Evaluation of High-Level User Interface Description Languages for Use on Mobile and Embedded Devices

Jan Van den Bergh **Kris Luyten** **Karin Coninx**

Expertise Centre for Digital Media
Limburgs Universitair Centrum
Wetenschapspark 2
B-3590 Diepenbeek-Belgium
{Jan.VandenBergh, Kris.Luyten, Karin.Coninx}@luc.ac.be

ABSTRACT

Model-based design and the use of high-level user interface descriptions languages (HLUID) have been proposed for the design of multi-platform user interfaces. In this paper we present an analysis of required properties for HLUID so that they can be effectively used for the design of multi-platform user interfaces that can be used on mobile and embedded devices. Two HLUID, SEESCOA[1] XML and XForms basic profile, are evaluated. The former is used in a model-based design method, Dygimes, and the latter is a candidate recommendation of the World Wide Web Consortium. Based on this analysis, adaptations to SEESCOA XML and an adapted structure for the use in a new version of Dygimes, supporting the design of context sensitive user interfaces, are presented.

INTRODUCTION

The increase in diversity of computing platforms used in the every day life has caused a search for a methodology and models that can ease development for multiple platforms. One proposed method is the use of model-based design, which allows the design of user interfaces for multiple platforms and/or multiple contexts of use through the use of high-level models. These models all address one particular aspect of the design of the user interface and the application for which it is designed.

In parallel to this effort, the need for standards is recognized in industry. In the specification of user interfaces, this can be noticed in the growing compliance of current web browsers with the recommendations of the World Wide Web Consortium (W3C). This organization is actively encouraging web-authors to also separate the different aspects of the user interface through the design of new standards and new versions of existing standards. These standards also promote the separation of the different aspects of web pages: content and structure (XHTML, XForms, VoiceXML), and presentation (CSS, XSLT).

In this paper we elaborate on our work to merge certain aspects of the two approaches (model-based and standards-based declarative design) and on how well they are fit for mobile devices using a Java-based implementation. We will address into more detail the use of XForms as a high-level user interface description language and look at the effects of its use in the Dygimes design-process and tools[4]. In the discussion we will concentrate on the features offered in the basic profile of XForms since it is a version especially targeted towards mobile devices.

After the presentation of some related work we will start this paper by giving an overview of the Dygimes approach and discussing requirements for a HLUID to be used on mobile devices, followed by a detailed discussion of SEESCOA XML, the high-level user interface specification language used in the approach. After that we will shortly present the relevant aspects of XForms. The next sections give an evaluation of both approaches and the impact of a changed SEESCOA XML in an adapted Dygimes approach that supports the design of context-sensitive user interfaces The final section presents the conclusion and future work.

RELATED WORK

Several approaches have been taken to describe functionality for display on mobile devices. Nichols et al. [11] designed an XML-based description specifically targeting the use of mobile devices as a remote control for complex appliances, such as a hi-fi installation. Their description consists of the description of three different components: states, commands and descriptions. Relations between components can be expressed by groups and conditional statements. How the specification is translated in user interface objects (such as buttons and

[1]SEESCOA stands for "Software Engineering for Embedded Systems using a Component-Oriented Approach". `http://www.cs.kuleven.ac.be/cwis/research/distrinet/projects/SEESCOA/`

text fields) is left to the rendering engine. This notation is however too specific for our purposes; we want a more general approach.

A different approach is taken by Marucci et al.[9], they propose the use of model-based approach for the design of user interfaces for multiple platforms, among which a PDA or a mobile phone. In this approach, one starts by defining a global task model, which is refined for the different platforms. The refined task models can be translated to abstract user interface descriptions that include information about the composition of a dialog, in which the abstract interaction objects are split in several meaningful categories, as well as the dialog transitions. This abstract user interface description is semi-automatically translated to a concrete user interface description that is rendered at runtime. They mention support for adaptivity by the derivation of a user model, which can be updated at runtime, from the task model. The user model, then could have influence on the user interface. No detailed explanation of this adaptive process at runtime is provided. This approach differs from ours in that they use transformations between task model, high-level user interface specification and concrete user interface specification at design time. In the process of the transformations, however, some information that is useful (such as the actions of the user, the application objects manipulated through the user interface) and other platform-specific information is added.

UIML[1] is an XML-based meta-language for the specification of user interfaces that separates the different parts of the user interface description in different parts of the XML. A technical committee of OASIS[2] is is working to make it an open standard. Despite the fact that the specification forces separate specification of user interface structure, style, interaction and toolkit bindings (vocabulary), most specifications of the user interface structure still have toolkit-specific information in all parts of the user interface description. Ali and Abrams[2], however, proposed a generic vocabulary, which makes it possible to design a single user interface structure for a diversity of platforms. Although this vocabulary makes it possible to design user interfaces for more systems, it doesn't have some properties we would like it to have: it does not preserve semantic information that connects certain parts of the user interfaces. The connection between a text field and the label that describes the content of the text field are not inherently connected, while this is the case in XForms and SEESCOA XML. UIML is less suitable for small mobile devices since the display of a small user interface requires a complete vocabulary containing potentially lots of unused control specifications to be read.

XIML[8] is a meta-language that perhaps could express all information we want it to express; it allows specification of tasks, high-level and concrete user interfaces as well as a domain-specific information. Its greatest weakness and a reason we cannot be sure it meets all our needs is that it is a closed specification from one company that has a very restrictive license, which is something that is far from desirable for a specification that should be used on a multitude of devices.

Mitrović and Mena[10] proposed to use another declarative language with an open specification, XUL[6] for the description of user interfaces for mobile systems. They use XUL as the basic specification but it is transformed to HTML and WML using XSL transformations for display on mobile devices. XUL does not meet our criterion that is has to be a high-level user interface description language.

DYGIMES

Dygimes[4] is a framework that uses a model-based approach to design user interfaces for mobile devices and embedded systems. In the approach task models and high-level user interface descriptions are combined into a single specification that can be rendered by a runtime environment. Limited styling support and interaction with web services is provided. Figure 1 shows the different specifications that are used to define the user interfaces and their corresponding conceptual models as specified by Calvary et al.[3] in the reference framework for plasticity.

The central specification is a task model in ConcurTaskTrees notation[12] (CTT). It is a hierarchical task model that has four types of tasks: user tasks (tasks performed by the user without interaction with the device), interaction tasks (tasks involving an interaction of the user with the application), application tasks (performed by the application) and abstract tasks (split into two or more subtasks of different types). Tasks are interconnected with temporal operators that can be used to determine which tasks are active during the completion of a task.

The leaf-nodes of the CTT are annotated with high-level user interface descriptions, using SEESCOA XML, discussed in more detail in the following section. This enables us to generate user interfaces from the annotated task model. Custom mappings between the high-level interactors in SEESCOA XML and the concrete interactors used to show the user interface[7] can be defined. Limited styling support is also available.

We are extending the Dygimes approach to support context-sensitive user interfaces on mobile devices. For this approach we are developing a new XML-based notation that integrates information about context, tasks, and high-level user interface controls. The requirements we determined for the HLUID are:

high-level specification The user interface specification should describe the parts of the user interface at

[2]Organization for the Advancement of Structured Information Standards

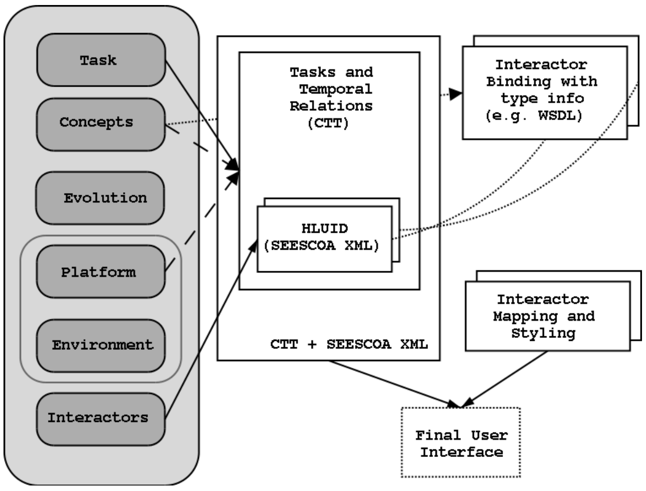

Task
Concepts
Evolution
Platform
Environment
Interactors
Tasks and Temporal Relations (CTT)
HLUID (SEESCOA XML)
CTT + SEESCOA XML
Interactor Binding with type info (e.g. WSDL)
Interactor Mapping and Styling
Final User Interface

Figure 1. Dygimes specifications

a high level as SEESCOA XML does.

expressive The user interface description should not limit the possibilities of a mobile device to render the user interface. Special controls for special needs (e.g. password and date entry) may not be excluded from use due to limitations in the expressiveness of the HLUID. This also implies that (limited) extensions should be possible without the introduction of new controls in the HLUID.

customizable Styling and custom mappings should be possible and dynamically determined; designers do not want to be limited in their creativity by defaults and organizations want to be recognized.

embeddable The specification will be used embedded in an XML notation of the Contextual ConcurTask-Trees, an extension of the CTT introducing context interaction in the task model.

compact The specification should have a compact form for efficient use on mobile devices and embedded systems.

local and remote interaction It should be possible to handle both local and remote interaction involving both sending and receiving data.

SEESCOA XML

SEESCOA XML is a XML-based language for the specification of high-level user interface descriptions. It was a stand-alone specification that is now used in conjunction with the ConcurTaskTrees notation [12] in the Dygimes process[4].

Characteristics

The most important characteristics of SEESCOA XML are:

Multiple platform support The use of high-level descriptions of the controls as well as the ability to make logical groups and the specification of possible splitting of groups should make rendering on multiple platforms possible.

Local and remote interaction The specification allows the definition of different communication channels between the user interface and the program logic. Currently XML-RPC, SOAP and local java event handling are supported.

Human readable The specification should be readable by both man and machine.

Separation of concerns The user interface specification should be as independent as possible of the presentation and the programming logic.

Scalable for embedded systems The specification is especially targeted towards embedded systems, e.g. suitable to be stored in limited memory space like on a Radio Frequency Identifier Tag,. . .

Specification

A typical user interface description using SEESCOA XML is organized as shown in figure 2. The group tag identifies elements of the user interface that have some kind of logical relation: e.g. all the controls to specify a date. Groups can contain other groups, controls and constraints. The constraints included in a group specify the spatial relation between the different children of the group.

There is a limited set of controls that can be presented. Each control is specified within a *interactor*-tag that contains a control-specific tag. All controls have an *info* tag that can be used to describe the purpose of the control. The information in the info tag can be displayed as a separate control or as a part of the main control as can be seen in figure 2, showing a partial user interface description for making an appointment.

SEESCOA XML provides no information about the presentation of the user interface, except for layout purposes. One can specify constraints for controls and groups within a common group. Four linear spatial constraints are supported (left, right, above and under) and are honoured as much as possible. When it is impossible to place all controls on the screen as desired, the user interface can be split into several layers. The designer preferences are taken into account here: groups can also be marked “non-splittable”.

There is limited styling support for SEESCOA XML. Styling is specified using an XML format and supports mapping of the controls specified in SEESCOA XML to the controls that are rendered as well as styling of the mapped controls. Customized mappings can be created for all controls of a specific type, for a subset thereof or for a single control. The selection of the appropriate mapping-rule is determined by the type of control and by (part of) the name of the controls[7]. The appearance of the controls can be customized per (set of)

```
<?xml version="1.0"?>
<ui>
<title>Make Appointment</title>
<group>
    <group name="exampleGroup">
        <interactor>
            <range name="hour">
            <info>hour</info>
            <min>0</min>
            <max>23</max>
            <start>9</start>
            <tick>1</tick>
            </range>
        </interactor>...
        <interactor>
            <textfield name="description">
            <info>Description</info>
            <size>40</size>
            </textfield>
        </interactor>...
        <interactor>
        <button name="ok">
            <info>OK</info>
            <action type="Java">
               <!-- action description-->
            </action>
        </button>
        <constraints>
            <!-- spatial contraints-->
        </constraints>
    </group>
</group>
</ui>
```

Figure 2. Example SEESCOA XML

mapping rule(s). The implementation of styling, however, is currently only specified for platforms supporting Java AWT.

The binding with the functional core is provided through *action* tags, which have a *type*-attribute that specifies the invocation protocol. Action elements are generic elements that can be used as sub elements for all available controls. Each manipulation of the control will execute the invocations specified within the action elements. The XML within the action element is not restricted within the XML Schema; this allows developers to attach their own invocation protocols to controls. At the time of writing we have successfully used direct method invocation (dmi), XML-RPC and SOAP for this purpose. Within the SEESCOA system it was possible to invoke the specific SEESCOA components.

XFORMS

XForms[5] is a W3C recommendation for the design of forms for web-based applications using an XML-based specification. It is designed to be the replace the current form-specification in HTML in the modularized XHTML 2.0 specification. Its possibilities, however, are much richer than those offered by HTML form's.

Characteristics

The characteristics of XForms can be specified as:

XML for instance data This makes direct validation and processing by the application backend possible (no need for marshaling of data). It also ensures that data is internationalization ready.

Multiple platform support High-level description of user interface controls makes multiple device support possible.

XML event handlers The use of declarative event handlers for the most common events reduces the need for imperative scripts for this purpose.

Strong typing All instance data is strongly typed enabling client-side checking of supplied data and saving a round-trip for invalid data.

Extensibility All parts of XForms are extensible through the use of namespaced attributes and/or tags.

Reuse Existing schemas are reused (e.g. XML Schema for typing and XML Events for event handler specification)

Enhanced accessibility XForms separates content and presentation. User interface controls encapsulate all relevant metadata such as labels, thereby enhancing accessibility of the application when using different modalities.

There are different conformance levels for the XForms standard; full and basic profile. The basic profile is meant for enabling XForms parsers in mobile devices. Since this is what we are interested in, we will only discuss the XForms basic profile from now on.

One of the special properties of XForms is that it is not meant to be used on its own, but rather that is would be embedded in other specifications, such as XHTML. It therefore has no root-tag, as other XML-based languages have and can be used in different places in a XML document. Differentiation between XForms and the data in the host language is done by the use of namespaces.

Specification

A XForms specification consists of two major parts, which are embedded in a host-language as depicted in figure 3:

model this part of the description describes the types of information that will be manipulated or used by the form controls and the submission method that is used to communicate with the application logic

form controls a high-level description of the form controls that are used to input, manipulate and/or output data or submit data to a server

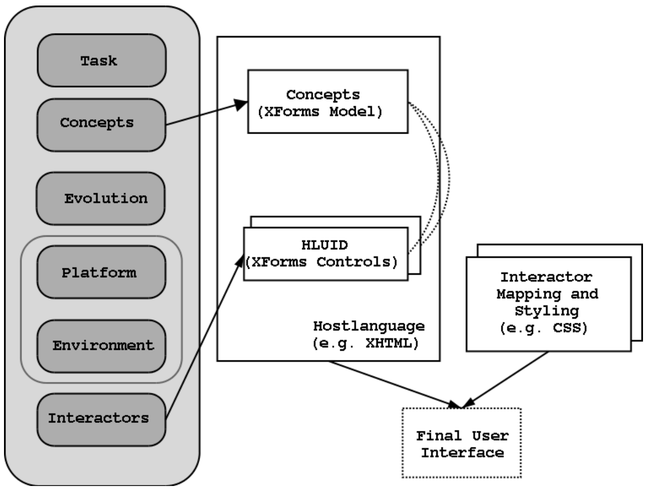


Figure 3. XForms specifications

```
<xforms:model>
    <xforms:instance>
        <appointment ...>...
             <hour>9</hour>
             <description/>
        </appointment>
    </xforms:instance>
    <xforms:bind constraint=".&lt;24"
      nodeset="/appointment/hour"
      type="xsi:nonNegativeInteger"/>
    <xforms:submission
      action="examples.Appointment"
      method="dygimes:dmi" id="submit"/>
</xforms:model>
```

Figure 4: XForms model example, corresponding controls in figure 5

We will discuss the most relevant parts the specification into more detail.

Model

The model part consists of two parts; the first describes the structure of the *instance* data that will be submitted by a *submission* specified in the second part. The second part is also used for the specification of the type of the elements in the instance data.

Datatyping of the instance data specified in the model can be done by binding the instance data to simple XML Schema datatypes[3] and XForms data types using the bind tag and XPath. Further restriction on the possible values of instance data can be posed by specifying a *constraint*, an XPath expression evaluating to *true()* or *false()*. The *xforms:bind* tags in figure 4 show examples of such expressions. The value of hour is a non-negative number and constrained to be smaller than 24.

[3]in the Full Profile almost all XML-schema datatypes can be used

```
<xforms:group>
    <xforms:input ref="/appointment/hour">
        <xforms:label>Start time</xforms:label>
    </xforms:input> ...
    <xforms:textarea
       ref="/appointment/description">
        <xforms:label>Description</xforms:label>
    </xforms:textarea>
    <xforms:submit submission="submit">
        <xforms:label>OK</xforms:label>
    </xforms:submit>
</xforms:group>
```

Figure 5. XForms controls example

The *xforms:submission* tag specifies how the information in the *xforms:instance* should be handled. The method describes the type of submission, which can be arbitrary, although a limited set of guaranteed methods are described in the recommendation, such as the different HTTP-methods and (local) storage.

User Interface

XForms supports similar controls as SEESCOA XML as can be seen in figure 8. In most cases controls in both languages can easily be matched. There are a couple of exceptions: SEESCOA XML has no special control for password entry, nor for multi-line text input and file upload. A special remark should be made about the controls for information display and for giving commands.

The *output* control has no specific meaning other than that it is used to display data described in the model. How the data is presented to the user is only specified in the mapping/styling. This contrasts to the approach in SEESCOA XML where different controls are used for the display of text or visual data (images).

The command functionality is represented in SEESCOA XML by a single control, the *button*. XForms, however, has two separate controls for this purpose; *trigger*, which can be used to perform some manipulation of the user interface, and *submit*, which is used to submit data to the functional core.

XForms also supports dynamic user interface specifications through the use of *switch*, *repeat* elements which allow dynamic insertion and removal of controls of the user interface without the need for a (separate) script. This makes it possible to specify dynamic lists in XForms, which can be difficult or impossible in other user interface description languages (such as SEESCOA XML).

Figure 5 shows the controls that can be used in conjunction with the model in figure 4 to generate an interface equivalent to the one specified in figure 2. One difference that can be noted is that instead of an *input* is used instead of a *range* which is used in the example with SEESCOA XML. This is caused by the restrictions

control modifier	SEESCOA XML	XForms
information about ...	info	label
extra information	*not supported*	hint
help	*not supported*	help
error notification	*not supported*	alert
action	action	action
selection items	item	item
selection item group	*not supported*	choices
size hints	*not supported*	appearance

Figure 6: Tags providing actions or information about control

on type binding for the range in XForms, which cannot be bound to an integer-based type. Both the specification in SEESCOA XML and the one using XForms support the same restriction on the input for the hours and minutes.

Styling can be done by using CSS, although to reach all the desired styling support (including mapping to concrete controls) future standards are needed. Furthermore, XForms leaves the specification of the attributes needed for style up to the containing document structure.

EVALUATION HLUID

We will now revisit our requirements for a HLUID that appropriately supports mobile and embedded systems. The first requirement we posed, high-level specification, is met by both SEESCOA XML and XForms. Both have a similar level of abstraction as can be seen in figures 8 and 6. The level of abstraction is also similar to that used in other approaches[2, 9].

To evaluate expressiveness regarding mobile devices, we looked at the MIDP 2.0[13] (Mobile Information Device Profile). This specification for java virtual machines on small mobile devices should give a good idea on what functionality will be reliably present on mobile devices and, thus, should be supported. Figure 8 shows that the expressiveness of XForms meets, and even exceeds our needs (there is no standardized way for giving hints and help about certain controls in MIDP 2.0). The typing and constraint support of XForms allow a MIDP 2.0 rendering engine to use a DateField a a Textfield with constraints (see figure 7), and more generally enable native support of type-specific controls. XForms also supports alerts, notifications given on wrong data-entry (figure 6). Alerts are also present in MIDP 2.0 (class `Alert`. These things are not possible with the current SEESCOA XML and thus SEESCOA XML, as is, does not meet the criterion of expressiveness. The XForms specification on the other hand does not provide layout-hints, nor provisions for a canvas that can be used for image display, etc.

Both HLUID allow customization and thus both satisfy the third requirement. Both specifications are also compact by design and thus satisfy the fourth requirement.

Type	MIDP 2.0 Controls	XForms
text	TextF.(ANY)	string
numeric	TextF.(NUMERIC)	integer
URL	TextF.(URL)	anyURI
email	TextF.(EMAILADDR)	*xpath*
phone number	TextF.(PHONENUMBER)	*xpath*
decimal	TextF.(DECIMAL)	decimal
date	DateF.(DATE)	date
date/time	DateF.(DATE_TIME)	dateTime
time	DateF.(TIME)	time

Figure 7: Typing contraints for Textfield and Datefield in MIDP 2.0 and corresponding XForms typing

The presentation of the controls can be based on the type of data in XForms and can also be customized using style sheets. The support in current software for advanced styling is limited but recommendations in development promise standardized support through style sheets. Support for data-specific controls is not possible using the original SEESCOA XML, however the controls specified in XML can have designer-specified instantiations.

The last requirement is that of the possibility of both local and remote interaction. This is a drawback of the XForms 1.0 specification; it is mainly targeted at web applications and thus there is no specification for local interaction, furthermore the interaction with the application logic is limited to submission of data.

In order to meet all our requirements and minimize the implementation required for rendering on e.g. a MIDP 2.0 device, we will make the controls of SEESCOA XML type-aware. A set of minimal supported data-types will be provided, data types as well as the "concepts" are described in a separate section, the earlier given example in the new SEESCOA XML is shown in figure 9. The full specification of the actions will also reference to the data in this section rather than the controls showing the data (not shown in the figure).

INTEGRATION IN TASK-BASED NOTATION

The previous sections discussed the use of XForms and SEESCOA XML as a high-level user interface description language, without considering the specification in which it will be incorporated. This section will discuss the use of HLUID in the task model.

In the discussion of Dygimes, we mentioned that tasks are annotated wit HLUID. This annotation ensures that navigation and composition can be derived from the Enabled Task Sets (ETS), which are defined as "a set of tasks that are logically enabled to start their performance during the same period of time" in [12]. Fig. 10 shows a preliminary version of the annotation tool.

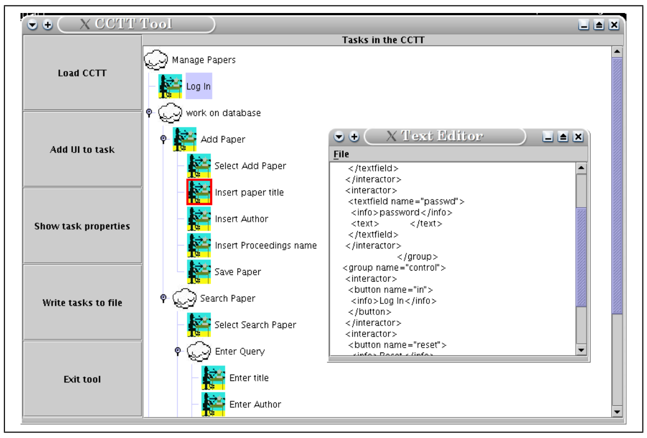
Integration of the adapted SEESCOA XML into the

control	SEESCOA XML	XForms	MIDP 2.0
text input	textfield	input	TextField/TextBox
multiline textfield	textfield	textarea	TextField/TextBox
single selection	choice (choicetype=”single”)	select1	ChoiceGroup/List
multiple selection	choice (choicetype=”multiple”)	select	ChoiceGroup/List
password field	*not supported*	secret	TextField (PASSWORD)
information display	label, canvas	output	StringItem, Canvas
range	range	range	Gauge (int)
upload file	*not supported*	upload	*not supported*
command	button	trigger, submit	Command
group	group	group	Form

Figure 8. Supported controls by SEESCOA XML, XForms and MIDP

```
<ui>
<title>Make Appointment</title>
<group>
    <group name="exampleGroup">
        <interactor>
            <range name="hour"
      ref="/application/hour"
               model="app">
            <info>hour</info>
            <min>0</min>
            <max>23</max>
            <tick>1</tick>
            </range>
        </interactor>...
        <interactor>
            <textfield name="description"
      ref="/application/description"
      model="app" type="multiline">
            <info>Description</info>
            </textfield>
        </interactor>...
        <interactor>
        <button name="ok">
            <info>OK</info>
            <action type="dmi">
               <!-- action description -->
            </action>
        </button>
        <constraints>
            <!-- spatial contraints>
        </constraints>
    </group>
</group>
</ui>
```

Figure 9. Example new SEESCOA XML



```
   </textfield>
  </interactor>
  <interactor>
   <textfield name="passwd">
    <info>password</info>
    <text>         </text>
   </textfield>
  </interactor>
                  </group>
 <group name="control">
  <interactor>
   <button name="in">
    <info>Log In</info>
   </button>
  </interactor>
  <interactor>
   <button name="reset">
```

Figure 10. The CTT annotation tool

CTT will create a partial duplication of information since both specification provide their own method of specifying data relevant to respectively a HLUID and a task. This, together with the need for specification of context within the model to enable creation of context-sensitive user interface, led to the creation a separate specification, Dygimes XML. The specification will consist of three parts: specification of *concepts*, specification of *context* and specification of tasks as can be seen in figure 11. Tasks in Dygimes XML can have HLUID attached to them. The HLUID descriptions are specified using the adapted SEESCOA XML with references to the *concepts* for data-binding and type information. Both the concepts and the context will be specified using the XForms model.

CONCLUSION

We have discussed high-level user interface descriptions on mobile devices using two examples, SEESCOA XML and XForms, and their application to mobile devices. We did this by identifying requirements and evaluating to which extend these specifications satisfied those requirements. We found that the existing SEESCOA XML was reasonably suited for the specification of user interfaces for mobile systems. However, the lack of knowledge about the type of the manipulated data and

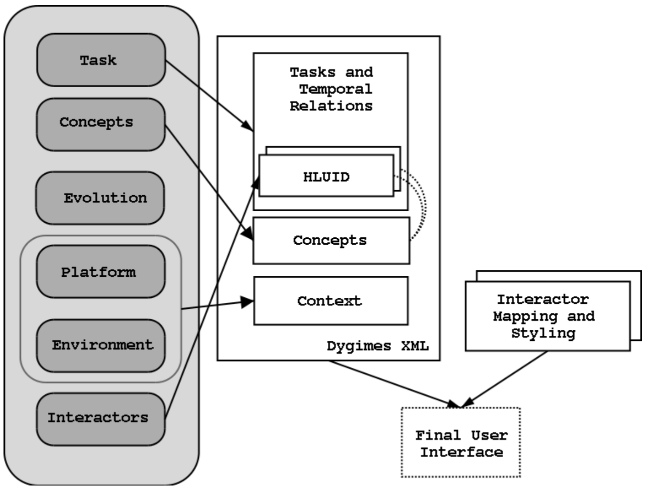


Figure 11. CCTT specifications

the direct link with widgets to extract data were noted as required features for the generation of flexible interfaces from a HLUID.

We specified adaptations to SEESCOA XML, necessary to fulfill all requirements. The new specification of SEESCOA XML will be made available as part of the Dygimes XML specification, which will integrate the XForms data model, at the time of the workshop. We have started implementation of a MIDP 2.0 renderer for the (new) SEESCOA XML and support for the full Dygimes XML is planned.

ACKNOWLEDGEMENTS
Our research is partly funded by the Flemish government and European Fund for Regional Development. The SEESCOA project IWT 980374 (1999-2003) is directly funded by the IWT (Flemish subsidy organization).

REFERENCES
1. Marc Abrams, Constantinos Phanouriou, Alan L. Batongbacal, Stephen M. Williams, and Jonathan E. Shuster. UIML: An appliance-independent XML user interface language. *WWW8 / Computer Networks*, 31(11-16):1695–1708, 1999.
2. Mir Farooq Ali and Marc Abrams. Simplifying construction of multi-platform user interfaces using uiml. In *Proceedings of UIML 2001*, March 8–9 2001.
3. Gaëlle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Nathalie Souchon, Laurent Bouillon, and Jean Vanderdonckt. Plasticity of user interfaces: A revised reference framework. In *Task Models and Diagrams for User Interface Design*, pages 127–134, Bucharest, Romania, July 18-19 2002. TAMODIA 2002.
4. Karin Coninx, Kris Luyten, Chris Vandervelpen, Jan Van den Bergh, and Bert Creemers. Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems. In *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI 2003*, pages 256–270, Udine, Italy, 8–11 2003. Springer.
5. World Wide Web Consortium. *XForms 1.0, W3C Recommendation 14 October 2003.* World Wide Web, http://www.w3.org/TR/2003/REC-xforms-20031014/.
6. danm@netscape.com. *Introduction to a XUL Document.* World Wide Web, `http://www.mozilla.org/xpfe/index.html`, october 1999.
7. Jan Van den Bergh, Kris Luyten, and Karin Coninx. A Run-time System for Context-Aware Multi-Device User Interfaces. In *HCI International 2003, Volume 2, Crete, Greece*, pages 308–312, June 2003.
8. Jacob Eisenstein, Jean Vanderdonckt, and Angel Puerta. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In *IUI 2001 International Conference on Intelligent User Interfaces*, pages 69–76, 2001.
9. Luisa Marucci, Fabio Paternò, and Carmen Santoro. *Supporting Interactions with Multiple Platforms Through User and Task Models*, pages 217–238. Wiley.
10. Nikola Mitrović and Eduardo Mena. Adaptive user interface for mobile devices. In *Interactive Systems. Design, Specification, and Verification. 9th International Workshop DSV-IS 2002, Rostock (Germany)*, pages 47–61. Springer Verlag, June.
11. Jeffrey Nichols, Brad Myers, Thomas K. Harris, Roni Rosenfeld, Stefanie Shriver, Michael Higgins, and Joseph Hughes. Requirements for automatically generating multi-modal interfaces for complex appliances. In *IEEE Fourth International Conference on Multimodal Interfaces*, pages 377–382, 2002.
12. Fabio Paternò. *Model-Based Design and Evaluation of Interactive Applications.* Springer, 2000.
13. James E. Van Peursem. *JSR 118: Mobile Information Device Profile 2.0.* World Wide Web, `http://jcp.org/en/jsr/detail?id=118`.