AutoForms: a tool to enable rapid application development of form-based, web database applications

Shamsul Arefin Shaikh
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd

Recommended Citation
https://lib.dr.iastate.edu/rtd/20036

This Thesis is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
AutoForms: a tool to enable rapid application development of form-based, web database applications

by

Shamsul Arefin Shaikh

A thesis submitted to the graduate faculty in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Simanta Mitra, Co-major Professor
Leslie L Miller, Co-major Professor
Manimaran Govindarasu

Iowa State University

Ames, Iowa

2003

© Shamsul Arefin Shaikh, 2003. All rights reserved.
Graduate College
Iowa State University

This is to certify that the master's thesis of

Shamsul Arefin Shaikh

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy
IN THE NAME OF ALLAH,  
THE MOST GRACIOUS,  
THE MOST MERCIFUL  

This thesis is dedicated to my parents 
and 
to a very special person with whom I hope to spend the rest of my life.
# Table of Contents

List of Figures vi
Acknowledgements vii
Abstract viii

1 Introduction 1
  1.1 General Introduction 1
  1.2 Work Done 1
  1.3 Thesis Organization 2

2 Related Work 3
  2.1 Current Research 3
  2.2 Database Interface Tools 4
    2.2.1 Microsoft Access 5
    2.2.2 dBase Plus 5
    2.2.3 Other Small-Scale Tools 5
    2.2.4 Multi-user/Large-Scale Tools 6
  2.3 XML Application for UI Representation 6
    2.3.1 XForms 7
    2.3.2 User Interface Markup Language (UIML) 10
  2.4 Application Development Environments (ADE) 11
    2.4.1 Oracle9i Developer Suite 11
    2.4.2 Sun ONE Studio Community Edition 12
    2.4.3 Other ADEs 13

3 Vision for AutoForms 14
  3.1 Motivation 14
  3.2 Components 16
    3.2.1 Application Specification 17
    3.2.2 Visual Tool 17
    3.2.3 Converter 17
    3.2.4 Static Checker 18
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.5 Integration with Development Environments</td>
<td>18</td>
</tr>
<tr>
<td>3.2.6 Extractor</td>
<td>18</td>
</tr>
<tr>
<td>4 Design</td>
<td>19</td>
</tr>
<tr>
<td>4.1 Component Interactions</td>
<td>19</td>
</tr>
<tr>
<td>4.2 XML Representation</td>
<td>19</td>
</tr>
<tr>
<td>4.2.1 XForms</td>
<td>19</td>
</tr>
<tr>
<td>4.2.2 AutoForms XML</td>
<td>20</td>
</tr>
<tr>
<td>4.3 XML-to-Application Converter</td>
<td>22</td>
</tr>
<tr>
<td>4.3.1 Issues</td>
<td>22</td>
</tr>
<tr>
<td>4.3.2 Database Interface</td>
<td>23</td>
</tr>
<tr>
<td>4.3.3 Application Platform</td>
<td>24</td>
</tr>
<tr>
<td>4.3.4 Conversion</td>
<td>25</td>
</tr>
<tr>
<td>5 Results</td>
<td>26</td>
</tr>
<tr>
<td>5.1 Tools Used</td>
<td>26</td>
</tr>
<tr>
<td>5.2 Sample Input</td>
<td>26</td>
</tr>
<tr>
<td>5.3 Sample Output</td>
<td>27</td>
</tr>
<tr>
<td>6 Summary</td>
<td>30</td>
</tr>
<tr>
<td>6.1 Improvements</td>
<td>30</td>
</tr>
<tr>
<td>6.2 Future Work</td>
<td>30</td>
</tr>
<tr>
<td>Appendix A: Sample Forms in XML</td>
<td>32</td>
</tr>
<tr>
<td>References</td>
<td>35</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1 Basic XForms Structure 8
Figure 2.2 A typical XForms implementation 9
Figure 3.1 AutoForms Application Overview 16
Figure 4.1 Database interface of generated application 23
Figure 5.1 Single Table Form 27
Figure 5.2 Multiple Table Form 28
Figure 5.3 Multiple Table with Linked (Nested) Forms 29
Acknowledgements

First and foremost, my sincere thanks go to Dr. Simanta Mitra for working with my ups and (mostly) downs for the last three years and for all his invaluable advice during the research.

I would like to thank all my research group colleagues for all their help and valuable comments during the course of the research and the write up of the thesis: Mamatha Balasubramanian, Ruchi Dhingra (soon to be Ruchi Rao!), and Junaid Babar.

Denise and Scott Nichols deserve a special mention for being my “adopted” parents for the last year.

My thanks also go my parents, Dr. Shaikh Suhrabuddin and Surayya Begum, and my siblings, Shamsuddoha Shaikh and Nishat Parveen, for being there when I needed them.

Last but not least, special mention goes to my fiancée, Rumana Yasmeen. If it were not for her, this thesis would not have been completed.
Abstract

A significant effort is required to create identical database applications for more than one web application platform. A change in application requirements can also force the change of platforms. A feature of current application building tools is to allow the modification of one form at a time. This results in large number of person-hours necessary to change an application with a large number of forms. Working with one form at a time also denies the application developer the ability to check for consequences of the change in the single form on the whole application.

The problems being tackled in this thesis are: the large number of person-hours necessary to write identical applications for multiple web application platforms as a result of demand or due to a change in application requirement; the large number of person-hours necessary for changing an application one form at a time due to application development tool limitations; the inability to find out the consequences of a change in a form has on the whole application.

Various academic and industrial approaches to the above problem, with respect to application development tools and application and user interface representation, were investigated to aid in the specification of a new tool. AutoForms, a tool to enable rapid development of form-based database interface applications, is proposed to resolve these issues. This tool uses a text and XML based application specification to generate applications for multiple platforms. A static checker is used to validate the whole application after a form has been changed.

The results of this thesis can be summarized as follows: a XML representation for an application framework which includes database schema and user interfaces; a prototype for converting the XML representation to a web application platform, namely Java Server Pages (JSP). The results show that the approach being taken is the correct one to aid in the alleviation of consequences of the problems at hand.
1 Introduction

1.1 General Introduction

There are many different web application platforms that can be used to deploy database applications (e.g. JSP, Applets, ASP, PHP, etc.). These web application platforms offer different features and functionalities, but are not interchangeable. If an application is available in one platform, many person-hours are required to convert the application to a different platform. Most of the available tools allow us to develop database applications for one platform. There is a need for a tool which will allow the generation of database applications for multiple platforms.

AutoForms, the proposed tool, enables the rapid development of form-based web database applications and conversion to multiple platforms. It internally represents applications in a Text or XML format. The tool includes a static checker which checks for various design and implementation errors in the application specification and the generated application. It also includes an extractor which will allow an application to be converted to the application specification so that it can manipulated and reconverted to the same or different web application platform.

1.2 Work Done

An XML application specification has been proposed to represent a generic web database application. This application specification consists of proposed extensions to XForms, a recommendation undergoing the standardization process, and additional XML representations developed specifically for AutoForms.

As a proof-of-concept, a converter for transforming the XML application specification has been implemented for the JSP web application platform.
1.3 Thesis Organization

The rest of the thesis is organized in five chapters.

In Chapter 2, current research into the database interfaces for the web is described. Various database interface tools, both small-scale and large-scale, are investigated. Various XML applications for the user interface representation are also discussed.

In Chapter 3, different aspects of the AutoForms tool are discussed, including the motivation behind and advantages of the proposed tool. The different components of AutoForms are also described.

In Chapter 4, a basic design of AutoForms and the generated application is described.

In Chapter 5, some of the results of our implementation are given.

In Chapter 6, a general conclusion, improvements to the current implementation, and the future work are discussed.
2 Related Work

A database application is an application which frequently utilizes a database. A web application is a website which has a one-to-one relationship and establishes a unique session with each and every visitor with contrast to content-based websites. A web application also allows the user to “create, manipulate, and permanently store data” [BAX 03]. Thus we can define a web database application as a web application which frequently utilizes a database.

A web application platform is an execution environment for web applications. There are various web application platforms which can be used to create web database applications. Various such approaches are shown in [FEN 98]. These approaches can be broadly categorized as server-side, client-side and middle-layer. Many of these approaches have been implemented.

As the primary goal is to develop a database interface application for the web, various database interface applications and application development environments with the capability of generating applications with database connectivity were investigated. Various methods of specifying applications in XML, especially form-based applications were also investigated for possible inclusion in the final application specification developed for this research.

2.1 Current Research

The following research was looked at for the purpose of getting an overview of the different types of research activities involving web database interfaces. In all cases, a new interface has been proposed that has some features that were not covered by previous research or available tools. This differs greatly from our research as we are trying to come up with a tool that will target various application platforms rather than coming up with a completely new platform.

In [PAP 96] a tool is described which allows automatic generation of an HTML-based interface to a relational database. The interface is automatically generated from the database structure, thus eliminating the need to manually maintain the HTML code base for the interface. Each table creates links based on the foreign and primary keys defined for the
table. These links can be used for navigation between the tables. During the browsing, a query can be refined by specifying constraints on the queries. The SQL query automatically generated by the system can be displayed. Alternatively a manual SQL Query can also be submitted for processing by the tool.

In [HU 96] a generic web to Database interface building tool, WebinTool is described. WebinTool consists of an interface tool language which allows the developer to write a form files describing the application. The form file specifies the Web Interface displays and database manipulations. An interpreter is used to locate the application and identify the requested form by parsing the requesting parameters, and then generating the HTML files by parsing the form and performing databases operations if necessary.

In [VAR 94] a method for specifying user interfaces to databases by specifying special tags, ZHTML, are described. These tags are parsed to create the necessary user interfaces when the application is run.

2.2 Database Interface Tools

Most database vendors provide tools so that users can create both simple and complex interfaces to databases. Some tools are geared towards producing single user interfaces whereas other databases are geared towards providing large scale multi-user interfaces. The first two tools Microsoft Access and dBase Plus allows the developer to comprehensive set of database interfaces. As far the database interfaces are concerned, they have all the features that we require for our research. The latest version of this software also supports the generation of web applications. But the generated web application does not meet the requirements of our research as the generated application is very platform specific.

Some of the other database interface tools such as Visual FoxPro [MS 02b] and Corel Paradox [COR 02] have good database interface features but do not have very comprehensive support for web based applications.

Some of the larger database vendors also provide tools which allow the developer to build interfaces to their databases. Most of these tools provide the maintainability that we are looking for in our research but most of these tools are also geared towards standalone
application development. All the tools generate applications for platforms which are preferred or sold by the database vendor where as our objective is to target as many platforms as possible.

2.2.1 Microsoft Access

Microsoft Access [MS 02] is designed for small scale databases. Traditionally, Microsoft Access has allowed the developer to create user interfaces (forms and reports) from table and query specifications. It allows the developer to visually create table relationships which then can be used by wizards to generate forms and reports. Microsoft Access uses Visual Basic for Applications (VBA) for data processing requirements for final application. One of the new features available in the latest version, Microsoft Access XP, is the “Data Access Page Designer”. This tool allows the developer to create Web Interfaces to Microsoft Access databases. The generated web application is the Microsoft Internet Information Server (IIS) ISAPI Application.

2.2.2 dBase Plus

dBase [DBA 02] is one of the more popular small-scale database and interface building tools. The early versions provided fourth-generation languages (4GLs) to create User Interfaces, queries and reports. The current version, dBase Plus, allows the developer to create user interfaces, queries, relationships visually with the help of wizards. dBase Plus includes a tool called dQuery II. This allows the developer to create standalone and web-based applications. As with Microsoft Access, the web application is an IIS ISAPI Application.

2.2.3 Other Small-Scale Tools

There are other tools which allow the developer to create web-based applications. Microsoft FoxPro 8 allows the creation of web applications with the help of XML Web Services. Corel Paradox 10 has a “Publish to HTML” feature which allows the user to create web applications.
2.2.4 Multi-user/Large-Scale Tools

The majority of non-open-source database vendors provide tools for creating databases, user interfaces and web applications. The following table lists some of the more popular databases and the tools provided to create applications.

<table>
<thead>
<tr>
<th>Database</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle9i Database</td>
<td>Oracle9i Developer Suite (includes JDeveloper, Forms Developer, Reports Developer, Software Configuration Manager)</td>
</tr>
<tr>
<td>IBM DB2 Universal Database V8.1</td>
<td>Various DB2 Development and Business Intelligence Tools</td>
</tr>
<tr>
<td>IBM Informix (Dynamic Server, Extended Parallel Server)</td>
<td>Various Informix Application and Business Intelligence Tools</td>
</tr>
<tr>
<td>Sybase SQL Server</td>
<td>PowerBuilder 9.0</td>
</tr>
</tbody>
</table>

2.3 XML Application for UI Representation

XML is the current standard for representation data and it has also been used to represent user interfaces. There are many representations available as identified by the Technology Report [OAS 02a]. Two of the representations chosen for this research were XForms 1.0 from W3 Consortium and User Interface Markup Language (UIML) from Harmonia/Virginia Tech HCI.

XForms 1.0 is geared as the next generation of web forms which is going to be used instead of the forms support currently available in HTML 4.01/XHTML 1.0. This was chosen for our research as it provides a comprehensive support for HTTP 1.1 based web application platforms as well as a robust support of forms.

The UIML meets almost all of the goals set for our research. It was designed from ground up to be able convertible to multiple platforms. One of the major drawbacks for UIML is that developer still has to include some target platform code in the form specification. Although
the limitations of this method has been reduced by the designers of UIML by allowing the
target platform code to be unified at one place in the form of libraries. It is more geared
towards general applications rather than database interfaces as is hoped for our research. A
technical committee has recently been formed at OASIS to standardize UIML. UIML does
not fully support the web applications. It relies on other formats such as XForms or XHTML
to provide full support for web applications.

Both XForms and UIML are new technologies and are currently going through the
standardization processes. It will be a benefit for our research to use the standardized
representations as it is our objective to be able to interact with the various tools that are
currently available.

2.3.1 XForms

XForms 1.0 [W3C 02] is an XML application representing web forms. It is currently a
candidate recommendation of the W3 Consortium (W3C). W3C hopes that XForms will the
next generation of web forms. The traditional web forms present in HTML 4.01 and XHTML
1.0 will no longer supported in XHTML 2.0 Recommendation. XHTML 2.0 relies on
XForms 1.0 completely to provide forms support.

XForms specification divides XForms in three different parts: Model, Instance Data, and
User Interface. XForms Model is a device-independent XML form definition which can work
with most proprietary and standard user interfaces. XForms User Interface provides a set of
standard set of user interface controls, similar to the ones available in HTML 4.01/XHTML
1.0. The Instance Data defines the data that is to be manipulated in the form. The diagram
below shows the basic XForms concept. The XForms Submit Protocol is the protocol used
for the form to use the instance data.
Some of the key goals of XForms are:

- Support various browsers.
- Decoupled data, logic and presentation.
- Advanced forms logic.
- Improved internalization.
- Seamless integration with other XML tag sets.

The typical structure XForms XML file is as follows:

```xml
<xforms:model>
  <xforms:instance>
    <!-- Data Definition -->
  </xforms:instance>
  <xforms:bind ref="name1" />
  <xforms:bind ref="name2" />
  <xforms:submit />
</xforms:model>
<!-- XForms User Interface Elements -->
```
A typical XForms implementation is shown in the diagram below:

![Diagram of XForms implementation]

**Figure 2.2 A typical XForms implementation**

As XForms is a new and upcoming technology, implementations are still in the beginning stage and are still evolving. Some of the noteworthy implementations are as follows

**2.3.1.1 IBM XML Forms Package**

“The IBM XML Forms Package is a toolkit consisting of software components designed to showcase the possibilities presented by XForms. The package consists of two main components: the data model component and the client component.

The data model component provides a set of Java APIs for creating, accessing, and modifying XForms data models. This package also includes a JSP tag library that provides a set of tags for use inside JSP code. The tag library interfaces with the XForms data model component APIs, thus providing JSP authors a means of accessing these APIs from within their JSP code. A detailed description of the data model APIs and the tag library, as well as their use, can be found in the documentation for the XML Forms data model.

The client component includes two technologies: An XForms processor control and a Java XForms compiler.” [IBM 02c]
2.3.1.2 **Orbeon Open XML Framework (OXF)**

"OXF is an XML processing framework leveraging the power of J2EE and the flexibility of XML to provide enterprise application developers with a robust next-generation application development platform." [ORB 03]

With the latest version of OXF, development support for XForms is also included.

2.3.2 **User Interface Markup Language (UIML)**

UIML 3.0 [UIM 02] is jointly developed by Harmonia and the Human Computer Interaction (HCI) program at Virginia Tech. A technical committee has been recently formed at the Organization for Advancement of Structured Information Standards (OASIS) to standardize UIML. The goal of this technical committee is "to develop a specification for an abstract meta-language that can provide a canonical XML representation of any user interface." [OAS 02]

2.3.2.1 **LiquidUI™**

Harmonia has developed a set of tools to develop and deploy UIML user interfaces using XML, J2EE and web services technologies.

LiquidUI™ consists of 4 parts:

- **UIML Authoring Tools**: This tool will "allow users to graphically edit UIML interfaces, eliminating the need to manually write UIML code." [UIM 02]

- **UIML Server**: "The UIML server receives request for user interfaces client devices in the form of a URL, constructs a UIML user interface that satisfies the request, invokes the appropriate LiquidUI™ renderer to translate the UIML to the target language used by the browser, and delivers the resultant UI to the client." [UIM 02]

- **UIML Proxies**: "The UIML proxies allow UIML interfaces to call back-end applications that support standard protocols. UIML proxies are available for Common Object Request Broker Architecture (CORBA), Enterprise Java Beans (EJB), Lightweight Directory Access Protocol (LDAP), and Remote Method Invocation (RMI)." [UIM 02]
UIML Renderers: Harmonia provides renderers for Java, HTML, WML, and VoiceXML. Each renderer takes user interfaces in UIML and then creates the necessary user interfaces for the particular platform. These renderers are then used by the UIML server.

2.4 Application Development Environments (ADE)

There are many application development environments available as listed in section 2.2.4. One particular tool of note is the Oracle9i Developer Suite. One primary reason for choosing this tool for analysis for this research was that the full-featured tool was downloadable for members for the Oracle Technology Network (OTN). Membership to the OTN is free.

Oracle9i Developer Suite is also a tool which fits the objectives of our research with respect to the capabilities provided by the tool. One of the main drawbacks is that the web application platform the tool supports are the ones that are only available from Oracle.

One tool of note, not listed in 2.2.4 is the Sun ONE Studio. A feature-limited version is available for download. The basis for the development environment is an open source project called NetBeans. Some of the features available in the free version, Sun ONE Studio 4 Community Edition, are useful and meet some of the objectives of our research.

2.4.1 Oracle9i Developer Suite

Oracle provides a set of comprehensive tools which their database and application server environments. These tools allow the development of database and business intelligence applications. Two of the tools available are JDeveloper and Forms Developer.

JDeveloper is an ADE which has a GUI Form Designer and wizards for designing standalone applications. It allows the development of web applications based on JSP/Servlets. The Forms developer is advertised as an “Internet Rapid Application Development Tool” [ORA 02]. Oracle claims that the Forms Developer is “a highly productive, end-to-end development environment for building enterprise class, database-centric Internet Applications” [ORA 02]. One of the main drawbacks is the resultant application can only run on Oracle Application platform. The Form specification is stored in a proprietary binary format. Oracle has released
a binary-to-XML format (and reverse) tool which converts the binary format to an XML format. The XML file can then be manually edited and then converted back to the binary format for use in the Forms Developer environment. Before conversion the XML file is verified against a XML schema file for correctness. Application correctness is not checked during the conversion back to the binary format.

2.4.2 Sun ONE Studio Community Edition

The Sun ONE Studio ADE [SUN 02] is based on the NetBeans ADE. The NetBeans ADE [NET 02] is a Sun Microsystems sponsored open source project for creating ADEs. The Sun ONE Studio ADE includes several features which are not available in the open source NetBeans version. The Community Edition of Sun ONE Studio is freely available and does not include many features helpful for building enterprise class applications. These features are only available in the complete version of Sun ONE Studio, which is sold by Sun.

Both versions of ADEs allow the building of web applications based on JSP/Servlets. No wizards are available for building web-based forms for the web applications.

The Sun ONE Studio offer several database related features which are not available in NetBeans ADE. These are the Database Schema and JDBC Form Wizard.

The Database Schema Wizard allows the developer to query a database connection to extract all the table information and store it as an XML file. This information can then used for development purposes. One limitation of this feature is that not all database schema queries are supported by the database, thus resulting in the extraction of incomplete information from the database.

The JDBC Form Wizard allows the user to automatically connect to databases, choose a certain number of tables and fields, types of navigation and modification buttons, and generate a JDBC and helper library based, Java standalone application. As noted in the previous paragraph, one of the main disadvantages of using this wizard is not the wizard itself but the fact that the wizard completely relies on the database schema query functions provided by the database and the database drivers.
The Form Designer uses an XML format to store all the form description. The information from the XML file is used to generate the Java file for rendering the form.

2.4.3 Other ADEs

Almost all of the ADEs provide wizards for connecting to and/or generating database interfaces. JBuilder Enterprise (this feature is not available in the freely available JBuilder Personal) includes various wizards for creating database related user interfaces. It also provided a wizard for creating database interfaces with nested queries.
3 Vision for AutoForms

3.1 Motivation

A developer desiring to target products in the various web application platforms would have to rewrite the product for the new platform from scratch. The requirements of an application being developed may also change suddenly, forcing a change of platforms. This can result in loss of many person-hours of work.

Most current tools available allow us to work on one form at a time. This method has many disadvantages. For an application with a large number of forms, the turn-around time for changing an element on each form is very high as the whole testing procedure has to be repeated. This can result in a huge loss of productivity. Working on one form at a time also does not allow the developer to check the impact of changing a form will have on the whole application.

Therefore the objective of the research is to develop an application development tool for building form-based database interfaces for the web. The main objective of this tool would be to address all the issues outlined above. The tool should be easy to use and allow easy maintainability of the application and should target multiple web application platforms. Most current tools available provide wizards that allow the creation of user interfaces (forms) that interact with databases. Most of these tools allow the generation to one particular platform. It is envisioned that AutoForms will provide the ability to generate user interfaces to multiple environments from a single application specification.

The advantages of AutoForms can be summarized as follows:

- The ability to generate of user interfaces in various platforms (e.g. JSP, PHP, ASP).
- The ability to check the application specification for consistency checks and errors.
- The ability to generate applications which conform to generated applications in widely used tools so that these tools can be used to further develop the application.
• The ability to convert back to the application specification from a generated application or an existing application so that the application specification can be modified using the tools provided by AutoForms.

• The ability to use external version control systems (such as CVS). Version control systems are designed to track text based code (such as XML) and not binary. This will allow multiple developers to work on the same project.
3.2 Components

The following diagram shows the different components of AutoForms:

![Diagram of AutoForms components]

**Figure 3.1 AutoForms Application Overview**

As shown in Figure 3.1, AutoForms consists of a Visual Tool, Application Specification (Text and XML Format), the Converters (Applet, JSP, ASP, etc.), the Static Checker and the generated Application.
3.2.1 Application Specification

The application specification contains the database schema, relationships, form relationships, etc. There are two formats: XML and the text-based format. The text-based format is mainly for readability purposes, so that the application developer can edit the text-based specification. The XML format is more for processing purposes mainly due to the huge number of tools that are available for processing and transforming XML documents.

3.2.2 Visual Tool

The visual tool will allow the developer to:

- Attach to a database to query about the current schema that is being used, or create a new database schema so that a database can be created.
- Create relationships between tables by querying for foreign keys from the database connection or visually creating relationships between tables.
- Use a wizard to create interfaces to the chosen database fields.
- Customize the generated user interfaces using a GUI Editor.
- Create reports from databases.

3.2.3 Converter

AutoForms consists of various converters. Each converter takes as input all the XML specification documents specifying an application, such as the database schema and form definitions, and generates the necessary files for running the application in the particular platform.
3.2.4 Static Checker

The static checker is responsible for checking errors in the application specifications. Formal methods may also be used for application correctness.

The static checker will check for, but is not limited to, the following errors:

- Broken links,
- Unreferenced data items, that is, whether a referenced data points to non-existent data in the database.
- Circular links

3.2.5 Integration with Development Environments

It is expected that the generated application will eventually be able to be read in various application building environments, such as JDeveloper from Oracle, Sun ONE Studio/NetBeans from Sun Microsystems. The external tools can be then used for modifying or maintaining the generated application.

3.2.6 Extractor

The extractor will convert the generated application back to the application specification (Text and XML). This will be particularly helpful if the application has been modified using a third party application development environment. Another use of the extractor would be for reverse engineering applications to the AutoForms application specification.
4 Design

Most of the issues facing the design of AutoForms lie in the design of the generated application.

4.1 Component Interactions

Figure 3.1 shows how the components of AutoForms interact with each other. The two formats of the application specification, i.e. the Text and the XML format, are interchangeable. The XML format can be converted to the application platform of choice by using the appropriate converter. The generated application is then compiled and deployed. The extractor will take the generated application or an application and convert it to the application specification. The static checker will verify the application specifications and output diagnostic messages to help the developer take appropriate action.

4.2 XML Representation

The XML application specification is based on the XForms 1.0 candidate recommendation.

4.2.1 XForms

The reasons for choosing XForms as the base for AutoForms XML representation are:

- XForms provides all the necessary tags for representing HTTP-protocol based forms. This is advantageous as the majority of the web platforms (JSP, ASP, PHP, etc.) rely on HTTP-based form processing.
- XForms is easily extensible. It was designed in such a way that other user interface formats can be easily incorporated.
- Within a few years, it is expected, that with wider acceptance of XHTML 2.0, all web forms will be XForms-based. At that time, converting AutoForms XML application specification to new platforms will be very easy, as the current conversion requires conversion to HTML forms first.
- XForms can be easily used to simulate forms on non-HTTP based web platforms.
XForms is an event-based representation of forms. The event-based model can only partially simulated with HTML 4.01 and scripts JavaScript 1.2 on HTTP 1.1 but can be fully supported on client-side platforms such as Java Applets or ActiveX controls. Different platforms will thus support a different set of features offered by XForms.

XForms 1.0 is intended to completely replace HTML Forms in HTML 4.01 and XHTML 1.0. XHTML 2.0 does not have a separate forms section. XHTML 2.0 relies completely on XForms 1.0 to provide forms support.

4.2.2 AutoForms XML

For a complete XML specification for an application, XForms is not enough. XForms relies on presentation formats such as XHTML 1.0 and 2.0 for presentation.

As the AutoForms is an extension to XForms 1.0, the full set of features is supported. The XForms 1.0 recommendation is specified in [W3C 02]. AutoForms XML format provides information such as database schema and user interface controls not provided in the XForms specification.

The current specification is divided into two parts:

- Database Representation.
- Form Representation.

4.2.2.1 Database Representation

The general structure of the database representation is shown below:

```xml
<database
    name="name"
    uri="database uri"
    driver="driver name"
    username="username"
    password="password">
    <table name="table-name">
        <field nullable="yes or no" primary-key="yes or no">
            <name>field-name</name>
            <type>data-type</type>
        </field>
    </table>
    <relationship name="relationship-name" type="one-to-one or one-to-many">
        <start>table.field</start>
        <end>table.field</end>
    </relationship>
</database>
```
4.2.2.2 Form Representation

The form representation follows the XForms 1.0 recommendation and is therefore divided into two parts: the model and the user interface.

The instance part of the XForms model specification is used to correlate the form controls to the database. The general structure is shown below:

```
<data>
  <form-field name="control-name" dbref="database-ref"/>
  <sub-form name="sub-form-name">
    <form-field name="control-name" dbref="database-ref"/>
  </sub-form>
</data>
```

Three new user interface elements have been included in the existing set of XForms user interface elements: table control, navigational controls and modification controls.

A table is a necessary part of any database application as data is often shown in a tabular format. XForms does not support any tabular control. The general structure of AutoForms table control is shown below:

```
<table-control ref="data-reference">
  <label>The label for the table</label>
  <column ref="data-reference">
    <label>Column Label</label>
  </column>
</table-control>
```

Navigational controls are necessary when navigating through the contents of a table. The structure of these controls are shown below:

```
<navigate>
  <start>start</start>
  <previous>previous</previous>
  <next>next</next>
  <end>end</end>
</navigate>
```

Modification controls are necessary when modifying elements in a table. The structure of these controls are shown below:
4.3 XML-to-Application Converter

4.3.1 Issues

For each web application platform, the following has to be considered:

- Where the application logic is running, i.e. whether the application is a server-side application (e.g. JSP), client-side application (e.g. Applet) or a middle-tier application (e.g. CORBA).

- How the application is going to launched and terminated.

For the application platforms based on HTTP 1.1 protocol, such as JSP, ASP, and PHP, further consideration has to be given to the stateless nature of the protocol. An application can only be run using the navigation buttons provided by the application rather than the navigation buttons provided by the Internet browser. The Internet browser navigation buttons can only be used with static web pages. The use of the navigation buttons will result in inconsistent pages being sent to user from the server.

One of the major issues of the generated application is the implementation of nested queries. It is expected that the final product will have the ability to generate all the necessary queries based on just the name of the fields from various tables and the relationship between the tables.

For an application to be interactive, each request has to be sent to the web server, which after processing the request, will then send the output.
4.3.2 Database Interface

Generated applications on different web platforms will have their own unique way to connect to a database. This is so because of the different structure and design methodologies used by the different web platforms.

![Diagram of Database Interface](image)

Figure 4.1 Database interface of generated application

Figure 4.1 shows a diagram depicting how the database interface works in a JSP environment. The JSP engine is responsible for most of the code execution. Each JSP page utilizes helper files which call the database helper library to interact with the database.
4.3.3 Application Platform

For server-side web platforms, each of the XML specification form controls are converted to the corresponding HTML 4.01 form controls. An example of such a conversion is given below.

A text input control in XForms is given by:

```xml
<input ref="name" id="control-id">
<label>Text Input Label</label>
```

The corresponding code in HTML 4.01 would be as follows:

```html
<label for="control-id">Text Input Label</label>
<input type="text" name="ref" id="control-id" value='code-to-retrieve-value-from-database'>
```

The value attribute, in the HTML 4.01 code, contains the platform dependent code which retrieves the value from the database.

For client-side platforms, e.g. Java Applets, each XML specification form control is converted to a series of code. Unlike the server-side platforms, a one-to-one relationship between the XML specification and the generated code does not exist. This is mainly because code to represent a certain user interface element may need to initialized and executed in non-sequential code.

For example in Java Applets, using Swing components, the variables for the user interface elements (JLabel, JTextField, etc) are usually defined and initialized at different portions of the code.
4.3.4 Conversion

As the Form specification is based on the XForms specification, the XML file is divided into three distinct parts: Data Definition, Form/Data Behavior, and the User Interface.

4.3.4.1 Data Definition

This part provides the information required to generate all the helper files to access the data from the database. The helper files will be executed by the server. For JSP-based applications, the helper files are Java Bean elements used to add interactive capability to the JSP pages, in addition to the ability to access data from the database.

4.3.4.2 Form/Data Behavior

This part provides the information required to generate code that will decide on form behavior such as whether a certain data is visible or editable, the type of data to accept, etc. For JSP Pages is achieved using JavaScript to check for data validation and form control behavior.

4.3.4.3 User Interface

This part provides information for the user interface controls which are linked to the data definition parts and Form/Data Behavior part. For JSP pages, this section generates HTML form elements. It uses JSP code and Java Beans generated from section 4.3.4.1 to interact with the database.
5 Results

The objective was to implement a converter for JSP platform as a proof-of-concept.

The components necessary for implementing the converter for JSP are as follows:

- A Java database helper library which uses JDBC to create a connection and interact with the database.
- A converter for the database schema XML to Java class file utilizing the database helper library.
- A converter for the form specification to generate a JSP file and a JavaBean class file for the corresponding JSP file.

All these components were implemented to include the necessary features for an application to be specified and generated.

5.1 Tools Used

The JSP engine used was Apache Tomcat 4.0.3 implementing the specifications for JSP 1.2 and Servlets 2.3.

Apache XalanJ 2.4.1 was used as the XSLT toolkit. Some of the advantages of using this tool is that it offers a comprehensive set of extensions to the standard XSLT library. It also allows the inclusion of Java code to take care of processing which cannot be done by standard XSLT features.

5.2 Sample Input

Three forms are given in Appendix A. The resultant application is described in the following section.
5.3 Sample Output

The following diagrams show various types of forms that are available using the XML-to-JSP Converter. Three forms showing different aspects are described below.

![Single Table Form](image)

Figure 5.1 Single Table Form

Figure 5.1 shows a basic form which has been generated from a single table. The table consists of only two fields: the publisher id and the publisher name. As shown, the form consists of navigation buttons which can be used to navigate in a table. In this case, the Next and the Last buttons are disabled because the last record of the table is being displayed.
Figure 5.2 Multiple Table Form

Figure 5.2 shows a form that has been generated from two tables. The first table, book table, has three fields: the book id, the book name and the publisher id. The second table, publisher table, has two fields: the publisher id and the publisher name. A relationship exists between the two tables, where for every book id, there exists a publisher. All the fields are listed in the form specification, with the exception of the publisher id from the book table. The application will automatically look up the relationship information and display the appropriate data from the database.
Figure 5.3 shows a form that has been generated from four tables. It is an extension of the form shown in Figure 5.2. In addition to the relationships already mentioned for Figure 5.2, for each book id, there are entries in a separate table for the various authors for each book. This table in turn has links to an author table which refers to another table to get the name of the author. The result is then displayed as shown in Figure 5.3. The nested query generation is only partially automatic in the current implementation. The field linking the main query to the nested query has to be manually specified. All table relationships are automatically generated from the list of fields in the form.
6 Summary

A tool, AutoForms, was proposed which will reduce the person-hours spent in generating a new application for a new platform and in maintaining an already existing application. An application specification for database interface application for the web was created by combining XForms, an emerging standard, and a XML representation designed for the tool.

A prototype of a converter to convert an XML based application specification to JSP has been implemented. In a similar manner, converters for other platforms can also be built.

6.1 Improvements

The current implementation can be improved in at least the following two ways:

- Implementing JSP Tags. JSP Tags are used to add functionality to JSP pages. This can be done by linking the tags to predefined libraries. Apache Software Group distributes a set of open source JSP tag libraries. Implementing Tag libraries can make the conversion from XML to JSP much simpler as functionality that is not directly available in the JSP can be simulated through the JSP tag libraries.

- Implementing JSP 2.0 instead of JSP 1.2. JSP 2.0 includes many features that reduce the amount of Java scriptlets used within a JSP Page.

6.2 Future Work

Converters for platforms, such as ASP, PHP, ColdFusion, need to be implemented.

Further research need to done in the following areas:

- Identify and make extensions to the application specification to include data processing capabilities. This concept can also be expanded from database applications to standalone applications for the web.

- Specify a text based representation for the application specification.

- Identify a method for generating single-level and multiple-level nested forms from a list of fields in a query.
• Identify a database framework which will allow for multiple table updates and queries.

To make AutoForms a complete tool, the following components have to be implemented:

• The Visual Tool.
• The Static Checker.
• A converter to interchange between the XML representation and the text based representation.
• The Extractor.
Appendix A: Sample Forms in XML

Single Table Form

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<autoforms:form xmlns:xforms="http://www.w3.org/2002/xforms/cr"
    xmlns:autoforms="http://www.cs.iastate.edu/~shaikhs/autoforms"
    name="pub"
    database='books'>
  <xforms:model>
    <xforms:instance>
      <autoforms:data>
        <autoforms:form-field name="pubid" dbref="publisher_table.id"/>
        <autoforms:form-field name="publisher" dbref="publisher_table.name"/>
      </autoforms:data>
    </xforms:instance>
    <xforms:bind/>
  </xforms:model>
  <xforms:group>
    <xforms:label>Publisher Details</xforms:label>
    <xforms:input ref="pubid">
      <xforms:label>Publisher ID : </xforms:label>
    </xforms:input>
    <xforms:input ref="publisher">
      <xforms:label>Publisher Name : </xforms:label>
    </xforms:input>
  </xforms:group>
  <autoforms:navigate>
    <autoforms:goto-start>First</autoforms:goto-start>
    <autoforms:goto-previous>Previous</autoforms:goto-previous>
    <autoforms:goto-next>Next</autoforms:goto-next>
    <autoforms:goto-end>Last</autoforms:goto-end>
  </autoforms:navigate>
</autoforms:form>
```
Multiple Table Form

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
  <xforms:model>
    <xforms:instance>
      <autoforms:data>
        <autoforms:form-field name="bookid" dbref="book_table.id"/>
        <autoforms:form-field name="bookname" dbref="book_table.name"/>
        <autoforms:form-field name="pubid" dbref="book_table.publisher"/>
        <autoforms:form-field name="pubname" dbref="publisher_table.name"/>
      </autoforms:data>
    </xforms:instance>
    <xforms:submission action="book_basic.jsp" method="POST"/>
  </xforms:model>

  <xforms:group>
    <xforms:label>Book Details</xforms:label>
    <xforms:input ref="bookid">
      <xforms:label>Book ID : </xforms:label>
    </xforms:input>
    <xforms:input ref="bookname">
      <xforms:label>Book Name : </xforms:label>
    </xforms:input>
  </xforms:group>

  <xforms:group>
    <xforms:label>Publisher Details</xforms:label>
    <xforms:input ref="pubid">
      <xforms:label>Publisher ID : </xforms:label>
    </xforms:input>
    <xforms:input ref="pubname">
      <xforms:label>Publisher Name : </xforms:label>
    </xforms:input>
  </xforms:group>

  <autoforms:navigate>
    <autoforms:goto-start>First</autoforms:goto-start>
    <autoforms:goto-previous>Previous</autoforms:goto-previous>
    <autoforms:goto-next>Next</autoforms:goto-next>
    <autoforms:goto-end>Last</autoforms:goto-end>
  </autoforms:navigate>
</autoforms:form>
```
Multiple Tables with Linked (Nested) Forms

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<autoforms:form xmlns:xforms="http://www.w3.org/2002/xforms/cr"
xmlns:autoforms="http://www.cs.iastate.edu/~shaikhs/autoforms"
name="book_author"
database="books">
  <xforms:model>
    <xforms:instance>
      <autoforms:data>
        <autoforms:form-field name="bookid" dbref="book_table.id"/>
        <autoforms:form-field name="bookname" dbref="book_table.name"/>
        <autoforms:form-field name="publisher" dbref="publisher_table.name"/>
        <autoforms:sub-form name="authors" parameter="book_table.id"
constraint="book_author_table.book_id">
          <autoforms:form-field name="authorid" dbref="book_author_table.author_id"/>
          <autoforms:form-field name="authorname" dbref="author_table.name"/>
        </autoforms:sub-form>
      </autoforms:data>
    </xforms:instance>
    <xforms:bind>
    </xforms:bind>
    <xforms:submission action="book_author.jsp" method="POST"/>
  </xforms:model>
  <xforms:group>
    <xforms:label>Main Book Details</xforms:label>
    <xforms:input ref="bookid">
      <xforms:label>Book ID : </xforms:label>
    </xforms:input>
    <xforms:input ref="bookname">
      <xforms:label>Book Name : </xforms:label>
    </xforms:input>
    <xforms:input ref="publisher">
      <xforms:label>Publisher Name : </xforms:label>
    </xforms:input>
  </xforms:group>
  <autoforms:table-control ref="authors">
    <xforms:label>Authors</xforms:label>
    <autoforms:column ref="authorid">
      <xforms:label>Author ID</xforms:label>
    </autoforms:column>
    <autoforms:column ref="authorname">
      <xforms:label>Author Name</xforms:label>
    </autoforms:column>
  </autoforms:table-control>
  <autoforms:navigate>
    <autoforms:goto-start>First</autoforms:goto-start>
    <autoforms:goto-previous>Previous</autoforms:goto-previous>
    <autoforms:goto-next>Next</autoforms:goto-next>
    <autoforms:goto-end>End</autoforms:goto-end>
  </autoforms:navigate>
</autoforms:form>
```
References


[SUN 02] Sun Microsystems, Sun ONE Studio 4,

(current May 2003)

[UIM 02] UIML.org, User Interface Markup Language (UIML), http://www.uiml.org/
(current May 2003)

Database Applications”, Proceedings of the First International WWW
Conference, Geneva, Switzerland, May 1994,

[W3C 01] W3 Consortium, XForms, http://www.w3.org/MarkUp/Forms/, (current May
2003)

[W3C 02] W3 Consortium, XForms 1.0 Candidate Recommendation,
2003)

[WDB 96] ESO, A Web to Database Interface, 1996,