(1) Workshop: Workshop on Databases and Software Engineering.

(2) Title: DESDL: A Data Extraction Service Description Language.

(3) Abstract:

In this paper, we design an XML-based description language, named Data Extraction Service Description Language (DESDL), for data extraction services. In DESDL, the users can describe a set of services each of which extracts data from the designated web pages and then saves these data into local databases or navigates into next services. The features of this language include:

(a) Query expressions that specify the rules to extract data from designated web pages.

(b) Multi-way navigation that the users can use to traverse web pages.

(c) Plug-in code, named DESDLet, that users can use to define the process of extracted data, e.g., save them into databases or use them to navigate into more pages.

In this paper, we also implement the system for DESDL and demonstrate it by using it to implement a price-comparison site where we extract product information from over 50 electronic-commerce sites in Taiwan. In our experience, one engineer only needs about one working day to write a DESDL script to extract product information from one E-commerce web site. This greatly reduces the cost of maintaining such a web site.

(4) About Authors:

I-Chen Wu[*], Jui-Yuan Su[*], Loon-Been Chen[†], Kuang-Ting Chien[*]

[*] Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu, Taiwan.

[†] Department of Information Management, Chin-Min College, Tou-Fen, Miao-Li, Taiwan.

E-mail: icwu@csie.nctu.edu.tw.

TEL: +886-3-5731855. FAX: +886-3-5733777.

(5) Contact Person: I-Chen Wu

(6) List of Keywords: Data extraction, XML, HTML, XML-QL, XPath, WIDL, DESDL.

Workshop on Databases and Software Engineering

DESDL: A Data Extraction Service Description Language

I-Chen Wu[*], Jui-Yuan Su[*], Loon-Been Chen[†], Kuang-Ting Chien[*]

[*] Department of Computer Science and Information Engineering

National Chiao Tung University, Hsinchu, Taiwan

[†] Department of Information Management, Chin-Min College, Tou-Fen, Miao-Li, Taiwan

TEL: +886-3-5731855 FAX: +886-3-5733777

icwu@csie.nctu.edu.tw

Contact person: I-Chen Wu

Abstract:

In this paper, we design an XML-based description language, named Data Extraction Service Description Language (DESDL), for data extraction services. In DESDL, the users can describe a set of services each of which extracts data from the designated web pages and then saves these data into local databases or navigates into next services. The features of this language include:

(1) Query expressions that specify the rules to extract data from designated web pages.

(2) Multi-way navigation that the users can use to traverse web pages.

(3) Plug-in code, named DESDLet, that users can use to define the process of extracted data, e.g., save them into databases or use them to navigate into more pages.

In this paper, we also implement the system for DESDL and demonstrate it by using it to implement a price-comparison site where we extract product information from over 50 electronic-commerce sites in Taiwan. In our experience, one engineer only needs about one working day to write a DESDL script to extract product information from one E-commerce web site. This greatly reduces the cost of maintaining such a web site.

Keywords: Data extraction, XML, HTML, XML-QL, XQuery, XPath, WIDL, DESDL.

1. Introduction

With the rapid development of World Wide Web (WWW) recently, it becomes more and more important for users to obtain information over Internet. In order to collect useful information over Internet, users first search them from some portal sites, such as Yahoo! [20][21], and then browse by clicking through the related pages. However, users sometimes want to further process the browsed pages, such as extraction of some products' names and prices into database for later uses. For example, some price comparison Web sites [22] need to use such tools to extract information for their users to compare prices; in addition, some companies may need such tools to help them regularly grab some important business news. For such applications, the users usually want to extract only some significant segments inside HTML or XML pages [4][6][14][15][16].

In order to solve this problem, many researchers proposed page extraction languages [2][7][10][12][13][18][19] so that the extraction processes follow the guides of the scripts in the languages. These languages can be divided into the following two classes.

(1) Query-based languages: In this type of page extraction languages, such as XML-QL [2], XQL [10] and XQuery [13], etc., users write query expressions to extract data among pages. Importantly, the sequence of extracting pages is not specified in advanced. For example, if we want to extract the price information of a product in page A and the product information in page B, the scripts in XML-QL do not indicate which page to retrieve first.

(2) Service-based languages: In this type of page extraction languages, such as WIDL [7] (Web Interface Definition Language), we write several services each of which extracts

data from one designated web page and then processes the extracted data (e.g., saves them into local databases or uses them to navigate into other pages). The sequence of processing pages is well specified in advanced. For the example above with pages A and B, we need to specify which to extract first in WIDL [7].

Although specifying the sequence of processing pages surely incurs the overhead on designing data extraction, it is still important and necessary to specify them for those extraction applications with *web pages* for the following reasons:

- The cookie problem. Many web sites ask users to login with accounts and passwords, before allowing users to browse more pages. So, usually, they set cookies in the first homepage, so that they can recognize the users for next pages. Since most query-based languages do not support sequencing, we cannot do data extraction from the web site with the cookie problem in these languages.
- The problem of security mechanisms. Similarly, in most query-based languages, we cannot access encrypted pages (using SSL protocol) without starting browsing the first homepage or without specifying correct security information.
- The problem of the referral header options. We cannot make use of this information in most query-based languages.
- The problem of using "GET" or "POST" to access web pages. Similarly, we cannot indicate the method in most query-based languages.
- Many web pages have "next page" links that we use to find more data in the following pages. In most query-based languages, it is hard to specify these pages and queries among these pages.

The above indicates that it is more practical to use service-based languages to extract

data from web pages than to use query-based languages. For example, it is more appropriate to use service-based languages to build price comparison web sites, a major application to motivate the research of this paper. Therefore, in this paper, we will focus on the service-based languages. Note that query-based languages are normally used to extract data from HTML or XML files stored locally or in an environment under control.

In this paper, we propose a new service-based language, named Data Extraction Service Description Language (DESDL), an XML-based description language for data extraction services. In DESDL, the users can describe a set of services each of which extracts data from the designated web pages and then saves these data into local databases or navigates into next services. Basically, the DESDL language is extended from WIDL, proposed by Webmethods [7]. The features of this language include:

(1) Query expressions that specify the rules to extract data from designated web pages. Currently, we choose XPath [18] as the format of query expressions and also will leave some rooms for future extensions.
(2) Multi-way navigation that the users can use to traverse web pages. Note that WIDL allows users to navigate into another page only.
(3) Plug-in code, named *DESDLet*, that users can use to define the process of extracted data, e.g., save them into databases or use them to navigate into more pages.

Due to the first feature, we can leverage the current techniques for query expressions. Due to the second feature, we can traverse a whole Web category tree (e.g., Yahoo category) in DESDL while we can only do single-chain-like operations (such as supply-demand chains) in WIDL. Due to the third feature, DESDLets (currently in Java) provide programmers with an elegant solution to design and extend various services. From above, DESDL can cover a

large set of web applications than WIDL and have extensibility and flexibility.

We also implement the system for DESDL and demonstrate it by using it to implement a price-comparison site where we extract product information from over 50 electronic-commerce sites in Taiwan. From our empirical experiences, one programmer only needs about one working day to write a DESDL script to extract product information from one E-commerce web site. This greatly reduces the overhead of maintaining such a web site.

In this paper, Section 2 reviews some languages such as HTML, XML, XPath, XML-QL and WIDL. Section 3 describes the specification of our DESDL language. Section 4 describes our system for DESDL and demonstrates our system by using it to implement a price-comparison web site.

2. Review

In this Section, we will review the languages HTML, XML, XPath, XML-QL and WIDL. In this paper, we focus on the data extraction of web pages most of which are in HTML and some of which are in XML. In our data extraction services, we currently use XPath as the format of query expressions that specify the rules to extract data from designated web pages. Note that though XPath was originally designed for XML files, it is actually also suitable to describe the web pages in HTML.

Then, we will briefly describe the languages XML-QL (an example of query-based languages) and WIDL (an example of service-based languages) and illustrate the problems of using them for data extraction.

HTML

In order to support all kinds of information presentation (including multimedia) in web pages, W3C Consortium proposes the standard, HTML (HyperText Markup Language), a markup language defined from SGML (Standard Generalized Markup Language). An HTML example is shown in Figure 1 below.

```
<H1>Jui-Yuan Su</H1>
<TABLE BORDER="1">
  <TR>
    <TD>Telephone number:</TD> <TD><B>(02)23456789</B></TD>
  </TR>
  <TR>
     <TD>Address:</TD> <TD><B>1001, Ta Hsueh Road </B></TD>
  </TR>
</TABLE>
```

Figure 1. A segment of an HTML document

One of problems in HTML is not to keep semantics in the document. For example, in Figure 1, the author name is in the tag <H1>. However, the tag is only for presentation, not for the meaning of "author name". In order to solve this problem, there is a need to define various markup languages or tags. Although SGML [5] is a good candidate to serve this purpose, it is too complex to support SGML. Therefore, W3C Consortium defines a simplified version, XML (eXtensible Markup Language) [14].

XML

Based on XML, we can redefine a marked up language so that the document in Figure 1

can be rewritten as in Figure 2 below. Now, this document preserves the meaning of *author name* for “Jui-Yuan Su”.

```
<Author>
    <Name>Jui-Yuan Su</Name>
    <Phone> (02)23456789</Phone>
    <Address>1001, Ta Hsueh Road</Address>
</Author>
```

Figure 2. A segment of an XML document

XPath

XPath (XML Path) is a language for referencing parts of an XML document. XPath has become a standard in W3C Consortium. XPath works in the XQuery 1.0 and XPath 2.0 Data Model [17]. The data model specifies what information in a document is accessible. Since XML (or HTML) documents are tree-structured, a node-labeled tree is used to represent the document.

In XPath, a path expression locates nodes inside a tree, and returns an array (or a sequence) of nodes in the document order. A path expression is always evaluated from the corresponding context document. Path expressions include rooted path expressions (starting from the root node) and relative path expressions (starting from a relative node).

A rooted path expression consists of “/” or “//”, followed by an expression, while a relative path expression consists of two expressions, separated by “/” or “//”. For relative path expressions, a “/” by itself selects the root node of the context document. A “//” is a short form for “/descendant-or-self::node()/”. For example, in Figure 2, “//Name” is a short form

for “/descendant-or-self::node()/child::Name”, and so it will select any Name element in the document. And, “/Author/Name/text()” indicates “Jui-Yuan Su”.

XML-QL

XML-QL, submitted to W3C by AT&T Labs [2] in 1998, is a query language in which users can extract information from XML documents. The syntax of XML-QL, similar to SQL in databases, is based on “select-where” structure. An XML-QL script of the XML document in Figure 2 is illustrated in Figure3, and the output is shown in Figure 4.

```
WHERE <Author>
          <Name>$n</>
          <Address>1001, Ta Hsueh Road</>
       </> IN "handset.xml"
CONSTRUCT <NCTU-Author>
               <Name>$p</>
          </> IN figure2.xml
```

Figure 3. An XML-QL script for the XML document in Figure 2

```
<NCTU-Author>
     <Name>Jui-Yuan Su</Name>
</NCTU-Author>
```

Figure 4. The output for the XML-QL script in Figure 3

Furthermore, XML-QL supports the query among several XML documents. In Figure 5 (below), an XML-QL script is illustrated to extract product title `$t` and product price `$p` from a record in the file `http://www.ubid.com.tw/products.html`, and the same product title `$t` and product information `$i` from a record of the file `http://www.ubid.com.tw/detail.html`. Then, construct a new record with (`$t`, `$p`, `$i`).

```
WHERE <table>
         <tr><td>$t</><td>$p</></>
       </> IN http://www.ubid.com.tw/products.html
       <table>
         <tr><td>$t</><td>$i</></>
       </> IN http://www.ubid.com.tw/detail.html
CONSTRUCT ... <td> $t </> <td> $p </> <td> $I </>
```

Figure 5. An XML-QL script to extract data from two documents

However, the problem with XML-QL or other query-based languages is: The sequence of extracting pages, such as `http://www.ubid.com.tw/products.html` and `http://www.ubid.com.tw/detail.html`, are not specified in advance. Therefore, there is no information that the system can use to judge which to browse first. As described in Section 1, this may cause the following problems on accessing many web sites: cookies, security referral header options, form submission, etc.

WIDL

WIDL (Web Interface Definition Language) was submitted to W3C in September 1997. It is a service-based language that provides programmers with interfaces to manipulate semi-structure data and services such as CGI-bin [8], database or back-end systems. It is a way to represent request and response interactions such as form submission, web site navigation [1][3][9] and data extraction over standard Web protocols.

Since WIDL is service-based, we can use the extracted data (from the current page) to access next page. Thus, the problems, such as cookies, security referral header options, form submission, etc., can be avoided. Figure 6 (below) is an example of WIDL script for a web

page to extract a product record with product title and price. For variable `title`, we use the reference "`doc/table.tr[0].td[0].value`" to extract the first cell of the first row of a table in the document (note that `doc` is required to indicate the document).

```
<WIDL NAME="FindProduct" BASEURL="http://www.ubid.com.tw" ... >
 <SERVICE NAME="Products" URL="/products.html" OUTPUT="ProductOutput"/>
  <BINDING NAME="ProductOutput" TYPE="OUTPUT">
     <VARIABLE NAME="title" TYPE="String"
             REFERENCE="doc/table.tr[0].td[0].value"/>
     <VARIABLE NAME="price" TYPE="String"
             REFERENCE="doc/table.tr[0].td[1].value"/>
  </BINDING>
 </SERVICE>
</WIDL>
```

Figure 6. A WIDL script to extract a product with product title and price

WIDL can only support supply-chain-like navigation. For example, in Figure 6, the result of the extracted fields, such as title and price, can be redirected to another service. Basically, WIDL cannot support more complex navigation, such as multi-way navigation that allows users to navigate the whole category of a portal site.

3. DESDL

In this paper, we propose a new service-based language, named DESDL. The users can write scripts in DESDL to extract data and to process them easily.

A DESDL script is enclosed by the element `<DESDL>`. This element contains two types of elements: The element `<INIT>` specifies the initial URL and the initial page extraction service.

The element <SERVICE> is used to specify how to extract data from web pages and to navigate to next pages.

```
<DESDL>
  <INIT URL="..." SERVICE="GetPersonalData"/>
  <SERVICE NAME="GetPersonalData"/>
    <VAR NAME="NAME" PATH="//H1/text()"/>
    <VAR NAME="TELNUM" PATH="//TR[0]/TD[1]/text()"/>
    <VAR NAME="ADDRESS" PATH="//TR[1]/TD[1]/text()"/>
  </SERVICE>
</DESDL>
```

Figure 7. (a) A DESDL script to extract data from the HTML document in Figure 1

```
<DESDL>
  <INIT URL="..." SERVICE="GetPersonalData"/>
  <SERVICE NAME="GetPersonalData"/>
    <VAR NAME="NAME" PATH="//AUTHOR/NAME/text()"/>
    <VAR NAME="TELNUM" PATH="//AUTHOR/PHONE/text()"/>
    <VAR NAME="ADDRESS" PATH="//AUTHOR/ADDRESS/text()"/>
  </SERVICE>
</DESDL>
```

Figure 7. (b) A DESDL script to extract data from the XML document in Figure 2

For example, Figure 7(a) shows a DESDL script to extract data from the HTML document in Figure 1. Initially, the DESDL system initially loads a web page at the URL specified in the element <INIT>, and use the service GetPersonalData (specified in the element <INIT> too) to process the data extraction. In the service GetPersonalData, the elements <VAR> define the variable name and its value to the expression. So, in the script, we can extract the names, the telephone numbers and the addresses of the authors from the HTML document in Figure 1 and put these values into those variables named NAME, TELNUM and ADDRESS respectively. Figure 7(b) shows a DESDL script that can extract the same data from the XML document in Figure 2.

The language DESDL has the following three main features:

(1) Query expressions that specify the rules to extract data from designated web pages.

(2) Multi-way navigation that the users can use to traverse web pages.

(3) Plug-in code, named DESDLet, that programmers can use to define the process of extracted data, e.g., save them into databases or use them to navigate into more pages.

We will describe these features as follows.

Query Expressions

Since XPath [18] is already the standard of W3C Consortium as describe above, we basically choose XPath as the standard of query expressions of DESDL. For example, in Figure 7, we can specify the query expressions in the attribute `PATH` of the element `<VAR>`. However, in order to make users write scripts more easily, we also allow users to write some descriptions in a simplified manner. For example, "`/a['foo']`" means that the anchor selected contains at least one '`foo`' in the text part.

Multi-way Navigation

In DESDL, when extracting data inside a service, we can also invoke next services by using the element `<INVOKE>`, which includes two main attributes: `SERVICE`, the name of next service, `URL`, the URL of the next page to request.

Furthermore, in DESDL, we can invoke extraction services in the multi-way manner. That is, after extracting data from a page, we can continue to process several pages next. In

order to support multi-way navigation, the element <SERVICE> may contain the element <FOREACH>. In the element <FOREACH>, the attribute FROM indicates an array variable evaluated from an XPath expression earlier. In the DESDL system, for each data in the array variable, do everything inside the element.

```
<DESDL>
  <INIT SERVICE="SERVICE1" URL="..."/>
  <SERVICE NAME="SERVICE1">
    <VAR NAME="PRODUCTS" PATH="//table/tr"/>
    <FOREACH FROM="$PRODUCTS">
      <VAR NAME="Title" PATH="td[0]/text()"/>
      <VAR NAME="Link" PATH="td[1]/a[1]/@href"/>
      <VAR NAME="Price" PATH="td[2]/text()"/>
      <INVOKE URL="$Link" SERVICE="SERVICE2"/>
    </FOREACH>
  </SERVICE>
  <SERVICE NAME="SERVICE2">
    ...
  </SERVICE>
</DESDL>
```

Figure 8. An example of using multi-way navigation

For example, in Figure 8, the variable PRODUCTS contains an array of all rows (each of which corresponding to a product) in tables, and, therefore, the DESDL system will process all elements inside <FOREACH> for each row. Then, for each table row (or product), extract the title, link (to the page with product information) and price from the row into the variables Title, Link, Price, respectively; and invoke new service via the variable Link. Thus, this implies that we can invoke each product information page. For a portal site with a tree of category, we will easily use this way to navigate the whole category tree.

DESDLets

In our research, we find that many users may want to modify or slightly modify the behavior of the invoke operation. Therefore, we support a mechanism, named DESDLets, that allows users to change the behavior at <INVOKE>. In DESDL, the element <INVOKE> contains an attribute DESDLET, that specifies the DESDLet routine to process the invocation. Currently, the DESDLet routine is written in Java.

The DESDL system passes three types of data into DESDLet: the variables that have been specified, the service name, and the URL to be invoked. Then, DESDLets process these data. For example, DESDLets can store the extracted value into databases, or check to avoid duplication in navigation if the URL has been invoked earlier. Finally, DESDLets can decide whether to invoke the action specified in the <INVOKE> element to load next page and call the next service.

Currently, there are some DESDLets implemented in the system. The first is to check duplicated URLs. The hash table is implemented in the DESDLets to store the URL that the system has been navigated. DESDLets check the URLs in the hash table to ensure the same pages will not be accessed more than once. If the URLs are not accessed before, DESDLets insert the URLs into the hash table. We use two variables, KEY and KEY0, for keys in the hash table. The variable KEY is the same if the URL was invoked in the same service. KEY0 is the same if the URL was invoked in the same DESDL script.

The second is to store data into database. The data are processed and then save into database. The third is to log records. Specific records are logged for analysis and tracking.

In the next section, we illustrate our DESDL system by a real application for the price

comparison.

4. Application: Price Comparison

Based on DESDL, we have implemented a price comparison system that extracts price information of products from some E-commerce web sites. The categories of E-commerce web sites include those for selling communication materials, computer hardware and software products, book stores, movies, music disks, daily necessities, etc. In this system, one script of DESDL is written for each site. The system parses the script, traverses related pages, extracts related information (such as prices) of products from these web sites, and saves them into database.

The process of traversing web site is to describe sequences of extracting information of products. Currently, most of the shopping sites are constructed by a large product category. Once a product is located, there is a list of pages for all records of the product. Each category contains subcategories. Each subcategory contains a list of products. Product list pages may contain a series of links via buttons of “next page” because all products may not be able to be displayed in one page.

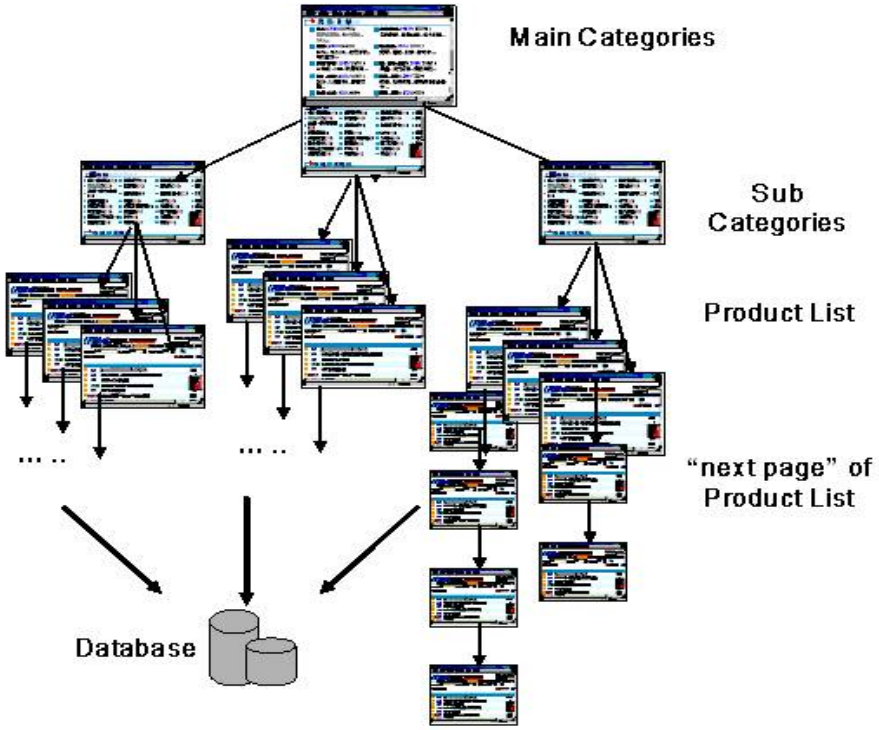

Figure 9. The page hierarchy of an e-commerce web site

In the rest of this Section, we will use the bidding web site, `http://www.ubid.com.tw` (abbr. UBID below), as an example to explain how the DESDL scripts are used to extract data. Figure 9 shows the page hierarchy of this web site.

The web page model of UBID is described as follows. The homepage of UBID contains links to several main categories. Each main category contains links to several subcategories. Each subcategory links to product list page. If there are a lot of products in one subcategory, the product list page will contain a link to next page of product list. The formats of product list page and "next page" of product list page are the same.

```
<DESDL NAME="UBID">
  <INIT SERVICE="MainCategory" .../>
  <SERVICE NAME="MainCategory">
    <FOREACH ...>
      ...
      <INVOKE SERVICE="SubCategory" ... />
    </FOREACH>
  </SERVICE>
  <SERVICE NAME="SubCategory">
    <FOREACH ...>
      ...
      <INVOKE SERVICE="ProductList" ... />
    </FOREACH>
  </SERVICE>
  <SERVICE NAME="ProductList">
      ...
  </SERVICE>
</DESDL>
```

Figure 10. A DESDL script for the web site UBID

```
<SERVICE NAME=”MainCategory”>
  <VAR NAME=HyperLink PATH=”...../a”/>
  <FOREACH FROM=$HyperLink>
    <INVOKE SERVICE=”SubCategory” PATH=”@href”/>
  </FOREACH>
</SERVICE>
```

Figure 11. The `MainCategory` service

Three services are defined in this example, `MainCategory`, `SubCategory` and `ProductList`, as shown in Figure 10. The `MainCategory` service loads the initial page at the URL of the `<INIT>` element, extracts the URLs of the subcategory pages, and then invokes the next pages via these URLs with the `SubCategory` service. Obviously, `MainCategory` uses multi-way navigation to invoke next subcategory pages in this case. The `MainCategory` service is shown in greater detail in Figure 11. First, it extracts all hyperlinks to the

subcategory into the variable `HyperLink`, and then uses `FOREACH` to navigate to each subcategory page.

```
<SERVICE NAME=" SubCategory">
  <VAR NAME=HyperLink PATH=".....a"/>
  <FOREACH FROM=$HyperLink>
    <INVOKE SERVICE="ProductList" PATH="@href"/>
  </FOREACH>
</SERVICE>
```

Figure 12. The `SubCategory` service

Next, similarly, the `SubCategory` service, as shown in Figure 12 (below), loads the pages that are invoked by `MainCategory` service, extracts the URLs of the product list pages, and then invokes the next pages via these URLs with the `ProductList` service.

```
<SERVICE NAME="ProductList"/>
  <VAR NAME="Products" PATH="....."/>
  <FOREACH FROM=$Products>
    <VAR NAME="Title" PATH="td[0]/text()"/>
    <VAR NAME="Info" PATH="td[1]/text()"/>
    <VAR NAME="Price" PATH="td[2]/text()"/>
    <INVOKE DESDLET="DESDL.save2db"/> <!--Save into DB>
  </FOREACH>
  <VAR NAME="HyperLink" PATH="//a['next']"/>
  <FOREACH FROM=$HyperLink>
    <INVOKE SERVICE="ProductList" PATH="@href"/>
  </FOREACH>
</SERVICE>
```

Figure 13. The `ProductList` service

Finally, the `ProductList` service, as shown in Figure 13 (below), loads the pages, extracts the URL of the next product list page, and then may invokes the pages via this URLs

with the `ProductList` service, if the next page exists. More importantly, the `ProductList` service also extracts the data of each product in the page and saves into the database.

5. Discussions and Conclusion

In this paper, we design an XML-based description language, named DESDL, for data extraction services. In DESDL, the users can describe a set of services each of which extracts data from the designated web pages and then saves these data into local databases or navigates into next services. The contribution of this paper is summarized as follows.

(1) Propose a service-based language, DESDL, for data extraction.
(2) Propose the first data extraction system for multi-way navigation. So, based on this feature, we can extract data from a complicated page hierarchy of a portal site.
(3) Support a flexible design by allowing programmers to plug-in code.

We have already implemented the system for DESDL and demonstrated it by implementing a price-comparison site where we extract product information from over 50 electronic-commerce sites (see Appendix 1) in Taiwan. In this experience, one programmer only needs one working day to write a DESDL script to extract product information from one E-commerce web site. This greatly reduces the overhead of maintaining such a web site. We deeply believe the DESDL system can be used in many practical Internet applications in the future.

6. References

[1] T. Berners-Lee, R. Fielding, H. Frystyk, "rfc1945 - Hypertext Transfer Protocol - HTTP/1.0", http://www.faqs.org/rfcs/rfc1945.html, May 1996.

[2] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu, "XML-QL: A Query Language for XML", http://www.w3.org/TR/NOTE-xml-ql/, Aug. 1998.

[3] R. Fielding, J. Gettys, J. Mogul, H.Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "RFC2616: Hypertext Transfer Protocol -- HTTP/1.1", http://www.faqs.org/rfcs/rfc2616.html, Jun. 1999.

[4] Steve Holzner, "XML Complete", McGraw-Hill, 1998.

[5] ISO (International Organization for Standardization). *ISO 8879:1986(E). Information processing -- Text and Office Systems -- Standard Generalized Markup Language (SGML).* First edition -- 1986-10-15. [Geneva]: International Organization for Standardization, 1986

[6] Sean Mcgrath, "XML by example: building E-commerce applications", Prentice Hall PTR, 1998.

[7] Phillip Merrick, Charles Allen, "Web Interface Definition Language", http://www.w3.org/TR/NOTE-widl, Sep 1997.

[8] NCSA HTTPd Development Team, "The Common Gateway Interface（CGI）", http://hoohoo.ncsa.uiuc.edu/cgi, Jan. 1998.

[9] Netscape Communications Corporation, "Persistent Client State HTTP Cookies", http://www.netscape.com/newsref/std/cookie_spec.html, 1998.

[10] Jonathan Robie, Joe Lapp, David Schach, "XQL：XML Query Language", http://www.w3.org/TandS/QL/QL98/pp/xql.html, Sep. 1998.

[11] Sun Microsystems, Inc., "Java(TM) 2 SDK Documentation", http://java.sun.com/products/jdk/1.2/download-docs.html, 2002.

[12] W3C Consortium, "XML Query", http://www.w3c.org/XML/Query, Apr. 2000.

[13] W3C Consortium, "XQuery 1.0: An XML Query Language", http://www.w3.org/TR/xquery/, April. 2002.

[14] W3C Consortium, "Extensible Markup Language (XML) 1.0 (Second Edition)", http://www.w3.org/TR/2000/REC-xml-20001006, October 2000.

[15] W3C Consortium, "Hyper Text Markup Language", http://www.w3c.org/Markup/, Jan. 1998.

[16] W3C Consortium, "HTML 4.01 Specification W3C Recommendation", http://www.w3c.org/TR/html401, Dec. 1999.

[17] W3C Consortium, "XQuery 1.0 and XPath 2.0 Data Model"， http://www.w3.org/TR/query-datamodel/, April. 2002.

[18] W3C, "XML Path Language (XPath) 2.0", http://www.w3.org/TR/2001/WD-xpath20-20011220, Dec. 2001.

[19] W3C, "XML Pointer Language (XPointer) Version 1.0", http://www.w3.org/TR/2001/CR-xptr-20010911/xptr.html, Sep. 2001.

[20] Yahoo! Inc, "Yahoo Search Engine", http://www.yahoo.com, 2002.

[21] 網擎資訊軟體股分有限公司, "Openfind 網路資源搜尋", http://www.openfind.com.tw, 2001.

[22] 傳資訊股分有限公司, "ubid 買賣王", http://www.ubid.com.tw, 2001.

Appendix 1. List of 50 E-commerce Sites processed in DESDL

Web sites	URLs	Web sites	URLs
買賣王	http://www.ubid.com.tw	拍賣王	http://www.bid.com.tw
明日世界	http://www.tomorrot.cm.tw	摩比家	http://www.mobihome.com.tw
3Clife	http://www.3clife.com.tw	都會賣	http://www.citymart.com.tw
金石堂	http://www.kingstone.com.tw	燦坤	http://www.tkec.com.tw
遠流	http://www.ylib.com.tw	博客來	http://www.books.com.tw
安瑟	http://www.answer.comt.w	AAGO	http://www.aago.com.tw
Ishopping	http://www.ishopping.com.tw	阿拉網	http://www.alot.com.tw
新絲路	http://www.silkbook.com.tw	搜主義	http://www.soidea.com.tw
酷奇	http://www.cuki.com.tw	Roses	http://www.roses.com.tw
集誠堂	http://www.learning123.com.tw	E 世代	http://www.1stlibrary.com.tw
網購	http://www.wantgo.com.tw	賣蕃天	http://market.yam.com.tw
博碩	http://www.drmaster.com.tw/	HOT	http://www.hot.com.tw
天下文化	http://www.bookzone.com.tw	卡旺城	http://www.com1.com.tw
三民書局	http://www.sanmin.com.tw/	3chome	http://www.3chome.com.tw
愛炫2000	http://ifsashion2000.tw.to	華彩	http://www.soft2u.com.tw
第三波	http://www.acertwo.com.tw	敬禮網	http://www.giftgo.com.tw
NBWorld	http://www.nbworld.com.tw	500	http://www.500.com.tw
四方書網	http://www.4book.com.tw	夢想家	http://shop.xwriter.com/
Onlineshop	http://www.onlineshop.com.tw	東森	http://www.etstore.com/
橘子速銷	http://www.orangenet.com.tw/	DG	http://www.dglive.com.tw/
有話好說	http://www.justsayit.com.tw/	僑品	http://shop.store.yahoo.com/
InfoMall	http://www.infomall.com.tw/	Compaq	http://athome.compaq.com.tw/
英達資訊	http://www.go3c.com.tw/	百易網	http://buyeasy.tw.to/
國眾購物	http://www.leomart.com/	萬塔奇	http://www.otic.com.tw/