Proceedings of the IASTED International Conference
HUMAN-COMPUTER INTERACTION
November 14-16, 2005, Phoenix, AZ, USA

THE WALK-AWAY GUI: INTERFACE DISTRIBUTION TO MOBILE DEVICES

Micael Sjölund
Department of Computer and Information Science
Linköping University
Linköping, Sweden
email: micsj@ida.liu.se

Anders Larsson and Erik Berglund
Department of Computer and Information Science
Linköping University
Linköping, Sweden
email: andla@ida.liu.se and eribe@ida.liu.se

ABSTRACT
To walk-away from a desktop application with a small, specific portion of the GUI for data collection or data monitoring is a desirable method of enabling adaptation of the tools that we use to interact with computers and computer networks. It follows the principles of how users often use paper for mobile work. Although we have powerful PDAs and mobile phones for mobility, our current GUI frameworks do not enable such a behavior.

This paper presents the notion of *walk-away GUI* and discusses its successful implementation in current GUI frameworks. We also present a walk-away runtime framework, CHUCK, and two walk-away GUI examples. The paper shows that a mechanism for walk-away GUIs is possible but may also require significant changes to GUI frameworks for a more general and wide-spread implementation, in particular with regards to UI performance.

KEY WORDS
Ubiquitous computing, mobile interaction, remote application control, distributed user interfaces

1 Introduction

Context-aware computing promise the ability to focus a user interface to the situation at hand by understanding what it is the user aims to do and thereby simplifying interaction [1]. Furthermore, it holds the promises of enabling non-trivial mobile applications on mobile devices, such as PDAs and mobile phones, where interaction surfaces are highly limited, redefining the information system to be more suitable for mobility and mobile devices.

However, people have been doing this type of situated redefinition of their information tools for a long time. Do you remember scribbling down a few variables on a piece of paper, and then walk away to collect the data on your paper and later returning to the desktop and manually entering the data back into the computer. Sometimes we do this several times per day. This is a common form of mobile work using the type of desktop applications we have today, though not always in a very computerized form.

The desire to take a few variables out of the information system and into the world for data collection or for look-up, as part of a work task is commonplace – to *walk away* with a small subset of our information tools. This is unfortunately hard to accomplish using current GUI technology. Granted, we can write things down in documents, transfer these documents onto mobile devices and carry them around, and today, many applications are based on documents. But what about variables that are stored in a database somewhere that is not accessible by the user other than through GUIs? Furthermore, many of the documents we work on are much larger than the desired subset relevant to the current situation. What about choosing only a small subset of an existing document, not the entire 200 page document, and still being able to easily fill in the relevant information without having to go back and inserting them at the right place manually?

Is it possible to mimic how people work with paper onto the computerized devices of today and thereby provide a simple yet powerful way of enabling context-sensitivity or situational adaptation? We believe so. By constructing walk-away functionality in our GUI structures, it should become possible to, in the spur of the moment, grab a piece of a GUI and transfer that onto a mobile device such as a Smartphone or a PDA without loosing the connection back to the application across a network. In fact, we also believe that this way of computerizing walk-away behavior would be a general mechanism for improving robustness and reduce cognitive workload. Given the dynamic nature of humans and thereby also our tendency to get distracted, such computer support would be an improvement.

In this paper we present and discuss, situated decomposition and distributing of partial UI to mobile devices: *Walk-Away GUIs*. We describe a reference implementation enabling users to walk away with a subset of a MS Excel spreadsheet document and collect data. We also show how the system was implemented using an XML-based GUI description language, Views [2], and the .NET Compact Framework [3] as one possible platform.

2 Related Work

Today there are no general-purpose way of sharing a *partial* GUI from a desktop application to a hand-held device. Our project is focused on that very issue. Here we present some similar projects and contrast them to our work.

Bandelloni and Paternò [6] has similar motivations and thoughts, as well as a similar structure of their system, but they work with distributing web applications, with which the GUI system does not communicate all GUI events. Groulax present a toolkit for migrating part of a GUI to other devices in [7]. They have on-line communication of GUI events and feedback, but they do not highlight the need to distribute part of your current application in the spur of the moment. [8] does provide a mobile GUI generated from appliance functionality, with remote interaction capability. However, we want the user to choose what parts to distribute and try to keep a fluidity of interaction with the application.

The Pebbles project [9] has a similar system that is focused on how to control a PC from a handheld device. They have performed research on how to control PC applications from a palmtop computer. Our project differs from Pebbles in a few aspects. First, we want the user to spontaneously grab an area of choice from the desktop computer onto the handheld, which is illustrated with our spreadsheet example. Secondly, Pebbles seem to have focused on close-range usage of a PDA together with a PC to render more effective input to applications, whereas this project is focused on the situation when the user needs a part of the GUI in the hand when walking away from the desktop to perform a task.

GUI serialization into XML have been accomplished by Luyten et al. in [10], where they implemented transformation of Java AWT and Swing interfaces into XML using reflection and inspection methods.

Other systems have remote interaction implemented, but are mostly targeted to household appliances [8, 11, 12]. None of the projects are targeted toward remote interaction with application GUIs.

3 Walk-Away Example Scenario

Let us look more closely at an example scenario for the walk-away behavior of users.

A female nurse working in a hospital needs to collect some information about a patient's test results, and store them back into a Excel worksheet. She grabs a preprinted paper form and walks over to the labs and collects the desired information. She then *walks back* to workstation and types in the acquired information. As the information is input to the system the nurse realizes that some information about the patient is missing, she grabs a yellow-sticker note and walks over to the patient and writes down the information and then again *walks backs* to the workstation and stores the information in the application.

This small example illustrates how people work, although it presents a set of unnecessary and time-consuming problems such as walking back and forth to the work station and the double work of moving data from paper forms to information systems. These problems are difficult to solve with current information system working methods.

One solution for this problem is to distribute the specific spreadsheet cells to a hand-held device, such as a PDA or a smartphone, walk to the examination room and fill in the numbers on the device, which automatically updates the spreadsheet document. As the nurse returns to the workstation and notices that the patient information is missing he moves those spreadsheet cells to the hand-held devices and walks over to the patient to retrieve the desired data. The nurse will not need to go back to the workstation and insert the numbers sitting by the desk. She can rely on the system to take care of the document update. The problem of walking back and fourth between the rooms to double check and collect additional data is also removed. Inserting the information directly into her hand-held instead of walking back to the office lets her continue with other work where ever she is and also removes the risk of being distracted and forgetting to input the collected data. We believe that the cognitive load on the user is decreased when the nurse can perform computer tasks such as data input in the vicinity of the data source.

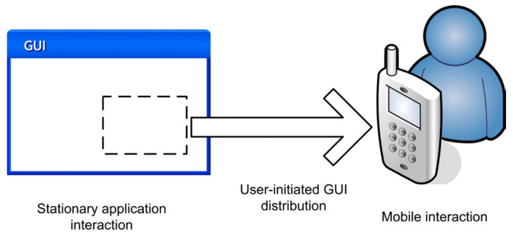

GUI
Stationary application interaction
User-initiated GUI distribution
Mobile interaction

Figure 1. Walk-away GUI principle.

As a general description for these kinds of tasks, of *casual* needs to distribute a part of an application interface to a handheld computer, we use the term *Walk-Away GUI*.

4 Walk-Away GUI

The screen–mouse–keyboard computer interaction method is a paradigm that has stuck to most working environments, although powerful mobile devices are present and capable of providing the interface to the user "anywhere, anytime", the slogan of pervasive computing. In our opinion, handheld devices are an unused resource for computer interaction in workplaces where stressful environments and complex work-flows is a fact. Information handling is often related to the activities surrounding the user and her context, so moving the UI to her hands is a step in the right direction.

A Walk-Away GUI, by other words, is an expression for flexibly grabbing part of a graphical interface and put it in your hand for continuing interaction. Being able to walk away from your desktop and collect data, bringing along the very pieces of an information system relevant to your current context. Just-in-time GUI decomposition and distribution and the subsequent handling of network connectivity during user interaction. The principle is illustrated in Fig. 1.

Walk-away GUIs can be said to be a subset of distributed user interfaces (DUIs) [4].

5 Method for Walk-Away

Today there are no general-purpose ways of sharing a GUI from a workstation application to a hand-held device, neither parts of it or the GUI as a whole. This is because of several different reasons:

- Workstation application interfaces are coded in different programming languages (C++, C#, Java etc.), using different GUI libraries to implement the presentation layer (MFC, Windows Forms, Swing).
- The GUI libraries most often used do not provide any means to extract information about the GUI at runtime.

So, how do we accomplish the task of distributing GUI components to another device and keeping the connection with the workstation application? The method we have used is to generate GUI description for the phone on the application-side, transfer it over network and rendering it on the hand-held at runtime. When the user interacts with the hand-held GUI, network messages between the distributed part of the GUI and the application are interchanged.

Our manner of using abstract description for GUI description differs from bitmap-level distribution of the screen. In our approach, we will not get real WYSIWYS (what-you-send-is-what-you-see), so it will give some interaction discontinuity for the user. But when distributing to for instance a smartphone, the input/output facilities on the device is far from as powerful as on the desktop computer, and a bitmap-level distribution would obstruct runtime adaptation of the GUI.

6 Reference Implementation

As examples of walk-away behaviour on an application-level, we have developed a framework, CHUCK, for on-the-fly migration of UI components to a hand-held device. The implementation consists of an client application running on a smartphone, and a server "daemon" running on a workstation. The server program is extensible to support different running applications, letting them distribute a GUI via an XML description to the hand-held. Fig. 2 shows the overall design of the walk-away system. A client daemon runs on the handheld and listens for incoming requests for displaying a GUI. A server daemon handles the connections between the workstation applications and the distributed user interfaces, as well as keeping track of devices that are connected.

As GUI description, we have chosen the XML-based user interface description language Views (version 2) [2]. The interface renderer has been ported from the .NET Framework to the .NET Compact Framework, and extended with distributed event and feedback messaging.

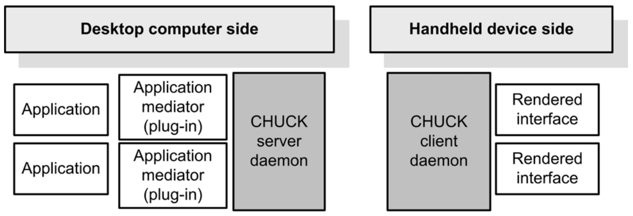


Figure 2. Overview of the CHUCK walk-away framework design

6.1 Client

The client, running on the Windows Mobile 2003-based smartphone, is a GUI renderer and a GUI runtime networking device. The program is running on top of the .NET Compact Framework, using the built-in GUI library, Windows Forms, with reflection to build GUI widgets.

The GUI description language, Views, is in its current implementation used as a front-end to the Windows Forms constructs, with little support for error checking in its current implementation. But it is powerful in the sense that it can use all widgets derived from `Windows.Forms.Control`.

We added support for runtime distributed events and feedback, as the user interacts with the application. The widget events are subscribed to from the workstation application as specified in the Views GUI description, and communicated over network. Updating a widget property is performed through sending a message containing the widget name, property name and property value.

6.2 Server

The server application is built as a TCP-server listening for devices on a specific port, together with a framework for implementing connections to workstation applications. These are implemented as plug-ins to the server, and are loaded and initialized at startup. The plug-ins are the link to applications that want to distribute their GUI to a hand-held device, for instance a spreadsheet walk-away plug-in as the connection between the server and MS Excel.

Requirements for the plug-ins are that they can describe the GUI to be distributed in Views XML format, and that they have the event handling methods corresponding to the widgets implemented. Each plug-in must also implement the `Chuck.API.IDUIApp` interface. The plug-in receives a `Chuck.API.IDUIDevice` object used to communicate with the hand-held through the methods `ShowGUI()` and `SendFeedback()`.

The networking is using TCP/IP through wire, Bluetooth or GPRS. The system assumes constant network connection in and a network roaming algorithm has not yet been implemented. This and network degradation handling is definitely an important feature of a ubiquitous walk-away system. Ping times for GPRS is currently insufficient for most GUI applications.

6.3 Walk-Away Plug-Ins

The CHUCK Plug-ins are loaded and initialized at server application startup. In this framework, a plug-in has a few responsibilities:

- The ability to create the GUI in Views XML format to send to the device. This is a part in this implementation that in the future can be interchanged with UI serialization into abstract formats. With an abstract GUI description of the application, the part of the UI to be distributed can be transformed to the specified walk-away device.
- Provide GUI event handling functionality in parity with names given in Views code. The plug-in specifies which events it wants to listen to, and provide names of its event handling methods.
- Handle the interaction with the desktop application (e.g. MS Excel in the spreadsheet case). The plug-in is thereby a mediator between the walk-away device and the application that lets the user utilize walk-away behaviour.

Additionally, the plug-in is notified through events from the server when a device connects to the server, i.e. when a device is present on the network.

Apart from the necessary implementation features, the plug-in can implement optional features.

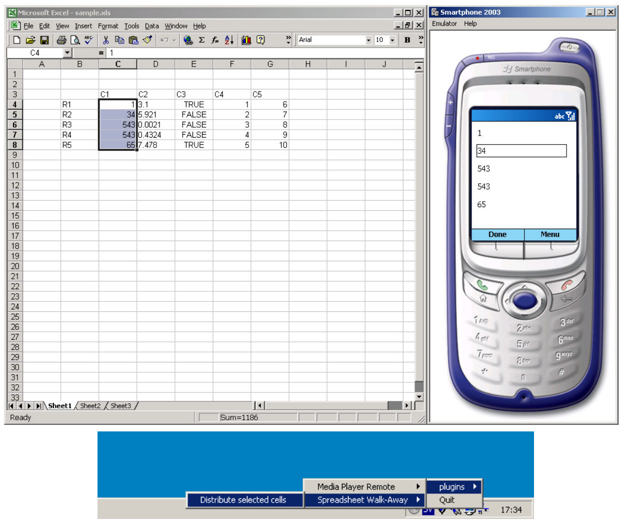
- It can create it's own GUI on the workstation that is constructed in the plug-in `Start()` method, as in the Movie player case, where a Windows Media Player ActiveX control is embedded in a Windows Forms application.
- It can provide a taskbar menuitem, which is loaded into the server application taskbar menu upon loading the plug-in.

As of now, there are implementations of two plug-ins to the CHUCK walk-away server.

6.3.1 Spreadsheet Walk-Away Plug-In

As a reference implementation, closely related to the scenario presented in the Introduction, an implementation of spreadsheet cell distribution to a hand-held device was implemented.

With the plug-in, the workstation user can select a number of cells in a Microsoft Excel spreadsheet with the mouse, and then by a click on the server application icon in the Windows taskbar, send the cells to be displayed on the smartphone, as shown in Fig. 3. The user is then able to modify the cell values, while wirelessly connected to the workstation. The plug-in interacts with the open Excel application via interop.



	A	B	C	D	E	F	G
3			C1	C2	C3	C4	C5
4		R1	1	3.1	TRUE	1	6
5		R2	34	5.921	FALSE	2	7
6		R3	543	0.0021	FALSE	3	8
7		R4	543	0.4324	FALSE	4	9
8		R5	65	7.478	TRUE	5	10

Figure 3. These pictures show how the cells from MS Excel are distributed via the server application to a smartphone running the walk-away client.

6.3.2 Media Player Remote Control Plug-In

The Views-based media player remote control, as shown in Fig. 4, is a plug-in with a media player GUI embedded that can send its movie controls as widgets to the handheld device. More specifically, when a device is present, it hides its movie controls and sends the relevant widgets to the handheld. This lets the user utilize the phone as a remote control when for instance watching a movie.

The first implementation of the Views media player remote was based on an older version of Views that contained an UI renderer that did not use reflection in the same way as the Views 2.0 system does. Details for that implementation is provided in [5].

7 Discussion

In this paper we have shown how to capture GUI session states to distribute parts of the GUI to a handheld device and walk away with this very specific GUI as part of adapting UIs to situations. We have shown that it is possible to create a general mechanism for such Walk-Away GUI and implemented a framework with XML-based GUI distribution. So far we support GUI distribution using the Views XML-language for GUI descriptions on Smartphone devices. In the future it is likely that XML-languages for GUI components will be commonplace. Mozilla's interface language XUL [13], W3's UIML [14] and the UI markup language of the Longhorn operating system, XAML [15] are examples of XML UI description languages leading this trend.

With experiences from the development of the walk-away framework with the media player plug-in and the Excel plug-in, we have also discovered important issues con-

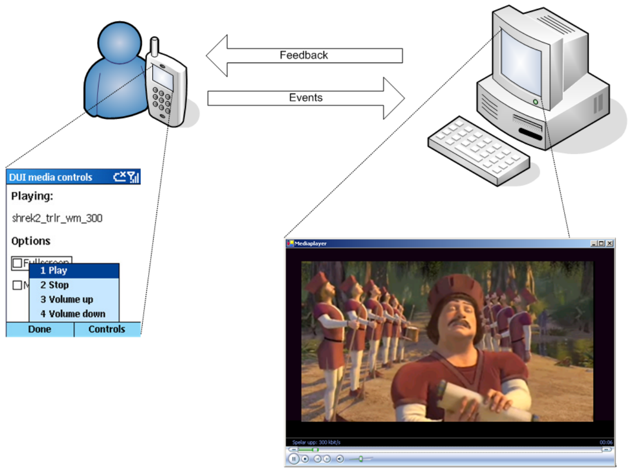


Figure 4. Screenshots of the movie player remote control and illustration of network communication.

cerning further development of general walk-away GUIs.

UI Performance: Distributing partial GUIs from on computer onto another mobile device currently introduces fair amounts of network communication for low-level UI handing. The network cannot always provide the responsitivity required by a GUI to get a good user experience, especially when the connection is wireless. In networks where connectivity may fluctuate this can also cause unwanted UI behavior, such as event messages being queued and then appear in a blur or even be lost over the network. The solution would be to move more of the code over to the handheld device and thus reduce, in particular, the amount of low-level communication needed. However, the age old Model–View–Control design pattern [16] that most GUI models implements, complicates the issue. Since appearance and logic is separated in many applications it is a non-trivial task to automatically determine which code to move over to the handheld device. A more thorough redesign of the underlying frameworks is required and to some extent old applications may not be able to support good UI performance because of this earlier design for source-code modularity.

GUI design control: In our framework, it is possible to take a part of a GUI and distribute it to a handheld device. However, for both usability and appearance reasons, developers may want to limit which parts of a GUI is distributable, which may not be separated, to provide a frame with a company logo and so fort. All in all, developers are likely to want to control, how the distribution of GUIs can take place. Needed are new concepts for layout engines, enabling developers to describe the distribution of various components. Furthermore, visualization of these many different possible designs on handheld devices may be required for testing. In this sense, walk-away behavior introduces requirements on development environments.

8 Conclusion

From our work with walk-away GUIs we conclude that it is indeed possible to create this type of flexible deconstruction and just-in-time distribution of GUIs as a method of providing situational adaptation for users of software applications. In fact, it is possible to create general methods that work with programs without developing dedicated distributed systems. Users may in a given situation walk away with a very specific part of a GUI as a means of acquiring context-aware behavior from software applications.

However, our work has also uncovered some underlying changes needed in the GUI models and tools. Clearly, a more general model of distributing GUIs that handled these shortcomings needs to be developed to enable the more general use of this desirable UI feature.

Acknowledgements

This work has been possible due to support from Microsoft Research and Santa Anna IT Research Institute

References

[1] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, US, 1994.

[2] Judith Bishop and Nigel Horspool. Developing principles of GUI programming using Views. In *Proceedings of the 35th SIGCSE technical symposium on Computer science education*, pages 373–377. ACM Press, 2004.

[3] Microsoft .NET Compact Framework. `http://msdn.microsoft.com/mobility/netcf/default.aspx`.

[4] Erik Berglund and Magnus Bång. Requirements for distributed user-interfaces in ubiquitous computing networks. In *Proceedings of MUM2002*, Oulo, Finland, 2002.

[5] Micael Sjölund, Anders Larsson, and Erik Berglund. Smartphone views: Building multi-device distributed user interfaces. In *Proceedings of Mobile HCI '04*, 2004.

[6] Renata Bandelloni and Fabio Paternò. Flexible interface migration. In *Proceedings of the 9th international conference on Intelligent user interface*, pages 148–155. ACM Press, 2004.

[7] Donatien Groulax, Peter Van Roy, and Jean Vanderdonckt. Migratable user interfaces: Beyond migratory interfaces. In *Proceedings of Mobiquitous 2004*, 2004.

[8] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating remote control interfaces for complex appliances. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 161–170. ACM Press, 2002.

[9] Brad A. Myers. Using handhelds and pcs together. *Commun. ACM*, 44(11):34–41, 2001.

[10] Kris Luyten and Karin Coninx. An XML-based runtime user interface description language for mobile computing devices. In *Proceedings of the 8th International Workshop on Interactive Systems: Design, Specification, and Verification-Revised Papers*, pages 1–15. Springer-Verlag, 2001.

[11] Fridtjof Feldbusch, Alexander Paar, Manuel Odendahl, and Ivan Ivanov. The btrc bluetooth remote control system. *Personal Ubiquitous Comput.*, 7(2):102–112, 2003.

[12] Alexandre Sanguinetti, Hirohide Haga, Aya Funakoshi, Atsushi Yoshida, and Chiho Matsumoto. FReCon: a fluid remote controller for a freely connected world in a ubiquitous environment. *Personal Ubiquitous Comput.*, 7(3-4):163–168, 2003.

[13] XML user interface language (XUL). `http://www.mozilla.org/projects/xul/`.

[14] UIML: An appliance-independent xml user interface language. `http://www8.org/w8-papers/5b-hypertext-media/uiml/uiml.html`.

[15] Longhorn markup language (code-named XAML) overview. `http://longhorn.msdn.microsoft.com/lhsdk/core/overviews/about%20xaml.as%px`.

[16] Glenn E. Krasner and Stephen T. Pope. A cookbook for using the model-view controller user interface paradigm in smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.