A Single-Authoring Technique for Building Device-Independent Presentations

Candy Wong, Hao-hua Chu, Masaji Katagiri
DoCoMo Communications Laboratories USA, Inc.
Seamless Experience Environment Laboratory
{wong, haochu, katagiri}@docomolabs-usa.com

Abstract

One of the major challenges in the prevailing web page development is *device heterogeneity*. Currently, devices such as cell phones, PDAs, and PCs are already capable of providing access to the Internet. In the near future, we are expecting other devices such as car navigation systems to have the same capabilities. As the number of these devices increases, the traditional approach of developing a separate web page for each device results in too many redundant efforts. It would take too much effort for authors to learn different device-specific markup languages and tools, and then to implement and maintain a large number of device-specific web pages of the same web application. To meet this challenge, we propose a single-authoring technique called *ScalableWeb*. It allows authors to build a *device-independent presentation model* at design time. The device-independent presentation model can be used to generate device-specific presentation at runtime, with the possibilities of customization from authors.

Introduction

Nowadays, we see numerous devices having sufficient computing, networking and display capabilities to run a web browser. However, due to device heterogeneity such as screen sizes and input methods, not all devices can access to the same web page and not all accessible web pages are fully functional on all devices. One of the feasible solutions is to provide a single-authoring technique, with which authors can build a device-independent presentation model that can be used to generate device-specific presentation.

Since building a device-independent web page that can be accessible from any kinds of devices is an enormous problem, we would like to approach this problem by taking one step at a time and first considering the most common web-supported devices. They include PCs and Pocket PCs that accept HTML [1], and cell phones that accept cHTML or WML [2]. We also start by initially focus on web pages and web applications which presentations are consisted of purely graphical widgets, and exploiting high-level form controls such as buttons, forms, and text fields. We select graphical modal as it is the most frequently used modality; and we concentrate on the high-level controls as they are commonly exploited in e-commerce web applications.

Challenges

Within our scope of interest, there are three major set of challenges:

- Presentation Accessibility – the ability for a system to produce a presentation data that can be rendered on a target device
- Presentation Quality – the ability for a system to produce a highly usable presentation
- Function Customization – the ability for an author or a user to customize functions of a web application

For presentation accessibility, it is caused by device-specific markup languages (e.g., PCs, and Pocket PCs support HTML, but DoCoMo cell phone only supports cHTML) and input methods (e.g., PCs support keyboards and mouse, Pocket PCs support stylus and virtual keyboards, and DoCoMo cell phones support keypads). For presentation quality, it is affected by different device screen sizes (e.g., PCs' screen size is above 800 x 600 pixels, Pocket PCs' screen size is around 320 x 240 pixels, and DoCoMo cell phones' screen size is around 130 x 120 pixels). For function customization, it is about task preference. Task preference means a task may be suitable for a few devices but not for others. For example, a text inputting task may be suitable for PCs but not for cell phones; as cell phones do not provide a convenient input mechanism. Task preference can also be based on delivery contexts [3] other than device types. Using the example in the W3C Authoring Scenario informal draft [4], if a user is in a hurry, he/she

may choose to disable all images presentation and just download the text. With the additional task preference option, the user can further minimize the download time by just selecting the task that he/she is interested.

Proposed Solution

To solve the above problems, we have looked at the following techniques:

- Mapping
- Task customization
- Single layout specification
- Control transformation

Mapping is used to abstract all device-specific markup languages. At design time, an abstract markup language is provided to authors to build a device-independent presentation model. At runtime, all abstract markups will be interpreted as a set of device-specific markups, based on the device's supported markup language. The supported markup language information is one of the delivery contexts that we assumed to be retrieved from the CC/PP protocol [5].

Task customization is used to specify the preferred tasks for each device. The specification can be done by authors or users, and the specification can be based on the device type, the interaction pattern, or other delivery context. At this moment, we only allow authors to make the specification, and the specification is based on the device type.

Both the single layout specification and control transformation are used to layout device-specific presentation. In our proposal, we put extra emphasis on the layout, as it directly affects the web application popularity and user experience. There are several proposals such as XIML [6], UIML [7] and XWeb [8] which are also targeting this problem. Their approaches can be categorized into two types: the autonomous approach and the automatic approach. The autonomous approach generates presentations autonomously by requiring very detailed layout from authors. For example, UIML requires authors to specify the presentation layout on each platform, which requires authors to have thorough knowledge on each device's capabilities. The automatic approach generates presentations automatically by leaving very little layout controls to authors. For example, XIML and XWeb generate presentations automatically requiring little inputs from authors. However, authors may not be allowed to specify their transformation preferences.

In our proposal, we introduce a new layout mechanism which constitutes a middle ground between the two approaches. Authors provide a *single* layout specification based on the presentation of the device with the *largest* screen size. From the *single* specified layout, we can deduce different layouts for devices with smaller screen sizes. Since large screen devices can accommodate a more complex layout than small screen devices, we can easily paginate the large and complex presentation, into smaller and simpler presentations for small screen devices. When the paginated presentation is too big to fit onto the target screen, we apply control transformation. A sample control transformation is shown in Figure 1.

The usage of all of the above techniques is discussed in the next section.

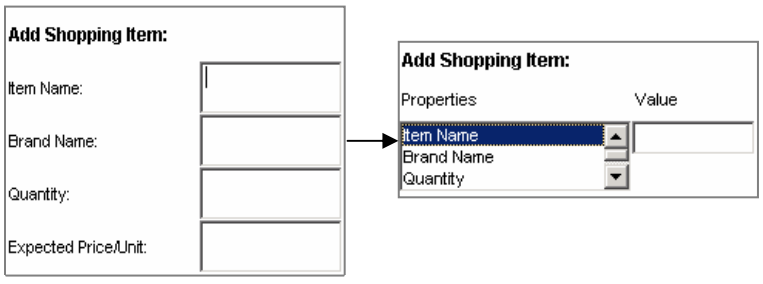


Figure 1: Sample control transformation

Single Authoring Technique

In order to achieve device independence, authors have to provide us a high level description of their web applications. The minimum information that we need is the tasks involved in a web application. Paterno et al [9] defines a task as an activity that has to be performed to reach a goal. It can be either a logical activity such as *retrieving information* or a physical activity such as *selecting a button*. This definition is adopted throughout this paper.

Authors can provide the tasks information to us by building a task model. Our task model is basically a decomposition of end-user's logical tasks. The task model has a tree-like structure. From the task model, authors can construct a device-independent presentation model by adding abstract controls as the leaves of the task model. The abstract controls are the presentations of the lowest-level logical task. Figure 2 depicts a device-independent presentation model for a *search item* web application. The root node, which represents an entire web application, occupies the top of the tree. Child nodes of the root node are task nodes representing different end-users' tasks. Each task node can be further divided into sub-task nodes, sub-sub-task nodes, and so forth, until the leaf nodes are represented by abstract controls. For example, the Search task node shown in Figure 2 is divided into 3 sub-task nodes, and they are represented by a "Search for the item:" abstract label, a "Search" abstract button, and an abstract text field.

On each task node, we also allow authors to specify a list of heuristics. These heuristics will be exploited by the task customization, layout algorithm, and transformation. The heuristics specify that (1) a detail layout for each node based on a Java Grid Bag Layout Constraint, which is the most flexible layout constraint, (2) *task preference* describes whether a task is suitable for a particular device, (3) *priority* denotes the desired layout sequence of each widget, (4) *split-ability* indicates whether the widgets can be paginated over multiple pages, and (5) *importance* indicates whether the widget is a necessary component or an optional component to represent an application function. For example, the Search Item task in Figure 2 has 3 sub-tasks. In order to perform a Search Item task, an end-user must enter the item's name and press a button to initiate the searching process. However, an end-user is not required to specify the sorting preferences and the display options, as they are just enhancements for displaying the search result. Thus, widgets associated with the Search task are considered as necessary and the remaining widgets are optional.

After the device-independent presentation model is constructed, a rendering engine can perform task customization, layout, transformation, and render the device-independent presentation model into device-specific presentations.

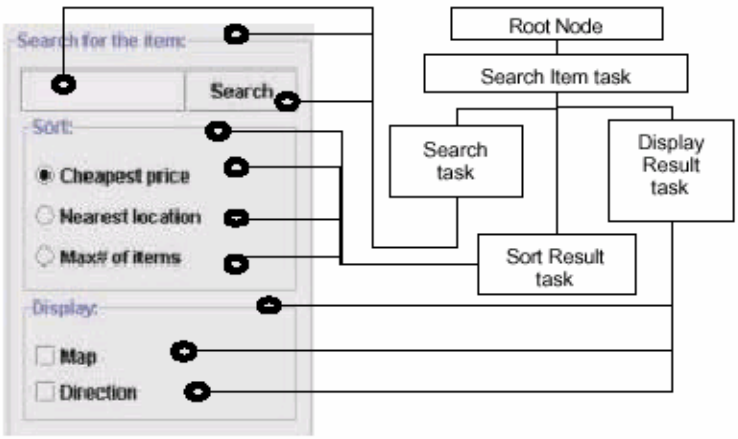


Figure 2: Device-independent presentation model of a *search item* web

Example

In the following example, we exploit the above discussed techniques to generate a PC presentation, a Pocket PC presentation, and a cell phone presentation from a device-independent presentation. The single layout specification is based on the PC presentation. Other implementation details are reported in [10, 11].

In Figure 3, a one-page PC presentation is paginated into a four-page Pocket PC presentation, and into a 5-page cell phone presentation. Also, from the Pocket PC presentation, we can notice the layout of some controls are re-arranged in order to fit the presentation onto the device's screen. From the cell phone presentation, we can further observe that transformation is applied on the first and the third page. Transformation is applied as the layout re-arrangement cannot shrink the PC presentation down to a cell phone screen's size.

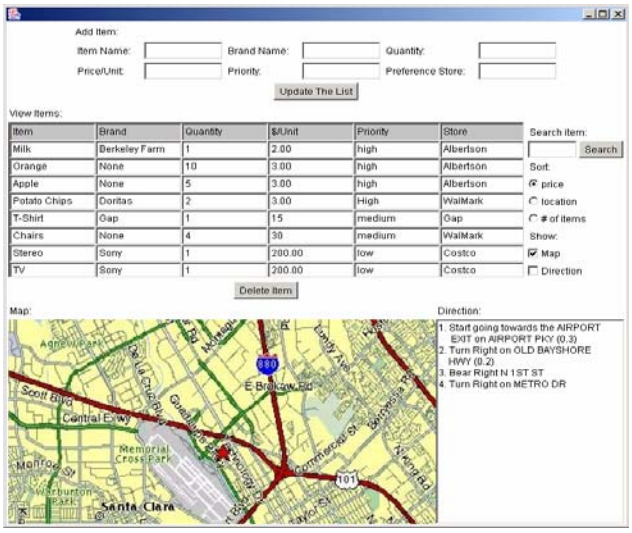
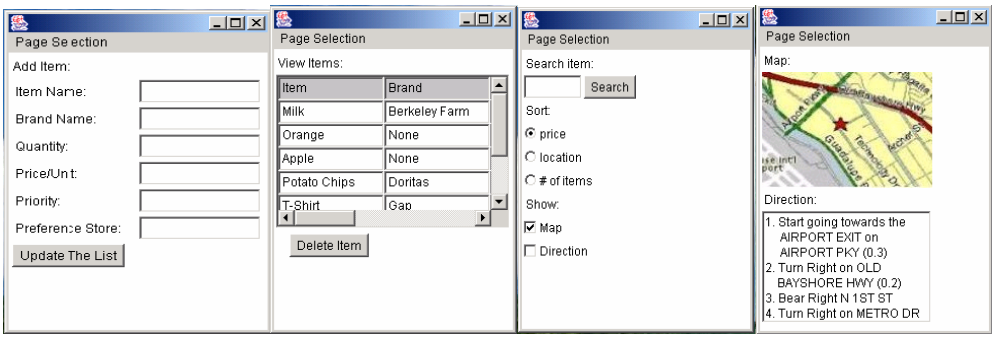
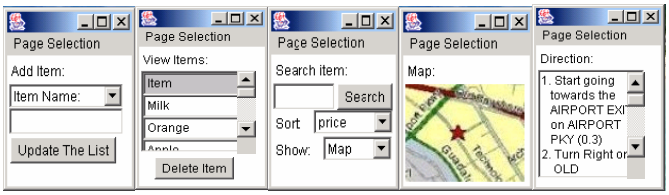


Figure 3: Sample a) PC presentation, b) Pocket PC presentation, and c) cell phone presentation, generated from a device-independent presentation

a) PC Presentation



b) Pocket PC Presentation



c) Cell Phone Presentation

Discussion

Based on our experience in implementing the above example, we provide a list of open questions which we believe can make device independence approach more feasible.

1. *Smooth transition from concrete authoring to abstract authoring*
   Authors are used to author a web page with the prior advanced knowledge of the target device's capabilities. To achieve device independence, authors need to switch from this concrete authoring practice to a new abstract authoring approach. What are the most appropriate steps or required guidance for this transition?

2. *Consistency and Customization*
   The generated device-specific presentations can be very consistent on different devices, or they can be fully customized for each individual device. However, the extreme of either approach can degrade the usability drastically. The consistency can minimize users' learning time in using the same web application on any device, but the presentation may not fully leverage the target device's capabilities. Thus, the presentation may not provide its maximum usability. On the other hand, customization can increase each presentation's usability, but users' may require a long learning time for each totally different presentation. Therefore, usability of the overall web application can be downgraded as users may have difficulties in identifying the corresponding features on various devices. We need to find out the balance between the consistency and customization.

3. *What is the good set of heuristics from authors to transformation engine so that it can generate high quality presentations?*
   In order to generate device-specific presentations from a device-independent presentation model, we need authors to provide heuristics or hints for device-specific presentation generations. We should keep the number of required heuristics to a minimum. Also, the heuristics should be easy to identify and specify, so that the effort for specifying the heuristics is considerable less than the effort for authoring separate device-specific presentations.

4. *How does single authoring approach accommodate common iterative design process in UI?*
   Most of the UI design is an iterative process. During each design iteration, the designer may need to see the generated device-specific presentation and may want to customize directly on the device-specific presentation. However, it is a challenge on how to preserve these device-specific customizations so that they are not lost when the designer makes future modifications on device-independent presentation model or other device-specific customizations.

Conclusion

Based on our experience with our prototype, we believe that single authoring approach can be made into a feasible approach. However, there are several open challenges mentioned above. We hope to gain more feedback and insight on these issues from the W3C Authoring Technique workshop.

Reference

1. W3C, "HTML 4.0 Specification", W3C recommendation, April 1998, *http://www.w3.org/TR/1998/REC-html40-19980424/*
2. W3C, "Compact HTML for Small Information Appliances", W3C Note, February 1998, *http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/*
3. W3C, "Device Independence Principles", W3C working draft, September 2001, *http://www.w3.org/TR/2001/WD-di-princ-20010918/*
4. W3C, "Authoring Scenarios for Device Independence", W3C informal public draft, July 2002, *http://www.w3.org/2001/di/public/as/as-draft-20020729.html*
5. W3C, "CC/PP Implementors Guide: Privacy and Protocols", W3C working draft, December 2001, *http://www.w3.org/TR/2001/WD-CCPP-trust-20011220/*
6. Eisentein, J., Vanderdonckt, J., and Puerta, A., "Applying Model-Based Techniques to the Development of UIs for Mobile Computers", *Pro. of ACM IUI'01*, January 2001, pp. 69-76.
7. Harmonia, Inc., "User Interface Markup Language (UIML) Draft Specification", January 2000.
8. Olsen, D., Jefferies, S., Nielsen, T., Moyes, W., Fredrickson, P., "Cross-modal Interaction using XWeb", *Proc. of ACM UIST'00, November 2000.*
9. Fabio Paterno, "Model-Based Design and Evaluation of Interactive Applications", August 1999, Springer-Verlag London Limited 2000.
10. Chu, H., Song, H., Wong, C., and Kurakake, S., "Seamless Applications over Roam System", *UbiTools'01* (Part of *ACM UbiComp'01*), September 2001, *http://choices.cs.uiuc.edu/UbiTools01/*.
11. Wong, C., Chu, H., and Katagiri M., "GUI Migration across Heterogeneous Java Profiles", *Pro. of ACM SIGCHI-NZ'02, July 2002.*