i-Cube: A Tool-set for the Dynamic Extraction and Integration of Web-Data Content

Frankie Poon Kostas Kontogiannis

{fpoon, kostas}@swen.uwaterloo.ca

Department of Electrical &
Computer Engineering
University of Waterloo
Canada

Abstract

Over the past decade the Internet has evolved into the largest public community in the world. It provides a wealth of data content and services in almost every field of science, technology, medicine, business, leisure, and education just to name a few. However, this exponential growth came at the price of increased complexity for the end-user to categorize, prioritize, and select in a customizable way the information and services that are provided by millions of Web sites across the Internet. This paper presents the i-Cube environment, a toolset that allows for Internet data and content originally available as HTML Web pages and programmatic scripts to be denoted, modeled, and represented in the form of XML documents. These XML documents conform to specific Document Type Definitions and other structural constraints that are fully customizable by the end-user or the service provider. The approach is based on representing HTML document data content in the form of annotated trees. Specific areas of interest and data content in the original HTML document that need be encoded in the form of an XML representation, are represented as a collection of annotated sub-trees in the tree that corresponds to a large HTML document. A service integration module allows for different categories of analysis and presentation rules to be invoked according to script based user-defined logic.

1. Introduction

The i-Cube platform (Figure 1) is an integrated environment that provides the facilities for extracting data content from HTML in the form of a tree data structure, representing the extracted data content in XML format, and mediating the resulted XML data according to a set of lightweight service-logic. The primary goal of the i-Cube platform is to allow rapid deployment of new web-based information services (i-services) that are derived from traditional web applications, through HTML wrapping and data mediation. Secondly, the environment aims to facilitate the creation, integration, and distribution of services in a distributed Web-based environment. In this context, services are related to extraction and processing of Web data content as this can be retrieved by specific service providing Web sites.

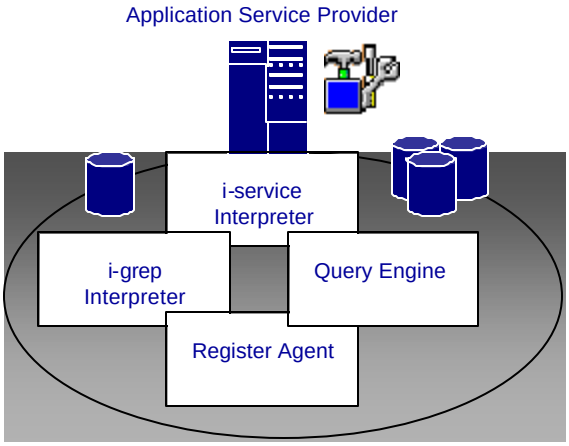

Figure 1. The i-Cube Integrated Environment

The i-Cube platform serves four main functions, namely, the creation, invocation, registration and query of XML-based i-service and i-grep specifications. The above functions are performed by the following five major components of the i-Cube Integrated Environment:

- **i-editor** – a graphical user interface for creating, editing, querying and registering of i-service and i-grep specifications.
- **i-grep Interpreter** (HTML wrapper) - performs extraction from an HTML source and maps the extracted data into an XML data structure, based on a set of rules defined in an i-grep specification.
- **i-service Interpreter** (XML mediator)– interprets service-logic that controls the invocations of i-grep requests and performs mediation on the data sets resulted from the invocations.
- **Query Engine** – responsible for searching according to i-service and i-grep specifications that are based on data content description, data content structure, and data content location.
- **Register Agent** – registers i-service and i-grep specifications composed by certified remote clients in the repositories.

We define an *information grep* (*i-grep*) to be a user-created specification for extracting structured information from an HTML page, where the resulted data is mapped to a XML document. An *i-grep* specification consists of a location of the HTML page, a set of input parameters, an XML Document Type Definition (DTD) [18] for modeling the related data to a Web-site data content structure, and a set of data content extraction rules. An *i-grep* specification is represented in XML in order to facilitate the interchange of specifications between service providers and service builders.

Service specific logic, including the sequence of *i-grep* invocations and the manipulations of data (sorting, set operations etc) are defined in an *information service* (*i-service*) specification. An *i-service* specification can be viewed as a simple application that manipulates data resulted from a set of *i-grep* invocations. The specification is also represented in XML for the inherent data interchange concern. An XSLT [19] document can also be attached to an *i-service* specification for applying presentational transformation on the resulted *i-service* XML data.

We give the notions of *i-service* and *i-grep* to separate the two levels of abstraction, between service specific logics that should be processed by an application and extraction and mapping processes that should be carried out by an HTML wrapper and the mediator.

2. i-Cube Major Components

2.1 The i-editor

The i-Cube editor is a front-end graphical interface that allows users, ranging from web developers to end-users of the web, to create extraction rules, define XML representation of Web data content and associated mapping and processing rules, using a declarative approach. The editor is written in Java, and is essentially an HTML browser with added features for creating *i-service* and *i-grep* specifications. The design goal of this editor is to provide users the exact look-and-feel of their web browser, such that they can easily select information and data content that is of their interest and should be extracted from the HTML page they currently browse.

The editor is based on a parser that parses an HTML document (fetched from an URL) and renders it for displaying on the editor's browser. At that point, a user can create an i-grep specification by highlighting HTML segment of their interest. The screenshot in Figure 2 illustrates the rendering of the HTML source as this is received by the Web-service (in this case the Canadian bookstore Chapters.ca). The significant difference between the editor and a Web browser is that the editor allows the user to highlight specific portions of the rendered page and represent this highlighted segment as a tree based data structure. The editor in operation is depicted in Figure 3 where a highlighted by the user segment is illustrated (the price $63.70). The editor will assist on modeling specific data content in a fully customizable way for the user. Note that once a data entity is modeled, similar HTML pages (i.e. information for another book) that originate from the same Web service provider can be analyzed automatically.

Figure 2. Screenshot of the editor browser.

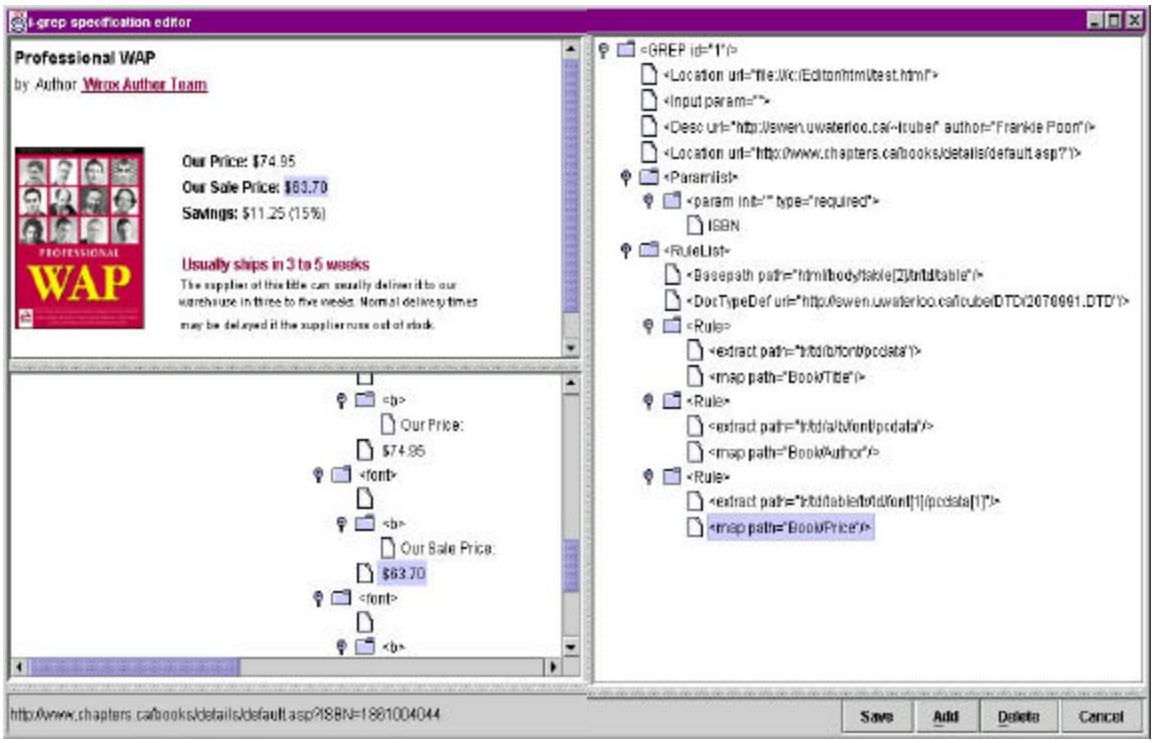

Figure 3. Screenshot of the i-grep editor Window.

The i-grep editor takes two steps in its initialization. First, the URL is parsed to obtain information on the protocol, host, path, application and input parameters required. Second, a *basepath* (see Section 3) is formulated to uniquely locate the selected data content in the original HTML document. This path forms the basis of our HTML *hierarchy-based* extraction approach.

Next comes the declarative process for defining how data elements are to be extracted and mapped. The user can select a text node from the HTML tree and specific a name for the data element (e.g. the text $63.70 that is denoted as a Book/Price data element), which leads to the generation of a <Rule> tuple consisting of an extraction rule and a mapping rule. Based on the *basepath*, an extraction rule with the path "tr/td/table/tr/td/font" is generated and is associated with a mapping rule to the data element "*Book/Price*." Manual editing of the generated rules user is also allowed.

Upon completion of the editing, an XML DTD is inferred from all the specified data elements. The resulted i-grep specification will be saved to a repository locally or through the *register agent* to a remote repository. Note that the inferred DTD will also be stored at the corresponding site, and the i-grep specification will include a reference to this DTD through a URI.

Composed i-grep specifications are added to in the i-grep specification tree in the Editor's main window. The user can then add individual i-grep specifications to an i-service specification, and choose the type of service-logic associated with the i-grep invocations. Data specific logic such as sorting and set operations can be applied on the i-grep result sets. The current prototype requires the user to insert an operation tag (e.g. <Sort>) to encapsulate the set of participating i-grep invocations. The modeled i-service specification can be saved to a local or to a remote repository through the *register agent*. Finally, the editor also provides the user interfaces for querying and registering data content specifications from various sources (see Query Engine and Register Agent Sections for a more detailed discussion).

```
<?xml version="1.0"?>
<GREP id="2078991">
<Desc url="http://swen.uwaterloo.ca/~icube/" author="Frankie Poon">
       Chapters ISBN Search Engine
<Location url="http://www.chapters.ca/books/details/default.asp" type="app"/>
<Paramlist>
       <param init="" type="required">ISBN</param>
</Paramlist>
<RuleList>
       <Basepath path="html/body/table[2]/tr/td/table"/>
       <DocTypeDef uri="http://swen.uwaterloo.ca/icube/DTD/2078991.DTD"/>
       <Rule>
              <extract path="tr/td/b/font/pcdata"/>
              <map     path="Book/Title"/>
       </Rule>
       <Rule>
              <extract path="tr/td/a/b/font/pcdata"/>
              <map     path="Book/Author"/>

       </Rule>
       <Rule>
              <extract path="tr/td/table/tr/td/font[1]/pcdata[1]"/>
              <map path="Book/Price"/>

       </Rule>
       <Rule>
              <extract path="tr/td/a/b/font/pcdata"/>
              <map     path="Book/Delivery"/>
       </Rule>
       <Rule>
              <data type="cdata"/>
                     http://www.chapters.ca
              </data>
              <map  path="Book/URL"/>
       </Rule>
</RuleList>
</GREP>


2078991.DTD
<!DOCTYPE Book [
       <!ELEMENT Book(Name,Author,Price,Delivery)>
       <!ELEMENT Title(#PCDATA)>
       <!ELEMENT Author(#PCDATA)>
       <!ELEMENT Price(#PCDATA)>
       <!ELEMENT Delivery(#PCDATA)>
       <!ELEMENT URL(#CDATA)>
>
```

Figure 4. Example of an i-grep specification and the inferred DTD

2.2 The i-grep Interpreter

The i-grep interpreter is essentially an HTML wrapper that encapsulates the functionality for modeling, extracting, and mapping HTML data content according to an XML-based i-grep specification. The interpreter consists of an Extraction and Mapping Rule Engine (Figure 5). As inherited from an HTML wrapper, its primary responsibility is for retrieving an original HTML document and applying extraction rules on the document. The Mapping Rule Engine then applies the mapping rules for constructing the final XML data structure from the data extracted.

The i-grep interpreter uses an XML parser to parse the i-grep specification, and realizes the locations and input requirement for fetching the HTML page. The HTML page is then processed to output the final XML data structure.

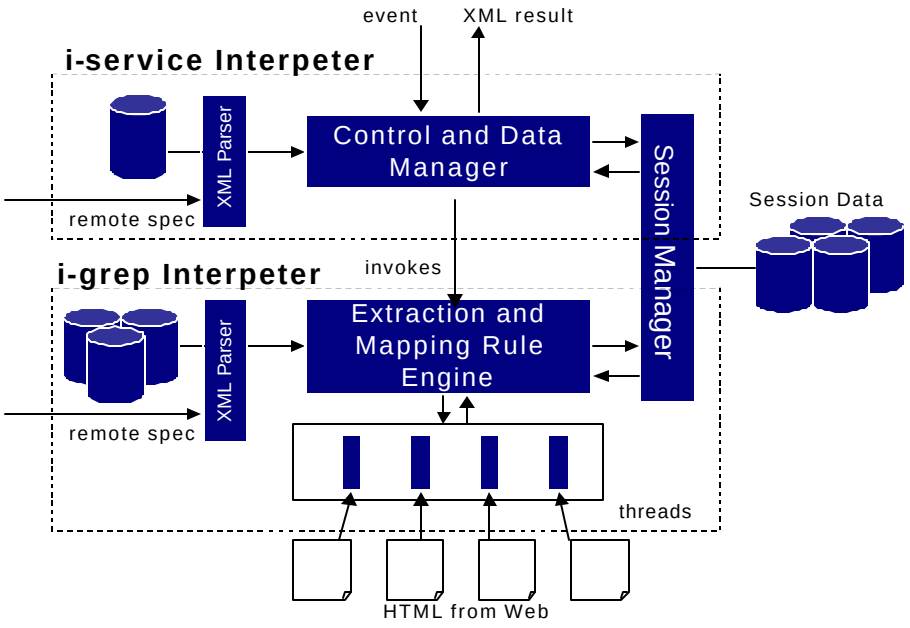

Figure 5. The i-service and i-grep Interpreter

The result set of every i-grep invocation is stored in a session data object, belonging to a particular i-service request, managed by the Session Manager. The i-service interpreter can apply operations to manipulate data among the i-grep result sets stored in the session object. The extraction and mapping process is discussed later in this paper.

2.3 i-service Interpreter

Control Mediation

The i-service Interpreter (Figure 5) is responsible for applying higher-level logic for controlling i-grep invocations, and second, it performs data manipulation on result sets resulted from the invocations. In this way, distributed services located virtually everywhere in the world, can be combined on as required basis using script-based transaction scenarios, forming thus collaborative systems [6], [7]. The customization of the transaction and integration logic required by various processes to complete complex services, opens new opportunities in Web-enabled e-Commerce and e-Business environments. In this sense, business partners can customize their business transaction process models to fit specific needs or, specific contract requirements. This customization is transparent to third parties, and provides means to complete business transactions accurately and on-time. Organizations can enter the e-Business arena by building and deploying extensible and customizable services over the Internet using existing data content that can be delivered over the Web and is readily available as a service over the Internet. Moreover, virtual agencies that provide a wide range of services can be formed by integrating existing functionality and content over the Web. For example, a virtual travel agency can be formed, by composing in a customized manner, services that are readily available in various travel related Internet Web sites. Client processes may post requests to the virtual agency. The agency can enact its transaction logic (scripts) in order to integrate and compose data and services from a wide spectrum of sites. In this scenario, data about pricing, availability and, travel related special offers, can be fetched by various sites, processed by the agency and presented to the client in a customized and competitive way for the agency.

Data Mediation

Every invoked i-grep (HTML wrapping) stores its resulted data in a session object, corresponding to the current i-service request. This allows the i-service interpreter to apply data integration related operations on the result sets, such as sorting and set operations. Our current prototype is able to sort and apply *or* operation to some i-grep result sets. Figure 6 illustrates an example of how a sort operation using in an i-service specification can be defined on the resulting data content (`Book/Price`) extracted from various sites.

```
<?xml version="1.0"?>
<SERVICE id="3453342">
<Desc url="http://swen.uwaterloo.ca/icube/" author="Frankie Poon">
       Meta ISBN Book Search Service
</Desc>
<Location url="http://www.chapters.ca/books/details/default.asp"
type="app"/>
<Paramlist>
       <param type="required" initval="">ISBN</param>
</Paramlist>
<Sort order="ascend" key="Book/Price">
<DocTypeDef uri="http://swen.uwaterloo.ca/icube/DTD/3453342.DTD">
       <grep id="2078991"/>
       <grep id="4581155"/>
       <grep id="901546"/>
       <grep id="24159757"/>
</Sort>
</SERVICE>
```

Figure 6. Example of an i-service specification

In addition to manipulating i-grep result sets, the i-service interpreter has a special feature for *promoting* a particular i-grep service. Section 2.6 discusses in more detail the different roles of i-Cube clients. In short, this feature gives end-users the flexibility of invoking more formally defined i-services and i-grep specifications, created by either the service provider or the original web site. This feature involves sending a query request to registered i-Cube platforms, specifically to their Query Engine, for finding services that match a particular data structure, location and description. Once a speciation is matched, the i-service interpreter either pulls the new specification from the remote site, or requests the invocation of the specification at the remote site.

2.4 Query Engine

The Query Engine of the i-Cube platform provides a searching facility for specifications that match the description, data structure and URL specified in a query request. This module is beneficial to end-users who wish to reuse existing specifications. Moreover, end-users can look for a more formal specification, either at the service provider site or the original web site, to replace the one that they have defined previously.

Specifications that match the query constraints are either returned to the user as an XML document, or invoked at the i-Cube platform where the specification resides. In general, the Query Engine offers two classes of query services. *Explicit Querying* is used by end-users who wish to compose their own i-service specifications and who want to know if formal specifications have been previously defined by the service providers or the original web site. This type of querying is performed by end-users through the Editor user interface.

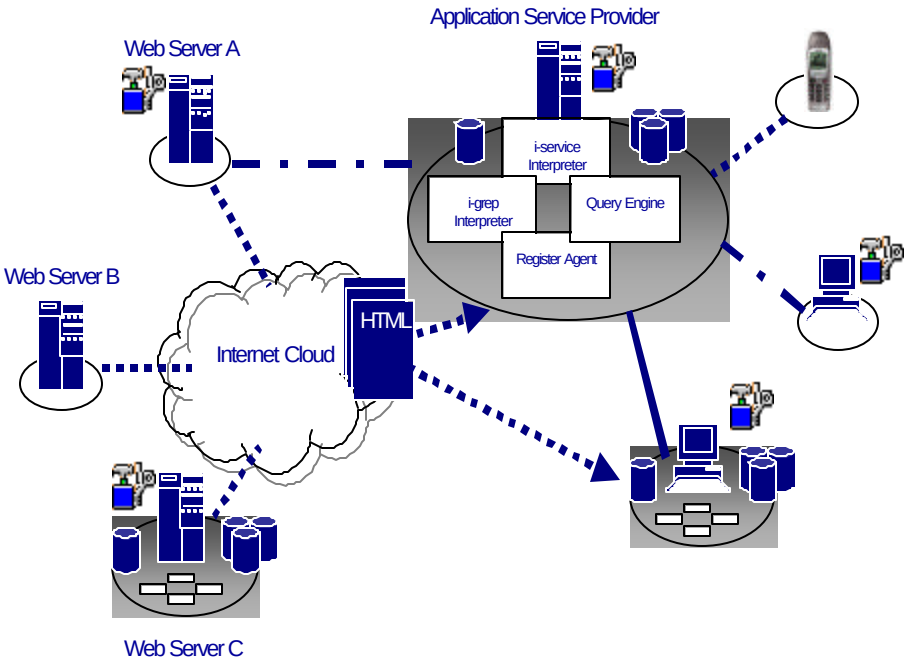

Figure 7. i-Cube clients and their operating environments

The second type of querying service is called *Transparent Querying*, which is primarily used by the i-service interpreter. Composers of i-service specifications can indicate that they would like to invoke more formal i-grep specifications defined at the service provider (or original web site), if such specifications exist. A query request is sent to the corresponding remote query engine, if a specification exists at the remote site, then the interpreter will discard the local specification and invoke the remote one. This scheme is also known as *promoting* as mentioned in previous section.

2.5 Register Agent

The Register Agent provides an interface for remote certified i-Cube clients to register composed i-service and i-grep specifications to the repositories. This allows clients who do not possess the full i-Cube platform to submit customized i-Cube specifications to a service provider. The service provider in turn, will provide the application platform for processing their specifications upon request of the owner of the specifications.

2.6 Roles of i-Cube clients

We now focus on demonstrating the different roles that are played by different i-Cube clients and their corresponding operational environments. In Figure 7, three types of i-Cube clients, service provider, end-users and original web sites are illustrated.

Service provider

A *service provider* is a dedicated i-Cube platform that offers i-Cube based services to end-users. In addition to enhanced web services that it developed for end-users, it also allows i-Cube certified clients to register their composed specifications through the register agent, providing them an application platform for invoking these services. In the figure, the application service provider that offers enhanced web services to end-users represents the *service provider*. The service provider consists of all five of the i-Cube components described in the previous section.

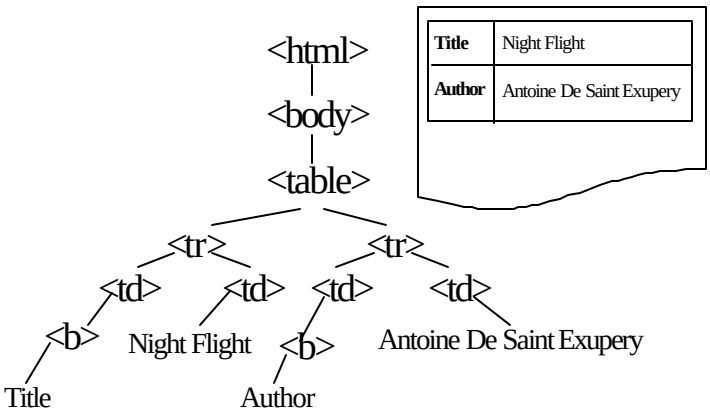

Figure 8. An Example of HTML hierarchy

Internet clients such as handheld devices and desktop users can use the wide selections of enhanced web services offered at the site, which are represented as a typical HTML page. An example will be an HTML form that contains a Textbox and a Submit button for an enhanced search engine of books from various online bookstores where the form corresponds to a particular i-service specification. Once the user has submitted the HTTP request, the service provider carries out the appropriate service-logic, invokes the corresponding i-grep specifications, processes the result sets, and presents the resulted data in HTML format. Following the previous book example site, it means that the list of book titles is returned to the user as an HTML page.

Web-Site

A *web site* is where the HTML information originates. For non i-Cube enabled web sites, it is transparent whether enhanced web services have been derived from their web applications. It is certain that if enhanced services are to put into a commercial use, legal agreement must be established between the service provider and the web site. On the other hand, for web sites that provide their own enhanced web services to the users, but do not wish to apply major changes to their existing infrastructure, they can adopt the i-Cube platform as the wrapping agent for rapid deployment of new web-services. Web sites can also compose their own specifications and register them at the service provider site, making the services available to a wide group of end-users.

End-User

End-users correspond to the users of these enhanced web-services. In the figure, the handheld device and the desktops are classified as end-users. Typical clients, like desktop users (non i-Cube enabled) can use these services by navigating on the service provider web site, where different selections of web-services are presented to them either as hyperlinks or input forms, in HTML format. The type of services will be transparent to the clients, in terms of whether they are i-Cube based or not.

On the other hand, i-Cube-enabled clients are permitted to compose their own specifications or build on existing specifications offered by service providers and web sites. They can either register their composed specifications at an i-Cube-enabled service provider, or invoke these services locally.

3. Extracting XML data structure from HTML

Implementation of the i-grep Interpreter

The i-grep Interpreter uses an XML parser to read information contained in an i-grep specification. It forms a complete URL request by attaching the URL and the input parameters, if appropriate, for fetching an HTML document. To assure that the document is well formed, it is being cleaned up using the combination of a HTML parser and HTML Editor Toolkit from the Java Swing class. Then according to the extraction

rules specified, the extraction rule engine extracts information, and maps each piece of information to the corresponding element in the XML data structure.

HTML hierarchy

An HTML document is a structured document that contains text and hyperlink data stored under different presentational tags. An HTML document can be parsed by an HTML parser to construct a DOM tree that corresponds to its HTML hierarchy, such as the one in Figure 8. Each node in the figure corresponds to an HTML tag, where each leaf node corresponds to a PCDATA node for text chunk, or an empty tag such as "
" or "<IMG>". The significance of this tree hierarchy is that an HTML *tree path* can be derived to locate a particular node in the hierarchy.

Extraction Approach

Our extraction approach takes advantage of the tree hierarchical structure that is used for denoting HTML data content in a Web page in order to localize and extract specific data elements from the HTML document representing such a page. We first define the concept of an *absolute HTML tree path* to be a path for locating a specific node on a HTML tree. This path adopts similar features in XPath [20] to locate elements on a HTML tree, starting from the root node to a specific node on the tree. Using a Document Object Modeling (DOM) implementation, it is straight forward to navigate on the HTML tree along the path, "*html/body/table/tr[0]/td[1]/pcdata*," in order to obtain the text data "Night Flight" from the HTML document in Figure 8.

Each HTML element that should be traversed is separated by the character "/" to represent a parent-to-child relationship in the tree hierarchy. Identical children are indexed with respect to their sequential appearances at the parent node. The syntax "element[id]" is used for this purpose to indicate a child node at a specific index, such as td[0], td[1], td[2], etc.

Once the corresponding HTML node is located, three types of data can be chosen for extraction: PCDATA (Parsed Character DATA), CDATA (Non-parsed Character DATA) and attribute values. They are specified at the last element in the path as "*pcdata*", "*cdata*", and "*element(@attributename)*."

It should be noted that the HTML hierarchical path alone is only effective at extracting static HTML sources that are not subject to change, and sources with no repeating data structures (e.g. 7 matches resulted from a book title query). To extend our extraction approach – such that it can provide some degree of flexibility when extracting data from a modified source, and can identify and extract one or more data structures contained within the HTML document -- we include the notion of *keyword definition*.

Keywords are string constants associated with a data structure that carry some contextual meanings and are defined during the creation time of i-grep specification. For every keyword defined, its corresponding HTML hierarchical path, with reference to the *basepath*, is computed and a conditional expression is formulated based on the keyword constant "*element/pcdata=keyword*."

Extraction Process

Figure 9 illustrates the rule listing of an i-grep specification that corresponds to extracting the data content from Figure 8. The extractor first locates the *basenode* that contains the descendants constituting the data structure, which is specified by the *basepath*.

After locating the *basenode*, the extractor applies a validation process to examine whether the structure located under this node is valid according to the two criteria: HTML hierarchy and keyword definitions. Each data to be extracted is defined as an individual <Rule> tuple consisting of the HTML hierarchy and one or more keyword definitions. If the extractor has successfully located the keywords and the data according to the HTML hierarchy, then it proceeds to the next <Rule> tuple. If however, the extractor has failed to locate the *basenode* in the initial stage, or failed to locate the HTML hierarchy and keywords associated with the data structure, then it will declare the process as a failure.

A recovery phase will proceed in the event of a failure. The recovery phase assumes the existence of the data structure -- that is, it preserves the rules contained in a <Rule> tuple -- but under a different *basepath.*

The extractor then begins a search on the HTML tree, starting among the sibling nodes of the base node, in an attempt to locate the same data structure specified. The extractor continues to search at the ancestors and descendants one level at a time, starting at the *basenode* level, until a matching structure can be found.

```
<RuleList>
        <Basepath path="html/body/table[0]"/>
        <Rule>
                <extract from="tr[1]/td[1]/pcdata">
                <keyword from="tr/td[0]=Title">
                <map to="Book/Title">
        </Rule>
        <Rule>
                <extract from="tr[2]/td[1]/pcdata">
                <keyword from="tr[2]/td[0]=Author">
                <map to="Book/Author">

        </Rule>
</RuleList>
```

Figure 9. The rule listing of an I-grep specification

A successful recovery phase updates the specification by inserting a new *basepath* to the rule list. This provides a recommended alternative for the extractor to locate the data structure before a recovery phase should take place. The new *basepath* will become the primary path while the original one will become the secondary of the searching criteria.

In many cases in addition to using the exact HTML hierarchy to locate specific data content that is to be extracted, it is also necessary to match specific larger portions of the HTML structure. This requires techniques in applying approximate or partial pattern matching that can be based on regular expressions or stochastic techniques. Work on regular expression based pattern matching has been investigated within the context of bio-informatics, image processing, and in the area of compiler optimization [1], [5], [14].

Mapping to an XML DTD

Once the data structure is successfully extracted, it is assigned to an interchangeable format. For this purpose, we adopt Extensible Markup Language (XML) as the interchangeable format for the extracted data. Mapping rules are created at i-grep specification creation time. Similar to our extracting rules, we use a notation analogous to XPath to designate the mapping relationship between the extracted data set and the corresponding XML Document Type Definition (DTD) that models the domain at the specific Web site and service (i.e. bookstore, travel agency).

A mapping rule corresponds to a many-to-one relationship between extracted data and an element in the XML DTD. It can contain one or more extraction rules, where all the extracted data (string based) are concatenated together, with each separated by a white space. Following the example, a rule for mapping the extracted book name to the <Name> element in the DTD is illustrated in Figure 9.

4. Applications

4.1 Building a Meta Book Search Engine.

We now illustrate a specific example of using our integrated environment for building a meta book search engine -- a service for comparing the prices of a book from different online bookstores. Using the editor, any user can create an i-service based on a set of i-grep specifications for checking the price of a book from various online bookstores. Most online bookstores offer a service for searching a book title based on its

Figure 10. ISBN search results for other online bookstores

ISBN. To simplify our example, let's assume our meta search engine offers the same service, but for five different web sites.

From the editor's browser, we visit an online bookstore, for example, Amazon.com and use its advanced search engine to look for a specific book title. The page showing in Figure 3 is actually the result of submitting the ISBN query "1861004044" on its advanced search engine facility, which displays the book title "Professional WAP" and its related information in HTML. In this context, we are interested in creating a service that displays the *name* of the bookstore, the *title*, *price*, *availability* of the book as well as an URL link to the result page, in order to achieve so, we highlight on the browser the portion that corresponds to these information and create an I-grep specification based on the selection.

The editor then brings us to a declarative process for defining the data structure that corresponds to the selected portion as shown in Figure 4. As an example, we define the element *title* and associate this element with the text "Professional WAP" contained in that page. The editor, in turn, will generate the corresponding extraction and mapping rules for this data element. We can fill in the *name* element with the appropriate bookstore name. A small description about this specification can also be included, like "search book title using ISBN."

Since all ISBN searches for books on a specific site will generate HTML documents with identical structure, this mapping will work for all HTML pages generated as a result of an ISBN book search. It becomes then apparent how this process can be applied similarly to other online bookstores as in Figure 9. The result is a set of I-grep specifications that takes a common input parameter (ISBN) and its output conforms to our book information domain model format Book:{bookstore,title,author,price,delivery,url}. This is feasible because the mapping conforms to a domain model (i.e. DTD) for a given type of service (i.e. book store) results obtained from various other sites can also be mapped to the same domain model and DTD. This allows for direct algorithmic processing such as sorting the obtained results by a specific field. In this context, domain models and DTDs for other services such as airline ticketing or B2B transactions can be easily built and customized to accommodate specific post-processing requirements.

Your Results: ISBN 1861004044

Store	Name	Author	Price	Availability
Fatbrain.com	Professional WAP	Charles Arehart, [more]...	$41.95	In Stock - Orders received by 4p.m. ship the same day
bamm.com	Professional WAP	Charles Arehart, [more]...	$43.19	In Stock: Ships with 2-3 days [more]...
BN.com	Professional WAP	Charles Arehart, [more]...	$47.99	In Stock: 24 hours (Same Day) [more]...
Amazon.com	Professional WAP	Charles Arehart, [more]...	$47.99	Usually ships within 24 hours.
Borders.com	Professional WAP	Wrox Press Inc., [more]...	$50.00	This title usually ships in 24 hours.

Figure 11. Book Price Comparison Results

The next step in the process is to compose an i-service specification based on these i-grep specifications, such that we can apply control logic to the i-grep invocations and also apply data manipulating operations such as sorting in this case, on the resulted i-grep data sets. All the specifications created are then registered at the service provider's repositories.

To access the new service, assuming we have created a very simple HTML entry page that contains an HTML form that consis ts of a Textbox and a Submit button. We can type in the desired ISBN number and perform a meta search on all the associated online bookstores. The backend processing will involve fetching the corresponding i-service specification and associated i-grep specifications and invoking them respectively. Each i-grep invocation will extract the book data from a HTML web site, based on an ISBN, which is then stored in a Session Data Object at the remote site. The result sets are then sorted according to their prices as defined in the i-service specification; the final data structure is an XML document that encapsulates all the book data called Booklist:{Book}.
A default XSLT template is provided by the service provider for transforming the resulted XML data set into HTML, then we will be able to see the list of books, including the bookstore, title, author, price, availability and an URL link to the bookstore presented in a HTML table format as shown in Figure 10.

4.2 Notification Service.

A service can be created for monitoring changes on a web site [13], and the resulted information can be sent to the end-user through email, or other types of notification mechanism such as the PUSH mechanism in WAP [17]. To illustrate a practical example, assume an online library book search system that returns to the user the information of a book, including its name, author and status. A book that is currently on loan will display that status information: "*Status: On Loan*," user can make use of this information and define a service for monitoring changes of the status until the book becomes available, for example "*Status: In Library*." At that time, the service will compose a notification email indicating the availability of the book. All it requires is that the service periodically analyze the HTML page for the user.

4.3 Migrating from HTML to VoiceXML and WML applications.

The introduction of new presentational markup languages not only provides tailored presentational means for different types of devices, but also at the same time aims to attract a new domain of web users. VoiceXML [16], for example, aims to attract users who are familiar with interactive voice system; similarly, WML is designed for mobile handsets that adopt the WAP standard [17]. To migrate existing HTML pages into these markup languages, it is possible to use transformational language such as XSLT (eXtensible Stylesheet Language for Transformation) to define one-to-one, or one-to many mapping between the elements in two markup languages. However, it becomes a very difficult task for the XSL

developer to write rules for the rather long and complex structure of HTML. It will be ideal if significant segment of the HTML page is represented in a XML format that carries contextual meaning (data structure) using the i-Cube editor, such that the XSL development process will become more obvious to the developer.

5. Related Work

Different approaches for extracting information from HTML sources have been recently developed. While many approaches focus on the definition of a formal and expressive extraction grammar [8], [11], and [12], others [2][3] focus on an SQL-like language for querying semi-structured information from HTML sources. The inherent problematic nature of writing extraction rules manually has motivated the development of graphical tool to assist the generation of extraction rules. In This context, W4F [15] uses a similar hierarchical-based navigation approach for extracting structure from HTML. Its extraction wizard provides a graphical interface for visualization of annotated HTML segment to assist the writing of extraction rules.

Other approaches take a step further to enhance the extraction rule generation process by incorporating an inductive learning mechanism [9] and using a two-phase code generation framework for the wrapper generation process. A micro-feedback approach is also used in order to customize the generated wrapper at runtime.

It should be noted that many of the explored approaches taken, like those presented in [4][9] and [12], are aimed at the generation of a set of extraction rules that is fed to a code generator for building a standalone wrapper application. Our approach is different in that we focus on the building of an interpreter that understands extraction and mapping specification, in addition to the location of the HTML pages. We adopt the XML technology in our interpreter design to encourage the exchange of wrapper specifications over the Internet, which in turn, allows derived information services to expose to a wider domain of web users. Finally, work that deals with control integration aspects of various distributed services using scripting languages, such as the Event-Condition-Action (ECA) framework has been presented in [10].

6. Conclusion

In this paper we have presented the i-Cube integrated environment and demonstrated its capabilities for rapid deployment of web-based information services to end-users by wrapping existing HTML pages and reusing data structure that they provide. We have also presented the extraction and mapping rules that are used for precisely extracting data from HTML into a corresponding XML data structure. In particular, the extraction rules that we propose make use of the inherent tree characteristic of HTML hierarchy, which allows for the denotation and localization of data content based on an HTML tree path. At the service level, a prototype system that allows for the definition of the control logic of invoking i-grep specifications and operations that pertain to specific data mediation processes.

The integrated environment provides a set of tools for creating, customizing, invoking, registering and querying of composed specifications. This makes i-Cube an environment, not only suitable for the commercial Web sites, but also suitable for service providers who wish to provide to end-users a new domain of web services derived from existing ones. Moreover, end-users can benefit from using the tool for creating customized web services, which works in conjunction with a registering mechanism for storing their composed specifications at a remote service provider, serving as an application platform for the end-users.

From an end-user perspective, the environment provides a common platform for deploying customizable web services tailored to user's own needs. Existing web systems, on the other hand, can easily realize the added benefit of deriving and deploying new domain of web services without the expense of applying changes in their existing infrastructure. Finally, the service provider who acts as the broker of the entire architecture plays the important role of driving the process of distributing web services among web systems and end-users.

References

[1] Aho, A., Ganapathi, M., Tjiang, S., “Code Generation Using Tree Matching and Dynamic Programming" ACM Transactions on Programming Languages and Systems, vol. 11, No. 4, October 1989, Pages 491-516.

[2] G. Arocena, A. Mendelzon, G. Mihaila. “Applications of a Web Query language,” Proceedings of the 6th International WWW Conference, Santa Clara, California, April 1997.

[3] Gustavo Arocena and Alberto Mendelzon. “WebOQL: Restructuring Documents, Databases, and Webs,” In Proceeding of the SIGMOD Conference, Seattle, June 1998.

[4] N. Ashish and C. Knoblock. “Wrapper Generation for Semi-structured Internet Sources.” Workshop on Management of Semistructured Data. Ventana Canyon Resort, Tucson, Arizona.

[5] B. S. Baker, ``Parameterized Pattern Matching:Algorithms and Applications", Journal Computer and System Sciences, 1994.

[6] J.A.Bergstra, P.Klint, “The Discrete Time ToolBus,” http://adam.wins.uva.nl/\~olivierp/toolbus/index.html, February 1995

[7] Control and Coordination of Complex Distributed Services (C3DS), http://www.newcastle.research.ec.org/c3ds

[8] A. Gal, S. Kerr, J. Mylopoulos “Information Services for the Web: Building and Maintaining Domain Models", International Journal of Cooperative Information Systems, 8(4):227-254, 1999.

[9] Ling Giu, Calton Pu, Wei Iian. “XWRAP: An XML-enabled Wrapper Construction System for Web Information Sources.”

[10] Richard Gregory, Kostas Kontogiannis, “Customizable Integration in Web-enabled Environments,” Lecture Notes in Computer Science, Special Issue on Engineering Distributed Objects (to appear).

[11] J.Hammer, H.Garcia-Molina, J.Cho, R.Aranha, and A.Crespo. “Extracting Semisructured Information from the Web,” In Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, 1997.

[12] Gerald Huck, Peter Fankhauser, Karl Aberer, and Erich J.Neuhold, “JEDI: Extracting and Synthesizing Information from the Web,” In COOPIS, New-York, 1998.

[13] Ling Liu, Calton Pu, Wei Tang, Dave Buttler, John Biggs, Paul Benninghoff, Wei Han, Fenghua Yu. `` CQ: A Personalized Update Monitoring Toolkit". Proceedings of the ACM SIGMOD, May, 1998.

[14] Myers, E., Miller W. "Approximate Matching of Regular Expressions", Bulletin of Mathematical Biology, Vol.51, No.1, 1989, pp.5-37.

[15] A. Sahuguet and F.Azavant. “Wysiwyg Web Wrapper Factory (W4F),” In Proceedings of WWW Conference, 1999.

[16] VoiceXML Forum, “VoiceXML Specification 1.0,” 2000. URL: http://www.voicexml.org/

[17] WAP Forum, “Wireless Application Protocol Architecture Specification”, 1998. URL:

[18] World Wide Web Consortium (W3C), “Extensible Markup Language (XML) Version 1.0,” 1998.

[19] World Wide Web Consortium (W3C), “Extensible Markup Language (XSL) Version 1.0,” 1998.

[20] World Wide Web Consortium (W3C), “XML Path Language (XPath) Version 1.0,” 1999. http://www.wapforum.org/

[21] Ying Zou, Kostas Kontogiannis, “Web Based Specification and Integration of Legacy Services,” CASCON 2000, Toronto, 2000.