

## **ERRDS: A CASE TOOL TO GENERATE AN ER DATA MODEL FROM A RELATIONAL DATABASE SCHEMA**

NABIL ARMAN  
Palestine Polytechnic University  
Hebron, Palestine

**Abstract:** A relational database (RDB) schema is a description of database requirements in terms of a set of relations and a set of integrity constraints. An Entity-Relationship(ER) data model is a high-level conceptual data model that is used frequently for the conceptual design of databases. ER data models represent a concise description of users' data requirements without including implementation details. Because of that, ER data models are usually used to communicate with non-technical users since they are easier to understand. Some relational database designers used the concept of a universal relation and perform normalization to come up with the relational database schema, without developing an ER data model. We advocate that the best practice for a relational database design is to start with developing a conceptual schema like an ER data model and then map it to a relational database schema (as many CASE tools support). In this article, a case tool to perform the reverse process, which is generating an ER data model from a relational database schema, is presented. This tool is very useful in obtaining a conceptual schema from a relational database schema. This tool can also be thought of as a kind of reverse engineering case tool that aids in the reverse-engineering of legacy databases to consider new implementation technology options.

**Keywords:** Conceptual schema, ER models, automated software engineering, case tools.

### **1. INTRODUCTION**

The relational data model represents the relational database as a collection of relations, where each relation resembles a table of values. Various constraints on data can be specified in the form of relational constraints, including domain constraints, key constraints, and referential integrity constraints.

An ER data model is a high-level conceptual data model that is used frequently for the conceptual design of databases. ER data models represent a concise description of user's data requirements without including implementation details. Because of that, ER data models are usually used to communicate with non-technical users since they are easier to understand. The main building blocks of an ER diagram are entity types and relationship types. The process of mapping an ER data model to a relational database schema is well-documented in the literature [1, 2]. Many CASE tools are capable of doing this mapping process automatically like

Oracle Designer, ER Win, etc. On the other hand, little emphasis has been given to the reverse process, which is generating an ER data model from a relational database schema. In this article, a CASE tool that can generate an ER data model from a relational database schema is presented. In addition, the details of the algorithm that is used are explained.

On one hand, some vendors invest in developing integrated CASE tools, which can aid in the whole system development process. However, these systems are expensive to be purchased and justified for small-scale systems. On the other hand, there is an increasing interest in developing CASE tools that aid in a specific phase of system development, such as the normalization process of a relational database system [4]. A tool that performs the identification of composition relationships for UML Class Diagrams is presented in [5]. The modeling of web-based dialog flows for automatic dialog control is explained in [6]. Software systems unit test selection based on operational violations is presented in [7]. These tools are meant to aid in specific task rather than being used in the whole system development process. ERRDS represents a system, in this category, that is aimed at the reverse-engineering of a relational database schema and obtaining an ER schema from this relational schema.

### **2. GENERATING AN ER DATA MODEL FROM A RELATIONAL DATABASE SCHEMA**

Consider the relational database schema:

$R_1(\underline{A_{11}}, A_{12}, \dots, A_{1i})$   
 $R_2(\underline{A_{21}}, A_{22}, \dots, A_{2j}), FK: A_{22} \rightarrow R_1(A_{11})$   
 $R_3(\underline{A_{31}}, A_{32}, \dots, A_{3k})$   
 $R_4(\underline{A_{41}}, A_{42}, \dots, A_{4l}), FK: A_{41} \rightarrow R_1(A_{11}) \text{ and } FK: A_{42} \rightarrow R_3(A_{31})$   
 $R_5(\underline{A_{51}}, A_{52}, \dots, A_{5m})$   
 $R_6(\underline{A_{61}}, A_{62}, \dots, A_{6n})$

This schema will be used to illustrate the process of generating an ER from a relational database schema. Primary keys are underlined. Foreign keys are represented using arrows where the arrow starts from the referencing attribute and points to the referenced attribute.

A relational database schema consists mainly of the following:

1. Relations
2. Primary Keys
3. Foreign Keys

An ER schema consists mainly of the following:

1. Entity Types
2. Attributes
3. Relationship Types

It is requested to generate ER constructs that can be mapped to the relational database constructs. The process is outlined in the following steps:

1. Relations: Relations are mapped to Entity Types. Primary keys of the relations become the entity types' key attributes. Relations' attributes become the entity types' attributes. For example,  $R_1(\underline{A_{11}}, A_{12}, \dots, A_{1i})$  is mapped to an entity type with a key attribute  $\underline{A_{11}}$  and regular attributes  $A_{12}, \dots, A_{1i}$ .

2. Foreign Keys: A foreign key is represented by an attribute or a set of attributes named "Referencing Attribute(s)" in the referencing relation that refer to an attribute or set of attributes named "Referenced Attribute(s)" in another relation or in the same relation when there is a recursive relationship type. Foreign keys in a relational database schema represent the relationship types in an ER schema. Therefore, they are mainly used to specify the relationship types between/among entity types. A set of different cases may occur:

- 2.1 Relations with one foreign key: If a relation has one foreign key then there is a relationship type between the entity types that represents this relation and the entity type that represents the relation having the "Referenced Attribute". The cardinality ratio could be a 1:1 or N:1. Since 1:1 is a special case of N:1, we generally choose N:1. For example, FK:  $A_{22} \rightarrow R_1(A_{11})$  means that there is a relationship type between the entity type representing  $R_2$  and the entity type representing  $R_1$ .
- 2.2 Relations with two foreign keys: If a relation has two foreign keys then there is a relationship type between the entity types that participate in this relationship (no entity type in this case). The entity type that represents the relation having the first "Referenced Attribute" and the entity type that represents the relation having the second "Referenced Attribute" become the participating entity types in this relationship type. The cardinality ratio of this relationship type is M:N. If a relation with two foreign keys has additional attributes, these additional attributes become the

relationship type attributes. For example, the foreign keys FK:  $A_{41} \rightarrow R_1(A_{11})$  and FK:  $A_{42} \rightarrow R_3(A_{31})$  in  $R_4$  means that this relation represents a M:N relationship type between the entity type representing  $R_1$  and the entity type representing  $R_3$  with the remaining attributes in  $R_4$  representing relationship type attributes.

- 2.3 Relations with more than two foreign keys: If a relation has more than two foreign keys then there is an n-ary relationship type among these entity types that participate in this relationship type (no entity type is generated in this case). The entity type that represents the relation having the first "Referenced Attribute" and the entity type that represents the relation having the second "Referenced Attribute" ...etc, become the participating entity types in this relationship type. The cardinality ratio of this relationship type is M:N in all sides. If a relation with more than two foreign keys has additional attributes, these additional attributes become the relationship type attributes. In fact, this is a generalization of 2.2 and the example in 2.2 applies here.

4. Relations with all primary key attributes: If a relation's primary key consists of another relation's primary key and other attribute(s), then the relation was a result of a mapped multivalued attribute. In this case, the attribute other than the original relation's primary key becomes a multivalued attribute of the original relation. If there were more than one attribute, other than the primary key, then the multivalued attribute is also composite and it becomes a complex attribute in the entity type that corresponds to the original relation. For example, relation  $R_5$  contains  $A_{11}$  as part of its primary key and  $A_{11}$  is  $R_1$ 's primary key. Thus, the attributes of  $R_5$ , other than  $A_{11}$ , represent a multivalued/complex attribute of the entity type representing  $R_1$ .

5. Relations with compound primary keys: If a relation's primary key consists of another relation's primary key and other attributes, with additional attributes not being part of the primary key, then the relation was a result of a mapped weak entity type. In this case, a weak entity type is generated with the attribute(s) that were part of the primary key become the partial key of the weak entity type and the rest of the attributes become the weak entity type attributes. For example,  $R_6$  has  $A_{11}$  as part of its primary key and  $A_{11}$  is  $R_1$ 's primary key (i.e., the entity type representing  $R_1$  is the owner entity type of the weak entity type representing  $R_6$ ). Thus,  $R_6$  represents a weak entity type with  $A_{62}$  as a partial key and the remaining attributes represents the weak entity type attributes.

### 3. DEVELOPMENT PHASES

ERRDS is a CASE tool application that is used to generate a conceptual schema from a relational database schema. The remaining of this section is organized as follows: In section 3.1 the system requirements analysis is presented. Section 3.2 presents software system design. In section 3.3, software system implementation is summarized. Finally, section 3.4 presents software testing and the demo of the system.

#### 3.1 SOFTWARE SYSTEM REQUIREMENTS ANALYSIS

ERRDS functional requirements can be summarized by the following functions:

- Reading Relational Database Schema through a GUI interface or from an XML file. The system input will be a group of relations that represents a certain relational database schema with referential integrity constraints. These constraints are represented using the primary keys and foreign keys.
- Generating an ER schema according to the relations and integrity constraints.
- Producing the ER schema either as a textual description, that can be saved in a text file for future use, or as an XML file with self-explanatory tags.

These major activities represent the major functional requirements of the system. The non-functional requirements are similar to those required by any software application.

#### 3.2 SOFTWARE SYSTEM DESIGN

The software system design consists of two major activities, namely, GUI interface design, and the application design. The GUI interface design is a standard windows application interface with minimal number of controls to improve the efficiency of the system. Sample screens are shown in the testing phase in section 3.4. The application includes functions to handle the input and output in different formats, as explained before. The main algorithm for the generation of the ER schema from the relational database schema is summarized by the following pseudo code:

```
ERRDS_Main_Algorithm(Input: RDB schema,
Output: ER schema)
begin
  for each relation r in RDB schema do
    begin
      Store the input in an array after reading from the
      GUI interface or extracting from an XML file.
      // Check the number of attribute in PK
      if number of attribute in PK==1 then
        The relation represents a regular entity type
      else if number of attribute in PK ==2 and
```

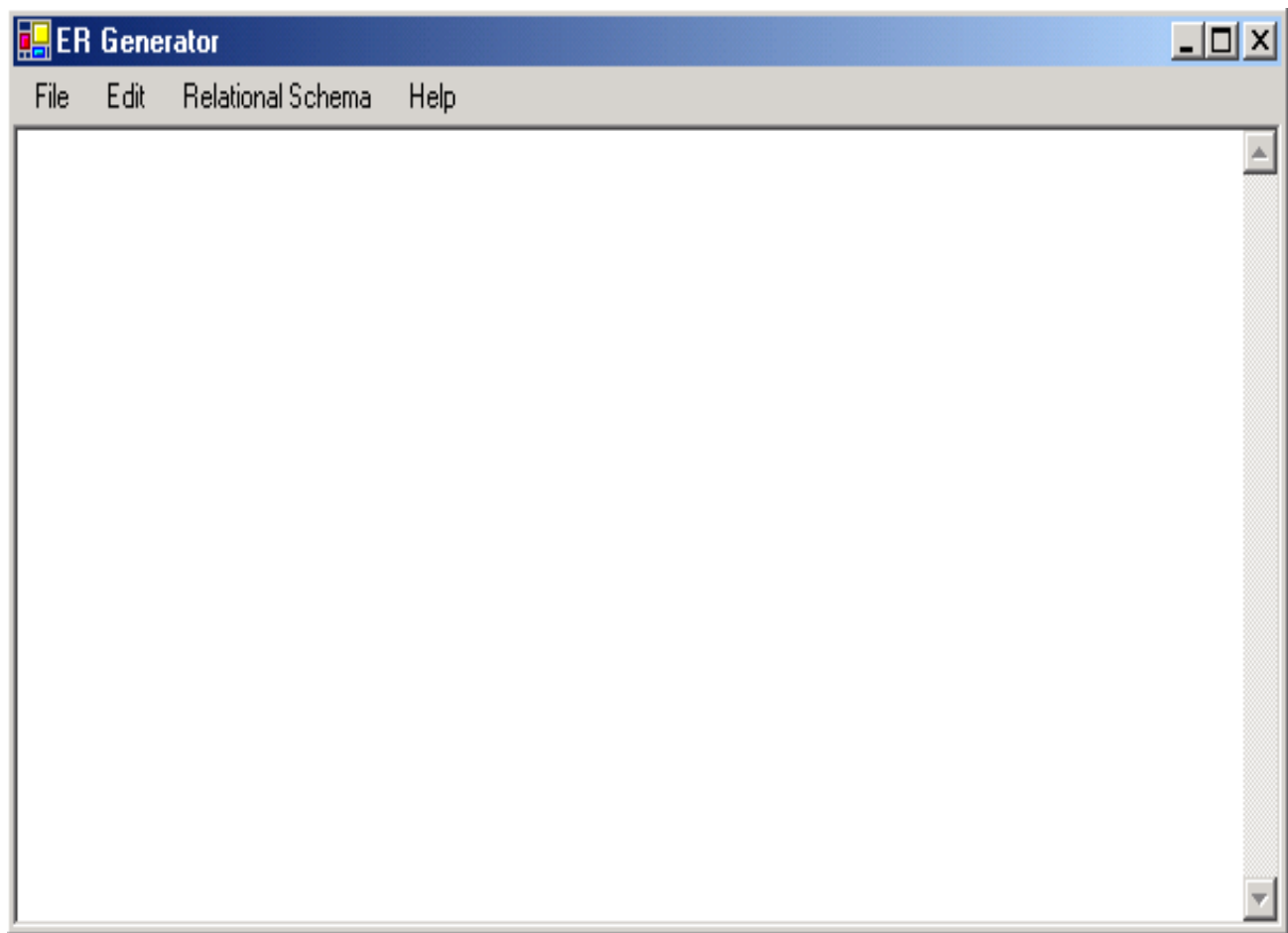
```
the relation is all-key relation then
  The relation represents a multivalued attribute
else if number of attribute in PK>2 and
  the relation is all-key relation then
  The relation represents a complex attribute
else
  The relation represents a weak entity type
// Handling relationship types
if number of FKs==1 then
  The entity type representing relation r
  participates in a relationship type
else if number of FKs=2 then
  if FKs are primary key of the relation then
    The relation represents a binary M:N
    relationship type
  else
    The entity type representing relation r
    participates in two relationship types
else if number of FKs>2 then
  if FKs are primary key of the relation then
    The relation represents an n-ary M:N
    relationship type
  else
    The entity type representing relation r
    participates in n relationship types
  end for
end
```

#### 3.3 SOFTWARE SYSTEM IMPLEMENTATION

ERRDS System was implemented using Microsoft VB.NET, which is part of Microsoft Visual Studio.NET. VB.NET has many advantages, including rich GUI components, improved availability and scalability, simplified development environment, simplified deployment, and improved performance, to name just a few [3].

The main GUI interface, implemented using Microsoft VB.NET, is shown in Figure 1.

The File menu has a number of menu items whose functionality is self-explanatory. It includes an Open menu item to open an existing file. A Save As menu item is used to save the generated ER model description in a file. Finally, the Exit menu item is used to exit the application. The Edit menu has a number of menu items whose functionality is self-explanatory. It includes the standard Edit operations like Copy, Paste and Delete menu items. The Relational Schema menu has two menu items. The User Input specifies that the relational database schema to be input to the application is to be read from a GUI form. The XML File menu item specifies that the relational database schema to be input to the application is to be read from an existing XML file. Finally, the Help menu contains two menu items. The View Help menu item is used to show a number of tips to help and guide the database designer in using the system. The About menu item shows a brief description of the application and other relevant information.



**Figure 1. Main GUI of ERRDS**

The image shows a window titled "Add Relation" with a standard Windows-style title bar. The window contains several input fields and buttons. On the left side, there are four grouped input sections: "Relation Name" with a single text box; "Primary Key" with a text box and a "More Attributes" button; "Foreign Keys" with a text box and a "More Attributes" button; and "Attributes" with three text boxes and a "More Attributes" button. On the right side, there is a vertical stack of buttons: "Save/New Relation", "Process", "Help", and "Close".

**Figure 2. Input Form**

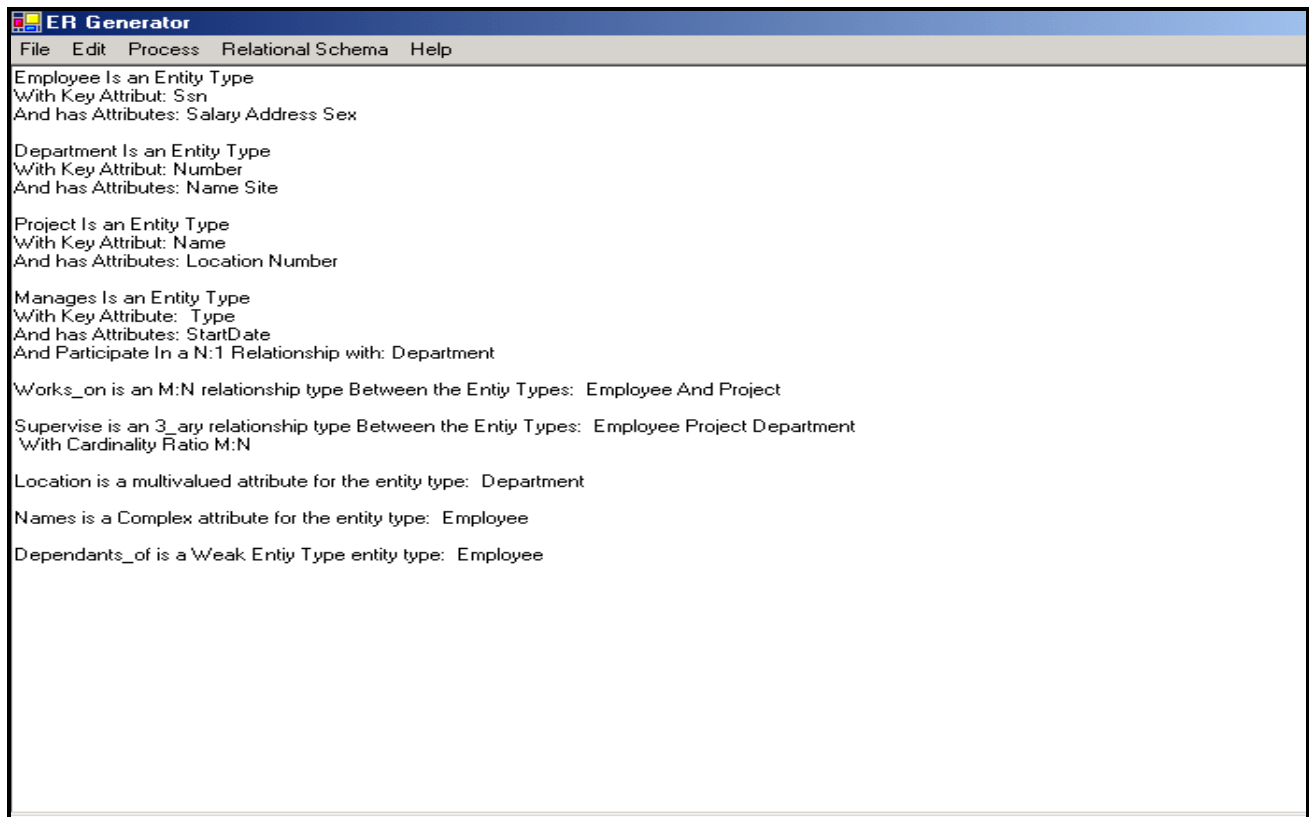


Figure 3. Output Form

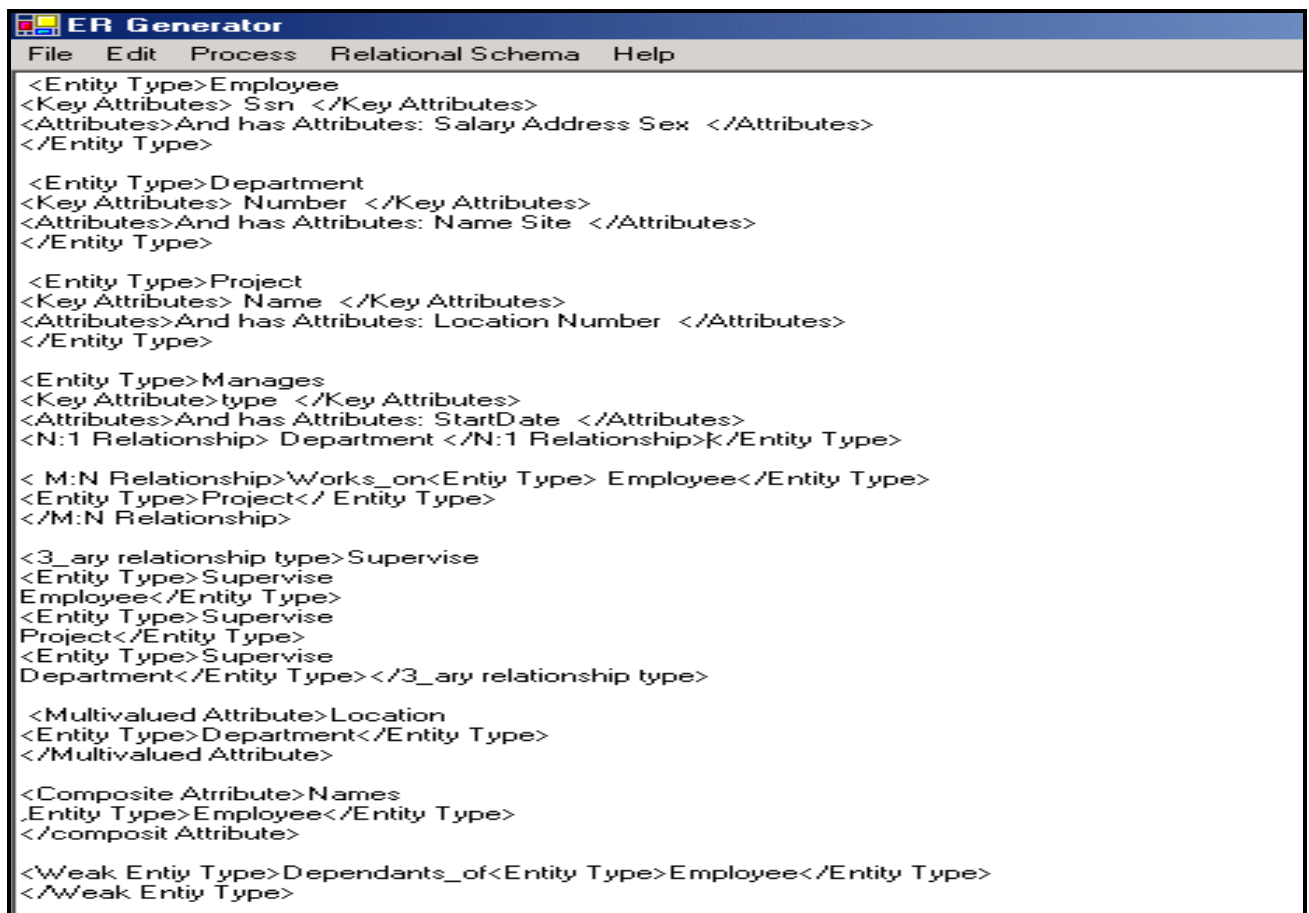


Figure 4. Output in XML Format

### **3.4 SOFTWARE SYSTEM TESTING AND DEMO**

Users/database designers can interact with the system GUI interface by completing simple forms and selecting the appropriate menu items. A user can add a relation using a form as shown in Figure 2.

The output describing an ER schema from a relational database schema is shown in Figure 3.

The input may come from an XML file that can also be viewed from the application. The output of the application can be in XML format as shown in Figure 4. Only the relevant content of the XML document is displayed. Other meta information is not shown to save space.

## **4. KEY SUCCESS FACTORS AND ADVANTAGES**

ERRDS is a CASE tool application aims at providing the relational database designers community with an easily accessible way to generate an ER schema from a relational database schema, instead of purchasing an expensive CASE tool like Oracle Designer or other tools. More specifically, ERRDS system: reduces the efforts to generate an ER schema from a relational database schema using a cost-effective tool. Therefore, ERRDS can be considered as a contribution to the promotion of automated software engineering.

## **5. CONCLUSION**

A CASE tool that generates an ER schema from a relational database schema is presented. This tool can help in migrating legacy databases to newer and more powerful database servers by producing a conceptual schema that is very useful in database development. The tool can also help in maintaining the original database schema. Putting this system in use doesn't mean that the development of the system has ended but more development can be done to improve the system efficiency and its functionality. We are considering the generation of an ER Diagram using a specification format similar to the Microsoft Visio format so that the diagram can be opened in Microsoft Visio. In addition, obtaining relational database schema directly from relational DBMS/servers such as Oracle and MS SQL servers and generating the output as described, is under consideration.

## **REFERENCES**

- [1] Elmasri, R. and Navathe, S., *Fundamentals of Database Systems*, Addison Wesley, 2004.
- [2] Silberschatz, A., Korth, H. and Sudarshan, S., *Database System Concepts*, McGraw Hill, 2005.
- [3] Deitel, H., Deitel, P. and Nieto, T., *Visual Basic.NET How to Program*, 2nd Edition, Prentice Hall, 2002.
- [4] Arman, N., "Normalizer: A Case Tool to Normalize Relational Database Schemas," *Information Technology Journal*, pp. 329-331, Vol. 5, No. 2, ISSN: 1812-5638, 2006.
- [5] Milanova, A., "Precise Identification of Composition Relationships for UML Class Diagrams," *Proceedings of ASE-2005: The 20th IEEE Conference on Automated Software Engineering*, pp. 76-85, IEEE CS Press, November, Long Beach, California, 2005.
- [6] Book, M. and Gruhn, V., "Modeling Web-Based Dialog Flows for Automatic Dialog Control," *Proceedings of ASE-2004: The 19th IEEE Conference on Automated Software Engineering*, IEEE CS Press, November, Linz, Austria, 2004.
- [7] Notkin, X., "Tool-Assisted Unit Test Selection Based on Operational Violations," *Proceedings of ASE-2003: The 18th IEEE Conference on Automated Software Engineering*, IEEE CS Press, November, Montreal, Canada, 2003.