Adding Flexible Input Device Support to a Web Browser with MundoMonkey

Daniel Schreiber
TU Darmstadt
Darmstadt, Germany
schreibertk.informatik.tu-darmstadt.de

Abstract

Computer applications are increasingly used in non-desktop settings, e.g. at a public kiosk systems or on mobile phones. Thanks to the widespread availability of web browsers for different platforms, web interfaces are often employed in these settings. However, current browsers lack sufficient support for flexibly adapting to non-desktop settings, e.g. ad-hoc changes of input devices. A use case for this is, e.g., an interactive shopping window that presents a web interface for buying products. If the browser in the shopping window supported ad-hoc changes of input devices, the customer could dynamically attach carried input devices, e.g., a mobile phone, to the browser and interact with it. In this paper, we present a solution to the problem of dynamically connecting input devices in a non-desktop setting to a browser, based on the MundoMonkey Firefox extension for interactive spaces. Using our approach, the unmodified web user interface can be used with arbitrary input devices in ways that cannot be realized by synthesizing mouse and keyboard events. In our approach, the customization to the device at hand is performed transparently to the application developer by the end-user.

1 Introduction

Web interfaces are commonly used to make applications accessible for a wide range of usage situations. Web interfaces are extremely portable, e.g., they can be used on mobile phones, on public kiosk systems and on the office desktop. Still, there are many more situations, in which web interfaces cannot be used efficiently, because no browser exists that supports the situations characteristics with respect to in- and output devices.

If one such situation is important enough, e.g., because it is directly related to business value, a customized solution can be implemented for a controlled environment like a warehouse or a hospital following the design approach of [Klug and Mühlhäuser, 2007]. In more open environments, where a heterogeneous set of users interacts with different input devices in a way depending on their preferences and the situation, such an approach is not feasible. For example, in a shopping mall users may want to interact with an interactive shopping window using their personal devices for input, as sketched in Figure 1. The user's mobile phone can act as an input device, e.g., by using voice recognition via the built-in microphone or gesture recognition via the phone's camera or remotely controlling a mouse cursor via the phone's touchscreen. Thereby, the choice which modality is used for interaction depends on the situation, e.g., on the noise level and the user, e.g., her motor skill level.

In this paper, we argue that, ideally

A the browser running the web interface, e.g., the one situated in an interactive shopping window, is able to automatically connect to and receives input from devices carried by the user, and

B the user (or at least the shop assistant) can easily install support for novel interaction techniques and devices.

Reaching both subgoals should not require assistance from the developer of the web application nor assistance from the developer of the browser. We assume that in the envisioned open environments an existing interface will be used much more frequently in a novel situation or with a novel device than a new application for existing devices and situations will be developed. This is in contrast to desktop computing where an application would always be accessed with the same device setup. However, for web applications this is already true to some degree, as they may be accessed with different and novel browsers. Therefore, we put an emphasis on web user interfaces in our research, as these also bring many advantages, like easy deployment, and already address a wide range of usage situations as stated above. In this paper, we present a novel approach for connecting input devices to a web browser in an open environment, building on the MundoCore middleware [Aitenbichler *et al.*, 2007]. We integrated access to the middleware into an end-user scripting framework for web interfaces with the MundoMonkey extension for Firefox. Thereby,

- i) the actual interaction device used to carry out an interaction technique is determined automatically at runtime and
- ii) the set of supported interaction techniques can be determined by an end-user with minimal effort.

2 MundoCore Middleware for Interactive Spaces

To reach subgoal A, from the introduction, the different devices in an open environment, e.g., in front of a window in a shopping mall, need to be able to communicate with each other without configuration. A suitable approach to solve this problem, is to require that all devices use a common communication middleware like [Vanderhulst *et al.*, 2007] or [Johanson and Fox, 2002]. To this end, we

Figure 1: Users in a shopping mall interact with interactive shopping windows using different input devices.

employ the MundoCore middleware [Aitenbichler *et al.*, 2007]. MundoCore is a Pub/Sub middleware with language bindings for Java, Objective-C, C and C++. To connect a device to the MundoCore middleware a proxy service is needed that receives input events from the device hardware and publishes it as MundoCore events. Proxy services for many input devices are readily available, e.g., a voice recognition device [Aitenbichler *et al.*, 2004] or the Wii Remote. The ensemble of a hardware device and the corresponding MundoCore service publishing events will be called an *interaction resource* in this paper.

The ensemble of all interaction resources together with additional context sensors providing information build up the *interactive space* surrounding the browser. Thereby, we consider the interactive space as a generalization of desktop and mobile phone environments, for which current browsers are designed.

Several techniques can be used on the side of the browser to decide which of the available interaction resources should currently be used for interaction. One possibility is to decide based on location information, e.g., to accept input from devices belonging to a nearby user, or a user looking at the browser's screen, as proposed in [Braun *et al.*, 2004]. Another option, giving more control to the user, is a meta-user interface allowing the user to associate with the browser and configure the interaction means [Vanderhulst *et al.*, 2009; Schreiber and Hartmann, 2008]. Thus, the first subgoal A is reached by combining a communication middleware with context awareness and/or a suitable meta-user interface.

However, designing a browser for interactive spaces requires much more flexibility in supporting different input techniques, as the devices in the interactive space may change in unforeseen ways, e.g. requiring support for a so far unknown voice input device.

3 MundoMonkey

To reach the second subgoal B from the introduction, we designed the MundoMonkey Firefox extension. MundoMonkey [Schreiber *et al.*, 2009][1]. It takes the flexibility and rich output adaptation capabilities of end-user scripting for web interfaces and augments them with support for handling input from the interactive space surrounding the browser. Handling of input and output modifications are performed by MundoMonkey *interaction strategies*. MundoMonkey *interaction strategies* are JavaScript files that are executed in the context of the web application in the browser. This allows them to easily modify the output by operating on the DOM of the web page in the browser.

MundoMonkey is built on top of Greasemonkey[2] and Greasemonkey scripts can serve as basis for MundoMonkey *interaction strategies*. As one important improvement compared to Greasemonkey, MundoMonkey allows *interaction strategies* to connect to the MundoCore [Aitenbichler *et al.*, 2007] middleware for interactive spaces and thereby react to events from interaction resources and sensors in the interactive space. For example, *interaction strategies* can attach an event listener to a speech recognition service to react on user utterances.

The end-user can install a new *interaction strategy* like any Greasemonkey script, by pointing the browser to a URL. The set of available *interaction strategies* can thus be easily installed by the end-user, e.g., the shop assistant in the mall. The widespread use of Firefox extensions and Greasemonkey scripts shows in our opinion that installing *interaction strategies* is possible for the end-user without any assistance from developers.

At runtime, MundoMonkey automatically selects the appropriate *interaction strategies* for the devices at hand. MundoMonkey does so, by matching the device type to the required input for all *interaction strategies* and then selecting the ones that are best suited to handle the input from the devices. This setup can then be overridden by the user with the means of a meta-user interface.

Our approach requires that every *interaction strategy* is implemented in a generic way, i.e. able to work with any web user interface. Otherwise, installation of web user interface specific strategies would be needed, which would impose severe scalability problems considering the huge number of existing web user interfaces. Still, it is possible to support a wide range of input devices and interaction techniques by using MundoMonkey strategies. In

[1] https://leda.tk.informatik.tu-darmstadt.de/cgi-bin/twiki/view/Mundo/MCFirefox

[2] http://www.greasepot.net

the next section, we present one example of such an interaction strategy. Especially interaction techniques that cannot be realized by synthesizing mouse and keyboard events, e.g., interaction techniques that require the combination of output modification and input handling, benefit from MundoMonkeys output modification capabilities.

4 Voice Interaction Strategy

As a use case for MundoCore and MundoMonkey, we present an interaction strategy for voice. Thereby, the voice input device connected to the Firefox browser is the Talking Assistant [Aitenbichler *et al.*, 2004]. The Talking Assistant is a small device that is always carried by the user. The idea is, that the user in a mall can use the voice recognition capabilities of the Talking Assistant with a browser in an interactive shopping window. If somebody wearing a Talking Assistant approaches the shopping window, the Talking Assistant is connected to the browser and can control the web page loaded in the browser via voice commands.

The problem solved by the voice interaction strategy is to provide a recognition grammar for the web page at hand to the Talking Assistant. This has to be done without any specific knowledge about the web application, as otherwise a new strategy for every interface would be required. Thanks, to the access to the DOM of the web interface in the browser, the strategy can be implemented in a web interface agnostic way.

The recognition grammar provides phrases for controlling every interactive element of the web page, extracted from the DOM tree. Thereby, the content of the element is padded with fixed phrases for making the interaction more natural. To access a link on the page, the user can e.g. say "click on <link text>, please". For interactive form fields the fieldname is not so easy to extract. We used existing algorithms for determining the labels of interactive elements on the web page, [Leshed *et al.*, 2008; Hartmann *et al.*, 2008]. Once the page is loaded and processed, the resulting grammar is sent to the Talking Assistant via the MundoMonkey Extension. Figure 3 shows an example for a rule generated for a link with the label "character", more complicated rules may also involve variables, i.e. inline dictation, whose values are passed back to the strategy as parameters. The grammar is specified in the MS SAPI5.3 format[3].

Once the grammar was sent to the Talking Assistant, the reactive part of the strategy handles all recognized utterance. To do so, it stores a specific callback function for every grammar rule. The callback functions are generated while processing the page and creating the grammar. For example, matching of the rule in figure 3 results in following the "character" link in the web page. Our voice interaction strategy supports different actions for the different HTML form element types and HTML links.

5 User Study

To evaluate whether our voice interaction strategy provided usability which is comparable to other state of the art voice user interface techniques that do not benefit from the flexibility of MundoCore and MundoMonkey, we compared it against the built-in voice control of the Internet Explorer in Windows Vista.

[3]http://msdn.microsoft.com/en-us/library/ms723632(VS.85).aspx

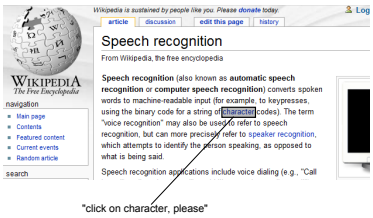

Figure 2: Example page used in the user study. The character link could be activated by saying "Click on character, please".

```
<rule id="command4">
  <O><O>please</O>
    <L>
      <P>select</P>
      <P>click<O>on</O></P>
    </L>
  </O>
  <P>character</P><O>please</O>
</rule>
```

Figure 3: One rule of the grammar generated for the example page.

The study was conducted using a within subject design. Participants were members of our department and students ($n = 10$). Every participant completed a task with the Firefox browser, augmented with our voice input strategy extension and the Vista voice control for Internet Explorer. Thereby the Talking Assistant also used the Vista Speech Recognizer, so we reduced the difference to just the mapping of speech recognition results to actions in the web page. This procedure allowed us to test our interaction strategy against the built-in Internet Explorer strategy. See Figures 5 and 4 for an overview on the setup in both conditions. Although we tested only one web user interface in the user study, the strategy works with other websites, e.g., the one of E-Bay or of online travel-agencies. Thereby the example shows, that a strategy can be efficiently implemented without tailoring it to a specific website.

The task performed by the participants was to gather information from Wikipedia articles, which could be easily replaced by the contents of a shopping catalog in the mall scenario. Participants had to scroll down twice (they were told to "Find the information on the bottom of the page"). Then they were instructed to select a certain link ("follow the third link in the list") and then select a link of their choice ("follow any link on this page that interests you"). After the task, participants filled out the SUS usability questionnaire [Brooke, 1996]. The order of conditions was counterbalanced to control for learning effects. We found the ratings of our strategy ($M = 62.5, SD = 17.16$) were significantly better than the rating for the Vista Voice Interaction ($M = 51.5, SD = 18.33$) using a dependent samples t-test ($t(9) = 2.45, p < .05$), see figure 6. The effect size was medium to large with $Cohens'd = 6.4$.

The goal of the study was not to prove superior usability of the voice interaction strategy but more whether voice interaction could be implemented within MundoMonkey with comparable usability to commercial systems. Therefore we did not use a larger sample size or obtain detailed quantitative data. The reason why the voice strategy of MundoMonkey achieved a higher SUS score is probably the higher recognition rate of the Vista Speech Recognizer

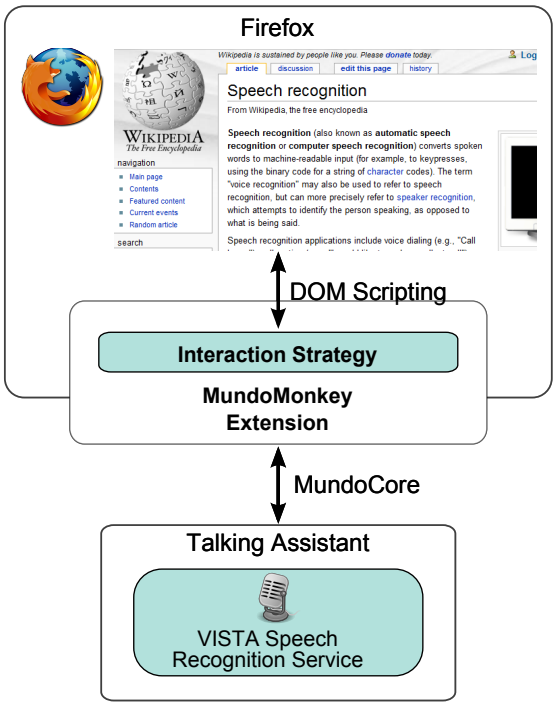

Figure 4: Setup in the voice interaction strategy condition used in the study

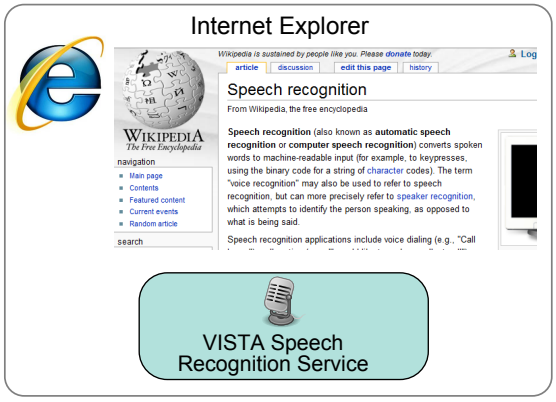

Figure 5: Setup for the builtin Internet Explorer strategy condition used in the study

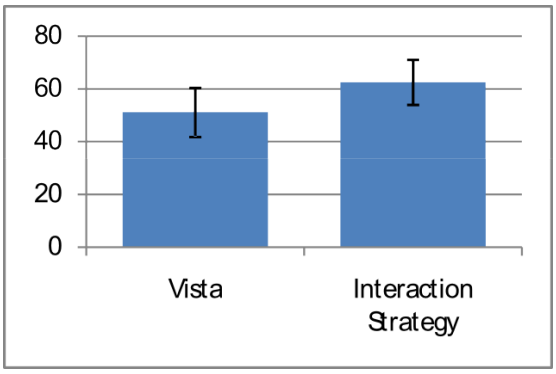

Figure 6: Our voice interaction strategy achieved a signaficantly higher SUS usability score compared to the builtin Internet Explorer strategy.

when using a small recognition grammar for the website at hand (as in the voice interaction strategy) compared to the dictation grammar used in the Vista condition.

Implementing voice interaction as *interaction strategy* within MundoMokey has several advantages over embedding it in the operating system or the browser, as e.g. done in Windows Vista. The recognition is done on the Talking Assistant, which exclusively is used by a single user. This allows the voice recognition engine of the Talking Assistant to be highly customized for a single speaker, greatly improving recognition performance. Further, the flexibility of MundoCore allows different Talking Assistants to dynamically associate with the browser and interact with the web application, as required in a mall which is populated by many users.

6 Related Work

User interfaces that adapt to the context of use, i.e. the interaction devices, the user experience ant the user's other tasks [Calvary *et al.*, 2005] without having to change their implementation through programming are a pertinent problem in research. In this section approaches to solving the problem from various fields will be analyzed according to their strength and weaknesses.

W3C Ubiquitous Web Applications Activity The problem of widening access to web applications to other devices and modalities is targeted by the W3C Ubiquitous Web Applications Activity which *focuses on mechanisms to reduce the cost for developing and delivering applications to a wide range of devices, including the means to adapt to user preferences and environmental conditions*. This goal is pursued by proposing extensions to the HTML standard, e.g. X+V. With such an approach, legacy applications, e.g. an existing travel booking web application, will have to be rewritten to benefit from these extensions. Supporting emerging input techniques will require an additional rewrite of the application. An alternative would be to map the new input techniques to fit an existing application in the browser, as it is proposed in this paper.

Model Driven UI Development Model driven UI development as e.g., proposed in [Berti *et al.*, 2004] allows the application developer to specify the UI in a device- or modality independent user interface description language (UIDL, see [Souchon and Vanderdonckt, 2003] for an overview of existing UIDLs) at an abstract level. This description is then automatically adapted to the device at hand. This approach gives the programmer very powerful tools to specify user interfaces. As noticed in [Gilroy and Harrison, 2005], it forces end-users and device vendors to integrating new devices or interaction techniques into the model transformation engine before they can be used, which is rather complex. As an alternative, we are focussing on allowing the end-user to customize the interaction with a web application in an ad-hoc easy way, thereby not giving the programmer additional support for specifying the interface.

Adaptive Toolkits Adaptive toolkits like SUPPLE++ [Gajos *et al.*, 2007] layout GUIs with respect to the physical abilities of the user or the requirements of the situation (e.g. to support users wearing gloves). SUPPLE++ relies on a proven mathematical model, which is able to automate

the layout task using recorded user traces and an interaction cost model describing the situation. Implementing interaction techniques, like voice interaction or techniques which require modification of the UI as a whole instead of at widget level are more difficult to implement with such an approach. Also, the focus is not on supporting dynamically changing input devices at runtime, which would require to connect the adaptive toolkit to a suitable communication middleware.

Assistive Computing The above mentioned approaches provide advanced tools to the developer, which require a rewrite of legacy applications to benefit the end-user. Approaches from the area of assistive computing like [Wang and Mankoff, 2003; Carter *et al.*, 2006] do not require a change of the application to support a specific usage context. Instead they mimic the expected environment to the application and map it to the actual environment, e.g. by controlling a mouse cursor with a one-switch device. However, they do not support highly dynamic interactive spaces as e.g. encountered in the mall scenario, where one application is used by different users with different devices.

Ubiquitous Computing Approaches from the area of ubiquitous computing that explicitly target dynamically changing devices like [Dragicevic and Fekete, 2004b; Serrano *et al.*, 2008; Ballagas *et al.*, 2003; Dragicevic and Fekete, 2004a] do not work together with HTML interfaces, thereby losing the advantages web applications already provide, like easy deployment. Although one could theoretically implement a web browser which makes use of these approaches, supporting some interaction techniques would still be difficult. For example, the mentioned tools do not support output modification sufficiently. For recognition based input, like speech recognition devices this is a drawback as these are best handled by providing disambiguation options to the user, e.g., in the form of a list to choose from [Mankoff *et al.*, 2000]. In our approach output adaptation and presenting disambiguation lists is easily possible. The results from [Hartmann and Schreiber, 2009] show, that displaying suggestions from uncertain contextual data sources allows to increase usability of applications.

End-user scripting Altering web pages in the browser is supported by end user programming tools like [Bolin *et al.*, 2005; Little and Miller, 2006; Bigham and Ladner, 2007]. This approach allows the end user to tailor interaction with web pages to her specific situation without requiring to rewrite the application. However, these programming environments do not support communication with external input devices at the client side. This drawback is remedied by MundoMonkey, which thus enables to use the very successful end-user scripting approach for web applications to target the adaptation to input devices as well.

7 Conclusions

Web interfaces are widely used to provide access to applications in many non-desktop usage situations. However, current browsers are too inflexible to support the different situations we encounter in open environments, like a shopping mall. To remedy this, we presented a solution relying on i) MundoCore for dynamically connecting input devices to the browser at runtime, and ii) MundoMonkey for letting the end-user add support for new types of input devices in the form of interaction strategies without requiring changes to existing applications.

As an example we presented an implementation of voice input as interaction strategy, which has significantly better usability than the built-in voice control for Internet Explorer. Additionally, it can easily connect to different input devices, e.g., in a mall environment. Compared to other approaches, MundoMonkey allows the implementation of *interaction strategies* that handle the input to applications and additionally are able to modify the output of the application. For example, this makes it possible to combine voice input with suggestions from contextual data, as implemented in [Hartmann and Schreiber, 2009].

Currently, the interaction strategies need to be installed manually, however, they could also be downloaded automatically from the device of the end-user, which would further simplify the process.

References

[Aitenbichler *et al.*, 2004] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. Talking Assistant: A Smart Digital Identity for Ubiquitous Computing. In *Advances in Pervasive Computing*, pages 279–284. Austrian Computer Society (OCG), Austrian Computer Society (OCG), 2004.

[Aitenbichler *et al.*, 2007] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. MundoCore: A Lightweight Infrastructure for Pervasive Computing. *Pervasive and Mobile Computing*, 2007.

[Ballagas *et al.*, 2003] Rafael Ballagas, Meredith Ringel, Maureen Stone, and Jan Borchers. istuff: a physical user interface toolkit for ubiquitous computing environments. In *Proceedings of CHI*, 2003.

[Berti *et al.*, 2004] Silvia Berti, Francesco Correani, Giulio Mori, Fabio Paternò, and Carmen Santoro. Teresa: a transformation-based environment for designing and developing multi-device interfaces. In *Proceedings of CHI*, 2004.

[Bigham and Ladner, 2007] Jeffrey P. Bigham and Richard E. Ladner. Accessmonkey: a collaborative scripting framework for web users and developers. In *Proceedings of the international cross-disciplinary conference on Web accessibility (W4A)*, 2007.

[Bolin *et al.*, 2005] Michael Bolin, Matthew Webber, Philip Rha, Tom Wilson, and Robert C. Miller. Automation and customization of rendered web pages. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 163–172, Seattle, WA, USA, 2005. ACM.

[Braun *et al.*, 2004] Elmar Braun, Gerhard Austaller, Jussi Kangasharju, and Max Mühlhäuser. Accessing web applications with multiple context-aware devices. In Maristella Matera and Sara Comai, editors, *Engineering Advanced Web Applications*, pages 353–366. Rinton Press, December 2004.

[Brooke, 1996] J. Brooke. Sus-a quick and dirty usability scale. *Usability Evaluation in Industry*, pages 189–194, 1996.

[Calvary *et al.*, 2005] Galle Calvary, Jolle Coutaz, Olfa Dassi, Lionel Balme, and Alexandre Demeure. *Towards a New Generation of Widgets for Supporting Software Plasticity: The Comet*, pages 306–324. 2005.

[Carter *et al.*, 2006] Scott Carter, Amy Hurst, Jennifer Mankoff, and Jack Li. Dynamically adapting guis to diverse input devices. In *Proceedings of Assets*, 2006.

[Dragicevic and Fekete, 2004a] Pierre Dragicevic and Jean-Daniel Fekete. The input configurator toolkit: towards high input adaptability in interactive applications. In *Proceedings of AVI*, 2004.

[Dragicevic and Fekete, 2004b] Pierre Dragicevic and Jean-Daniel Fekete. Support for input adaptability in the icon toolkit. In *Proceedings of ICMI*, 2004.

[Gajos *et al.*, 2007] Krzysztof Z. Gajos, Jacob O. Wobbrock, and Daniel S. Weld. Automatically generating user interfaces adapted to users' motor and vision capabilities. In *Proceedings of UIST*, 2007.

[Gilroy and Harrison, 2005] Stephen W. Gilroy and Michael D. Harrison. Using interaction style to match the ubiquitous user interface to the Device-to-Hand. In *Engineering Human Computer Interaction and Interactive Systems*, pages 325–345, 2005.

[Hartmann and Schreiber, 2009] Melanie Hartmann and Daniel Schreiber. Augur: Providing context-aware interaction support. In *Engineering Interactive Computing Systems (EICS'09)*, Carnegie Mellon University, Pittsburgh, USA, 2009.

[Hartmann *et al.*, 2008] Melanie Hartmann, Torsten Zesch, Max Mühlhäuser, and Iryna Gurevych. Using similarity measures for context-aware user interfaces. In *Proceedings of 2nd International Conference on Semantic Computing*, page (to appear). IEEE, IEEE, August 2008.

[Johanson and Fox, 2002] Brad Johanson and Armando Fox. The event heap: A coordination infrastructure for interactive workspaces. In *Proceedings of the Fourth IEEE Workshop on Mobile Computing Systems and Applications*, 2002.

[Klug and Mühlhäuser, 2007] Tobias Klug and Max Mühlhäuser. Modeling human interaction resources to support the design of wearable multimodal systems. In *ICMI '07: Proceedings of the ninth international conference on Multimodal interfaces*, pages 299–306, New York, NY, USA, 2007. ACM.

[Leshed *et al.*, 2008] Gilly Leshed, Eben M. Haber, Tara Matthews, and Tessa Lau. Coscripter: automating & sharing how-to knowledge in the enterprise. In *Proceeding of CHI*, pages 1719–1728, 2008.

[Little and Miller, 2006] Greg Little and Robert C. Miller. Translating keyword commands into executable code. In *Proceedings of UIST*, pages 135–144, Montreux, Switzerland, 2006. ACM.

[Mankoff *et al.*, 2000] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of CHI*, 2000.

[Schreiber and Hartmann, 2008] Daniel Schreiber and Melanie Hartmann. Association: Unobtrusively creating digital contracts with smart products. In *Proceedings of Smart Products: Building Blocks of Ambient Intelligence (AmI-Blocks'08)*, 2008.

[Schreiber *et al.*, 2009] Daniel Schreiber, Melanie Hartmann, and Max Mühlhäuser. Mundomonkey: Customizing interaction with web applications in interactive spaces. In *Engineering Interactive Computing Systems (EICS'09)*, Carnegie Mellon University, Pittsburgh, USA, 2009.

[Serrano *et al.*, 2008] Marcos Serrano, Laurence Nigay, Jean-Yves L. Lawson, Andrew Ramsay, Roderick Murray-Smith, and Sebastian Denef. The openinterface framework: a tool for multimodal interaction. In *Proceedings of CHI*, 2008.

[Souchon and Vanderdonckt, 2003] Nathalie Souchon and Jean Vanderdonckt. A Review of XML-compliant User Interface Description Languages. In *Proceedings of International Workshop Design, Specification, and Verification of Interactive Systems*, 2003.

[Vanderhulst *et al.*, 2007] Geert Vanderhulst, Kris Luyten, and Karin Coninx. Middleware for ubiquitous service-oriented spaces on the web. In *AINAW '07: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, pages 1001–1006, Washington, DC, USA, 2007. IEEE Computer Society.

[Vanderhulst *et al.*, 2009] Geert Vanderhulst, Daniel Schreiber, Kris Luyten, Max Mühlhäuser, and Karin Coninx. Edit, inspect and connect your surroundings: A reference framework for meta-uis. In *Engineering Interactive Computing Systems (EICS'09)*, Carnegie Mellon University, Pittsburgh, USA, 2009.

[Wang and Mankoff, 2003] Jingtao Wang and Jennifer Mankoff. Theoretical and architectural support for input device adaptation. In *Proceedings of CUU*, 2003.