Component-Based Development of Mobile Assistants with the ELEPHANT System

Ilhan Aslan
Fraunhofer ESK
Munich
Germany
Ilhan.aslan@esk.fraunhofer.de

Dyuti Menon
Fraunhofer ESK
Munich
Germany
dyuti.menon@esk.fraunhofer.de

ABSTRACT
We designed a component-based development process for mobile assistants that we realized in the ELEPHANT (ELEments for Pervasive and Handheld AssistaNTs) system. The ELEPHANT system encompasses a middleware solution for mobile assistants. In a first step users can utilize this system to mash-up interactive components into adaptive application nodes. In a second step users can compose a workflow of a full functional application by linking application-nodes. The overall goal of the system is to minimize the effort required for creating, managing and using mobile assistants with enhanced capabilities such as utilizing user and device presence information and accessing hardware near sensor data. We propose a distributed system and the use of concepts found in social software for collaboratively developing mobile assistants.

Categories and Subject Descriptors
H.5.2 [**User Interfaces**]: User interface management systems (UIMS), H.5.3 [**Group and Organization Interfaces**] Collaborative Computing

General Terms
Management, Design, Human Factors, Theory

Keywords
Mobile assistants, Context-aware, Middleware, Design process

1. INTRODUCTION
Characteristics of mobile personal assistants are context adaptation, multiple interaction modalities and specialized interaction design, e.g. design for small touch sensitive screens and devices with specific sensors [14, 23]. However, these characteristics are also challenges for the Mobility community. Mobiles have been criticized for their bad usability due to restricted recourses [21], e.g. display sizes, bandwidth and bad data connectivity. Today we observe that some of the restrictions are eliminated due to the increasing computation power, memory and better connectivity of mobile devices.

Mobility 2009, Sep 2-4, Nice, France Copyright © 2009
ACM 978-1-60558-536-9/00/0009……$5.00

Nevertheless the development of personal assistants is due to subjective and ambiguous interpretation of context information and different user preferences, in addition to the heterogenic landscape of mobile devices [13, 14], very complex. On the other hand, the spread of Web 2.0 solutions in the last few years created "active web users" [28, 30] who not only consume information and services but also produce information and services; for example, through Wikis, Blogs but also through the use of Mashup applications. A Mashup is an application that combines data, either through APIs or other sources, into a single integrated user experience [29]. Mashup editors allow non-programmer end-users to mash-up information sources and services to meet their information needs [10]; examples are Yahoo Pipes, Google Mashup Editor, Microsoft PopFly and IBM QEDWiki. These tools allow for visual creation by connecting services and sources together in various ways. The area of mobile assistance is one with highly personalized requirements. We believe that end-user involvement is essential in the process of designing and developing mobile assistance solutions for a broad range of application fields. The goal of the system and the underlying concepts that we present in this paper are to provide tools for end-users to easily compose/mash-up personalized mobile assistants that are context sensitive and utilize the full functionality of the target mobile device. To be able to compose value added mobile assistants, it is important to identify a framework for mobile assistance and a concept for dividing mobile assistance solutions in small elements. Our research is based on the assumption that it is possible to break down the interaction process found in mobile assistants down to small interactive components that we refer to as elements. An element in the ELEPHANT system (E-element) is a sub-application following the traditional Model-View-Controller pattern. Each element (e.g. a widget or an offline application linked to resources) is tagged with descriptive information (Meta data) such as what interactions are provided, what kind of sensors are accessed, what kind of web services are linked and even kinds of situations and activities the element is suitable for. We hypothesize that given a range of tools; end users are capable to mash-up elements in smart application nodes and compose a workflow of these smart nodes to create useful, mobile and context sensitive assistants (E-application). In the following we first describe related work and present our contribution. Then we describe the theoretical model and the architectural components of the ELEPHANT system in more detail.

We have implemented and tested a first version of our authoring tool (E-composer) to compose E-application [31]. Therefore, we do not focus on the UI of a specific authoring tool but the underlying design process and components of the ELEPHANT system in this paper.

1.1 Related Work

Improving the development for mobile and ubiquitous applications has been a challenge and subject for many researchers [19, 17, 12, 18]. The design paradigm for mobile and ubiquitous applications is moving from a user- and task-based approach to an activity- and context-based approach [6, 7, 11, 15]. Activity theory differentiates between activities, actions and operations that are in a hierarchical relation to each other [15].

Previously, many researchers have proposed solutions at a conceptual level or for early prototyping of ubiquitous applications. Also, due to the heterogenic landscape of ubiquitous applications solutions were proposed for a subclass of ubiquitous applications; for example, Li and Landay focus in [19] on the class of applications that improves the awareness of one's activities as well as the activities of others (e.g. monitoring fitness status in a family through digital picture frames). Whereas, when designing for mobile applications, the mobile device (e.g. the mobile phone) is the main and very often the only human machine interface to ubiquitous machine functionalities in the environment [20]; in ubiquitous applications the interface to machine functionalities is various and woven into the fabric of everyday life [25]. The difference between designing for mobile and ubiquitous applications is in the realization of the interface between user and machine. Having said this, the design process on a conceptual level; for example, defining tasks or activities seems to be very similar for ubiquitous and mobile applications.

Previously, successful prototypes of mobile assistants with different new characteristics have been presented. For example in [24] Wasinger et al. presented a prototype shopping assistant with advanced interfaces for multi modal interaction (e.g. combination of different input and output modalities). In [8] Bohnenberger et al. presented also a mobile shopping assistant with focus on user requirements and decision theoretic planning. In [2] Aslan et al. presented an information system and multiple mobile services that were designed to ease the everyday challenges for tourists. The COMPASS2008 assistance services were context sensitive regarding the interaction modalities and user profiles. Usability and acceptance of the COMPASS2008 solutions were evaluated in [22] with positive results. These works aimed to improve the user assistance with mobile devices by implementing new functionalities and providing higher personalization and adaptive content within the use context. It is undeniable that mobile devices provide useful information by reporting user location and accessing user profiles. However, when regarding assistance systems it is also important to consider effects of the assistance; for example in [1] the authors propose that instead of trying to reduce the amount of cognitive load whenever possible, it seems to be necessary to support the mental elaboration of specific information in a way adaptive to the situation. This seems to make sense when the assistance aims to have an effect on the human behavior and knowledge (e.g. assisting to keep healthy, or assist in learning). Consequently, the design of assisting systems is not a trivial one. Therefore specialists from the application areas are needed within the development process. These specialists are in most cases non-programmers and need tool support for the development process. Research in mobile assistance has evolved in respect to resource adaptation; however, integrating these findings into the development process of mobile and ubiquitous assistants is still challenging. There are many framework and system approaches that have been dealing with this subject; for example, Ballagas et al. present in [5] a prototyping framework for new sensor based interfaces. The myMytileneCity guide [16] is a prototype tourist guide that allows end-users to choose the content that they are interested in and builds a custom J2ME application for offline use on the move. The Activitydesigner described by Li and Landay in [19] supports designers in building activity-based prototypes. Designers can create detailed activity models from media-rich representation of everyday observations and build stream-based interaction behaviours. Daniel and Matera propose in [9] a framework for the development of context-aware web applications; a component-based development approach for mashing up context-aware web applications that is a similar approach to our component-based composing of mobile assistants that we present in this paper.

Similar to Daniel and Matera's approach we aim with our approach at empowering the users (both developers and end users) with an easy-to- use tool for mashing up applications by integrating ready (adaptive) services or application components. Differences to our approach are in the composing of the application nodes and the use of context. Whereas, in the ELEPHANT system the user can define a general workflow; the workflow in Daniel and Matera's approach is automated through the components that are mashed up, which is typical for traditional mash-up systems. The behavior of the nodes in our ELEPHANT applications have similarities to the mashed up web applications, however we address a new challenge in setting these "mashed up" nodes into a designed workflow for building assistants that is defined through context and user interaction. We also mash up offline and online content and functionalities which is important for mobile usage.

Similar to Li and Landay's work we also address applications for extended periods of time (e.g. not only for the runtime of single application but a sequence). The way we see it is that Li and Landay's focus is strongly on activity based modelling for ubiquitous application whereas the ELEPHANT concept focuses more on flexible composition of UI modules, specifically for assistance applications, in a sense activity-based modelling of the assistance process is possible, but is not mandatory. Our goal is to easily build mobile assistance with adaptive functionalities similar to the solutions presented by Aslan et al in [2]. However, we want to provide tool support to end-users for easily developing and modifying solutions that go even beyond the functionalities described by Aslan et al.

Further related research areas are Hypermedia systems and User Interface Management Systems, in that how ELEPHANT nodes that come with their own UI can be linked with each other. However, the linking procedure is straight forward in our current version (see section 3.2 and 3.3). Therefore we will deal with these research areas in detail, in our future work where we plan to improve our current linking procedure.

1.2 Contribution

We designed a very flexible multilateral process for creating mobile and context sensitive applications; such as for example assistants for work, travel or learning. Users (both end-users and developers) can create their own application workflow based on their understanding of the application e.g. based on sensor

information, layout, media, interaction modalities, user input or abstract concepts such as tasks or activities.

- In doing so, changes in the mental model of users, that is, for example moving from a task based design concept to an activity based concept is supported.
- Our design process is not specific to an application area (e.g. eLearning, Healthcare, Logistics or Transportation) but to a functional domain, namely mobile assistance solutions.
- Our design process for mobile assistants results in highly reusable components. This supports collaborative iteration levels in developing, designing, sharing and tagging E-elements and E-application within communities.
- In our best understanding our approach is the first that combines user composed application workflows with "mashed-up" sub-applications (E-elements) to build mobile assistance solutions.

We realized this process in our ELEPHANT system that we explain in detail in the following sections.

2. CONCEPTUAL EXPLORATION

2.1 The Theoretical Framework

Wandke defines in [24] a conceptual framework for assistance where two qualifications for assistance systems are defined:

- a first qualification of assistance is interactivity
- a second qualification is that some functions have to be performed by the humans and some by machines.

The majority of mobile assistants help users to achieve goals by providing stepwise assistance. Each step involves interaction; such as presenting information to the user, accessing context information or allowing the user to input information. We believe that there are three factors that an advanced mobile assistant has to consider: 1. Time and Workflow/Progress 2. User Interaction and 3. Context.

Furthermore, the impact of these factors on each other has to be considered. Figure 1 presents in an abstract way the complexity of these factors. One specific application of the mobile assistant presented in figure 1 can be presented as a line starting from time zero to a time position where the goal of the assistance is reached. In our presentation the box encompassing the plane would be the whole assistance application. Implementing the functionality of the whole box would be for end-users a very difficult task; however, since we look at assistance applications and assistance is in general a step-wise interactive procedure, it should be possible to break down the bigger box into smaller boxes (see figure 2). For example, the first box could be one that represents a machine functionality (e.g. locating the position of the user).

The second box could be presenting information to the user depending on the location information. Using this information the user could accomplish a human functionality (e.g. walking to the next shop or choosing the right bolt.)

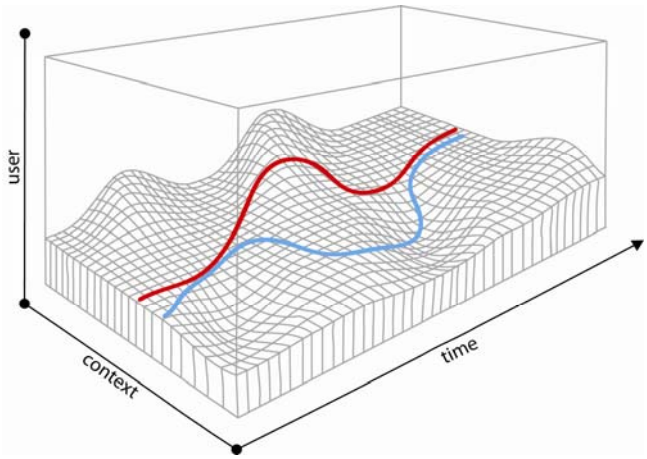

Figure 1: Simplified visualization of the impact of the three factors on executions of the mobile assistant

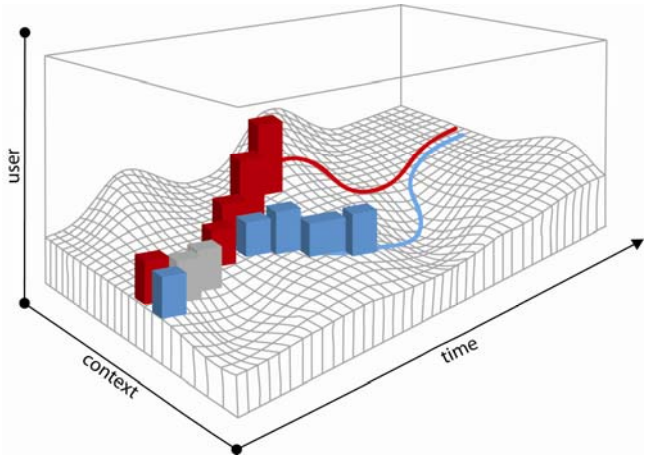

Figure 2: Representing the functionality of a mobile assistant with a series of smaller interactive blocks

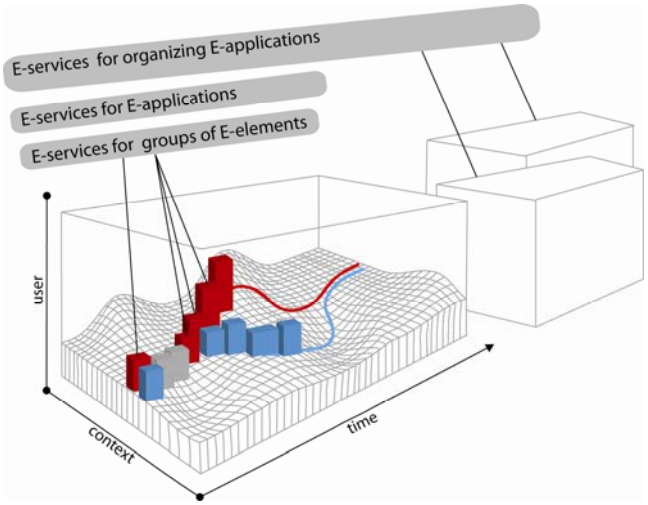

Figure 3: Simplified visualization of the impact of the three factors on executions of the mobile assistant

In the ELEPHANT system we have E-application and E-elements. An E-application is the encompassing bigger block (see figure 2). E-elements are the smaller blocks that provide human and machine functionalities.

The ELEPHANT system's elements based development process also supports an activity based modeling. In Activity-Centered Design (ACD) [11] a set of perspectives on design practices are given. Human activities are described through an activity hierarchy. The three key elements are activities, actions and operations. Actions are accomplished through operations.

Through grouping E-elements, actions as also activities can be modeled. Further more, through E-services it is possible to model long term activities (3). That is, organizing and planning repeated application of one or more E-applications during a long term (e.g. multiple language learning applications within a year).

2.2 Fieldwork

In order to supplement our knowledge of theory with real requirements of end-users we conducted interviews, a paper based task and a usability test with a first prototype authoring tool with 11 users with different backgrounds.

Here we summarize briefly the results from the paper-based test that influenced our decisions related to the ELEPHANT system's design process. The goal of the tests was for the test subjects to create a mobile assistant, which would assist a friend who would shortly be visiting the city of Barcelona. This mobile assistant would assist the visitor with the Spanish language by helping them with the translation of common phrases (to buy tickets, order food etc.) and also be a guide to sightseeing in the city of Barcelona (by providing background information on the interesting places to see). The test subjects were told the objective and provided with a list of content they had at their disposal to create this assistant. The content included text data, images, video clips and audio files, all in reference to Barcelona and the Spanish language.

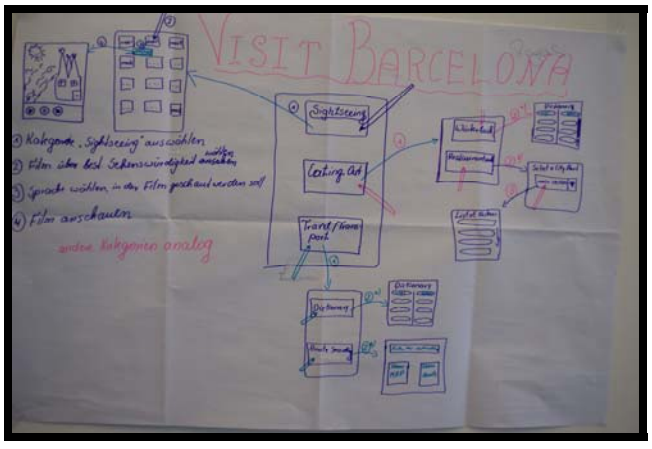

Figure 4: Example result of one of the subjects

Keeping the generation of a Barcelona mobile assistant as the common goal, the task was to design a paper based model. During the test, the test subjects had the complete freedom to design their own structure, and fill it with content.

The results of the interviews and the paper-based tests support our hierarchical step-wise assistance model (8 of 10 subjects' model were hierarchical) (see figure 5 for an example). One of the ten test subjects used only text and arrows. Only one subject didn't deliver any "useful" result, the subject had a very strong technical background. He stated that the task was too abstract.

3. THE ELEPHANT SYSTEM

In this section we first present the overall architecture of the ELEPHANT system and later we present the components in more detail.

3.1 System Architecture

The architecture of the ELEPHANT system is similar to typical web 2.0 solutions. A Client application, namely the ELEPHANT composer (E-composer) is an editor-tool that provides access to users within the community to ELEPHANT repositories. Through functionalities provided by the components: Element Manager, Application Manager and E-services (see figure 5) users can for example search, tag E-elements and E-applications or manage own E-elements, E-applications or E-services. Typically, the ELEPHANT system runs on a dedicated server that can be accessed through a gateway. On the mobile device the ELEPHANT Interpreter (E-interpreter); the ELEPHANT system's local runtime environment, provides a bunch of core functionalities, among other things user and application specific information. That is, the E-interpreter can also accesses the repositories and services depending on the connectivity of the mobile device (e.g. the device presence) through the gateway and the managing components.

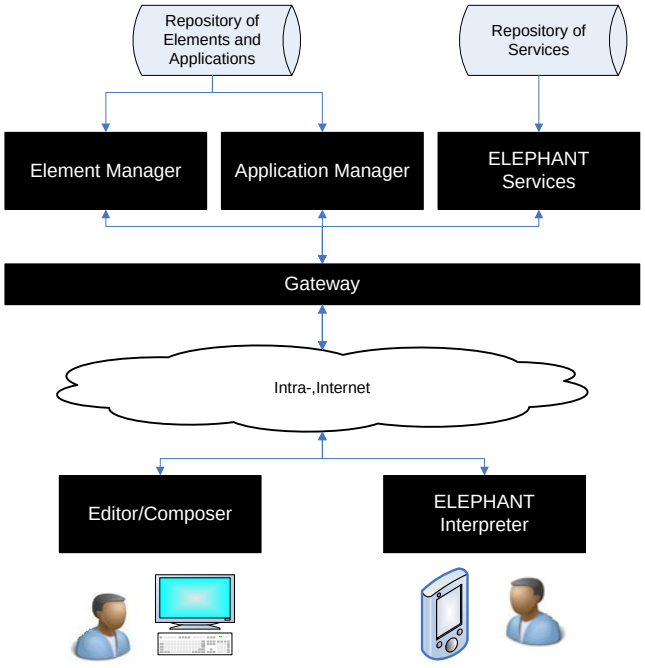

Figure 5: Simplified overview of the ELEPHANT system architecture

3.2 ELEPHANT Elements (E-elements)

When dividing the stepwise assistance process into elements for each step, it is not so easy to identify what exactly a step in terms of functionality is. Therefore, we are not too strict on the definition of E-elements' functionalities (see figure 2). However, E-elements implement a Model-View-Controller (MVC) pattern, where the MVC modules can be distributed to services, E-services as also extern services (e.g. Flickr, Youtube, googlemaps, openstreetmap etc.), the E-Interpreter and the E-application package (see figure 6).

The controller part of the MVC pattern implemented in E-elements can be native; that is, developed by the E-element's designer (e.g. implemented in a scripting language). The controller part can also be linked to a service; as for example it is typical with widgets. A currency calculator widget, that has a view implemented in HTML and runs embedded in the browser can, for example access an extern currency converter service via HTTP requests to receive up to date information.

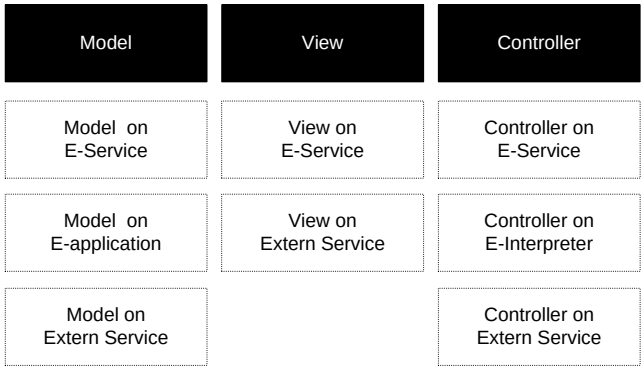

Figure 6 : The Model-View-Controller components can be distributed to services or the E-Interpreter

As a consequence the view part can also be rolled out to the service (e.g. HTML, or Flash snippets). However, in most cases the E-elements come with their native view (e.g. graphical and auditive layouts and interactive UI components).

The Model part in E-elements can be native (e.g. multimedia attached by the designer of the E-element), linked in the process of composing an E-application or located on a server (e.g. map images highlighting the current position of the user).

Figure 7 presents two examples of E-elements. For these two E-elements the Model part is not implemented/linked by the designer of the E-element; it's up to the author that composes an application to link these "media templates" with content. An author could use the element on the left hand side to present a set of images and text related to the images, allowing the user to navigate through the images by using the arrow buttons. In that sense the element provides native control for going back and forward through a list of images. The template on the right hand side could be used to present the user a list of items. E-element designers can use an API of core functionalities provided by E-services and the E-interpreter.

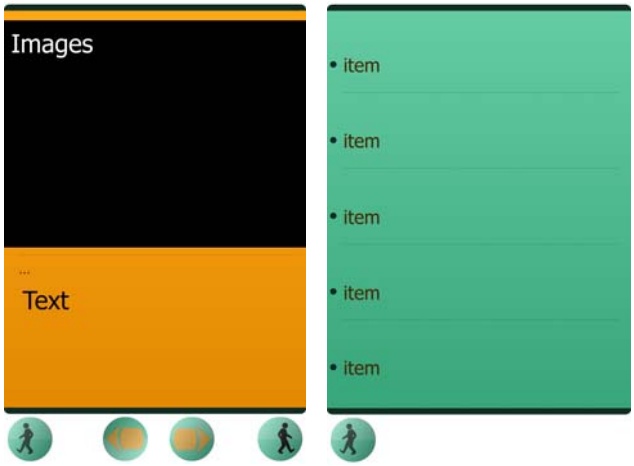

Figure 7: Two templates are presented. On the left hand side a template to present a set of image and text. On the right hand side a list with items.

For example, the functions nextAppNode and previousAppNode are provided by the E-interpreter to step to the next or previous E-element in the E-application's workflow (e.g. in figure 7, the E-element on the left hand side provides two buttons for these functionalities). The E-interpreter decides based on the structure of the E-application composed by the designer of the E-application and the context conditions at runtime (e.g. user input, user and device presence) which E-element is the next or the previous in application nodes (see section 3.3).

Characteristics and properties of use for each E-element in the repository are stored in a XML-based Meta description language. This language describes E-elements in terms of:

- Presence of the device; that is, properties of the E-element that are related to availability and status of sensors and hardware specifications (e.g. WLAN, Bluetooth, battery, bandwidth, camera, microphone, headset or accelerometer) in an understandable terminology for the user.
- Presence of the user; that is, properties that are related to an actively set status of the user.
- Presence of the service; that is, properties that are related to E-services (e.g. identity and password management).
- Application area (e.g. eLearning, Healthcare, Transportation, etc.)
- Abstract functionality, such as functionality related to tasks and activities (e.g. working out, reading, etc.)
- Comments made by the community.

The properties of E-elements are set by the designer in the first place but later accumulatively modified by the community. The function of these descriptive properties is many folded; first, to improve search and filtering operations and serve as a basic description for end-users. Second, allow semantic tests of the mash-up of multiple E-elements into application nodes.

3.3 Application Nodes and ELEPHANT Application (E-application)

An application node in the ELEPHANT system is a node in the workflow of an E-application. This workflow is presented through a graph (see Figure 8). In its simplest form an application node is a single E-element that is a sub-application following the MVC pattern. As mentioned before, the workflow of a mobile application has to consider dynamics caused by the changing use of context. When defining the workflow structure for the application the factor context has the potential to cause many forks in the structure.

The ELEPHANT system implements a mash-up concept to ease the definition of application nodes and the composition of the workflow structure for the application. Supposing that developers of E-applications have access to a repository of heterogenic E-elements where each is described in ELEPHANT's description language regarding the use of context (e.g. user presence, device presence, activity, task, connectivity, interaction modalities and web-services). Developers can mash-up/cluster multiple E-elements and content (e.g. sources of content, online as offline). On runtime the E-Interpreter chooses depending on the live context the matching E-elements and content sources.

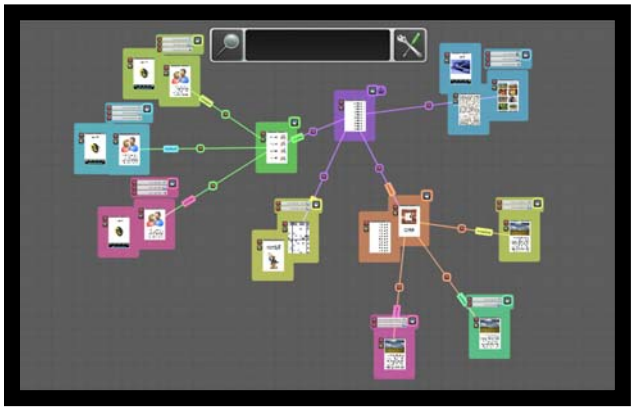

Figure 8: Screenshot of ELEPHANT's authoring tool

An E-application is an implementation of an application that is designed to assist the user stepwise with machine functionalities. The motivated user is supported to achieve a predefined goal; for example, prepare a specific meal, organize a party, help with dialogs in foreign languages. The assistance procedure may be composed of multiple steps that are implemented in a series of E-elements, in single E-elements or sub-routines provided by E-elements. The workflow structure of an E-application is a tree structure with application nodes as nodes of the tree. Each application node is a mashed up cluster of content-sources and E-elements. Figure 8 presents an E-application composed by our current editor (E-composer). We conducted a usability test of the E-composer to screen our preliminary ideas in implementing a user interface for editing E-applications that we plan to publish in a subsequent paper. The ELEPHANT system supports collaboration while editing an E-application. That is, users can load available E-application, modify components, structures or content to their liking and upload and tag it as their own version.

3.4 ELEPHANT Services (E-services)

E-services are web-services with SOAP APIs provided by the ELEPHANT System. Typically E-element's functionalities are encapsulated. Through E-services it is possible to implement smarter E-elements; for example, E-elements that store state information on a buffer provided by an E-service. In doing so it is possible to create adaptable E-elements. Trough E-service E-elements have access to data from earlier used E-elements and E-application. This is important when the assistance need to set up on earlier performances (e.g. history data) of the user (see also figure 3).

Other E-services are for managing user information (e.g. passwords). Some E-elements access extern web services that need a login and a password (e.g. flickr, facebook, amazon etc.). Each time such an E-element is used the user would have to type login and password before accessing the sources. An E-service that manages user passwords can provide a single sign on mechanism. Access to E-services is managed through IDs (e.g. user, E-element and E-application ID).

3.5 ELEPHANT Interpreter (E-interpreter)

The E-interpreter is an application running on the mobile device. It consists of two parts.

First, a virtual machine written in the native language of the mobile OS that provides access to sensor information on the device (e.g. VC++ on Windows Mobile devices). The virtual machine is a Socket Server accepting multiple clients (e.g. clients related to specific sensors) (see figure 9).

The second part is the manager. The manager is a client application that interprets the use of context (e.g. through interpreting device and user presence) and loads depending on the context E-elements from the workflow of the E-application. The manager provides E-elements access to sensor information by forwarding requests to the virtual machine and providing interoperability. The E-interpreter is the ELEPHANT system's local runtime environment.

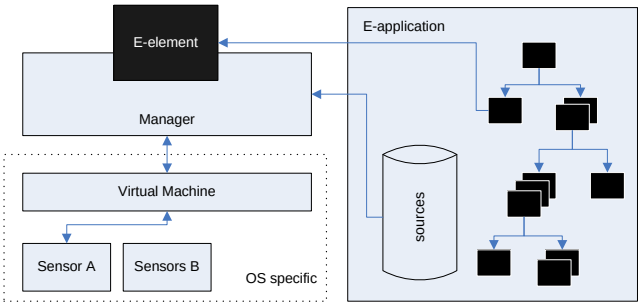

Figure 9: Runtime environment on the target mobile device

The E-interpreter has through the virtual machine access to state information of the mobile operating system's sensors e.g. state of Bluetooth, WLAN, Headset, Camera etc. We use the state information to set the device presence.

The Manager provides some essential functions to manage multiple mobile assistants running on the same device. First it provides a home screen allowing to choose an assistant (Figure 10.1) or to search (Figure 10.2) for existing mobile assistants. At

the home screen the user can also set his presence (see Figure 10.3). When the user chooses an assistant the first E-element is launched (see Figure 10.4); the user is able to see user and device presence on the top bar of the E-element.

There is only one button on the left upper corner to blend in the home screen. Each E-element in the structure of an E-application has a unique id. Each id matches a directory name in the directory structure that is set up for each assistant on the mobile device. All offline resources assigned to an E-element by the E-Composer application reside in the directory assigned to this particular E-element. This arrangement simplifies the management of E-elements, resources and their interplay.

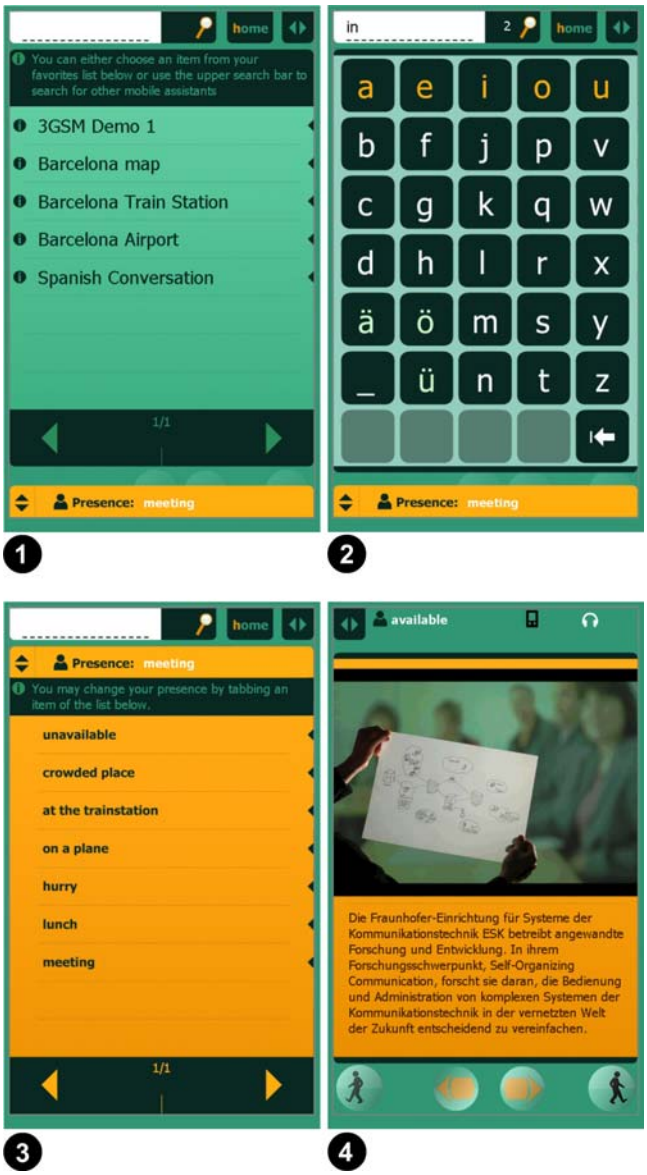

Figure 10: Screenshots of the E-Interpreter UI (1-3) and an E-element (4) on runtime

3.6 Implementation

For implementing our current system we use on the platform side Ruby on Rails and MySQL. E-elements and E-application are implemented in Flash (AS2).Our prototype editor (E-Composer) is implemented in Flash (AS3). We use Adobe's packager to build installable E-applications (e.g. packaged as sis and cab files). In our current E-interpreter the virtual machine is implemented in VC++ for a Windows Mobile device. We have also implemented a client to access the state information of the windows mobile operating system's sensors e.g. state of Bluetooth, WLAN, Headset, Camera etc. that we use to set the device presence. In our current implementation the E-interpreter uses only device and user presence to choose one E-element from an application node.

4. CONCLUSION AND FUTURE WORK

In this paper we presented our component based design process for mobile assistants. We described the theoretical framework for developing mobile assistants and a small-sized field study. We presented the ELEPHANT system where we realized our design process. We described the single components: E-elements, E-application, E-services and the E-interpreter.

In this paper, we did not describe a use case due to limited space; however, during our development we used typical eLearning scenarios to explain and create useful prototype demonstrators.

E-elements are currently developed with Adobe Flash plus a documentation of the additional API. We plan to provide an environment to emulate E-element functionalities to support development and test of E-interpreter APIs. The E-interpreter is currently using user and device presence information to choose an E-element out of an application node. In future we plan to evaluate other strategies (e.g. setting up an E-service to provide context interpretations). We will also evaluate in detail the strategies used by Daniel and Matera [9] and adapt ours based on these results. To manage the quality of ELEPHANT's components we will evaluate reputation models. We are currently planing a large-size study where we will provide open access to the ELEPHANT system and our authoring tool.

ACKNOWLEDGMENTS

This work was funded in part by the Bavarian Ministry of Economic Affairs, Infrastructure, Transport and Technology within the project „Dynamische Plattformen für Verteilte Systeme".

5. REFERENCES

[1] I. Aslan, M. Schwalm, J. Baus, A. Krüger, and T. Schwartz. Acquisition of spatial knowledge in location aware mobile pedestrian navigation systems. In MobileHCI '06: pp 105–108, 2006

[2] I. Aslan, F. Xu, H. Uszkoreit, A. Krüger, and J. Steffen. Compass2008: Multimodal, multilingual and crosslingual interaction for mobile tourist guide applications. INTETAIN, volume 3814 of Lecture Notes in Computer Science, pages 3–12. Springer, 2005.

[3] I.Aslan, F. Xu, H. Uszkoreit,A. Krüger and J. Steffen. The compass2008 smart dining service. In Proceedings of intel ligent Technologies for interactive Entertainment (INTETAIN), Italy, 2005.

[4] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan. The smart phone: a ubiquitous input device. Pervasive Computing, IEEE, 5(1):70–77, 2006.

[5] R. Ballagas, F. Memon, R. Reiners, and J. Borchers. istuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous 1 computing. In CHI '07, pages 1107–1116, New York, NY, USA, 2007. ACM.

[6] H. Beyer. Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[7] S. Bodker. Through the Interface: A Human Activity Approach To User Interface Design. CRC, 1 edition, 1990.

[8] T. Bohnenberger, O. Jacobs, A. Jameson, and I. Aslan. Decision-theoretic planning meets user requirements: Enhancements and studies of an intelligent shopping guide. In Hans Gellersen, Pervasive Computing: Third International Conference, pages 279–296. Springer, Berlin, 2005.

[9] F. Daniel and M. Matera. Mashing up context-aware web applications: A component-based development approach. In WISE '08: Proceedings of the 9th international conference on Web Information Systems Engineering, pages 250–263, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] H. Elmeleegy, A. Ivan, R. Akkira ju, and R. Goodwin. Mashup advisor: A recommendation tool for mashup development. In ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services, pages 337–344, Washington, DC, USA, 2008. IEEE Computer Society.

[11] G. Gay and H. Hembrooke. Activity-Centered Design: An Ecological Approach to Designing Smart Tools and Usable Systems (Acting with Technology). The MIT Press, 2004.

[12] B. Hartmann, S. R. Klemmer, M. Bernstein, L. Abdulla, B. Burr, A. Robinson-Mosher, and J. Gee. Reflective physical prototyping through integrated design, test, and analysis. In UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology, pages 299–308, New York, NY, USA, 2006. ACM.

[13] D. Heckmann. Ubiquitous User Modeling. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006.

[14] M. Jones and G. Marsden. Mobile Interaction Design. John Wiley & Sons, February 2006.

[15] V. Kaptelinin and B. A. Nardi. Acting with Technology: Activity Theory and Interaction Design (Acting with Technology). The MIT Press, October 2006.

[16] M. Kenteris, D. Gavalas, and D. Economou. An innovative mobile electronic tourist guide application. Personal Ubiquitous Computing, 13(2):103–118, 2009.

[17] K. Leichtenstern and E. Andre. User-centred development of mobile interfaces to a pervasive computing environment. In ACHI '08: Proceedings of the First International Conference on Advances in Computer-Human Interaction, pages 114–119, Washington, DC, USA, 2008. IEEE Computer Society.

[18] Y. Li, J. I. Hong, and J. A. Landay. Topiary: a tool for prototyping location-enhanced applications. In UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology, pages 217–226, New York, NY, USA, 2004. ACM Press.

[19] Y. Li and J. A. Landay. Activity-based prototyping of ubicomp applications for long-lived, everyday human activities. In CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, pages 1303–1312, New York, NY, USA, 2008. ACM.

[20] J. Sheridan, T. Ballagas, M. Rohn, and J. Borchers. The Smart Phone As Input Device for Ubiquitous Computing. In IEEE Pervasive Computing, 2005.

[21] M. Sharples. Big Issues in Mobile Learning. Report of a workshop by the Kaleidoscope Network of Excellence in Mobile Learning Initiative. University of Nottingham, 2006.

[22] H. Uszkoreit, F. Xu, W. Liu, J. Steffen, I. Aslan, J. Liu, C. Müller, B. Holtkamp, and M. Wojciechowski. A successful field test of a mobile and multilingual information service system compass2008. HCI , pages 1047–1056, 2007.

[23] R. Wasinger. Multimodal Interaction with Mobile Devices: Fusing a Broad Spectrum of Modality Combinations. IOS Press, Amsterdam, The Netherlands, The Netherlands, 2007.

[24] R. Wasinger, Antonio Krüger, and O. Jacobs. Integrating intra and extra gestures into a mobile and multimodal shopping assistant. In Pervasive, pages 297–314, 2005.

[25] M. Weiser. The computer for the 21st century. Scientific American, 265(3):94–104, September 1991. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html.

[26] J. Yu, B. Benatallah, F. Casati, F. Daniel, M. Matera, and R. Saint-Paul. Mixup: A development and runtime environment for integration at the presentation layer. In Luciano Baresi, Piero Fraternali, and Geert-Jan Houben, editors, ICWE, volume 4607 of Lecture Notes in Computer Science, pages 479–484. Springer, 2007.

[27] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera. A framework for rapid integration of presentation components. In WWW '07: Proceedings of the 16th international conference on World Wide Web, pages 923–932, New York, NY, USA, 2007. ACM.

[28] N. Zang and M. Beth Rosson. Web-active users working with data. In CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems, pages 4687–4692, New York, NY, USA, 2009. ACM.

[29] N. Zang, M. B. Rosson, and V. Nasser. Mashups: who? what? why? In CHI '08: CHI '08 extended abstracts on Human factors in computing systems, pages 3171–3176, New York, NY, USA, 2008. ACM.

[30] N. Zang and M. B. Rosson. What's in a mashup? and why? Studying the perceptions of web-active end users. In Visual Languages and Human- Centric Computing,, pages 31-38, VL/HCC 2008.

[31] http://www.esk.fraunhofer.de/elephant