Seamless Cross-Application Workflow Support by User Interface Fusion

Pascal Bihler and Günter Kniesel
Institute of Computer Science III, University of Bonn
Römerstr. 164
53117 Bonn, Germany
{bihler, gk}@iai.uni-bonn.de

ABSTRACT

In a mobile and ubiquitous computing environment, it becomes evident that users perceive tasks not as connected to some specific application, but rather to some special context. Consequently, borders between applications providing distinct features are blurred and programs need to be interwoven. Features belonging to the same task or workflow need to be presented together. This presentation depends on the current application and user context, but also on the capabilities and constraints of the execution environment. In a ubiquitous computing environment, devices differ in display capabilities, input-output-interactions and user habits. This paper identifies the problem of disruptive cross-application workflows in ubiquitous computing and proposes dynamic user interface fusion to support the user in handling such workflows. In addition, a framework for dynamic user interface fusion is proposed.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Graphical user interfaces (GUI); C.5.3 [**Microcomputers**]: Portable devices; H.5.2 [**User Interfaces**]: User interface management systems (UIMS); H.5.2 [**User Interfaces**]: Theory and methods; H.5.2 [**User Interfaces**]: Screen design

General Terms

Algorithms, Design, Human Factors

1. INTRODUCTION AND MOTIVATON

Many every-day scenarios of utilizing a desktop or hand-held computer involve *cross-application workflows* that require the use of a variety of different programs in order to fulfil a given task. For instance, answering an e-mail request for the results of the recently organized conference might need access to various information sources, use of different programs to integrate the data and compile a set of concise charts, access to the address book for locating the addresses of people who should also be informed and finally return to the e-mail reader for answering the request with the compiled information.

The effectiveness of such cross-application workflows strongly depends on the ability to preserve as much as possible the relevant information of the working context when switching programs. On classical desktop computers this is not difficult. Their screen size allows users to arrange the windows of different applications next to each other or slightly overlapping so that all relevant information can still remain in view and directly accessible (see figure 1(a)).

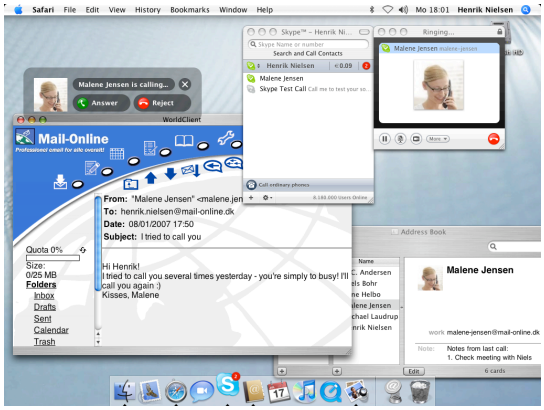
On a mobile device, the situation is radically different. Display resources and user interaction options are typically very limited. Therefore, current mobile operating systems tend to display only the user interface of the focus application, ignoring other applications which might be relevant to the current workflow (see figure 1(b)). We call this behaviour *greedy screen allocation*. Greedy screen allocation disrupts the user's perception of information and actions relevant to her working context. In order to find again the hidden information the user is forced to switch to another application, losing the user interface of her primary application (e.g. the phone controls) out of sight.

The disruption caused by greedy screen allocation is particularly annoying when the screen is 'wasted' on an application displaying interface elements that are not relevant to a working context. This occurs rather often, since typically, user interfaces of classical applications provide access to a variety of features at the same time, of which only some are really needed for a given workflow. In Figure 1(c)) we highlighted the elements that are relevant in the context in which Henrik receives a phone call from Marlene.

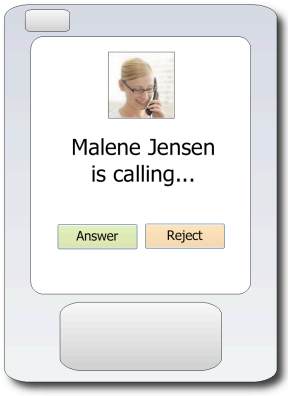
The contributions of our paper are

- identification of the problem of disruptive cross-application workflows in ubiquitous computing,
- the proposal of dynamic user interface fusion to supporting seamless cross-application workflows,
- identification of technical challenges for interface fusion not solved by existing approaches,
- a review of supporting techniques possibly applicable in this area.

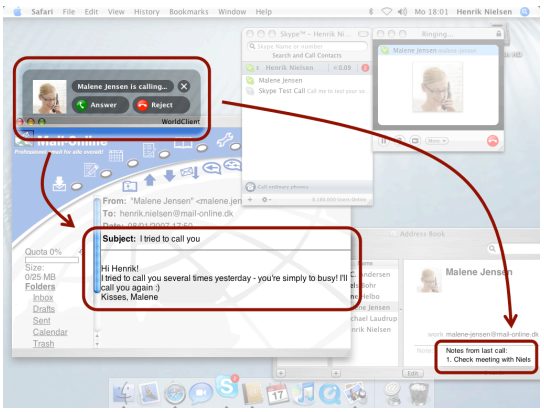
Workshop on Multiple and Ubiquitous Interaction Aarhus, Denmark, 28-30 March 2007



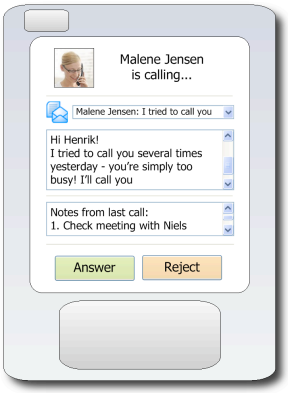
(a) Multiple applications involved in call answering



(b) Greedy screen allocation on mobile device



(c) Application-spanning dynamic context



(d) Cross-application dynamic UI fusion on a PDA

Figure 1: When an application gets the focus on a desktop computer, other applications are still visible (a). On a mobile device, just the UI of a single application is displayed at a time (b) hiding related information from other applications (c). With *display segmentation*, GUI elements of several features related to the current workflow are fused, even if provided by different information sources (d).

2. DYNAMIC USER INTERFACE FUSION

In order to support seamless cross-application workflows even on resource-constrained mobile devices, the computing system should be able to provide a task-oriented user interface that blurs the borders between different programs.

The idea of *dynamic user interface fusion* is to automatically compose the user interfaces of the features required in a particular working context, independent of the feature provider. A feature provider can be a single program, several concurrently running applications on the mobile device, unanticipated program adaptations [10] or independent services distributed in a Pervasive Service Environment [3].

The simplest case of dynamic user interface fusion is *display segmentation*: A *working context* defines several features that are important for the current workflow, so interfaces for these features need to be rendered together. Scenarios that call for *display segmentation* are:

- *Incoming phone call* (see Figure 1): On a smart phone, information about the caller (name, recent contacts, etc.) could be displayed next to the incoming call notification. Having related information in view can ease the user's decision whether to accept the call.

- *Shared use of embedded displays:* Ubiquitous computing seamlessly integrates into the users device environment, which imposes the possibility to share visualization devices for different features: A body sensor could enhance a smart wristwatch with a health checking feature, which is displayed alongside to the watch's main function.

An example of more complex interface fusion is *UI element sharing*, which means that different features share the screen area and functionality of a particular interface element. This could be the case with a *location driven information manager:* Based on the current user location, personal information coming from different programs (such as bookmarks and notes) are displayed in a common control, e. g. a listbox. In [10], we examine the use case of visiting a trade fair. When approaching a specific stall, the relevant data from unconnected sources is displayed together to support an upcoming business meeting.

3. APPROACHING DYNAMIC USER INTERFACE FUSION

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 and way.

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

Model-based or semantical interface definitions prescind from the concrete interface representation on a given device in favor of a meta-description of the user interface (see e. g. [5, 12]). In our approach, sketched in Figure 2, it is the task of the *feature providers* to describe semantically every available feature and its user interfaces. These descriptions may evolve and change over time as the user interacts with the described features and external context changes can trigger feature adaptation (and therefore UI adaptation).

The described interface elements are put into mutual relationship with each other by the *Context-Aware Interface Decorator*, based on the current execution context. This context, provided by a separate *context provider*, comprises the current user task and external parameters. The context and the identified semantic relationships between elements are themselves parameters for a priorization of the interface elements.

The semantic description of the available interfaces, decorated with relationship and priority information, is the input for a *Semantical 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. The layout of one particular interface can even change dynamically triggered by changes of the display environment when applications migrate from one device to another or new visualization devices become available. For instance, a phone application might display contact information about an incoming call on the user's wristwatch when the mobile phone's display is covered.

For automatically generated user interfaces, well-established design and usability guidelines [7] need to be taken into account. Ubiquitous and mobile computing challenges those well established rules and guidelines with new requirements [2]. The specific solutions for ubiquitous interaction interfaces are still under research.

In the next section we review know approaches that can contribute to meeting these challenges.

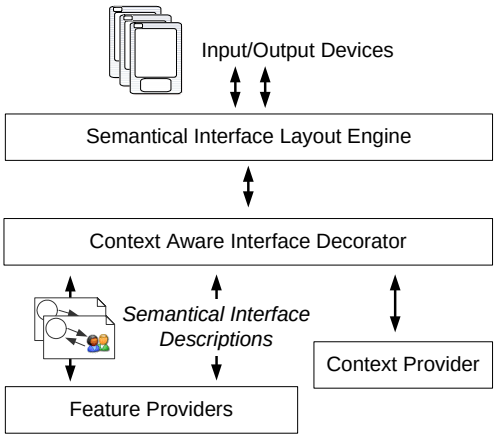


Figure 2: Semantical descriptions of the user interfaces of application's features are mutually related and prioritized based on the current context before rendered as actual user interface.

4. RELATED WORK

Different approaches for dynamic user interface adaptation have been proposed based on models such as roles and tasks [13, 12], data input- and output flows and annotations [11], or visual component composition [18]. 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 [17]. Bisignano et al. [4] sketch a framework for adapting user interfaces on devices for ubiquitous computing. The authors adapt the GUI and the displayed content but also 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 [15], 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. Recent developments are XIML [14] and UsiXML [9]. *XIML* is one of the few abstract interface languages considering a "dynamic presentation reorganization". The authors show that dynamic adaptation of an abstract interface element is possible by mapping it to different representations. However, the idea seems to be discontinued and no framework for complex application interface adaptation was proposed. In continuation of XIML *UsiXML* was developped. This language supports different levels of detail (Task & Concept, Abstract User Interface, Concrete User Interface, Final User Interface). The TransformiXML-Tool [16] supports rule based transformations between the different abstraction layers. For supporting interface reuse Lepreux [8] provides a theoretical foundation for merging existing GUI layouts statically.

The Apple iPhone [1] supports display segmentation at a very basic level, e. g. ongoing phone calls are visualized while

searching for related documents. Whereas this device marks a milestone in mobile UI development, its visual and computational interlocking of independent applications seems not to be context-driven, yet.

Relationship derivation based on current context such as time, location, or any other sensor data has been researched for several years now, paralleled by approaches to reflect this in the user interface [6].

5. SUMMARY AND OUTLOOK

In this paper, we addressed the problem of supporting seamless cross-application workflows in ubiquitous computing with adequate user interfaces. As a solution, we proposed *dynamic, context-driven fusion* of features from independent applications or service providers. Dynamic interface fusion integrates independent interface elements into one consistent user interface if the provided features are semantically related. The fused interface is rendered depending on the *physical device constraints*, which might change dynamically if displays are exchanged or applications migrate to other devices. As a first attempt to support dynamic interface fusion we presented a *layered architecture* that separates Interface Element Providers (e. g. applications, pervasive services), context providers, a Context-Aware Interface Decorator, and an Semantical Interface Layout Engine.

Our next step will be 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 disruptive patterns. This shift represents an important step towards seamless ubiquitous computing and defines therefore a central requirement of next generation computing research.

6. ACKNOWLEDGEMENTS

This reserach work is part of the Context Sensitive Intelligence (CSI) project under the direction of Armin B. Cremers and Holger Mügge. The project is financially and conceptionally supported by the Deutsche Telekom Laboratories (T-Labs, http://www.deutsche-telekom-laboratories.de).

7. REFERENCES

[1] Apple Inc. *Apple iPhone*, 2007. http://www.apple.com/iPhone.

[2] S. Bødker. When second wave HCI meets third wave challenges. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 1–8, New York, NY, USA, 2006. ACM Press.

[3] P. Bihler, L. Brunie, and V.-M. Scuturici. Modeling User Intention in Pervasive Service Environments. In *Proc. of the EUC'2005*, pages 977–986, Dec 2005.

[4] M. Bisignano, G. D. Modica, and O. Tomarchio. Dynamic User Interface Adaptation for Mobile Computing Devices. In *SAINT-W '05: Proceedings of the 2005 Symposium on Applications and the Internet Workshops (SAINT 2005 Workshops)*, pages 158–161, Washington, DC, USA, 2005. IEEE Computer Society.

[5] J. V. den Bergh and K. Coninx. Model-based design of context-sensitive interactive applications: a discussion of notations. In *TAMODIA '04: Proceedings of the 3rd annual conference on Task models and diagrams*, pages 43–50, New York, NY, USA, 2004. ACM Press.

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

[7] Experience Dynamics Corp. Science of Usability - User Interface Style Guides. online, April 2006. http://www.experiencedynamics.com/science_of_usability/ui_style_guides/.

[8] 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.

[9] 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 *Proceedings of the ACM AVI'2004 Workshop "Developing User Interfaces with XML: Advances on User Interface Description Languages"*, pages 55–62, May 2004.

[10] 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 Workshop für Selbstorganisierende, Adaptive, Kontextsensitive verteilte Systeme (SAKS)*, 2007. to be published.

[11] 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.

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

[13] 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.

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

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

[16] 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.

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

[18] X. Xiaoqin, X. Peng, L. Juanzi, and W. Kehong. A component model for designing dynamic GUI. In *Proc. of PDCAT'2003*, pages 136–140, Aug. 2003.