1. Summary of Achievements to date

The Internet and the World Wide Web (Web) have become one of the most popular means for disseminating scientific data from a variety of disciplines. A critical challenge for enabling scientists to access scientific data sources is the problem of locating, accessing, and fusing information from a rapidly growing, heterogeneous, and distributed collection of data sources available on the Web [BCC02].

In SciDAC SDM project, the main assignment to the Georgia Institute of Technology team (according to the proposed work) is to develop advanced information extraction and information integration technologies on top of the XWRAP technology originated from Georgia Tech [LPH01]. We have developed XWRAPComposer technology to enable the XWRAP code generator to generate Java information wrappers that are capable of extraction of data from multiple linked pages. These information wrappers are used as gateways or adaptors for scientific information mediators to access and fuse interesting data and answering complex queries over a large collection of heterogeneous scientific information sources.

Our accomplishments over the SciDAC sponsored years (July 2001 to July 2004) can be summarized along two dimensions. Technically, we have produced a number of major software releases and published over 30 research papers in both international conferences and international journals. The planned software releases include

1. Five Java wrappers and five WDSL-enabled wrappers for SDM Pilot scenarios, which were released in early 2003,
2. The XWRAPComposer toolkit (command line version) which was first released in late 2003 and then released in Summer 2004,
3. Five Ptolemy wrapper actors which were released first in Summer 2003, and then released again in Fall 2005.
4. The decomposable XWRAPComposer actor in Ptolemy, which we have made it available as open source in end of 2004 and tested it in early 2005.

The main objective of the XWRAPComposer toolkit is to produce an automated end-to-end wrapper code generator that takes a generic description of a service class of search/service interfaces and the URL of a particular interface that is a member of class, and produces a functional wrapper that will take a service class specific query and produce XML results derived from the query results of the particular interface.

In addition, we have made effort to broaden the impact of our research results to wider audiences by offering advanced seminars and research presentations at conference panels, giving invited talks, and visiting more than 20 universities and industry or government labs. There are two distinct features of the XWRAPComposer toolkit: (1) To our knowledge, XWRAPComposer is the first and only wrapper code generator today that is capable of generating wrappers to wrap data from multiple pages. (2) Java Wrappers generated by XWRAPComposer is capable to exhibit information extraction workflow logic. Therefore, each wrapper can be viewed as a multi-page data extraction workflow program generated by the XWRAPComposer. Over the course of this sponsored research we have published results in six international journals and 30 publications in refereed conferences. We have also released the XWRAPComposer software at sourceforge.com and online.

**Personnel Training.** In addition to research and software releases, through this sponsored research we have trained and produced excellent personnel. Three PhD students who are partially funded by this project have been graduated:

- David Buttler defended his PhD dissertation and joined the DoE Lab – Lawrance Livermore National Laboratory (LLNL) in Sept 2003, continuing working on various DoE projects;
- Dan Rocco defended his dissertation research in August 2004 and joined the Computer Science Department at University of West Georgia as a tenure-track faculty, continuing his research on deploying the XWRAP Composer technology to dynamically discovering and categorizing Bioinformatic Web sources;
- Wei Han defended his dissertation research in May 2003 and joined IBM Almaden Research Center, continuing his expertise in building code generation systems.

The Georgia Tech SDM website is at [http://disl.cc.gatech.edu/SDM/](http://disl.cc.gatech.edu/SDM/). The XWRAP Composer code generator (command-line version) can be downloaded at: [http://disl.cc.gatech.edu/XWRAPComposer/](http://disl.cc.gatech.edu/XWRAPComposer/).

## 2. XWRAP Composer Overview

XWRAP systems aim at semi-automatically generate Java code (Wrapper) for extracting useful content from Web pages. XWRAP Elite is designed to generate wrapper programs for extracting information (primarily query-answers) from the deep Web data sources. XWRAP Composer is designed on top of XWRAP Elite, aiming at providing information extraction across multiple linked Web pages. A typical example is the Matt’s scenario [Liu+05].
The XWRAP Composer requires three types of input descriptions to generate the Java code of the given Web source.

- The first input description is called Interface Description which is used to specify the URL of the Web source to be wrapped, and the keyword or entry query used to obtain the pages from the search interface of the given web source.
- The second input description is called Output Service Class Description. It describes the XML tagging information and how extracted data content can be tagged as specified.
- The third input parameter is the Composer Script Language, which consists of both the QA control logic specification and the data extraction logic required. The QA control logic specifies the possible control flows of the search interface of a given web site.

An example of such control logic is as follows. The NCBI BLAST interface will respond to a BLAST query with at least three different control flows: the first one is the waiting page which tells how long one needs to wait for the results to be returned. The second one is the full result page which is returned by following the indirection link on the intermediate waiting page. The third one is the error page which is returned when following the indirection link on the waiting page. Thus the Composer wrapper needs to generate code pieces to handle all three cases.

3. Generating XWrap Composer Script

The generated XWrap Composer script follows the interface specification laid out in the attached specification document, and must conform to the schema definition. Two versions of the schema definition are provided: one in RelaxNG compact syntax for easier readability, and a standard XSD description generated from the compact syntax. For tools to make the translation, see http://thaiopensource.com/relaxng/.

The following figure gives an example of how the DynaBot service classifier generate the Xwrap Composer script for web site http://fugu.hgmp.mrc.ac.uk/blast/
The generated Xwrap Composer script is given in the following figure.

```xml
- <InputSchema>
  - <xsd:schema>
    - <xsd:complexType name="input">
      <xsd:element name="sequence"/>
    </xsd:complexType>
  </xsd:schema>
- <OutputSchema>
  - <xsd:schema>
    - <xsd:complexType name="Alignment" type="xsd:string">
      <xsd:complexType maxOccurs="2" minOccurs="1">
        <xsd:element name="taxa" type="xsd:string"/>
        <xsd:element name="score" type="xsd:string"/>
        <xsd:element name="hitLen" type="xsd:string"/>
      </xsd:complexType>
    </xsd:complexType>
  </xsd:schema>
</OutputSchema>
</InputSchema>
<ExtractionTasks>
  <ReadFile datatype="XML" outputid="serviceClassDescription">
    <FileName>../alignment.scd</FileName>
  </ReadFile>
  <HttpFormQuery outputid="formQuery" method="POST">
    <urlBase>http://fugu.hgmp.mrc.ac.uk/blast/</urlBase>
    <parameter name="OOF_ALIGNMENT" value="8"/>
  </HttpFormQuery>
</ExtractionTasks>
```

Figure 2 Xwrap Composer Wrapper Script
4. Automated Wrapper Generation and Registration

The XwrapComposer compiles wrapper interface, outerface and extraction script into an XWrapComposer wrapper, which contains an executable Java program and a set of configuration files. The configuration files include the input and output schema obtained from interface and outerface, resource files used in the data extraction phase such as XSLT files. The standard XwrapComposer toolset provides extensions of the generated wrappers for application integration in two forms, i.e., the Web-service Enabled Wrappers, and the Ptolemy II actors.

To assist the usage of Xwrap Composer wrappers and simplify the wrapper generation and management overhead, we designed an online wrapper generation toolset that allows DynaBot users to generate and register Xwrap Composer wrappers from the wrapper scripts that they obtain from the DynaBot service classifier. This toolset is available at http://disl.cc.gatech.edu/LDRD/html/protected/repository.htm

We use the following work through example to show how this toolset works. The example web site we use is http://fugu.hgmp.mrc.ac.uk/blast/. It provides a standard BLAST interface. After obtain the Xwrap Composer wrapper script as shown in Figure 2, the user can upload this script through the following interface, available at http://disl.cc.gatech.edu/lrdscript/html/registerwrapper.htm

![Figure 3 Xwrap Composer wrapper generation interface.](image-url)
The generated wrapper will be stored online in our wrapper repository. So far, we have generated 51 wrappers for the project demo. The following figure gives an overview of our wrapper repository.

<table>
<thead>
<tr>
<th>Wrapper ID</th>
<th>Wrapper Name</th>
<th>Author Info</th>
<th>Source URL</th>
<th>Release Date</th>
<th>Version</th>
<th>Service Description</th>
<th>Wrapper Script</th>
<th>Wrapper Code</th>
<th>Wrapper Code Execution</th>
<th>Execution Snapshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>1021</td>
<td>bigx</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1020</td>
<td>alien</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1023</td>
<td>scififord</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1024</td>
<td>saintdois</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1025</td>
<td>aph anxious</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1026</td>
<td>aph anxious</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1027</td>
<td>peep</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1028</td>
<td>bineb</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1029</td>
<td>harvard_rebar</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1030</td>
<td>harvard.apps</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1031</td>
<td>geneopedia</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1032</td>
<td>ou_slot</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1033</td>
<td>ou_so_par</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1034</td>
<td>columbia</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1035</td>
<td>indico</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1036</td>
<td>ou_finance</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1037</td>
<td>ou_soon</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1038</td>
<td>ou_so_bib</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1039</td>
<td>bocombio</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1040</td>
<td>souarxiv</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
<tr>
<td>1041</td>
<td>ou_so_swape</td>
<td>Damien Cocks</td>
<td>[URL]</td>
<td>07/06/2004</td>
<td>1.0</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
<td>raw</td>
</tr>
</tbody>
</table>

Figure 4 Wrapper Repository

The generated wrapper source code can be downloaded from this interface. Using the library and scripts that we released, this wrapper source code can be compiled on users’ local machines and executed as command line Java application. The following figure gives the example of the wrapper source code.
Our online toolset also provide a wrapper execution interface that allows the user to execute those registered wrappers. This wrapper execution interface abstract the service provided by the BLASTN websites by presenting a uniform interface to the end users. The end user can choose to monitor the wrapper execution in details or only check the query result.

The following figure gives an overview of this interface.
The following is an example of the detailed wrapper execution trace.
The query result will be parsed by the service class description library to extract the information of interest to the user. In our example, the gene alignment information is extracted. Example is given in the following figure.

**Figure 7 Wrapper execution trace**
5. Wrapper Generation and Execution in Ptolemy II Environment

Ptolemy is a modeling tool to assemble concurrent components, providing a friendly GUI environment to connect task components and govern the interaction between them. The task components have to conform to Ptolemy’s actor interface in order to be integrated in Ptolemy.

We have design and implemented a wrapper generation and execution environment under the Ptolemy II environment such that uses can use their local resources to generate and execution Xwrap Composer wrappers without relying on our online toolset and wrapper repository.

The following figure presents the outlook of our wrapper generation and execution interface under Ptolemy II.
The execution result will be displayed as XLM document, as showed in the following example.

6. Borad Impact of SDM-GT XWRAP Composer development
We have obtained one year extension at the end of the third year (August 14, 2004) for our SciDAC SDM project award from DoE SciDAC program. In the extended period, we dedicate our efforts on the following tasks.

(1) We have completed the final software release using Ptolemy platform chosen by the SDM project team for the integration efforts. We have included the screen shots of our XWRAP-Ptolemy system on our Web site at http://disl.cc.gatech.edu/XWRAPComposer/.

(2) We have created a more scientists-friendly interface, released the XWRAPComposer as an open source software, and marketed it as one of the unique products offered by the SciDAC SDM project. One can download the final release at http://disl.cc.gatech.edu/XWRAPComposer/ or at sourceforge.com.

(3) We have created a case study to show scientists the benefits and the potential usage of the XWRAPComposer as an effective SciDAC SDM enabling technology. This case study includes the production of 100+ BLAST wrappers generated from XWRAPComposer code generator system, the integrated search interface for scientists to access over 100+ BLAST web sources through one click interface, the workflow steps and intermediate results produced within each information extraction and wrapping process, and a user study. The case study suite and the associated prototype system can be found at http://disl.cc.gatech.edu/LDRD/.

Dan Rocco was the lead student working on the case study with the assistance of Jianjun Zhang, another PhD student at the College of Computing, Georgia Tech, scheduled to graduate in 2007. We have produced exciting results and the document reporting this case study will appear in the International Conference on Web Services (ICWS 2005) in Orlando this July. Our study shows that the SDM-XWRAP development has made important impact on both the Web Services research and the research in Bioinformatics engineering. The software development we have produced under DoE SciDAC program has been used in classrooms at the Georgia Tech campus and been downloaded by several other institutions worldwide.

**Personnel.** Georgia Tech SciDAC SDM team consists of two faculty members: Prof Ling Liu and Prof. Calton Pu, and a number of PhD graduate students: David Buttlar, James Caverlee, Wei Han, Dan Rocco, Keke Chen and Jianjun Zhang. The amount of support received by the listed students under SciDAC SDM ranges from 3 months to 12 months, depending on the amount of work and research contributions they can make and the level of relevance of their research results to the SciDAC SDM objectives.

**Publication of results from SciDAC support:**


23. [HBP01] Wei Han, David Buttler, Calton Pu. Wrapping Web Data into XML SIGMOD Record, July 2001