Unified Model for Testing Object-Oriented Application Development Tools

Pavel P. Oleynik <xsl@list.ru>,
Shakhty Institute (branch) of Platov South Russian State Polytechnic University (NPI), Rostov-on-Don, Russia

Abstract. The paper presents a unified model for testing tools for object-oriented application development. Based the available papers were identified shortcomings of existing work and identified the following optimal criteria, which shall comply the resulting model:
1. To deep inheritance hierarchies
2. To presents of multiple inheritance hierarchies
3. To presents of abstract classes in the hierarchy
4. To presents of multiple (n-ary) associations
5. To presents of associations with attributes
6. To presents of a composition between classes
7. To presents of recursive associations
8. To presents of associations between classes belonging to the same inheritance hierarchy
9. To presents of association classes
10. To presents between the association class and other classes
11 To presents enumerations in model
With a unified graphical language UML class diagram unified model testing. The paper we verified compliance with the resulting implementation of the selected criteria was presented.
Currentlythe implementation of applications using object-oriented programming languages and relational databases. To overcome the object-relational mismatch it is necessary to implement object-related mapping patterns presents. The paper presents three methods used to represent the class hierarchy highlighted the advantages and disadvantages of each method.
For test the feasibility a unified model chosen development environment SharpArchitect RAD Studio which is designed object applications in C# and are implementing a relational database. The paper presents the developed object model in the form a class diagram showing the interfaces and inheritance relations diagram containing all the tables and columns the resulting database.
In the conclusion recommendations on the areas for further development work and identified the need of implement a unified model with other approaches proposed by the authors was used.

Keywords: UML, Object modeling, Design of Information Systems, Databases, Object-oriented design, Object-Relational Mapping Patterns, Impedance Mismatch

DOI: 10.15514/ISPRAS-2015-27(3)-7

For citation: Oleynik Pavel P. Unified Model for Testing Object-Oriented Application Development Tools. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 3, 2015, pp. 101-114. DOI: 10.15514/ISPRAS-2015-27(3)-7.

1. Introduction

At the moment there are many tools provide object approach to application development. Despite the existence of their own advantages and disadvantages the main goal is provide the advantages of the developer of object-oriented paradigm. The paper are describes in detail the unified model test tools development of object-oriented applications for demonstration, graphical Unified Modeling Language which used. The practical implementation of the model is demonstrated by the use of classical methods (patterns) object-relational mapping (ORM) in the tool, developed the author. The object model is put into a relational database environment. This approach is most justified from the point of view the author, because the RDBMS is the most popular type of database management systems now.

2. Design of the unified testing model

When designing a unified testing model used the same approach as in the description of the design patterns in [1]. This approach is involves the description of reusable solutions widespread problems in software development without reference to particular domain. The main task of this section – is a description of the model and the structural elements (classes and associations), and not the correctness of the model and the accuracy of its fitness for a particular domain area.

Standard graphical language modeling various aspects of object systems is the language UML. This language is namely structural class diagrams will be discussed in this paper. As a result under the unified model test tools development of object-oriented applications we mean a class diagram, consisting of classes and attributes and containing common practice relationship classes.

The idea of the article is not new and there are works of similar subjects. In [2] has attempted to construct a unified model testing. However, there were no multiple (n-ary) associations and association with attributes that are an integral part of any complex information system.

In [3] presented test model to study the design of object-oriented databases. But the model is relatively simple, which is justified by its purpose. This article used dignity previously existing works and corrected drawbacks of them.

Before designing a unified model testing were nominated optimality criteria (OC) is representing the requirement of a certain structural elements in the class diagram, and which must comply with the finished implementation. Have been put forward the following requirements for the unified model test tools development of object-oriented applications:

1. Must have deep inheritance hierarchies. In realworld applications, very often there are deep hierarchy, is the relational of inheritance and combining transitive least three classes.
2. To presents of multiple inheritance hierarchies. This will show a variety of options and modes available in the development tool.
3. To presents of abstract classes in the hierarchy. Abstract classes cannot have instances in the system and described as a container for attributes and methods used in the inherited (instantiated) classes.
4. To presents of multiple (n-ary) associations. In applications that automate realworld domains, often an association involving three or more classes. Such a relationship is called multiple or n-ary associations.
5. To presents of associations with attributes. Many domains contain attributes that do not belong to certain entities (classes), and their values appear only in the organization of associations between instances of classes. The designing unified model should have associations with attributes.
6. To presents of a composition between classes. Composition - an association between the classes which are Part and Whole. The peculiarity is that the class represents a Part can belong to only one instance of the class that represents the Whole. In this class represents the Whole manages the life cycle is a class represents a Part. When removing the Whole all Parts also deleted. This peculiarity of behavior is very important for many application domains.
7. To presents of recursive associations. Recursive call the association, the ends of which bind the same class. These relationships allow you to implement a hierarchy of subordination.
8. To presents of associations between classes belonging to the same inheritance hierarchy. In terms of implementation is necessary to provide the implementation of the association, the edges of which are associated classes belonging to the same inheritance hierarchy, are represents the base class and the child together.
9. To presents of association classes. Association class - an association which at the same time a class. Especially the use of that class association represents a unique association, i.e. combination of instances of classes in this association is unique.
10. To associationed between the association class and other classes. From a theoretical point of view, the association class is a class, so it can participate in other associations. From the point of view of the implementation of the class association presents a class that contains the attributes (fields or properties of the programming language) that refer to other classes. In turn, for the organization of the association with the class association necessary depending class to create an attribute whose type supports class association.

11. To presents enumerations in model. From a theoretical point of view, enumeration is a set of predefined constants, and the user can not extend this set by adding new values.

In accordance with the selected criteria was implemented hierarchy shown in Fig. 1.

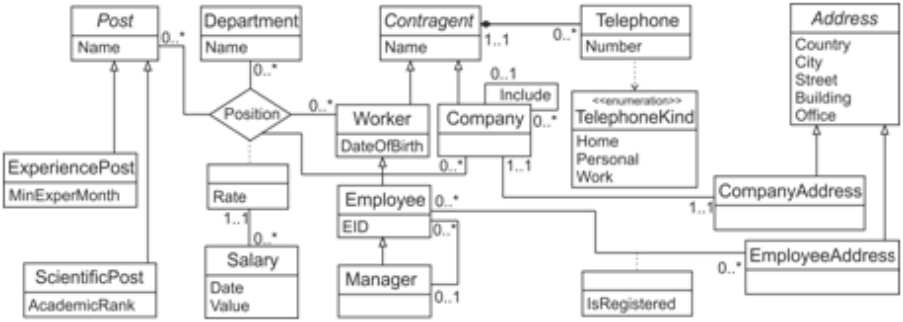


Fig. 1. Unified model for testing object-oriented applications development tools

Consider the appointment of the main classes of diagrams are presented. As mentioned earlier this class diagram is a fictional and is not intended to describe a particular domain therefore contains some illogical (fictional) classes and associations.

For representation of employees and organizations assigned to the base abstract class Contragent. Inherited Company class is present organizations and the class Worker is the base for the employee of organization. Inherited Employee class is an employee and an attribute EID, representing the employee unique number. Class Manager is the staff who are heads of other workers.

Post an abstract class is a position that can be occupied by staff. Inherited class ExperiencePost is a position that requires a minimum amount of experience of the applicant, expressed as number of months (attribute MinExperMonth). The second class is implemented ScientificRank describes the position of the applicant, which requires the presence of a scientific degree, whose name is value in the attribute AcademicRank.

For presentation departments of organizations and entering into an n-ary association a class of Department was introduced. Salary class is paid wages, accrued to employees occupying positions represented by a complex association which called Position.

Class Telephone allows saving the number of phone of company. Phone type (like Home, Personal, Work) represented by enumeration TelephoneKind. For presentation address used by the base abstract class Address. Two derived class CompanyAddress and EmployeeAddress used to represent the address of the organization and address of the employee, respectively.

Check the conformity of the model presented previously selected criteria of optimality. The need for a deep class hierarchy, represented by at least three transitive inherited classes, described OC_1 and implement a class Contragent, Worker, Employee, Manager. In addition to this, there are two hierarchies: 1) Post, ExperiencePost (ScientificPost); 2) Address, CompanyAddress (EmployeeAddress). I.e. the model contains multiple inheritance hierarchies, therefore, the condition OC_2. The presence of abstract classes in the hierarchy due OC_3 and holds classes Post, Contragent and Address.

OC_4 requirements are also performed as there are n-ary association Position, combining classes Post, Department, Worker, Company. Described association has an attribute Rate, which implemented class association and binary association between Employee and EmployeeAddress classes also contains an attribute (IsRegistered) it can be argued that the requirement OC_5 fulfilled.

Each contractor represented derived from Contragent classes, a list of telephone numbers represented instances of Telephone, and both classes related with composition, OC_6 requirement is satisfied. Unified model allows you to store information about a group of companies, organize the tree structure using a recursive association connects Company class with a same. The presence of recursive association dictated OC_7.

In OC_8 written requirement for associations between classes belonging to the same inheritance hierarchy. Figure 1 between classes Employee and Manager provides this association satisfying OC_8. As previously noted, the models have a association class Position, which corresponds OC_9. Described association class is linked with addition association with Salary class. This is a consequence of the implementation OC_{10}. The presence of the models listed due to the implementation of OC_{11}. Of the present disclosure can be seen that the unified model is fully consistent with all previously selected criteria of optimality. Therefore we can move on to the implementation of the unified model.

3. The classical object-relational mapping patterns

To implement of this model development environment software systems based on the organization of the metamodel object system presented in [4-5] was used. This development environment is called SharpArchitect RAD Studio and as storage of information uses a relational DBMS. Because information system is designed in terms of object-oriented paradigm, and implemented in a relational database environment, there is a so-called "object-relational impedance mismatch" to overcome the consequences of which object-relational mapping patterns are used. The most commonly used patterns for represent the class hierarchy.

In SharpArchitect RAD Studio implemented three classic patterns for implementing object-oriented inheritance relationships of classes in a relational structure (relational tables), presented in Fig. 2 [2, 4].

Consider the basic patterns is presented in more detail. Single Table Inheritance pattern physically represents an inheritance hierarchy of classes in a single relational database table whose columns correspond to the attributes of all classes within the hierarchy and allows you to display the structure of inheritance and to minimize the number of joins that must be performed to extract information. In this pattern each instance of the class represented by one row of the table. When you create the object values are entered only in the columns of the table that match the attributes of the class, and all the rest are empty (have a null-value).

The pattern has advantages:

- In the structure of the database contains only one table are representing all classes of whole hierarchy.
- To selection of instances of classes hierarchy do not need to make the joins of tables.
- Move fields from a base class to a derived (as well from the derivative in the base) does not require changes to the structure of the tables.

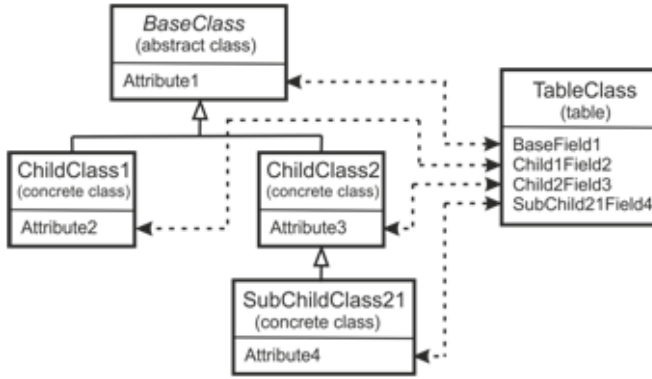
The pattern has disadvantages:

- In the study of the structure of the database tables can cause problems, because not all the columns in the table are intended to describe each domain class. This complicates the process of refining the system in the future.
- If you have a deep inheritance hierarchy with a large number of attributes, many columns can have empty values (null-values). This leads to inefficient use of the available space in the database. However, modern DBMS can compress strings containing a large number of null-values.
- Table may be too large and contain a huge number of columns. The main way to optimize the query (to reduce the execution time) is created a covering index. However, the index set and a large number of queries to a single table can lead to frequent blockages that will have a negative impact on the performance of software applications.

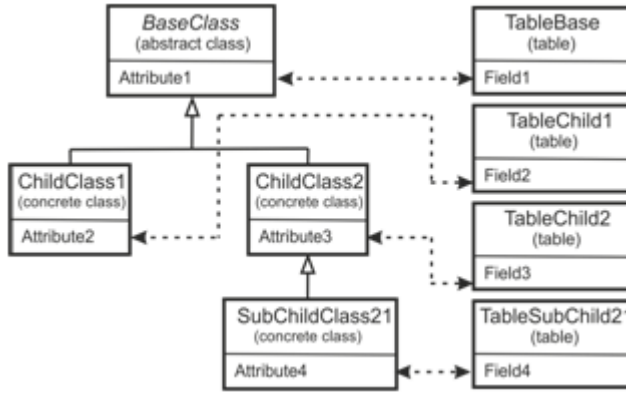
An alternative pattern is called Class Table Inheritance, representing a hierarchy of classes for one table for each class (as an abstract and concrete). Class attributes are mapped directly on the columns of the corresponding table. With this method, the key is the task of joins the respective rows of several database tables that represent a single object of domain.

The pattern has the following advantages:

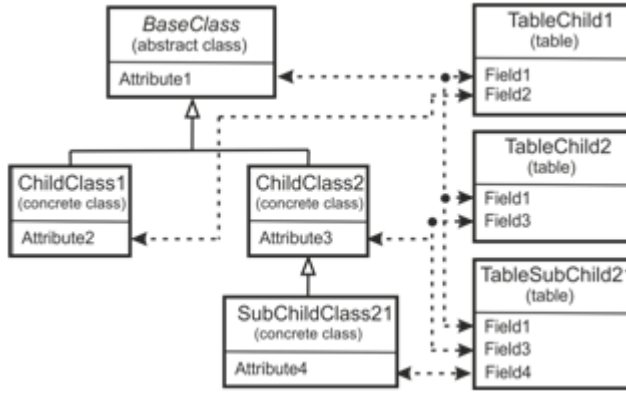
- Each table contains a field, the corresponding attribute of a certain class. The therefore tables are easy to understand and take up little space on your hard drive.
- The relationship between the object model and relational database schema is simple and clear.



Single Table Inheritance pattern



Class Table Inheritance



Concrete Table Inheritance

Fig. 2. Classical object-relational mapping patterns which used to represent the class inheritance in the form of a relational structure (relational tables)

However, there are disadvantages:

- When you are create an instance of a particular class you want to upload data from several tables, which requires either their natural joins or a plurality of database calls followed by join results in memory.
- Move the fields in the derived class or base class requires changes in the structure of several relational tables.
- Base class table can become weaknesses in performance, since access to such tables will be carried out too often, leading to a variety of locks.
- High degree of normalization can be an obstacle to the implementation of unplanned advance queries.

The Concrete Table Inheritance pattern present is an inheritance hierarchy of classes using one table for each concrete (non-abstract) class of the hierarchy. From a practical perspective, this pattern assumes that each instance of the class (object), which is in memory, will be shown on a separate row in the table. In addition, each table in our case contains columns corresponding to attributes as a particular class, so all of his ancestors.

The advantages are that:

- Each table not contains extra fields, so that it is convenient to use in other applications that do not use object-relational mapping tools.
- When creating objects of a certain class in the application memory and retrieve data from a relational database sample is made of a single table, i.e. is not required to perform relational joins.
- Access to the table is carried out only in the case of access to a particular class, thus reducing the number of locks imposed on the table and spread the load on the system.

There are disadvantages:

- Primary keys can be inconvenient by handling.
- There is no ability to model relationships (association) between abstract classes.
- If the class attributes are moved between base classes and derived classes needed to change the structure of several tables. These changes are not as often as in the case of Class Table Inheritance pattern, but they cannot be ignored (as opposed Single Table Inheritance pattern in which these changes are absent).
- If in base class to change the definition of at least one attribute (for example, change the data type), it will require to change the structure of each table representing a derived class because a superclass fields are duplicated in all tables of its derived classes.
- In implementing the method of searching for data in the abstract class is required to view all the tables represents an instance of the derived classes. This requires a large number of database calls.

Selection of an required ORM-pattern depends on the initial logical model, i.e. from the class hierarchy of the domain. At the same time can be used two or more ORM-patterns, which is associated with the need to optimize the structure of a relational database and reduce the number of tables used, which will increase the speed of data retrieval queries.

After describing SharpArchitect RAD Studio object-relational mapping patterns which are available to the developer we can start implementing the unified model for testing tools.

4. Implementation of the unified testing model

In order to simplify the implementation of the three existing class hierarchies in Fig. 1 will separate in available ORM-patterns. The result is shown in Fig. 3.

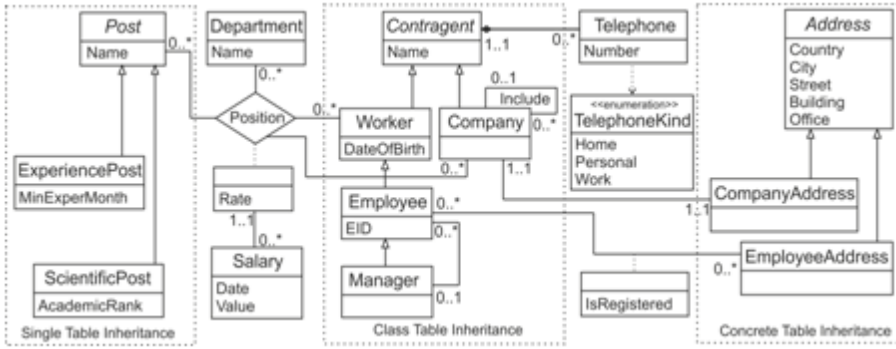


Fig. 3. The using of the classical ORM-patterns for the implementation of the unified model for testing object-oriented applications development tools

The Single Table Inheritance for the class hierarchy Post, ExperiencePost (ScientificPost) was used. As a result, it is assumed that in the RDB will create one single table (relational table), which will be retained instances of all listed non-abstract classes. For the class hierarchy with classes Contragent, Worker (Company), Employee, Manager uses the Class Table Inheritance pattern. I.e. for all classes regardless of whether he or abstract concrete will create a separate table in RDB. Address class is abstract and has no association with other classes in model, so it will not create a separate table in the RDB. And for child classes will be created two tables (one for each heir). I.e. in hierarchy Address, CompanyAddress (EmployeeAddress) was used Concrete Table Inheritance. For other classes outside the hierarchy described, will be created on a separate relation table.

One of the main features of SharpArchitect RAD Studio support multiple inheritance is implemented by means of interfaces C# language construction, as described in detail in [4]. Used C# language does not support this syntax as an association. To represent the binary associations, regardless of the multiplicity was

used properties (property construction), containing a single value or collection of values.

Multiple n-ary association are represents a separate class, the attributes of these associations (as well as the attributes of binary associations) are converted into property of classes. To simplify information searching and extraction of all the associations are bidirectional both ends of the relevant classes there are properties whose type corresponds to the opposite end of the class association. All of the above arguments are presented graphically in Fig. 4.

In implementing the interfaces used language C#, so it is impossible italics abstract classes. Bidirectional associations are shown corresponding arrows connecting classes. In implementing the association used the following approach. From the "one" was declared property, which is a type of list (C# type IList<>), containing the elements, which is a type of class, located on the side "to-many". From the "to-many" is declared in the class property whose type is a class, located on the side "one". Association of the "many-to-many" (without attributes) can be represented by two lists is declared in class antagonisms. In a SharpArchitect RAD Studio development environment has a number of base classes that implement the most common functionality. For example, the class IBaseRunTimeDomainClass is the root of all domain classes. To implement the tree structure will enough inherited from IBaseRunTimeTreeNodeDomainClass. At the time code generation will automatically generate additional attributes Nodes and Owner, allow you to save a reference to the parent and subnodes, respectively. It is implemented in such a way recursive association. For submission to the transfers and sets used syntax construction "enum".

Applying the classical ORM-patterns was obtained relational database schema of the unified model now. Fig. 5 is depicts the result.

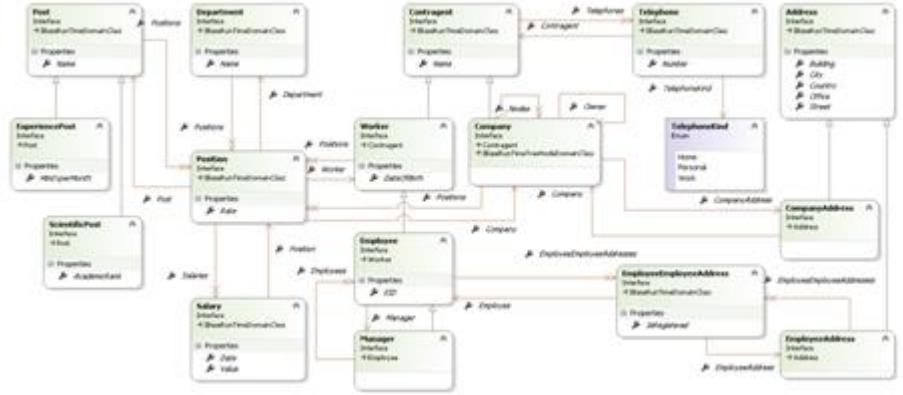


Fig. 4. Then unified model for testing object-oriented application development tools, implemented in SharpArchitect RAD Studio in C#

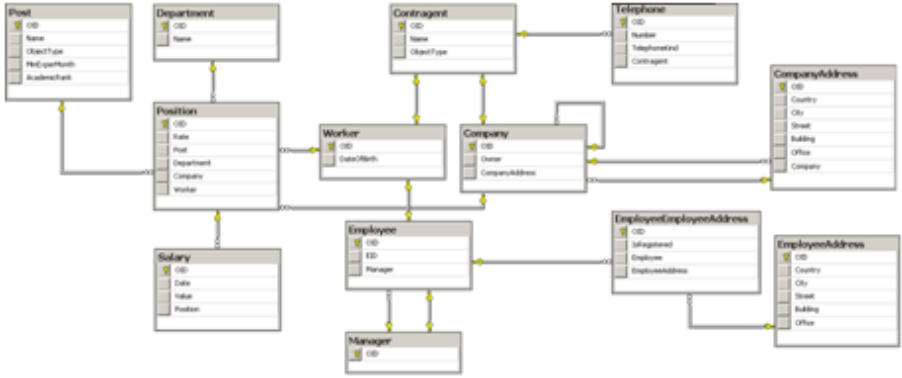


Fig. 5. Then unified model for testing object-oriented application development tools, implemented in SharpArchitect RAD Studio in C#

Figure requires is explanation. For all posts submitted by three classes of Post, ExperiencePost and ScientificPost, created one single table Post, which has all the attributes of classes. Additionally, there is a column in the table OID, representing an object identifier (primary key in a relational model). ObjectType column contains the identifier of the class whose objects are stored in the form of table rows. This value by the application to create a class of object-oriented programming language and to load the attribute values is used.

In implementing Class Table Inheritance pattern have been created for the table Contragent for abstract class and table Worker, Company, Employee, Manager for the concrete classes. Instances of classes are physically stored in multiple database tables. A copy of the Manager class is stored in all tables.

In implementing the Concrete Table Inheritance pattern is applicable for classes Address, CompanyAddress and EmployeeAddress, was created two tables: CompanyAddress and EmployeeAddress, because CompanyAddress class is abstract. All abstract class attributes stored in tables physically specific classes.

For an n-ary association Position create a separate table as well as for the binary association linking the Employee class and EmployeeAddress, for that created the table EmployeeEmployeeAddress, containing foreign keys.

Note that for the enumeration Telephone-Kind separate table is not created. An approach representations enumeration values as a bit mask and store it in the form of an integer value, where appropriate attributes are used. So the table has a column Telephone TelephoneKind, SQL-type is Integer.

After analyzing of the above it can be argued that shown in Fig. 5 implementation, created in a development environment SharpArchitect RAD Studio, fully consistent with the unified model for testing object-oriented application development tools, presented in Fig. 1.

5. Conclusion

Further development of the unified model is to test the feasibility of a variety of application development environments. In this alternative implementation is planned and using the approach presented by other authors dealing with similar scientific problems.

References

[1]. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, USA, 1994, 395 pp.

[2]. Oleynik P.P. A unified model for testing object-relational mapping tools // Object Systems – 2011: Proceedings of the Third International Theoretical and Practical Conference. Rostov-on-Don, Russia, 10-12 May, 2011. Edited by Pavel P. Oleynik. - 65-69 pp. (In Russian), http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf

[3]. Oleynik P.P. Test model for training in design of object-oriented databases // Object Systems – 2014: Proceedings of the Eighth International Theoretical and Practical Conference (Rostov-on-Don, 10-12 May, 2014) / Edited by Pavel P. Oleynik. – Russia, Rostov-on-Don: SI (b) SRSPU (NPI), 2014. – pp 86-89. (In Russian), http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf

[4]. Oleynik P.P. The Elements of Development Environment for Information Systems Based on Metamodel of Object System // Business Informatics. 2013. №4(26). – pp. 69-76. (In Russian), http://bijournal.hse.ru/data/2014/01/16/1326593606/1BI%204(26)%202013.pdf

[5]. Oleynik P.P., Computer program "The Unified Environment of Rapid Development of Corporate Information Systems SharpArchitect RAD Studio", the certificate on the state registration № 2013618212/ 04 september 2013 (In Russian).

Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений

Олейник Павел Петрович <xsl@list.ru>,
Шахтинский институт (филиал) Южно-Российского государственного политехнического университета им. М.И. Платова, Россия, Ростов-на-Дону

Аннотация. В данной статье представлена унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений. На основе имеющихся литературных источников были выделены недостатки имеющихся работ и определены следующие критерии оптимальности, которым должна соответствовать полученная модель:

1. Необходимо наличие глубоких иерархий наследования

2. Присутствие нескольких иерархий наследования
3. Наличие абстрактных классов в иерархии
4. Присутствие множественных n-арных ассоциаций
5. Наличие ассоциаций с атрибутами
6. Присутствие композиции между классами
7. Наличие рекурсивных ассоциаций
8. Наличие ассоциаций между классами, входящими в одну иерархию наследования
9. Присутствие класса-ассоциации
10. Наличие ассоциаций между классом-ассоциацией и другим классом
11. Присутствие в модели перечислений

С помощью графического унифицированного языка UML была представлена диаграмма классов унифицированной модели тестирования. В работе проверено соответствие полученной реализации выделенным критериям.

В настоящее время для реализации приложений используют объектно-ориентированные языки программирования и реляционные базы данных. Для преодоления объектно-реляционного несоответствия необходимо реализовать методы (паттерны) отображения. В статье описаны три метода, используемых для представления иерархии классов, выделены достоинства и недостатки каждого метода.

Для проверки реализуемости унифицированной модели выбрана среда разработки SharpArchitect RAD Studio, в которой спроектировано объектное приложение на языке C# и реализована структура реляционной БД. В статье представлена разработанная объектная модель в виде диаграммы классов, на которой показано наследование интерфейсов и диаграмма отношений, содержащая все таблицы и столбцы полученной БД.

В заключении даны рекомендации по направлениям дальнейшего развития работы, и определена необходимость реализовать унифицированную модель с помощью других подходов, предложенных авторами.

Ключевые слова: UML, Объектное моделирование, Проектирование информационных системы, Базы данных, Объектно-ориентированное проектирование, Методы (паттерны, шаблоны) объектно-реляционного отображения, Объектно-реляционное несоответствие

DOI: 10.15514/ISPRAS-2015-27(3)-7

Для цитирования: Олейник Павел Петрович. Унифицированная модель тестирования инструментов разработки объектно-ориентированных приложений. Труды ИСП РАН, том 27, вып. 3, 2015 г., стр. 101-114 (на английском языке). DOI: 10.15514/ISPRAS-2015-27(3)-7.

Список литературы

[1]. Гамма Э. и др. Приёмы объектно-ориентированного проектирования. Паттёрны проектирования, СПб: Питер, 2001. – 368 с.: ил. (Серия «Библиотека программиста»)

[2]. Олейник П.П. Унифицированная модель для тестирования инструментов объектно-реляционного отображения // Объектные системы - 2011: материалы III Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2011 г.) / Под общ. ред. П.П. Олейника. - Ростов-на-Дону, 2011. - С. 65-69., http://objectsystems.ru/files/Object_Systems_2011_Proceedings.pdf

[3]. Олейник П.П. Тестовая модель для обучения проектированию объектно-ориентированных баз данных // Объектные системы – 2014: материалы VIII Международной научно-практической конференции (Ростов-на-Дону, 10-12 мая 2014 г.) / Под общ. ред. П.П. Олейника. – Ростов-на-Дону: ШИ (ф) ЮРГПУ (НПИ) им. М.И. Платова, 2014. - С. 86-89., http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf

[4]. Олейник П.П. Элементы среды разработки программных комплексов на основе организации метамодели объектной системы // Бизнес-информатика. 2013. №4(26). – С. 69-76.

[5]. Олейник П.П., программа для ЭВМ "Унифицированная среда быстрой разработки корпоративных информационных систем SharpArchitect RAD Studio", свидетельство о государственной регистрации № 2013618212 от 04 сентября 2013 г.