2010

An Adaptive User Interface Framework for eHealth Services based on UIML

Joël Vogt
University of Fribourg, Joel.Vogt@unifr.ch

Andreas Meier
University of Fribourg, andreas.meier@unifr.ch

23rd Bled eConference
eTrust:
Implications for the Individual, Enterprises and Society
June 20 - 23, 2010; Bled, Slovenia

An Adaptive User Interface Framework for eHealth Services based on UIML

Joël Vogt

Department of Informatics, University of Fribourg, Switzerland

joel.vogt@unifr.ch

Andreas Meier

Department of Informatics, University of Fribourg, Switzerland

andreas.meier@unifr.ch

Abstract

New sensory technologies and smaller, more capable mobile devices open opportunities for pervasive computing in the healthcare sector. Patients as well as medical professionals are, from a information and communication technology (ICT) point of view, better equipped than ever before. Despite this, many hospitals and other healthcare service providers have yet to exploit the potential unleashed by these technologies. In this paper, we present a framework for adaptive user interfaces for home care and smart hospital services. The framework uses the current context to provide healthcare professionals or patients with simpler, more efficient user interfaces. In a home care environment, user interface adaption is needed to tailor user interfaces to patients needs and impairments. In a smart hospital, user interface adaption considers medical professionals' preferences and priorities. In addition, by using context to make input suggestions simplifies the input and limits the scope for errors. Our frameworks uses a model-based approach and includes the current context in the interface generation process.

Keywords: Human-Computer Interaction, eHealth, Adaptive User Interfaces, UIML

1 Motivation

The increasing complexity of the healthcare sector has created the need to improve efficiency, quality and security of healthcare services while maintaining cost effectiveness. The Economist Newspaper (2009) explains that the ICT investments are helping to drive costs down in the healthcare sector. The widespread dissemination of ICT enables more portable healthcare services that are customized to the individuals' need. For example, patients' physiological parameters can be monitored remotely and the data can be sent to a medical specialist, thus avoiding expensive checkups and allowing the patients to lead a normal life. The almost instant availability of this data allows medical

professionals to react immediately, instead of having to wait for the next medical checkup (The Economist Newspaper Limited, 2009). The dynamic environment in which healthcare processes execute, offers new challenges and opportunities for the ICT. The User Interface (UI) is the direct point of interaction between the medical professional or the patient and the application. It is thus of utmost importance that the user can easily execute his intention at the UI and interpret the results displayed.

The prevalence of smartphones, such as the Apple iPhone or smartphones running Google Android that have the versatility of computers as well as the reemergence of tablet PCs, offer new, flexible ways to interact with information. Additionally, wireless sensors to track patients physiological parameters and in- and outdoor location tracking systems (e.g. GPS and Ubisense[1]) have open new possibilites to improve healthcare, for patients as well as for medical professionals.

Therefore, interfaces that are developed for a single device type may no longer be adequate. For example, a workflow management system to support administrative workflows in a hospital will be used by different actors (e.g. physicians, nurses) and the same actor may use the application on multiple devices. Sainfort et al., (2007) emphasize the need for research in the development of adequate UI in the healthcare domain. They predict that "[. . .] interfaces will automatically adapt themselves to the user [. . .], task, dialogue, environment and input/output modes in order to maximize the effectiveness of the HCI" (Sainfort et al., 2007, p.675).

Model-based UIs offer a promising approach to automated UI generation in a multi-device environment. UI designers develop the models upon which the UIs are based. During run-time, the UI is generated by applying a set of transformation algorithms to different types of models.

In this research, we develop a framework for mobile and adaptive user interfaces for the healthcare domain. The framework is developed as part of the Location and Context Aware eHealth Infrastructure (LoCa) project (Fröhlich et al., 2009). The LoCa framework consists of two main layers: The adaptive Workflow Management System (WFMS) layer (developed with the OSIRIS Next framework (Möller, 2009) and Human-Computer Interaction (HCI) layer. We use of he shelf technologies. The LoCa framework runs on Google Android, an open source Operating System (OS) available on a number of mobile devices. The HCI framework uses a set of XML languages for the models and artifacts. The adaptive UI is based on UIML, a highly flexible markup language for the design of UIs, which is an OASIS standard and supported by several organizations in the private and academic sector.

Section 2 presents a simple use case to illustrate the use of the framework. Section 3 elaborates on the conceptual models, followed by a description of the HCI architecture in section 4. In section 5 related research is discussed. This paper closes with an outlook on future work in section 6.

2 Use Case: Patient Visit

To illustrate the application of the HCI framework, consider the following use case in a smart hospital: Each medical employee is equipped with a Android smartphone with touchscreen display. Additionally, they have access to workstations and may also use

[1] http://www.ubisense.net

tablet PCs. The smart hospital is equipped with the indoor location tracking system Ubisense. Each room has an entertainment multimedia set for each patient, to watch TV, access the internet or listen to music. Additionally, a Hospital Information System stores the patient record of every patient. The patient record contains information including the patient's personal information (e.g. name, date of birth), current room, admission date and medical history. Patients have their heart rate and blood oxygen saturation constantly monitored. These measuring devices are registered to a specific user and have a bluetooth interface. In addition, some of the devices for measuring additional physiological parameters (e.g. glucose meter) that are carried by a nurse are also equipped with a Bluetooth interface.

The patient Mrs. Thrace has injured her knee. She is scheduled for a knee surgery in the late afternoon. The orthopedist visits Mrs. Thrace after breakfast to discuss the surgery. He switches on his smartphone that runs the LoCa framework. The LoCa system looks up his patients that are in his proximity and present a list of patients to choose from, in this case Mrs. Thrace. The orthopedist selects Mrs. Thrace's electronic health record. The system knows, based on the orthopedist's user profile, his preferences and, based on the device profile of his smart phone, the device attributes, including the size of the screen. The system displays a scaled down X-Ray image of Mrs. Thrace's knee and additional textual information. The LoCa system recognizes Mrs. Thrace's multimedia set as a possible host to run the process. The orthopedist selects the multimedia set to which the curent task is seamlessly migrated. The LoCa system adapts the display to the new device constraints and displays the image in high resolution. Afterwards, at the end of the consultation, the orthopedist migrates the task back to his smartphone and takes a few notes. He then returns to his office. To finish his report, he migrates the task to his workstation, which is more suited for larger text inputs.

3 User Interface Models

A common approach in the development of adaptive interfaces is to use a set of models (e.g. for domain and context of use) to generate and further refine a user interface artifact or artifacts until the final user interface is adapted to best match the user's need, given the constraints set by the context in which the user operates (Jaquero et al., 2009). The adaption process is shown in figure 2.

The Cameleon reference framework proposes four different abstraction layers for adaptive interfaces for a step by step generation of a user interface. The user interface presented to the user is referred to as the Final User Interface (FUI). The FUI is rendered by the UI toolkit of the given platform, for example GTK+ or Java Swing. The FUI is derived from a Concrete User Interface (CUI). This model is basically the same as the FUI, but independent of the toolkit description language. The CUI is generated from the Abstract User Interface (AUI). The AUI describes the interface independent of interaction modalities and devices. The AUI is derived from a domain model (the task and concepts model combined) (Calvary et al., 2003). The following section discusses the models applied in the interface adaption process (i.e. task model, concepts model, context model and UI model) followed by a description of the automated adaption process.

3.1 Task Model

Task analysis captures the way a human plans to perform a given task. Humans naturally tend to decompose problems into subproblems. Thus a common way to structure

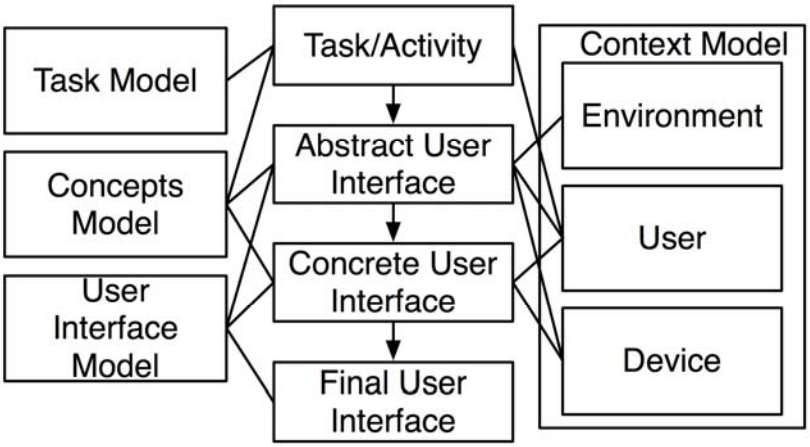

Figure 2. Models and artifacts in the adaption process

task models is a hierarchy. A number of tasks models have been developed. One of the more frequently used notation in the UI development is the ConcurTaskTrees (CTT) by Paternò et al., (2000). CTT has four types of tasks: system tasks that are conducted by the system alone, user tasks that are only allocated to a user, interaction tasks that are conducted by the user with the system and abstract tasks. Tasks are further divided in subtasks. Tasks are connected to their siblings through temporal operators. CTT also supports cooperative tasks that imply the action of more than one user. Van den Bergh and Coninx (2004) have extended the CTT notation to support dynamic context information. With this context-aware extension, this notation permits to model complex interaction and handle unexpected events that occur in a dynamic healthcare environment. For example, if a physician, who is currently visiting a patient, is required in the emergency room, the current activity can be suspended and the more urgent task suggested, if approved by the user, and loaded.

3.2 Concepts Model

The concepts model describes the objects with which the user interacts through tasks (Van den Bergh and Coninx, 2004b). Various healthcare related vocabularies for type level semantics and instance level semantics (e.g. patient care, index, cataloging biomedical literature) are available. Type level semantics describe the structure of information objects while instance level semantics structure the content of information objects (Beyer et al., 2004). For the prototype to validate the use case presented in section 4, we chose Clinical Document Architecture – Confoederatio Helvetica (CDA-CH) to describe type level semantics and ICD-10 for the instance level semantics. CDA-CH was favored for this research due to the fact that it has found wide acceptance in Switzerland and is easy to test in hospitals.

CDA-CH is an adaption of the XML based on Health Level 7 (HL7) Clinical Document Architecture (CDA) (release 2) healthcare documentation standard for Switzerland. A CDA-CH object consists of a header and a body. The body may contain a number of objects, including multimedia objects and references to external objects (HL7 Benutzergruppe Schweiz, 2009).

ICD-10 is a standard developed by the World Health Organisation (World Health Organization, 2007) which is used by many healthcare related institutes. We use this ontology to use a uniform vocabulary for the description of the patient's diseases.

3.3 Context Model

The context model stores the relevant information that is collected or managed by the system (Jaquero et al., 2009). Dey (2001) declares "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

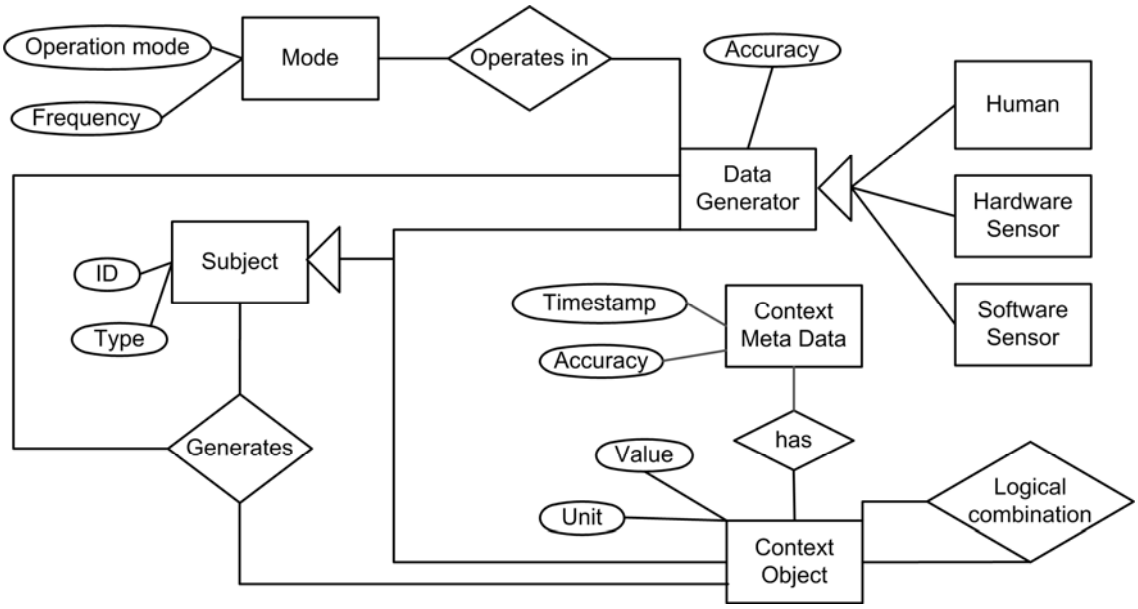

Figure 3. LoCa Context Model (obtained from Fröhlich et al., 2009)

The LoCa project provides an upper level context model (Fröhlich et al., 2009), shown in figure 3. A subject is any entity in the environment, such as a person, a device or an information object. In the LoCa model, subject can be further subdivided into two types: data generators which produce information about the current context and the context information themselves. A data generator can be a human (e.g. the patient who manually enters his weight), a hardware device (e.g. position devices or sensors) or a software process (e.g. monitoring daemon). Data generators can operate in a specific mode. The fact that context information are themselves subjects permits to annotate context data with further contextual information. For example, with this model, the location and time when a given measurement was taken, can be captured. The measurement is context data and at the same time the subject that is being further described with the date and a location. Context objects can be aggregated. The LoCa context model can further be refined by including domain specific models.

According to the Cameleon reference framework, context can be subdivided in three categories: user, device and environment (Calvary et al., 2003).

- The user model contains information about the current user. Generally speaking "the user's knowledge, interests, goals, background, and individual traits."

(Brusilovsky and Millán, 2007). We use a subset of the HL7 Reference Information Model to users, namely the "Person" class and the "Employee" and "Patient" class (HL7 Benutzergruppe Schweiz, 2009).

- The device model describes physical device and software attributes of the platform the framework is running on. Attributes include the CPU speed, screen size, sound capability, OS, etc. A standard for device models is the User Agent Profile Specification (Wireless Application Protocol Forum, Ltd., 2001).
- The environment model contains information about the environment in which the process executes (Jaquero et al., 2009).

3.4 Abstract User Interface Model

A generic vocabulary, a vocabulary that is platform and modality independent, will be developed for this framework containing abstractions suited for abstract interface objects in e-health. Using abstract vocabularies permits to support a wide variety of devices (Luyten et al., 2005; Ali, 2004). These classes are then mapped to the CUI for the UI toolkit that is used. We use two types: composite objects that contain other abstract user interface objects and atomic abstract user interface objects. Atomic abstract user interface objects are the leafs in a abstract user interface tree. Abstract composite user interface objects can either be structured when the order of the child objects matters or unstructured, if the order is arbitrary.

The abstract user interface model has to be extended relative to the concepts model. Thus in the context of this paper, the AUI model will be extended to include CDA-CH ontology objects (see section 3.2).

3.5 Concrete User Interface Model

The CUI model is platform dependent. It specifies the widget types and makes assumptions about the look and feel of the user interface. This model is then mapped to the FUI, in this case the Android UI. A UIML renderer for Android is being developed as a Bachelor thesis at the EDM, Hasselt University in Belgium.[2]

3.6 Final User Interface: Android

Android provides its own UI toolkit. The abstract classes are "View" and "View Group". A View Group object can contain several View objects. View Group is a sub class of View. New widgets can be added by third party developers. Android interfaces can either be defined in Java or in a XML vocabulary. Android separates the user interface in structure, style and content. View objects are arranged in a tree structure. The layout of view objects is defined with layout View objects. Layout views are subclasses of View group. View objects can be parametrized in a uniform manner with styles and themes (Google Inc., 2010).

4 The HCI Architecture

The HCI layer is decoupled from the workflow engine (see figure 4), thus it could easily be used with another workflow engine to add HCI process steps. The integration with the workflow engine is through message passing: whenever the workflow needs to exe-

[2] http://bramgoffings.blogspot.com/

cute a process step that requires human-computer interaction, it sends a message to the HCI module with the unique id of the task model it has to perform. The HCI module loads the requested task model and launches the task. The workflow engine waits for the HCI module to complete the step and resumes operations.

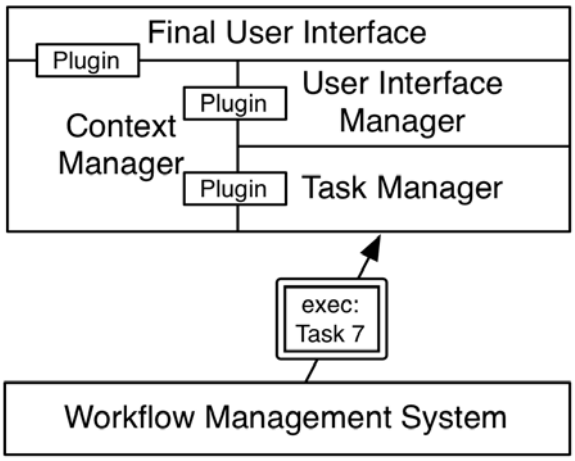

Figure 4. Architecture of the HCI Module

Separating the business process modeling and workflow execution from the human-computer interaction has several advantages:

- Reducing complexity: The business process description is not cluttered with atomic steps that the user has to perform, but are too fine grained for the process description. Modeling the user interaction separately reduces the complexity of the process model.
- Separation of concerns: A process engineer might not have the knowledge required for user interface design nor might he be interested in such details. With this approach, the process engineer and the user interface designer can focus on their area of expertise.
- Increased flexibility: The task model can be adapted without changing the overall process. The workflow engine only specifies the required input.

The architecture of the software consists of three tiers: the task manager, the interface manager and the presentation tier (see figure 4). Each layer receives information from the n−1th layer and provides information to the n + 1t h layer. Furthermore, each layer communicates with the Context Manager (CM). The CM is the single source of contextual information for the layers. A layer however does not interact directly with the CM. Instead, each layer uses a plugin that offers a uniform interface to the CM. This additional abstraction permits to replace the CM, without affecting the transformation layers in the framework.

The interface tier manages the user interface artifacts. When the task manager first pushes the task model, it compiles a UIML AUI based on the abstract vocabulary that was customized for the domain in which the process performs (in this case the patient visit). The creation of this UIML artifact is performed once every time the workflow engine requests a specific task model to be executed for a given workflow instance. During the execution of the HCI process step, the main purpose of this artifact is to

manage the structure and state of the interface (i.e. UI objects composition) and to apply changes to the interface if changes in the context demand it.

The interface tier maps the AUI into the CUI and propagates changes of the structure in the AUI to the CUI. From the CUI, the interface manager updates the FUI, which is managed by the platform dependent toolkit. The FUI is only presents an interface to the user and captures the user's input, which it directly passes to the context manager. The dialog flow is controlled by the interface tier.

4.1 User Interface Structure

A number of languages for abstract dialogue modeling are available. Popular examples are UsiXML and UIML.

- UsiXML has the advantages that it follows the abstractions layer of the Cameleon reference framework and offers a comprehensive toolkit. However, UsiXML cannot be extended without altering the language (Meskens et al., 2009).
- UIML is a markup language to create UI vocabularies. These are either mapped to concrete representations or to other UIML vocabularies.

UIML will be chosen for several reasons. First, UIML provides a clear separation of content, structure, behavior and style. UIML is modality-independent. Furthermore, it is meta language, thus does impose any mental models or dictate specific widget types. New vocabulary can be build without changing the language. Additionally, UIML uses XML, this facilitates the integration with other XML technologies. Finally, UIML is an open standard that is being further developed by a number of academic and private organisations (the current version under development is 4.0 (OASIS User Interface Markup Language (UIML) TC ,2009)).

UIML has been criticized for being too tightly coupled to the underlying modality for which a given vocabulary was designed (de Melo et al., 2009). This issue is however been addressed by improvements of the UIML standard and combination with other XML technologies (Helms et al., 2009).

UIML is based on the Meta-Interface Model (MIM). The MIM is divided in three components (Phanouriou, 2000):

- Presentation: The presentation component describes the interface vocabulary. It contains the classes that can be used in an interface, their properties and event handling. The presentation component maps the vocabulary classes to either concrete widgets or another UIML vocabulary.
- Logic: The logic components abstract the concrete Application Programming Interface that the interface component uses to communicate with the underlying application layer. The methods defined in this component are used by the interface component.
- Interface: The interface is described in an abstract way by separating the structure in different concerns. The elements that are used in the interface are described in the peers (the presentation and logic component).

4.2 Adaption Process

This section describes the adaption process for the user interface. The goal of this framework is to automatically optimize the user interface during run time to best meet the actors needs. To achieve this, at every stage transformation heuristics are applied to construct the next interface artifact until the FUI is presented to the user. The transformation rules are categorized in different types which are applied at different stages of the transformation process. Transformation rules are created by human experts and codify best practice patterns for predefined situations as well as fall back rules, as suggested by (de Melo et al., 2009). Rules at different stages can be developed independently of each other. At each stage, different contextual information are included. The transformation is based on the Cameleon framework. Figure 2 has given a high level overview of the models and artifacts involved in the UI generation. Our approach veers from the Cameleon framework by consulting the context at every step of the UI transformation. For example, the device model is already consulted to create the AUI, since different device types differ in screen size. Furthermore, the models are adapted during run-time and the context is applied to assist the user to articulate his input by offering a selection of input choices.

The task and concepts model form the basis for the user interface. The transformation rules generate an UIML abstract user interface based on the task and concepts model. This transformation step is already context dependent. The AUI is generated based on the device model (screen size of the device), user preferences and impairments and the amount of input information that can be presented based on the current context. The system then updates the AUI to present input value selections based on the current context. For example, if the nurse needs to enter the patient's blood sugar level and the system finds a glucose meter that is registered for the patient the nurse is currently treating, a abstract user interface object containing the value provided by the glucose meter will be added to the AUI. If the framework cannot find suitable context information to assist the user, the human data generator (i.e. the user himself) will have to provide the input. The framework adds an abstract input interface object.

Next, the AUI is mapped to the CUI. The device model is used to select the CUI UIML vocabulary that corresponds to the installed UI toolkit and to make device dependent adaptions (e.g. based on the screen size). The user model is used to adapt the layout handling to user's preferences for a customized arrangement of the user interface. Once the CUI is correctly parametrized, the UIML file is transformed to the FUI and presented to the user.

5 Discussion of Related Work

We see it as important to use context-aware adaption for user interfaces in the healthcare domain. The inclusion of the user and device model is required to adapt a user interface to varying needs of medical professionals and, in a home care scenario, to the patient's preferences and impairments. In addition, to use the context during run-time to help the user with the input selection, may simply and accelerate the interaction and reduce errors. Our approach fulfills these requirements. Moreover, by using UIML and clearly distinguishing between the UI architecture and the WFMS engine, the HCI framework presented is technology independent and therefore more portable.

A number of adaptive user interfaces and mobile healthcare applications have been developed in recent years:

- De Melo et al., (2009) propose an adaptive user architecture for board computers in cars, which allows to integrate different device types (e.g. smart phones, mp3 players) with the in-car environment. The goal is to access these devices through the uniform interface of an in-car system. Device only provide a task model and abstract user interface description. Their framework leans on the Cameleon framework. They describe the task model and in the same step the abstract user interface model, in UML. Transformation rules are used to map the abstract user interface to different modalities and optimise the final user interfaces. The user interfaces are described in their own OWL language. The system supports switching modalities during run-time and adding new devices. Unlike our approach, the abstract user interface is not derived from the task model. This, as the authors explain, makes transformation easier. However, the abstract user interface cannot be compiled in function of the current context. Additionally, user preferences and environmental information are not considered in the adaption.
- Martinez-Ruiz et al., (2008) developed a framework for adaptive user interfaces on mobile devices. They use the web pages, which need to be displayed correctly on a variety of devices, to make their case. Their approach uses UsiXML, CTT and the Cameleon framework. The transformation based on several iterations. They use the device model and the CTT to compile an abstract user interface suitable for the display. This is further refined for the platform type. Their approach differs from the framework presented in this paper in the UI language used and in the models consulted for the adaption. Our approach also includes the user model in the transformation and the models are adapted during run-time to the current context, for example to make input suggestions to the user.
- Portmann et al., (2010) are developing a smart environment, in which the user navigates by using a mobile device. The device receives context information from available sensors. In their approach, they use a fuzzy expert system to decide which information is relevant for the user based on the user model (his profile and location). Their adaption focuses on the content displayed in a dialog. The goal is to provide the data that meets the user's current profile and location. They do not, however, follow a clear separation of layers in their adaption framework. This makes support for different devices and output modalities is difficult. Nor does their system use the current context to aid the user with input selection.

The use of mobile devices in the healthcare sector have also been explored in a number of projects:

- Zao et al., (2008) developed HealthPal, a personalized smart phone to assist elderly people. Part of their system is a smart phone, optimized to ease interaction of the device for senior citzen. Senior citizen can use Health Pal to communicate with healthcare providers, receive medical instructions, manage their diet and measure their physiological parameters, which are sent to their healthcare provider.. The adaption of the user interface tries to lower the

complexity for the user. The adaption is based on the user model and the environment. It displays the menu items based on the user's activities focus and further arranged based on their frequency of use.HealthPal also unburdens the user by using the context. It can connect to the user's wearable biosensors and read the measured physiological parameters.

- The system presented by Zao et al. is is highly specialized for support of senior citizens. The framework is not designed with the possibility to be easily ported to different domains.
- Ardissono et al., (2008) developed a context-aware Service-Oriented Architecture framework for adaptive web-based applications that can adapt the UI and application logic based on the user's context. To test their approach, they implemented a e-health application to help coordinate activities required to monitor patients with heart diseases. The user interface and the workflow systems are coupled. The user interface is created based on a task description. In a first step, the system selects XSL Transformation artifacts developed for the user's current device. The user interface objects are grouped to fit the device screen and in function of the user's interests. The pages are then rendered from that representation for the target user. The system differs from our approach in several ways. Their system is web based, we do not predefine the final format. Furthermore, their framework differs in a sense that it does not clearly separate the abstract user interface and concrete user interface and does permit a more fine grained adaption and, while they too chose a modular approach, they do not emphasize the separation of workflow engine and user interface component.

6 Conclusion and Outlook

This paper presented an adaptive HCI framework for adaptive WFMS in the healthcare environment. Our goal is to facilitate the medical professional's daily work. By using contextual information, the UI is rendered in real-time to present the user with a consistent and relevant interface. The next steps of the ongoing research project is the development of a prototype for Google Android as a proof of concept. The prototype uses the UIML engine for Android, developed at the EDM. The prototype will be evaluated with healthcare professionals as part of the LoCa project.

Acknowledgement

We express our gratitude to Nadine Fröhlich, University of Basel, Dr. Anthony Dyson, Chief Information Officer at Medgate AG, Dr. med. Serge Reichlin, Head Health Innovation at Siemens Schweiz AG and Professor Dr. Thomas Strahm, University of Berne for their invaluable input.

References

Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., & Steggles, P. (1999). Towards a better understanding of context and context-awareness. In 1st international symposium on Handheld and Ubiquitous Computing, September 27-29, 1999 (304-307), London, UK: Springer.

Ali, M.F. (2004). A transformation-based approach to building multi-platform user interfaces using a task model and the user interface markup language. PhD thesis, Blacksburg, Virginia, USA.

Ardissono, L., Furnari, R., Goy, A., Petrone, G. & Segnan, M. (2008). A SOA-Based Model Supporting Adaptive Web-Based Applications. In Third International Conference on Internet and Web Applications and Services, June 18-23, 2008 (708–713). Washington, DC, USA: IEEE Computer Society Press.

Beyer, M., Kuhn, K.A., Meiler, C., Jablonski, S. & Lenz, R. (2004). Towards a flexible, process-oriented IT architecture for an integrated healthcare network. In 2004 ACM Symposium on Applied Computing, March 14-17, 2004 (264-271). New York, NY, USA: ACM.

Brusilovsky, P. & Millán, E. (2007). User models for adaptive hypermedia and adaptive educational systems. Lecture Notes in Computer Science, 4321 (2007), 3-53.

Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. Interacting with Computers, 15 (3), 289-308.

de Melo, G., Honold, F., Weber, M., Poguntke, M. & Berton, A. (2009). Towards a flexible UI model for automotive human-machine interaction. In 1st International Conference on Automotive User Interfaces and Interactive Vehicular Applications, September 1-22 2009 (47–50). New York, NY, USA: ACM.

Dey, A. K. (2001). Understanding and Using Context. Personal and Ubiquitous Computing. 5 (1), 4-7.

The Economist Newspaper Limited. April 16, 2009. Fixing health care. Last checked on November 11, 2009, from http://www.economist.com/opinion/displaystory.cfm?story_id=E1_TPQJNQSJ.

Fröhlich, N., Meier, A., Möller, T., Savini, M., Schuldt, H. & Vogt, J. (2009). LoCa - Towards a Context-aware Infrastructure for eHealth Applications. In 15th International Conference on Distributed Multimedia Systems (DMS'09), September 10-12, 2009 (52-57). Skokie, IL USA: Knowledge Systems Institute Graduate School

Goole Inc. (January 28, 2010) ANDROID developers User Interface. Last checked on February 10, 2010, from http://developer.android.com/guide/topics/ui/index.html.

HL7 Benutzergruppe Schweiz. (2009). CDA-CH. 1.2. Schwerzenbach, Switzerland.

Helms, J., Schaefer, R., Luyten, K., Vermeulen, J., Abrams, M., Coyette, A. & Vanderdonckt, J. (2009). Human-Centered Engineering Of Interactive Systems With The User Interface Markup Language. In A. Seffah, J. Vanderdonckt, & M. C. Desmarais (Eds.), Human-Centered Software Engineering (139-171). London, UK: Springer

Jaquero, V.L., Montero, F., Molina, J. P. & González, P. (2009). Intelligent User Interfaces: Past, Present and Future. In M. Redondo, C. Bravo & M. Ortega (Eds.), Engineering the User Interface: From Research to Practice (259-270). London, UK: Springer

Luyten, K., Thys, K. & Coninx, K. (2005). Profile-Aware Multi-Device Interfaces: An MPEG-21-Based Approach for Accessible User Interfaces. In Accessible Design in the Digital World, August 23-25, 2005. Dundee, Scotland, UK: Digital Media Access Group.

Martinez-Ruiz, F. J., Vanderdonckt, J. & Arteaga, J. M. (2008). Context-Aware Generation of User Interface Containers for Mobile Devices. In Ninth Mexican International Conference on Computer Science, October 6-10, 2008 (63-72). USA: IEEE Computer Society.

Meskens, J., Haesen, M., Luyten, K. & Coninx, K. (2009). User-Centered Adaptation of User Interfaces for Heterogeneous Environments. In M. Angelides, P. Mylonas & M. Wallace (Eds.), Advances in Semantic Media Adaptation and Personalization, Volume 2 (43-66). Boston, MA, USA: Auerbach Publications

Möller, T. December 9, 2009. OSIRIS Next. Last checked on February 9, 2010, from http://on.cs.unibas.ch/documentation.html

OASIS User Interface Markup Language (UIML) TC. (2009). User Interface Markup Language. 4.0. USA.

Paternò, F. (2000).Model-Based Design and Evaluation of Interactive Applications. London, UK: Springer

Phanouriou, C. (2000). UIML: A Device-Independent User Interface Markup Language. PhD thesis, Blacksburg, Virginia, USA.

Portmann, E., Andrushevich, A., Kistler, R. & Klapproth, A. (2010). Prometheus – Fuzzy Information Retrieval for Semantic Homes and Environments. In International Conference on Human System Interaction, May 13-15, 2010. IEEE Society

Sainfort, F., Jacko, J. A., Edwards, P. A. & Booske, B. C. (2007). Human-Computer Interaction in Health Care. In A. Sears & J. A. Jacko (Eds.), The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, Second Edition (661-678). USA: CRC Press.

Van den Bergh, J. & Coninx, K. (2004). Contextual ConcurTaskTrees: Integrating dynamic contexts in task based design. In Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, March 14-17, 2004 (13-17). Orlando, FL, USA: IEEE Society

Van den Bergh, J. & Coninx, K. (2004b). Model-based design of context-sensitive interactive applications: a discussion of notations. In 3rd Annual Conference on Task Models and Diagrams, November 15-16, 2004 (43-50). New York, NY, USA: ACM.

Wireless Application Protocol Forum, Ltd. (2001). User Agent Profile Specification 20-Oct-2001. Mountain View, CA, USA: Open Mobile Alliance Ltd.

World Health Organization. (2007). International Classification of Diseases. Last checked on January 25, 2010, from http://www.who.int/classifications/icd/en/

Zao, J. K., Fan, S. C., Wen, M. H., Hsu, C. T., Hung, C. H., Hsu, S. H. & Chuang, M. C. (2008) Activity-Oriented Design of Health Pal: A Smart Phone for Elders' Healthcare Support. EURASIP Journal on Wireless Communications and Networking. 2008, 1-10.