Int J Softw Tools Technol Transfer (2009) 11:453–468
DOI 10.1007/s10009-009-0124-3

SPECIAL SECTION ON WEB SYTEMS EVOLUTION

Consistent Web site updating based on bidirectional transformation

Keisuke Nakano · Zhenjiang Hu · Masato Takeichi

Published online: 3 November 2009

Abstract A transformation-based Web site can keep the contents of a Web site consistent by furnishing a single database and a set of transformation programs, each generating a Web page from the database. However, when someone notices an error or stale content on a Web page in this style of Web site construction, the Web site maintainer must access a possibly huge database to update the corresponding content. In this paper, we propose a new approach to Web site construction based on bidirectional transformation, and report our design and implementation of a practical updating system called Vu-X. We bring the idea of bidirectional transformation to Web site construction, describing not only a forward transformation for generating Web pages from the database but also a backward transformation for reflecting modifications on the Web pages to the database. By use of the bidirectional transformation language Bi-X, we can obtain both transformations only by specifying a forward transformation. Our Vu-X system is implemented as a Web server built upon the Bi-X transformation engine, which can keep the content of Web sites consistent by updating Web pages in WYSIWYG style on Web browsers.

K. Nakano (✉)
The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan
e-mail: ksk@cs.uec.ac.jp

Z. Hu
GRACE Center, National Institute of Informatics,
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
e-mail: hu@nii.ac.jp

M. Takeichi
Department of Mathematical Informatics, University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
e-mail: takeichi@mist.i.u-tokyo.ac.jp

Keywords Web site maintenance · XML transformation · Bidirectional transformation · Web site generation · HTML editor

1 Introduction

Maintaining a Web site is costly and time-consuming [18], which requires consistent and frequent updating of the contents to maintain their freshness and quality. The maintenance problem generally covers any type of update to a Web site, including recognizing its structure and modifying its contents. A more specific problem has to do with replicated information: it is onerous for Web site administrators to avoid inconsistency between all Web pages provided on their own Web site because some of the pages may contain the same information, which should be synchronized.

A natural solution to this problem is to prepare a single database that stores the information for all Web pages on the Web site, and to consider a Web site as a set of views that are generated from some existing raw data (database) [4,11,12] through transformation programs. We call this *transformation-based Web site construction*. This helps to easily keep all Web pages consistent since all pages are generated from the same database.

Figure 1 shows a simple instance of such transformation-based Web site construction. The database is usually constructed in XML format, which is convenient for describing structured information. Transformation programs $p_1, \ldots, p_n$ are used to generate an HTML source corresponding to a Web page from the XML data. Typically, these programs are written in XSLT [26].

Transformation-based Web site construction, however, is not as good at frequently updating its contents as simple Web

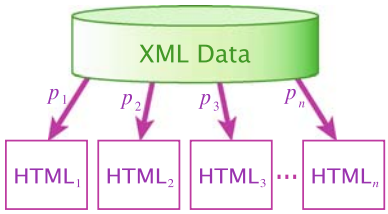


Fig. 1 Transformation-based Web site construction

site construction that consists of a set of independent HTML files.

When someone points out an error or stale content on a page at the Web site, it is not easy to find which part of the database should be updated, particularly when the database is very huge or when the maintainer is not the person who designed the database. It would be nice if we could correct errors, update content, or change the layout just by editing the HTML view of the Web page, which results from the transformation, instead of accessing the database to update it. Furthermore, it would be ideal if we could edit Web pages on Web browsers in WYSIWYG style.

This paper proposes a new approach to solving the Web maintenance problem based on *bidirectional transformation*, together with a practical updating system called Vu-X (pronounce as "view X"). We divide the problem into two parts, i.e., how to edit Web pages on Web browsers in WYSIWYG style and how to reflect modifications of the HTML source to the original database. The former is attained by adding JavaScript to the HTML source.

The latter, which is more difficult, is accomplished by a novel use of *bidirectional transformation* [13,16,20], which is a new technique for synchronizing data.

Bidirectional transformation involves a pair of transformations: forward transformation maps one data structure called a source onto another called a view, and backward transformation reflects changes in the view to the source. Bidirectional transformation has many practical applications including the synchronization of replicated data in different formats [13], presentation-oriented structured documents development [16], interactive user interface design [21], coupled software transformation [17], and the well-known mechanism for *view updating*, which has been intensively studied in the database community [5,9,14,15,19].

In our approach shown in Fig. 2, forward transformation p_i transforms XML data into an HTML source, while backward transformation $\tilde{p}_i$ reflects a modification of the HTML source to the database. Since the HTML source generated by forward transformation generally does not have sufficient information to construct the corresponding XML data, backward transformation takes not only an updated HTML source but also the original XML data to generate the updated XML data. These two transformations, p_i and $\tilde{p}_i$, should be consistent with each other. We chose a bidirectional XML

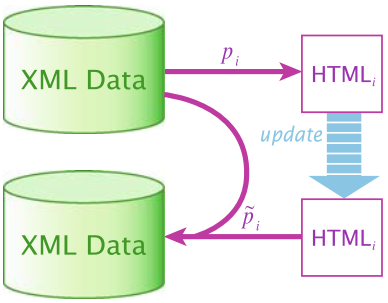


Fig. 2 Bidirectional transformation between XML data and HTML sources

transformation language Bi-X [20] to specify the bidirectional transformation. One advantage of using Bi-X is that a program written in the Bi-X language only describes forward transformation and the corresponding backward transformation is automatically derived.

We have implemented the Vu-X system[1], which facilitates the maintenance of a Web site based on bidirectional transformation. The Vu-X system has five main features:

1. It enables us to easily maintain consistency between multiple Web pages by means of transformation-based Web site construction.
2. It enables us to edit a generated HTML source itself by means of bidirectional transformation.
3. It does not force us to specify two transformations to attain a bidirectional transformation using the Bi-X language.
4. It enables us to edit both HTML sources and Bi-X programs in WYSIWYG style.
5. It does not force to install any special software to use the Vu-X system because it is provided as a Web application.

The remainder of this paper is organized as follows. We start by giving an overview of the Vu-X system with some examples of practical situations in Sect. 2. After summarizing the basic ideas of bidirectional transformation and the bidirectional transformation language Bi-X in Sect. 3, we explain the architecture and the implementation of our Vu-X Web site updating system in Sect. 4. In Sect. 5, some experiments on Web site construction using our system are shown. We discuss several issues on the current implementation in Sect. 6. We summarize related work in Sect. 7, and conclude the paper in Sect. 8.

The result of this paper, but only with informal explanation for bidirectional transformation and the Vu-X system,

[1] The Vu-X system can be played with Web browsers such as Microsoft Internet Explorer and Firefox and its URL is http://www.psdlab.org/vux/start.html. You may login with your email address for ID and no password is required to try our system while possible operations are limited for security.

was presented at the 10th IEEE international symposium on web site evolution [23]. In addition to the result, this paper discusses experimental result, possibility and limitation on our implementation.

2 Overview of Vu-X system

This section explains how users maintain a Web site with the Vu-X system, giving some examples of practical situations. We summarize features of the Vu-X system after that.

2.1 Web site updating through the Vu-X system

The Vu-X system is implemented as a Web application that runs on Internet Explorer or Firefox. Users are not required to install special software to employ the Vu-X system. Users start the Vu-X system for updating their own Web sites by logging in with a correct password. Only allowed persons can update the Web sites. The XML data and Bi-X programs for the Web site are registered through the Vu-X system. After logging in, they open a small window to select the name of the XML data and Bi-X program that corresponds to the Web page they want to edit. Here, users have to choose either of two modes, content updating or code editing. In the former, they can update information in the XML data through editing the corresponding HTML source. In the latter mode, they can edit the Bi-X program or create a new Bi-X program from scratch. Both modes provides a WYSIWYG editor that runs on Web browsers.

We generally employ the content updating mode to edit the content of the Web site and the code editing mode to change or define the layout of Web pages.

When users select the name of the XML data and Bi-X program, the corresponding Web page is loaded on their Web browser as seen in Fig. 3 no matter which mode is chosen. The Web page to be updated is displayed in the lower right portion of the browser's window, shown with plain white background in the figure.

The HTML source of the Web page is generated from the designated XML data and Bi-X program. It contains a JavaScript code so that users can edit it on the Web browser and reflect the change to the XML data or Bi-X program. Let us look at some scenarios involving updating Web pages in content updating or code editing mode.

2.1.1 Scenario: changing the name of laboratory

Assume that users want to change the name of laboratory that occurs many times on a single Web page as seen in Fig. 4 which is zoomed from Fig. 3. The name may occur at the footer of all Web pages on the Web site. If all occurrences are generated from the same part of the XML data, it suffices

Fig. 3 Screenshot of Vu-X system

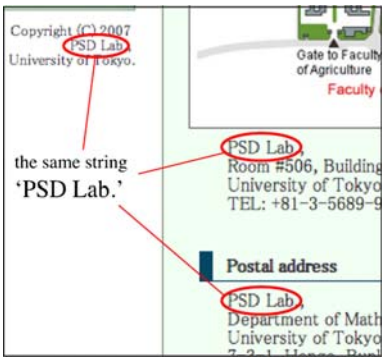


Fig. 4 Three occurrences of name of laboratory

for users to click and edit just one of the occurrences on the Web browser. Then, the other occurrences of the string are automatically updated because the corresponding part of the XML data is updated through backward transformation and forward transformation modifies all occurrences of the string. This editing is done in the content updating mode because it involves updates of the XML data.

2.1.2 Scenario: updating publication list

Assume that users want to update a list of publications sorted by year in descending order. The XML data contain the information on all publications including the title, author(s), and year of publication for all entries. The Bi-X program describes a (forward) transformation from the XML data to an HTML source for the Web page for the publication list. The list is represented as a `ul` component in the HTML

source. When users click it, a small window pops up for them to edit the items in the list (Fig. 5). The window contains a list of items in the `ul` component and several buttons for editing functions and reflecting the updates. The Vu-X system supplies several editing functions for a `ul` component, i.e., modification to an item, deletion of an item, and insertion of a new item. Users can also copy and paste an existing item. When they want to modify the year of a publication, they click the publication item to open another editing window.

Since the item consists of multiple HTML components as

```
<li>
    <span>K. Nakano</span>
    <span>Consistent Web Page Updating</span>
    <span>CCC Workshop</span>
    <span>2006</span>
</li>,
```

the small editing window that has opened graphically shows these components, each of which opens another editing window for the HTML component by clicking it.

Users click the component `<span>2006</span>`, which represents the year of publication, to modify the text `2006` into `2005`. After the modification, they click the "OK" button to close each window and reflect the modification in the previous small window. When all small windows are closed, the Web page on the Web browser is reloaded. Interestingly, the occurrence of the publication in the list will be automatically moved because the year of publication is changed. This is, as far as we are aware, one of the unique functions that cannot be found in other Web site maintenance systems. This editing is done in the content updating mode because it involves update of the XML data.

2.1.3 Scenario: changing layout of web page

Assume that users want to change the layout of a Web page from a `table`-based style into a `div`-based one. In a transformation-based Web site construction, this kind of information about layout styles is specified by the transformation program but not the XML data. Therefore, what users should do in this situation is to modify the transformation program. The Vu-X system provides a function to edit Bi-X programs with a graphical user interface. Users click the `table` component then the system invokes the Bi-X *code builder*, which is provided as a Java applet, so that they can modify a fragment of the Bi-X program that generates the `table` component from the XML data. The detail will be explained in Sect. 4.

Since the Vu-X system shows how the Web page is generated from the actual XML data when editing a Bi-X program, users can easily update the layout of the Web page. This editing is done in the code editing mode because it involves update of the Bi-X program.

2.2 Features of the Vu-X system

The Vu-X system supports a Web site construction based on bidirectional transformation. We summarize several advantages of the Vu-X system in updating Web sites.

As the first advantage of the Vu-X system, users can easily maintain consistency between multiple Web pages on the Web site. The Vu-X system employs an extension of transformation-based Web site construction furnished with XML data and Bi-X programs, each of which generates a Web page from the XML data. Since every piece of information shared by multiple Web pages occurs once in the XML data, all occurrences of this information in Web pages are definitely synchronized.

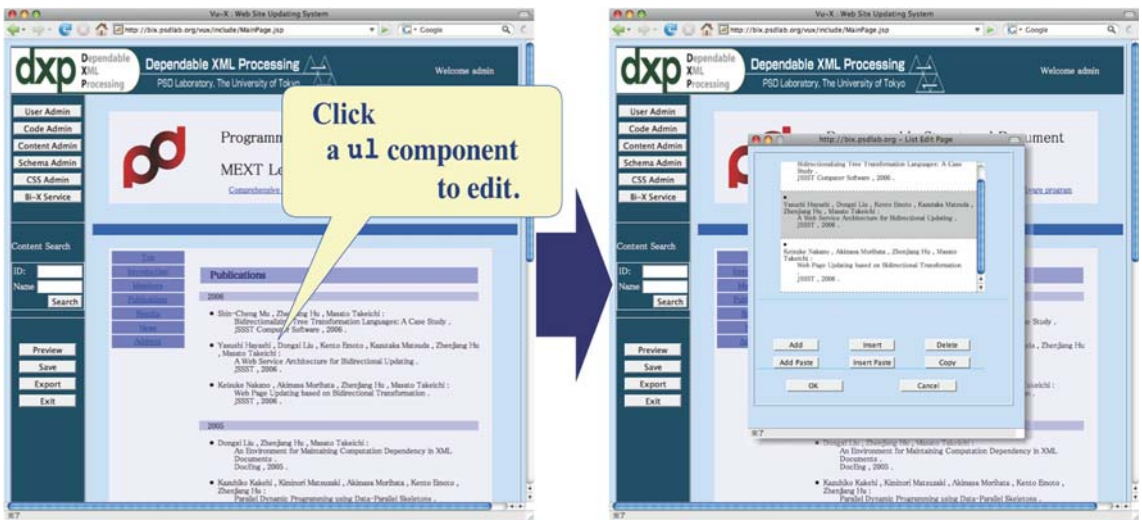


Fig. 5 Editing a `ul` component on the Web browser

Second, users can edit the generated HTML source itself based on bidirectional transformation. Modifications to the HTML source are consequently reflected in the XML data by the backward transformation. Since visitors to the Web site see the result of the forward transformation, the updated information is consequently reflected in all Web pages. The generated HTML source is modified with a WYSIWYG editor provided by the Vu-X system as the content updating mode.

Third, users do not have to specify both forward and backward transformations to attain a bidirectional transformation thanks to the bidirectional transformation language Bi-X. It suffices to only specify forward transformation. The corresponding backward transformation is automatically derived. All transformation programs in the Vu-X system that generate Web pages are written in Bi-X.

Fourth, users can easily generate and edit a Bi-X program with a WYSIWYG editor provided by the Vu-X system as the code editing mode. They construct a Bi-X program step by step by discerning the transformation result. For programmers who are familiar with the XQuery language, the translation tool from an XQuery program to a Bi-X program [20] is useful. As another option, users may reuse Bi-X programs which we wrote for Web sites of several kinds of organization as shown in Sect. 5.

Finally, users do not have to install any special software to work with the Vu-X system because it is implemented as a Web application with JavaScript and Apache Tomcat. The WYSIWYG editor in both content updating and code editing modes runs on standard Web browsers. We have tested the performance with Firefox and Microsoft Internet Explorer.

3 Bidirectional transformation and Bi-X language

Bidirectional transformation is a basis for our Vu-X Web site updating system. This section summarizes the general concept of bidirectional transformation and gives a brief introduction to a bidirectional XML transformation language, Bi-X, which is employed in the current Vu-X system.

3.1 Bidirectional transformation

Bidirectional transformation [13], originating from the view-updating technique in the database community [5,9,14,15,19], involves a pair of transformations between a *source* and its *view*: the first transformation, called *forward transformation*, maps sources to views and is used to reflect changes in a source in its view. The second transformation, called *backward transformation*, maps views to sources and is used to reflect changes in a view in its source. Since a view generally has less information than its source, backward transformation takes not only a changed view but also the original source to generate a source corresponding to the new view.

Forward and backward transformation should be consistent in order to make a bidirectional transformation *well-behaved* in the following sense. Let $fwd : S \rightarrow V$ and $bwd : V \times S \rightarrow S$ be forward and backward transformation, respectively, where S is a set of sources and V is a set of views. The well-behaved bidirectional transformation must satisfy two properties, *acceptability* and *consistency* [13,16,20]. Acceptability indicates that no update on the view implies no update on the source, i.e., $bwd\,(fwd(s), s) = s$ for any source $s \in S$. When a view generated from a source by forward transformation is not changed, backward transformation must generate the same source for the view and the source. Consistency indicates that permitted updates on the view should be reflected to the source, i.e., $fwd(bwd(v, s)) = v$ for any source $s \in S$ and any view $v \in V$. Forward transformation generates the same updated view from the reflected source.

It is difficult to describe forward and backward transformation programs for a well-behaved bidirectional transformation because they must be consistent. In general we need to simultaneously maintain forward and backward transformation so that they are consistent. Few bidirectional transformation languages, such as Boomerang [13,7], X/Inv [16], and Bi-X [20], have been proposed to support easy bidirectional programming. Although our framework does not depend on choice of bidirectional language, we chose Bi-X (as will be discussed later in the next subsection).

The main reason of our choice is that the Bi-X language allows us to write a transformation program that deals with duplication of a part of source, which makes a well-behaved bidirectional transformation difficult. The reader may imagine the difficulty by considering a situation where one of duplicated parts is updated on a view.

3.2 Bi-X: Bidirectional transformation language

The Bi-X bidirectional transformation language has been presented by Liu et al. [20]. A program written in Bi-X specifies a bidirectional transformation between two XML documents (or XML fragments). The main feature of Bi-X is that all programs have two semantics corresponding to forward and backward transformation. It suffices to write a Bi-X program only for forward transformation. Programmers do not need to consider the consistency between forward and backward transformation to specify bidirectional transformation. In this paper, we give just a brief introduction of the Bi-X language and two examples of Bi-X programs. See [20] for details on the Bi-X language and its theoretical background.

A Bi-X program is written in XML format. The syntax of the Bi-X language is given by the following grammar:

$P :=$ <xid/>
| <xchild/>
| <xconst>X</xconst>
| <xseq>$P \ldots P$</xseq>
| <xmap>P</xmap>
| <xpar>$P \ldots P$</xpar>
| <xchcont>$P \ldots P$</xchcont>
| <xmakeatt>$P\ P$</xmakeatt>
| <xrename>P</xrename>
| <xif>$P\ P\ P$</xif>
| <xwithtag>s</xwithtag>
| <xlet><var>v</var>P</xlet>
| <xvar>v</xvar>

where v is a name of variable, X is an arbitrary XML fragment (including a string), and s is a name of element. A Bi-X program sequentially processes a (current) input XML document to get (a list of) output XML documents step by step. Thus every construct in the Bi-X language represents a transformation from the current input (which may be the previous output) to an output (which may be the next input). The construct <xid/> means identity transformation. <xchild/> extracts a list of all children of the current input. <xconst>X</xconst> outputs X for any input. <xseq>$P_1 \ldots P_n$</xseq> sequentially applies $P_1, \ldots, P_n$ to the current input. <xmap>P</xmap> can take a list of XML elements as its input. This construct applies P to each element in the input and returns a list of the outputs. <xpar>$P_1 \ldots P_n$</xpar> concatenates all outputs of $P_1, \ldots, P_n$ as sibling nodes. <xchcont>$P_1 \ldots P_n$</xchcont> replaces the child of the current input with a list of all outputs of $P_1, \ldots, P_n$. <xmakeatt>$P_1\ P_2$</xmakeatt> adds an attribute whose key and value are the output of P_1 and P_2, respectively, both of which should be text strings. <xrename>P</xrename> changes the name of the element into the output of P, which should be a text string. <xif>$P_1\ P_2\ P_3$</xif> applies P_2 to the current input if the output by P_1 is not empty, otherwise it applies P_3. Typically, we use <xwithtag>s</xwithtag> for P_1, which returns the input itself if the name of the element is s, otherwise returns an empty output. <xlet><var>v</var>P</xlet> binds the current input to variable v and the variable can be referred in following process P by the construct <xvar>v</xvar>. Binding variable is useful mainly in two cases where the bound content is used more than once and where the following construct loses the input information such as the <xconst> construct.

The basic idea of the design of the Bi-X language is to give forward and backward semantics for each construct in structural way similar to an inversion algorithm in [16]. Let fwd_P and bwd_P denote forward and backward transformation for a Bi-X program P, respectively. For example, for a Bi-X program $P =$ <xseq>$Q\ R$</xseq>, fwd_P and bwd_P are defined by

$$fwd_P(s) = fwd_R(fwd_Q(s))$$

$$bwd_P(v, s) = bwd_Q(bwd_R(v, fwd_Q(s)), s)$$

using fwd_Q, fwd_R, bwd_Q, and bwd_R. See [20] for the detailed updating algorithm on the Bi-X language. The important point is that the backward transformation is always automatically obtained as long as the (forward) transformation is given as the Bi-X program.

Figure 6 shows an example of a Bi-X program. The forward transformation by this program takes an XML document which includes content of a paper and returns `body` element of an HTML document with a table of contents. The input XML data is assumed to form as shown in Fig. 7.

The program can informally be read as follows: sequentially transform (<xseq>...) the input data by first obtaining a list of all their children (<xchild/>). Bind the list to a variable `children` (<xlet>...) to refer it later. Next create a `body` element (<xconst>...) and add all outputs of the following processes as its children (<xchcont>...). As the first child, put an `h1` element which is obtained by loading the content of variable `children` (<xvar>...), extracting an element whose tag name is `title` from the list (<xmap>...), and changing the tag name into `h1` (<xrename>...). As the second child, put a `ul` element which is obtained by creating a `ul` (<xconst>...) and adding a list of `li` elements. Each `li` element is obtained by loading the content of variable `children` (<xvar>...) and extracting all `author` elements as `li` elements (<xmap>...). As the third child, put an `h2` element with a child text `Abstract` (<xconst>...). As the fourth child, put a `p` element obtained by extracting an `abstract` element as `p` element (<xmap>...) from the list in the variable `children` (<xvar>...). As the fifth child, put an `h2` element with a child text `Table of Contents` (<xconst>...). As the sixth child, put a `ul` element which contains a list of `li` elements each of which is obtained by extracting a `title` element under `section` as `li` element (<xchcont>...). As the last child (children), put an `h2` element with a title and a `p` element with a paragraph by loading the content of variable `children` (<xvar>...), changing the tag name `title` and `para` into `h2` and `p`, respectively, and removing the other elements.

For example, when an XML document S shown in Fig. 7 is offered as an input of the forward transformation of the program, output V is an HTML fragment in Fig. 8.

Assume that a new author is inserted into the first `ul` element of V as

```
<xseq>
 <xchild/>
 <xlet><var>children</var><xseq>
  <xconst><body/></xconst>
  <xchcont>
   <xseq>
    <xvar>children</xvar>
    <xmap><xwithtag>title</xwithtag></xmap>
    <xrename><xconst>h1</xconst></xrename>
   </xseq>
   <xseq>
    <xconst><ul/></xconst>
    <xchcont><xseq>
     <xvar>children</xvar>
     <xmap><xseq><xwithtag>author</xwithtag>
      <xrename><xconst>li</xconst></xrename>
     </xseq></xmap>
    </xseq></xchcont>
   </xseq>
   <xconst><h2>Abstract</h2></xconst>
   <xseq>
    <xvar>children</xvar>
    <xmap><xwithtag>abstract</xwithtag></xmap>
    <xrename><xconst>p</xconst></xrename>
   </xseq>
   <xconst><h2>Table of Contents</h2></xconst>
   <xseq>
    <xconst><ul/></xconst>
    <xchcont><xseq>
     <xvar>children</xvar>
     <xmap><xseq><xwithtag>section</xwithtag>
      <xchild/>
      <xmap><xseq><xwithtag>title</xwithtag>
       <xrename><xconst>li</xconst></xrename>
      </xseq></xmap>
     </xseq></xmap>
    </xseq></xchcont>
   </xseq>
   <xseq>
    <xvar>children</xvar>
    <xmap><xseq>
     <xwithtag>section</xwithtag>
     <xchild/>
     <xmap><xseq>
      <xif><xwithtag>title</xwithtag>
       <xrename><xconst>h2</xconst></xrename>
       <xseq><xwithtag>para</xwithtag>
        <xrename><xconst>p</xconst></xrename>
       </xseq>
      </xif>
     </xseq></xmap>
    </xseq></xmap>
   </xseq>
  </xchcont>
 </xseq></xlet>
</xseq>
```

Fig. 6 A Bi-X program

```
<paper>
  <title>Vu-X system</title>
  <author>Keisuke Nakano</author>
  <author>Zhenjiang Hu</author>
  <abstract>
    Vu-X supports a Web site construction
    based on bidirectional transformation.
  </abstract>
  <section>
    <title>Introduction</title>
    <para>Transformation-based Web site
          should be bidirectional.</para>
  </section>
  <section>
    <title>Vu-X architecture</title>
    <para>Vu-X is based on bidirectional
          transformation.</para>
  </section>
  <section>
    <title>Conclusion</title>
    <para>Vu-X supports consistent Web site
          construction.</para>
  </section>
</paper>
```

Fig. 7 An XML fragment S for a paper

```
<ul>
    <li>Keisuke Nakano</li>
    <li>Zhenjiang Hu</li>
    <li>Masato Takeichi</li>
</ul>.
```

The backward transformation will return an XML document in which the new `author` element is added to the source S as follows.

```
    <author>Keisuke Nakano</author>
    <author>Zhenjiang Hu</author>
    <author>Masato Takeichi</author>
```

According to the backward semantics of the Bi-X language [20], the position of a new element to be inserted depends on the position of the new element on the view and the Bi-X program. If they specify the transformation by the `<xmap>` construct, the order of items should be the same for the source and view. If they specify a kind of sorting transformation by year, the order of items in the source can be arbitrary. In the backward semantics of the Bi-X language, the new item is added to the end of the item list.

Consider another example of updating which involves duplication of the source data. This example illustrates that a Bi-X program can deal with duplication. It is generally difficult to attain bidirectional transformation for a program including duplication because a duplicated view should be synchronized for every update.

In the view V of Fig. 8, a text `Vu-X architecture` appears twice in `li` and `h2` element. If either text of them is changed into `Vu-X system`, the backward transformation of the program will return S in which the corresponding text data in a `title` element is replaced with `Vu-X system`. Applying forward transformation to the XML source again, we will have XML view V in which both texts `Vu-X architecture` are synchronized to be text `Vu-X system`. When multiple updating for duplicated elements conflicts (e.g., two occurrences are independently changed into `Vu-X system` and `Vu-X`), the backward transformation fails according to the backward semantics of the Bi-X language [20].

We show an example involving an insertion which requires an intricate updating. Consider an insertion of a new `li` element (e.g., `<li>Acknowledgment</li>`) to the end of

```
<body>
  <h1>Vu-X system</h1>
  <ul>
    <li>Keisuke Nakano</li>
    <li>Zhenjiang Hu</li>
  </ul>
  <h2>Abstract</h2>
  <p>
    Vu-X supports a Web site construction
    based on bidirectional transformation.
  </p>
  <h2>Table of Contents</h2>
  <ul>
    <li>Introduction</li>
    <li>Vu-X architecture</li>
    <li>Conclusion</li>
  </ul>
  <h2>Introduction</h2>
  <p>Transformation-based Web site
     should be bidirectional.</p>
  <h2>Vu-X architecture</h2>
  <p>Vu-X is based on bidirectional
     transformation.</p>
  <h2>Conclusion</h2>
  <p>Vu-X supports consistent Web site
     construction.</p>
</body>
```

Fig. 8 An HTML fragment V generated by Bi-X program

children of the first `ul` element of V. The backward semantics of Bi-X language tells us that such insertion is possible if the *schema* of the source (input XML data) is specified. Since the schema indicates that every `section` element has two elements `title` and `para` as its children, the insertion will be reflected as

```
<section>
    <title>Acknowledgment</title>
    <para></para>
</section>
```

in the source. See [20] for the detail. In the Vu-X system, users can store an XML data together with its schema to attain this kind of insertion updating.

Even though Bi-X language helps us to describe a consistent bidirectional transformation, it may be still difficult to write a large Bi-X program for practical use. To facilitate Bi-X programming, the Vu-X system provides a graphical user interface to generate a Bi-X program. As the other way to write Bi-X programs, users may employ a translation tool from an XQuery program to a Bi-X program [20].

One may think that a bidirectional transformation can be attained just by relating every element or text in the view to the corresponding part in the source without relying on bidirectionality of the Bi-X language. The relation is useful for reflecting updates on the view in the source. However, it does not suffice to apply to our Web site updating system. For instance, it is difficult to reflect insertion of a new entry to a list on the view where the list is generated by collecting entries from many different places in the source. Users must specify where and how the new entry is inserted to the source. In contrast, the Bi-X language easily copes with this situation using type information of sources that is similar to XML schema. Users do not have to specify additional instructions for the case. The updating mechanism of the Bi-X language is found in [20].

4 Vu-X: web site updating system

The Vu-X system provides an integrated environment to maintain a Web site based on bidirectional transformation. It is an extension of transform-based Web site in Fig. 1, where transformations $p_1, \ldots, p_n$ are bidirectional. This section presents the architecture of the Vu-X system and explains how the two updating modes introduced in Sect. 2 are implemented in the Vu-X system.

4.1 Architecture of Vu-X system

The Vu-X system is implemented as a Web server based on Apache Tomcat so that users can update their own Web site through standard Web browsers, e.g., Firefox and Microsoft Internet Explorer, and thus no special software is required to install it.

Figure 9 outlines the architecture of the Vu-X system. The system consists of five parts: the database of XML data and Bi-X programs, Bi-X processor, editable HTML generator, HTML-embedded Bi-X editor, and Bi-X code builder. We will explain these one by one.

Database of XML data and Bi-X *programs* It stores the XML data and Bi-X programs registered by users. Each XML data or Bi-X program is accessed by authorized users with

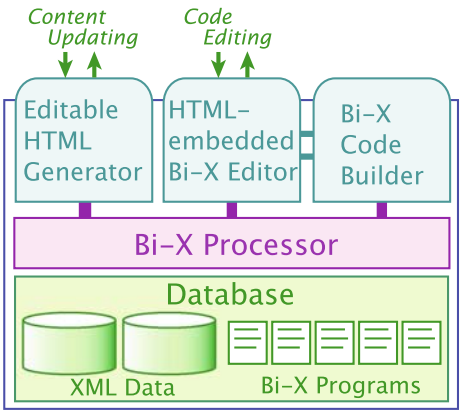


Fig. 9 Architecture of Vu-X system

its identifier and read/write instruction. Note that we do not have to store any HTML sources for Web pages on the Web site. They are generated by the forward transformation of the Bi-X programs from the XML data. This part also stores HTML-embedded Bi-X programs, which will be explained later.

Bi-X *processor* It executes the forward/backward transformation for a given Bi-X program. The forward transformation takes the identifier of the XML data and returns an HTML source. The backward transformation takes the identifiers of the XML data and an updated HTML source and returns updated XML data. This part can be easily replaced by another processor if one wants to specify bidirectional transformation in a bidirectional transformation language other than Bi-X.

Editable HTML generator It adds a JavaScript code to a given HTML source and creates an *editable HTML*, which allows us to edit it on Web browsers. The obtained editable HTML and the original HTML source have the same view on Web browsers, except that users can click a component of the view of the editable HTML on the Web browser to edit the component in a small popped-up window. This part is employed in the content updating mode.

HTML-embedded Bi-X *editor* It provides a WYSIWYG HTML editor that runs on Web browsers. Users can also insert multiple Bi-X programs together with HTML components. This editor can generally create an HTML source that contains Bi-X programs, called an *HTML-embedded* Bi-X *program*, which will be discussed in detail in Sect. 4.3. This part is employed in the code editing mode.

Bi-X *code builder* It provides a graphical user interface to enable Bi-X programs to be easily constructed. Users can write a Bi-X program using actual XML data that is an input of forward transformation. The Bi-X code builder shows a partial result of the transformation for each fragment of the Bi-X program. It is provided as a Java applet and is invoked by clicking an HTML component related to a Bi-X program in the HTML-embedded Bi-X editor. This part is employed in the code editing mode.

4.2 Content updating

Figure 10 outlines an initial action in the content updating mode. Content updating starts with a user's request to the Vu-X system with the identifiers of the XML data and a Bi-X program. The Vu-X system executes forward transformation of the Bi-X program for the designated XML data through the Bi-X processor to obtain a transformation result that is an HTML source. Next, the editable HTML generator of the

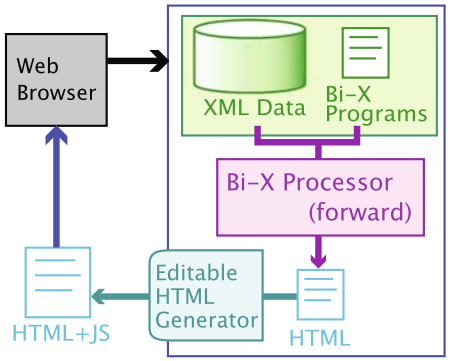


Fig. 10 Initial action in content updating mode

Vu-X system adds a JavaScript code to the HTML source so that users can edit it on their Web browsers. The JavaScript code does not change the layout on the Web browsers.

The JavaScript code plays two roles.

The first is to allow us to edit in WYSIWYG style for several kinds of HTML components. Possible editing depends on the kind of clicked component. For example, a text component allows us to edit its text data and a `ul` component allows us to modify an item on the list, delete an item, or insert a new item as explained in Sect. 2.

The second role of the JavaScript code is to transmit an updated HTML source to the Vu-X system by clicking the "OK" button to finish editing. The transmitted HTML source does not include the added JavaScript code.

Figure 11 shows how the Vu-X system treats the updated HTML transmitted from the Web browser. The updated HTML is passed to the Bi-X processor to execute backward transformation of the Bi-X program with the original XML data to update the XML data in the database. Then, the Vu-X system again calls the Bi-X processor for forward transformation of the Bi-X program with the updated XML data. It generates a new HTML source corresponding to the updated XML data. Note that the HTML source is not always the same as the HTML source updated by users because the edited part may influence the other part in the HTML source where the Bi-X program contains duplication. After forward transformation, the Vu-X system generates the editable HTML source by again adding a JavaScript code to the obtained HTML source. Then, users can edit another component on the Web page to update the content of the XML data.

4.3 Code editing

Like content updating, code editing starts with a user's request to the Vu-X system with the identifiers of the XML data and a Bi-X program (Fig. 12).

In the code editing mode, users edit or create an *HTML-embedded* Bi-X *program* through a graphical user interface using the HTML-embedded Bi-X editor of the Vu-X system.

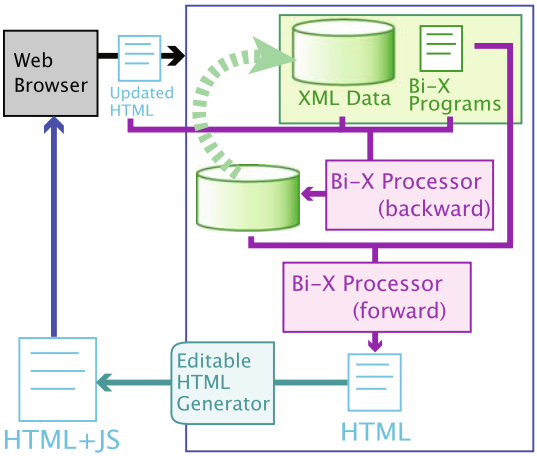


Fig. 11 Update reflection in content updating mode

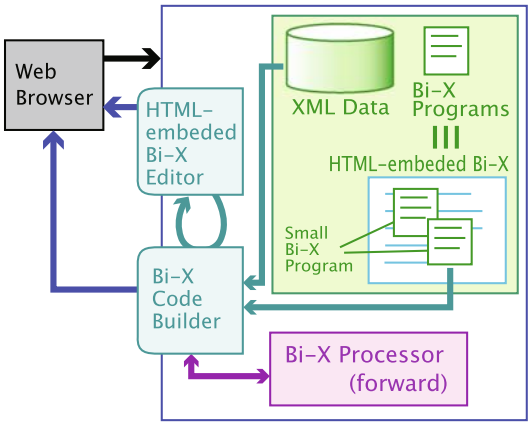


Fig. 12 Code editing mode

An HTML-embedded Bi-X program has a form analogous to PHP or JSP as shown in Fig. 13 where two `bixpro` elements contains Bi-X programs.

An HTML-embedded Bi-X program represents a bidirectional transformation whose forward transformation generates an HTML source obtained by replacing each `bixpro` component in the HTML-embedded Bi-X program with the result of the forward transformation of the Bi-X program in the component. For example, assume that an XML data in Fig. 7 is offered as an input for the HTML-embedded Bi-X program in Fig. 13. The program contains two Bi-X programs which generate HTML fragments

```
<p>
    Vu-X supports a Web site construction
    based on bidirectional transformation.
</p>
```

and

```
<ul>
    <li>Keisuke Nakano</li>
    <li>Zhenjiang Hu</li>
</ul>,
```

respectively. Therefore, this HTML-embedded Bi-X program specifies a transformation which generates an HTML source obtained by replacing two `bixpro` elements with these HTML fragments as shown in Fig. 14.

The Vu-X system provides a transformation tool from an HTML-embedded Bi-X program to a single Bi-X program in order to use it in the content updating mode. Both the Bi-X program and the HTML-embedded Bi-X program are stored as a pair in the database of the Vu-X system.

The code editing mode supplies WYSIWYG HTML editors such as Adobe Dreamweaver [1] and Microsoft Expression Web [10]. There are two differences from the existing WYSIWYG HTML editors. The first is that users can add `bixpro` components to create an HTML-embedded Bi-X program. The Bi-X program in the `bixpro` component is edited by the Bi-X *code builder*, which will be explained later. The second difference is that the editor is implemented as a Web application using JavaScript. Users do not have to install any special applications to edit it.

When a new `bixpro` component is created or the existing `bixpro` component is clicked, the Bi-X code builder shown in Fig. 15 is invoked to edit a Bi-X program to be inserted. The Bi-X code builder is implemented as a Java applet that communicates with a JavaScript code in the HTML-embedded Bi-X editor. The Bi-X code builder receives a Bi-X program in the clicked `bixpro` component from the HTML-embedded Bi-X editor and sends back an edited Bi-X program to the HTML-embedded Bi-X editor.

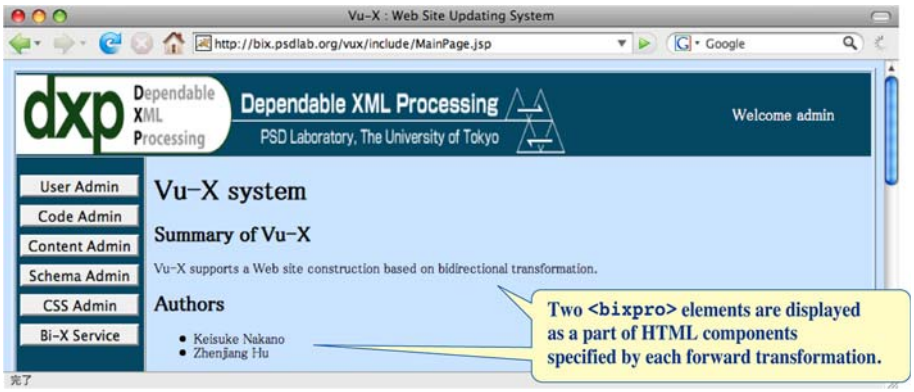
The Bi-X code builder helps users to write a Bi-X program using an actual input XML. It is difficult to write a Bi-X program without an actual input because the Bi-X language requires a *point-free style* programming where intermediate data is implicitly passed from one instruction to the next instruction. Hence, the Bi-X code builder provides an programming environment which requires an actual input XML.

The Bi-X code builder has several buttons and three tree views. Each button corresponds to a construct in the Bi-X language. When the construct requires arguments, the Bi-X code builder opens a prompt window to provide these. The tree view at the left shows an abstract syntax tree of the Bi-X program that users want to edit. There are two cursors colored green and red in the view. The green cursor stands for the current editing position. They can easily place a new Bi-X construct immediately behind the green cursor by clicking the corresponding button. The red cursor represents the previous editing position. The middle tree view shows an XML tree, which is a transformation result at the position designated by the red cursor in the Bi-X program. The tree view at the right shows an XML tree, which is a transformation result at the position designated by the green cursor in the Bi-X program. In Fig. 15, a part of the Bi-X program from the red cursor to the green cursor at the left tree view

Fig. 13 An HTML-embedded Bi-X program with two `bixpro` elements

```
<html>
  <head><title>Vu-X</title></head>
  <body>
    <h1>Vu-X system</h1>
    <h2>Summary of Vu-X</h2>
    <bixpro>
      <xseq>
        <xchild/>
        <xmap><xseq>
          <xif><xseq><xwithtag>abstract</xwithtag></xseq>
            <xseq><xrename><xseq><xconst>p</xconst></xseq></xrename></xseq>
            <xseq><xconst/></xseq></xif>
        </xseq></xmap>
      </xseq>
    </bixpro>
    <h2>Authors</h2>
    <bixpro>
      <xseq>
        <xchild/>
          <xmap><xseq>
        <xif><xseq><xwithtag>author</xwithtag></xseq>
          <xseq><xrename><xseq><xconst>li</xconst></xseq></xrename></xseq>
          <xseq><xconst/></xseq></xif>
        </xseq></xmap>
        <xlet><var>authors</var><xseq>
          <xconst><ul/></xconst>
          <xchcont><xseq><xvar>authors</xvar></xseq></xchcont>
        </xseq></xlet>
      </xseq>
    </bixpro>
  </body>
</html>
```

Fig. 14 HTML-embedded Bi-X editor



contains the `xchild` construct which extracts all children of the root of the current input. Hence we can see this change comparing the right view and the middle view.

After editing a Bi-X program on the Bi-X code builder, the editing stage returns back to that of the HTML-embedded Bi-X editor. The `bixpro` component occurs as an HTML component, which is the transformation result of the inserted Bi-X program with the XML data, by sending a request for forward transformation to the Bi-X processor in the Vu-X system. Users can continue to edit the HTML-embedded Bi-X program in WYSIWYG style.

When users click the "Save" button on the HTML-embedded Bi-X editor, the HTML-embedded Bi-X program is converted into a single Bi-X program so that it can be used in the content updating mode of the Vu-X system. Both the generated Bi-X program and its original HTML-embedded Bi-X program are stored as a pair in the database of the Vu-X system. The former is employed in the content updating mode and the latter in the code editing mode.

5 Experiment on web site construction

We constructed several Web sites through the Vu-X system for experiments. Each Web site is intended for a research laboratory, researcher, small-sized enterprise, or family. Figure 16 shows a (front) page in the Web site of the research laboratory.

Visitors to the page will see the only lower right portion (with white background) of the image. The image is a screen-shot when the Web maintainer updates the page through the Vu-X system. This Web page (an HTML source of 9.9 kB) is

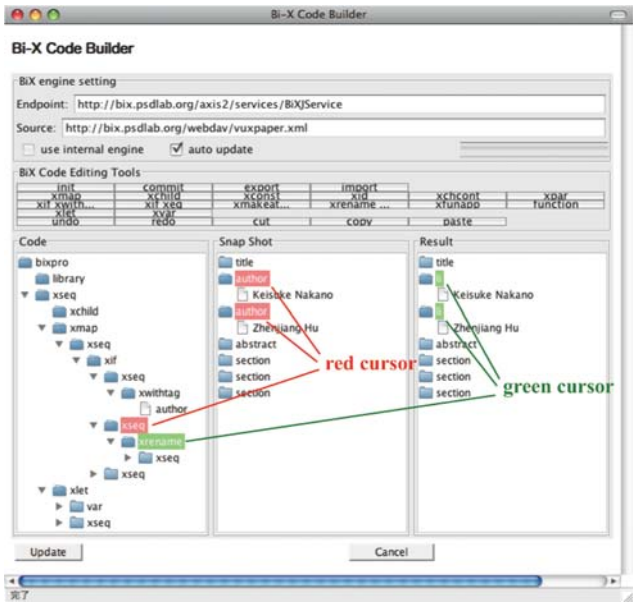


Fig. 15 Bi-X code builder

Fig. 16 A Web page constructed by the Vu-X system

generated by a Bi-X program (of 23.9 kB). We measured the size of all generated HTML documents for each Web site to show the size of the Web site. Though the considered Web sites are not so large, it suffices to demonstrate advantages of the Vu-X system. Larger Web sites can similarly be constructed by writing more Bi-X programs since each Web page corresponds to a Bi-X program.

We first asked a Web designer company to construct these Web sites in HTML and reconstructed all HTML sources as XML database and Bi-X programs for at most one week for each Web site. We employed the XQuery-to-BiX translation tool [20] to get most of these Bi-X programs.

Table 1 shows the size of an XML data and the size of Bi-X programs for each Web site construction. For every Bi-X program (counterpart of a Web page), we measured the size of generated HTML source and how much the Web page contains information which is shared with the other pages through the XML data. When we edit a Web page of the high percentage of shared contents, the edited part may affect the other pages with a strong possibility.

The same XML data is employed for two Web sites, 'laboratory A' and 'laboratory B', since they are intended for the same laboratory. Their difference is mainly the layout of their Web pages. Users can reuse not only XML data but also Bi-X programs.

By employing the same Bi-X programs and replacing a XML data, we can construct a Web site for another organization in the same layout. In this case, the Vu-X system can also be used as a *content management system* (CMS) by preparing many kinds of Bi-X programs as templates, where users do not have to understand the Bi-X language. We will discuss more about the relationship between our system and CMS in Sect. 7.

6 Discussion

The current Vu-X system still requires more improvement for practical use though it provides many functions to attain consistent Web site updating. In this section, we discuss on possible extension of the Vu-X system and several issues on the current implementation.

6.1 Possible extension of the Vu-X system

The framework of the Vu-X system can be considered as a combination of a WYSIWYG HTML editor and bidirectional transformation. We discuss the possibility of replacing them using the existing technology.

First, consider replacing the WYSIWYG HTML editor. We have independently developed the WYSIWYG HTML editor for the Vu-X system so that updates on the editor can be reflected back to the database. If the existing WYSIWYG HTML editor can be extended with a function for communicating with the database, the editor will be able to be used as a front-end of the Vu-X system. In order to do so, the WYSIWYG HTML editor should be extensible using a kind of plug-ins. For instance, it is better to have a way to capture and hook events of operations in the editor. Otherwise, users have to maintain communication by hand between the HTML editor and the Bi-X processor in Vu-X.

Next, consider replacing bidirectional transformation language. We employed the Bi-X language for the Vu-X system. There exist few languages for bidirectional transformation. A possible choice is Boomerang [7,13]. It suffices to write

Table 1 Size of files required for Web site construction

Web site (size of XML data, kB)	Bi-X programs (size, kB)	Size of generated HTML (kB)	Percentage of shared information (%)
Laboratory A	`access.bix` (16.2)	3.1	40.4
(17.8)	`entrance.bix` (16.8)	4.8	59.2
	`index.bix` (14.2)	3.7	43.8
	`lecture.bix` (15.2)	3.5	39.5
	`lecture_01.bix` (14.5)	3.4	59.0
	`lecture_02.bix` (14.5)	3.0	53.5
	`member.bix` (13.8)	3.2	42.0
	`member_01.bix` (19.2)	5.8	74.4
	`member_02.bix` (19.2)	6.9	78.6
	`project.bix` (13.9)	3.8	53.9
	`techrep.bix` (14.9)	4.3	55.3
	`techrep_2007.bix` (15.8)	3.8	58.0
	`techrep_2008.bix` (15.8)	3.1	47.9
	`thesis.bix` (15.4)	4.8	58.9
Laboratory B	`access.bix` (14.5)	2.7	42.8
(17.8)	`entrance.bix` (15.2)	4.4	62.2
	`index.bix` (12.3)	3.1	48.9
	`lecture.bix` (13.6)	3.1	41.3
	`lecture_01.bix` (12.8)	3.0	59.3
	`lecture_02.bix` (12.8)	2.6	53.0
	`member.bix` (12.4)	2.9	43.8
	`member_01.bix` (17.6)	5.4	78.8
	`member_02.bix` (17.6)	6.6	82.4
	`project.bix` (12.2)	3.4	57.1
	`techrep.bix` (13.4)	3.9	58.1
	`techrep_2007.bix` (14.2)	3.4	61.6
	`techrep_2008.bix` (14.2)	2.7	51.0
	`thesis.bix` (13.7)	4.4	61.6
Laboratory C	`access.bix` (19.2)	2.5	27.6
(19.3)	`index-e.bix` (23.9)	9.9	15.9
	`index.bix` (18.6)	4.3	38.7
	`links.bix` (15.1)	2.3	30.2
	`members.bix` (15.6)	2.6	34.7
	`papers.bix` (16.7)	5.2	71.6
	`research.bix` (14.6)	4.5	72.6
	`seminars.bix` (15.4)	3.3	58.0
Researcher	`a_career.bix` (13.6)	2.0	42.3
(11.7)	`activities.bix` (13.5)	3.2	65.2
	`awards.bix` (13.6)	1.7	39.6
	`index.bix` (16.6)	2.6	46.2
	`lectures.bix` (14.1)	2.3	41.7
	`links.bix` (14.1)	2.2	27.1
	`p_career.bix` (13.7)	2.8	51.7
	`publications.bix` (13.8)	3.8	66.2

Table 1 continued

Web site (size of XML data, kB)	Bi-X programs (size, kB)	Size of generated HTML (kB)	Percentage of shared information (%)
Enterprise	`about.bix` (11.5)	2.0	47.8
(9.3)	`contact.bix` (12.2)	2.1	48.6
	`event.bix` (12.3)	2.0	41.0
	`index.bix` (13.9)	2.2	48.5
	`privacy.bix` (11.6)	3.6	71.0
	`products.bix` (11.9)	3.8	72.2
	`recruit.bix` (12.1)	1.7	36.7
	`works.bix` (11.5)	2.5	58.6
Family	`hobby.bix` (13.3)	1.5	47.3
(3.9)	`index.bix` (13.8)	2.2	55.9
	`links.bix` (13.7)	1.7	53.9
	`other.bix` (12.9)	1.3	44.7
	`photo.bix` (12.4)	2.7	70.5
	`profile.bix` (14.3)	2.2	60.6

a single transformation in this language to specify a bidirectional transformation like that in the Bi-X language. However, if one wants to employ it as a back-end of the Vu-X system, it would be nice to have an editor that allows us to write a program in WYSIWYG style as is done by the HTML-embedded Bi-X editor and Bi-X code builder.

6.2 Issues on the Vu-X system

There is still room for improvement in the current Vu-X system, which are realized by our experiments. Let us discuss several issues to be solved for practical and convenient use.

6.2.1 Maintaining multiple XML data

The current Vu-X system assumes that database for a Web site is represented by a single XML data. However, users may want to construct their Web sites using multiple XML data to share a part of their contents with other Web sites. We need to support management of such distributed XML data in terms of bidirectional transformation. Fortunately, the Bi-X language provides the `<input>` construct to support the multiple inputs. What we need for this issue is to rearrange the database management system and provide an interface for the Bi-X function manage multiple XML data for a single Web site in the Vu-X system.

XML data distribution leads another benefit for our system. Backward transformation by the Bi-X processor replaces the whole XML data with updated XML data. If the data is separated into multiple XML data, it is updated more efficiently.

6.2.2 Sharing a part of transformation programs

The current Vu-X system assumes that a Web page is generated by a single Bi-X program. However, users may want to share a part of the Bi-X program with other Web page to unify the layout of Web pages in their Web sites. Fortunately, the Bi-X language provides the `<?import?>` instruction to load other Bi-X programs. What we need for this issue is to rearrange the database management system for Bi-X programs and the Bi-X code builder.

6.2.3 Access control for security and updatability

The current Vu-X system has just a simple security mechanism which certifies through a password whether an user is permitted to edit or not. However, users may want to give different permissions for different editors. We need to introduce an access control mechanism to designate permissions. It will be helpful for multiple users to simultaneously update the same Web site without conflict. We may have to combine existing access control mechanisms for XML data [6,8,22,25,27] with bidirectional transformation.

Access control mechanism is required for not only security but also updatability. In the content updating mode of the Vu-X system, it is not possible for users to edit an element which is generated by the Bi-X program independent of XML data. For example, consider a fragment of a Bi-X program, `<xconst><h2>Abstract</h2></xconst>` in Fig. 6, which generates an `h2` element by its forward transformation without information in the XML data. Though a small editing window for the text is popped up when users click this element in the content updating mode, backward transformation fails to reflect any modification to the XML data

because there is no corresponding text. It would be nice to statically detect such invalid modification and visually show its impossibility, e.g., by graying out the part. It may be nicer for the Vu-X system to suggest to users that the modification can be done in the code editing mode if possible.

6.2.4 Visualizing side effects

The current Vu-X system cannot display which part will be changed and synchronized before updating. Though automatic synchronization provided by the Vu-X system is very useful for users, sometimes they may want to know which part will be synchronized. It is possible to find parts which are changed by a modification through applying backward and forward transformation. We need to consider a nice view for visualizing the positions to be changed. It is difficult to show all of them because such side effects may not always happen at the same Web page under editing.

7 Related work

There are dozens of WYSIWYG HTML editors, such as Adobe Dreamweaver [1], Microsoft Expression Web [10], and Aptana Studio [3]. Most of these, however, require special software to be installed to employ them. Though some of them runs on the Web browser the WYSIWYG HTML editor, e.g., Namo Web Editor Control [24], they still provide a tool for editing HTML sources one by one. Since they do not take dependencies between multiple HTML sources into account, the Web site maintainer has to consider their consistency, which is in sharp contrast to transformation-based Web site construction. Although the WYSIWYG HTML editor provided by the Vu-X system is less powerful than existing ones in the sense that only modifications that can be reflected back to the XML data are allowed, our approach can be applied to these by integrating them with bidirectional transformation based Web site construction as discussed in Sect. 6.

The Web site construction based on bidirectional transformation may remind readers Web content management systems (Web CMS), such as Wiki and Weblog software, which allows their contents to be updated on the Web browser with no or little knowledge of database management or layout descriptions. If users want to change the layout of their Web site, they will choose one of provided templates in the Web CMS. In the Vu-X system, a similar Web site construction can be accomplished by providing Bi-X programs as templates. Users who are familiar to HTML can customize templates in code editing mode of the Vu-X system. Furthermore, users who acquire a programming skill of Bi-X or XQuery language can freely change the layout as they like. A major advantage of the Vu-X system is that template developers write a program only in one-direction from the database to HTML sources. On the other hand, the existing Web CMS generally requires the developers to take care of how modification on the view is reflected to the database.

Our design of the Vu-X system based on bidirectional transformation was greatly inspired by pioneering work [7, 13] on data synchronization based on bidirectional transformation, which originated from work on view updating [5,9, 14,15,19] in the database community, where modifications to view could be reflected back to the original database. We borrowed this technique to enable Web site maintenance with one significant extension: editing operations can not only modify the view but also the transformation code, which has not been exploited before.

8 Conclusion

This paper has proposed a novel approach to Web site construction based on bidirectional transformation and presented an implementation of a practical updating system called Vu-X, which can keep the content of Web sites consistent by updating Web pages in WYSIWYG style on Web browsers. Since the Vu-X system employs the Bi-X bidirectional transformation language, it suffices to only specify forward transformation and the corresponding backward transformation is automatically derived. Furthermore, the Vu-X system provides a WYSIWYG tool to enable Bi-X programs to be easily constructed.

Even though the Vu-X system facilitates consistent Web site updating, it would require much effort to introduce it to existing Web sites, because we have to restructure their contents in XML format and prepare Bi-X programs from scratch to benefit from the Vu-X system. We intend to develop in future work a "switchover" assistant tool for existing Web sites which is constructed without the Vu-X system.

A current shortcut way is to prepare XML database and XQuery programs for existing Web sites and employ XQuery-to-BiX tool [20] as we did for our experiment (Sect. 5).

Acknowledgments We would like to thank the staff members of Dong Wu Information System Co. Ltd. who mainly implemented the Vu-X system following our design decisions. We are also grateful to Dongxi Liu, Yasushi Hayashi, Kento Emoto, Kazutaka Matsuda and Akimasa Morihata for their cooperation in earlier implementation of the Vu-X system and Hideya Iwasaki for his constructive comments on this paper. We are indebted to anonymous reviewers for insightful comments.

References

1. Adobe Dreamweaver CS4. http://www.adobe.com/products/dreamweaver/
2. Apache Tomcat. http://tomcat.apache.org/
3. Aptana Studio. http://www.aptana.com/studio/

4. Atzeni, P., Mecca, G., Merialdo, P.: To weave the Web. In: Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB), pp. 206–215 (1997)
5. Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Trans. Datab. Syst. **6**(4), 557–575 (1981)
6. Bertino, E., Castano, S., Ferrari, E., Mesiti, M.: Specifying and enforcing access control policies for XML document sources. World Wide Web J. **33**, 139–151 (2000)
7. Bohannon, A., Foster, J.N., Pierce, B.C., Pilkiewicz, A., Schmitt, A.: Boomerang: resourceful lenses for string data. In: Proceedings of the 35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pp. 407–419. ACM Press, New York (2008)
8. Damiani, E. De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. ACM Trans. Inf. Syst. Secur. **52**, 169–202, ACM Press, New York (2002)
9. Dayal, U., Bernstein, P.A.: On the correct translation of update operations on relational views. ACM Trans. Datab. Syst. **7**(3), 381–416 (1982)
10. Microsoft Expression Web. http://expression.microsoft.com/
11. Fernandez, M.F., Florescu, D., Kang, J., Levy, A.Y., Suciu, D.: Overview of strudel—a Web-site management system. Netw. Inf. Syst. J. **1**(1), 115–140 (1998)
12. Fernandez, M.F., Florescu, D., Levy, A.Y., Suciu, D.: Declarative specification of Web sites with Strudel. VLDB J. **9**(1), 38–55 (2000)
13. Foster, J.N., Greenwald, M.B., Moore, J.T., Pierce, B.C., Schmitt, A.: Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. In: Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL), pp. 233–246. ACM Press, New York (2005)
14. Gottlob, G., Paolini, P., Zicari, R.: Properties and update semantics of consistent views. ACM Trans. Datab. Syst. **13**(4), 486–524 (1988)
15. Hegner S.J.: Foundations of canonical update support for closed database views. In: Proceedings of the 3rd International Conference on Database Theory (ICDT), pp. 422–436. Springer, Berlin (1990)
16. Hu, Z., Mu, S.-C., Takeichi, M.: A programmable editor for developing structured documents based on bidirectional transformations. In: Proceedings of the 2004 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM), pp. 178–189. ACM Press, New York (2004)
17. Lämmel, R.: Coupled software transformations (extended abstract). In: 1st International Workshop on Software Evolution Transformations, pp. 31–35 (2004)
18. Lau, T., Staczek, J.: A contextual inquiry-based critique of the strudel Web site maintenance system. Technical Report TR-99-01-01, Department of Computer Science and Engineering, University of Washington (1999)
19. Lechtenbörger, J., Vossen, G.: On the computation of relational view complements. ACM Trans. Datab. Syst. **28**(2), 175–208 (2003)
20. Liu, D., Hu, Z., Takeichi, M.:Bidirectional interpretation of XQuery. In: Proceedings of the 2007 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM), pp. 21–30 (2007)
21. Meertens, L.: Designing constraint maintainers for user interaction. http://www.cwi.nl/~lambert/ (1998)
22. Murata, M., Tozawa, A., Kudo, M., Hada, S.: XML access control using static analysis. In: Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS), pp. 73–84 (2003)
23. Nakano, K., Hu, Z., Takeichi, M.: Consistent Web site updating based on bidirectional transformation In: Proceedings of the 10th IEEE International Symposium on Web Site Evolution (WSE), pp. 45–54 (2008)
24. Namo Web Editor Control. http://www.namo.com/products/webeditorcontrol.php
25. Qi, N., Kudo, M., Myllymaki, J., Pirahesh, H.: A function-based access control model for XML databases. In: Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM), pp. 115–122 (2005)
26. XSL transformations (XSLT) version 2.0. http://www.w3c.org/TR/xslt20/, 2006
27. Yu, T., Srivastava, D., Lakshmanan, L.V.S., Jagadish, H.V.: Compressed accessibility map: efficient access control for XML. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), pp. 478–489 (2002)