Profile-Aware Multi-Device Interfaces: An MPEG-21-Based Approach for Accessible User Interfaces

Kris Luyten, Kristof Thys and Karin Coninx
Hasselt University
Expertise Centre for Digital Media
Wetenschapspark 2
B-3590 Diepenbeek, Belgium
{kris.luyten, kristof.thys, karin.coninx}@uhasselt.be

The wide diversity of consumer devices has led to new methodologies and techniques to make digital content available over a broad range of devices with minimal effort. In particular the design of the interactive parts of a system has been the subject of a lot of research efforts because these parts are the most visible and are critical for the usability (and thus use) of a system. One thing that is missing in many current approaches is the ability to combine these new methodologies and techniques with a user-centric approach to ensure preferences from and requirements for a specific user are taken into account besides the device adaptations. In this paper we analysed the applicability of MPEG-21, part 7: Digital Item Adaptation, for the adaptation of a user interface to user characteristics. We show how the high-level XML-based user interface description language UIML in combination with an MPEG-21-based user profile enables designers to create accessible and personalised multi-device user interfaces. Using this combination results in user interfaces that can be deployed on a broad range of devices while taking into account user preferences with minimal effort.

This approach enhances accessibility to digital items on various platforms, since all interactions with digital items should be supported by an appropriate user interface.

User Interface Descriptions, XML, MPEG-21, user profiles.

1. INTRODUCTION

The design of user interfaces has a great influence on the usability of a system. The complexity of the interface design process is not only related to the complexity of the software system the user interface is being built for, but is also influenced by other factors such as the user or the context-of-use. Because of the evolution of different types of consumer devices like PDA's, cellphones and set-top boxes, together with a continuous growing user audience, creating accessible user interfaces is becoming a more difficult challenge everyday. A multitude of consumer devices can offer the same services by adapting the user interface to the capabilities of the system. A wide distribution of these services by means of different channels (mobile, interactive digital television, web, …) enforces the designer to make the user interface "granny-proof" i.e. usable by anyone. This leads to a contradiction: the user interface of a service should be generalised on the one hand because it has to be deployed on multiple devices, and specialised towards different user profiles on the other hand to make it more usable. We address this challenge by combining high-level user interface descriptions to create multi-device user interfaces with user profiles to determine the user preferences and requirements for a user interface.

XML-based high-level user interface descriptions (UIDL) [9] have proven to be very suitable to support multi-device user interface design. These UIDLs are labelled "high-level" because the interface is abstracted away from the concrete implementation: e.g. an UIDL could define a "choice from a range" that can be mapped onto a slider widget, a list widget or a text entry widget when the UIDL is transformed into the concrete final user interface. We will use the UIML [2] as a high-level UIDL; this language offers several advantages useful for creating multi-device user interfaces such as a separation of concern, domain-specific vocabularies and a structured description of the user interface.

On the content side, the MPEG-21 specification [4, 12] is a framework that embodies and relates several existing MPEG standards. It aims to enable the use of multimedia resources on a wide range of devices, independent on network infrastructure. As such it is an emerging standard to enable universal multimedia access. The work presented in this paper focuses on Part 7 of the MPEG-21 specification, *Digital Item Adaptation* [13] (see figure 1), and its influence on the adaptation of the user interface. In combination with a high-level UIDL this promises to offer a powerful means to build effective personal multi-device multimedia interfaces. To demonstrate the feasibility of this approach we implemented a user agent that combines an UIML document with an MPEG-21-based profile and renders the adapted user interface.

The remainder of this paper is structured as follows: section 2 gives an overview of related work in this area: both XML-based UIDLs and existing approaches towards user profiling are presented here. Next, section 3 gives more detail about the adaptation of the final user interface according to the user preferences. To support this, section 4 shows a case study where the usability of this approach is assessed. Finally, section 5 concludes this paper with a discussion about the obtained results and gives an overview of the future work.

2. RELATED WORK

XML-based high-level UIDLs have already been widely investigated and have proven to be suitable for multi-device user interface creation [9]. Besides the integration of a *user model* in a few existing UIDLs, only the traditional web browsers show practical use of user agents that can take a user profile into account. UsiXML [8] is an example of a language that takes the user profile into account, and can provide the intended users with a generated interface that meets their specific requirements. The drawback is the lack of a straightforward definition for a user profile: this information is only stored in mapping rules that can influence the user interface instead of as a separate profile that serves as input for such mapping rules. The most suitable UIDL for our goals is UIML which is described into detail in section 3.1. The separation of user interface style properties and other aspects of a user interface provides the necessary benefits to build multi-device *and* personalized user interfaces. Our implementation uses Uiml.net, an extensible open source implementation of this specification that can be used on multiple devices [10].

MPEG-21 is a well-known standard framework, but most of the existing research is focused on multimedia content transformation such as scalable video coding for example. Although MPEG-21 has an extensive user profile definition, it has been used only rarely to adapt the user interface itself. Other related work using MPEG-21 can be found in [11]: this work emphasises universal multimedia access by adapting the content to the user's environment. The Digital Item Adaptation Engine, shown in the middle of figure 1, is the core of the MPEG-21 content adaptation. Since the internals for this engine are not specified in the MPEG-21 standard, its operation is dependent on the specific implementation. In this work we are mainly interested in the *Usage Environment Description Tools* specification as input for a user agent that can adapt the user interface according to this information. Listing 1 shows an example listing of a part of an MPEG-21 document that specifies the visual impairment of the user.

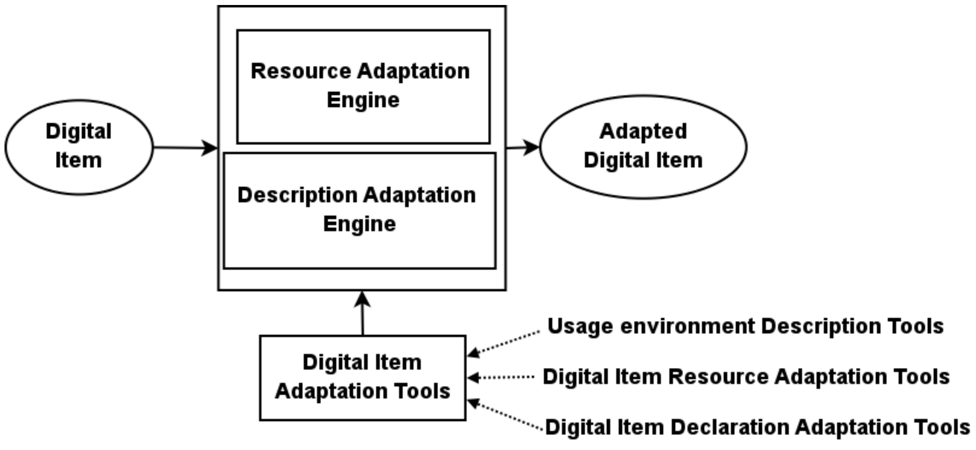

Digital Item
Resource Adaptation Engine
Description Adaptation Engine
Adapted Digital Item
Digital Item Adaptation Tools
Usage environment Description Tools
Digital Item Resource Adaptation Tools
Digital Item Declaration Adaptation Tools

FIGURE 1: MPEG-21 Digital Item Adaptation overview from [13]

```
<DIA>
  <Description xsi:type="UsageEnvironmentType">
    <UsageEnvironment xsi:type="UserCharacteristicsType">
      <UserCharacteristics xsi:type="AccessiblityCharacteristicsType">
        <Visual>
          <Blindness eyeSide="right"/>
          <LowVisionSymptoms>
            <LossOfFineDetail><TextualDegree>Mild</TextualDegree></LossOfFineDetail>
            <CenterVisionLoss><NumericDegree>0.7</NumericDegree></CenterVisionLoss>
            ...
          </LowVisionSymptoms>
        </Visual>
      </UserCharacteristics>
    </UsageEnvironment>
  </Description>
</DIA>
```

LISTING 1: MPEG-21 Visual Impairment example from [13]

The use of intelligent agents in combination with MPEG-21 can also be found in other work such as [5] and [7] but with a different scope than proposed here. An intelligent agents is defined as a software entity that perceives and acts in an environment, being social, pro-active, re-active and autonomous. The goal of intelligent agents in [7] is to reduce computer-user interaction and to enhance the user's experience while consuming multimedia items. Just as in [7], the work presented in [5] is focused on content delivery, but this time a broader context is used such as the location and surroundings of the user. The agent's main task is to take into account user preferences solely when selecting content, as most existing agent implementations do. Intelligent agents intercept the request a user makes and instead of returning a complete set of choices concerning the chosen multimedia item, it makes a significant number of choices for the user automatically. We want to take this approach one step further by letting intelligent agents adapt the user interface itself instead of deciding on which content to deliver, and enhancing the user's experience as well as the accessibility of the content itself.

3. PROFILE-DRIVEN USER INTERFACE ADAPTATION

3.1 High-Level User Interface Descriptions with UIML

An UIML document exists of several parts [1] that are shown in figure 2. The **Interface** section describes four parts of the user interface:

Structure: describes the "hierarchy" of the user interface. It defines the different parts that are contained in the user interface, and the interactor name of each part.

Style: describes properties of the parts defined in the structure. This allows to change properties of the interactors like text, colour, …

Content: separates the content of the interface of the other parts.

Behaviour: defines rules with actions that are triggered when some condition is met. Some kind of event mechanism is offered to the user interface designer this way.

Vocabularies are referred to as *peers* in the UIML specification: this contains the mapping with the concrete user interface toolkit. The logic section describes the mappings with the software interface to communicate with the application logic.

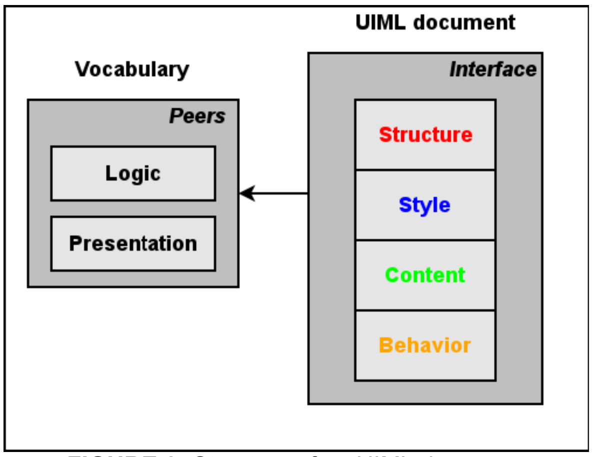


FIGURE 2: Structure of an UIML document

The style section of UIML is partly *generic* and partly *platform specific*. The former contains the set of properties that can be applied independent of the target widget set and consumer device that is being used, the latter contains the set of properties that are specific for a widget set and device. If the style description is limited to the set of generic properties, the user interface can be transferred to a broader range of consumer devices. Nevertheless, the platform specific properties make sure the special capabilities of a particular widget set or device can also be used which is important to build more attractive user interfaces.

Listing 2 shows an example of a very basic UIML document. In the *structure* part, a container and a button are defined as "c_body" and "b_hello". Throughout the document we refer to these parts with these identifiers. In the *style* part, we define the style properties of the user interface elements. The body has a size, and the button gets a position and a label. The *content* part is left out in this example, and in the *behavior* section, an action is defined when a specified condition is fulfilled. When the user clicks the button (b_hello == ButtonPressed), the function "Console.println" is called with the specified parameter. Finally, the *peers* section links to an external vocabulary, shown in Listing 3, defining the mappings between the classes and properties used in this document and the "swf" (System.Windows.Forms) specific classes and properties.

```
<?xml version="1.0"?>
<!-- <!DOCTYPE uiml PUBLIC "-//Harmonia//DTD UIML 2.0 Draft//EN" "UIML3_0a.dtd"> -->
<uiml>
         <interface>
                  <structure>
                           <part id="c_body" class="Container">
                                    <part id="b_hello" class="Button"/>
                           </part>
                  </structure>
                  <style>
                           <property part-name="c_body" name="size">300,300</property>
                           <property part-name="b_hello" name="position">50,50</property>
                           <property part-name="b_hello" name="label">Hello</property>
                  </style>
                  <behavior>
                           <rule>
                                    <condition>
                                             <event part-name="b_hello" class="ButtonPressed"/>
                                    </condition>
                                    <action>
                                             <call name="Console.println">
                                                      <param>Uiml says hello!</param>
                                             </call>
                                    </action>
                           </rule>
                  </behavior>
         </interface>
         <peers>
                  <presentation base="swf-1.1.uiml"/>
         </peers>
</uiml>
```

LISTING 2: UIML example

```
<uiml>
         <presentation base="swf-1.1" id="SWF">
                  <d-class id="Container" used-in-tag="part" maps-type="class" maps-to="System.Windows.Forms.Panel">
                           …
                           <d-property id="size" maps-type="setMethod" maps-to="Size">
                                    <d-param type="System.Drawing.Size"/>
                           </d-property>
                           …
                  </d-class>

                  <d-class id="Button" used-in-tag="part" maps-type="class" maps-to="System.Windows.Forms.Button">
                           …
                           <d-property id="label" maps-type="setMethod" maps-to="Text">
                                    <d-param type="System.String"/>
                           </d-property>
                           <d-property id="position" maps-type="setMethod" maps-to="Location">
                                    <d-param type="System.Drawing.Point"/>
                           </d-property>
                           …
                  </d-class>
         </presentation>
</uiml>
```

LISTING 3: UIML vocabulary (swf-1.1.uiml)

Instead of hard coding the user interface (and thus the adaptability strategies) of the application, we store the user interface as a set of UIML documents. Each part of the user interface will be rendered *at run-time* from the related user interface descriptions. This gives us the ability to adapt the properties of the user interface as explained in section 3.2 each time a part of the user interface has to be accessed by the user. On the other hand: this also implies the user interface will appear on screen after a small delay because the user interface is dynamically built and adapted while the application is in use. This can be avoided by keeping user interface parts in memory and reuse them without a new adaptation and rendering stage. These "in-memory" parts will not be able to adapt to changes in the user profile.

3.2 Profile-aware properties

A user interface property can be expressed as a triple (p, v, x), where p is a user interface part or class of user interface parts, v is a property of p and x is the value of v for p. Based on this notation we can define a particular user interface configuration as a set of user interface properties: $\{t_1, t_2, ..., t_n\}$ with $t_i = (p_i, v_m, x_k)$. A profile is a mapping function $f : \{t_1, t_2, ..., t_n\} \to \{t_1, t_2, ..., t_m\}$ with $n, m \in \mathrm{N}$ and $1 \le n \le m$ (a user profile should not remove properties that were already set). This definition implies a *user agent* can *change* the value of a property that was specified in the user interface description or *set* a value for a property that was not specified in this description. Figure 3 shows the user agent's primary function: it serves as a property adaptation mechanism that transforms the properties contained in an UIDL according to the profile. The user agent executes a profile on a set of interface properties and does so by interpreting the data in the profile and relating it to the properties of the user interface.

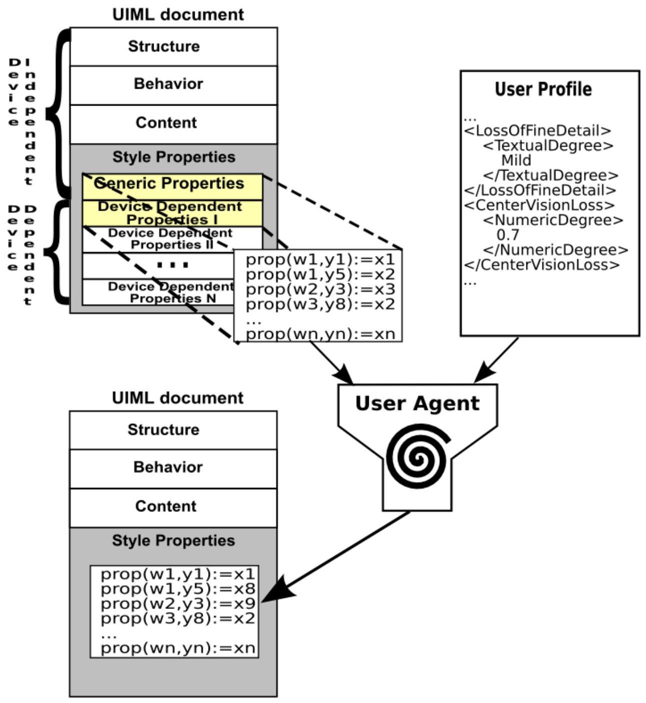


FIGURE 3: Applying a user profile to a high-level user interface description

The user agent only changes the style properties of a UIML document, the other parts, structure, content and behavior, are left untouched. Since the behavior and the structure of a user interface are mostly device independent this poses no problem. The content could differ according to the device that is being used. We will rely on other MPEG-21-based mechanisms to support the content adaptation, such as the systems shown in [5] or [7]. Content adaptation such as scalable video coding is not the scope of this work; it is mostly dependent on the device profile and is almost always dependent on functionality of the service provider to transform the content appropriately. After the user agent has updated the properties, the UIML document is reassembled and can be rendered on screen. Notice the properties of the resulting UIML document can no longer be divided into generic and device specific properties, but can be labelled as “profile specific” because the properties are now targeted towards a particular device and user.

In our approach the *structure* of an interface is also considered as a property. For example, the Gnome usability guidelines [3] contain an entry that states: “Arrange controls in your dialog in the direction that people read”. The direction people read can be part of the user profile and requires a more complex behaviour of the user agent since it also has to *restructure* the structure as defined by the user interface specification. The solution is to consider the arrangement (direction) of controls as a property of their parent part, since parts are hierarchically nested and each subpart has a property that defines its placement within the parent part.

3.3 Encoding Accessibility

To provide the reader with a concrete idea about the responsibilities of the user agent software, table 1 lists some examples of MPEG-21 user characteristics and their effect on the user interface properties. In the left column the MPEG-21 characteristic is listed, and the right column shows which user interface properties will be manipulated to apply to this characteristic. The properties that can be manipulated are defined in a generic UIML vocabulary that can be used to create multi-device interfaces. The mapping from the left column to the right column is executed by the user agent as part of the mapping function identified in section 3.2. For this purpose the rule shown in the right column of table 1 is written as the triple (p, v, x), and the user agents changes the value x according to some predefined adaptation strategy. For example: the *Colorvisiondeficiency* characteristic can impose a change of the values x_1 and x_2 of (p, v_1, x_1), (p, v_2, x_2) where p matches all parts that belong to the class “labels”, v_1 is the *foregroundcolour* property and v_2 is the *backgroundcolour* property.

MPEG-21 CHARACTERISTICS	UIML PROPERTY ADAPTATION
Presentation Preferences	
Audiopresentation	Change the *volume* property of audio used in the application.
Displaypresentation	Change the *content* property of the affected user interface parts.
Accessibility Characteristics	
Audio	Change the *visibility* property of every button linked to an Audio-fragment, change the *visibility* property of substitute labels with an explanatory text.
LowVisionSymptoms	Change *foregroundcolour*, *backgroundcolour* and *fontsize* properties of labels.
Colorvisiondeficiency	Change *foregroundcolour* and *backgroundcolour* properties of labels.
MobilityCharacteristics	Depending on the mobility of a user, e.g. highway vehicular, urban vehicular and pedestrian, the *size* and *visibility* properties are changed.
Natural Environment Characteristics	
Illuminationcharacteristics	Change the *colour* property of the user interface parts.

TABLE 1: Example Profile Mapping Rules

Notice these rules should not be hard-coded in the user agent, but, if required, a set of universally applicable rules is selected. By making the properties of a user interface explicit through a high-level UIDL, the method for customisation of the user interface that leads to a better usable, personal and accessible user interface has become more straightforward and flexible.

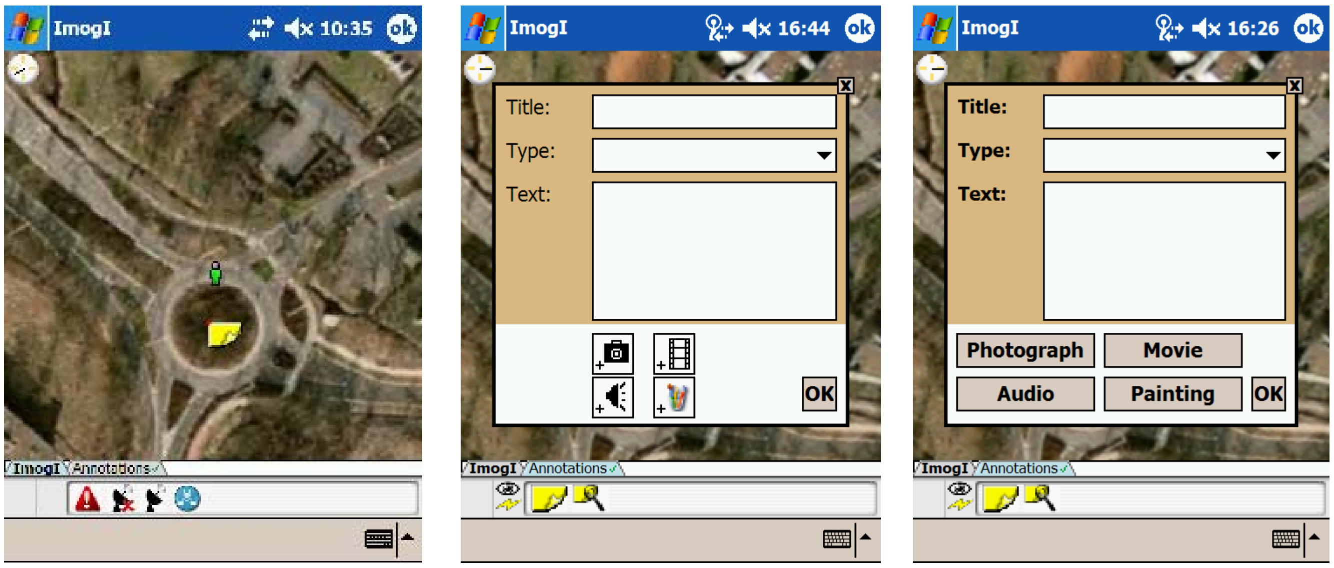
4. CASE STUDY: A LOCATION-AWARE NOTE-TAKING SERVICE

In previous work, we created a mobile guide framework to support the deployment of custom location-aware services as layers upon a map-based interface. [14]

Nowadays, such services are being accessed by a very diverse user audience. As an example we built a service similar to the GeoNote concept [6]: physical locations can be related with pieces of information. An annotation has a content type, can contain text, movies, audio, images or self-made drawings. A user can create annotations at any point, can add information to it and save this annotation. Figure 4 shows this service: 4(a) provides a map-based interface that shows where notes are positioned, and figure 4(b) shows the dialog for adding, reading or editing notes. Notice the user profile specifies a preference for buttons with large text instead of icons for people with a visual impairment.

Since only the dialog in figures 4(b) and 4(c) is built using UIML, there is a clear difference between the adaptation of the foreground dialog and the background map-based interface. The former adapts according to the user characteristics, while the latter does not change its representation. An application can be made completely “profile-aware” if it uses UIML for the complete user interface description. The benefit of our approach is the possibility to extend an existing user interface with new, UIML-based, dialogs that are profile-aware.

Several properties of this interface are adapted according to the user profile. For example: when the current user is visually impaired, we can change the style of the UIML document, by increasing the font-size, enlarging the buttons and making the icons in this control invisible (or vice versa). Figure 4(c) shows what the same user interface looks like with the profile of a visually impaired person.



(a) Service running on PDA (b) Normal sight (c) Visually impaired

FIGURE 4: Location-aware note-taking service with different user profiles

5. CONCLUSIONS AND FUTURE WORK

We described an approach to create personalized and more accessible multi-device user interfaces by combining user profiles and a high-level XML-based user interface description language. A user agent processes the user characteristics described in a MPEG-21 compliant user profile, and selects and manipulates properties of the user interface accordingly. UIML is used to describe these user interfaces because it provides a clean separation of the user interface structure, style, content and behaviour. UIML is used to create multi-device user interfaces, but it only describes the presentation of a user interface. In a broader context, the user profile does not only influence the presentation, but also the tasks the user executes and the navigation through the user interface. The next step in our work is to also take into account the task model that specifies the different tasks the user interface supports and adapt this model according to the user profile.

We demonstrated the use of this approach with an example case study. A location-aware service for a PDA could adapt its user interface according to the user profile, although its interface was only written once in UIML. This case study proves the approach is feasible and can reduce the time needed to build user interface that can adapt according to a user profile and a device. The designer can create these type of interfaces by means of two orthogonal specifications (MPEG-21 and UIML) that are used independent of each other.

REFERENCES.

[1] Abrams, M. and Helms, J. (2004) User Interface Markup Language (UIML) Specification version 3.1. Technical report, Harmonia.

[2] Abrams, M., Phanouriou, C., Batongbacal, A.L., Williams, S.M. and Shuster, J.E. UIML: An Appliance-Independent XML User Interface Language. *WWW8 / Computer Networks*, **31**(11-16):1695–1708

[3] Benson, C. Elman, A., Nickell, S. and Robertson, C. (2004) Gnome human interface guidelines 2.0. World Wide Web. `http://developer.gnome.org/projects/gup/hig/`.

[4] Burnett, I., Van de Walle, R., Hill, K., Bormans, J., and Pereira, F. (2003) Mpeg-21: Goals and achievements. *IEEE Multimedia*, **10**(4):60–70.

[5] El-Khati, K., Zhang, Z. E., Hadibi, N., and Bochmann, G.V. (2004) Personal and service mobility in ubiquitous computing environments. *Wireless Communications and Mobile Computing*, **4**:595–607.

[6] Espinoza, F., Persson, P., Sandin, A., Nystr¨o;m, H., Cacciatore, E., and Bylund, M. (2001) Geonotes: Social and navigational aspects of location-based information systems. In *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 2–17, London, UK, 2001. Springer-Verlag.

[7] Kim, M., Lim, J., Kang, K., and Kim, J. (2004) Agent-based intelligent multimedia broadcasting within MPEG-21 multimedia framework. *ETRI Journal*, **26**(2):136–148.

[8] Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., L’opez-Jaquero, V. (2004) USIXML: a Language Supporting Multi-Path Development of User Interfaces. In *The 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with the 11th International Workshop on Design, Specification and Verification of Interactive Systems, Tremsbttel Castle, Hamburg, Germany*, pages 89–107.

[9] Luyten, K., Abrams, M., Limbourg, Q., and Vanderdonckt, J., editors. (2004) Developing User Interfaces with XML: Advances on User Interface Description Languages. *Satellite workshop of Advanced Visual Interfaces (AVI) 2004*, Expertise Centre for Digital Media.

[10] Luyten, K., and Coninx, K. (2004) Uiml.net: an Open Uiml Renderer for the .Net Framework. In Quentin Limbourg, Robert Jacob, and Jean Vanderdonckt, editors, *CADUI 2004*, volume 4. Kluwer Academic.

[11] Sun, H., Vetro, A., and Asai, K. (may 2003) Resource adaptation based on mpeg-21 usage environment descriptions. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, volume 2, pages 536–539.

[12] Vetro, A. (Jan 2004). Mpeg-21 digital item adaptation: Enabling universal multimedia access. *IEEE Multimedia*, **11**(1):85–87.

[13] Vetro, A., Timmerer, C., and Devillers, S. (2003) Iso/iec 21000-7 fdis: Mpeg-21 digital item adaptation.

[14] Luyten, K., Coninx, K., Houben, G., and Winters, F.. Blended Maps and Layered Semantics for a Tourist Mobile Guide, Accepted for the 11th International Conference on *Human-Computer Interaction, Las Vegas, Nevada, USA*, July 22-27, 2005