Personalized Web Views for Multilingual Web Sources

When browsing information on large Web sites, users often receive too much irrelevant information. The WWW Information Collection, Collaging, and Programming (Wiccap) system lets ordinary users build personalized Web views, which let them see only the information they want — and in the way they prefer. It provides a set of GUI tools, including a mapping wizard, extraction agent, and presentation toolkit, that hide the internal representation from ordinary users, but let experienced users create more advanced mapping rules. Wiccap encapsulates HTML- and language-specific details from the original Web site, facilitating the creation of personalized Web views for multilingual sites.

Zehua Liu, Wee-Keong Ng, and Ee-Peng Lim
Nanyang Technological University

The vast amount of irrelevant information on most large Web sites can overwhelm users. Advertisements and even some portions of a site's core content are often of no interest to users. Moreover, the content they are interested in is often hidden amid the unwanted information. To alleviate this information overload, users need to be able to specify *personalized Web views* that let them see only what they want to see on a given site.

As the amount of information available in local languages increases, personalized Web views will also need to work seamlessly with multilingual information sources (mainly Web sites in various languages). An especially challenging problem is designing a language-independent framework for creating and browsing personalized Web views.

To attack these problems, we combine the personalized Web view concept with Web information-extraction techniques. Our software framework, the WWW Information Collection, Collaging, and Programming (Wiccap) system, lets ordinary users create and browse personalized Web views. The framework includes a well-defined process for creating personalized Web views and a set of tools that hides the technical construction details, making Wiccap easy to use. The underlying mechanisms for implementing the Web views ensure that the framework and personalized Web views are language independent.

Personalized Web Views

In this article, we define a personalized Web view rather loosely, depending on the extent of customization. For example, a *partially* personalized Web view might omit one component. (We use "Web view" and "personalized Web view" inter-

 Published by the IEEE Computer Society 1089-7801/04/$20.00 © 2004 IEEE

changeably to refer to both complete and partial views when there is no ambiguity.)

A personalized Web view has three components, as Figure 1 shows. The *global logical view* represents users' perception of a Web site's logical structure. It contains most of the information a user wants from a site. A personalized Web view usually has one global view. However, users can have more than one global view if they combine data from multiple sources (that is, Web sites).

The second component, *customization parameters*, lets users refine and filter the original global views to keep only the portions they are interested in.

The *presentation parameters* component lets users select a visual style in which to display the information. As users typically have several Web views active at the same time, they can schedule the views in different time slots and at different intervals in a TV-program-like fashion.

For example, as a database researcher, Andy wants to know when the Stanford database group publishes new material. He creates a personalized Web view by picking the global view that models the corresponding site. The scope might not be detailed, but he specifies that information should be updated weekly and only new items reported. He also schedules new items to be shown at 8:30 a.m. every Monday. Being a soccer-lover, Andy also wants to read sports news every day. He creates another Web view with the global view of Sina News (a popular Chinese news portal) and confines the scope to the sports section. Finally, he creates a view that lets him read headline news every day at lunchtime. To save time, he sets it so that news titles are displayed in an auto-scrolling list with external links to the full articles.

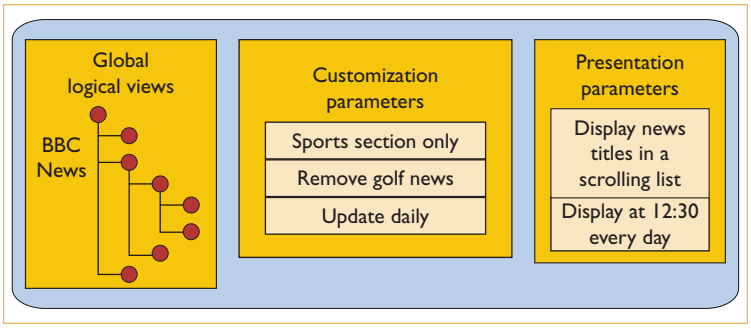

Figure 1. A personalized Web view for sports news from an online news site. The Web view has three components that control information extraction and refinement as well as its presentation and scheduling.

Wiccap Architecture

The Wiccap system constructs personalized Web views simply and efficiently. The system has a three-layer architecture, as Figure 2 (next page) shows, and each layer is responsible for creating one component in a three-step process:

- The *mapping wizard* takes information from a particular Web site and produces global logical views (or *mapping rules*) for the site.
- The *network extraction agent* (Neat) lets users customize these logical views according to their preferences and extracts and transforms the desired information from the Web site.
- Finally, the *Web information player and programmer (Wipap) presentation toolkit* lets users apply different presentation styles and templates to personalize the information.

Separating this process into three steps is important to our goal of empowering ordinary users to create their own Web views. This lets expert users perform the information-modeling task (first step), which requires some technical knowledge (especially in the subtask of specifying the extraction details), while still enabling ordinary users, whose main interest is simply obtaining information, to invoke the view-customization and presentation tasks (second and third steps). Without such explicit separation, users might extract inaccurate (because of ordinary users' lack of knowledge) or undesired information (because expert users misunderstand needs). The three-layer architecture is accessible to ordinary users while maintaining high accuracy with the extracted information and high efficiency with the logical-view creation process.

The system stores intermediate information, such as global and customized views, using XML documents together with XML schemas that define their formats. Thus, any XML-enabled application can easily receive the extracted results. For instance, an information-integration system could accept multiple Web sites' global logical views (the first-layer output) and integrate them to allow uniform access.

Mapping Wizard

The mapping wizard automates and facilitates the process of creating global logical views, which are represented internally by a logical data model.

Logical Data Model

In most existing systems, such as NoDoSe and

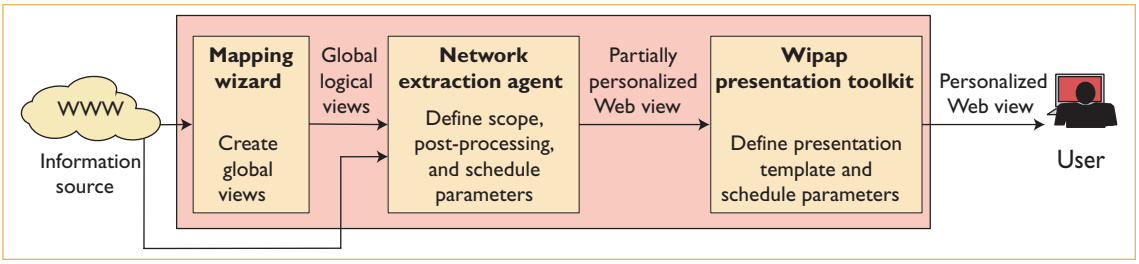

Figure 2. Wiccap system architecture. Each layer in the architecture creates one component of a personalized Web view.

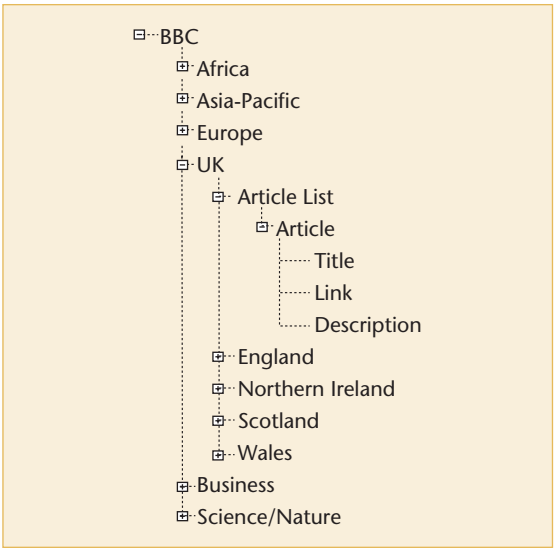

Figure 3. Logical view of BBC Online News. Most users understand this hierarchy of information and can use it to find most of the information they want to extract.

WebViews,[1,2] users specify the data to be extracted directly on the Web page. This usually requires technical knowledge about HTML and relevant technologies. Providing ordinary users with a logical view representing the target Web site and letting them specify the data to be extracted based on this view is much easier and less risky. For this to be possible, such logical views must reflect most users' understanding of the Web site and represent most of the data that users might want to extract.

The Wiccap data model[3] (WDM) represents global logical views. It captures Web sites' commonly perceived logical structures instead of their physical file directory structures. For instance, a newspaper Web site typically lists sections of the paper, such as world, local, and sports news. Each section might have subsections or a list of articles, which in turn might include several items, such as the title, abstract or summary, the article itself, and perhaps related articles. This hierarchy of information is the commonly perceived structure of a newspaper Web site; it reflects most users' concept and understanding and covers most of the data that users would like to extract.

Figure 3 depicts part of the logical structure for the BBC Online News Web site (http://news.bbc.co.uk/), and Figure 4 shows the corresponding XML representation. The Wiccap data model defines two sets of XML elements:

- A set (including `Wiccap`, `Section`, `Region`, `Record`, and `Item` elements) for describing the target Web site's logical structure. This set hides the physical structure of the Web site from users.
- A set (including `Mapping`, `Link`, and `Locator` elements) for maintaining the mapping between the logical and physical structures. This set lets the Wiccap extraction engine obtain data within a page, follow hyperlinks to obtain data existing across multiple pages, and automatically submit HTML forms.

Multilingual Support

To allow personalized Web views to work with Web sites in various languages, the data model must encapsulate language-encoding details so that users creating personalized Web views can ignore the language-dependent aspects. As noted earlier, the system uses XML to store intermediate data, including the global logical views. XML's mandatory support of internationalization, especially Unicode, lets us pass the burden of handling multilingual issues to the XML parser. Wicapp stores the global logical views in XML with UTF-16 encoding, which includes most characters in non-English languages, including East Asian languages such as Chinese.

In addition, separating the two sets of elements in WDM keeps the language-dependent details at the level of the second set, where the mapping details are specified. We use the second set of elements to define the patterns used in the

```
<!-- WiccapNewsBBC.xml -->
<?xml version="1.0" encoding="UTF-16"?>
<Wiccap Name="BBC" Group="News"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="WiccapNews.xsd">
 <Mapping>
  <Link Type="Static">http://news.bbc.co.uk</Link>
 </Mapping>
 ... ...
 <Section Name="UK">
  <Mapping>
   <Link Type="Dynamic"> ... ... </Link>
  </Mapping>
  <Region Name="ArticleList">
   <Mapping>
    <Locator Type="ByPattern"> ... ... </Locator>
   </Mapping>
   <Record Name="Article" NumOfItem="3">
    <Mapping> ... ... </Mapping>
    <Item Type="Link" TagFilter="Off" Description="the link to the news">
     <Mapping><Locator Type="ByPattern"><LocatorPattern>
      <BeginPattern><![CDATA[<a href="]]></BeginPattern>
      <EndPattern><![CDATA[">]]></EndPattern>
     </LocatorPattern></Locator></Mapping>
    </Item>
    <Item Type="Title" Description="the title of the news">
     <Mapping><Locator Type="ByPattern">
      ... ...
     </LocatorPattern></Locator></Mapping>
    </Item>
    <Item Type="Description" Description="the description of the news">
     <Mapping><Locator Type="ByPattern"><LocatorPattern>
      <BeginPattern><![CDATA[</A>]]></BeginPattern>
      <EndPattern><![CDATA[<BR]]></EndPattern>
     </LocatorPattern></Locator></Mapping>
    </Item>
   </Record>
  </Region>
  <Section Name="England"> ... ... </Section>
  <Section Name="Northern Ireland"> ... ... </Section>
 </Section>
 <Section Name="Business">
 ... ...
</Wiccap>
```

Figure 4. XML representation of the global logical view of the BBC Online News site in Figure 3. Experienced users could specify what information to extract and how to extract it using this view.

extraction rules, whereas we use the first set purely to specify the logical structure. The first set is therefore independent of language-specific details. In fact, the extraction patterns often consist only of plain HTML tags, which do not involve language-dependent issues. Thus, even the second set of elements rarely involves language encoding. This is also an important difference between extracting information from the Web and extracting it from free text, which requires natural-language processing techniques and is thus language dependent.

Visual Utility Tools

The mapping wizard tool automates the process of creating the global logical views. A supervised wrapper-generation tool (in Web information-extraction terminology) interactively helps users define what to extract from the Web site, how to extract the data, and how to organize it into meaningful logical views.

The mapping wizard implements a set of GUI-based utility tools to accelerate the global logical view creation process (a more detailed discussion is available elsewhere[3]). These tools automatically derive extraction rules (that is, the second set of WDM elements) for each data item, relieving users of having to manually study HTML sources to decipher the rules. Most tools operate on the Web site's HTML browser view, making it easy to specify the target data. When used together, the utility tools reduce the time required to generate a global logical view for a given Web site.

Network Extraction Agent

Neat helps ordinary users customize Web view parameters and manage extraction jobs that retrieve data from Web sites based on (partially) personalized views. The simple user interface, shown in Figure 5 (next page), is easy for users with little technical knowledge of information extraction to use. Using Neat with Wipap produces the final personalized Web views.

A user creates a Web view by first selecting a global view from a list of those available for different Web sites. The user then configures the following parameters:

- *Scope*. Neat presents the selected view in a tree-like structure to let users specify which

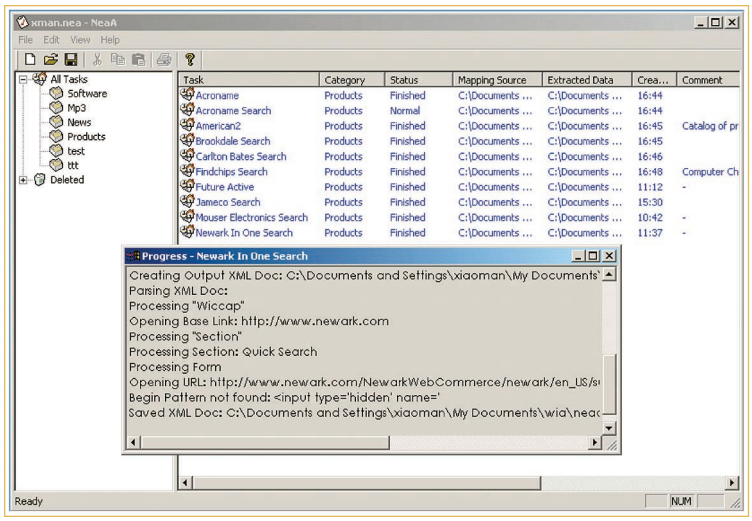

Figure 5. Network extraction agent. The left panel organizes personalized views into categories and the right panel displays the views in the current category. The progress of information extraction by the agent is shown in the pop-up progress window.

parts of the logical view are to be extracted. For example, from the logical view of the BBC Online News site, users can indicate an interest in only the sports news by selecting the subtree rooted at the "Sports News" node. When extracting information, the agent takes only the selected parts of the global view.

- *Filtering.* Neat filters partial views based on different conditions. The user sets up "condition–action" pairs to specify actions (typically removing some content) to be taken in response to given conditions. Examples of the filtering condition include, "Keep only news article items with the word 'World Cup' in the `title` or `description`," or "Remove all product items from Dell with a value of more than US$1,000 in the `price` attribute."
- *Incremental updating.* Users might expect to be informed only of newly published papers in digital libraries or to see only "new" articles at a newspaper Web site. This parameter lets users customize the Web view to incrementally update only when the original Web site changes. This feature is particularly useful for sites with regularly updated lists of records (such as news sites).
- *Consolidation.* This parameter lets users integrate information from multiple global views describing similar or related content into a single Web view. For example, if the user has personalized Web views for both the *Washington Post* and CNN Web sites, the headline news would likely cover similar events even though the article titles might not match.
- *Schedule.* Different sites update data at different rates. For example, newspaper Web sites might be updated once per day, whereas stock information sites might be updated every minute. To ensure a Web view's freshness, users can indicate when and how frequently updates should be performed, similar to most scheduling software, such as the Microsoft Task Scheduler.

A step-by-step wizard guides users to specify each of these parameters using point-and-click actions. All parameters are optional. In the extreme case, when a user omits all parameters, the resulting Web view is identical to the original global view and requires the user to manually initiate extraction.

The View Customization Language is a declarative language for representing the parameters internally. VCL comprises a series of operators, such as `keep` and `delete`, that describe how to incrementally transform global views into personalized Web views. Figure 6 shows an example VCL program. Each operator defined in VCL corresponds to a parameter described earlier; thus VCL provides the exact expressive power required to perform view transformation. Users need not be aware of VCL's syntax because front-end GUI tools shield them from the VCL representations. When constructing an instance of a Web view, the extraction agent interprets and executes the corresponding VCL rules to extract and transform information.

The process of specifying the parameters does not depend on the original Web sites' language. The global logical view's structure is independent of the language encoding. Because users at this step only see and operate on the view's logical structure, the language-specific details are naturally hidden from them.

Wipap Presentation Toolkit

The Web views created by Neat cannot be considered fully personalized if the content's presentation does not match user preferences. Wipap lets users further customize their views by incorporating information about how to present the views (using a presentation template), and when to present them (using a presentation schedule). It also shows the views according to defined parameters. Wipap finalizes the personalized Web views so that users can consume them.

Setting the template parameter simply involves

selecting a template from a list of predefined presentation templates. The current implementation uses Macromedia Flash to display information because it is flexible and displays content in an animated and appealing fashion.

A calendar-styled *program wizard* lets users conveniently specify when each Web view will be presented onscreen and for how long, much like adding an appointment into a Microsoft Outlook calendar. Users can configure the Web view to recur at any interval. This presentation schedule, which resembles a TV program schedule, is intuitive and familiar to most users.

Because these two parameters are relatively simple, we consider the task of representing them as a straightforward implementation issue. Thus, we do not develop a formalism for it.

The Wipap presentation toolkit is the most flexible layer in the Wiccap architecture. Although we present the information using Flash, many different implementations of the third layer are possible — from simple HTML Web pages, to a well-organized restructured Web site, to special-purpose client-side applications. We could incorporate all these implementations as long as they allow users to configure the presentation and scheduling parameters.

Figure 7 shows a snapshot of the Wipap system at runtime. The Wipap GUI's design criteria are quite similar to Neat's, making the application user-friendly and intuitive. The Wipap interface is similar to Windows Media Player. On the left side, the list of frequently used functions is available as a set of buttons. The right side consists of the main panel for showing the active Web view's content (in the center) and the view's flattened skeleton (on the right, below the Back button). On the top are two shortcuts to allow users to change the presentation templates and the active Web view.

```
create webview BBCViewWorldTechNews as
    keep only /Wiccap/Section[1] and /Wiccap/Section[5]
    from BBCGlobalView
create webview BBCViewNoWar as
    delete /Wiccap/Section/Region/Record
        where ./Item[@Type="Title"] contains "iraq"
    delete /Wiccap/Section/Region/Record
        where ./Item[@Type="Description"] contains "iraq"
    from BBCViewWordTechNews
create webview BBCViewIncUpd as
    incupd /Wiccap/Section/Region/Record
        using key ./Item[@Type="Title"]
    from BBCViewNoWar
create webview BBCViewSchedule as
    update start at 08:30 on 30/06/2003 4.8 repeat every 1 day
    from BBCViewIncUpd
```

Figure 6. A View Customization Language program for a partial Web view. Wiccap uses VCL to represent customization parameters internally.

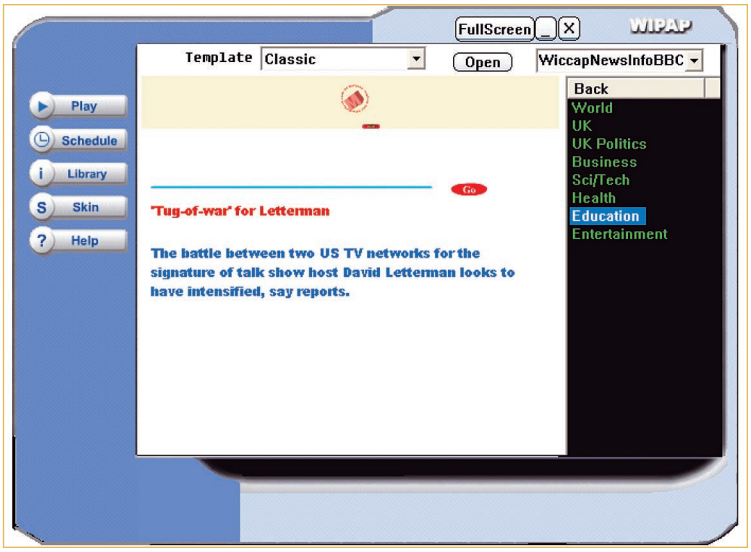

Figure 7. The Wipap system at runtime. The buttons on the left correspond to frequently used functions, the center panel shows the active view's content, and the list on the right shows the view's skeleton.

User Experience

Internal users have created personalized Web views using the Wiccap system for several Web sites (in English and Chinese) of different genres, including online newspapers, online bookstores, digital libraries, product catalogs, and even for personal account information in our university library. Users generally state that they can create their own Web views using the tools without much difficulty and that they are by and large satisfied with the alternate presentation styles offered by the presentation toolkit. In particular, users found the system's incremental updating and scheduling features useful. They also reported that the mapping wizard's learning curve is still relatively steep. We are, in general, satisfied with these preliminary results. However, we would also like to carry out a formal evaluation of the system's usability when the development reaches a more stable stage.

Related Work in Personalized Web Views

The personalized Web views of the WWW Information Collection, Collaging, and Programming (Wiccap) system offer more flexibility and functionality than the personalization services actively provided by the information providers' Web sites. Wiccap also differs from traditional third-party Web-based personalization tools because it is client-based and requires no cooperation from the server side.

Web Montage[1] is a server-side system that automatically generates a start page with links and content by mining users' routine browsing patterns. It assumes the Web server's access log is available, which is usually not the case with uncooperative Web sites. WebViews[2] lets users create Web views to extract a fragment of a Web page using a VCR-style recording component. In Wiccap, a personalized Web view models an entire Web site and offers more flexibility to customize the views. It also has postprocessing parameters lacking in both systems.

Wiccap is also related to Web information extraction systems such as RoadRunner and W4F,[3,4] which researchers frequently refer to as "wrappers" or "wrapper-generation systems." Laender and colleagues briefly survey such systems.[5] These systems lack important parameters such as postprocessing and information presentation, making them incomplete as end-user systems. Nevertheless, some systems' wrapper-generation modules (for example, Xwrap,[6] Lixto,[7] and DEByE[8]) offer interactive user interfaces that help users generate wrappers. These modules could replace Wiccap's first layer to address users' diverse needs.

References

1. C.R. Anderson and E. Horvitz, "Web Montage: A Dynamic Personalized Start Page," *Proc. 11th Int'l World Wide Web Conf.* (WWW11), ACM Press, 2002, pp. 704–712.
2. J. Freire, B. Kumar, and D. Lieuwen, "WebViews: Accessing Personalized Web Content and Services," *Proc. 10th Int'l World Wide Web Conf.* (WWW10), ACM Press, 2001, pp. 576–586.
3. V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. 27th Int'l Conf. Very Large Data Bases* (VLDB '01), Morgan Kaufmann, 2001, pp. 109–118.
4. A. Sahuguet and F. Azavant, "Building Intelligent Web Applications Using Lightweight Wrappers," *Data & Knowledge Eng.*, vol. 36, no. 3, 2001, p. 283–316.
5. A.H.F. Laender et al., "A Brief Survey of Web Data Extraction Tools," *ACM SIGMOD Record*, vol. 31, no 2, 2002, pp. 84–93.
6. L. Liu, C. Pu, and W. Han, "XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources," *Proc. 16th Int'l Conf. Data Eng.* (ICDE '00), IEEE CS, 2000, pp. 611–621.
7. R. Baumgartner, S. Flesca, and G. Gottlob, "Visual Web Information Extraction with Lixto," *Proc. 27th Int'l Conf. Very Large Data Bases* (VLDB '01), Morgan Kaufmann, 2001, pp. 119–128.
8. A.H.F. Laender, B.A. Ribeiro-Neto, and A.S. da Silva, "DEByE: Data Extraction by Example." *Data & Knowledge Eng.*, vol. 40, no. 2, 2002, pp. 121–154.

As part of our future work, we plan to develop mechanisms for maintaining global logical views in the face of change. This might involve change detection at the source Web site and change propagation to individual Web views that have been created. Support for integrating views across multiple Web sites is another important research area.

References

1. B. Adelberg, "NoDoSE: A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents," *Proc. ACM SIGMOD Int'l Conf. Management of Data* (SIGMOD '98), ACM Press, 1998, pp. 283–294.
2. J. Freire, B. Kumar, and D. Lieuwen, "WebViews: Accessing Personalized Web Content and Services," *Proc. 10th Int'l World Wide Web Conf.* (WWW10), ACM Press, 2001, pp. 576–586.
3. Z. Liu et al., "Towards Building Logical Views of Websites," *Data & Knowledge Eng.*, vol. 49, no. 2, 2004, pp. 197–222.

Zehua Liu is a PhD candidate in the School of Computer Engineering at the Nanyang Technological University, Singapore, where he obtained his BASc. His research interests include Web information extraction and retrieval, digital libraries, and databases. Contact him at liuzh@pmail.ntu.edu.sg.

Wee Keong Ng is an associate professor in the School of Computer Engineering at the Nanyang Technological University and director of the Center of Advanced Information Systems, which is affiliated with the School of Computer Engineering. His research interests include Web warehousing, information extraction, electronic commerce, and data mining. Ng has MSc and PhD degrees from the University of Michigan. He is a member of the ACM and the IEEE Computer Society. Contact him at awkng@ntu.edu.sg.

Ee Peng Lim is an associate professor in the School of Computer Engineering, Nanyang Technological University. His research interests include Web warehousing, digital libraries, and database integration. He has a PhD from the University of Minnesota. Lim is an associate editor of the *ACM Transactions on Information Systems* (TOIS) and an editorial review board member of the *Journal of Database Management*. Contact him at aseplim@ntu.edu.sg.