Interactive System Architecture for Layered Applications

Jonathan Diehl **Thorsten Karrer** **Jan Borchers**

RWTH Aachen University
Ahornstr. 55, 52074 Aachen, Germany
{diehl, karrer, borchers}@cs.rwth-aachen.de

AUTHORS

Jonathan Diehl is a 4th year PhD candidate of the HCI program at RWTH Aachen University. Jonathan's main research interest is in information-centric applications on mobile and ubiquitous systems. Throughout his work, he has identified many challenges caused by current system architectures, which led to his interest in rethinking the way software is developed today.

Thorsten Karrer is a 4th year PhD candidate of the same program. His work is focussed around interaction with time based media and navigating digital documents. His work on bringing rich interaction techniques for video navigation to the mobile led to his interest in nomadic and adaptive UIs.

Jan Borchers is a full professor and head of the Media Computing Group at RWTH Aachen University. His interests lie in post-desktop user interfaces and bridging the gap between the digital and the physical world. He supervises Jonathan's and Thorsten's PhD theses.

TRAVEL REQUIREMENTS

If the submission is accepted, one of the authors will require (at least partial) funding for travel and accommodation to be able to attend the workshop. Giving Jonathan and Thorsten the chance to participate in the workshop would be much appreciated even if no additional funding is available.

INTRODUCTION

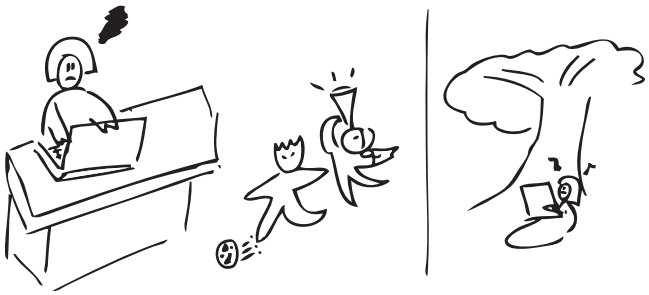
Today, users have access to a multitude of interactive systems, such as desktop computers, cell phones, and interactive whiteboards. However, all these systems are designed to be used mostly independently and are severely limited in their cross-system capabilities: applications are encapsulated by the system they run on and cannot easily transfer or extend their user interface (UI) or information space (IS) to other systems.

The challenge, here, is an architectural one: applications are designed with a fixed deployment system in mind, and there is no support by the system architecture to go beyond the boundaries of that system: (1) UIs cannot easily be adapted to the available interaction modalities, and (2) ISs cannot easily expand beyond system boundaries.

Adaptive User Interfaces

Users should be able to use the most appropriate interaction modalities available, at any given time, to operate their applications. Which input and output modality is appropriate can change during the interaction with the application and switching between them should be seamless for the user.

Christine usually works on her desktop computer at home, but sometimes her kids can be so noisy, she has to retreat with her laptop to the backyard. She does not mind the transition because the UIs of her desktop applications automatically migrate to her laptop, and she can immediately focus on what she had been doing before. Sometimes, she grabs her new tablet computer instead, because she likes the simplicity of its multitouch interface. Typing is not that great, but the simplified UI really helps her find the functions she needs fast.



This seamless adaption of the user interface has been envisioned, especially in the area of ubiquitous computing, a long time ago, and the technology to realize it has been around almost equally long. Nevertheless, no commercial systems available today come close to fulfilling this vision.

The main architectural challenges are that applications cannot easily define multiple UIs adapted to the available interaction modalities, and that applications cannot easily transfer or extend their UIs to other systems.

A notable exception are web applications that can be designed to support multiple interaction modalities. Many web sites provide adapted UIs for desktop and mobile clients. However, web applications are not always appropriate, because they require connectivity and are severely limited in their support of specialized modalities.

Comprehensive Information Spaces

Users should be able to access and manipulate all of their information independent of the system they operate.

Paul is talking to his business partner on the office phone about the latest development of their project, while taking notes on his desktop computer. After lunch, he meets Sandy in the kitchen and goes over his notes with her on his mobile phone. Together, they develop a new concept on the kitchen whiteboard. Later, at the project meeting, he discusses the new concept with the team while showing and annotating the sketch on the large interactive screen in the meeting room.

The technology to provide this comprehensive integration of information among all systems exists today, but is not available in such a seamless manner.

The architectural challenge is that the interactive system does not provide any support for accessing information or making information accessible on multiple systems, such that the IS can neatly span across these systems.

Many applications, especially on the mobile phone, have built-in synchronization of their IS with other systems or rely on remote servers to host the IS. However, these solutions are not consistent and often lead to fragmentation problems, because the user's information is spread across multiple ISs that are not compatible with each other.

CURRENT AND PROPOSED SOFTWARE ARCHITECTURE

Comparing the scenarios described above with current practices of how users interact with their information across applications and their applications across devices, we believe that current system architectures hinder the evolution of desirable 'device ecologies'.

Instead of developing a single application with multiple interfaces for different interaction modalities, much effort is put into re-inventing applications for each device. However, this approach does not allow seamless migration of the user's workflow between devices at runtime, or dynamic extension of an application's UI across multiple modalities. E.g., typing a text message on your cell phone with your laptop's keyboard cannot be easily realized with today's architecture.

Similarly, current practices to realize a comprehensive IS across applications and devices is limited by either tedious synchronization procedures or by manual information transfer from the various devices or remote servers. File-based information storage in the cloud is becoming a widely available option, but whether and how to the access them should be in the hands of the user through the interactive system, not in the hands of the application developer.

The widely accepted Model-View-Controller paradigm, splits the application structure into the UI (view), IS (model), and everything else (controller). This split has shown to be very effective in improving code re-usability and clarity, but it has failed to truly separate the UI or the IS from the application. We plan to extend this separation into a layered application architecture, where layers are completely separated through well-defined, yet flexible interfaces.

Layered Application Architecture

We propose to develop an interactive software architecture, where applications are split into the following layers:

1. **The Interface Layer** defines how the interface is laid out and which UI elements are used.
2. **The Interaction Layer** defines how the user interacts with the application through its UI.
3. **The Application Layer** defines the functional capabilities of the application and the application logic.
4. **The Information Layer** defines the user information model and information logic, such as data consistency and manipulation operations.

These layers communicate through a set of flexible interface languages, so each layer can be dynamically exchanged or moved to another system at runtime. Consequently, layers can be developed on any platform in any language, as long as they support the communication standards defined by the architecture. This way, an application running on a Microsoft Windows PC could migrate its interface to an Apple iPad.

Several aspects of this approach have been explored in the past, and we plan to use the published results to develop the proposed interactive system architecture and extend existing toolkits with the functionality to create and communicate between the four layers. We will then create a proof-of-concept prototype to demonstrate the flexibility of the new interactive system architecture.

EXPECTATIONS FOR THE WORKSHOP

At the workshop, we hope to explore challenges of developing the proposed interactive system architecture (how to define the layer abstraction and communication standards) and to discuss its implications on software development.

ACKNOWLEDGMENTS

We thank Chatchavan Wacharamanotham for the valuable discussions and last-minute sketches. This work was partially funded by the B-IT Foundation and the UMIC Excellence Cluster at RWTH Aachen University.