Personalizing Web Sites with Mixed-Initiative Interaction

Saverio Perugini and Naren Ramakrishnan

The IT industry has interpreted "personalization" in many ways (D. Riecken, "Personalized Views of Personalization," *Comm. ACM*, vol. 43, no. 8, Aug. 2000, pp. 26-28). Personalization refers to the automatic adjustment of information content, structure, and presentation tailored to an individual user. Commercial Web sites increasingly employ personalization to help retain customers and reduce information overload. For instance, Amazon's e-commerce site is estimated to have at least 23 different types of personalization (J. Riedl, "Personalization and Privacy," *IEEE Internet Computing*, Nov.-Dec. 2001, pp. 29-31).

When the user and the browser take turns changing the flow of interaction, it means improved personalization for the user and simple implementation choices for the site designer.

But what does it mean for a Web site to be personalized? Some Web sites, especially e-commerce sites, welcome returning users and remember personal details, such as credit card numbers. Other Web sites track purchase patterns and recommend specific products. Still others provide browsing aids, such as top-10 visited links.

A Web site is personalized if a user can interact with the site in an expressive way to achieve his information-seeking goals. Thus, personalizing the user's interaction is the best way to achieve personalization. This approach is complementary to thinking of personalization in terms of content relevance, delivery speed, and qualitative criteria such as utility and customer satisfaction.

Inside

Human Dialogues

To recognize personable interaction with a Web site, consider the human-to-human dialogues between a camera buyer and a dealer in the "Human Dialogues" sidebar.

Both conversations involve the specification of camera attributes, but they differ in important ways. The buyer in the first dialogue responds to questions in the order the dealer chooses to pose them. The dealer has the initiative at all times, and we refer to such an interaction as a *directed dialogue*. In the second dialogue, when the dealer prompts the buyer about camera manufacturer, the buyer instead responds with information about single-lens reflex (SLR), his choice of type, in line 3 of dialogue 2. The buyer in dialogue 2 thus takes the conversational initiative from the dealer. Nevertheless, the conversation does not stall, and the dealer continues with the other aspects of the information-gathering activity. In particular, the dealer registers that the buyer has answered a different question than the one he was asked, and the dealer refocuses the dialogue in line 4 to the issue of manufacturer (this time, narrowing down the available options). Such a conversation—where the dealer and buyer exchange initiative—is called a *mixed-initiative interaction* (J.F. Allen and colleagues, "Towards Conversational Human-Computer Interaction," *AI Magazine*, Winter 2001, pp. 27-37).

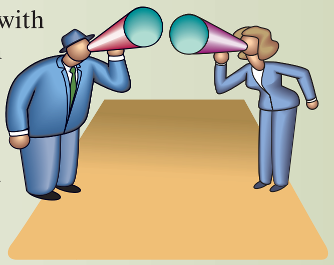
Our goal is to provide an interaction instrument so the user can take the initiative in Web site interactions.

MIXED-INITIATIVE INTERACTION

How can we have similar interaction flexibility with a Web site? More importantly, what does it mean to take the initiative from a Web site? Users predominantly interact with Web sites by clicking on presented hyperlinks. Any time we click on a

Human Dialogues

To understand personalized interaction with a Web site, consider the following human dialogues between a camera buyer and a dealer. Flexibility of Web site interaction similar to that in Dialogue 2 requires an interaction instrument so that the user can take the initiative.



Dialogue 1

1. Buyer: "I want to buy a camera."
2. Dealer: "Sure, is there a particular manufacturer you are interested in?"
3. Buyer: "Nikon."
4. Dealer: "What type of Nikon camera would you like?"
5. Buyer: "An SLR model."
6. Dealer: "Sure, we have those. Now, ..." (conversation continues to ascertain more details).

Dialogue 2

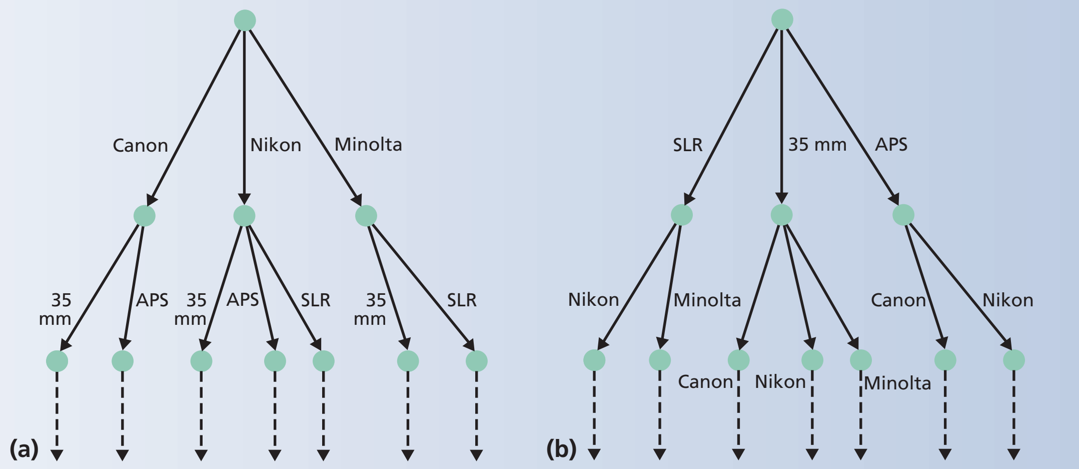
1. Buyer: "I want to buy a camera."
2. Dealer: "Sure, is there a particular manufacturer you are interested in?"
3. Buyer: "Not really, but it has to be SLR."
4. Dealer: "I see. Only Nikon and Minolta have SLR cameras."
5. Buyer: "OK, in that case, ..." (conversation continues).

hyperlink, we are responding to the choices already put forth by the Web site—in other words, this would be a directed dialogue. Browsing is, hence, not mixed-initiative, because the initiative always resides with the Web site.

Given this handicap and to support rich interactions, Web sites have traditionally hardwired multiple browsing interfaces to cover all possibilities. A Web site organization such as that shown in Figure 1a would be appropriate for the first buyer, who thinks of cameras primarily in terms of their manufacturer and only secondarily in terms of type. Conversely, Figure 1b would be appropriate for the second buyer, who thinks of cameras in terms of lens equipment first. Many Web sites are indeed organized along such multiple facets and shift responsibility to the user, who must employ the right interface for his information-seeking activity.

Such designs present two problems. The first is the explosion of scenario possibilities. If cameras are distinguished by, say, six independent attributes, then we have to support $6 \times 5 \times 4 \times 3 \times 2 = 720$ possible browsing organ-

Figure 1. Two organizations of a camera catalog: The first is by maker-type (a) and the second is by type-maker (b).



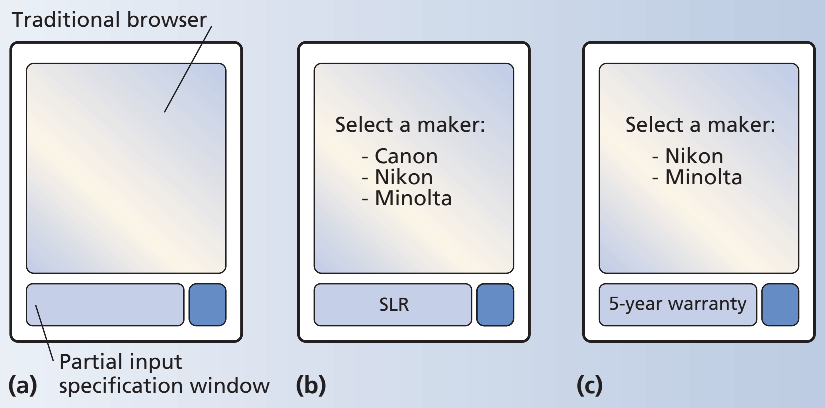
These figures show only two levels of the Web site hierarchy for ease of illustration. The nodes are Web pages, the edges denote hyperlinks, and labels on edges represent the text anchoring the hyperlinks.

izations! Web sites such as http://epicurious.com, a site for organizing recipes, actually take such an exhaustive approach and support all possible ways of interacting with an information system (M. Hearst, "Next Generation Web Search: Setting Our Sites," *IEEE Data Engineering Bulletin*, Sept. 2000, pp. 38-48).

The more fundamental problem with such designs is that they overspecify the personalization goals by anticipating all the forms of interactions that the site must support.

Web sites are not traditionally designed for mixed-initiative interaction; this is because Web interaction started out as a simple means for retrieving pages from a server. Moreover, the HTTP access protocol is stateless because it does not retain information about current user interactions for future use. Because of this traditional usage paradigm and statelessness, few interaction instruments exist for mixed-initiative interaction. Arguably the only interaction instrument that lets users take the initiative is the location URL box in many browsers—the user can choose to discard the current site and enter a different site's URL to browse. This form of mixed initiative is very restrictive and does not let users take the initiative *within* the current Web site.

Figure 2. Interface for personalized interaction with Web sites.



The top window of the interface (a) supports traditional browser functionality. At any point in the interaction, the user can supply personalization aspects out of turn (bottom two windows). You can implement such an interface as a browser toolbar. At the beginning of an example interaction, the user decides not to use any of the presented hyperlinks for camera model. Instead, he uses the toolbar to specify his camera type choice out of turn (b). Processing this input causes the browser to remove the Canon option from the model choices (c) because that manufacturer does not offer the specified type of camera. Once again, the user opts to use the toolbar to obtain warranty information (results not shown).

For the user to take the initiative in Web site interactions, our approach is to implement an out-of-turn interaction toolbar as a plug-in to many browsers such as Mozilla (http://www.mozilla.org) and Internet Explorer. With this instrument, the Web site implements only one design, but because the toolbar lets users take the initiative, the site can support any mixed-initiative interaction. The site designer no longer needs to support all possible interfaces directly in the hyperlink structure. For users, the interface appears less cluttered and the interaction resembles more of a real dialogue, with all its attendant advantages.

Figure 2 shows how this approach works. We will demonstrate how the Web site shown in Figure 1a supports both of the buyer-dealer interactions we discussed earlier. Figure 2b shows the top level of Figure 1a at the outset, which shows three camera manufacturers. This site trivially supports the first buyer, since he can proceed to click on "Nikon" first and then specify that he is interested in an SLR camera. This amounts to simple browsing. Because the second user doesn't wish to specify a maker at the outset, he uses the out-of-turn toolbar to specify the aspect "SLR" out of turn. The next stage of the dialogue (Figure 2c) shows that the browser has accounted for this input by revising the manufacturer list. Thus the browser plug-in now makes one site support multiple modes of interaction. Similarly, we could have used Figure 1b to support both users.

But how exactly does the out-of-turn toolbar work? Think of interaction with a Web site as a sequence of conditional statements to be evaluated, as shown in Figure 3a. Imagine these conditionals written in any programming language, such as C. Notice that the nested structure mimics the progressive drilling down within the hierarchy. For the user who clicks on "Nikon" (and, hence, responds to the initiative), Figure 3b models what that user wants to happen. That is, the choices now reflect the three types of cameras under Nikon: 35 mm, APS (Advanced Photography System), and SLR. On the other hand, Figure 3c models what the user who enters "SLR" wants to happen. That is, the choices should reflect the choices of manufacturers, and only two, Nikon and Minolta, should be available. We can think of Figure 3 as capturing requirements for program transformations. Interestingly, the same program transformation algorithm can support both browsing and out-of-turn interaction.

This transformation algorithm is called *partial evaluation* (N.D. Jones, "An Introduction to Partial Evaluation," *ACM*

Figure 3. Modeling Web site interactions in a program.

```
if (Canon)
   if (35 mm)
   …
   if (APS)
   …
else if (Nikon)                                   if (Nikon)
   if (35 mm)              if (35 mm)             …
   …                       …
   if (APS)                if (APS)
   …                       …
  if (SLR)                 if (SLR)
   …                       …
else if (Minolta)                                 else if (Minolta)
  if (35 mm)                                      …
   …
  if (SLR)
   …

(a)                        (b)                    (c)
```

Original Web site (a) can display differently as the result of browsing for Nikon (b) or out-of-turn interaction with SLR (c).

Figure 4. Modeling Web site interactions in an XML format.

```
<site>
  <Canon>
    <35 mm>
      …
    </35 mm>
    <APS>
      …
    </APS>
  </Canon>
  <Nikon>
    …
  </Nikon>
  <Minolta>
    …
  </Minolta>
</site>
```

Computing Surveys, Sept. 1996, pp. 480-503). Partial evaluation simplifies portions of programs given some (but not all) of their input. In this view, Figure 3b results from partially evaluating the input program (Figure 3a) with respect to the variable "Nikon," and Figure 3c results from partial evaluation of the input program with respect to SLR.

As a concept in computing, partial evaluation is at least 30 years old and the approach we show here helps relate it to information personalization. It is automatic, in that off-the-shelf partial evaluators take programs such as Figure 3a as input and produce programs such as Figures 3b and 3c as output. Dozens of partial evaluators are available for transforming programs written in languages such as C, Fortran, Lisp, Scheme, Haskell, Java, and Prolog.

To recap, we have reduced the personalization of Web sites to partially evaluating a representation of interaction (N. Ramakrishnan, "PIPE: Web Personalization by Partial Evaluation," *IEEE Internet Computing*, Nov.-Dec. 2000, pp. 21-31). Remember that after simplifying the representation, we have the option of personalizing it with further partial evaluation, using a different input. In this way, supporting any mixed-initiative interaction is possible. Although we can use any partial evaluation software to implement such a Web personalization engine, it is easily achievable using the Extensible Style Sheet Language Transformation (XSLT) engine.

USING XSLT FOR PERSONALIZATION

XSLT is a mature technology for transforming XML documents from one vocabulary into another. As such, XSLT can implement many transformations, but we specifically use it here to support the partial-evaluation transformation (S. Mangano, *XSLT Cookbook*, O'Reilly & Associates, 2002).

We specify an XSLT transformation in the form of pattern-action rules in a style sheet, and the XSLT engine then recursively applies these rules, starting from the root of a tree-structured XML document. Whenever the XSLT engine encounters a match, the style sheet describes the particular actions to take.

To use XSLT for personalization, we must first model programs such as Figure 3a in XML. Figure 4 shows one possible approach to modeling the camera's Web site. Then for each user input, we prepare a suitable XSLT style sheet, outlining the transformation, and apply it on the XML source. For instance, for the user interested in "SLR cameras," the style sheet of Figure 5 would be appropriate. This style sheet specifies that you can simplify the "SLR" hyperlink and remove the "APS" and "35 mm" hyperlinks (because they are mutually exclusive with respect to "SLR"). We then apply additional post processing transformations to prune dead ends. For example, Canon has no SLR models so the transformation can remove it. Figure 6 shows an XSLT style sheet for pruning dead ends.

XSLT also facilitates other post-processing activities, many aesthetic and originally handled by ad-hoc mechanisms such as shell scripts. For example, typical partial evaluators will rename variables in a specialized program. Although such conventions do not affect the resulting program's semantics, they do shatter the original symmetry between link labels and program variables. XSLT obviates the need to reconcile such differences because it does not rename XML elements unless told to do so.

In addition, with the help of JavaScript, we can easily implement a "feature(s) gleaned thus far: … " label at every stage of the interaction, to orient users. Lastly, high-level XSLT functions, such as sorting, simplify tasks such as link label ordering on re-created Web pages.

The transformation can be implemented as part of a proxy server that is designed to process input events communicated from the browser (either clicking on hyperlinks or specifying aspects out of turn). Recall that you must convert the shrunken XML document from an XSLT processor back into a Web site for a user to browse. The user might then proceed to click on any remaining links or might require a further degree of personalization, both of which another XSLT transformation can handle. Remember that clicking on a given hyperlink and entering out-of-turn input correspond to the partial-evaluation operation.

PROTOTYPE IMPLEMENTATION

To demonstrate these ideas, we designed a personalization system for the US Congressional portion of the Web site for Project Vote Smart (http://vote-smart.org). The site contains information about members of Congress; users interact with the site by choosing a state, party, branch of Congress, and seat. We represented the nonXML Web pages in an XML notation and extracted the site's hierarchy for presenting pages using a depth-first crawl of the site. A depth-first crawl is merely a way to capture the hierarchy as shown in Figure 3a.

Figure 7 shows an example of how mixed-initiative interaction proceeds. The site initially asks a user to choose a state. But in this example, the user prefers to specify politicians in terms of party and branch of Congress (such as "Democratic senators"). The result of the XSLT transformation removes many states, restricting the options to only those states that have Democratic senators. A demo of the site is available at http://pipe.cs.vt.edu.

In this demonstration example, we also exploited interesting dependencies underlying politicians' attributes. For instance, if the user enters "senior seat," he is referring to a senator, not a representative. So, we can partially evaluate with respect to these additional variables. As another example, entering "North Dakota" and "representative" in the current political landscape defines a unique member of Congress because North Dakota has only one representative, so the interface doesn't need party information. It is impor-

Figure 5. Style sheet generated for a user interested in SLR cameras (specified by XML element SLR).

```
<?xml version="1.0"?>

<!— Template for the query: SLR —>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

<xsl:output method="xml"/>

<xsl:template match="SLR">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="APS"/>

<xsl:template match="35 mm"/>

<!— matches any node, including the root —>
<xsl:template match="*|@*">
  <xsl:copy>
      <!— continues on any nodes except the root, is
       actually impossible at this point any way —>
      <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>
</xsl:stylesheet>
```

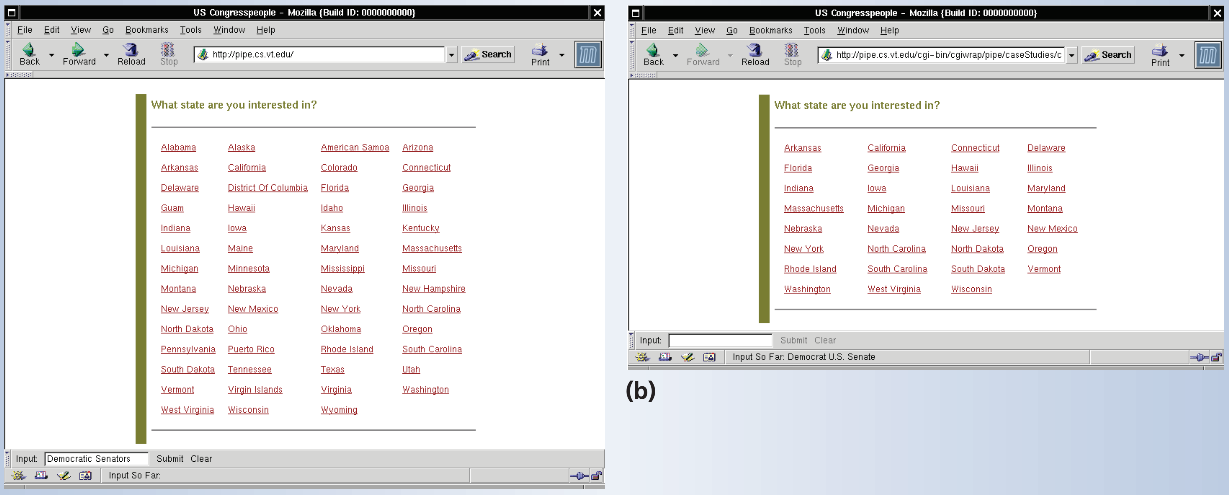
Running this style sheet through an XSLT processor with the XML document in Figure 4 emulates partial evaluation for personalization and transforms the document to reflect the site shown in Figure 3c.

Figure 6. Style sheet for pruning dead-ends.

```
<?xml version="1.0"?>

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/
   Transform" version="1.0">

<xsl:output method="xml"/>

<xsl:strip-space elements= "*"/>

<xsl:template match="@* | *[child: :node()]">
   <!— prunes dead-end nodes —>
   <!— only keeps nodes the have at least one text node —>
   <xsl:if test="descendant::text()">
     <xsl:copy>
        <xsl:copy-of select="@*"/>
          <xsl:apply-templates select= "@* | node()"/>
     </xsl:copy>
   </xsl:if>
</xsl:template>

</xsl:stylesheet>
```

We use this style sheet in conjunction with other style sheet transformations to post-process a resulting specialized XML document.

Figure 7. Prototype browser implementation for mixed-initiative interaction.



US Congresspeople - Mozilla {Build ID: 0000000000}

http://pipe.cs.vt.edu/

What state are you interested in?

Alabama	Alaska	American Samoa	Arizona
Arkansas	California	Colorado	Connecticut
Delaware	District Of Columbia	Florida	Georgia
Guam	Hawaii	Idaho	Illinois
Indiana	Iowa	Kansas	Kentucky
Louisiana	Maine	Maryland	Massachusetts
Michigan	Minnesota	Mississippi	Missouri
Montana	Nebraska	Nevada	New Hampshire
New Jersey	New Mexico	New York	North Carolina
North Dakota	Ohio	Oklahoma	Oregon
Pennsylvania	Puerto Rico	Rhode Island	South Carolina
South Dakota	Tennessee	Texas	Utah
Vermont	Virgin Islands	Virginia	Washington
West Virginia	Wisconsin	Wyoming	

Input: Democratic Senators Submit Clear

Input So Far:

(a)

US Congresspeople - Mozilla {Build ID: 0000000000}

http://pipe.cs.vt.edu/cgi-bin/cgiwrap/pipe/caseStudies/c

What state are you interested in?

Arkansas	California	Connecticut	Delaware
Florida	Georgia	Hawaii	Illinois
Indiana	Iowa	Louisiana	Maryland
Massachusetts	Michigan	Missouri	Montana
Nebraska	Nevada	New Jersey	New Mexico
New York	North Carolina	North Dakota	Oregon
Rhode Island	South Carolina	South Dakota	Vermont
Washington	West Virginia	Wisconsin	

Input: Submit Clear

Input So Far: Democrat U.S. Senate

(b)

The starting Web page for personalizing information about US Congressional officials (a) presents the user with options for clicking on a state of interest. The user, however, decides not to pursue this option and instead specifies his information need out of turn (b). The system responds with the set of possible states that satisfy the user's criteria. The interaction then continues (not shown).

tant to consider such facets in delivering a compelling personalized experience. The net effect of such considerations will be the initialization of multiple program variables based on the user's input, and the site created at every stage reflects an accurate summary of the remaining dialogue options.

OTHER APPLICATIONS

The mixed-initiative interaction facet is very common in human-to-human conversations and even some automated telephone dialogues (such as those provided by Tellme Networks Inc., which powers applications that allow users to find information on restaurants, sports, traffic, and stocks by dialing an 800 number). Our research helps bring mixed-initiative interaction to bear on Web site interaction as well.

The work presented here lets us study mixed-initiative interaction in other areas as well, especially the popular VoiceXML dialogue management architecture. VoiceXML is a markup language designed to simplify the construction of voice response applications (S. McGlashan and colleagues, "Voice eXtensible Markup Language: VoiceXML," version 2.00, VoiceXML Forum, Oct. 2001).

Initiated by a committee comprising AT&T, IBM, Lucent Technologies, and Motorola, VoiceXML has emerged as a standard in telephone-based voice user interfaces and in delivering Web content via voice. It describes interaction using a markup language not unlike that used in Figure 4: VoiceXML markup tags describe prompts, forms, and fields that constitute a dialogue. VoiceXML supports both directed and mixed-initiative dialogues.

After our initial research into the partial evaluation basis for mixed-initiative interaction, we investigated how VoiceXML supports mixed-initiative functionality. We showed that VoiceXML's form interpretation algorithm (the engine handling the dynamics of interaction) is actually a partial evaluator in disguise (N. Ramakrishnan, R. Capra, and M.A. Pérez-Quiñones, "Mixed-Initiative Interaction = Mixed Computation," *Proc. ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation* (PEPM), ACM Press, Jan. 2002, pp. 119-130). This shows that partial evaluation of a representation is truly a robust way of providing mixed-initiative interaction.

The XSLT transformation approach presented here lets us unify other forms of Web site personalization. The representation that we are partially evaluating can also model dynamic content (such as queries to databases). Consider a Web site that presents voter statistics according to various dimensions such as race, age, sex, and region. As the user browses the hyperlink structure, the site generates on-the-fly statistics by issuing queries to an underlying relational database and aggregating the results. Using the XSLT transformation approach, we can support multiple modes of drilling down and rolling up the hierarchy by partial evaluation, weeding out the unneeded individual queries.

Inverse personalization, where the user knows what information she needs and wants to locate it, is another area we are actively investigating. For instance, consider a Web site categorizing apartments for prospective renters. Using inverse personalization, we can issue a request such as "Under what conditions will Rockville Apartments be the only choice?" You can also implement such personalization queries using XSLT.

We are also exploring the design of multimodal Web interfaces where, for instance, the user interacts using both the traditional interaction instruments and via voice. Here, we can use the SALT (Speech Application Language Tags) standard to support out-of-turn voice input and apply the transformation ideas presented here to personalize Web sites that employ SALT markup.

Finally, the proliferation of wireless devices and the need to quickly present only the most important content on resource-starved handheld computers will make transformation approaches to personalization more important (C.R. Anderson, P. Domingos, and D.S. Weld, "Personalizing Web Sites for Mobile Users," *Proc. Tenth Int'l World Wide Web Conf.*, WWW2001, ACM Press, pp. 565-575).

The ideas presented here serve as a basis for providing such forms of personalization in the context of mixed-initiative interaction. ■

Saverio Perugini *is a PhD student in computer science at Virginia Polytechnic Institute (Virginia Tech). His research interests are information personalization and Web technologies. Contact him at sperugin@cs.vt.edu.*

Naren Ramakrishnan *is an assistant professor of computer science at Virginia Tech. His research interests are problem-solving environments, data mining, and personalization. Contact him at naren@cs.vt.edu.*

For more information on this or any other computing topic, visit our Digital Library at http://computer.org/publications/dlib.

We acknowledge the suggestions of the two anonymous referees whose comments have improved this article's presentation. This work is supported in part by US National Science Foundation, grant IIS-0136182.