An extended rule framework for web forms: adding to metadata with custom rules to control appearance

Atia M. Albhbah and Mick J. Ridley

Abstract— **This paper proposes the use of rules that involve code to implement more semantics for web forms. Separation between content, logic and presentation of web applications has become an important issue for faster building and easy maintaining. Just as CSS applied on the client side to control the overall presentation of web applications, a set of rules can give a similar consistency to the appearance and operation to any set of forms that interact with the same database. We develop of rules to order web form elements and query forms using Reaction RuleML format in conjunction with database metadata rules. Database metadata can be extracted from system catalogue tables in typical relational database systems by several tools depending on Web application environment. The results show that this mechanism successfully insulates application logic from code.**

Index Terms— web forms, database metadata, Reaction RuleML, XML.

I. Introduction

Many web based applications, in commercial and scientific areas use forms to enter data for storing or querying database systems. These database systems contain information about data stored as a set of system catalogues which are known as metadata [1]. In addition, metadata consists of information such as; list of database tables, column names, and all integrity constraint rules, which will be used to control data that is saved and manipulated in the database tables. If applications are built by hand all this information in a database should be embedded into the application. This can be time consuming and may require ongoing maintenance.

Automatic and dynamic generation of Web applications is moving into the mainstream in the web development field nowadays, because many business agencies are looking to change their database applications into on-line systems, and the growth of technologies like XML [2], XHTML [3] and many others, have pushed them to update their Web applications using existing databases to take advantages of these technologies. Separation between content, logic and presentation of Web applications has become an important issue for faster building and easy maintaining, e.g. CSS applied on the client side to let Web developer control the overall presentation of their Web applications. In addition CSS uses rules to control the appearance of the document [4].

In this paper we develop the initial use of RuleML in conjunction with database metadata originally proposed in [5] to a more general framework that allows "common sense" and "domain specific" rules to be included in the system. The aim is to extend the automation of web forms so that more semantic information is used in a consistent fashion. We investigate the use of Reaction RuleML on the server side to give a consistent use of variables and therefore a consistent look and feel to forms across pages within applications using a database. We know that web site maintenance is a problem and just as use of CSS on the client side can give consistency to the appearance of pages generally; use of a set of rules can give a similar consistency to the appearance and operation to any set of forms that interact with the same database.

We illustrate our approach with the development of a student based example later but introduce it with a simple UK car registration example. If a database column reg_no was defined as char (8) and not null we would use the database metadata to produce a form of suitable size, which would be required for data entry. However a textbox can have additional data put into it, with the addition of domain specific knowledge we could supplement the behaviour of the form to enforce a maximum of 7 characters and that only numbers, spaces and uppercase letters (excluding I and O) were permitted. This behaviour should be applied to any form using the reg_no field.

Therefore we propose a framework as structured in Fig.1 [6] where we supplement database metadata rules with common sense rules and introduce a second rulebase of domain specific rules.

The common sense rules add functionality not limited to a specific domain but also not supported by database metadata which is often limited by factors such as the type system of the database itself [7]. In this category are rules like those mapping a column called password to a password type form input.

In this paper we investigate the use of Reaction RuleML on the server side to give consistent use of variables and therefore a consistent look to forms across pages within applications using database. Therefore we use Reaction RuleML format to store database metadata rules and save it as rulebase, In addition, we propose a framework as in Fig.1 that divides the rules into two types. The first one is to save the common sense rules and the second to save domain specific rules which will help to develop a prototype system that can generate automatic and dynamic web entry forms for Web applications, and development of rules that involve code to implement more semantics, development of rules to order and or group form elements.

Rules on the Web have become an increasingly important issue during the last couple of years in both industry and academia areas. It has been concluded that when rules are

Manuscript received September 21, 2011; revised November 28, 2011.
Atia M. Albhbah and Mick J. Ridley are with the Department of Computing, School of Computing informatics & media, University of Bradford, Richmond Road, Bradford, BD7 1DP, UK. (e-mail: A.Albabah@student.bradford.ac.uk; M.J.Ridley@Bradford.ac.uk)

embedded in application code it becomes difficult to locate and change the logic [7], and each modification requires recompiling the application code. Hence, separating rules from application code allows easily manipulation of the rules. RuleML (Rule Markup Language) is an XML-based markup language which allows rules to be expressed as modular components using standard XML tags [8]. RuleML format can be used to represent metadata retrieved from database [9]. In this case RuleML could be used to save rules retrieved from database metadata as a rulebase which in turn separates the rules from the application code to improve accessibility, provide more flexibility, and control.

This paper is organised as following: in section II we review related work on interfaces and provide an introduction to RuleML and Reaction RuleML. Section III illustrates the use of Reaction RuleML0.2 implemented on a prototype using PostgreSQL accessed via PHP. Conclusions and suggestions for future development are presented in section IV.

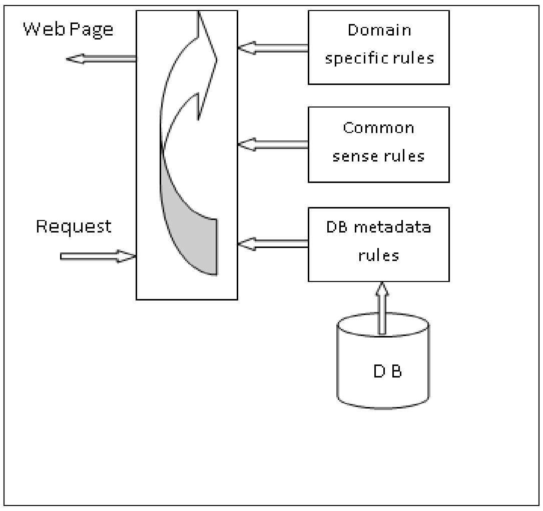


Fig. 1. Structure of the proposed framework.[6]

II. Previous Related Work

A. Metadata driven interfaces

Weiner et al. [10] describe an approach of dynamically generate web based database interfaces. This was using a manually developed metadata table, which contains information about the elements required to represent a data model (table names, field names, data type, and links between tables). In the described model since the metadata is built by hand it is possible that the web interface will not be an accurate representation of the database and it needs more effort.

Elsheh and Ridley [7] proposed a model which aims to generate dynamic Web entry forms based on metadata extracted from system tables. They used the java servlet class to convert the extracted metadata via JDBC into an XML document. A set of rules has been developed and applied to database metadata which is used to map each column to specific user interface controls. In addition, the XML document is transformed into an XHTML document using XSLT stylesheet, which is returned back to the user as Web entry form. The set of rules of this scheme is embedded in the application code where it is difficult to locate and change their logic.

Mgheder and Ridley [11] suggested an approach that uses metadata stored in system tables in databases (columns name, type, size etc.) to develop generic user interface elements. They used PHP as the server script and the database abstraction library ADOdb to achieve their goal. The metadata is extracted from the database by using the ADOdb metadata methods. This metadata information combined with a developed set of rules is used to automatically map each column in the database table to a specific user interface control. The proposed model uses a set of rules which are extracted from the database to build the Web form; these rules are built within the application code, where it is not easy to maintain them.

Bertossi and Jayaraman [9] describe a methodology that uses metadata for a virtual and relational data integration system under the local-as-view (LAV) approach. They used a standard format based on XML and RuleML for representing metadata. This design allows data sources to be added to the system or removed without affecting any other data sources.

Albhbah and Ridley [5] proposed a new way where a rulebase is used in conjunction with database metadata to develop automatic Web forms. They described how to develop an abstract way to achieve the goal. In their approach, the separation of the rules from the application code using Reaction RuleML format to represent metadata retrieved from database, which saved as a rule base has improved accessibility, provided more flexibility, and control.

Albhbah and Ridley [6] proposed a frame work which aims to divide the rules into two types. The first one is to save the common sense rules and the second to save domain specific rules which will help to develop a prototype system that can generate automatic and dynamic web entry forms for Web applications.

B. RuleML

Rules on the Web are becoming a more mainstream topic these days and are expected to play an important job in the success of the Semantic Web [8]. Rule Markup Languages (RuleML) will be the vehicle for using rules on the Web [12]. RuleML has been defined by the Rule Markup Initiative to express a family of Web rules to support both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, transformation and reaction [8]. It is used to create a basis for a universal rule Markup Language using standard XML tags, which helps to specify rules, and allows exchange, manipulation and analysis of rules. RuleML is a family of sublanguages which was launched in August 2000 and it is now at version 1.0, which was released in 2010 [8]. The initiative is very flexible as it uses XML and it is not limited only to proposing a language but also it can be translated to some targeted rules engines (e.g. RuleML to JESS). Before executing RuleML rules, the rules need to be translated to an inference engine language, such as Java Expert System Shell (JESS) or Prolog to be executed. But in our approach we focus on how to use RuleML format to save rules as readable base, details are provided in section III.

In Fig.2, RuleML specifies different types of rules [13] which are described as follows:

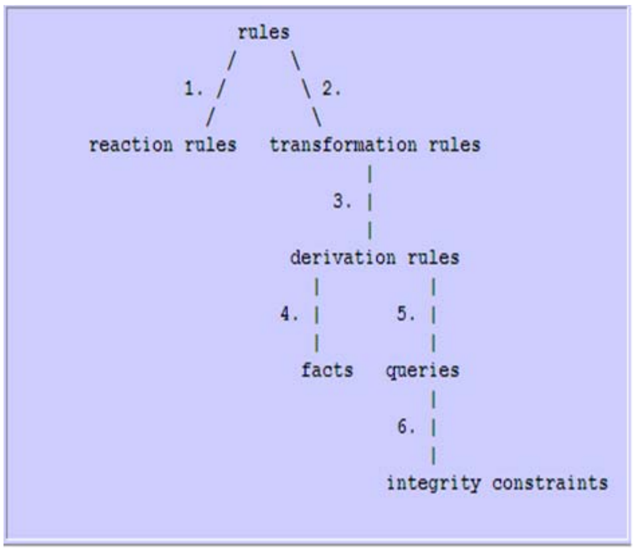


Fig. 2. A graphical view of RuleML rules [13].

- Reaction Rules (event-condition-action rules) can only be applied in the forward direction in natural, observing/checking events/conditions and performing an action if and when all events/conditions have been recognized/fulfilled as in the example "*When a share price drops by more than 5% and the investment is exempt from tax on profit, then sell it*" [12]. The reaction rule specifies the reactive behavior of a system in response to events.
- Transformation Rules (functional-educational rules).
- Derivation Rules (implication-inference rules) can be applied in both forward and backward directions as in the example "*A gold customer is a customer with more than $1Million on deposit*" [12].
- Facts as in the examples "*Tom sells TV to Landa*", "*A module is option*".
- Queries "*provide 10% fees discount for students attending all the lectures*".
- Integrity Constraints (consistency-maintenance rules) as in the example "*A driver of a rental car must be at least 25 years old*" [12].

Example of the general form of RuleML 1.0 syntax is given below [14]:

```
<!-- Implication Rule 1: Backward notation of 'then' and 'if' roles, as in Logic Programming, and forward notation using natural 'if' ... 'then' order, as in textbook logic, with exact same meaning
"The discount for a customer buying a product is 5.0 percent if the customer is premium and the product is regular."
Notice that the ternary discount relation is applied via an Atom. Furthermore, a Data constant can syntactically be an entire phrase like "5.0 percent". It will unify only with variables and with Data having exactly the same spelling (incl. spaces)
        -->
<Implies>
        <if>
        <And>
        <Atom>
        <Rel>premium</Rel>
        <Var>cust</Var>
        </Atom>
        <Atom>
        <Rel>regular</Rel>
        <Var>prod</Var>
        </Atom>
        </And>
        </if>
        <then>
        <Atom>
        <Rel>discount</Rel>
        <Var>cust</Var>
        <Var>prod</Var>
        <Data>5.0 percent</Data>
        </Atom>
        </then>
</Implies>
```

C. Reaction RuleML

Reaction RuleML is a branch of the RuleML family; it is described as a general language and rule interchange for the family of reaction rules [15]. Reaction RuleML introduced different types of production, action and reaction rules into the native RuleML syntax. The design aim of Reaction RuleML is to be easy to learn and facilitate fast maintenance with less risk [16].

The general syntax for Reaction RuleML has been updated to be easy to use, where more tags have been added to the new version (0.2) [17]. The general form of the Reaction RuleML0.2 syntax is given below:

```
<Rule style="active" evaluation="strong">
<label><!-- metadata --></label>
<scope><!-- scope --></scope>
<qualification><!-- qualifications --></qualification>
<oid><!-- object identifier --></oid>
<on><!-- event --></on>
<if><!-- condition --></if>
<then><!-- conclusion --></then>
<do><!-- action --></do>
<after><!-- postcondition --></after>
<else><!-- else conclusion --></else>
<elseDo><!-- else/alternative action --></elseDo>
<elseAfter><!-- else postcondition --></elseAfter>
</Rule>
```

The building blocks of the Reaction RuleML 0.2 general form is contain one general rule form "<Rule>", three execution styles "active/messaging/reasoning" and two evaluation "weak/strong". This kind of rules suits the type of rule we are concerned with (see section III) and will be used in the prototype development.

III. Application Development

This section introduces a prototype development system which aims to design a framework using the Reaction RuleML 0.2 format to save and implement rule bases to overcome the limitations that database metadata information is only used when data enters the database and use the rules with the metadata to generate automatic and dynamic Web forms, and to support a composite attribute which consists of a group of values from more than one domain [6]. Query forms can be designed and generated using set of rules.

The general idea of the prototype implementation was the creation of a Web form to evaluate to what extent we can use the relational database metadata and build the rule base using a Reaction RuleML 0.2 format to save the rules as two types the first one is common sense Rulebase and the second one is domain specific Rulebase, which will be used to build adaptable dynamic database interfaces. The metadata will be extracted using a number of PHP's PostgreSQL functions. The following objectives are intended to be achieved.

- Extract metadata.
- Apply domain specific Rulebase.
- Apply common sense Rulebase.
- Apply query Rulebase.
- Generate Web form element.

A. Building RuleML metadata rule bases

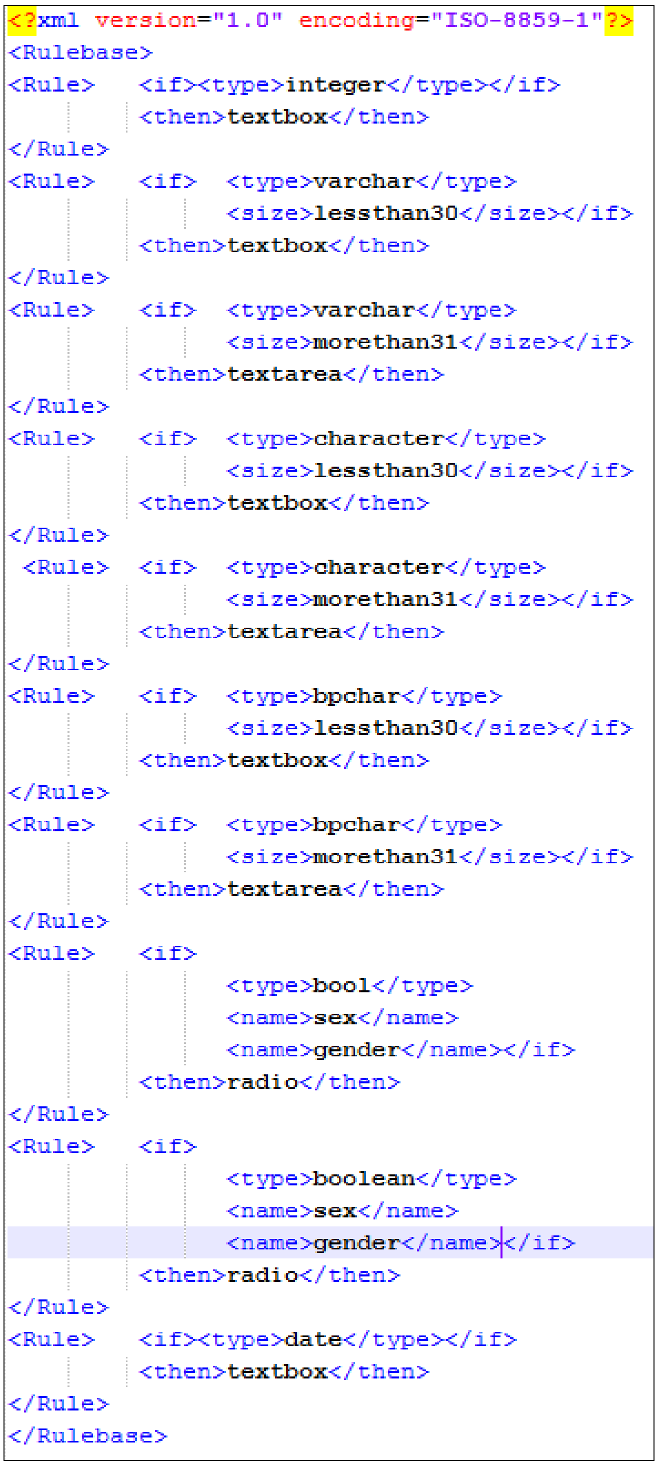
In this section we introduce two types of rules that can be applied to the information that exists in database metadata; the first rule base is to save all common sense rules e.g.:

- Rule 1: if a column is integer type, then it should be mapped to textbox Web form control.
- Rule 2: if a column is character type and its length is less than or equal to 30 (for example), then it should be mapped to textbox Web form control.
- Rule 3: if a column is character type and its length is more than 30, then it should be mapped to textarea Web entry form.
- The condition on the length of the field in Rule 2 and Rule 3 could be more or less and could be changed.
- Rule 4: if a column is Boolean type, then it can be implemented as a group of radio buttons or drop down menu. But in this framework radio buttons will be used. In addition in this rule we can use the name of the column to make the Web form more clear.
- Rule 5: if a column is date type, then it could be mapped to a textbox and the format of the date is presented as label.
- Rule 6: if a column name is password or sub set of these words, then this column should be mapped into a password textbox.

Rules 1 – 5 are generic rules, based on data type information held in database metadata.

Rule 6 is common sense rule based on column name information held in database metadata.

Some of the above developed rules using RuleML format are illustrated in Fig. 3 as:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rulebase>
<Rule>    <if><type>integer</type></if>
          <then>textbox</then>
</Rule>
<Rule>    <if>   <type>varchar</type>
                 <size>lessthan30</size></if>
          <then>textbox</then>
</Rule>
<Rule>    <if>   <type>varchar</type>
                 <size>morethan31</size></if>
          <then>textarea</then>
</Rule>
<Rule>    <if>   <type>character</type>
                 <size>lessthan30</size></if>
          <then>textbox</then>
</Rule>
 <Rule>   <if>   <type>character</type>
                 <size>morethan31</size></if>
          <then>textarea</then>
</Rule>
<Rule>    <if>   <type>bpchar</type>
                 <size>lessthan30</size></if>
          <then>textbox</then>
</Rule>
<Rule>    <if>   <type>bpchar</type>
                 <size>morethan31</size></if>
          <then>textarea</then>
</Rule>
<Rule>    <if>
                 <type>bool</type>
                 <name>sex</name>
                 <name>gender</name></if>
          <then>radio</then>
</Rule>
<Rule>    <if>
                 <type>boolean</type>
                 <name>sex</name>
                 <name>gender</name></if>
          <then>radio</then>
</Rule>
<Rule>    <if><type>date</type></if>
          <then>textbox</then>
</Rule>
</Rulebase>
```

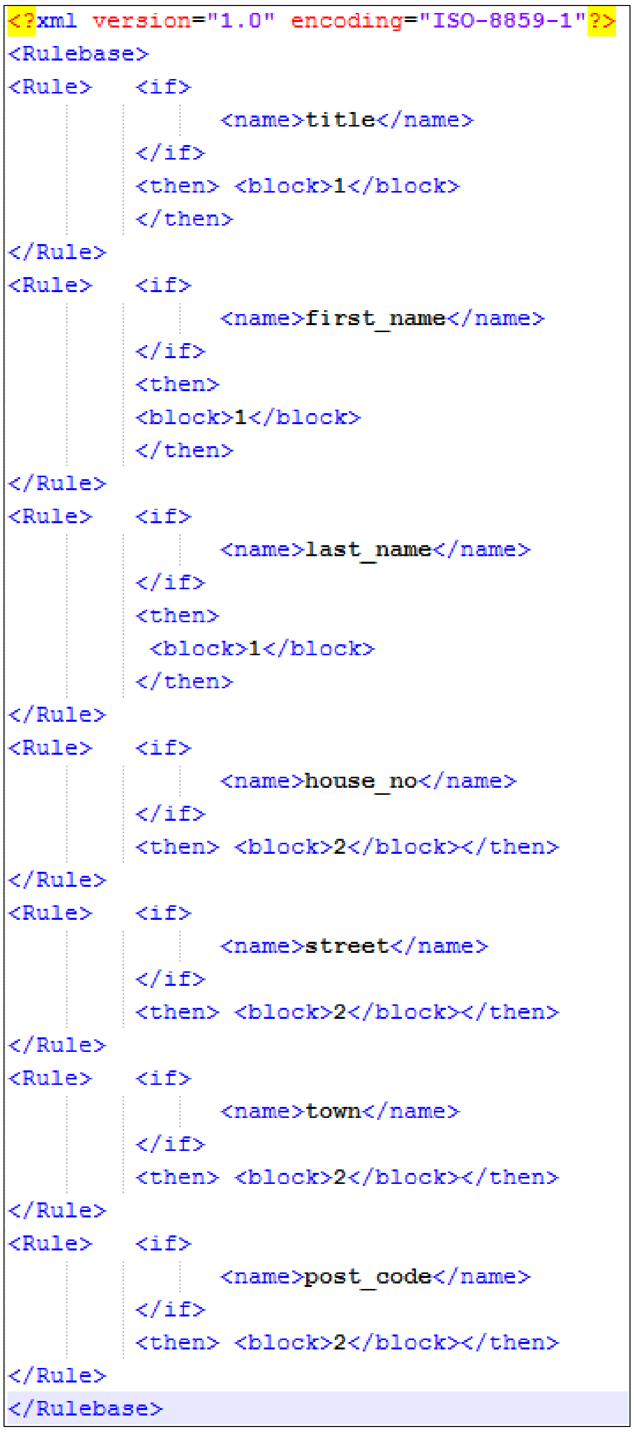
Fig. 3. Metadata rule base in Reaction RuleML 0.2 format as common sense rules.

The second rule base is to save all domain specific rules, which can be divided to two sets of rules, using our example, which is student information, we develop two sets of domain specific rules the first one can be applied to generate input forms and the second one will be applied to generate the query forms as:

The first domain specific rule set is:

- Rule 1: if a column name is title then it should be grouped to block1.
- Rule 2: if a column name is first_name then it should be grouped to block1.
- Rule 3: if a column name is last_name then it should be grouped to block1.
- Rule 4: if a column name is house_no then it should be grouped to block 2.
- Rule 5: if column name street then it should be grouped to block 2.
- Rule 6: if column name town then it should be grouped to block 2.
- Rule 7: if column name post_code then it should be grouped to block 2.

Some of the above developed rules using RuleML format are illustrated in Fig. 4 as:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rulebase>
<Rule>    <if>
                <name>title</name>
          </if>
          <then> <block>1</block>
          </then>
</Rule>
<Rule>    <if>
                <name>first_name</name>
          </if>
          <then>
          <block>1</block>
          </then>
</Rule>
<Rule>    <if>
                <name>last_name</name>
          </if>
          <then>
           <block>1</block>
          </then>
</Rule>
<Rule>    <if>
                <name>house_no</name>
          </if>
          <then> <block>2</block></then>
</Rule>
<Rule>    <if>
                <name>street</name>
          </if>
          <then> <block>2</block></then>
</Rule>
<Rule>    <if>
                <name>town</name>
          </if>
          <then> <block>2</block></then>
</Rule>
<Rule>    <if>
                <name>post_code</name>
          </if>
          <then> <block>2</block></then>
</Rule>
</Rulebase>
```

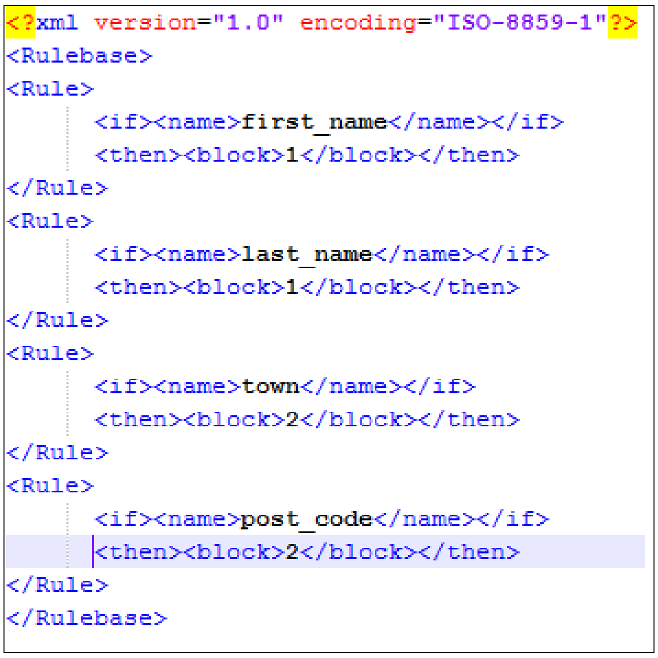
Fig. 4. Metadata rule base in Reaction RuleML 0.2 format as domain specific input form rules

The second domain specific rule set will be applied to generate the query form as we set the rules as:

- Rule 2: if a column name is first_name then it should be grouped to block1.
- Rule 3: if a column name is last_name then it should be grouped to block1.
- Rule 2: if a column name is town then it should be grouped to block2.
  Rule 3: if a column name is post_code then it should be grouped to block2.

In this section we use our domain specific rules to overcome the lack of semantic content available automatically from a database. We may know that frist_name and last_name are related items and should be grouped but this information is not available automatically. In most RDBMSs the only connection is in the similarity of the column names. In the example although the "name" elements are named similarly the "address" elements are not. In some DBMSs the columns could be implemented as a composite type but this is not commonly done.

Some of the above developed rules using RuleML format are illustrated in Fig. 5 as:



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rulebase>
<Rule>
      <if><name>first_name</name></if>
      <then><block>1</block></then>
</Rule>
<Rule>
      <if><name>last_name</name></if>
      <then><block>1</block></then>
</Rule>
<Rule>
      <if><name>town</name></if>
      <then><block>2</block></then>
</Rule>
<Rule>
      <if><name>post_code</name></if>
      <then><block>2</block></then>
</Rule>
</Rulebase>
```

Fig. 5. Metadata rule base in Reaction RuleML 0.2 format as domain specific query form rules .

B. *Retrieving metadata from the database*

We start implementing our approach by creating a database table which contains student information, to illustrate a range of data types and other conditions (nullability) as:

```
    CREATE TABLE student (
  student_id serial NOT NULL,
  title character(6) NOT NULL,
  first_name character(20) NOT NULL,
  last_name character(10) NOT NULL,
  house_no character(5) NOT NULL,
  street character(20) NOT NULL,
  town character(20) NOT NULL,
  post_code character(10) NOT NULL,
  CONSTRAINT student_id PRIMARY KEY
(student_id))
```

A number of PHP's PostgreSQL functions are used to make a connection to the database and retrieve the metadata [5].

C. *Generate the Web forms*

A general purpose PHP script was written which loops through all the metadata for each column and uses the Reaction RuleML 0.2 rulebases. It tests to see which rules apply and then uses those rules to build the form elements on the fly as:

- In Fig. 6 using the common sense rules where for example the student_id is mapped to a textbox of the appropriate size (found from the metadata) and marked as required since it is specified as non null, its label is formatted as described below. Every column in the database table is mapped to a specific Web form control element. The label of each control element is the actual column's name in the database, retrieved from the database table metadata. PHP functions can used to produce a user friendly label. For instance, functions are used to replace underscores which separate words in a column's name by spaces and change the first character of all words to upper case. As a guide to the user and to make the form simpler we have used (*) for the required fields (columns that are primary key or specified as not null). This can be supplemented with JavaScript to ensure a value is provided.

•

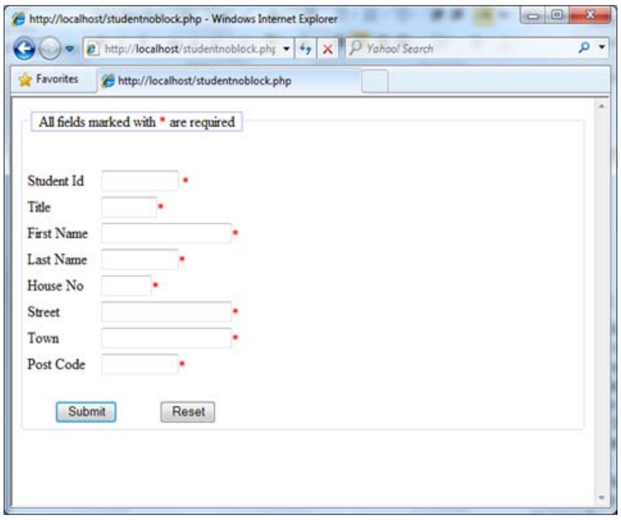

http://localhost/studentnoblock.php - Windows Internet Explorer
http://localhost/studentnoblock.php
Yahoo! Search
Favorites
http://localhost/studentnoblock.php
All fields marked with * are required
Student Id *
Title *
First Name *
Last Name *
House No *
Street *
Town *
Post Code *
Submit
Reset

Figure 6. User interface input form generated automatically using metadata and Reaction RuleML 0.2.

- In Fig. 7 using the first domain specific rules, which are applied to generate web entry form to group the composite attributes which consist of a group of values from more than one domain. For example the attributes (title, first_name, last_name,) were mapped as a block to give more semantic to the form, and the attributes (house_no, street, town, post_code) were mapped as a block to group the elements together as a composite attribute. The rules represent all the semantic information which is not in the database metadata to order the web form elements.

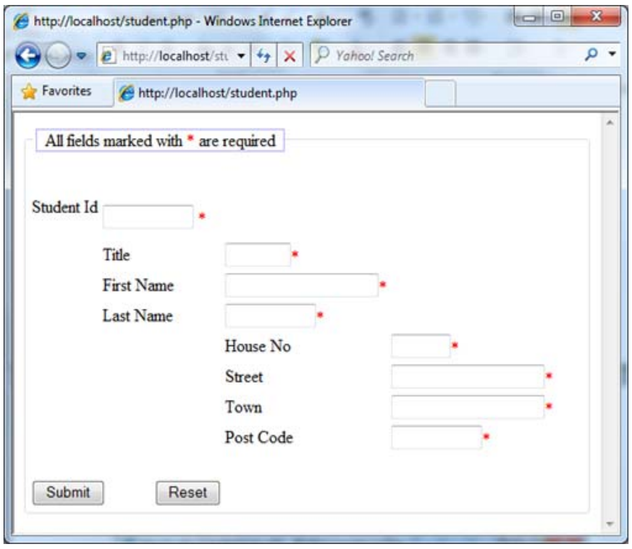


Fig. 7. User interface input form generated automatically using metadata and Reaction RuleML 0.2 grouped the attributes as a composite attribute.

- In Fig. 8 using the second domain specific rules, which are applied to generate web query form to order the required attributes. For example the attributes (first_name, last_name, town, post_code) have been ordered to generate the query form as in Fig. 8. The rules could be used to order and generate more forms which can contain different attributes to help the user to get the needed information.

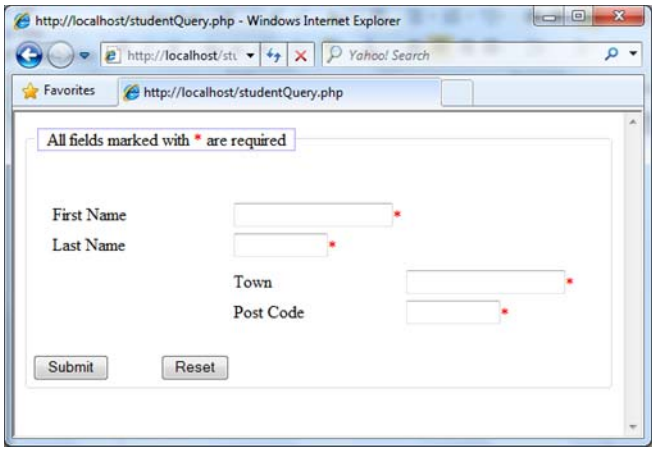

http://localhost/studentQuery.php - Windows Internet Explorer
http://localhost/stu
Yahoo! Search
Favorites
http://localhost/studentQuery.php
All fields marked with * are required
First Name *
Last Name *
Town *
Post Code *
Submit
Reset

Figure 8. User interface query form generated automatically using metadata and Reaction RuleML 0.2.

IV. Conclusion and Further Work

Automatic and dynamic generation of Web forms is entering the mainstream in web development for supporting developing online systems. Rules extracted from database metadata and used to generate the Web forms when embedded in the application code are not efficient due to the difficulty of locating and changing the logic. In this paper we proposed an approach which separates the rules as independent entity from the application code, by using Reaction RuleML0.2 format as rulebase. The system evaluation is carried out using Reaction RuleML0.2 format to save the developed rules, PostgreSQL as a DBMS, PHP for server side programming, HTML and JavaScript for client side programming. As a result a Web form for user interface is generated dynamically. This approach aims to produce common sense rules and introduce a second rulebase of domain specific rules using Reaction RuleML0.2, and development of rules that involve code to implement more semantics, development of rules to order and or group form elements.

No performance issues have been noted yet but we will investigate further when we have larger, more realistic sets of rules working with real databases in place. Additionally the size of database's metadata is usually small compared to the actual data, and only increases with the number of tables rather than the volume of data per table this has lead to small RuleML files which are not complex to parse.

There are a number of areas for further development which we hope to pursue. The current example only shows labeling for required fields this can be developed so that the rule involves applying a suitable piece of JavaScript code to enforce a non null value. In our current implementation we have used database metadata which in this case is derived from relational database system catalogs but could be obtained from other sources such as XML Schema and used in conjunction with XML data.

REFERENCES

[1] "System Catalogs," Available online at: http://www.postgresql.org/docs/8.1/static/catalogs.html, accessed on 20 Sep 2011.

[2] "Introduction to XML," available online at: http://www.w3schools.com/xml/xml_whatis.asp., accessed on 20 Sep 2011.

[3] "W3C XHTML Activities," available online at: http://www.w3schools.com/w3c/w3c_xhtml.asp, accessed on 20 Sep 2011.

[4] "Cascading Style Sheets home page" available online at: http://www.w3.org/Style/CSS/, accessed on 20 Sep 2011.

[5] A. M. Albhbah and M. J. Ridley, "Using RuleML and database metadata for automatic generation of web forms," in Proceedings of the 10th International Conference on Intelligent Systems Design and Applications (ISDA), 2010, pp. 790-794.

[6] A. M.Albhbah and M. J.Ridley, "A Rule Framework for Automatic Generation of Web Forms'," in Proceedings of the 4th IEEE International Conference on Computer Science and Information Technology (IEEE ICCSIT 2011), Chengdu, China, 2011.

[7] M. M. Elsheh and M. J. Ridley, "Using Database Metadata and its semantics to Generate Automatic and Dynamic Web Entry Forms," in Proceedings of the World Congress on Engineering and Computer Science (WCECS 07), 2007, pp. 654-658.

[8] "The Rule Markup Initiative," available online at: http://ruleml.org/, accessed on 20 Sep. 2011.

[9] L. Bertossi and G. Jayaraman, "Designing, Specifying and Querying Metadata for Virtual Data Integration Systems," Globe 2009, LNCS 5697, Springer, 2009, pp. 72-84.

[10] M. S. Mark Weiner, Abigail Cohen, "Metadata tables to enable dynamic data modeling and web interface design: the SEER example," International Journal of Medical Informatics, vol. 65(1), April 2002, pp. 51-58.

[11] M. A. Mgheder and M. J. Ridley, "Automatic Generation of Web User Interfaces in PHP Using Database Metadata," in Proceedings of the 3rd International Conference on Internet and Web Applications and Services, ICIW '08, IEEE Computer Society, 2008, pp. 426-430.

[12] W. Gerd, A. Grigoris, T. Said, and B. Harold, "The Abstract Syntax of RuleML-Towards a General Web Rule Language Framework," in Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence: IEEE Computer Society, 2004, pp. 628-631.

[13] "RuleML Design," available online at: http://ruleml.org/indesign.html, accessed on 20 Sep 2011.

[14] "Schema Specification of RuleML 1.0," available online at: http://ruleml.org/1.0/, accessed on 20 Sep 2011.

[15] "Reaction RuleML," available online at: http://ruleml.org/reaction/, accessed on 20 Sep 2011.

[16] "Reaction RuleML 0.1 Tutorial at RuleML’06" available online at: http://ruleml.org/reaction/index.htm#changes, accessed on 20 Sep 2011.

[17] "Primer Reaction RuleML 0.2," available online at: http://ruleml.org/reaction/0.2/index.htm, accessed on 20 Sep 2011.