View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Ghent University Academic Bibliography

Adaptive User Interface Support for Ubiquitous Computing Environments

Heiko Desruelle[1], Dieter Blomme[1], George Gionis[2], and Frank Gielen[1]
[1] Ghent University – IBBT, Ghent, Belgium
[2] National Technical University of Athens – DSS lab, Athens, Greece
{heiko.desruelle, dieter.blomme, frank.gielen}@intec.ugent.be, gionis@epu.ntua.gr

ABSTRACT
Application developers are increasingly facing the need to cover a wider variety of target devices. The diversity of devices supporting software applications is expanding from PC, to mobile, home entertainment systems, and even the automotive industry. Maintaining a viable balance between development costs and market coverage has turned out to be a challenging issue when developing applications for such a ubiquitous ecosystem. In this paper, we present the webinos approach as a means to enable adaptive user interfaces for web-based applications in ubiquitous computing environments. We propose a model-based user interface adaptation framework driven by a rich description of the target delivery context.

Author Keywords
Adaptation, context awareness, abstract user interface.

ACM Classification Keywords
D.2.2 Software Engineering: Design Tools and Techniques—*User interfaces*; H.5.2 Information Interfaces and Presentation: User Interfaces—*User-centered design*

General Terms
Design, Human Factors.

INTRODUCTION
The availability of connected devices is growing rapidly. In our everyday life, we already use a multitude of personal devices that are connected to the Web. The number of shipped smart-phones at the end of 2010 even surpassed the traditional computer segments for the first time in the US [10]. From PC, to mobile, to home entertainment and even in-car units, consumers should prepare for a connected experience. Through the Web, applications can be accessed whenever and wherever the user wants, regardless of the type of device that is being used. However, from an application development perspective, ubiquitous environments generally introduce challenging and time-consuming requirements. The variety of presentation and interaction modalities make it very hard to support a wide range of devices. Even with standardized and cross-platform web technologies such as HTML, CSS, and JavaScript, efficiently managing ubiquitous variability points remains an important ongoing research topic [9].

Web-based software systems are traditionally modeled by separation of concerns. The models are engineered along three orthogonal dimensions: the development phases, the system's views, and its aspects (as illustrated in Figure 1) [12]. The phase dimension sets out the different stages of web development, ranging from analysis, to design and implementation. Each of these phases requires a number of specific views addressing the system's content, its navigation structure, and its presentation to the end-user. Finally, the aspects dimension defines both the structural and behavioral aspects of each of the view models. The growing importance of ubiquitous applications emphasizes the need for fragmentation management within this web engineering model. This concern has to be handled throughout every stage of the application's development life cycle. As proposed by Kappel et al., adaptability can be considered as an additional web engineering dimension, crosscutting all three other web modeling dimensions [11].

For developers, the straightforward incorporation of such adaptability still remains an important challenge [17]. Ubiquitous applications should adapt dynamically to the current context of use, and even to contextual situations not foreseen at the application's design time. From this perspective, the webinos project aims to deliver a platform for web applications across mobile, PC, home media and in-car de-

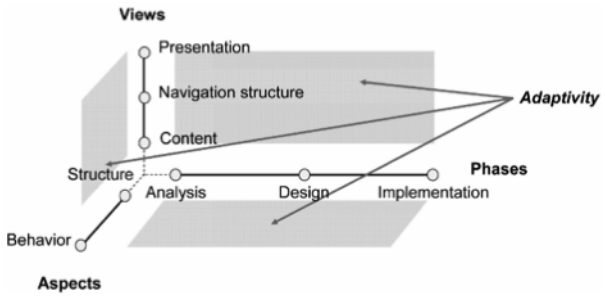

Figure 1. Adaptability as a crosscutting aspect on the traditional modeling dimensions of web engineering (from Koch et al. [12])

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
UIDL'2011, September 6, 2011, Lisbon, Portugal.

vices [22]. Webinos is a service platform project under the European Union's FP7 ICT Programme. The project represents a leap forward as a federated web runtime that offers a common set of APIs to allow applications to easily access cross-user, cross-service, and cross-device functionality in an open yet secure manner. Within the webinos project, work is being done to achieve a maximum level of independence from the various underlying operating systems and hardware. The aim of our research is to propose a number of frameworks that enable the development of self-adaptive ubiquitous web applications. The Model Driven Development (MDD) approach provides very useful support for this type of challenge. In this paper, we focus on enabling the adaptability of user interfaces according to an approach derived from the CAMELEON Reference Framework (CRF) [3]. We achieve this goal by incorporating webinos runtime support for context-aware transformation of abstract user interfaces description models.

The remainder of this paper is structured as follows. Section 2 discusses related work. Section 3 provides background on the webinos project and discusses the core web application runtime platform. Section 4 elaborates on the webinos approach in offering adaptive user interface support to application developers. In section 5 we discuss a case study in the e-learning domain for providing learning assistance to students with a disability. This use case demonstrates the goal of our approach in order to reach adaptability support that is driven by various contextual dimensions. Finally, future work and our conclusion are presented in Section 6.

RELATED WORK

The CAMELEON Reference Framework (CRF) is a result of the EU-funded FP5 CAMELEON Project [4]. The framework defines a context-sensitive user interface development process. The process is driven by an intrinsic notion of the current user context, the environment context, as well as the platform context. According to the CRF approach, user interface (UI) development consists of four subsequent stages:

1. Specification of the task and domain model, defining a user's required activities in order to reach his goals.
2. Definition of an abstract user interface (AUI) model. The AUI model expresses the application's interface independently from any of the delivery context attributes.
3. Definition of a concrete user interface (CUI) model, which generates a more concrete description of the AUI by including specific dependencies to the delivery context.
4. Specification of the final user interface (FUI), corresponding with the user interface in its runtime environment.

Figure 2 shows the interconnections and transformations between the above-mentioned CRF stages. The downward arrows depict reification processes. Reification is the transformation from a higher-level abstraction to a lower-level abstraction phase, hence inferring a more concrete UI description. The upward arrows, on the other hand, specify the abstraction processes. An abstraction is the inverse transformation of reification. The third transformation type is the translation, depicted by the horizontal arrows. The translation deals with adapting the UI description to changes in one of the context of use models. In this case, the UI description's abstraction level remains the same when performing a translation.

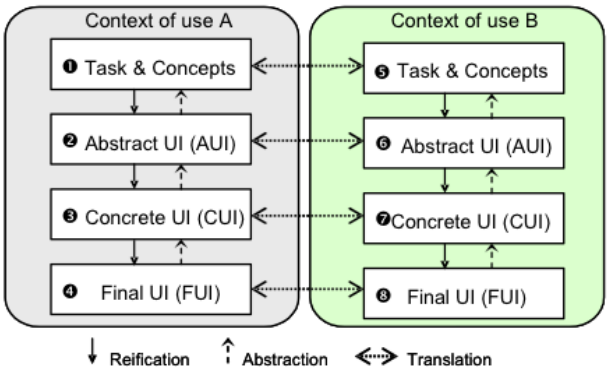

Figure 2. Context-aware UI development according to the CAMELEON Reference Framework (from Calvary et al. [3])

The Morfeo MyMobileWeb project [14] offers a framework that simplifies the development of web-based applications. MyMobileWeb specifically focuses on the mobile ecosystem. Alternative device spaces are not addressed by the project at this point. The framework applies a model-based approach to enable an automated application adaptation process based on the target delivery context. MyMobileWeb uses the IDEAL2 language [6] to enable higher abstraction levels during the application's development. Adaptation decisions are made at runtime, based on a contextual description of the target platform. The context space is modeled according to the W3C Delivery Context Ontology [5]. The mappings between the AUI and CUI descriptions are expressed using the standardized syntax of Cascading Style Sheets Level 2 (CSS2) [1].

The Model-Based UI Working Group (MBUI WG) [21] is a recently chartered W3C working group as part of the consortium's Ubiquitous Web Activity (UWA) [18]. Its goal is to work on standards that enable the authoring of context-aware user interfaces for web applications. The MBUI WG aims to achieve this type of adaptivity by means of a model driven design approach. In this context, the semantically structured aspects of HTML5 will be used as key delivery platform for the applications' adaptive user interface.

THE WEBINOS PLATFORM

Webinos is an EU-funded project. The project aims to define and deliver an open source platform, to enable web-based applications and services to be used in a consistent and secure manner over a broad range of connected devices. The supported devices range from mobile, to desktop, to home entertainment systems, and in-car units. In order to support this variety of devices and minimize the required efforts for application developers, webinos upholds a "single service for every device" ideology. Figure 3 depicts a high-level overview of the webinos platform structure. The sys-

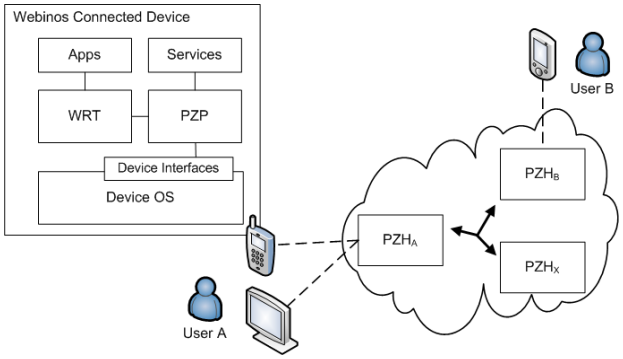

Figure 3. High-level overview of the webinos ubiquitous application runtime platform

tem's components are spread over the devices, as well as the cloud. The cloud components represent an important aspect of the platform, as these components enable users to access applications and services regardless of their device's physical boundaries. This seamless interconnection principle is centered around the notion of a so called Personal Zone. The Personal Zone groups a user's personal devices and services. To enable external access to devices and services in the zone, the webinos platform defines the Personal Zone Hubs (PZH). Each user has his/her own PZH running in the cloud. This facilitates access to someone's services over the Web from other devices. The PZHs are federated and provide support for discovering other people's hub, allowing users to easily share data and services. Although the system is designed with a strong focus on taking benefit from online usage, all devices in the Personal Zone have access to a local context model. This allows users to still operate their applications when being offline, or temporarily unable to access the Internet. Webinos provides offline support through the Personal Zone Proxy (PZP) component on the device. The PZP acts in place of the PZH when no Internet access is available.

The webinos Web Runtime (WRT) component can be considered as an extension to a traditional browser. It is capable of rendering web applications specified using standardized web technologies such as HTML, CSS, and JavaScript. The webinos WRT maintains a tight binding with the local PZP. This binding allows the webinos WRT to be much more powerful than traditional browser-based application environments, as it enables the runtime to interface with local device APIs and services. Moreover, the PZP also allows the system to connect and synchronize with other webinos devices through its binding with the PZH.

CONTEXT-DRIVEN USER INTERFACE ADAPTATION

As described in the previous section, the webinos platform uses the web to provide a rich delivery channel for ubiquitous applications. Nevertheless, there is still a need for optimizing applications to their specific delivery context. We use the Context-Aware Design Space (CADS) to visualize the required degree of application adaptability that is aimed for by the webinos platform (see Figure 4). The CADS is proposed by Vanderdonckt et al. [20] as a means to analyze and compare a software system's dimensions subject to adaptation. By default, the design space defines seven adaptation dimensions, but it can be extended with additional dimensions based on the application domain's specific needs. As shown in Figure 4, the strong fragmentation of ubiquitous computing in terms of interaction methods, hardware characteristics, software capabilities, etc. clearly requires a high level of adaptability.

In order to fulfill the requirements stated in the webinos CADS regarding user interface adaptability, we propose to incorporate a model driven approach derived from the CRF approach. Webinos aims to support both the reification and translation transformations from the AUI abstraction level and lower (i.e., CUI and FUI level). Webinos enables this process by means of two framework components: the context framework, and the abstract user interface framework. Both frameworks are part of the runtime platform. The two platform components are described in more detail further on this section. The webinos UI adaptation process is realized as a dynamic rendering process. The at runtime adaptation behavior potentially implies a number of important performance issues. Nevertheless, this design decision is made due to the enormous variety of devices and contextual attributes covering the ubiquitous ecosystem.

Webinos Context Model and Framework

Context management is an important aspect of the webinos platform. The webinos context model comprises four top-level submodels: the user context, the device context, the environment context, and the application context. The first three models are internally managed by the webinos system, whilst the application context model provides developers the opportunity to structure a situation's contextual description from their application's perspective. In addition, structuring the application's context information allows users to more easily open up data access from other devices and services, or even allow access from the outside world (e.g., by friends, family, collegues, etc.). The webinos context framework is built on top of these models. The framework is one of the core webinos service (see Figure

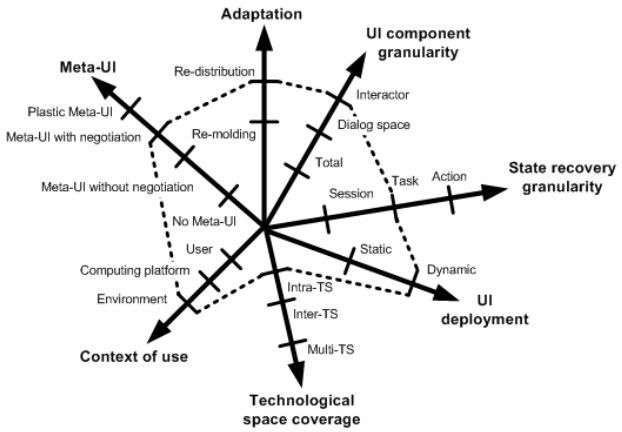

Figure 4. Context-aware design space (CADS) visualizing the adaptability levels supported by the webinos platform for each of the design dimensions

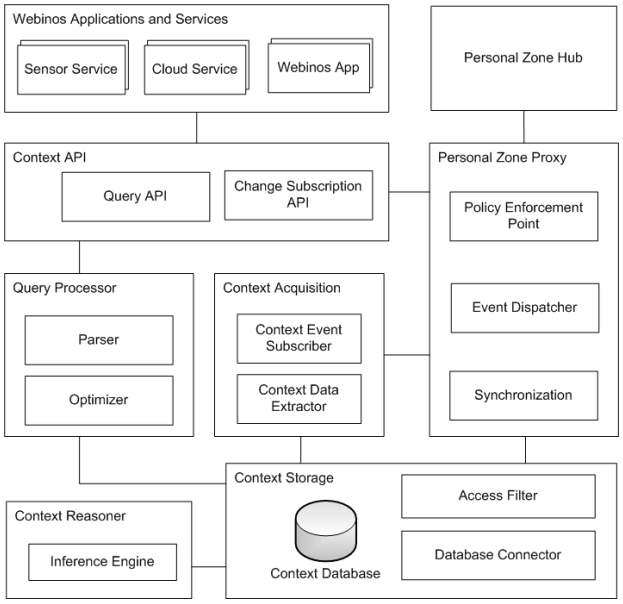

Figure 5. Architectural overview of the webinos context framework

3) and provides the necessary functionality for acquiring, storing, and inferring rich contextualized data. Other webinos applications and services rely on this framework to support the need for context-awareness during the execution of their operations. The framework is responsible for extracting and storing context data through the identification of specific context-related events that happen within webinos-enabled devices. In addition, the framework provides applications with an API to access such context information either by querying against the in-storage data or by being notified in real time regarding specific context changes. The context framework is closely coupled with the webinos policy and privacy enforcement framework in the PZP. This binding aims to ensure the secure handling of the often highly sensitive context data that is being stored and accessed. Figure 5 describes the conceptual architecture of the webinos context framework.

The Context API constitutes a component that enables applications to access the underlying volume of contextual data in a uniform way. The API provides interfaces that support two different modes for accessing context information:

- The Query API enables applications to execute targeted queries for specific context data in the storage system.
- The Change Subscription API enables applications to subscribe for specific events that are triggered by a contextual change within webinos. Subsequently, subscribed applications are notified in real time when such events occur.

API access requests are passed to the Query Processor. The processor parses the request and checks its execution rights in collaboration with the PZP. In case the request is granted by the PZP, the query is optimized and executed. Besides acting as policy enforcement point, the PZP is also responsible for dispatching context events to the Context Acquisition

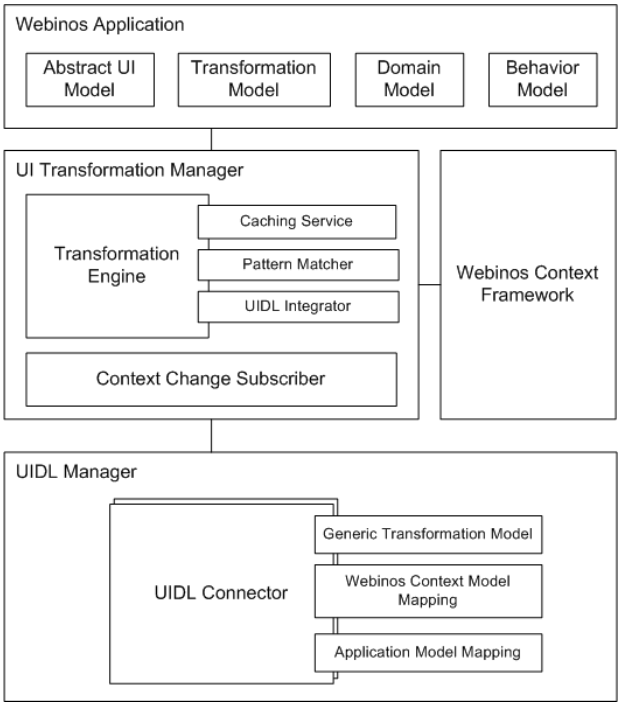

Figure 6. Architectural overview of the webinos adaptive user interface framework

component, and synchronizing contextual data between the local device and the Personal Zone (cfr. description of the general webinos architecture in Section 3).

Webinos User Interface Framework

The UI Framework is another core webinos service (see Figure 6). It enables application developers to focus the design of their user interfaces on the AUI abstraction level. The framework takes care of processing UI-related platform and context dependencies by using the AUI model to drive the CUI generation in a (semi-)automated way. Whilst the framework is capable of generating CUIs without human intervention, webinos wants to keep developers in the loop and enables them to specify their own transformation requirements throughout the entire UI generation process. The framework addresses this requirement by providing plugable support for User Interface Description Languages (UIDL). The extensible nature of the framework allows developers to use their preferred CAMELEON Reference Framework-compliant UIDL (e.g., UsiXML [19], or MariaXML [16]) to define their application's user interface. The only requirement to use a specific UIDL is the presence of a connector component for this particular UIDL in the webinos UI framework. The UIDL-specific connectors are required in order to enable the framework's automated transformation inference.

The UI Transformation Manager component is at the heart of the framework. The component drives the transformation processes, aiming for a contextually optimized final user interface through a series of reification and translation operations. The Transformation Manager comprises an inference

engine to attain this goal. The webinos UI transformation process relies on inference through dynamic pattern matching. Each transformation rule Φ within the transformation model can be represented as a conditional substitution operation

$$\Phi := P \; ? \; [S] \; : \; [T] \, . \tag{1}$$

The predicate P in Equation 1 is used to set the required condition before executing the transformation operation. The actual transformation is expressed in terms of the substitution $S = e\,[l := r]$, consisting of an expression e which is matched for pattern l that in turn is substituted by the expression r. T is an optional substitution, executed in case the required condition P is not met. The structural patterns are searched for in the AUI model. This matching step is performed in order to detect structural abstractions (i.e., widget structures) within the application's user interface description. The transformation's precondition expression P, on the other hand, is based on the variables in the context model and the application's domain model. The transformation rules are selected from the transformation model. In case multiple transformation rules match the discovered UI pattern, the default conflict resolution strategy is to allocate higher priority to rules defined by the application developer. This way, application developers maintain maximum control over the transformation process.

CASE STUDY: LEARNING ASSISTANCE FOR DISABLED STUDENTS

In this section, we demonstrate the concepts of our approach based on a case study from the e-learning domain. The presented case aims to provide optimized e-learning facilities to students with a certain disability. For students with a disability, learning assistance services can be indispensable to the successful pursuit of education. In general, course material can significantly benefit from accessibility adaptations for sighted, blinds, hearing impaired persons, etc. This process is very resource consuming and often requires the allocation of dedicated caretakers. Technologies such as Braille readers, text-to-speech (TTS), and speech recognition have considerably increased the accessibility of user interfaces. Using these technologies, disabled persons are given the opportunity to become more independent from their environment. Nevertheless, every type of disability imposes its own usability and structuring requirements on applications. Providing end-users with an optimal experience requires developers to address each of these requirements individually. This approach leads to ad hoc development processes, in which developers have to create and maintain multiple versions of their learning application for each specifically supported disability.

This case study emphasizes the need for adaptability processes driven by the user context, as well as the device and environment context. The remainder of this section will elaborate on the dynamic adaptation of an e-learning AUI into a contextually optimized CUI, performed within the webinos

```
<xforms xmlns:dm="http://www.myapp.com/domainmodel">
<group ref="dm:enrollment/student">
  <group ref="dm:enrollment/student/name">
    <input ref="dm:enrollment/student/name/first-name">
      <label>Your first name</label>
    </input>
        ...
  </group>

  <input ref="dm:enrollment/student/born">
    <label>Your birthday</label>
  </input>

  <group ref="dm:enrollment/student/address">
    <label>Home address</label>
        ...
  </group>

  <select ref="dm:enrollment/student/maritial-status">
    <label>Your maritial status</label>
      <item>
        <label>Single</label>
        <value>single</value>
      </item>
        ...
      <item>
        <label>Maried</label>
        <value>maried</value>
      </item>
  </select>
        ...

  <submit submission="dm:enrollment">
    <label>Submit</label>
  <submit>

</group>
</xforms>
```

Figure 7. Partial representation of the AUI model for a context-aware e-learning application using the W3C XForms standard

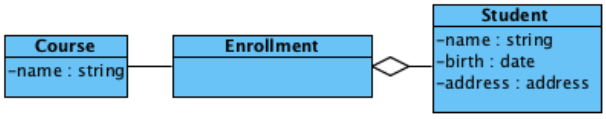

Figure 8. Simplified representation of e-learning application's domain model

UI framework. We will examine the case in which a user needs to be presented with an interactive form-based user interface. Form-filling is a frequently performed action in e-learning environments (e.g., online assessments, submitting assignments, etc.).

Abstract User Interface Model

An application developer starts the UI design process by defining the AUI model. The developer can choose any CRF-compliant description language, as long as the webinos UI framework contains the necessary connector component for that specific UIDL. In this case, the developer needs to create an abstract form-based user interface for applying to a certain e-learning course and decides to use W3C XForms [2] as abstract UI description language. The XForms description provides an overview of which control types should be presented in the UI, whilst remaining at a high abstraction level and not specifying how to exactly display these controls. A partial representation of the application form's AUI is shown in Figure 7. The form controls are abstracted to a level of input and selection components. Grouping controls provide a

high level means of hierarchical interface structuring. These grouping controls will also help in the detection of widget structures by the UI Transformation Engine. Moreover, all controls contain a reference to their corresponding entity in the application's domain model (see Figure 8, and the webinos UI Framework description in Section 4). For form-based interfaces these bindings provide additional semantics regarding structure, data types, data ranges, etc. This information can in turn be exploited by the UI Transformation Engine as rich a-priori knowledge concerning the adaptation process.

Transformation Model

The system's transformation model encompasses the reification and translation steps in order to obtain a final user interface that is optimized for the end-user's current context of use. The webinos UI Framework enables developers to fall back on a generic set of transformations provided by the platform. The standard set of transformations ranges from screen-fitting media adaptations, to adaptations based on generic accessibility and resource saving best practices [7][8]. On the other hand, application developers are still able to maintain control over the transformation process. Developers are free to refine their own transformation sets, specific to the application domain. As elaborated in Section 4, reification and translation rules can be expressed in terms of conditioned substitutions. Substitutions define structural changes to the AUI model. Furthermore, the link between the structural transformation and the conditionally required context attributes is realized through the specification of the transformations' conditional predicate description. These transformation preconditions can contain references to variables within the application's domain model, as well as the webinos Context Framework model. Nevertheless, based on privacy and security considerations, model queries are only permitted as long as the Personal Zone Proxy grants the application access to that specific type of information.

The process of filling in forms with a mobile device can be tedious task, as these devices lack the presence of a decent keyboard [15]. A potential translation rule for a form-based user interface in a mobile delivery context could be to match the AUI for optional input controls. If such a pattern is detected, the Transformation Manager can be instructed to remove the matching controls from the AUI. Alternatively, the translation rule can state to make such optional controls less obtrusive by rearranging them to the back of the form. Moreover, at the reification transformation level, the AUI can be transformed to a CUI with a look-and-feel matching the target device context.

On the other hand, taking the user context into account is also an important aspect. As indicated above in the case study description, the use of abstract user interfaces can support the process of optimizing UIs for people with specific disabilities. A useful reification rule in this context can be to transform AUI structures to a VoiceXML-based [13] CUI in case the end-user is blind or sighted. Figure 9 depicts the code snippet of a VoiceXML-based form filling CUI model after application of such reification ruleset. In the same way,

```
<vxml xmlns="http://www.w3.org/2001/vxml">
<form name="enrollment">
  <field name="course">
    <prompt>
      Specify your course enrollment
    </prompt>
    <grammar src="elearning.xml#courses"
             type="application/grammar+xml"/>
  </field>
    ...
  <field name="birthday" type="date">
    <prompt>
      Enter your birthday
    </prompt>
  </field>
    ...
  <field name="maritial-status">
    <prompt>
      Specify your maritial status
    </prompt>
    <option>Single</option>
    <option>Married</option>
  </field>
    …
  <block>
    <submit />
  </block>
</form>
</vxml>
```

Figure 9. Partial representation of the VoiceXML-based CUI model for a context-aware e-learning application

motor impaired persons can be supported. E.g, by transforming the default rendering of form interaction controls in order to simplify their selection.

CONCLUSION AND FUTURE WORK

It is essential to provide end-users with an UI design that is optimized to their specific context of use. When developing for ubiquitous computing environments, this requirement is even further emphasized. It is not a sustainable business to require developers to manually address all variability points of the ubiquitous ecosystem. Such an approach only leads to developers rapidly loosing market share, and on the other hand entire consumer segments being ignored due to their marginal revenue potential. Hence, the development of ubiquitous applications requires a higher level of abstraction.

In this paper, we presented the webinos platform approach as a means to support adaptive ubiquitous user interfaces. We propose an adaptive UI framework driven by a detailed description of the target delivery context. The framework handles the processing of context related user interface dependencies, whilst still providing developers the means to keep control over the adaptation process. The UI adaptation is performed on a high abstraction level through the incorporation of the CAMELEON Reference Framework (CRF).

We are currently working on a reference implementation of the webinos platform. The extensive evaluation of our platform has yet to be carried out. An iterative evaluation process is planned throughout the implementation. Various test groups will be addressed in order to validate our approach from the perspective of developers as well as end-users.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7-ICT-2009-5) under grant agreement number 257103.

REFERENCES

1. Bos, B., Celik, T., Hickson, I., Lie, H.W.: Cascading Style Sheets Level 2. Available at http://www.w3.org/TR/CSS2
2. Boyer, J.M.: XForms 1.1. Available at http://www.w3.org/TR/xforms
3. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J.: A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, pp. 289–308 (2003)
4. Cameleon project, http://giove.isti.cnr.it/projects/cameleon.html
5. Cantera, C.M., Rhys, L.: Delivery Context Ontology. Available at http://www.w3.org/TR/dcontology
6. Cantera, C.M., Roderiguez, C., Diaz, J.L.: IDEAL2. Available at https://files.morfeo-project.org/mymobileweb/public/specs/ideal
7. Chisholm, W., Vanderheiden, G., Jacobs, I.: Web Content Accessibility Guidelines 1.0. Available at http://www.w3.org/TR/WAI-WEBCONTENT
8. Connors, A., Sullivan, B.: Mobile Web Application Best Practices. Available at http://www.w3.org/TR/mwabp
9. Frederick, G.R., Lal, R.: The future of the mobile web. Beginning smartphone web development, pp. 303–313, Springer (2009)
10. International Data Corporation (IDC), http://www.idc.com/research
11. Kappel, G., Proll, B., Retschitzegger, W., Schwinger, W.: Modeling ubiquitous web applications: the WUML approach. Conceptual Modeling for New Information Systems Technologies, pp. 183–197, Springer (2002)
12. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-Based web engineering. Web engineering: modeling and implementing web applications, pp.157–191, Springer (2008)
13. McGlashan, S., Burnett, D.C., Akolkar, R., Auburn, R.J., Baggia, P., Barnett, J., Bodell, M., Carter, J., Deshmukh, M., Oshry, M., Rehor, K., Yang, X., Young, M., Hosh, R.: Voice Extensible Markup Language (VoiceXML). Available at http://www.w3.org/TR/voicexml30
14. Morfeo MyMobileWeb project, http://mymobileweb.morfeo-project.org
15. Nakagawa, T., and Uwano, H.: Usability Evaluation for Software Keyboard on High-Performance Mobile Devices. In: Proceedings of HCI International 2011, pp. 181–185, Springer (2011)
16. Paterno, F., Santoro, C., and Spano, L.D.: MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. ACM Trans. Comput.-Hum. Interact., pp. 324–353 ACM (2009)
17. Schauerhuber, A., Wimmer, M., Schwinger, W., Kapsammer, E., Retschitzegger, W.: Aspect-oriented modeling of ubiquitous web applications: the aspectWebML approach. In: 14th International Conference and Workshops on the Engineering of Computer-Based Systems, pp. 569–576 (2007)
18. W3C Ubiquitous Web Applications Activity, http://www.w3.org/2007/uwa
19. UsiXML, http://www.usixml.org
20. Vanderdonckt, J., Coutaz, D., Calvary, G., Stanciulescu, A.: Multimodality for Plastic User Interfaces: Models, Methods, and Principles. In: Multimodal user interfaces: signals and communication technology, Lecture Notes in Electrical Engineering, pp. 61–84 Springer-Verlag (2007)
21. W3C Model-Based UI Working Group Charter http://www.w3.org/2011/01/mbui-wg-charter.html
22. Webinos project, http://www.webinos.org