Supporting Cross-Application Contexts with Dynamic User Interface Fusion

Pascal Bihler and Holger Mügge

Institute of Computer Science III, University of Bonn
Römerstr. 164, 53117 Bonn, Germany
{bihler, muegge}@iai.uni-bonn.de

Abstract: Currently, user interfaces are defined by single applications but workflows may span multiple ones. In the result, users have to work with separated programs in parallel and deal with separated interfaces to fulfill their tasks. In desktop computing with extensive screens, working with several opened windows might not be eligible but bearable. For mobile devices with very restricted interaction capabilities (e. g. scarce screen estate), the limitations of traditional UIs become a barrier. Therefore we argue that not applications but workflows and tasks should be the determinant for the user interface. This paper introduces in the research question about how to identify the user's current task and focuses on how to reorganize user interfaces across multiple concurrently used applications.

1 Motivation and Introduction

When using desktop computers or ubiquitous devices like PDAs, features required to complete a given task usually span across different applications. In ubiquitous computing scenarios, display resources and user interaction options are typically very limited. Current mobile operating systems tend to display only the user interface of the application in focus, hiding other applications which might be relevant to the current workflow. In our recent publication [BK07], we call this behavior *greedy screen allocation*. Greedy screen allocation disrupts the user's perception of information and actions relevant to her working context. In order to find the hidden information again, the user is forced to switch to another application, loosing the user interface of the first application out of sight. Lack of orientation and a decreasing productivity might be the consequences. Therefore we argue that not the predefined interface of an application should define the UI view but workflows and tasks should be the determinant for user interfaces.

The first arising question is: how can we identify the user's current tasks and workflows? Since this is out of focus of this paper, we assume the working context to be available in form of an aggregated sensor.[1]

[1]Such a sensor will take several basic context data into account, like calendar entries, interaction history, running programs, existing indexed documents, accessible devices e. g. printers, incoming or currently open phone calls, etc.

In this paper, we focus on two other research questions: What kind of preparation is needed to break down the applications into features that can be provided with separate UIs? How can we reorganize and put together the user interface of multiple cooperatively used applications? Section 3 discusses these questions and provides the model-based approach of interface fusion. In section 4 we present related work that can be helpful for further investigations. Finally, in section 5 we give an outlook to planned future work and summarize the findings of this paper.

2 From Applications to Tasks

Even in ubiquitous computing where the computing unit and devices become embedded and interaction becomes seamless, adaptation of programs and user interfaces needs to be somehow traceable by the user. This traceability is assured if the users perceives adaptations as being related to his current interaction or working context. At the same time the adapting system is required to preserve interaction flows, i. e. the relation between the current interaction and the corresponding processing application. For example, an adaptation of the user interface should not lead to confusion by redirecting the voice input during a call away from a phone application to a karaoke machine, or more generally the mouse and keyboard focus from the current interface element to another one without any noticeable mediation.

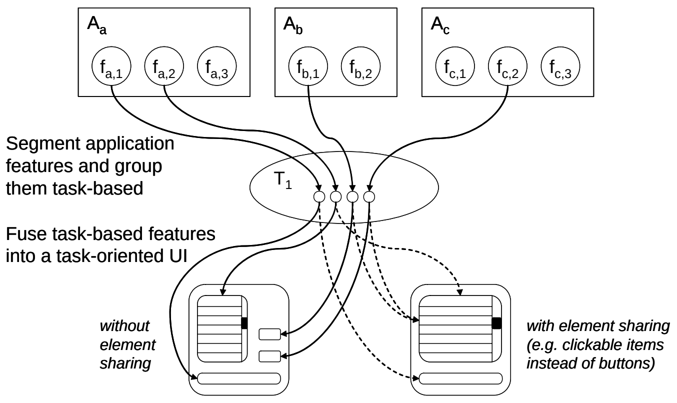
To achieve the goal of plausible UI adaptation the system is required to infer the user's current workflow from given data like the runtime environment or sensor input. In addition, to support cross-application tasks and guide perception focus, a way has to be found to merge relevant UI elements from different independent applications. As a first step for realization, we introduce a feature-oriented system perspective and concretize the raised research questions.

In a mobile and pervasive computing environment, borders between applications providing distinct features need to be blurred and controlling programs need to be interwoven. To support this adaptation, the following basic concepts are required:

Segmentation Developers are required to identify the independent features of their application. Even if still bundled together in one program, internally they are kept separately or split up. This might finally lead to different visual segments or other interactions (e. g. access to microphone, speakers etc.).

Grouping Due to the current task, features from concurrently running applications are grouped together and prioritized for visualization.

Fusion Different UI segments are merged into one perceivable unit, e. g. features of different applications are shown in an integrated manner. There are different concrete techniques of interface fusion: *Area Fusion* (which brings multiple UI elements into one shared area, but they still use different widget instances) and *Element Fusion* (different features use the same widget instance in a mixed mode, e. g. a list as shown in fig. 3(b))



A_a
$f_{a,1}$
$f_{a,2}$
$f_{a,3}$
A_b
$f_{b,1}$
$f_{b,2}$
A_c
$f_{c,1}$
$f_{c,2}$
$f_{c,3}$
Segment application features and group them task-based
T_1
Fuse task-based features into a task-oriented UI
without element sharing
with element sharing (e.g. clickable items instead of buttons)

Figure 1: Application features are segmented, grouped together due to a given task context and displayed using Dynamic User Element Fusion

3 Model-based Reorganization of Application Features

Context-based dynamic user interface fusion raises several challenges: identifying available features of concurrently running applications, extracting their interface elements relevant in a certain context, re-arranging them while balancing their competing requirements and achieving device constrained interface rendering in a flexible and portable way.

Feature segmentation requires semantic knowledge of application- or service-provided features and their user interfaces. A dynamic rendering of user interfaces against diverse and changing display parameters requires knowledge of the display element's meaning. This correlation motivates the use of model-based or semantic interface descriptions.

In our approach, sketched in Figure 2, the *feature providers* have to describe the semantics of each available feature and its user interfaces. For this, model-based UI reseach has already yielded a set of suitable desciption languages, e. g. XIML [PE01] or XForms [BLMR04]. The described interface elements are put into mutual relationship by the *Context-Aware Interface Decorator*, based on the current working context. The context and the identified semantic relationships between elements are themselves parameters for a prioritization of the interface elements.

The semantic description of the available interfaces, decorated with relationship and priority information, is the input for a *Semantic Interface Layout Engine*. It is the task of this engine to render the actual device's user interface taking into account device-specific constraints. Devices in mobile and ubiquitous computing differ in physical capabilities such as screen size and resolution, single or multiple output devices, and different interaction-patterns such as multi-touch, pen-input, etc. This may result in very different layouts for the same fused user interface. Work in this direction has already been carried out, as for example in [EVP00], and might provide a foundation for the Layout Engine.

The simplest case of dynamic user interface fusion is *Area Fusion*: A *working context* im-

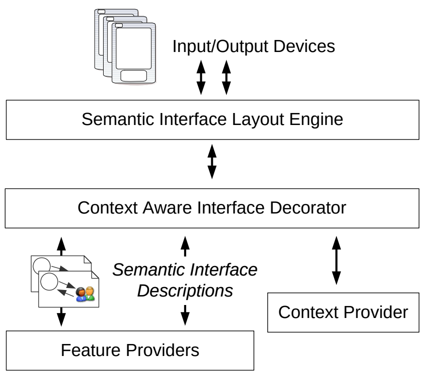

Input/Output Devices
Semantic Interface Layout Engine
Context Aware Interface Decorator
Semantic Interface Descriptions
Context Provider
Feature Providers

Figure 2: Semantic descriptions of the user interfaces of application's features are mutually related and prioritized based on the current context before rendered as actual user interface.

plies the important features for the current workflow, so their according interfaces need to be rendered together. A more complex interface fusion is *Element Fusion*, where different features share the screen area and functionality of a particular interface element.

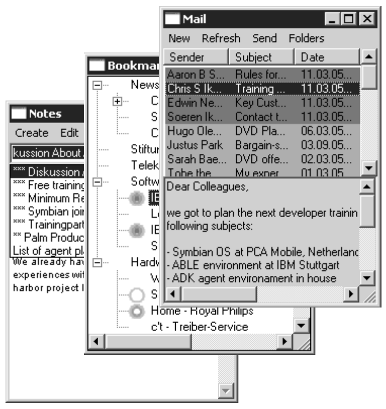
In our example, the user is about to have several business meetings at a fair, and related documents saved on her mobile device need to be at hand. This is supported by context-sensitive applications that filter and sort their documents according to the closest fair booth (see figure 3(a) and [MRS$^+$07]). Nevertheless, the user has to manually switch between them when requesting information provided by another feature. With *Element Fusion*, all applications present their information in a common control, e. g. a listbox (see figure 3(b)).

4 Related Work

Several approaches for dynamic user interface adaptation have been proposed based on models such as roles and tasks [PS00, Nov06], data input- and output-flows and annotations [NFHS05] or visual component composition [XPJK03]. These approaches adopt an application-based view and do not yet take cross-application interface fusion into account.

Fine-grained adaptability of interfaces is the research aim on plasticity of user interfaces, as described for example in [Thé01]. Bisignano et al. [BMT05] sketch a framework for adapting user interfaces on devices for ubiquitous computing. The authors adapt the GUI and the displayed content but focus only on single applications and do not take the whole collection of features provided by different applications on pervasive devices into account. Another approach for interface adaptation, described in [STB04], uses the AMF interaction model in order to deduct requirements from Abstract Interaction Objects.

A number of abstract interface definition languages and interpreters have been proposed in the last years (cf. XIML [PE01] and UsiXML [LVM$^+$04]). *XIML* is one of the few abstract interface languages considering a "dynamic presentation reorganization". Trans-



kussion About

*** Diskussion
*** Free training
*** Minimum Re
*** Symbian joir
*** Trainingpart
** Palm Produc
List of agent pl
We already hav
experiences wit
harbor project l

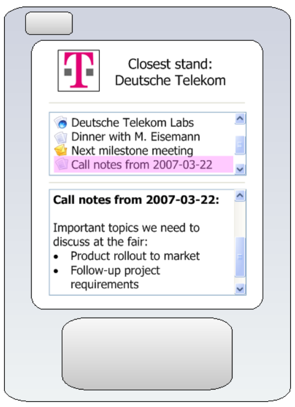
Sender	Subject	Date
Aaron B S...	Rules for...	11.03.05...
Chris S Ik...	Training ...	11.03.05...
Edwin Ne...	Key Cust...	11.03.05...
Soeren Ik...	Contact t...	11.03.05...
Hugo Ole...	DVD Pla...	06.03.05...
Justus Park	Bargain-s...	03.09.05...
Sarah Bae...	DVD offe...	02.03.05...
Tobe the	My exper	01.03.05

Dear Colleagues,

we got to plan the next developer trainin
following subjects:

- Symbian OS at PCA Mobile, Netherlanc
- ABLE environment at IBM Stuttgart
- ADK agent environament in house

(a) Three context-sensitive applications are used to deal with a single task


Closest stand:
Deutsche Telekom

Deutsche Telekom Labs
Dinner with M. Eisemann
Next milestone meeting
Call notes from 2007-03-22

Call notes from 2007-03-22:

Important topics we need to discuss at the fair:
- Product rollout to market
- Follow-up project requirements

(b) Sharing of UI elements

Figure 3: Greedy screen allocation (a) urges the user to continuously switch between different applications. Hence, even with context-sensitive filtering of data much information remains out of sight. With *Element Fusion*, several applications can share a user interface element and present their information in an integrated way (b).

formiXML [SLV$^+$05] supports rule based transformations between different abstraction layers. For supporting interface reuse Lepreux [LVM06] provides a theoretical foundation for merging existing GUI layouts statically. Bergh (cf. [dBC05]) works on reflecting current context-based user interfaces.

5 Summary and Outlook

In this paper, we address the problem of supporting cross-application context with *dynamic user interface fusion*, a technique to merge interface parts from independent applications or service providers. Dynamic interface fusion offers two techniques to integrate independent interface elements into one consistent user interface, namely Area Fusion and Element Fusion. Both are based on semantic specification of the application's features and user interaction characteristics. The paper illustrates with an example how *Element Fusion* can support the use of a location-driven personal information manager.

The next step we plan is an in-depth research of the options for implementing each proposed architecture component. In parallel with solving technical issues, the interaction of users with fused interfaces has to be evaluated in practical scenarios. Enhancements in usability can lead to a more natural interaction with the computing device compared to today's disruptions. We believe, this shift represents an important step towards seamless ubiquitous computing and defines a central requirement of computing research.

References

[BK07] P. Bihler and G. Kniesel. Seamless Cross-Application Workflow Support by User Interface Fusion. In S. Boedker C. Brodersen and C. N. Klokmose, editors, *Multiple and Ubiquitous Interaction - book of abstracts*. DAIMI PB-581, University of Aarhus, 2007.

[BLMR04] J. Boyer, D. Landwehr, R. Merrick, and T. V. Raman. XForms 1.1. online, Nov 2004. http://www.w3.org/TR/2004/WD-xforms11-20041115/ (last visited: 2007-06-21).

[BMT05] M. Bisignano, G. Di Modica, and O. Tomarchio. Dynamic User Interface Adaptation for Mobile Computing Devices. In *Proceedings of the SAINT-W '05*, pages 158–161, Washington, DC, USA, 2005. IEEE Computer Society.

[dBC05] J. Van den Bergh and K. Coninx. Towards Integrated Design of Context-Sensitive Interactive Systems. *PERCOMW*, 00:30–34, 2005.

[EVP00] J. Eisenstein, J. Vanderdoncki, and A. Puerta. Adapting to Mobile Contexts with User-Interface Modeling, 2000.

[LVM$^+$04] Q. Limbourg, J. Vanderdonckt, B. Michotte, L. Bouillon, M. Florins, and D. Trevisan. USIXML: A User Interface Description Language for Context-Sensitive User Interfaces. In *Developing UIs with XML, AVI'2004 Workshop*, pages 55–62, May 2004.

[LVM06] S. Lepreux, J. Vanderdonckt, and B. Michotte. Visual Design of User Interfaces by (De)composition. In G. Doherty and A. Blandford, editors, *Proc. of the 13th Int. Workshop on Design, Specification, and Verification of Interactive Systems DSV-IS'2006*. Springer-Verlag, Berlin, July 2006.

[MRS$^+$07] H. Mügge, T. Rho, D. Speicher, P. Bihler, and A. B. Cremers. Programming for Context-based Adaptability - Lessons learned about OOP, SOA, and AOP. In *Proc. of the SAKS-Workshop*, 2007. to be published.

[NFHS05] E. G. Nilsson, J. Floch, S. Hallsteinsen, and E. Stav. Model-based user interface adaptation. In *Mobile Computing and Ambient Intelligence: The Challenge of Multimedia*, Dagstuhl Seminar Proceedings, 2005.

[Nov06] O. Novacescu. Des IHMs composables pour les applications à base de composants. Lab report, Université de Nice Sophia-Antipolis, June 2006.

[PE01] A. Puerta and J. Eisenstein. XIML: A Universal Language for User Interfaces. http://www.ximl.org/ documents/XimlWhitePaper.pdf, 2001.

[PS00] R. R. Penner and E. S. Steinmetz. Dynamic User Interface Adaptation Based on Operator Role and Task Models. In *Proc. of the 2000 IEEE International Conference on Systems, Man, and Cybernetics*, volume 2, pages 1105–1110, 2000.

[SLV$^+$05] A. Stanciulescu, Q. Limbourg, J. Vanderdonckt, B. Michotte, and F. Montero. A transformational approach for multimodal web user interfaces based on UsiXML. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, pages 259–266, New York, NY, USA, 2005.

[STB04] K. Samaan and F. Tarpin-Bernard. The AMF Architecture in a Multiple User Interface Generation Process. In *Developing UIs with XML, AVI'2004 Workshop*, May 2004.

[Thé01] David Thévenin. *Adaptation en Interaction Homme-Machine : le cas de la Plasticité*. PhD thesis, Université Joseph Fourier, December 2001.

[XPJK03] Xie Xiaoqin, Xu Peng, Li Juanzi, and Wang Kehong. A component model for designing dynamic GUI. In *Proc. of PDCAT'2003*, pages 136–140, Aug. 2003.