

## A Framework For Rapid Development Of OLTP Information Systems: Transformation Of SQL Statements To Three-Tier Web Applications

Teduh Dirgahayu  
 Department of Informatics  
 Indonesian Islamic University  
 Yogyakarta, Indonesia  
 Email : [teduh.dirgahayu@uii.ac.id](mailto:teduh.dirgahayu@uii.ac.id)

**Abstract**—In this paper, we present a framework for rapid development of online transaction processing (OLTP) information systems. The framework assumes that a Web application will be developed on top of an available database. It includes a transformation that receives as an input a SQL statement and results in as an output a three-tier Web application. The transformation allows developers to generate an implementation specification, i.e. the source code of the presentation and logic tier of a Web application, from a SQL statement. It consists of a development method and three transformations. The transformation is to produce (i) a query form page, (ii) a query result page, and (iii) a Web services method that implement the SQL statement. To result in ready-to-deploy source code, developers must supply the transformations with additional information.

**Keywords:** *model-driven engineering; transformation; SQL statements; web applications; three-tier architecture*

### I. INTRODUCTION

Online transaction processing (OLTP) information systems collects various transaction data from the business processes of an enterprise. Nowadays most OLTP systems use relational databases as their main storages. SQL (structured query language) is used to manipulate the data in the databases.

Basically, the operations that an OLTP system does are (i) create a new transaction record; (ii) read one or more records, either for presenting the existing records or for supporting the creation of a new record, e.g. a value for the new record must be taken from a set of existing values (lookup table); (iii) update an existing record; and (iv) delete one or more records from the database. These operations correspond respectively to basic SQL statements, i.e., insert, select, update, and delete.

As web technologies are getting mature, more OLTP systems are developed as Web applications and deployed on the Internet. A Web application allows an enterprise to reach more customers. This is also driven by the fact that customers are ready and expect to be able to make transactions on the Internet. On October 2011, Site Analytics

(<http://siteanalytics.compete.com>) reported that Amazon and eBay have 468,825 and 535,243 unique visitors in a month (US data only), respectively.

A Web application can use either two-tier or three-tier architecture [1], as depicted in Fig. 1. The two-tier architecture is also called client-server architecture. In a client-server Web application, a user interacts with a web client that is responsible for providing (graphical) user interface and for implementing the business logic of the application. This web client interacts with database server(s). This architecture is simple and, therefore, best for applications with small number of users and a single database [2]; but its performance will suffer from heavy traffic from a large number of users.

In three-tier architecture, a Web application consists of a presentation, logic, and data tiers. The presentation tier is made up of a Web client that provides user interface only. The logic tier provides the business logic of the application and resides on an application server. It can be implemented, e.g. using Web Services, EJB, or .NET. The data tier stores the application's data on database server(s).

Although its structure and development process is more complex, a three-tier Web application offers the following advantages [1][2][3].

- **Performance.** The separation between the presentation, logic and data functionality allows better load balancing between their corresponding servers.

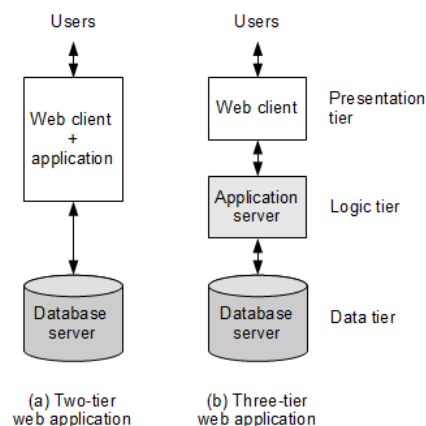


FIGURE 1. TWO ARCHITECTURE TYPES OF WEB APPLICATIONS

- *Flexibility.* Any change or modification of the functionality implementation in one tier does not affect the implementations in the other tiers.
- *Security.* More security policies can be enforced within the servers.

It should be noted that, in either architectures, some business logic can be implemented on database server as functions or stored procedures.

The objective of this paper is to propose a framework for transforming SQL statements to three-tier Web applications. The purpose of this framework is to allow rapid development of OLTP information systems as Web applications. Since this framework involves minimal manual effort from developers, the development cost can be reduced. This framework also offers code consistency and less error on the generated source code of a Web application. In turn, this potentially lowers the maintenance cost. In this paper, we focus on the transformation of SELECT statements.

This paper is further structured as follows. Section II discusses current development (*state-of-the-art*) on the automatic generation of Web applications. Section III presents an overview of our transformation framework. Section IV presents in more detail our transformation. Section V discusses the results produced by our transformation. Finally, section VI concludes this paper and identifies future work.

## II. MODEL-DRIVEN ENGINEERING OF WEB APPLICATIONS

Model-driven engineering (MDE) [4][5][6][7] has been widely applied in the development of Web applications [8] [10][14][15][16][18][19][20]. MDE is a development methodology that focuses on the separation of concerns between functionality and implementation specification (on a specific technology platform). In MDE, an implementation specification is obtained from the application of a transformation on a functionality specification. When a tool support for such transformation is available, MDE can reduce development cost, improve implementation quality, and speed up the development process.

A model-driven development methodology of three-tier Web applications is proposed in [8]. This methodology allows a developer to specify a Web application in three models, i.e., a data model in ERD (Entity-Relationship Diagram); a hypertext model for navigating between pages in a visual notation called *WebML* [9]; and a presentation model defining how the information should be delivered to the users. A combination of presentation and navigation models represents a workflow or business process supported by the Web application. Fig. 2 illustrates the application of this methodology.

This methodology provides a transformation that automatically generate executable code in JEE or .NET. The behavior of a Web application, including SQL statements, is taken into account during code generation. The generated code requires an application server with a specific runtime layer to execute. In [10], this methodology is used to develop context-aware web applications.

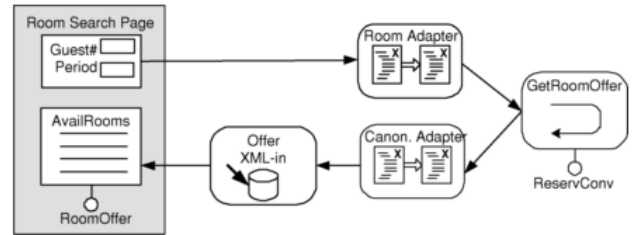


FIGURE 2. A MODEL-DRIVEN DEVELOPMENT METHODOLOGY IN [8]

Another methodology for development of three-tier Web application called *webSA* allows a developer to specify a Web application in four models, i.e., content, navigation structure, business process, and presentation [11]. This methodology integrates functional and architectural models of a Web application. The functional model can be specified in WebML [9], OO-H [12], or UWE [13]. This methodology provides a transformation to generate the application's architecture in JEE (that includes HTML, JSP and Web Services) or .NET.

Reference [11] illustrates the development of a Web application using *webSA*, in which UWE is used to specify the functional model. In [14], OO-H is used to specify the user interface and navigation structure of a Web application.

A model-driven development methodology called *AutoWeb* specifies a Web application in three models, i.e., structure, navigation and presentation [15]. This methodology provides two transformations (called *page generator* and *prototype generator*) of the models to a database schema and a two-tier Web application.

A development framework called *visualWADE* [16] specifies a two-tier Web application in three models, i.e., structure (information content), navigation, and presentation. All models are defined in UML. The framework is able to generate an implementation in PHP.

Several researches on model-driven engineering focus on the development of service compositions. The researches investigate the development of a Web services development framework or transformation from a workflow or business process model, e.g., in UML activity diagram or ISDL (Interaction System Design Language [17]), to executable code in BPEL (Business Process Execution Language) [18][19][20]. A service composition can be used as the logic tier of a Web application.

Using the methodologies or frameworks described above, a Web application is specified in several models; each of which represents different aspect of the application, e.g., content (data or information structure), hypertext (navigation structure), business process (control flow), and presentation. Different modeling languages or notation is required for different models because a modeling language is sufficiently expressive for its intended use only.

The methodologies introduce or use non-standard modeling languages, e.g., WebML, OO-H, ISDL, and extensions of UML. To be able to use the methodology, developers must be fluent in those languages. This requirement may hinder the acceptance of the methodology.

The use of standard languages is a determining factor for developers to accept and use a methodology.

OLTP information systems are used by most of the methodologies for illustration. Such information system is typically based on a relational database that makes use of SQL statements for data manipulation. However, from the descriptions of the methodologies, it is not clear when and where SQL statements should be embedded in the models. In the development of service compositions, SQL statements do not play a significant role and, thus, may be abstracted in a service composition model.

A transformation from functionality to implementation specification, including executable code, may benefit from the decomposition of the transformation into sub-transformations [11][19]. Such decomposition allows a large and possibly complex transformation to be developed as a number of smaller and simpler transformations. It also allows reuse of the sub-transformations.

### III. TRANSFORMATION FRAMEWORK

This section presents an overview of our transformation framework. Our framework targets a Web application whose presentation tier is in PHP programming language; logic tier is implemented as a Web service in PHP and NuSOAP [21] toolkit; and data tier uses MySQL database. The framework assumes that the database is available. No specific runtime support is required to execute the resulted Web application.

Fig. 3 shows an overview of our transformation framework. It consists of five main activities. *First*, a developer creates a SQL SELECT statement for querying a target database. This query specifies which data should be retrieved from which database tables under some condition. *Second*, the developer tests the SQL statement by execute it on the target database. The developer evaluates whether the query results in the expected data. *Third*, when the developer confirms that the data are as expected, the developer transforms the SQL statement to the source code of a three-tier Web application. Otherwise, the developer must correct the SQL statement until it results in the expected data. *Fourth*, the developer deploys the source code resulted from the transformation on the target servers. *Fifth*, the Web application is now ready and can be executed.

In order to transform a SQL statement to the source code of a Web application, the framework includes a set of transformations as depicted in Fig. 4. Transformation *T1a* and *T1b* receive a SQL statement and produce the source code of a query form page and a query result page, respectively. These pages constitute the presentation tier of a Web application. Transformation *T2* receives a SQL statement and produces the source code of a Web services method. This source code constitutes the logic tier of a Web application.

To produce correct and complete source code, the transformations must be supplied with information about the target database on which the SQL statement should run. Other information such as target file names, namespace, and application server's URL, must also be supplied to the transformations.

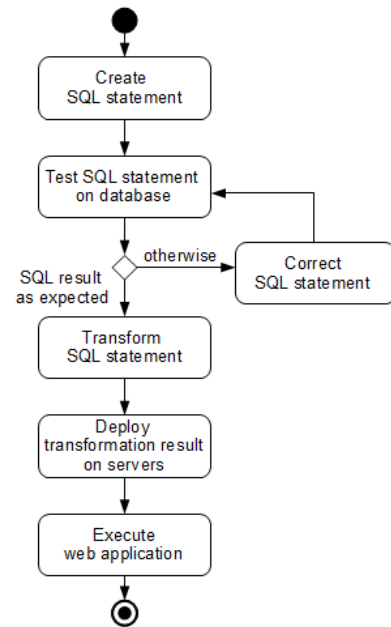


FIGURE 3. FRAMEWORK FOR RAPID DEVELOPMENT OF THREE-TIER WEB APPLICATIONS

Fig. 5 depicts how the source code resulted from the transformation should be deployed. The query form and the query result pages are to be deployed on a Web server. The business logic is to be deployed on an application server or a Web services container. Typically a Web services container can automatically generate the description of the services that are deployed on them (in Web Services Description Language, WSDL). Our framework thus does not include a transformation that is to produce service description.

After deployment on the target servers, the Web application can now be executed. The execution flow is depicted in Fig. 6. The execution order is indicated by the numbers in the figure. (1) A user fills in values on the text fields on a query form page. (2) The user submits these values to a query result page. (3) This query result page invokes a Web services method (business logic) by passing these values as parameters. (4) The business logic uses these values to query the database. (5) The database sends back the resulted data to the business logic. (6) The query result from the database is sent back to the query result page as the Web services method's return values. (7) The query result page formats the values and presents them to the user.

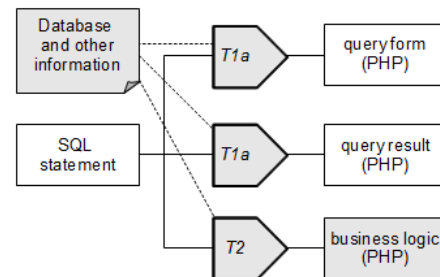


FIGURE 4. TRANSFORMATION OF SQL STATEMENT

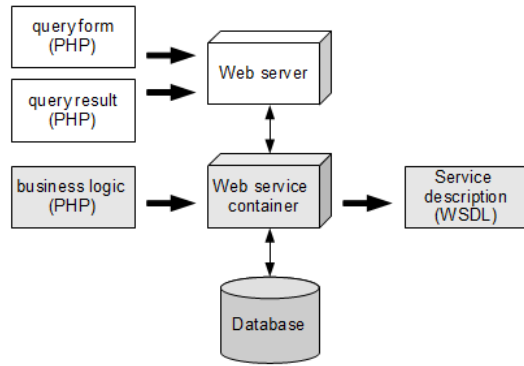


FIGURE 5. DEPLOYMENT OF TRANSFORMATION RESULTS

#### IV. TRANSFORMATION METHOD

The full syntax of a SELECT statement in MySQL is depicted in Fig. 7. For each transformation, we identify which parts of a SELECT statement are needed to produce the output. A template is used to ease the code generation.

A `select_expr` indicates the columns of the specified tables from which data will be retrieved. The columns can be given alias names. These alias names can be used to refer to the column in GROUP BY, ORDER BY, or HAVING clauses. A `select_expr` can be an asterisk (\*) to indicate all columns. Our transformations have not yet considered the use of alias names or asterisk.

In order to make clearer the description of our transformations, we use an example query in Fig. 8. We use a colon symbol preceding a variable name, e.g., `:category`, to indicate that the user must supply a value for that variable during execution.

##### A. Transformation *T1a*

Given a query, transformation *T1a* produces a query form page. This page is used by a user to supply values needed by the query to execute.

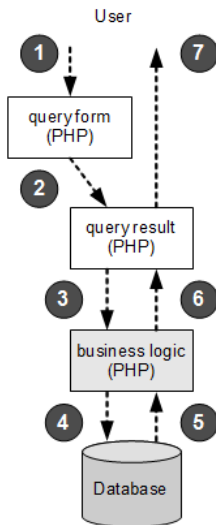


FIGURE 6. EXECUTION FLOW

```

SELECT
  [ALL | DISTINCT | DISTINCTROW]
  [HIGH_PRIORITY]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT][SQL_BIG_RESULT]
  [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE]
  [SQL_CALC_FOUND_ROWS]
  select_expr [, select_expr ...]
[FROM table_references]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
  [ASC | DESC], ...]
[LIMIT [{offset,} row_count | row_count
  OFFSET offset]]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name' export_options
  | INTO DUMPFILE 'file_name'
  | INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
    
```

FIGURE 7. SELECT SYNTAX IN MYSQL [22]

```

SELECT
  course.category, course.name, user.name
FROM course, user
WHERE course.category = :category
  AND course.name LIKE :coursename
  AND SUBSTR(course.id, 1,9) = user.id
ORDER BY course.name ASC
    
```

FIGURE 8. AN EXAMPLE QUERY

Transformation *T1a* identifies variables specified in `where_condition` and generates a two-column HTML table as depicted in Fig. 9 (table borders are made visible for clarity). For each row, the first column contains a variable name; and the second column contains a text field on which the user must supply a value. In the last row, this transformation adds a *Search* button.

The data filled in on the query form page will be sent to the Web server using the POST method. The transformation needs the following additional information:

- the name of the HTML form;
- the address of the Web server to which the data should be sent; and
- the query result page that handles the query.

This transformation puts this information as the attributes of a HTML form (see the bold texts in Fig. 10) that encloses the definition of the table of Fig. 9.

<b>category</b>	<input type="text"/>
<b>coursename</b>	<input type="text"/>
	<input type="button" value="Search"/>

FIGURE 9. QUERY FORM PAGE FOR THE EXAMPLE QUERY

```
<form name="queryform1" method="POST"
  action="http://web.server.com/
  queryresult1.php">
```

FIGURE 10. THE HTML FORM OF THE EXAMPLE QUERY FORM PAGE

**B. Transformation T1b**

Given a query, transformation *T1b* produces a query result page. During execution, this page does the following activities. *First*, the query result page receives and handles the value sent by the query form page. To support this activity, transformation *T1b* identifies the variables specified in `where_condition` of the query; and generates PHP variables and their assignments statements. Part 1 of Fig. 11 depicts the source code produced by this transformation for the example query.

*Second*, the query result page invokes a Web services method. To generate the corresponding source code, the transformation needs the following additional information (bold text in Part 2 of Fig. 11):

- the address of the target application server; and
- the Web services method to invoke.

*Third*, the query result page receives the return value(s) from the method invocation. To support this activity, transformation *T1b* identifies the variables specified in `select_expr`; and generates a list of parameters to be passed to the Web services method. This transformation needs additional information, i.e., the name of the method to invoke. Part 3 of Fig. 11 depicts the generated source code (additional information is in bold text).

*Fourth*, the query result page formats the return values in a HTML table. Transformation *T1b* identifies columns specified in `select_expr`; and generates a HTML table whose number of columns is as many as the number of identified columns. The names of the identified columns are used as the columns' headers. The transformation populates the HTML table using the values returned from the Web services method invocation. Part 4 of Fig. 11 depicts the source code produced by this transformation for the example query.

Execution of the source code results in a HTML table depicted in Fig. 12. Values 523 and %algorithm% are given to variables `:category` and `:courseName` of the example query, respectively.

**C. Transformation T2**

Given a query, transformation *T2* produces the source code of a Web services method that implements the query. The source code consists of the implementation of a Web services method and the registration of that implementation to a Web services container.

For implementing the Web services method, this transformation needs the following additional information (bold text in Part 1 of Fig. 13):

- the name of the method,

- the address of the target database server,
- a username and password to access the database, and
- the database name.

```
// 1. handle data from query form page
$category = $_POST["category"];
$courseName = $_POST["courseName"];

// 2. invoke web services method
require_once('nusoap.php');

$client = new soapclient('http://
  app.server.com/example/query1.php');

// 3. receive return values
$result = $client->call('query1', array(
  'category' => $category;
  'courseName' => $courseName));

// 4. format return values in table
echo "<table border='1'>
  <tr>
    <th>course.category</th>
    <th>course.name</th>
    <th>user.name</th>
  </tr>";

while($row = mysql_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row[0] . "</td>";
  echo "<td>" . $row[1] . "</td>";
  echo "<td>" . $row[2] . "</td>";
  echo "</tr>";
}
echo "</table>";
```

FIGURE 11. WEB SERVICES METHOD IMPLEMENTATION OF THE EXAMPLE QUERY

course.category	course.name	user.name
523	Algorithms and Programming 1	Zuhkri
523	Algorithms and Programming 2	Rahmadi

FIGURE 12. TABLE PRESENTING THE RESULT OF THE EXAMPLE QUERY

Transformation *T2* then constructs a proper query by replacing the variables in `where_condition` of the given query with the corresponding PHP variables.

For registering the method, this transformation needs the following additional information (bold text in Part 2 of Fig. 13):

- the target namespace, and
- the name of the method.

V. DISCUSSION

Given a valid query, our transformation framework can produce the source code of a three-tier Web application in seconds. The query form and query result pages of the presentation tier are very basic. A Cascaded Style Sheet (CSS) can be used to make the presentation more attractive.

A survey that studied five research projects and 47 development tools for data-intensive Web applications [1] grouped the tools into six categories: (1) visual editors and site managers; (2) Web-enabled hypermedia authoring tools; (3) Web and database integrators; (4) Web forms editor, report writers, and database publishing wizards; (5) multiparadigm tools; and (6) model-driven application generators. We consider that our framework is in category 4 and 6. Tools in category 4 aim at increasing productivity to rapidly deploy Web applications. Tools in category 6 provide highest level of support to the development of Web applications, from conceptualization to automatic generation of the implementation.

Some of the studied tools allow automatic code generation from a design specification, e.g. models in visual notation or database schema (tables or views). Procedural scripts need to be included manually to the models or to the generated Web applications. Our transformations receive a SQL statement and some additional information to generate ready-to-deploy source code of a Web application. No procedural script needs to be included manually.

The difference between declarative (SQL statement) and procedural paradigms increases the development time of and limit optimization opportunities in a Web application. For this reason, a language called *Hilda* was introduced to design data-intensive Web applications [23]. An automatic tool generates a relational database and Java servlets from a design specification in *Hilda*. Another language called *WebML* was introduced to specify Web applications [24]. A developer specifies visually the content of a Web application page and how it should be rendered. Such a non-standard language might be effective for code generation, but it might hinder developers from using it because of interoperability issues.

```
// 1. Method implementation
function query1($category, $courseName) {

    $con = mysql_connect("db.server.com",
"username", "password");

    mysql_select_db("database", $con);

    $query = "SELECT course.category,
UCASE(course.name), user.name
FROM course, user
WHERE course.category = " . $category . "
AND course.name LIKE '%" . $courseName .
"%' AND SUBSTR(course.id, 1,9) = user.id
ORDER BY course.fullname ASC";

    $result = mysql_query($query);

    mysql_close($con);

    return $result;
}

// 2. Registration of Web services method
require_once("nuSOAP/lib/nusoap.php");

$server = new soap_server();
```

```
$namespace = "http://example.com/webapp1";
$server->wsdl->schemaTargetNamespace =
    $namespace;

$server->configureWSDL("query1");
$server->register('query1');
```

FIGURE 13. WEB SERVICES METHOD IMPLEMENTATION OF THE EXAMPLE QUERY

A number of researches developed transformation from or to SQL, e.g., generation of SQL and stored procedure from the mapping between data sources and data warehouse [25]; transformation of SQL data definition language (DDL) to ontology in order to support semantic Web [26]; and transformation of entity-relationship query language to SQL [27]. Our transformation differs from those researches, i.e., SQL data manipulation language (DML) as input and the source code of a three-tier Web application as outputs, which need different methods.

## VI. CONCLUSIONS AND FUTURE WORK

We have presented a framework for rapid development of OLTP information systems as three-tier Web applications. This framework includes transformations of SQL statements to Web applications using the MDE approach. In this way, the framework offers better implementation quality at a reduced cost and development time, which will ultimately increase business competitiveness of an enterprise.

The transformations receives a SQL statement and results in a query form page, a query result page, and a Web services method implementing the business logic. To produce ready-to-deploy source code, the transformations need additional information, e.g., server name, database name, method name, and form name.

In this paper, we focus on the transformation of SELECT statement. Our future work will transform INSERT, UPDATE and DELETE statements in order to fully support the rapid development of three-tier OLTP Web applications. Furthermore, we will extend our framework so that it can integrate several SQL statements into a complete web application with minimal intervention from developers. This framework should be able to receive a business process model (e.g., in BPMN [28]), whose activities include SQL statements, and to automatically generate a business process navigation constraints between pages.

Our transformations are implemented in Java. A simple custom-made parser is developed to identify parts of a query. Although the parser is sufficient for handling queries with simple or medium complexity; it is not robust enough to handle queries with higher complexity, e.g., a SELECT statement whose where\_condition contains another SELECT statement. We are considering using ANTLR [29] to improve our parser.

## REFERENCES

- [1] P. Fraternali, "Tools and Approaches for Developing Data-Intensive Web Applications: A Survey," *ACM Computing Survey (CSUR)*, vol. 31, no. 3, 1999, pp. 227-263.
- [2] B. Furht, C. Phoenix, J. Yin and Z. Aganovic, "An Innovative Internet Architecture for Application Service Providers," *Proc. 33rd Annual Hawaii Intl. Conf. on System Sciences*, 2000, pp. 1-10.
- [3] A.O. Ramirez, "Three-Tier Architecture," *Linux Journal*, no. 75, 2000.
- [4] S. Kent, "Model-Driven Engineering," *Integrated Formal Methods*, LNCS, vol. 2335, 2002, pp. 286-298.
- [5] D.C. Schmidt, "Model-Driven Engineering," *IEEE Computer*, vol. 39, no. 2, 2006, pp. 25-31.
- [6] OMG, "Model Driven Architecture (MDA)," *ormsc/2001-07-01*, 2001.
- [7] OMG, "MDA Guide version 1.0.1," *omg/2003-06-01*, 2003.
- [8] I. Manolescu, M. Brambilla, S. Ceri, S. Comai and P. Fraternali, "Model-Driven Design and Deployment of Service-Enabled Web Development," *ACM Trans. Internet Technology (TOIT)*, vol. 5, no. 3, 2005, pp. 439-479.
- [9] S. Ceri, P. Fraternali and M. Matera, "Conceptual Modeling of Data-Intensive Web Applications," *IEEE Internet Computing*, vol. 6, no. 4, 2002, pp. 20-30.
- [10] S. Ceri, F. Daniel, M. Matera and F. M. Facca, "Model-Driven Development of Context-Aware Web Applications," *ACM Trans. Internet Technology (TOIT)*, vol. 7, no. 1, 2007, pp. 1-32.
- [11] S. Melia, A. Kraus and N. Koch, "MDA Transformations Applied to Web Applications Development," *Web Engineering*, LNCS, vol. 3579, 2005, pp. 883-905.
- [12] J. Gomez, C. Cachero and O. Pastor, "Extending a Conceptual Modelling Approach to Web Application Design," *Advanced Information Systems Engineering*, LNCS, vol. 1789, 2000, pp. 79-93.
- [13] N. Koch and A. Kraus, "The expressive Power of UML-based Web Engineering," *Proc 2nd Intl. Workshop on Web-oriented Software Technology*, 2002, pp. 105-119.
- [14] S. Melia and J. Gomez, "The webSA Approach: Applying Model-Driven Engineering to Web Application," *J. of Web Engineering*, vol. 5, no. 2, 2006, pp. 121-149.
- [15] P. Fraternali and P. Paolini, "Model-driven Development of Web Applications: the AutoWeb System," *ACM Trans. Information Systems (TOIS)*, vol. 18, no. 4, 2009, pp. 323-382.
- [16] J. Gomez "Model-Driven Web Development with VisualWADE," *Web Engineering*, LNCS, vol. 3140, 2004, pp.611-612.
- [17] ASNA, "ISDL Home," <http://isdl.ctit.utwente.nl/>
- [18] K. Baina, B. Benatallah, F. Casati and F. Toumani, "Model-Driven Web Service Development," *Advanced Information Systems Engineering*, LNCS, vol. 3084, 2004, pp. 290-306.
- [19] T. Dirgahayu, D. Quartel, and M. van Sinderen, "Development of Transformations from Business Process Models to Implementations by Reuse," *Proc. 3rd Intl. Workshop on Model-Driven Enterprise Information Systems*, 2007, pp. 41-50.
- [20] T. Dirgahayu, D. Quartel, and M. van Sinderen, "Transforming Internal Activities of Business Process Models to Services Compositions," *Proc. 4th Intl. Workshop on Model-Driven Enterprise Information Systems*, 2008, pp. 56-63.
- [21] NuSOAP, "SOAP Toolkit for PHP," <http://nusoap.sourceforge.net/>
- [22] MySQL, "MySQL 5.0 Reference Manual – SELECT Syntax," <http://dev.mysql.com/doc/refman/5.0/en/select.html>
- [23] F. Yang, J. Shanmugasundaram, M. Riedewald and J. Gehrke, "Hilda: A High-Level Language for Data-Driven Web Applications," *Proc. 22nd Intl. Conf. on Data Engineering*, 2006, pp. 32-43.
- [24] S. Ceri, P. Fraternali and A. Bongio, "Web Modeling Language (WebML): A Modeling Language for Designing Web Sites," *Computer Networks*, vol. 33, no. 1-6, 2000, pp. 137-157.
- [25] R. Rifaieh and N.A. Benharkat, "Query-based Data Warehousing Tool," *Proc. 5th ACM Intl. Workshop on Data Warehousing and OLAP*, 2002, pp. 35-42
- [26] S.H. Tirmizi, J. Sequeda and D. Miranker, "Translating SQL Applications to the Semantic Web," *Database and Expert Systems Applications*, LNCS, vol. 5181, 2008, pp. 450-464.
- [27] U. Hohenstein, "Automatic Transformation of an Entity-Relationship Query Language into SQL," *Proc. 8th Intl. Conf. on Entity-Relationship Approach to Database Design and Querying*, 1990.
- [28] OMG, "Business Process Model and Notation version 2.0," *formal/2011-01-03*, 2011.
- [29] T.J. Parr and R.W. Quong, "ANTLR: A Predicated-LL(k) Parser Generator," *Software: Practice and Experience*, vol. 25, no. 7, 1995, pp. 789-810.