose interface was designed using the so-called paper prototyping approach, which we applied in our development process for the first time. Owing to the fact that this approach includes users early in the development process, we were able to identify and elaborate on the most important interaction patterns for application composition even before implementing the final prototype. In addition, this shortened the overall duration of the development process and saved time which we would otherwise have spent building several fully-functioning intermediate prototypes. We plan to use such an approach for developing our research prototypes in the future.

Another finding was related to our decision to use Nokia 6131 mobile phones as the target devices for the system implementation. Although we acknowledge the high precision and robustness demonstrated by the NFC readers integrated in these devices, the devices' abilities to support advanced graphical user interfaces are constrained by the J2ME platform. As a result, we observed that our users were slightly disappointed when they saw that our application and the prototype do not support a touch screen-based GUI. Most of the evaluators already had some experience with this kind of interface and suggested switching to a more advanced mobile platform. Indeed, a touch screen-based GUI offers additional flexibility both for the users and the designers of user interfaces. We plan to use another mobile platform as the target device for future implementations of our system.

We also encountered the issue of feedback and the need to utilize various interface modalities to provide it. Initially, we designed our prototype with the vision that mobile devices would play the role of interaction tools in the ubiquitous environment. Although this vision is advantageous in multiple ways, it requires changing traditional flows of interaction, customary between users and their mobile devices. Indeed, in a traditional mobile application, user attention is always focused on the mobile phone's screen. In contrast, we have observed that users switch their attention between the mobile phone's screen to the environment when interacting with a ubiquitous application. As a result, the mobile phone's screen (i.e. visual modality) is no longer the only reliable source of information, which raises the need to design applications providing feedback in combination with haptic and sound stimuli.

User experience issues. We evaluated the feasibility of the prototype from the users' viewpoint during the user study which focused on usability, user feeling in control and user acceptance issues. We revealed during this study the similarity between the opinions of the expert and novice user groups, which generally followed each other hand in hand. Overall, the users expressed positive opinions regarding the functionality and look-and-feel of the iCompose interface and the prototype. We attribute this to our decision to design this interface using the principles of direct manipulation, intelligent assistance paradigm, and asynchronous interaction style. The principle of direct manipulation was realized by the NFC-based physical browsing interface which allowed users to choose resources in the environment by touching. Compared with other physical browsing techniques (i.e. pointing and scanning), touching is characterized as the most natural and intuitive interaction technique that requires the lowest cognitive load on the part of the users [59]. The intelligent assistance paradigm ensured the user felt in control when using our user interface. This paradigm advocates user interfaces where the users and the system collaborate towards achieving the same goal [51]. Compared with other paradigms, e.g. computer-aided or autonomous composition [16], intelligent assistance provides additional flexibility and the feeling of control. Finally, our decision to use the asynchronous interaction style was influenced by the nature of user interaction in ubiquitous environments that spans both space and time. We anticipate the users' need to compose applications while simultaneously performing multiple other tasks and engaging multiple different resources in the environment. In order to support this kind of interaction, the iCompose interface also includes the eventing mechanism running as a background process for processing user actions regardless of the currently active application.

During the user study, we also observed how users deal with context and context-awareness in our prototype. Although we expected these issues to be difficult for users to comprehend and cause various controversies, the participants raised only a few concerns. All these users' remarks were related to the issues of user control and privacy. For instance, some users wished to have control over 'friend lists' which our application uses to distribute invitations. They wanted to be able to access and edit these lists as they preferred these invitations to be sent to their friends only. Another person was concerned about the fact that anybody can potentially make a handshake with her mobile phone even without her authorization. Although we admit that this is indeed the way handshakes

work in our prototype, such a threat is only possible if a user leaves her/his mobile phone unattended. We plan to resolve these issues in future implementations of the prototype.

However, we should point out several limitations of this user experiment. The evaluation of a platform usually requires using concrete applications as examples. Hence, our experiment used the QuizBlasters application for this purpose. However, introducing applications might change the focus to evaluating the applications themselves. One possible solution to address this issue is the complete and thorough evaluation of the platform with several different applications using appropriate feedback collection methods. In addition, such thorough evaluation would require participants to utilize the prototype on a daily basis over a long period of time (over the course of several days or even weeks), so that users fully evaluate the context-aware features of the prototype.

We found several interesting directions for future research, however. One of them is the utilization of additional types of context which can be achieved by using an existing context management framework for mobile devices such as Contory [56], Perception Framework [21] or the solution suggested by Cadenas et al [11]. Another direction is the creation of an application framework to assist the design of ubiquitous applications. Such a framework will provide the tools, for example, for specifying composite applications, creating and editing the rules that define the conditions for triggering composition, and so on.

Concluding, we presented the context-aware application composition approach which supports a user's everyday activities and needs. This approach combines a physical and a graphical user interface on mobile devices with context-aware mechanisms for composing and controlling applications at runtime. Users compose applications by simply touching resources in the environment using their mobile devices, while the graphical user interface provides feedback and assists users by suggesting possible further actions. The combination of context information and a physical interface for application composition made it possible to design the system which our users found pleasurable, easy to control and suitable for their needs.

8. Acknowledgments

This work has been funded by Academy of Finland (as the Pervasive Service Computing project), by GETA (Finnish Graduate School in Electronics, Telecommunications and Automation), Walter Ahlström and HPY Research Foundations. The authors of this paper would like to thank all users who participated the user tests and helped us to collect data. The first author would like to acknowledge Richard James (from INRIA Paris-Rocquencourt) for his valuable language advice.

References

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC'99, pages 304–307, London, UK, 1999. Springer-Verlag.

[2] H. Ailisto, T. Matinmikko, A. Ylisaukko-Oja, E. Strommer, M. Hillukkala, A. Wallin, E. Siira, A. Poyry, V. Tormanen, and T. Huomo. Physical Browsing with NFC Technology. *VTT Tiedotteita*, 2400, 2007.

[3] Sandrine Beauche and Pascal Poizat. Automated Service Composition with Adaptive Planning. In *Proceedings of the 6th International Conference on Service-Oriented Computing (ICSOC'08)*, LNCS 5364, pages 530–537. Springer, 2008.

[4] Sonia Ben Mokhtar, Nikolaos Georgantas, and Valérie Issarny. COCOA: COnversation-based Service Composition in PervAsive Computing Environments with QoS Support. *Journal of Systems and Software*, 80(12):1941–1955, 2007.

[5] Antonia Bertolino, Guglielmo Angelis, Lars Frantzen, and Andrea Polini. The PLASTIC Framework and Tools for Testing Service-Oriented Applications. In *Proceedings of the International Summer School on Software Engineering (ISSSE 2006-2008)*, LNCS 5413, pages 106–139. Springer, 2009.

[6] Antonis Bikakis and Grigoris Antoniou. Distributed Defeasible Contextual Reasoning in Ambient Computing. In *Proceedings of the 2008 European Conference on Ambient Intelligence (AmI'08)*, volume 5355 of *LNCS*, pages 308–325. Springer, 2008.

[7] Andre Bottaro, Johann Bourcier, Clement Escoffier, and Philippe Lalanda. Context-Aware Service Composition in a Home Control Gateway. In *Proceedings IEEE International Conference on Pervasive Services*, pages 223–231, 2007.

[8] Gregor Broll, Enrico Rukzio, Massimo Paolucci, Matthias Wagner, Albrecht Schmidt, and Heinrich Hussmann. PERCI: Pervasive Service Interaction with the Internet of Things. *IEEE Internet Computing*, 13:74–81, 2009.

[9] Jeppe Brønsted, Klaus Marius Hansen, and Mads Ingstrup. Service Composition Issues in Pervasive Computing. *IEEE Pervasive Computing*, 9(1):62–70, 2010.

[10] John Buford, Rakesh Kumar, and Greg Perkins. Composition Trust Bindings in Pervasive Computing Service Composition. In *Proceedings of the 4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'06)*, pages 261–266, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Alejandro Cadenas, Carlos Ruiz, Iker Larizgoitia, Raúl García-Castro, Carlos Lamsfus, Iñaki Vázquez, Marta González, David Martín, and María Poveda. Context Management in Mobile Environments: A Semantic Approach. In *Proceedings of the 1st Workshop on Context, Information and Ontologies*, CIAO'09, pages 2:1–2:8, New York, NY, USA, 2009. ACM.

[12] Shin-Chih Chang, Chun-Feng Liao, Yong-Cheng Liu, and Li-Chen Fu. A Spontaneous Preference Aware Service Composition Framework for Message-Oriented Pervasive Systems. In *Proceedings of the 4th International Conference on Pervasive Computing and Applications (ICPCA'09)*, pages 441–446, Tamkang University, Taiwan, December 2009. IEEE Computer Society.

[13] Maria Chantzara, Miltiades Anagnostou, and Efstathios Sykas. Designing a Quality-Aware Discovery Mechanism for Acquiring Context Information. In *Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06)*, pages 211–216, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Jeannette Chin, Vic Callaghan, and Graham Clarke. An End-User Tool for Customising Personal Spaces in Ubiquitous Computing Environments. In *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing (UIC'06)*, LNCS 4159, pages 1080–1089. Springer, 2006.

[15] Oleg Davidyuk, Iván Sánchez, Jon Imanol Duran, and Jukka Riekki. Autonomic Composition of Ubiquitous Multimedia Applications in REACHES. In *Proceedings of the 7th International ACM Conference on Mobile and Ubiquitous Multimedia (MUM'08)*, pages 105–108. ACM, 2008.

[16] Oleg Davidyuk, Iván Sánchez, and Jukka Riekki. *CADEAU: Supporting Autonomic and User-Controlled Application Composition in Ubiquitous Environments*, chapter 4, pages 74–103. Pervasive Computing and Communications Design and Deployment: Technologies, Trends, and Applications. IGI Global, 2011.

[17] Oleg Davidyuk, Istvan Selek, Jon Imanol Duran, and Jukka Riekki. Algorithms for Composing Pervasive Applications. *International Journal of Software Engineering and Its Applications*, 2(2):71–94, 2008.

[18] Vicki de Mey. *Visual Composition of Software Applications*, chapter 10, pages 275–303. Object-Oriented Software Composition. Prentice Hall, 1995.

[19] Anind K. Dey and Alan Newberger. Support for Context-Aware Intelligibility and Control. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI'09, pages 859–868, New York, NY, USA, 2009. ACM.

[20] Giuseppe Ghiani, Fabio Patern, and Lucio Davide Spano. Cicero Designer: An Environment for End-User Development of Multi-Device Museum Guides. In *Proceedings of the 2nd International Symposium on End-User Development (IS-EUD'09)*, LNCS 5435, pages 265–274. Springer, 2009.

[21] Ekaterina Gilman, Xiang Su, Oleg Davidyuk, Jiehan Zhou, and Jukka Riekki. Perception Framework for Supporting Development of Context-Aware Web Services. *International Journal of Pervasive Computing and Communications*, 7(4):339–364, December 2011.

[22] Tom Gross and Nicolai Marquardt. Creating, Editing and Sharing Complex Ubiquitous Computing Environment Configurations with CollaborationBus. *Scientific International Journal for Parallel and Distributed Computing: Scalable Computing. Practice and Experience*, 11(3):289–303, 2010.

[23] Bob Hardian, Jadwiga Indulska, and Karen Henricksen. Exposing Contextual Information for Balancing Software Autonomy and User Control in Context-Aware Systems. In *Proceedings of the Workshop on Context-Aware Pervasive Communities: Infrastructures, Services and Applications*, 2008.

[24] Cristian Hesselman, Andrew Tokmakoff, Pravin Pawar, and Sorin Iacob. Discovery and Composition of Services for Context-Aware Systems. In P. Havinga, M.E.M. Lijding, N. Meratnia, and M. Wegdam, editors, *Smart Sensing and Context*, LNCS 4272, pages 67–81, Heidelberg, October 2006. Springer.

[25] Noha Ibrahim and Frederic Le Mouel. A Survey on Service Composition Middleware in Pervasive Environments. *International Journal of Computer Science Issues*, 1:1–12, August 2009.

[26] JSON schema. JavaScript Object Notation (JSON), Lightweight Open Standard for Human-Readable Data Interchange, http://www.json.org, June 2011.

[27] Swaroop Kalasapur, Mohan Kumar, and Behrooz Shirazi. Dynamic Service Composition in Pervasive Computing. *IEEE Transactions on Parallel Distributed Systems*, 18(7):907–918, 2007.

[28] Fahim Kawsar, Tatsuo Nakajima, and Kaori Fujinami. Deploy Spontaneously: Supporting End-Users in Building and Enhancing a Smart Home. In *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp'08)*, pages 282–291, NY, USA, 2008. ACM.

[29] Fernando Koch, John-Jules Ch. Meyer, Frank Dignum, and Iyad Rahwan. Programming Deliberative Agents for Mobile Services: The 3APL-M Platform. In *Proceedings of the AAMAS'05 Workshop on Programming Multi-Agent Systems (ProMAS'05)*, pages 222–235, 2005.

[30] kXML2 parser. kXML2, Lightweight Java-based XML Parser for Resource-limited Devices , http://kxml.sourceforge.net/kxml2/, June 2011.

[31] Margarita Vinagre Laranjeira. Social Interaction in Asynchronous Learning Environments. *Anglogermanica: Revista electronica periódica de filología alemana e inglesa*, pages 13–26, 2006.

[32] Jasper Lindenberg, Wouter Pasman, Kim Kranenborg, Joris Stegeman, and Mark A. Neerincx. Improving Service Matching and Selection in Ubiquitous Computing Environments: a User Study. *Personal Ubiquitous Computing*, 11(1):59–68, 2006.

[33] Ryusuke Masuoka, Bijan Parsia, and Yannis Labrou. Task Computing - the Semantic Web meets Pervasive Computing. In *Proceedings of the 2nd International Semantic Web Conference (ISWC'03)*, LNCS 2870, pages 866–881. Springer, Oct 2003.

[34] Irene Mavrommati, Achilles Kameas, and Panos Markopoulos. An Editing Tool that Manages Device Associations in an In-Home Environment. *Personal Ubiquitous Computing*, 8(3-4):255–263, 2004.

[35] Joseph Meloche, Helen Hasan, and Angelo Papakosmas. Support for Asynchronous Interaction in Group Experiential Learning. *Outlines. Critical Practice Studies*, 6(2):47–62, 2004.

[36] Alan Messer, Anugeetha Kunjithapatham, Mithan Sheshagiri, Henry Song, Praveen Kumar, Phuong Nguyen, and Kyoung Hoon Yi. InterPlay: a Middleware for Seamless Device Integration and Task Orchestration in a Networked Home. In *Proceedings of the 4th Annual IEEE Conference on Pervasive Computing and Communications*, pages 296–307. IEEE Computer Society, March 2006.

[37] Sonia Ben Mokhtar, Damien Fournier, Nikolaos Georgantas, and Valérie Issarny. Context-Aware Service Composition in Pervasive Computing Environments. In *2nd International Workshop on Rapid Integration of Software Engineering Techniques (RISE'05)*, pages 129–144, 2005.

[38] Elena Mugellini, Omar Abou Khaled, Stéphane Pierroz, Stefano Carrino, and Houda Chabbi Drissi. Generic Framework for Transforming Everyday Objects into Interactive Surfaces. In *Proceedings of the 13th International Conference on Human-Computer Interaction, Part III*, LNCS 5612, pages 473–482. Springer, 2009.

[39] Elena Mugellini, Elisa Rubegni, Sandro Gerardi, and Omar Abou Khaled. Using Personal Objects as Tangible Interfaces for Memory Recollection and Sharing. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction (TEI'07)*, pages 231–238, NY, USA, 2007. ACM.

[40] Hamid Mukhtar, Djamel Belaïd, and Guy Bernard. Dynamic User Task Composition Based on User Preferences. *ACM Transactions on Autonomous Adaptive Systems*, 6:4:1–4:17, February 2011.

[41] J. Nakazawa, J. Yura, and H. Tokuda. Galaxy: a Service Shaping Approach for Addressing the Hidden Service Problem. In *Proceedings of the 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems*, pages 35–39, 2004.

[42] Mark Newman and Mark Ackerman. Pervasive Help @ Home: Connecting People Who Connect Devices. In *Proceedings of the International Workshop on Pervasive Computing at Home (PC@Home)*, pages 28–36, 2008.

[43] Mark Newman, Ame Elliott, and Trevor Smith. Providing an Integrated User Experience of Networked Media, Devices, and Services through End-User Composition. In *Proceedings of the 6th International Conference on Pervasive Computing (Pervasive'08)*, LNCS 5013, pages 213–227. Springer, 2008.

[44] NFC Forum. NFC Data Exchange Format (NDEF), http://www.nfc-forum.org/specs/, June 2011.

[45] Justin Mazzola Paluska, Hubert Pham, Umar Saif, Grace Chau, Chris Terman, and Steve Ward. Structured Decomposition of Adaptive Applications. *Pervasive and Mobile Computing*, 4(6):791–806, 2008.

[46] Justin Mazzola Paluska, Hubert Pham, Umar Saif, Grace Chau, Chris Terman, and Steve Ward. Structured Decomposition of Adaptive Applications. In *Proceedings of the 6th Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'08)*, pages 1–10. IEEE Computer Society, March 17–21, 2008.

[47] Davy Preuveneers and Yolande Berbers. Automated Context-Driven Composition of Pervasive Services to Alleviate Non-Functional Concerns. *International Journal of Computing and Information Sciences*, 3(2):19–28, August 2005.

[48] Michael Przybilski. Distributed Context Reasoning for Proactive Systems. In *Proceedings of the 2005 Workshop on Context-Awareness for Proactive Systems (CAPS'05)*, pages 43–53, 2005.

[49] Anand Ranganathan and Roy H. Campbell. Autonomic Pervasive Computing Based on Planning. In *Proceedings of the International Conference on Autonomic Computing*, pages 80–87, Los Alamitos, CA, USA, 2004. IEEE Computer Society.

[50] Olli Rantapuska and Mia Lahteenmaki. Task-based User Experience for Home Networks and Smart Spaces. In *Proceedings of the International Workshop on Pervasive Mobile Interaction Devices*, pages 188–191, 2008.

[51] Charles Rich and Candace Sidner. DiamondHelp: A Generic Collaborative Task Guidance System. *AI Magazine*, 28(2):33–46, 2007.

[52] Jukka Riekki. RFID and Smart Spaces. *International Journal of Internet Protocol Technology*, 2(3-4):143–152, 2007.

[53] Jukka Riekki, Iván Sánchez, and Mikko Pyykkonen. Remote Control for Pervasive Services. *International Journal of Autonomous and Adaptive Communications Systems*, 3(1):39–58, 2010.

[54] Peter Rigole, Tim Clerckx, Yolande Berbers, and Karin Coninx. Task-Driven Automated Component Deployment for Ambient Intelligence Environments. *Pervasive and Mobile Computing*, 3(3):276–299, 2007.

[55] Peter Rigole, Chris Vandervelpen, Kris Luyten, Yolande Berbers, Yves Vandewoude, and Karin Coninx. A Component-Based Infrastructure for Pervasive User Interaction. In *Proceedings of Software Techniques for Embedded and Pervasive Systems*, pages 1–16. Springer, 2005.

[56] Oriana Riva. Contory: A Middleware for the Provisioning of Context Information on Smart Phones. In *Proceedings of the ACM/IFIP/USENIX 7th International Middleware Conference (Middleware'06)*, volume 4290 of *LNCS*, pages 219–239. Springer Berlin / Heidelberg, 2006.

[57] Stamatia Rizou, Kai Haussermann, Frank Durr, Nazario Cipriani, and Kurt Rothermel. A System for Distributed Context Reasoning. In *Proceedings of the 6th International Conference on Autonomic and Autonomous Systems (ICAS'10)*, pages 84–89, Los Alamitos, CA, USA, 2010. IEEE Computer Society.

[58] Romain Rouvoy, Paolo Barone, Yun Ding, Frank Eliassen, Svein O. Hallsteinsen, Jorge Lorenzo, Alessandro Mamelli, and Ulrich Scholz. MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In *Proceedings of Dagstuhl Seminar: Software Engineering for Self-Adaptive Systems*, LNCS 5525, pages 164–182, 2009.

[59] Enrico Rukzio, Karin Leichtenstern, Victor Callaghan, Paul Holleis, Albrecht Schmidt, and Jeannette Shiaw-Yuan Chin. An Experimental Comparison of Physical Mobile Interaction Techniques: Touching, Pointing and Scanning. In *Proceedings on the 8th International Conference on Ubiquitous Computing (UbiComp'06)*, pages 87–104, 2006.

[60] Hannamari Saarenpää, Hannu Korhonen, and Janne Paavilainen. Asynchronous Gameplay in Pervasive Multiplayer Mobile Games. In *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA'09, pages 4213–4218, New York, NY, USA, 2009. ACM.

[61] Mario Schuster, Alexander Domene, Raju Vaidya, Stefan Arbanowski, Su Myeon Kim, Jin Wook Lee, and Hun Lim. Virtual Device Composition. In *Proceedings of the 8th International Symposium on Autonomous Decentralized Systems (ISADS'07)*, pages 270–278, Washington, DC, USA, 2007. IEEE Computer Society.

[62] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Pearson Addison Wesley, 2004.

[63] Joao Pedro Sousa, Vahe Poladian, David Garlan, Bradley Schmerl, and Mary Shaw. Task-based Adaptation for Ubiquitous Computing. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 36:328–340, 2006.

[64] Joao Pedro Sousa, Bradley Schmerl, Vahe Poladian, and Alex Brodsky. uDesign: End-User Design Applied to Monitoring and Control Applications for Smart Spaces. In *Proceedings of the Working IEEE/IFIP Conference on Software Architecture*, pages 71–80. IEEE Computer Society, 2008.

[65] M. Takemoto, T. Oh-ishi, T. Iwata, Y. Yamato, Y. Tanaka, K. Shinno, S. Tokumoto, and N. Shimamoto. A Service-Composition and Service-Emergence Framework for Ubiquitous-Computing Environments. In *Proceedings of the 2004 Workshop on Applications and the Internet (as part of SAINT'04)*, pages 313–318, 2004.

[66] Aitor Urbieta, Guillermo Barrutieta, Jorge Parra, and Aitor Uribarren. A Survey of Dynamic Service Composition Approaches for Ambient Systems. In *Proceedings on Ambi-Sys Workshop on Software Organisation and Monitoring of Ambient Systems (SOMITAS'08)*, pages 1–8, ICST, Brussels, Belgium, 2008. ICST.

[67] Pasi Välkkynen, Marketta Niemelä, and Timo Tuomisto. Evaluating Touching and Pointing with a Mobile Terminal for Physical Browsing. In *Proceedings of the 4th Nordic Conference on Human-Computer Interaction: Changing Roles (NordiCHI'06)*, pages 28–37, New York, NY, USA, 2006. ACM.

[68] Geert Vanderhulst, Kris Luyten, and Karin Coninx. Put the User in Control: Ontology-Driven Meta-level Interaction for Pervasive Environments. In *Proceedings of the 1st International Workshop on Ontologies in Interactive Systems*, pages 51–56, Washington, DC, USA, 2008. IEEE Computer Society.

[69] Martijn Vastenburg, David Keyson, and Huib de Ridder. Measuring User Experiences of Prototypical Autonomous Products in a Simulated Home Environment. *Human Computer Interaction (HCI)*, 2:998–1007,

2007.

[70] Jose Viterbo, Markus Endler, and Gustavo Baptista. A Two-Tiered Approach for Decentralized Reasoning in Ambient Intelligence. *Intelligent Systems, IEEE*, 25(5):54–60, sept.-oct. 2010.

[71] James Watt, Joseph Walther, and Kristine Nowak. Asynchronous Videoconferencing: a Hybrid Communication Prototype. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 97–105, 2002.

[72] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265:94–104, 1991.

[73] Yuping Yang, Fiona Mahon, M. Williams, and Tom Pfeifer. Context-Aware Dynamic Personalised Service Re-composition in a Pervasive Service Environment. In *Ubiquitous Intelligence and Computing*, volume 4159 of *Lecture Notes in Computer Science*, pages 724–735. Springer Berlin / Heidelberg, 2006.

[74] Baopeng Zhang, Yuanchun Shi, and Xin Xiao. A Policy-Driven Service Composition Method for Adaptation in Pervasive Computing Environment. *The Computer Journal*, 53(2):152–165, 2007.

[75] Jiehan Zhou, Ekaterina Gilman, Juha Palola, Jukka Riekki, Mika Ylianttila, and Junzhao Sun. Context-Aware Pervasive Service Composition and Its Implementation. *Personal Ubiquitous Computing*, 15:291–303, March 2011.