Metamodel-Driven Design of Database Applications

Pavel P. Oleynik*

System Architect Software, Aston JSC, Rostov-on-Don, Russia

Abstract: This article describes the unified metamodel of object system which can be used for Metamodel-Driven Design (MMDD) of information system. At the beginning of the work carried out in-depth analysis of existing studies devoted to the organization different metamodels. Metamodel for representing fragments in the figures presented class diagrams of Unified Modeling Language (UML). The metamodel is realized by the author in own environment of development SharpArchitect RAD Studio. In the beginning of this article provides a general diagram which displays important associations. Next are separately shown metaclasses hierarchy which representing the entity classes of domain. There are also metaclasses to represent different types of class attributes. In addition, there are metaclasses to represent methods validation rules and visualization rules. This allows the system to implement the behavior the need for object-oriented design and allows to configure the graphical user interface. Each hierarchy is described in detail with an indication of abstract and concrete classes and associations in which they participate.

Any database system can be developed in terms of a unified metamodel of object system described in detail in this article. This work provides links to the work that is described in detail the experience of using the described metamodel. As further development of the work the author suggest the development of a formal mathematical apparatus describing applied domains and the development of UML-profile that facilitates the process of logical design of information systems in the framework of the proposed approach.

This paper describes a practical experience of the implementation of the authors of information system that automates the activities of a fast food restaurant. We studied in detail the functionality a similar system of the most popular in Russia.

Keywords: Metamodel-Driven Design, DDD, Object System Metamodel, Object-Oriented System, Information System, Database, Multiple inheritance, Design of information systems, Security of Information Systems, Permissions Model.

1. INTRODUCTION AND REVIEW OF EXISTING PUBLICATIONS

This article describes the unified metamodel of object system which can be used for metamodel-driven design (MMDD) of information system. The idea of the design based on the metamodel is not new since metamodels are used everywhere. Works [1-3] represent the metamodel of object database compliant with the ODMG standard. The SQL: 2003 standard includes metamodel that describes the object extensions of SQL [1, 3-4]. The graphical modeling language UML which is often used in the design of modern applications has the standard which governs its own metamodel described in [3, 5].

Practically all the metamodels have some disadvantages. Each author was trying to solve the problem own forces. Thus, in [3], the author has developed its own metamodel for implementing object database.

Metamodel does not exist independently of the rest of the application, and serves a certain purpose. So in [6-7] metamodel is used to facilitate the process of domain-driven design.

Metamodel is also used in the model transformations. So in [8-9] the principles of model transformations from the UML metamodel in the models of languages developed by the authors are described.

Mechanisms for evaluating the software quality based on metamodel with the introduction of various metrics are considered in [10-11]. Publications [12-13] are devoted to questions of the expansion of existing metamodels by adding new elements. In [1], the author offers his own hierarchy of atomic literal types, which can be used in any object system and built thanks to the author's experience.

2. UNIFIED OBJECT SYSTEM METAMODEL

In this article, we briefly review the metamodel used in a unified development environment for the rapid development of enterprise information systems, SharpArchitect RAD Studio [14]. In [6-7, 15-18] the full class diagram of metamodel was presented and detailed assignment of classes were described. Here we consider only the relevant parts for this article. Figure **1** shows a fragment of a unified metamodel of object system with display of the key associations that are important for further discussion.

*Address correspondence to this author at the System Architect Software, Aston JSC, Rostov-on-Don, Russia; Tel: +7 908 507 80 61;
E-mail: xsl@list.ru

E-ISSN: 2410-2938/15

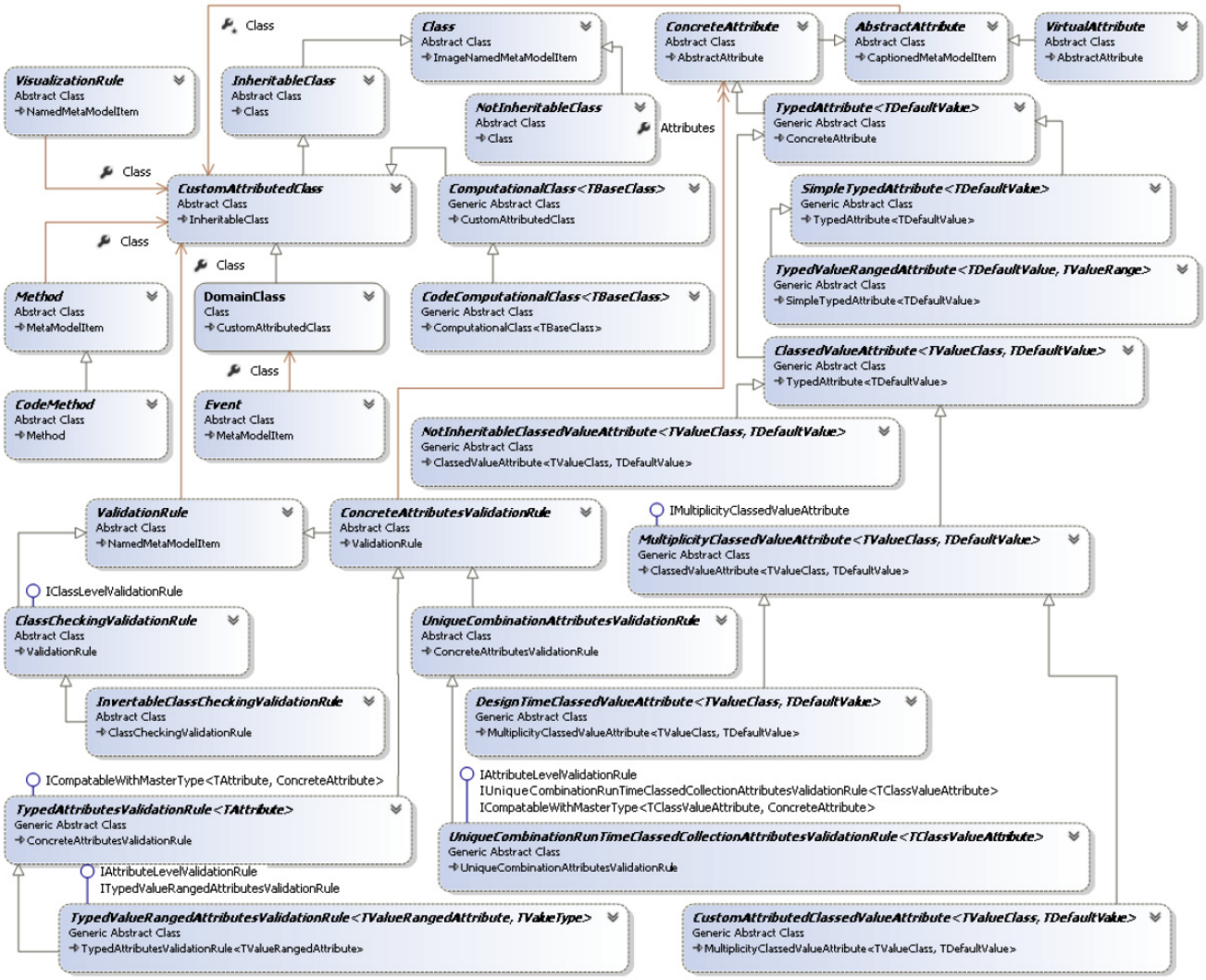

Figure 1: Fragment of a unified metamodel of object system

Consider some of the key hierarchies of metaclasses. Figure **2** is a diagram of the basic metaclasses used to represent different kinds of classes applied to describe the entity classes which are presented in the domain model.

An abstract metaclass Class is the root of the hierarchy. It has two inherited classes: 1) InheritableClass is used to represent metaclasses that can be inherited, *i.e*. support inheritance; 2) NotInheritableClass is used to represent metaclasses that can not be inherited. Metaclass Enum allows submit an enumeration or a set of values of a simple type.

Abstract base metaclass CustomAttributedClass is used to represent metaclasses that have the attributes. Metaclass DomainClass is used to represent domain classes. Instances of domain classes allow to describe the entity classes (such as Customers, Products, Sales), which objects (*e.g*. Ivanov, Bread) are stored in the database. To simplify the description instances of the domain classes will be called just domain classes (if not assumed otherwise).

Abstract metaclass ComputationalClass<TBaseClass> is the base for all calculated metaclasses ie those classes instances of which are not stored in the database and are computed at runtime (transient). For example, the turnover balance sheet is not stored directly in the database, and is calculated based on inventory, receipts and expenditures (which are the domain classes and represent the instances of the metaclass DomainClass).

Metaclass MethodParameterClass is used to represent a Parameter Class of methods. A design

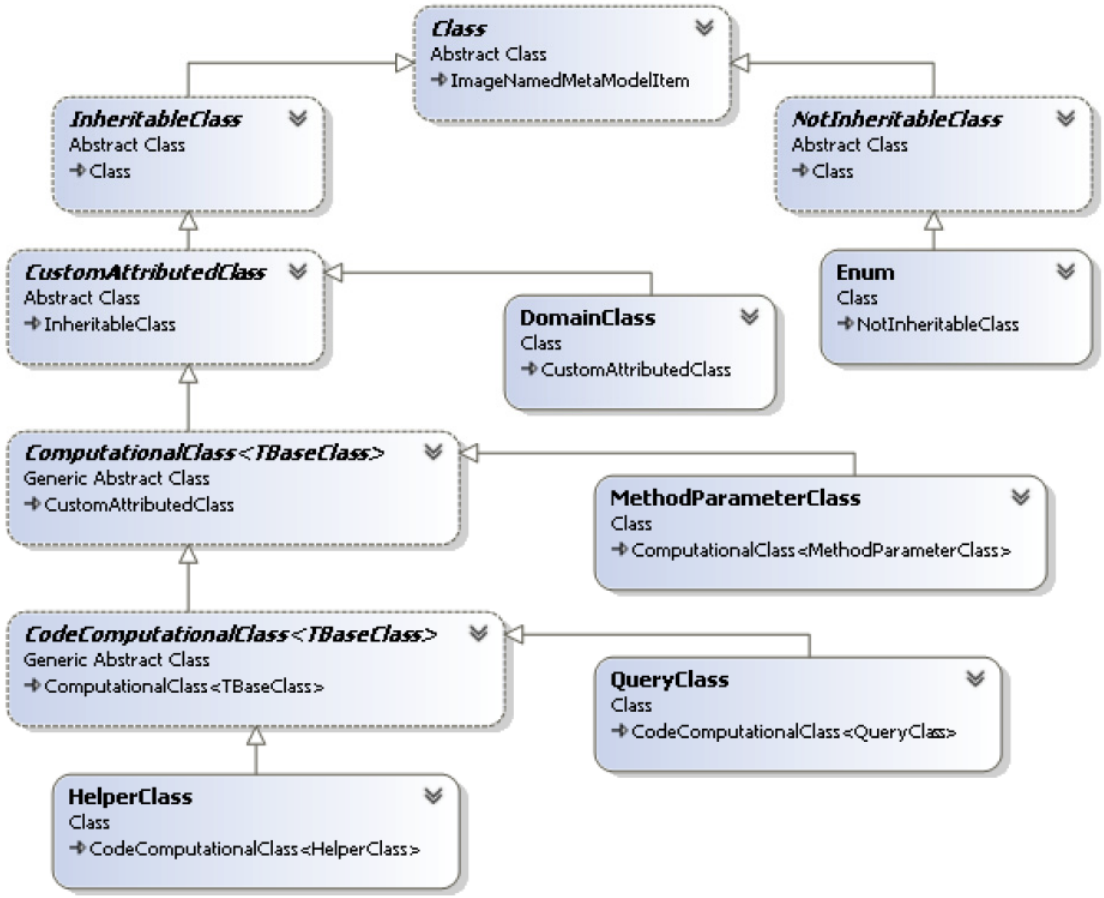

Figure 2: Basic metaclasses used to represent the entity classes of the domain model

pattern called Parameter object, the essence of which is the transfer of a set of parameters in the method as a single object (an instance of the metaclass) is implemented in SharpArchitect RAD Studio.

Abstract metaclass CodeComputational Class<TBaseClass> is the base for calculated metaclasses implemented using code in the language of C#. QueryClass is metaclass of query allowing to form the result based on queries to the database (usually based on Linq-object request but SQL-lines are also possible). HelperClass is used to represent the auxiliary metaclasses that can be displayed in the user interface and used for internal purposes in the implementation of business logic.

Now consider the metaclasses used to describe the class attributes shown in Figure **3**.

Root abstract metaclass representing an attribute is AbstractAttribute. Classes inherited from VirtualAttribute are used to represent attributes that were not created by the developer of the application domain, and were presented to the system. They are necessary for an understanding of the metamodel and simplify the software development process. SystemAttribute allows to describe the attributes that are system and are presented at the language of C#. Metaclass GeneratedAttribute is used to represent attributes automatically generated by the system. For example, if inheriting from the base tree class an attribute Node, which allows to get the child nodes and thus to form a hierarchical structure, are automatically added.

An abstract base metaclass ConcreteAttribute is used for presentation of attributes whose values can be defined by the user. Since the system is implemented in the language of C #, when saving values in the database the data types of this language are used. To describe this moment parameterized metaclass TypedAttribute<TDefaultValue> was added. TypeAttribute is used to represent the properties whose values can store a reference to the data type of the C# language.

Metaclass ClassedValueAttribute<TValueClass, TDefaultValue> is used to represent attributes whose

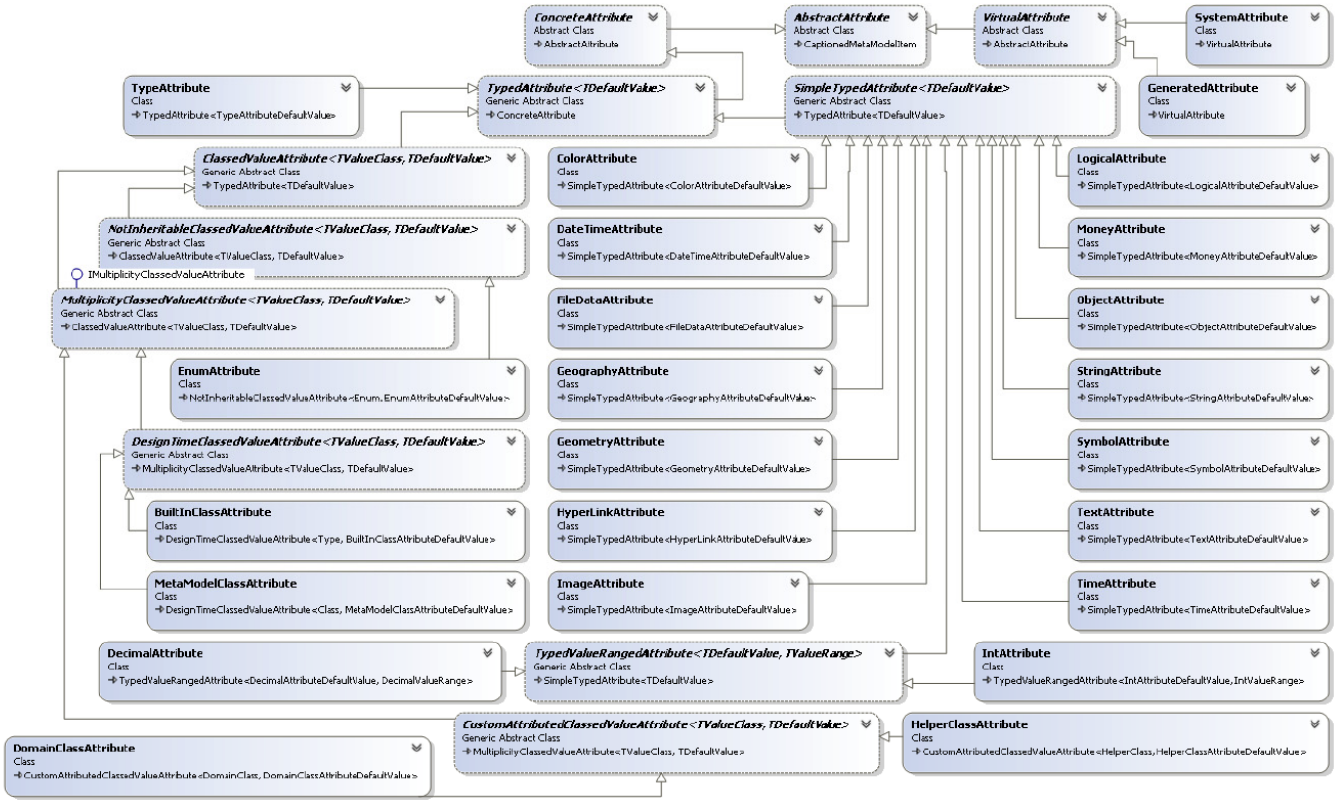

Figure 3: Basic metaclasses used to represent the attributes of classes

values are the instances of the different instances of entity classes present in the domain. Metaclass NotInheritableClassedValueAttribute<TValueClass, TDefaultValue> retains instances of non-inherited classes. For example, EnumAttribute inherited from the one described is used to store values of enumerations / sets.

Abstract metaclass MultiplicityClassedValue Attribute <TValueClass, TDefaultValue> is used to represent the values of the attributes that can store not only atomic values but also a collection of values. Metaclass DesignTimeClassedValueAttribute <TValueClass, TDefaultValue> allows saving a reference to instances of design time. So BuiltInClassAttribute is used to store objects of classes in the metamodel implementation in SharpArchitect RAD Studio. In its turn MetaModelClassAttribute saves information about the class metamodel of the application domain. Both described metaclasses allow manipulate metamodel at the moment of runtime. A similar approach is used in many modern programming languages supporting an extensive meta-information. So in C # there is a technology of reflection, which allows realizing such things.

Metaclass CustomAttributedClassedValueAttribute <TValueClass, TDefaultValue> is used to store instances of classes with attributes. The system has two child classes: 1) DomainClassAttribute saves a reference to the instance of a domain entity described in the metamodel using an instance of domain metaclass. Attribute of this type is used for the organization of the association relationships and serves to represent the relations with the object design of the domain. Metaclass HelperClassAttribute allows saving references to instances of helper classes.

Metaclass SimpleTypedAttribute <TDefaultValue> is abstract and serves the root of all the attributes to save the atomic literal value. All of this hierarchy is the result of many years of work, the premise of which and the intermediate solutions have been described in [1]. Metaclass ColorAttribute is used to store the color in the format of RGB. LogicalAttribute is used for storing Boolean values (true and false). Metaclass DateTimeAttribute is used to save the date-time values. If you want to present time only you should use TimeAttribute. There is metaclass MoneyAttribute for submission to the money attribute in the hierarchy. FileDataAttribute is used to save files of various formats. Attribute of type ObjectAttribute should be used to store any type of object. This approach is

similar to the use of type object in C#. Attributes of types GeographyAttribute and GeometryAttribute are used to save the geographical coordinates and geometric objects, respectively. Metaclasses StringAttribute and SymbolAttribute are used to represent character strings and individual characters respectively. If you want to save the text of unlimited length and with formatting you should use TextAttribute. Metaclass HyperLinkAttribute is used to represent hyperlinks to various resources. ImageAttribute is used for storing graphics (pictures, photographs and the like). Parameterized abstract metaclass TypedValueRangedAttribute <TDefaultValue, TValueRange> is used to represent atomic values, which may be within a certain range of values specified by the relevant enumeration (parameter TValueRange). Inherited metaclass IntAttribute can be used to store integer values and DecimalAttribute can be used to represent fractional values.

To implement the behavior in SharpArchitect RAD Studio different syntactic constructions and metaclasses are used. Class methods, metaclasses of which are shown in Figure **4** are the most commonly used.

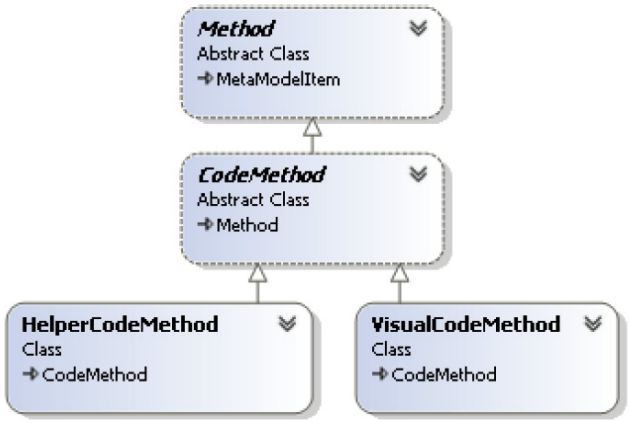

Figure 4: Basic metaclasses used to represent the class methods.

Method is the root abstract metaclass of method. Currently, the system supports only the methods implemented in the form of program code and submitted as instances of metaclasses inherited from CodeMethod. Metaclass VisualCodeMethod will create a visual method, which is displayed to the user in the form of a graphic element in the interface. HelperCodeMethod is a helper method that is used to call other methods and properties and is not involved in forming the interface.

Events are an integral part of behavior used in the development of object-oriented applications. Figure **5** shows metaclasses allowing to describe the various events of objects.

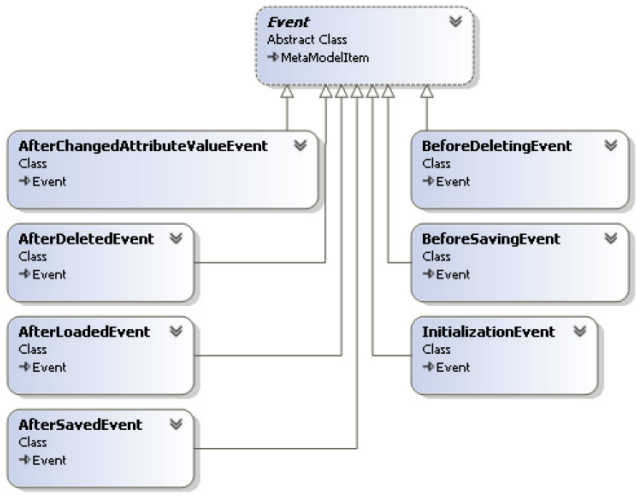

Figure 5: Basic metaclasses used to represent events.

The hierarchy is very simple. Event is the root abstract metaclass represented an event. AfterChangedAttributeValueEvent describes events of changing the attribute values and has in its composition, not only the name of the changed attribute, but also the possibility of obtaining the values before and after the change. Event AfterDeletedEvent is called after the removal of the object. Note that the object is not physically removed from the database, but only marked as deleted. This is due to the possible presence of links to other objects and the need to provide an opportunity to recover accidentally deleted object. Exactly after the installation of this label the event is called. Metaclass AfterLoadedEvent allows to create an event handler that occurs after loading the object from the database. Metaclass AfterSavedEvent describes an event that occurs after saving the object in the database. Metaclass BeforeDeletingEvent is used to represent an event that occurs before the removal of the object. If you want to perform certain actions before saving you must create an instance of the metaclass BeforeSavingEvent. The code of the new object initialization is performed in the object of type InitializationEvent.

As can be seen, SharpArchitect RAD Studio is a mature software product designed for the development of object-oriented database applications, and provides a unified metamodel enabling to describe both static and dynamic elements of the application. The IDE has been tested on different projects, described in [19-20].

Any database system can be developed in terms of a unified metamodel of object system described in detail in this article. As further development of the work the author suggest the development of a formal mathematical apparatus describing applied domains and the development of UML-profile that facilitates the process of logical design of information systems in the framework of the proposed approach.

3. USING METAMODEL-DRIVEN DESIGN FOR IMPLEMENTATION FAST FOOD RESTAURANTS INFORMATION SYSTEM

The fast food restaurant represents classical system of mass service therefore interest in its studying and formalization of subject domain arose for a long time [20-22]. The most popular information system automating activity of restaurants in Russia is the iiko system [22]. Existence of a set of introductions allows to claim that the described system is the standard, on its functionality and it is necessary to be leveled. Before own realization it is necessary to allocate the criterion of an optimality (OC) which contain functional features of future realization. The following requirements defining need were allocated:

1. To provide possibility of automation by means of uniform system as small cafe (or one restaurant), and the whole network of institutions;

2. To develop the developed graphic interface with support of touch screens;

3. To realize the operational multiuser to account orders;

4. To realize flexible architecture of the appendix with possibility of expansion in the future;

5. To provide possibility for the press of various forms of the reporting.

We will consider each criterion in more detail. Requirements of OC1 assume using of uniform IS irrespective of scales of the organization: from single restaurant to a large network. Often similar requirements are called scalability of system. Key advantage in the light of the considered task is decrease in costs of training of the staff (waiters, cashiers, managers, managing directors) and finally increases profit on activity.

Information system of restaurant of a fast food represents the system of mass service intended for processing of demands from clients. As a rule, in crowded places in rush hour the line of clients is formed and therefore it is necessary to take a number of measures for its reduction. Classical information systems, assume existence on workplaces of users personal computers with the keyboard and a mouse. At restaurants of a fast food most often monoblocks PC (often called all-in-one PC) with touch screens are used. Windows-based all-in-one PCs once earned little respect. While most of today's AIOs still lack the graphics horsepower for hard-core gaming (we'll show you one exception), the best models are far removed from the 98-pound weaklings of yore. Many AIOs use laptop parts, which minimize heat, power consumption, and the need for noisy cooling fans. If you crave more performance, pick a model that uses desktop components (the ones we've tested are still relatively quiet). Either way, everything—the CPU, memory, storage, and optical drive—is housed in the same unit as the display, so the computer's footprint equals that of a monitor. And since most all-in-ones ship with a Wi-Fi adapter as well as a wireless mouse and keyboard, the only cable they require is a power cord.

All-in-one specifications are a blend of what you'll find in conventional desktop systems and laptop PCs. The thinnest and most compact systems are almost completely built around the same power-efficient technology as laptops. Unlike with traditional desktop PCs, with an all-in-one computer, what you see is what you get—for the life of the PC. With few exceptions, you'll never be able to upgrade without chucking the entire machine, so choose accordingly. In addition to multitouch capabilities (to support Windows 8), you should consider three other key factors: display technology, display resolution, and display size. LCD panels that employ IPS (in-plane switching) or PLS (plane-line switching) technology are vastly superior to those based on TN (twisted nematic) technology. IPS and PLS displays are more expensive, and you might find them only in larger all-in-ones, but they are worth every cent.

For this reason the developed graphic interface with support of touch screens is described in requirements of OC2 necessary for the program. Thus the display resolution, as a rule, makes 1024 on 768 pixels and it is necessary to place optimum output information and elements of the graphic interface. For the purpose of reduction the cost of the equipment of the worker at a place, owners buy rather weak monoblocks with Intel Atom processors with a clock frequency of 1,66 GHz. These hardware impose restrictions on realization and a choice of a target programming language.

Business process of restaurant corresponds to system of mass service and in the simplified look assumes existence of one computer with the installed program access to which a great number of users of various categories have: waiters, managers, commodity researcher. Access is provided on the unique code stated in the proxy map. *I.e*. each user approaching the computer brings the proxy card to the reading-out element and having authorized in system carries out the demanded actions. Then it quits the system and the following user approaches. Service speed in many respects depends on the speed of login. *I.e*. it is required to realize the operational multiuser account orders that is written down above as OC3. Thus various groups of users need to provide various interface, optimum for performance of functions. For example, the waiter needs to bring and edit orders. The manager needs to look through and have opportunity to delete orders only. Thus it is necessary to provide possibility of performance of various operations under a certain role of the user. For example, the waiter for granting a discount has to invite the manager and only the last one can choose a discount and confirm it. The standard interface of the user allow to bring new commodity positions, to organize them in hierarchy that it is simplest to realize by means of such elements of management as the dropping-out lists necessary for the commodity researcher.

OC4 demands to realize flexible architecture of the appendix with possibility of expansion in the future. Progress doesn't stand still, constantly there are new devices and technologies. Therefore the development of flexible architecture will allow to keep investments in the future when serious completions of system are required. Now for the development of new software products object-oriented programming languages which main properties such as encapsulation, polymorphism, inheritance are used most often. For saving information in long-term memory the DB operated by DBMS is used. Now relational DBMS are the most popular. Because of distinctions of existence of essential distinctions in the organization and data processing in object-oriented programming languages and in relational control systems of data there is an object and relational discrepancy for which overcoming of consequences use methods (templates, patterns) object and relational display. Thus, for compliance of OC4 to one of decisions development of the client application in the OO-programming language is and as storage of information to choose relational DBMS is used. The primary goal of object-oriented development is the assurance that the system will enjoy a longer life while having far smaller maintenance costs. Because most of the processes within the system are encapsulated, the behaviors may be reused and incorporated into new behaviors. Object-oriented system tend to model the real world in a more complete fashion than do traditional methods. Objects are organized into classes of objects, and objects are associated with behaviors. The model is based on objects, rather than on data and processing.

Printing forms of the reporting is one of the integral elements of modern information system. Feature in our case that only the printer of checks (connected on USB or RS-232), using a paper 8 cm wide roll for printing is connected to a workplace of the user of IS. Therefore all created reports have to use this type of paper instead of the standard A4 format. The situation is complicated by the fact that besides the direct check (the account by request) it is necessary to print the report on cash change containing both detailed and summary information on all orders. The fact described above allowed to create requirements of OC5.

We will pass to consideration of realization of the described system. Design of the modern information systems developed in the OO-programming language is carried out with the help of creation of the chart of classes of the unified language of modeling of UML. The main goal is to become a common language for creating models of object oriented computer software. Benefits of UML can depend on many factors. In some situations it is likely to be more beneficial. I try to mention some of them. Likely to be more beneficial:

- Larger and completer is your subject, more benefit you can expect. Especially when they are relationship between different aspects
- If this SW maintenance means some extra development/extension, it could be very useful to use UML to clarify it. You can show existing system and the way it should be extended. You can of course always show the nerd.
- If your system is already modeled in UML, you can use it to locate the problem and plan further enhancements
- If both modeler and model reader know OO and have some experience in UML, they will almost always make it beneficial.
- If you want to generate some code further more

- If you need to support a system with no documentation, it could be useful to document it first (use reverse engineering to import the code and organize it in UML)

- If you plan to maintain this system for a long time and with lots of people

In Figure **1** this chart is submitted. Key feature of the OO-paradigm is possibility of the organization of hierarchy of classes by means of inheritance. The considered subject domain contains a set of reference books, such as Tables, Discounts, Contractors, Workstations, Places of storage of products, Units of measure. All these contain only one Name attribute therefore it makes sense to allocate the basic abstract class NamedObject and to inherit all reference books from it (see Figure. **6**).

The reference book of goods is hierarchical structure, breaking goods on categories and subcategories. In the program it is realized in the form of a tree and presented by the class Menu, containing the Owner attribute for saving the link to the parental knot and the calculated Nodes attribute containing knots, affiliated from this knot.

At creation of the lines of the order containing a product the client can refuse any ingredient. For the accounting of this refusal in the program the hierarchical reference book Modifier is created.

Two types of documents are provided in system by a root abstract class for which the class Document acts. The first represents the order and it is described by the class Order. For the description of the ordered dishes the set of lines which copy is described by the class OrderItem is used. The second document is CashChange, *i.e*. cash change.

We will consider compliance to the developed hierarchy to the criteria of an optimality allocated earlier. OC1 demands to provide possibility of automation by means of uniform system as small cafe (or one restaurant), and the whole network of institutions. Apparently from Figure **1** there is no binding to the certain organization, *i.e*. it was succeeded to unify system, as it was required to make.

The requirement to develop graphic interface with support of touch screens is provided in OC2. This requirement since it is the basic for end users is at the moment realized.

By means of separate structure of groups of users of and users the operational multiuser account is realized by orders. We will note that the developed hierarchy orthogonally to this requirement to realize OC3 won't make special work.

In the appendix the classical two-level architecture "client server" where as DBMS Microsoft SQL Server 2012 is used it is realized, and the client application is written on the .Net Framework platform. The first real database management program was IBM's Information Management Systems in 1968. Databases store large amounts of data. Companies use databases to store inventory, customer information, employee information, item and pricing information. Internet search engines use databases to find web pages. People use databases to store personal contact information, home inventory and even financial records. Database programs must quickly find and return this information. SQL (Structured Query Language) started in 1973 to facilitate access to databases. Based on the Sybase Program, Microsoft SQL Server 1.0 was released in May 1989. SQL Server 1.1, released in May 1990, was the first version to support Windows 3.0. The software offers several advantages to adopters.

Therefore, the system satisfies OC4 since it succeeded to realize flexible architecture of the appendix with possibility of expansion in the future.

Printing forms represent in essence selection of data during performance the multitables queries for RDBMS. After application of methods of object and relational display the set of relational tables which physically represent the classes presented in Figure **12** was received. Requirements of OC5 are as a result realized.

CONCLUSION

As can be seen, SharpArchitect RAD Studio is a mature software product designed for the development of object-oriented database applications, and provides a unified metamodel enabling to describe both static and dynamic elements of the application. The IDE has been tested on different projects, described before. Any database system can be developed in terms of a unified metamodel of object system described in detail in this article. As further development of the work the author suggest the development of a formal mathematical apparatus describing applied domains and the development of UML-profile that facilitates the process of logical design of information systems in the framework of the proposed approach.

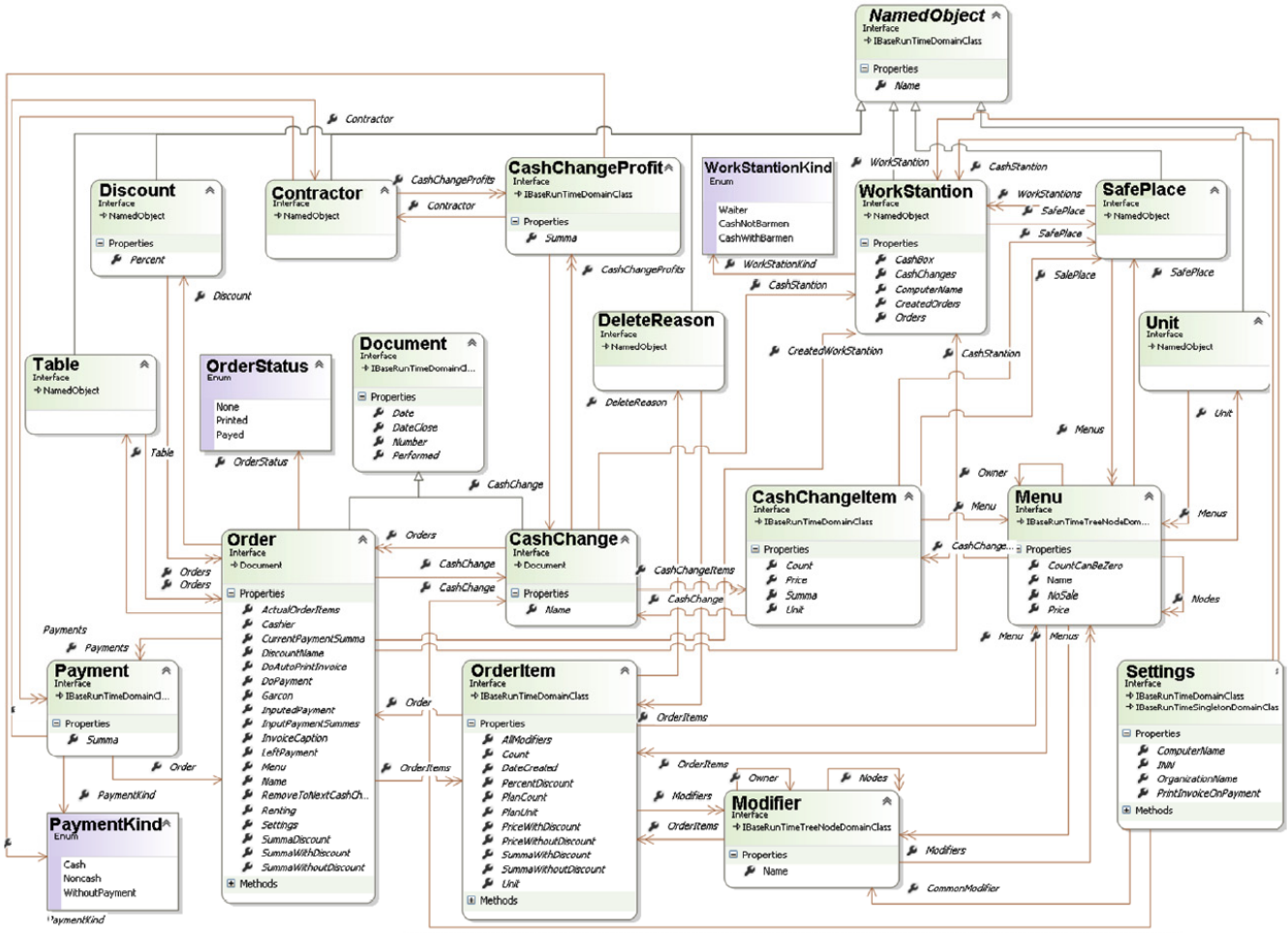

Figure 6: The UML chart of classes of information system for restaurants of a fast food.

In the future, it is planned to implement a mechanism for automatic construction of physical model using available instances of classes. The presented solution can be used in designing any database application and metamodel can be used as a basis to create a MDA tool.

From the present disclosure it can be seen that the proposed model allocation of access rights can be successfully used for applications different application domains, ie it is orthogonal thereto. Now there is the design and implementation of several applications in which safety is paramount. This will fully test the presented model and modify it in accordance with the found shortcomings.

Further development of Fast Food restaurant system is writing of various validation rules allowing to check reliability of information already at a data input stage. Since for realization of the described information system our own environment of development presented is used, similar restrictions are presented in the form of a set of logical expressions which detailed description can become the material for the following article.

REFERENCES

[1] Oleynik PP. Implementation of the Hierarchy of Atomic Literal Types in an Object System Based of RDBMS // Programming and Computer Software, 2009, Vol. 35, No.4, pp. 235-240. http://dx.doi.org/10.1134/S0361768809040070

[2] Cattell R.G., Barry D.K. The Object Data Standard:ODMG 3.0, Morgan Kaufmann Publishers, 2000, 288p.

[3] Habela P. Metamodel for Object-Oriented Database Management Systems. Ph.D. Thesis // Submitted to the Scientific Council of the Institute of Computer Science, Polish Academy of Sciences, 2002, 142p.

[4] Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation), http://wiscorp.com/sql200n.zip

[5] Iyengar S, Brodsky S. Metadata Integration using UML, MOF and XMI, November 2000, 88p.

[6] Oleynik PP. Domain-driven design the database structure in terms of metamodel of object system // Proceedings of 11th IEEE East-West Design & Test Symposium (EWDTS'2013), Institute of Electrical and Electronics Engineers (IEEE),

Rostov-on-Don, Russia, September 27 - 30, 2013, pp. 469-472.

[7] Oleynik PP. Using metamodel of object system for domain-driven design the database structure // Proceedings of 12th IEEE East-West Design & Test Symposium (EWDTS'2014), Kiev, Ukraine, September 26 - 29, 2014 http://dx.doi.org/10.1109/EWDTS.2014.7027052

[8] Soon-Kyeong K, Carrington D, Duke R. A metamodel-based transformation between UML and Object-Z // Human-Centric Computing Languages and Environments. Proceedings IEEE Symposia, 2001, 112-119 pp.

[9] Rahim LA. Mapping from OCL/UML metamodel to PVS metamodel //Information Technology, ITSim 2008. International Symposium, 2008, 1 - 8 pp.

[10] McQuillan JA, Power JFA Metamodel for the Measurement of Object-Oriented Systems: An Analysis using Alloy // Software Testing, Verification, and Validation, 2008 1st International Conference, 2008, 288 – 297 pp.

[11] Debnath N, Riesco D, Montejano G., Uzal R., Baigorria L., Dasso A., Funes A. A technique based on the OMG metamodel and OCL for the definition of object-oriented metrics applied to UML models // Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference, 2005.

[12] Debnath N, Riesco D, Montejano G, Grumelli A, Maccio A, Martellotto P. Definition of a new kind of UML stereotype based on OMG metamodel // Computer Systems and Applications, 2003. Book of Abstracts. ACS/IEEE International Conference, 14-18 July 2003.

[13] Misbhauddin M , Alshayeb M. Extending the UML Metamodel for Sequence Diagram to Enhance Model Traceability // Software Engineering Advances (ICSEA), 2010 Fifth International Conference, 22-27 Aug. 2010, 129 - 134pp.

[14] Oleynik PP, 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).

[15] Oleynik PP. Class Hierarchy of Object System Metamodel // Object Systems – 2012: Proceedings of the Sixth International Theoretical and Practical Conference. Rostov-on-Don, Russia, 10-12 May, 2012. Edited by Pavel P. Oleynik. 37-40 pp. (In Russian), http://objectsystems.ru/files/2012/Object_Systems_2012_Proceedings.pdf

[16] Oleynik PP. Class Hierarchy for Presentation Validation Rules of Object System // Object Systems – 2013: Proceedings of the Seventh International Theoretical and Practical Conference (Rostov-on-Don, 10-12 May, 2013) / Edited by Pavel P. Oleynik. - Russia, Rostov-on-Don: SI (b) SRSTU (NPI), 2013. 14-17pp. (In Russian), http://objectsystems.ru/files/2013/Object_Systems_2013_Proceedings.pdf

[17] Oleynik PP. 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

[18] Oleynik PP. Domain-driven design of the database structure in terms of object system metamodel // 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-onDon: SI (b) SRSPU (NPI), 2014. - pp. 41-46. (In Russian), http://objectsystems.ru/files/2014/Object_Systems_2014_Proceedings.pdf

[19] Oleynik PP, Kurakov YuI. The Concept Creation Service Corporate Information Systems of Economic Industrial Energy Cluster // Applied Informatics. 2014. №6. 5-23 pp. (In Russian).

[20] Oleynik PP, Yuzefova SYu, Nikolenko OI. Experience in Designing an Information System for Fast Food Restaurants // Object Systems – 2014 (Winter session): Proceedings of IX International Theoretical and Practical Conference (Rostov-on-Don, 10-12 December, 2014) / Edited by Pavel P. Oleynik. – Russia, Rostov-on-Don: SI (b) SRSPU (NPI), 2014. – pp. 12-16. (In Russian), http://objectsystems.ru/files/2014WS/Object_Systems_2014_Winter_session_Proceedings.pdf

[21] Nikolenko OI, Oleynik PP, Yuzefova SYu. Prototyping and Implementation of Graphical Order Form for the Information System of Fast Food Restaurants // Object Systems – 2015: Proceedings of X International Theoretical and Practical Conference (Rostov-on-Don, 10-12 May, 2015) / Edited by Pavel P. Oleynik. – Russia, Rostov-on-Don: SI (b) SRSPU (NPI), 2015. (In Russian), http://objectsystems.ru/files/2015/Object_Systems_2015_Proceedings.pdf

[22] Automation of restaurant iiko, (In Russian), http://iiko.ru/

Received on 10-06-2015 Accepted on 19-06-2015 Published on 06-07-2015

http://dx.doi.org/10.15379/2410-2938.2015.02.01.03