JOURNAL OF OBJECT TECHNOLOGY
Online at http://www.jot.fm. Published by ETH Zurich, Chair of Software Engineering. ©JOT, 2005

Vol. 4, No. 3
Special issue: 6th GPCE Young Researchers Workshop 2004

Personalization by Program Slicing

Saverio Perugini
Department of Computer Science
University of Dayton, OH

Naren Ramakrishnan
Department of Computer Science
Virginia Tech, VA

Personalization involves customizing information access to the end-user. As any new area of computer science research it lacks formal models to guide the design of systems. In this paper, we present a *modeling methodology*, based on generative programming, for personalizing interactions with hierarchical websites. The methodology entails modeling a user's interaction with a site in a program and applying program slicing to personalize the interaction. While preserving interactivity, this approach does not require the designer to anticipate all possible user interactions *a priori* and provide interfaces for each. Moreover, it provides a theoretical, systematic, and implementation-neutral way to design systems and is therefore a timely contribution to the young field of personalization as well as a novel application of generative programming.

1 INTRODUCTION

Personalization involves customizing information access to the end-user. Examples of systems include recommender systems [10] and adaptive websites [3]. The primary goal of a personalization system is to reconcile an information system's one-size-fits-all nature with the numerous, diverse, and evolving information-seeking needs and strategies of its users. This mismatch arises when a user has some information pertaining to her information need, but is unable to communicate it because the system is soliciting information which the user does not have.

Example: Navigating a Website

Consider Fig. 1 (left) which depicts a hyperlinked taxonomy of links to web resources akin to Yahoo!. At the top level, the site asks the user to select a category – arts or computers. A user interested in resources involving 'music' is thus unable to communicate her information need since the site does not contain a 'music' hyperlink at the top level (one is presumably nested deeper in the site under one of the two, or both, categories). Thus, to pursue this information-finding goal the user would have to follow all paths through the site to their leaves and manually filter out those

Cite this article as follows: Saverio Perugini and Naren Ramakrishnan: "Personalization by Program Slicing", in *Journal of Object Technology*, vol. 4, no. 3, April 2005, Special issue: 6th GPCE Young Researchers Workshop 2004, pp. 5–11, http://www.jot.fm/issues/issue_2005_04/article1

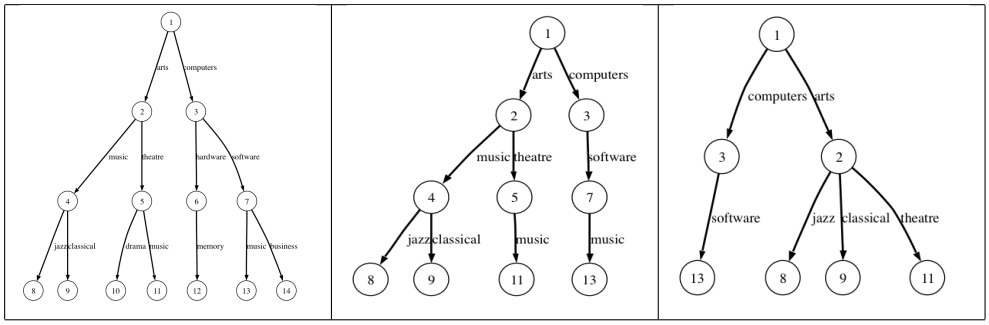


Figure 1: (left) Hypothetical hierarchical web directory with characteristics similar to those in Yahoo!. (right and center) Personalized versions of (left) w.r.t. 'music.'

which do not lead to information about 'music' – a tedious process!

Existing Approaches

The predominate solutions to this problem are symbolic links, such as those whose with a '@' appended to their label in Yahoo.com, and site-specific search tools, such as those at Amazon.com. A symbolic link is a special type of hyperlink which makes a directed connection between a webpage along one path through a website to a page along another path. The intended purpose of symbolic links on the web is to tighten the gap between an item's actual placement in a website and a user's prediction of that placement during information seeking. For example, the designer might include a symbolic link labeled 'music@' from the 'computers' sub-branch of a taxonomy to a page in the 'arts' sub-branch dealing with music to accommodate users who desire information about music but selected the top-level category 'computers.' In this manner, symbolic links give the illusion that an item is classified in more than one section of a website. Symbolic links prevent the user above searching for web resources on music from following dead-ends. Including symbolic links forces the designer to anticipate all possible categories users may perceive to contain information on a particular topic and provide links from those categories to the categories where the information actually resides. On the other hand, site-specific search tools provide the user with a textbox to enter a keyword query when she does not find what she seeks on the current page and return a flat list of ranked results (links). By returning a flat list these tools destroy the organization and context provided by the original classification. As a result they compromise and curb the interactive nature of information-seeking inherent in the hyperlinked classification. Information-retrieval researchers however have championed interaction as vital to information-seeking [5].

Desiderata

For these reasons we desire a solution which involves no anticipation while retaining interactivity. The goal of our research is to personalize these information-seeking interactions for users so that they can navigate the site in a manner reflective of their model of information-seeking without (i) requiring the designer to anticipate all possible interactions *a priori* and provide support interfaces (e.g., symbolic hyperlinks) for all of them and (ii) loosing the classification.

From the user perspective, our approach involves increasing the scope of addressable information (e.g., 'music') without enumerating several classifications. At any point in the interaction, including the top level of the site, when the user does not see what she likes on the current webpage, we provide an interface for her to communicate 'music' to the system, by, e.g., speaking to her browser. We interpret the submission of such an *out-of-turn* input as a desire to experience a path through the site containing a hyperlink labeled 'music.' And we retain those paths for the user and prune all others (ref. Fig. 1, center). Finally, we shrink each remaining path w.r.t. the communicated information. The result is a website personalized w.r.t. the user's input (ref. Fig. 1, right).

As any young and evolving area of CS research, personalization lacks software methodologies for designing systems in a systematic manner. In this paper, we briefly describe a modeling methodology, based on generative programming, for personalizing interactive information access in hierarchical hypermedia as illustrated above.

2 GENERATIVE MODEL

We developed a programmatic approach to personalizing interaction in hierarchical hypermedia. The approach involves modeling user interaction with a website programmatically and using program slicing to retain user specified paths through the site (and incidentally prune all others).

Modeling Interaction Programmatically

To model a user's interaction with the site in Fig. 1 (left) we *think* of interaction as being organized along a series of nested conditionals, where program variables model hyperlink labels (ref. Fig. 2, left). Notice that the structure of the nesting reflects the hierarchical nature of the site while the control flow of the program models choices made by the navigator en route to a leaf webpage. *Such a program is just a representation of interaction and not intended to ever be compiled or executed.*

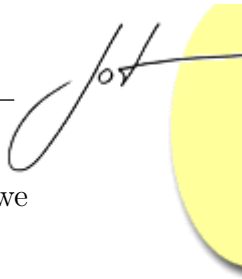
While the role of representation is recognized as important to personalization [7], most have modeled interaction, some even programmatically [9], to subsequently realize the specific interaction modeled [1]. Here we model interaction primarily

```
 1  if (arts)                  if (arts)                  if (arts)
 2    if (music)                 if (music)
 3      if (jazz)                  if (jazz)                  if (jazz)
 4        page = 8;                  page = 8;                  page = 8;
 5      if (classical)             if (classical)             if (classical)
 6        page = 9;                  page = 9;                  page = 9;
 7    if (theatre)               if (theatre)               if (theatre)
 8      if (drama);
 9        page = 10;
10      if (music);                if (music)
11        page = 11;                 page = 11;                 page = 11;
12  if (computers)             if (computers)             if (computers)
13    if (hardware)
14      if (memory)
15        page = 12;
16    if (software)              if (software)              if (software)
17      if (music)                 if (music)
18        page = 13;                 page = 13;                 page = 13;
19      if (business)
20        page = 14;
```

Figure 2: Modeling interaction programmatically. (left) Programmatic representation of interaction with the website modeled by Fig. 1 (left). (center) Representation of interaction with Fig. 1 (center) resulting from slicing (left) w.r.t. '`music`.' (right) Representation of interaction with Fig. 1 (right) resulting from partially evaluating (center) w.r.t. '`music` = `1`.'

	program	**backward (6, `vol`)**	**forward (1, `h`)**
1	`read (r, h);`	`read (r, h);`	`read (r, h);`
2	`cAr = π*r`^{2}`;`	`cAr = π*r`^{2}`;`	
3	`sAr = 2*cAr+2*r*π*h;`		`sAr = 2*cAr+2*r*π*h;`
4	`vol = cAr*h;`	`vol = cAr*h;`	`vol = cAr*h;`
5	`print (sAr);`		`print (sAr);`
6	`print (vol);`	`print (vol);`	`print (vol);`

Figure 3: Illustration of program slicing (simplified for purposes of presentation). (left) A program which takes the radius and height of a cylinder as input and computes and prints its surface area and volume. (center) A backward slice w.r.t. `vol` at line 6. (right) A forward slice w.r.t. `h` at line 1. (variable key: `r` = radius; `h` = height; `cAr` = circle area; `sAr` = surface area; `vol` = volume).



to achieve a *personalized* form of the particular interaction modeled. To do so we transform the representation using program slicing.

Transforming Programs

When the user communicates 'music' at the top level, to personalize her interaction we transform the representation by program slicing to retain those sequences annotated with 'music' and prune all others. Program slicing [2] is a theoretical operation used to extract statements that affect (or are affected by) the computation of a variable of interest at a point of interest from a program. There are several varieties of slicing; Backward and forward slicing are the two most relevant for our purposes (ref. Fig. 3). Slicing has been predominately applied to problems in software engineering such as debugging and reverse engineering [2]. Only few have applied it to web application development [11]. Our use of it here helps relate it to information personalization.

When the user says 'music' we forward slice the program in Fig. 2 (left) w.r.t. the `music` variable at all program points. This leads to the (`page` assignment) statements at lines 4, 6, 11, and 18 from which we backward slice the program. The result is a representation of interaction with the site which only contains paths involving hyperlinks labeled 'music' leading to terminal webpages containing information about 'music' (ref. Fig. 2, center). Finally, we partially evaluate [4] the program w.r.t. the user's input (statically set to 1) thereby removing all expressions involving the variable modeling the input (since it only appears in `if` statements). This results in a model of interaction with the personalized website (ref. Fig. 2, right) from which an actual site can be recreated.

3 DISCUSSION

This paper has provided a sneak peek into a modeling methodology, based on program slicing, for personalizing interactions with hierarchical hypermedia. The reader will have noticed that this approach includes the desired properties of personalization systems introduced above, namely interactivity without anticipation. Specifically, the same transformation technique is employed irrespective of whether the information communicated is currently solicited (communicated by clicking on a presented hyperlink) or solicited somewhere deeper in the site (communicated by speaking the browser). In this manner, we have unified multiple interaction techniques in a single (programmatic) framework. Moreover, our model provides a theoretical, systematic, and implementation-neutral way to design systems. Using this model, and the transformation capabilities of XSLT to implement slicing, we have personalized interactions with several websites [6]. Due to space limitations we are unable to expound on the model, but refer the reader to [8].

This is ongoing work that is part of a larger effort to model and realize information-

seeking interactions programmatically with generative and transformation techniques, including currying and reflection, as described in [6].

REFERENCES

[1] N. J. Belkin, C. Cool, A. Stein, and U. Thiel. Cases, Scripts, and Information-Seeking Strategies: On the Design of Interactive Information Retrieval Systems. *Expert Systems with Applications*, Vol. 9(3):pp. 379–395, 1995.

[2] D. W. Binkley and K. B. Gallagher. Program Slicing. In M. V. Zelkowitz, editor, *Advances in Computers*, Vol. 43, pp. 1–50. 1996.

[3] P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 11(1–2):pp. 87–110, 2001.

[4] N. D. Jones. An Introduction to Partial Evaluation. *ACM Computing Surveys*, Vol. 28(3):pp. 480–503, September 1996.

[5] J. Koenemann and N. J. Belkin. A Case for Interaction: A Study of Interactive Information Retrieval Behavior and Effectiveness. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96)*, pp. 205–212, Vancouver, Canada, 1996. ACM Press.

[6] M. Narayan, C. Williams, S. Perugini, and N. Ramakrishnan. Staging Transformations for Multimodal Web Interaction Management. In *Proceedings of the Thirteenth ACM International World Wide Web Conference (WWW'04)*, pp. 212–223, New York, NY, May 2004. ACM Press.

[7] E. P. D. Pednault. Representation is Everything. *Communications of the ACM*, Vol. 43(8):pp. 80–83, August 2000.

[8] S. Perugini. *Program Transformations for Information Personalization*. Ph.D. dissertation, Department of Computer Science, Virginia Tech, May 2004. ETD available at http://scholar.lib.vt.edu/theses/available/etd-06252004-162449/.

[9] C. Queinnec. The Influence of Browsers on Evaluators or, Continuations to Program Web Servers. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pp. 23–33, Montreal, Canada, September 2000. ACM Press. Also appears in *ACM SIGPLAN Notices*, Vol. 35(9), September 2000.

[10] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, Vol. 40(3):pp. 56–58, March 1997.

[11] F. Ricca and P. Tonella. Web Application Slicing. In *Proceedings of the International Conference on Software Maintenance (ICSM'01)*, pp. 148–157, Florence, Italy, November 2001. IEEE Computer Society.

ABOUT THE AUTHORS

Saverio Perugini is an Assistant Professor in the Department of Computer Science at the University of Dayton. His research interests include the application of program transformations to problems in information retrieval. Email: saverio@udayton.edu

Naren Ramakrishnan is an Associate Professor in the Department of Computer Science at Virginia Tech. His research interests are problem-solving environments, data mining, and personalization. Email: naren@vt.edu