THE USER INTERFACE AS A DOCUMENT

Adriana-Mihaela Tarța*

**Babeş-Bolyai University, Mathematics and Computer Science Faculty,*
Str. Mihail Kogălniceanu nr. 1, 400084 Cluj-Napoca, Romania
Tel: +40-264-405327, fax: +40-264-591906
adriana@cs.ubbcluj.ro

Abstract: In the last years there has been a great progress in the domain of computational devices. In the market today there are four types of devices that can run computer-generated user interfaces: desktop computers, palmtop and handheld computers, smart-phones and voice-phones, each of them with different processing power, storage capabilities and different abilities to download, store or run executable code. In order to minimize the work of user interface developers and to provide consistency across platforms, new languages for describing user interfaces have been developed. The present paper presents UIML, a universal language for describing platform independent user interfaces, the way an UIML specification may be used and the advantages of its use.

Key words: multi-platform user interface, UIML, specification language, declarative model

1. Markup languages for user interfaces

XML is a language often proposed for describing user interfaces because it is platform independent, it is declarative through the usage of XSLT, it provides rapid prototyping using a stylesheet, the results may be seen immediately in a browser and it's extensible because it's a markup language [4]. WML, VoxML, VoiceXML and UIML are all XML based languages used to describe the user interfaces.

UIML (User Interface Markup Language) gives a description of a user interface as a virtual tree of elements. The aim of UIML was to provide a universal way to describe user interfaces, independent of the platform and the constraints and "to reduce the time to develop user interfaces for multiple device families" [3]. The peculiarities of each device are isolated in a particular element, named <style>. Being an XML-based language, UIML uses a set of tags to describe the components of a user interface and does not contain any platform specific tag. The architecture of an UIML document is presented in Figure 1:

```
<?xml version="1.0" ?>
<!DOCTYPE uiml PUBLIC "-//UIT//DTD UIML 2.0 Draft//EN" "UIML2_0f.dtd">
<uiml>
   <head>…</head>
   <interface>…</interface>
   <peers>…</peers>
   <template>…</template>
</uiml>
```

Figure 1 – The architecture of an UIML document

An UIML document captures the elements that are common to any user interface: an enumeration of the user interface parts, events that occurs for those parts, presentation style and content, specified in the <interface> tag, and the interconnection to application logic through <peers> tag. The <head> element contains metadata about the current UIML document and is not considered part of the description of the interface. The <template> element wraps parts of the user interface that could be used in the description of the same user interface or across user interfaces. In order to render the UIML document in a platform specific user interface there is a need to use a render specific to each platform that has an associate vocabulary containing the set of widgets used to generate de user interface. The developers create the UIML document using the platform-specific vocabulary and then the document is rendered to the target platform.

1.1 Using UIML documents

UIML documents can be used in a client-server relation, or without using a server. If the UIML document is stored on a server, when the user invokes an application, two different things may happen. First, the UIML document may be compiled on the server, especially when the client device is not capable of downloading the application and the memory is limited (cellular phones), or, in the second case the document is interpreted while the user interacts with the application [2]. The second approach is more flexible and avoids the need to send executable code through a network connection. When using UIML without a server, the UIML document or the compiled code must be installed along with the application logic on the end-user device and at runtime, the rendered interface directly communicates with the application logic.

The UIML provides features to tailor the interface to different users or devices by using multiple structure/style/content/behavior elements and letting the user choose his preferences, or, the user interface may be created dynamically using the information stored in a database [2].

2. CONCLUSIONS

In this paper the need for a universal language for describing user interfaces for various platforms was emphasized. The paper presents the main ideas regarding the use of UIML, the elements of an UIML document and the way an UIML document may be rendered in an executable user interface. Some problems regarding the definition of more general vocabularies for UIML description and the layout of the components of a user interface on different platforms remain still a subject for further research.

3. REFERENCES:

1. Abrams M., Ali F., (2001), *Simplifying Construction of Multi-Platform User Interfaces Using UIML*, European Conference UIML 2001, France;
2. Abrams M., Phanouriou C., (1999), *UIML: An XML Language for Building Device-Independent User Interfaces*, Proceedings XML 99, Philadelphia;
3. Abrams M., Helm J., (2002), *User Interface Markup Language (UIML) Specification*, Document Version 08 February 2002, Language Version 3.0;
4. Luyten K., Coninx K., (2001), *An XML-based runtime user interface description language for mobile computing devices*, Proceedings of the 8th International Workshop on Interactive Systems: Design, Specification, and Verification-Revised Papers, p.1-15;