Dynamically generated multi-modal application interfaces - position paper -

Stefan Kost
TU Dresden, HTWK Leipzig
st_kost@gmx.de

ABSTRACT
This work approaches dynamic multi-modal application interfaces from a new point of view. The ongoing diversification of the user base and technology lays the foundation for the need of an holistic adaption infrastructure. Only designing individual adaption methods is not sufficient anymore. Providing such an infrastructure along with an open reference implementation is the objective of the Generalized Interface ToolKit (GITK) project. The software can generate, adapt and exchange interfaces at runtime. It works on various platforms and comes with several interface renderers. The solution is based on XML technology and defines an own markup language called Generalized Interface Markup Language (GIML).

Author Keywords
adaptive systems, adaptable systems, dynamic multi-modal application interfaces, UIMS

INTRODUCTION

In the last years a new development took place in our world. This work refers to it as *diversification*. It shows as two separate effects:

- **"technification" of all areas in life**
  Humans are outnumbered by technical devices already or real soon! Already these days it is nearly unavoidable to get in touch with technology. Therefore technology must be made accessible to everyone, everywhere and at every time not just physically.
- **"computerization" of devices**
  Many technical devices are more often multi-purpose appliance like computers. They have facilities for interaction and share basic common tasks.

So the challenge is to enable all the technology to all the people. This leads to a multi-dimensional adaption problem. The presumption that can be made here is, that software needs to adapt much more than it already does. Adaptation needs to work in a media-neutral fashion. It should be understood as a continuous process and not as something that happens once.

Interaction

This work deals with interfaces (see section Interfaces later in this article). Interfaces exist for the purpose of allowing *interaction*.

DEFINITION 0.1 (INTERACTION). *Interaction is the process of two or more systems exchanging data to perform a task.*

Participants of an interaction are called *interaction partners* or *interactors*. An *interactor* is a system with a variety of input and output channels. Each of these channels can submit data or stimuli of a certain type (like e.g. sound , visuals or touch). These submissions are subject to interpretation by the receiving system. The sum of the bandwidth

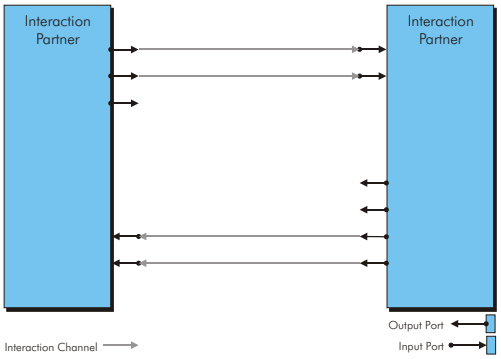

Figure 1: undisturbed interaction scenario

of the interaction channels defines the *potential interaction capabilities* of the system. These potential capabilities can only be used in the optimal case, where no obstacles hinder the interaction. The environment has such a blocking effect. Therefore the *effective interaction capabilities* are what remains after the varying blocking effect of the environment has been taken into account. [Stary, 1996, Dix et al., 1997]
It is obvious that the chances for successfully establishing enough links for an efficient interaction are not very good in the case shown in figure 2.
This work focuses on human-computer interaction and computer-computer interaction. To assure effective interaction, adaption is needed. For human-computer interaction it is desirable that the computer adapts to the needs of the human interaction partner. Using adaptive interfaces in the fields of

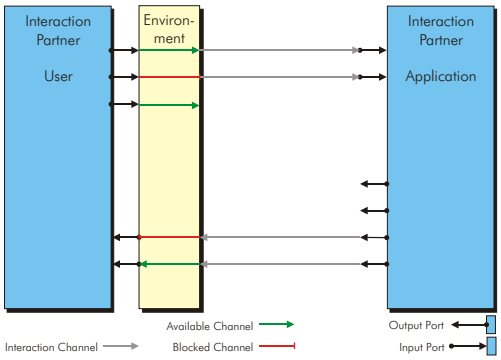

Figure 2: interaction scenario with environmental influence

computer-computer interaction allows to easily reuse an application in a different environment.

Adaption

Adaption is a key concept in this work, but also in the real world. Therefore a definition for the context of this work is required.

DEFINITION 0.2 (ADAPTION). *Adaption is the process of changing an object so that it complies to given requirements.*

One conclusion from this definition is that the object needs to be adaptable at all. It needs to offer different modes of operation that can be matched with the requirements. There are two kinds of adaption:

- **passive adaption** or **adaptable system** = system will be manually adapted by an external entity
- **active adaption** or **adaptive system** = system adapts itself automatically

[Fink et al., 1996]
Adapting an object needs knowledge about what changes are necessary for a desired effect. The overall knowledge regarding to adaption can be broken down on the base of single aspects.

DEFINITION 0.3 (ADAPTION METHOD). *An adaption regarding to one single aspect of the adaption object is the application of an adaption method. The method describes which changes are needed for specific requirements.*

The design of a good adaption method for human users requires knowledge from fields like cognitive science and psychology.
As presumed earlier in this article software needs to adapt. More precisely the interfaces are the *objects* to be adapted. The adaption process is controlled by parameters, the *adaption requirements*. In the case of this work these requirements are *adaption profiles* and consist of *environment profiles* and *user profiles*.

DEFINITION 0.4 (USER PROFILE). *The user profile describes the adaption requirements of the user and consists of the following parts:*

- *the **interaction capabilities** of the user as a communication partner. Interaction capabilities are a compound of the sensorical and motorical capabilities.*
- *the **interests and preferences** of the user relating to the style of the interaction.*
- *the **knowledge and competence** of the user regarding to the task.*

Defining an individual user profile is called *user modeling*. [Fink et al., 1997]

DEFINITION 0.5 (ENVIRONMENT PROFILE). *The environment profile describes a filter that applies to the capabilities part of the user profile. It temporarily restricts or even blocks certain interaction capabilities of the user.*

While the user profile can be seen as an object with nearly static properties, the environmental profile needs to be considered as highly dynamic.
To adapt an interface usually multiple adaption methods need to be applied. It sounds sensible to define an *adaption infrastructure* that handles the application of adaption methods. Designing such an infrastructure requires engineering skills from the area of computer science. Therefore the separation into adaption methods and adaption infrastructure reflects the relation to different areas in science.
Figure 3 graphically shows the relation of the previously defined terms for the scenario of human-computer interaction. Finally a short summary can be given:

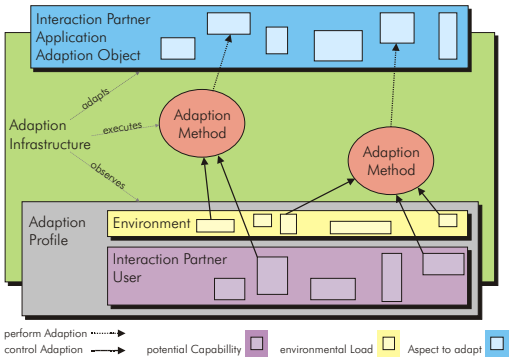

Figure 3: schematics of adaption and involved components

- the users' potential capabilities plus the current environment form the adaption profile
- the adaption profile controls the adaption process
- the adaption infrastructure provides means to read the profile and to choose and execute the respective adaption method
- the adaption method performs the adaption of one aspect of the application interface according to the requirements given by the adaption profile

[Stary, 1996, Dix et al., 1997]

Interfaces

DEFINITION 0.6 (INTERFACE). *An interface provides well defined access to functionality of an object from outside. It appears as a layer between two parties and aids their interaction.*

In [Phanouriou, 2000b] an interface is separated into four aspects:

- **structure**: the organization of interface objects
- **content**: resources used in the interfaces such as label texts and shortcut metaphors
- **style**: the presentation of interface objects
- **behavior**: defines the action to be performed on interaction with the interface objects

The application needs to provide structure, behavior and content. The presentation and choice of the content (e.g. for i18n) is dependent on the modality of the interface and the user profile. Therefore these aspects will be chosen by the adaption infrastructure. Finally style is an aspect that should be provided and handled by the adaption infrastructure.

Interface models

In the past various models for decoupled interface architectures have been suggested. A general criticism on models such as Seeheim, Arch, MVC and PAC is, that they aim to model adaptive systems, but do not represent the adaption process as such. These models only decouple components, but lack a definition of how adaption is driven (how to couple the right component-instances at run-time). Seeheim and Arch span a series of components between user and application, neglecting that there is an environmental influence affecting the interaction and that a user might carry out several tasks synchronously. In the past, when the models such as Seeheim and Arch have been defined, these two effects were hard to take into account for technical reasons or were less important. This has changed in the present. [Pfaff, 1985, various contributors, 1992]

Existing approaches

The goals of this work are similar to those of other projects. A big share of them either became dormant (AUIML) or seemingly have been discontinued (XIML). Another group of projects focuses on adaptive hypermedia applications. These projects usually develop adaption methods for their purpose and then an infrastructure to drive them. Furthermore there are solutions such a UIML and XUL which are active. XUL focuses on graphical interfaces only. It mainly serves as a operation system portability layer. Interfaces generated by UIML and XUL can not change their modality at runtime. With UIML the developer needs to specify all target interface variants that should be available later. Both languages use XML as a ”input file-format”. [Phanouriou, 2000a, Hyatt, 2000]. Finally some projects are quite similar like W3C XForms, but started in parallel or later as this work. XForms maintains separate XML documents for content and interface [Dubinko et al., 2003]. All approaches mentioned above have in common that they do not aim to provide a system, where interfaces can be adapted or even exchanged at runtime.

Aim of this the GITK project

The previous sections motivated that it is useful to distinguish between the *adaption infrastructure* and *adaption methods*. They further showed that an adaption has a multi-dimensional nature. Therefore a holistic approach to adaption is needed. Applications using this technology would then be adaptable, as they will use a pure functional interface description as an input and leave the generation of a concrete interface to the system.
In parallel more research is required in user modeling to define rich user profiles that can control the adaption process. This would turn the adaptable systems into adaptive systems. This work focuses on providing a fundamental adaption infrastructure, with a strong decoupling of application logic and interface presentation. On top of that a limited number of adaption methods will be implemented as a proof-of-concept. However it is not the objective of this project to develop new adaption methods, nor evaluating them.

THESIS

This work will show, that:

THESIS 0.1. *By limiting the presentational complexity a much greater universality can be achieved.*

THESIS 0.2. *There is no reason for adaptive solutions to mainly concentrate on graphical presentation.*

THESIS 0.3. *It is possible and even desirable to separate style related description from functional interface description.*

THESIS 0.4. *An interface can be generated, without the application needing to provide adaption profiles for different targets. In other words: even adaption methods can be generalized.*

THESIS 0.5. *It is possible and preferable to always have a default behavior, that can be overridden by adaption, instead of only relying on the adaption.*

THESIS 0.6. *A solution can be based on many standardized and well established technologies. In fact it can glue many specific solutions together, which already exist.*

THESIS 0.7. *Beside humans an application can be an end-user as well and therewith benefit from an adaptive solution.*

THESIS 0.8. *Adaptive technology is necessary for everyone and not just for minorities (like elderly or disabled people).*

SOLUTION

The solution presented in this work is called Generalized Interface ToolKit (GITK) and consists of three parts:

- an architecture related to the arch model that fits with the formerly defined adaption structure

- a domain independent markup-language that is called Generalized Interface Markup Language (GIML)
- a domain independent interface object hierarchy that is based on a canonical interface object naming scheme

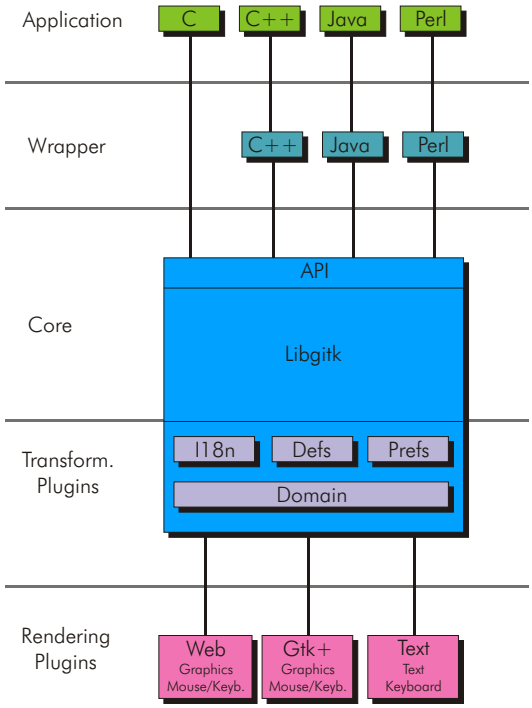

Figure 4: GITK architecture

The architecture part as shown in figure 4 has been implemented as a multi-layered software system. As a major difference to approaches like UIML, GITK not requires domain specific adaption of interfaces. The required domain specific knowledge is captured in the design of the rendering component. This sounds like a more practical approach as the application developer usually not has the knowledge and resources to cover all possible target domains. When designing a rendering component specialists can be included in the development team. The architecture presented here, can be extended, to allow applications to provide hints to the domain specific adaption on demand. This would be necessary when a generic solution is not enough.
A second key difference is that the XML interface description is used as an active dialog model. That means that the adaption processing heavily relies on XML technology such as XSLT, XPath and XML Namespaces. The advantage of this is, that there is no discontinuity in the use of technology that is processing the model in the transformation pipeline (see figure 5). The pipeline itself is maintained by the core library. This includes construction of a pipeline for a specific renderer, executing it and synchronizing both ends. At run-time the application feeds a dialog description into the pipeline. This serves as a structure onto which all variable aspects of an interface are overlaid. Then the core library initially applies all transformations to build the dialog description that the renderer understands. Each step of the pipeline adds or reconfigures aspects of the interface towards the requirements of the user. When the user works with the interface, all state changes such as navigation and data-entry are written back to the XML dialog model by the renderer (at the renderers end of the pipeline) and are synchronized with the applications end of the pipeline by the core library. This mechanism decouples the application and the interface instance even at runtime.

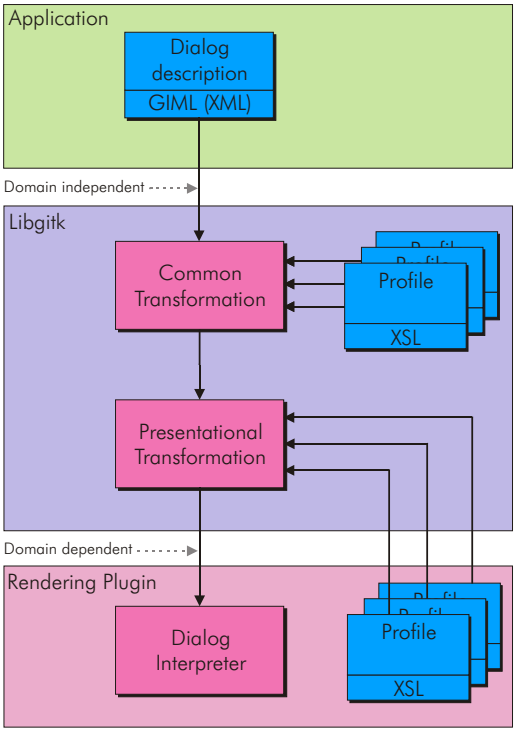

Figure 5: GITK processing pipeline

GIML is defined by a document type definition (DTD). This is currently being exchanged with W3C Schema. The markup language uses namespaces to separate the various aspects of dialogs (see section Interfaces) and namespaces do not work well with DTDs.
All interface objects are identified by a type. The type hierarchy uses only functional names. Thereby a ”push button” becomes an ”action”, as when used e.g. in the voice domain a ”push button” is not a meaningful concept. This abstraction layer allows each renderer to associate a domain dependent representation with the domain independent name.
Figure 6 shows an example dialog definition. One can see that the GIML is relative terse. It is important to note that the example only shows the input document. The application adds dynamic aspects like behavior at run-time by using the ore library API. The GITK software package comes with an introspection mechanism to look inside the XML pipeline at run-time.
This work comes with a free reference implementation. It is available as an active open-source project at http://gitk.sourceforge.net. The system is light-weight and portable. It is developed mostly in C and requires only a few libraries like glib and libxml2. It has been successfully tested on several Unix/Linux and Windows systems. The project consists of a core package, various renderers (text, gtk, opengl, phone, ...) and a set of examples. The core package comes with a

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE giml SYSTEM "http://gitk.sourceforge.net/giml.dtd">
<!-- $Id: gitkHelloUser_main.xml.in,v 1.7 2004/04/01 12:17:27 ensonic Exp $
 * @file    gitkHelloUser_main.xml
 * @author  Stefan Kost <ensonic@users.sf.net>
 * @date    Thu Jan 17 11:22:38 2002
 *
 * @brief  main dialog for gitkHelloUser.c
 * @ingroup gitkexamples
 *
-->
<giml xmlns="http://gitk.sourceforge.net/"
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:i18n="http://apache.org/cocoon/i18n/2.0">
    <dialog>
        <meta>
            <dc:title><i18n:text>query user identity</i18n:text></dc:title>
        </meta>
        <dialogwidgets>
            <dialogwidget id="Okay"/>
            <dialogwidget id="Cancel"/>
        </dialogwidgets>
        <widgetgroup>
            <label><i18n:text>identity</i18n:text></label>
            <widget id="UserName" type="characterinput_alphabetic">
                <label><i18n:text>user name</i18n:text></label>
                <disabled>true</disabled>
            </widget>
            <widget id="Sex" type="optionchoice_single_compact">
                <label><i18n:text>sex</i18n:text></label>
                <options>
                    <option><i18n:text>male</i18n:text></option>
                    <option><i18n:text>female</i18n:text></option>
                </options>
            </widget>
        </widgetgroup>
    </dialog>
</giml>
```

Figure 6: GIML dialog example

browser based management console, that allows to inspect the internals of the system and to simulate changes in the adaption requirements. [Kost, 2003]

CONCLUSION

The article started with a theoretical foundation. The terms related to interaction and adaption have been precisely defined. The problem analysis showed that the adaption problem has a multi-dimensional nature. This finding even more justified the separate exploration of *adaption infrastructure* and *adaption methods*. Then the objective of adaption - the interface - has been covered.

In the previous section a new adaption infrastructure that is able to integrate all kinds of adaption methods has been introduced. It is important to note that not only the XML language as such solves the problem of an abstract interface architecture. The interplay of language and architecture is what provides a flexible system. GITK reaches this goal by its pipeline concept and the intensive use of XML technology. The presented solution meets the requirements to design a holistic approach towards adaption. The included examples show the adaptability and the partial adaptiveness. The choice of examples outlines the kind of applications the GITK approach is useful for - administration tools, information management (CMS,PIM) software - all applications where a highly available clean interface matters more than a polished interfaces presentation. A second target group are rapid prototyping systems, as for these GITK can act as a interface prototype run-time environment. It would be interesting to research if a GITK interface description can be generated from an UMLi (Unified Modeling Language for Interactive Applications) model or even from a XML schema definition [Norman W. Paton and Paulo Pinheiro da Silva, 2002, Sperberg-McQueen and Thompson, 2004].

The current state of the project mainly affects software development and not yet the user, as it focuses on the adaption infrastructure and not the adaption methods.

FUTURE

To turn adaptable applications into adaptive systems *user profiles* are needed. User modeling and the ongoing technological development contribute towards that. Furthermore future devices will have more *sensors* to read from the environment.

Integrating such dynamic profiles and related adaption methods into the GITK infrastructure will extend the solution towards more kinds of applications.

REFERENCES

Dix, A. J., Finlay, K. E., Abowd, G. D., and Beale, R. (1997). *Human-Computer Interaction*. Prentice Hall, Pearson Education Limmited.

Dubinko, M., Klotz, L. L., Merrick, R., and Raman, T. V. (14 October 2003). Xforms 1.0. http://www.w3.org/TR/xforms/ , http://www.w3.org/MarkUp/Forms/.

Fink, J., Kobsa, A., and Nill, A. (1996). Useroriented adaptivity and adaptability in the avanti project. http://citeseer.ist.psu.edu/fink96useroriented.html.

Fink, J., Kobsa, A., and Nill, A. (1997). Adaptable and adaptive information access for all users, including the disabled and the elderly. In *Proceedings of the Sixth International Conference UM97*. Springer Verlag, Wien, New York.

Hyatt, D. (30 March 2000). The xptoolkit architecture. http://www.mozilla.org/xpfe/xptoolkit/index.html.

Kost, S. (2000-2003). Gitk - generalized interface toolkit. http://gitk.sf.net.

Norman W. Paton and Paulo Pinheiro da Silva (February 27, 2002). Umli - unified modeling language for interactive applications. http://www.cs.man.ac.uk/img/umli/index.html.

Pfaff, G. E. (1985). User interface management systems. In *EurographicSeminars*. Springer Verlag, Berlin Heidelberg New York Tokyo.

Phanouriou, C. (1999,2000a). Uiml - user interface markup language. http://uiml.org/.

Phanouriou, C. (2000b). *UIML: A Device-Independent User Interface Markup Language*. PhD thesis, Virginia Polytechnic Institute and State University.

Sperberg-McQueen, C. M. and Thompson, H. (17 Mar 2004). Xml schema. http://www.w3c.org/XML/Schema.

Stary, C. (1996). *Interaktive Systeme*. Friedr. Vieweg & Sohn Verlagsgesellschaft mbH.

various contributors (1992). A metamodel for the runtime architecture of an interactive system. *UIMS Tool Developers Workshop 1992: In SIGCHI Bulletin. 24(1). pp 32-37.*