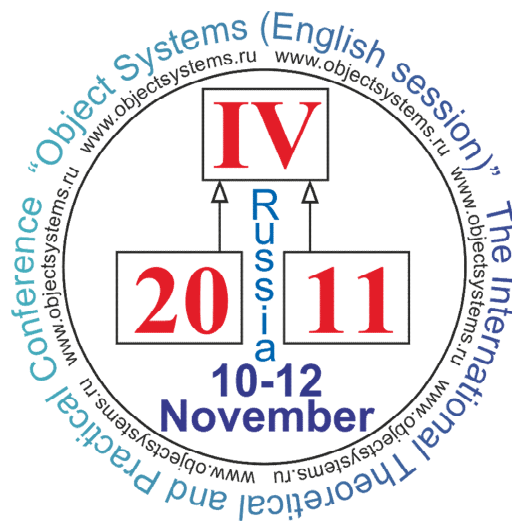




**Shakhty Institute (Branch) of
South Russian State Technical University
(Novocherkassk Polytechnic Institute)**

**South Russian State University of
Economics and Service**

**Alexander Technological Educational Institute of
Thessaloniki**



Object Systems – 2011 (English session)

**Proceedings of the Fourth International
Theoretical and Practical Conference**

Edited by Pavel P. Oleynik

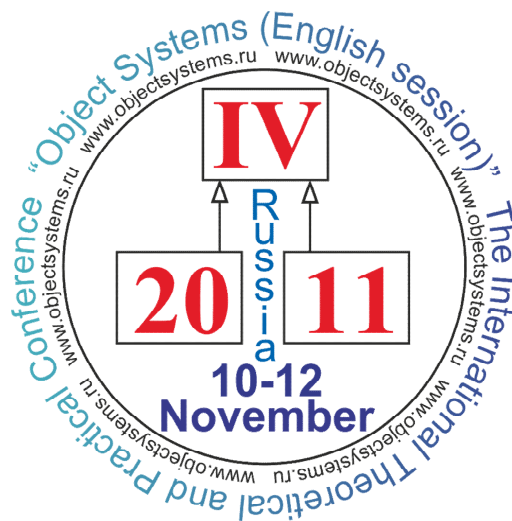
**Rostov-on-Don, Russia
10-12 November 2011**



**Shakhty Institute (Branch) of
South Russian State Technical University
(Novocherkassk Polytechnic Institute)**

**South Russian State University of
Economics and Service**

**Alexander Technological Educational Institute of
Thessaloniki**



Object Systems – 2011 (English session)

**Proceedings of the Fourth International
Theoretical and Practical Conference**

Edited by Pavel P. Oleynik

**Rostov-on-Don, Russia
10-12 November 2011**

Object Systems – 2011 (English session): Proceedings of the Fourth International Theoretical and Practical Conference. Rostov-on-Don, Russia, 10-12 November, 2011. Edited by Pavel P. Oleynik

The proceedings consist of papers which cover the topics of design work, implementation and maintenance of object systems, considering a broad range of problems. These papers were accepted to publish on the basis of the organization and program committee member's reviews.

Conference Organizers



**Shakhty Institute (Branch) of
South Russian State Technical University
(Novocheerkassk Polytechnic Institute)**



**South Russian State University of
Economics and Service**



**Alexander Technological
Educational Institute of Thessaloniki**

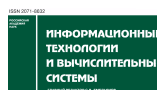
Information Partners



ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ
Ежемесячный теоретический и прикладной научно-технический журнал
(с приложением)



Journal
"AUTOMATION IN INDUSTRY"



Journal **"Information technologies
and computer systems"**

Community of System Analysts

Journal **"Information Technologies"** with monthly supplement

Conference Committee

1. Nikolay N. Prokopenko, Doctor of Sciences, Professor, Rector, South Russian State University of Economics and Service, Russia, Shakhty (**Chair of the conference**)
2. Pavel P. Oleynik, Ph.D., System Architect, Aston, Russia, Rostov-on-Don (**Co-chair of the conference**)
3. Euclid Keramopoulos, Ph.D., Lecturer, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki (**Co-chair of the conference**)
4. Vladimir I. Bozhich, Doctor of Sciences, Professor, Department of Information Systems and Radio Engineering, South Russian State University of Economics and Service, Russia, Shakhty
5. Vladimir I. Sidelnikov, Doctor of Sciences, Professor, Head of the chair of Economics and Applied Mathematics, Southern Federal University, Pedagogical Institute, Russia, Rostov-on-Don
6. Elvira Yu. Cherkesova, Doctor of Sciences, Professor, Head of the chair of Information Technologies and Management, Shakhty Institute (Branch) of South Russian State Technical University (Novocheerkassk Polytechnic Institute), Russia, Shakhty
7. Anatoly A. Mikhailov, Doctor of Sciences, Professor, Department of Automated Control Systems, South Russian State Technical University (Novocheerkassk Polytechnic Institute), Russia, Novocheerkassk
8. Vladislav G. Krawczyk, Ph.D., Associate Professor of Energy and Life Safety, South Russian State University of Economics and Service, Russia, Shakhty
9. Ignatios Deligiannis, Ph.D., Associate Professor, Head of Department, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki
10. Panagiotis Sfetos, Ph.D, Assistant Professor, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki
11. Apostolis Ambatzoglou, Researcher, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki

International Program Committee

1. Piotr Habela, Ph.D., Assistant Professor, Polish-Japanese Institute of Information Technology, Poland, Warsaw
2. Erki Eessaar, Ph.D., Associate Professor, Acting Head of the Chair, Faculty of Information Technologies: Department of Informatics, Tallinn University of Technology, Estonia, Tallinn
3. German Viscuso, MSc in Computer Science, Marketing, Versant Corp., Spain, Madrid
4. Sergei D. Kuznetsov, Doctor of Sciences, Professor, Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Chief Scientist, Institute for System Programming of Russian Academy of Science, ACM, ACM SIGMOD and IEEE Computer Society Member, Russia, Moscow
5. Anatoly A. Shalyto, Doctor of Sciences, Professor, Awarded by Russian State Government for achievements in education, Head of the chair of Programming Technologies, Saint-Petersburg State University of Information Technologies, Mechanics and Optics, Russia, Saint-Petersburg
6. Alexander Yu. Kiryutenko, Ph.D., CIO, Aston, Russia, Rostov-on-Don
7. Edward G. Galiaskarov, Ph.D., Associated Professor, Ivanovo State University of Chemistry and Technology, Russia, Ivanovo
8. Alexander V. Chekiris, Head of department of engineering design and normative-reference information, NII EVMservice, Belarus, Minsk
9. Irina Yu. Veklenko, System Analyst, Russia, Chernogolovka
10. Victor V. Malysheko, Ph.D., Associated Professor of Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia, Moscow
11. Ludmila Yu. Zhilyakova, Ph.D., Senior Researcher, V.A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Russia, Moscow
12. Karina J. Shakhgelyan, Doctor of Sciences, Head of IT-department, Vladivostok State University of Economics, Russia, Vladivostok
13. Pavel V. Dobryak, Ph.D., Associated Professor, Ural Federal University, Russia, Ekaterinburg
14. Alexander S. Baikin, Lead Systems Analyst, Avtomir, Russia, Moscow
15. Alexey I. Averin, Systems Analyst, Aston, Russia, Rostov-on-Don
16. Valery V. Laptev, Ph.D., Associated Professor of Automated Information Processing and Control System, Astrakhan State Technical University, Russia, Astrakhan
17. Ilya E. Ermakov, Lecturer, Polycarpov Institute of Technology at the State University – Educational-Scientific-Industrial Center, CTO, NPO "Tesla", Russia, Orel
18. Denis Yu. Ivanov, Consultant, IT Consulting, Russia, Saint-Petersburg

Reviewers

Vladimir I. Bozhich, Anatoly A. Mikhailov, Irina Yu. Veklenko, Victor V. Malysheko, Pavel V. Dobryak

Editors

Pavel P. Oleynik (**chief editor**), Valery V. Laptev, Edward G. Galiaskarov, Irina Yu. Veklenko, Ilya E. Ermakov

(c) **Object Systems – 2011 (English session)**, The Fourth International Theoretical and Practical Conference, 2011

(c) **Authors**, 2011

ISBN 978-5-9902226-7-0

Contents

Gainullin R.F. Semantic analysis of class UML-diagrams using the RV-grammars	6
Krylov A.Yu., Galiaskarov E.G. The basic organization of an ontology driven retrieval system	11
Zinchenko D.S., Galiaskarov E.G. Objects behavior modeling using fuzzy logic	17
Sirotsky A.A., Zvereva A.O. The distributed systems: modern representations and development tendencies	21
Keramopoulos E., Zounaropoulos M., Kourouleas G. A comparison study of object-oriented database management systems	26
Ermakov I.E. Scaling from OOP to SOA and Enterprise architectures. Complexity control and evolution support in software design	30
Astasheva E.I., Razinkin C.A. Modeling digital library management system based on the methods of queuing theory	33
Arvanitou E.-M., Ampatzoglou A., Deligiannis I. Teaching object-oriented analysis and design through computer games	36
Oleynik P.P. The implementation of the framework to create and manage exchange trading robots	42
Notes	46

SEMANTIC ANALYSIS OF CLASS UML-DIAGRAMS USING THE RV-GRAMMARS

Rinat F. Gainullin, Postgraduate student, Ulyanovsk state technical university, Russia, Ulyanovsk, r.gainullin@gmail.com

Abstract. This article focuses on the system of UML-diagrams analyses based on the RV-grammars. Also described an algorithm for the parser behavior in case of error detection. The analyzer must reduce the number of costly errors that allowed the design stage.

Introduction

UML provides support throughout the lifecycle of IP, and provide such a set of graphical tools - charts.

During creating a conceptual model to describe the business activities of the models of business use cases and activity diagrams to describe business objects - business object model and sequence diagrams.

During creating a logical model of IP description of system requirements is given in the form of models and descriptions of system use cases, and the preliminary design is done using class diagrams, sequence diagrams and state diagrams.

During creating a physical model of the detailed design is done using class diagrams, component diagrams, deployment diagrams.

Below is a diagram of the relationships between UML-diagrams. Indices arrow can be interpreted as the ratio "is the source of input for ..." (Eg, use case diagram is a data source for the activity diagrams and sequence).

Market Analysis design systems shows that many of them have no opportunity to verify the correctness of the diagrams. Those systems that have validation controls are typically limited syntactic correctness of the diagram. There is a separate group of systems that do not allow us to construct diagrams incorrect, but if they get a chart with errors, then it is no longer detected.

It is proposed syntax-directed diagram analyzer based languages automatic graphic RV-grammars, which can be easily integrated into systems used in practice of design.

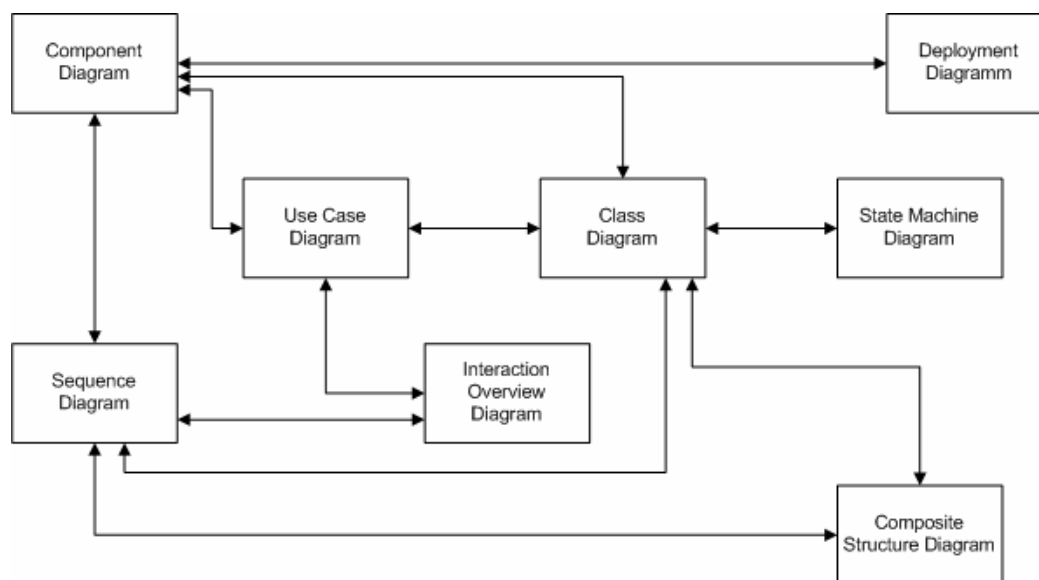


Fig. 1 - UML diagram relations

1. RV-Grammar

RV - grammar of a language L (G) is an ordered five non-empty sets $G = (V, \Sigma, \overline{\Sigma}, R, r_0)$ [1], where missing a tilde, the indices are larger:

- $V = \{v_e, e = \overline{1, L}\}$ - auxiliary alphabet;
- $\Sigma = \{a_t, t = \overline{1, T}\}$ - terminal alphabet graphic language;

- $\overline{\Sigma} = \{a_t, t = \overline{1, T}\}$ - kvaziterminalny alphabet;
- $R = \{r_i, i = \overline{1, T}\}$ - schema grammar G;
- $r_0 \in R$ - RV-axiom grammar.

Products $P_{ij} \in r_i$ is given by $P_{ij}: \tilde{a}_i \xrightarrow{\Omega_\mu[W_v(\gamma_1, \dots, \gamma_n)]} r_m$ Where

- $W_v(\gamma_1, \dots, \gamma_n)$ - n - ary relation that defines the type of operations on the internal memory as a function of $v \in \{0, 1, 2, 3\}$;
- Ω_μ - operator modifications, in some way changes the type of operation of the memory, and $\mu \in \{0, 1, 2\}$;
- $r_m \in R$ - the name of a complex product - a successor.

RV-grammar is automate-based grammar. It operates with inner memory to control correctness of diagrams. Finite state machine is used to choose rule by memory changing. A number of different structures are used as memory: stacks, lists, queries, etc. and combination of them. These conditions make possible to construct grammar with linear dependence from element count. All operation take limited time, because reading and writing data is constant time operation. In each tack of diagram analysis operate with only one element.

RV-developed sub-class of grammars - RVUML-grammar for the five kinds of diagrams: Activity, Sequence, Use Case, Class and package [3].

2. Grammar example

As an example, consider the grammar Class diagram. Table 1 is shown the terminal alphabet of the language and kvaziterminalny UML diagrams Class.

Table 1. Terminal and kvaziterminal alphabets Class diagrams


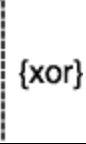
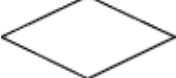



Term		Kvaziterm				
<table border="1"> <tr><td>Class name</td></tr> <tr><td>Class attribute</td></tr> <tr><td>Class operation</td></tr> <tr><td>Condition</td></tr> </table>	Class name	Class attribute	Class operation	Condition		C
Class name						
Class attribute						
Class operation						
Condition						
		Link _C				
		Link _n				
		A _{XOR}				
		A _{XOR}				
		A _{nm}				
		Link _G				
		Link _A				
		Link _K				

Table 2 is shown the grammar RVUML-Class diagrams. After the parsing is end it's necessary to control the operation:

$$* = W_2(i^{t(1)}, i^{t(2)})/W_3(i^{t(1)} \diamond \&\& i^{t(2)} \diamond 0), \\ W_2(i^{t(3)}, i^{t(4)})/W_3(i^{t(3)} \diamond \&\& i^{t(4)} \diamond 0),$$

$$\begin{aligned}
&W_2(i^{t(5)}, i^{t(6)})/W_3(i^{t(5)}) \triangleleft \&\& i^{t(6)} \triangleleft 0), \\
&W_2(i^{t(7)}, i^{t(8)})/W_3(i^{t(7)}) \triangleleft \&\& i^{t(8)} \triangleleft 0), \\
&W_2(i^{t(9)}, i^{t(10)})/W_3(i^{t(9)}) \triangleleft \&\& i^{t(10)} \triangleleft 0).
\end{aligned}$$

The result should remain blank list.

Table 2. RVUML-class diagrams grammar

Complex	Kvazitherm	The complex - the successor	RV - the ratio of
r0	Link _C	r1	$W_1(i^{t(1)})/W_3(i^{t(1)}) == 0 \parallel i^{t(2)} == 0$
	Link _n	r2	$W_1(i^{t(3)})/W_3(i^{t(3)}) == 0 \parallel i^{t(4)} == 0$
	Link _G	r3	$W_1(i^{t(5)})/W_3(i^{t(5)}) == 0 \parallel i^{t(6)} == 0$
	Link _A	r4	$W_1(i^{t(7)})/W_3(i^{t(7)}) == 0 \parallel i^{t(8)} == 0$
	Link _K	r5	$W_1(i^{t(9)})/W_3(i^{t(9)}) == 0 \parallel i^{t(10)} == 0$
	A _{XOR}	r6	
	A _{XOR}	r7	
r1	C	r0	$W_1(i^{t(1)})/W_3(i^{t(1)}) == 0 \parallel i^{t(2)} == 0$
r2	C	r0	$W_1(i^{t(3)})/W_3(i^{t(3)}) == 0 \parallel i^{t(4)} == 0$
r3	C	r0	$W_1(i^{t(5)})/W_3(i^{t(5)}) == 0 \parallel i^{t(6)} == 0$
r4	C	r0	$W_1(i^{t(7)})/W_3(i^{t(7)}) == 0 \parallel i^{t(8)} == 0$
r5	C	r0	$W_1(i^{t(9)})/W_3(i^{t(9)}) == 0 \parallel i^{t(10)} == 0$
r6	Link _C	r1	$W_1(t^{1m})$
r7	C	r0	$W_2(t^{1m})$
r8	C	r0	

3. Semantic error in UML class diagram

3.1. Inheritance

Inheritance is a property of a system that allows you to define a new class based on an existing, partly or wholly borrowed features. Class from which you are inheriting is called base, or parent superclass. A new class is called a descendant, or heir of the derived class.

Inheritance can be used to express the generalization or the specialization. The basic idea is that you define a new type by extending or modifying an existing one. In other words, the derived class has all data and methods of the base class, and has the new data and methods, and perhaps modify some of the existing methods. Different OO languages use different keywords to describe the mechanism (derivation, inheritance, sub-classing), for the class from which you inherit (the base class, parent class, superclass) and a new class (derived class, child class, subclass).

On a Natural language, this mechanism can be described as follows. There is a description of the objects of a certain type (such as a car (Car)), and there is a set of methods and properties in this description (eg, make (Car.start ()) and (stoped Car.stop ())). Such description contains only generic steps that can be performed with the car. Each specific type of car has its own specific sequence of actions to implement the start and stop.

For example, let's create a description of the VAZ. In a specific implementation should be described in the following sequence for the method VAZ.start ():

- insert key;
- turn the key all the way;
- release the key.

In other cars, this sequence may be different. For example, take the car with electronic ignition (let's Audi). Then the sequence of actions can be as follows:

- insert key;
- press the start button the engine;
- release the button.

It follows that each heir clarifies the use of his ancestor. So true is to create an heir which hasn't complete all declared methods. This type of error can be found in the class diagram by checking the list of methods of ancestor and descendant of the list of methods.

3.2. Encapsulation

Encapsulation is a property of a system that allows you to combine data and methods that work with them in the classroom and hide implementation details from the user.

For the traditional OO languages it is the presence of three access specifiers that indicate different levels of encapsulation of class: public, protected, and private. Public means to be visible to any other class, protected means to be visible by derived classes (descendants), private means: the lack of visibility from the outside. There are, however, differences in details. (For some languages, there are additional modifiers, but they do not reflect on class diagrams)

In the class diagram encapsulation is usually a mistake to the wrong position methods and properties. In practice, the PLO decided to do all of the properties as private. Also, if the design is rarely interested in the internal structure of classes and, consequently, private methods. We can therefore say that the image of public properties and private methods, is a mistake in the design.

3.3. Polymorphism

Polymorphism is a feature of the system to use objects with the same interface without any information about the type and the internal structure of the object.

Providing polymorphic behavior of objects makes it necessary to bind methods to be called the program (that is, to determine which specific method is called) not at compile time, and in the process of delivering, at a cost of extra time. So real dynamic binding is required for no more than 20% of the calls, but some of the OOP languages use it consistently.

Use of a polymorphism is strongly associated with the concept of inheritance, so most of the examples is the use of a polymorphism in the process of succession.

As mentioned above only specifies the use of a descendant of an ancestor. It follows the following rule - "If you inherit you can't lower the level of access to methods / properties." The order of access levels from high to low following: public, protected, private. This rule ensures that every object inherits from an interface will always be available publicly declared methods.

Also there is another rule that imposes a limit on the return type of a method: "When you inherit can't change the return type in the offspring." This rule ensures that when referring to a specific object, we obtain a certain type of error does not happen matching types.

These two rules can be monitored in the class diagram in the design phase.

4. Error recovery

The algorithm is based neutralization operations laid the stack, which stores the possible successor to the complexes formed during the synthesis of grammar [2].

The analyzer works in both interpretation and translation. In the first case is determined by the first error found (its location and type), and the analysis stops in the second - all possible errors.

Errors are divided into the following types:

1. error use of an entity, not owned by the language;
2. error using of graphical primitives together that belong to the language;
3. error use graphic primitives associated with their semantic meaning.

The first type of error is typical for novice designers and determined in the process of constructing the diagram.

The second type of error associated with the analysis of input and output of graphic elements, and elements can be related directly or be at "some distance". For example, a chart of activity for each element of a conditional branch should be placed, respectively, one and only one element of the merger. This type of error is found after a complete analysis of the chart.

The third type of error is determined by the type of the chart. The figure shows the situation when a merge operation in the chart of activity precedes the operations branch. This type of error can be detected only by introducing additional rules when checking items.

Conclusion

The analyzer is implemented for the editors of Visio and Dia, as well as for the system design of question-answer WIQA [4,5]. RV-grammar extends to use semantic information in diagrams.

References

1. Sharov, Afanasev A. Syntactically - implementation-oriented graphical language based on automatic graphical grammars // Programming. - 2005 .- № 6 .- 56-66 pp.
2. Sharov, Afanasev A/ Neutralization of syntax errors in the graphic language // Programming, 2008. - № 1. - 61-67 pp.
3. Afanasyev A., Gainullin R. Analysis graphic design workflow specifications on example of UML / Vestnik, 2010 .- № 4 - 42-46 p.
4. Sosnin P. Conceptual modeling of computerized systems. - Ulyanovsk: UISTU, 2008.
5. Gainullin R. Development UML-diagrams analyzer //Informatics and Computer Engineering - 166-170 pp.

THE BASIC ORGANIZATION OF AN ONTOLOGY DRIVEN RETRIEVAL SYSTEM

Alexander Yu. Krylov, 4th year student, Ivanovo State University of Chemical and Tecnology, Russia, Ivanovo, Krylov_Al@mail.ru

Edward G. Galiaskarov, Associate Professor, Ivanovo State University of Chemistry and Technology, Russia, Ivanovo, galiaskarov@isuct.ru

Introduction

At present times the information searching is very important for World Wide Web, because of the fact that Internet content increases every day and gets a lot of new facts and knowledge. According to “newsru.com” in May of 2009 all servers that are connected to Internet contained 487 exabytes of data [1]. Thus, the size of Internet data had reached a value close to 500 exabytes (that is equal 500 billion Gigabytes). A probability of finding necessary information increased very significantly, but a complexity of this task, with an increase of a number of queries and their complexity taken into account, still remains very high. It is connected with the fact that a simple context search takes a lot of time being rather ineffective yet. Modern search engines partly solve this issue using different specific information search algorithms. In addition to that such systems are intended for general search, which isn't dependent on any domain or a search theme. A use of such systems result in a great number of the found documents, that belongs to absolutely different domains and are often quite irrelevant to a user's search query or needs. The most advanced search engines surely provide extended search opportunities, however it doesn't solve the described above issue completely. Such thoughts bring us to the conclusion that general purpose search engines are able to solve the issue of an information search in a definite theme just partly.

Nowadays one of the ways to solving the problem is a so-called “intellectual” search, that allows to “comprehend” a submitted by a system user's query and find the documents corresponding to the sense of a query but not just containing the words which make up the expression of the query. It raises significantly a probability to find the information a user is interested in and to reduce a labour-intensiveness of the process at the same time.

In this article we would like to present a version of an intellectual search organization in a data array, based on a use of a domain orientied ontology that allow to “comprehend”, specify and supplement a user's query in order to define the most relevant required information.

There are a great number of philosophical and technical definitions of term “ontology”. We will use the following meaning: it is a computer resource, corresponding to a certain world view concerning a definite sphere of documents. At a formal level “ontology” is a system that consists of a notion set and a set of statements about these notions, on a basis of which it is possible to make up classes, objects, relationships, functions and terms [2]. In other words, ontology is a certain conceptual domain model, which contains all the classes of objects, their relationships and rules of the domain.

The next notion that we will use is a “search query” and it corresponds to source information for an implementation of a search by a system. Such information can be represented in quite different formats (it all depends on a search system arrangement and a target of a search) – whether text documents – then a search query correspond to a text, - or graphical files – where a search query is an image or a set of it's characteristics, - musical and so on.

Let's introduce such notions as “an upper level search” and “a lower level search”:

1. The upper level search – is a logical component of a system, which implements the use of a conceptual scheme of a domain. In this case it is a refinement and conversion of a user's search query with the help of ontology, i.e. it's “understanding/comprehension” by the system in the bounds of the used domain for a further context search;
2. The lower level search – is a logical component of a system, which implements an interaction of a search system with a physical data base of documents for the purpose of realization of an information search and sampling. The lower level search implements by

means of the use of some of already existing search engines for a context search in documents in a data base.

The use of such a search division according to levels is caused by a logical division of a conceptual domain scheme (ontology) and a base of documents.

The use of ontology makes it possible to reveal in its way a sense of a query, semantic relations between the terms it contains. That is why one of the main goals in development of such a search system is the most precise (with the help of ontology) description of a domain, represented by a database of documents by that a search is implemented. It is the mode that will allow raising the effectiveness of the use of ontology as much as possible for the purpose of solving a problem of information search in the bounds of a specified domain.

Technically this problem can be solved by means of a special software, that allows analyzing the text of a document and extract the most frequently used and important words out of it, that conforms to a theme described in the document.

1. The functioning of the system under development

Having assigned the main stages of a search passage in a search engine it is possible to present the scheme of the system functioning (Fig. 1), that solves the problem of an information search with the help of ontology.

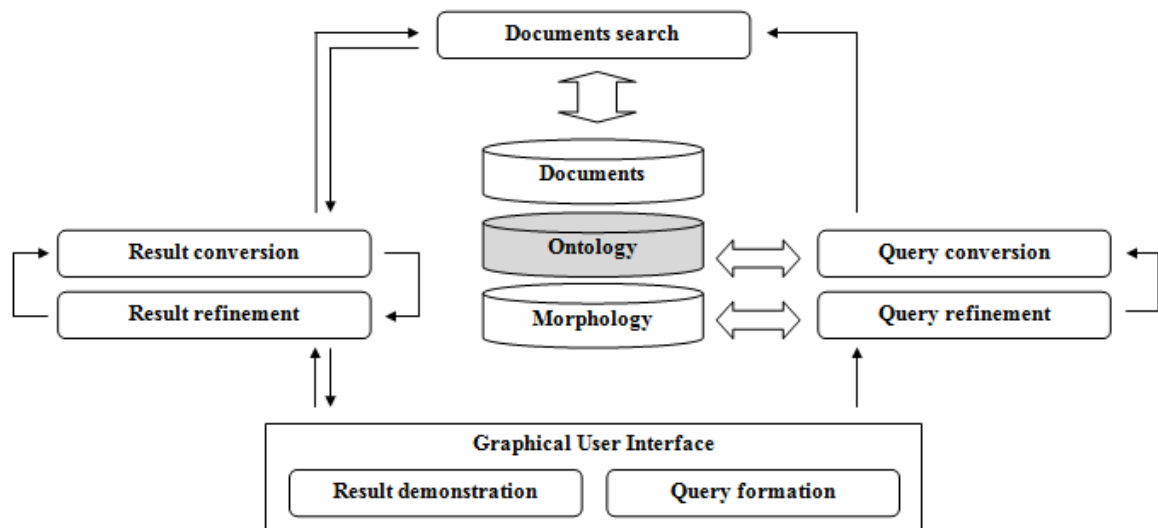


Fig. 1 - The scheme of a search query passage in the system

The concept of functioning of such a system is the following:

1. A formed user's search query is divided into its component words and run through a morphology subsystem that allows getting a set of word forms of all the elements of the query for the following conversion of the query. This phase is very important as ontology that is used in the system contains terms and definitions of a domain in a definite form to that at least one form of the query element should correspond so that it was possible to find this element in a conceptual scheme;
2. At this stage a "comprehension" of the query is implemented – the whole gained set of the word forms and the initial text of the query is run through ontology that allows defining the connections between terms – elements of the query. These can be connections like "part of hierarchy", "synonym", "association" and so forth. Using the gained connections, the user's query is supplemented with the compliant elements, thereby making up a search image of the query that is used for the search in documents array after that. This stage implements "the upper level of the search";
3. The search in documents array is implemented with the help of a special search engine (already embodied software) that in its turn implements "the lower level search"- a context search in documents database.

4. The gained results may be ranges, filtered, grouped and so on – everything that will allow a user to choose only the materials he/she is interested in among all that will have been found.

Such a query passage in the system may be iterative, that allows a user not only to make operations available at the last stage of the query passage in the system but also to modify the text of an initial query and to use it for search again, in the whole base or in the found results. Such an approach to the organization of an information search allows a backtracking of a query in the system, refining it while getting the results of a search and sampling, and also correcting it at each stage of the query passage in the system, where the current stage allows doing that. Therefore with the use of such an approach in the process of a query passage in the system there forms a search image of a required document that is the most relevant to the concrete user's request [3].

2. The architecture of the system under development

While implementing the described above engine of information search it is possible to present a system that supports the current algorithm functioning in the form of a package of components each of those implement separate operations that conform to it according to the scheme of a query passage within the system. It is necessary for the developed system to provide the following:

- a capability to use various approaches in the organization of domain schemes (a domain ontology, diagram of classes, glossary and so on) depending on a search goal;
- a dependence of a search theme only on a content of a database of documents and a domain scheme what makes other components of the system just a technological frame that can be used with another conceptual scheme and base of documents;
- a search process discreteness that provides a capability of backtracking of a search process at each of the stages of a search query passage within the system.

3. Graphical user interface

It gives a user of the system an interface for generation of a search query, query refinement at different stages of a passage within the system and an implementation of various operations with the search results. It implements in a form of html-pages in a user's browser generated by Java Server Pages.

4. Query processing subsystem

4.1. Query refinement module

It implements the first stage of a query passage within the system. Using a morphology module that locates in data subsystem and supports a compliant interface the query refinement module refines the query source text adding to it various word forms of the elements of the query. To get the results of the query refinement this module provides with a compliant interface – "IRefinedQuery".

4.2. Query conversion module

It implements the second stage of a query passage within the system. Using an ontology building module, that locates in data subsystem and provides a compliant interface the query conversion module converts a user's query proceeding from the gained connections of the query elements. To get a converted query this module provides with a compliant interface – "IConvertedQuery".

4.3. Documents list conversion module

It implements the forth stage of a query passage within the system. According to the specified descriptors of documents (for example, search results) it implements operations that provide the compliant documents in the form, convenient to a user's perception – these are: adding of metainformation and elements of the document text, setting of the rules sorting, filter and so on. For this purpose this module provides with a compliant interface – "IProcessedDocsList". As a user has an opportunity to implement various operations with search results, for this purpose there is supported an interface "IProcessedDocsList" – that is used for a setting of parameters of compliant operations at the specified documents (as the search results) and getting the results.

5. Search engine module

It implements the third stage of a query passage within the system. At the same time this module is the already developed software that implements indexing and search in documents in a data-base. “Sphinx” search system is used in this case. To perform a search the system connects to the data-base that stores documents and metainformation compliant to them. The search results are represented in the form of descriptors of the found documents and content compliant to the search. For the purpose of search implementation this module supports an interface, made in the form of web-service, an access to which is performed to a definite port.

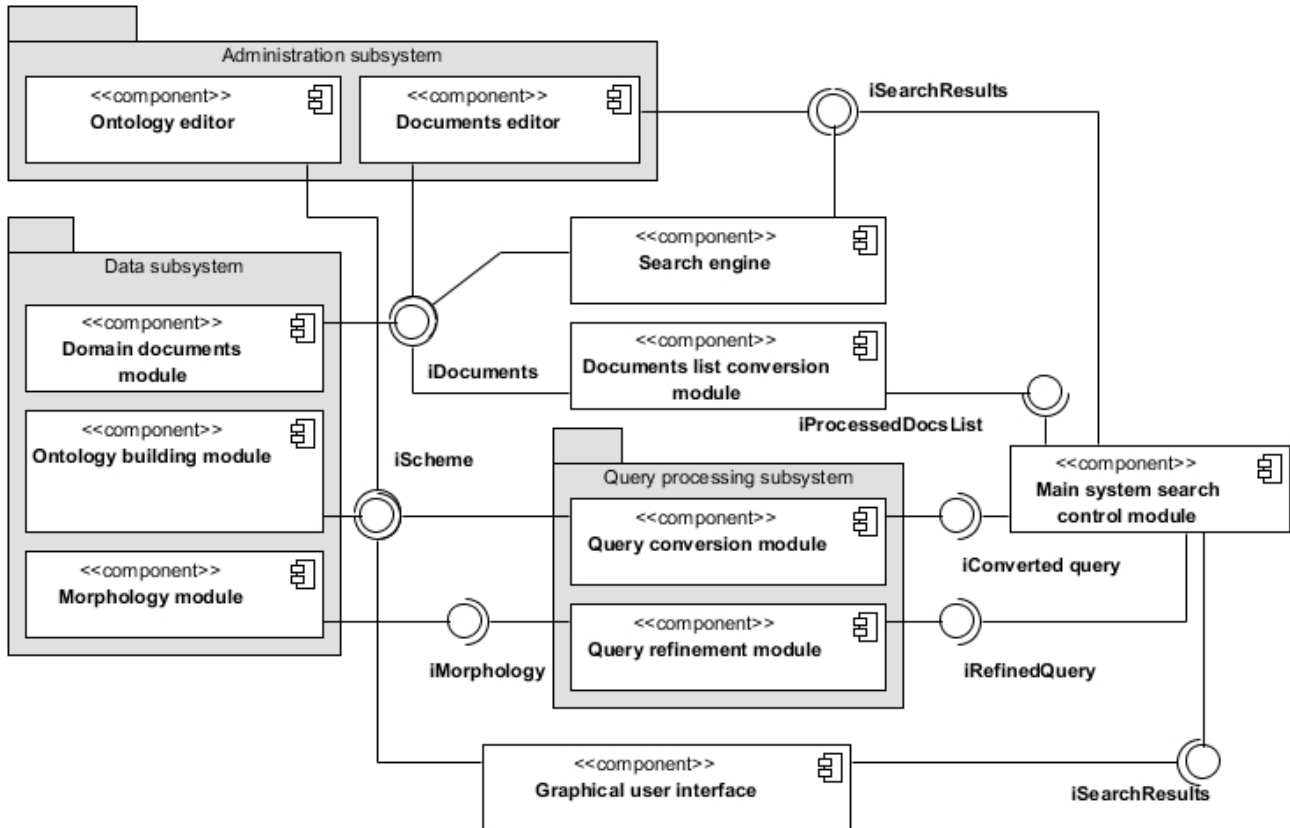


Fig. 2 - A diagram of components of the system under development

6. Data subsystem

6.1. Morphology module

This module is used by a query refinement module for the performance of search query conversion, input by a user. The module is implemented in the form of web-service producing various word forms of a received word.

6.2. Ontology building module

This module implements storage and delivering of a domain conceptual scheme for usage/editing. The conceptual scheme enclosed in this module is defined on OWL language. This module supports with an interface for a user access that allows overlooking the ontology that corresponds to a rubricator on a start stage. At the same time a choice of ontology (scheme of which will be delivered on a query) depends on the specified search parameters.

6.3. Domain documents module

This module encloses collections of documents and metainformation compliant to them according to that the search is performed by the context search module. This module provides interfaces of an access to documents for completion/editing of collections and also for the use of them for indexing/context search by a compliant system module.

7. Administration subsystem

7.1. Documents editor

It corresponds to special software that performs a preparation for loading, loading into the base, statistics computation and documents editing in the data base. It implements in two variants:

- on the system server an access to which is performed through web-interface;
- in the form of a graphical application for Windows operating system, in which all the operations on a preparation of documents are performed on a computer on which this application is launched that enables to somewhat diminish the server load.

In a process of functioning the editor uses the data base of documents for a loading of processing results and the context search model for a juxtaposition of handling documents with key words.

7.2. Ontology editor

This module provides the developers of ontology with an opportunity to build and edit the ontology that is used for a user's query conversion. For the implementation of this module functioning Protege ontology editor is used, that supports with all the functional required for this purpose.

8. Main system search control module

This module is the core of the whole system under development. It performs a search process and its control as well.

Conclusions

The developed architecture makes it possible to implement the described above requirements to the system and provides with some other advantages:

- an interchangeability of the system components that supports with an opportunity to change and replace the system components without touching the others;
- the system openness that enables an integration of already implemented components to it such as Data Base Management System, search engine, ontology editor and so on;
- an object organization that enables to create distributed system implementations (for example, on several functional servers). However, it is worth taking notice of the fact that the most dense co-location of components in a number of cases assist a significant improvement of the system safety.

The gained as a result of the development the technical frame of the system can be regarded as a framework for creating search systems in its way. This opportunity appears proceeding from the implementation of the second requirement to the system – the search theme dependence only on the ontology and the base of documents, and also on the flexible parameters of the system functioning such as an opportunity to switch on/switch off the use of morphology and conceptual scheme, a choice of given to a user of the system operations over a search results and so on.

The retrieval system is being developed according to the project “The informational system for the search of data and documents and the integration of knowledge in the subject field “public management” by means of theme ontology” by the grant of Russian Humanist Scientific fund № 11-02-12012v. (project director – A.V. Bogomolova, associate professor of Economic Faculty of M.V. Lomonosov MSU).

The ontology is being developed as an additional retrieval tool in “Russia” University Informational System for the integration of statistics data and subject analytical publications on a social-economic development of Russia, regions, municipalities [4, 5].

References

1. Data volume on the Internet is very close 5500 exabytes. It's 500 billion gigabytes. Technologies, Newsru.com, 2009. - [Internet resource]:[Article]. - URL: <http://hitech.newsru.com/article/19may2009/netvolume>
2. Lukashyevych N.V. Thesauri in information retrieval tasks. Moscow: Published by Moscow State University, 2011. 512 p.
3. Rosseeva O.I., Zagorulko Yu.A. Organization of efficient search based on ontologies. Proceedings of International Workshop on Computer linguistics and its applications “Dialog'2001”, v.2, 2001. -

[Internet resource]:[Article]. - URL: <http://www.dialog-21.ru/materials/archive.asp?id=7029&y=2001&vol=6078>

4. Yudina T.N. GIS technology for systems analysis subjects of Russian Federation for geostatistical data, *Prikladnaya Informatica*, 2011, №1 (31), p. 61-66.
5. Yudina T.N., Bogomolova A.V. MIS RUSSIA: building the infrastructure for a modern statistical education. Proceedings of the XII Russian Joint Conference “Internet and Modern Society” (IMS-2009), the 27 - 29 of October 2009, St. Peterburg, p. 8-12.

OBJECTS BEHAVIOR MODELING USING FUZZY LOGIC

Denis S. Zinchenko, 5-year student, Ivanovo State University of Chemistry and Technology, Russia, Ivanovo, denamorado@mail.ru

Edward G. Galiaskarov, Ph.D., Associated Professor, Ivanovo State University of Chemistry and Technology, Russia, Ivanovo, galiaskarov@isuct.ru

Currently one of the major problems of IT-industry is the intellectualization of software. Because of growth of stored information and requirements for the intensity of processing, experts of different subject areas need not only active assistance in decision making, but the full automation of the process. Thus, there is a problem of the experts' behavior simulation.

To solve the objects behavior modeling problem neural networks, finite automata, agent based systems, etc are used [1, 2]. A significant part of models are determinate. In this case object's behavior is strictly defined, depending on a number of conditions. However, this approach cannot be always applied to the human factor modeling. On the one hand a person must often make decisions in conditions of uncertainty. On the other hand a person takes different decisions in the same situations due to subjective reasons. Stochastic methods allow considering a probability. They are more applicable to obtaining statistics problems in the case of a large number of experiments. To make a decision in one case the use of random selection is not quite correct.

One of the ways to describe a human behavior is a using of fuzzy logic [3]. This method can be classified as determinate because of use of mathematical patterns and algorithms devoid of the element of randomness. On the other hand the use of such concepts as truth degree allows simulating a human behavior more flexible gradually getting rid of the uncertainty to solve the problem.

Let's consider in more detail a dialogue between the computer and the user in natural language as an example of human behavior simulation. That is the simulation of one of the talkers. Naturally, a communication on any subject is very vast and complex problem. Therefore, we consider a narrower task. A user accesses the virtual operator to pick up any product or service. The main purpose of the system is to choose the variant that suits user's requirements best of all. The main tasks in achieving this purpose are identification of user requirements and bring them to the formal type, convenient to automatically search for solutions. There are two main problems in the formation of user requirements:

- incompleteness of the requirements; there are some apparent to a person, so that they are not made by him at all;
- ambiguity of requirements caused by a multiple meaning of words of natural language and subjectivity meanings of these terms.

Consider the example of the problems described above. The customer selects the hotel voucher and makes the following requirements: "I want there to be not so hot". It would be clear to human that in this case the customer is unlikely to refer to a ski resort. But it is not clear to computer because this version is quite meeting the definition of "not so hot". As for a computer, as for the person, it would not clear what temperature the customer means when he says "not so hot".

To work with subjective concepts such element of fuzzy logic as a linguistic variable is used. Linguistic variable is cinque $\{x, T(x), X, G, M\}$, where

x is the name of variable;

$T(x)$ is the set of names of linguistic values of variable x (terms), each of which is a fuzzy variable on a set X ;

G is the syntactic rule for forming names of x values;

M is the semantic rule for associating each value with the concept [4].

Based on the definition, there are two main classes of objects: *LinguisticVariable* and *Term*. However, to take into account the synonymy property of natural language we depart from the classical definition. We separate two classes instead of a class of *Term*. The objects of *Term* are

mathematically defined values of linguistic variable. The words used to name these values we refer to the class *TermName*. The figure 1 shows the relationships between the described classes.

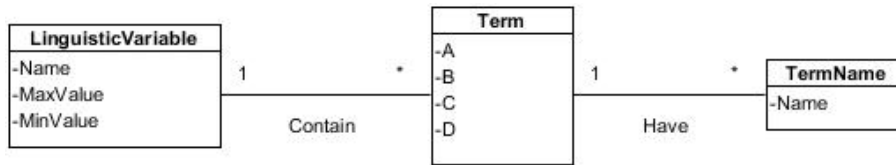


Fig. 1 - The fuzzy logic classes diagram

Consider the example with the selection of hotel vouchers. Customer may nominate requirements on the following criteria: cost, temperature, distance from the hotel to the beach, time on the road, etc. Common of all these criteria is that in the computer the information will be presented in numerical form in the appropriate measures while a person is likely to express their demands in a verbal narrative form. There are the criteria to be the linguistic variables. Every linguistic variable is characterized by a range of values.

The objects of *TermName* are words and their combinations, which can be used by customer in a case of values of linguistic variables (e.g. “expensive”, “hot”, “close to the sea”).

Compliance between the word and numeric value of linguistic variable is determined by the truth degree. The truth degree M is a number in the interval $[0;1]$, which describes how a statement is true. 1 matches an absolutely true statement, 0 matches an absolutely false statement.

The truth degree M of term T is described by quartet $\{a, b, c, d\}$, where x is a numerical value of linguistic variable.

$$M(T) = \begin{cases} 0, & x \leq a; \\ \frac{b-x}{b-a}, & a < x < b; \\ 1, & b \leq x \leq c; \\ \frac{x-c}{d-c}, & c < x < d; \\ 0, & x \geq d. \end{cases}$$

E.g. with the truth degree equal 1 we can say that the temperature of 35 degrees refers to the words “hot” and “heat”. Temperature of 25 degrees refers to these definitions with the truth degree equal 0.5 (Fig. 2).

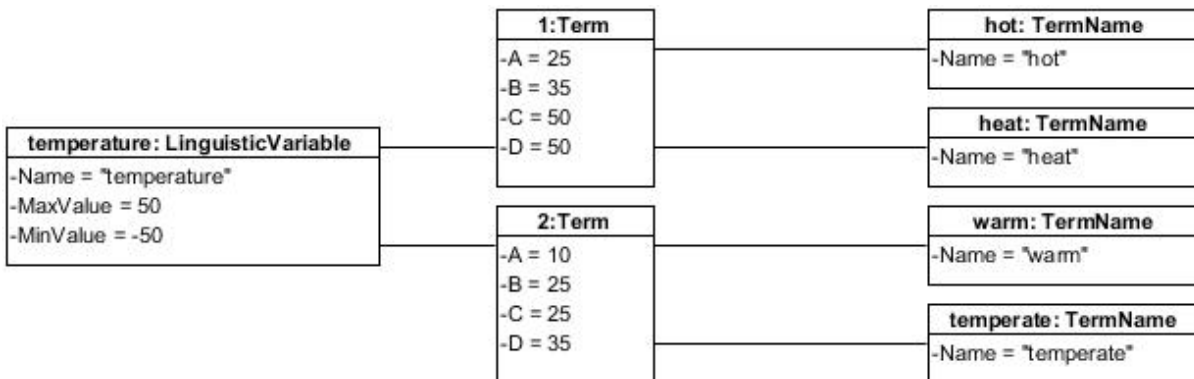


Fig. 2 - Object diagram

The problem of finding an optimal solution can be solved by exhaustion of all possible options and calculating the truth degree with each of them refers to the verbal description. The variant with the maximum truth degree will be the solution of the problem. When calculating the final degree of truth the following formulas of fuzzy logic can be used:

$$\begin{aligned}
 M(\text{NOT } A) &= 1 - M(A); \\
 M(\text{VERY } A) &= M(A)^2; \\
 M(\text{MORE OR LESS } A) &= M(A)^{1/2}; \\
 M(A \text{ AND } B) &= \text{MAX}(M(A); M(B)).
 \end{aligned}$$

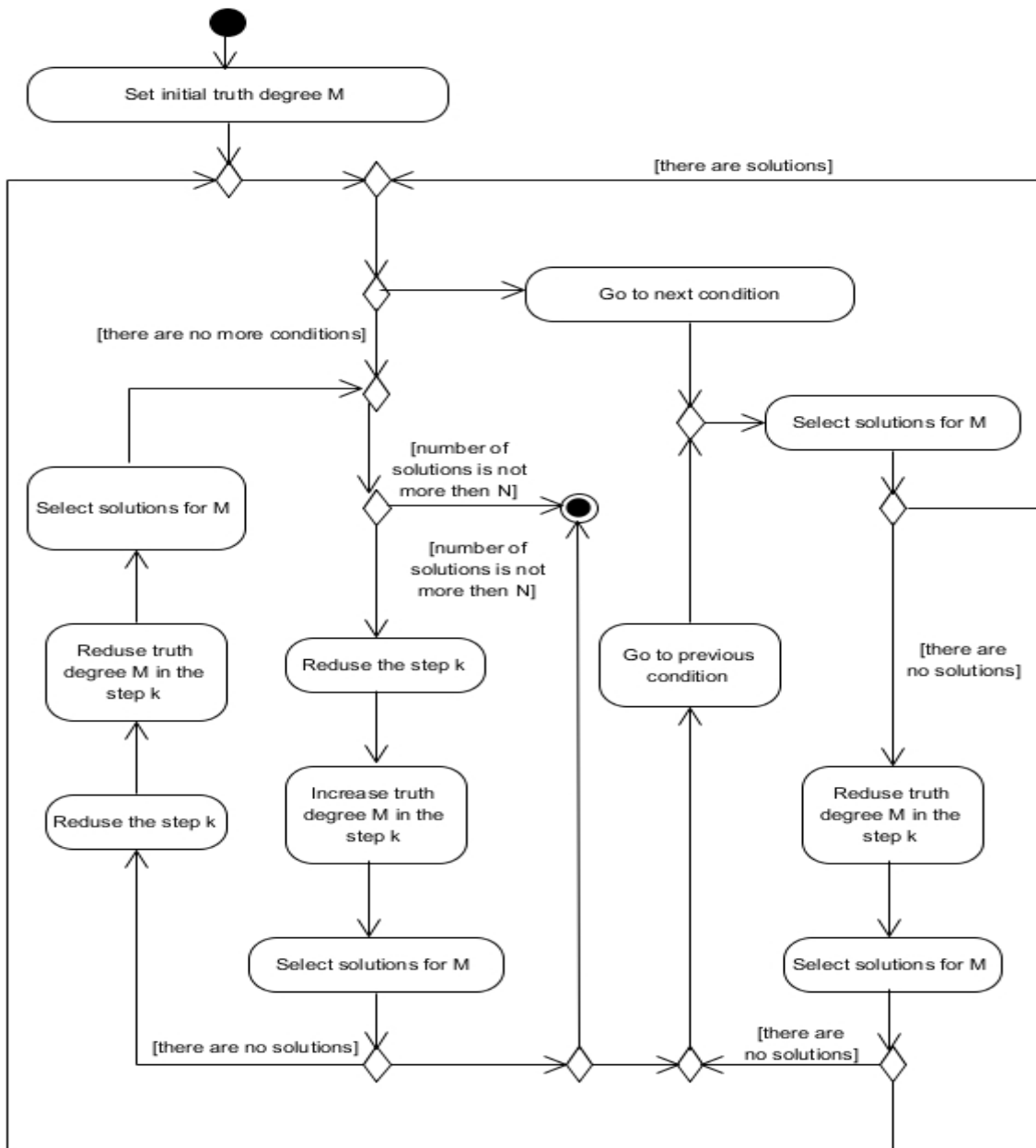


Fig. 3 - Algorithm of solutions search

The disadvantage of the above algorithm is shown by combining several conditions through the operator "AND", i.e. when a customer extends several conditions. If several versions have the same factor, which takes the lowest value, in fact only this factor affects the final decision.

In this regard, we propose another algorithm for selection of solutions, based on iterative analysis of every condition. In this case when iterating to the following condition there is no search from all possible variants, but only selected at the previous iteration (Fig. 3).

Every iteration consists on the selection of decision, which truth degree is not less than defined. If solutions are found, jump to the following condition. After an unsuccessful iteration, i.e. if after the transition to the next condition, the number of solutions was reduced to zero, we reduce truth degree first for condition under consideration, and then to all conditions in reverse order, until solutions are found.

If it reaches the last number of the conditions and the number of solutions is large enough, we need to make the iterative improvement of a defined truth degree in order to reduce the number of solutions.

There is a separate comparison for each criterion in the above algorithm. The influence of each factor will be approximately equal. In general, using this algorithm the truth degree of the first

criteria will be somewhat higher than in the past. This fact corresponds to the assumption that people most often refers to conditions in order of importance decreasing.

The above model allows to simulate the behavior of the experts in the decision making process, implementing the choice of a limited number of valid variants. As an expert, this model cannot guarantee that the solution will exactly refer to customer requirements. The solutions found can be considered close to optimal for the defined constraints.

References

1. Lychkina N.N. Modern simulation technology and its application in business information systems and decision support systems. Proceedings of the Second All-Russian scientific-practical conference on the use of simulation in industry "Simulation. Theory and Practice"- IMMOD 2005, St.Peterburg, the 19-21 of October, 2005.
2. Uskov A.A, Kuzmin A.V. The Intelligent control technology. Artificial neural networks and fuzzy logics. Moscow: Goryachaya linya-Telecom, 2004. 143 p.
3. Stuart Russel, Peter Norvig. Artificial Intelligence: A Modern Approach (3rd Edition). New Jersey: Prentice Hall, 2010.
4. Kruglov V.V., Dly M.I., Golubov R.Yu. Fuzzy logic and artificial neural networks. Educational book. M.: Izdatelstvo fiziko-matematicheskoy literatury, 2001. 224 p.

THE DISTRIBUTED SYSTEMS: MODERN REPRESENTATIONS AND DEVELOPMENT TENDENCIES

Alexei A. Sirotsky, Ph.D., The senior lecturer, The Russian state social university, Russia, Moscow, hotwater2009@yandex.ru

Anna O. Zvereva, The master of philology, English-First company, Russia, Moscow, glory-ann@yandex.ru

In modern conditions of scientific and technical progress systems of the distributed type win the increasing popularity and a wide circulation in all technical and economical areas.

So, the distributed systems (DS) become the most perspective and actual in problems:

- automated management by technological cars, processes and manufactures [1, 2];
- the automated gathering, processing, storage and transfer of the sociological, economic, scientific and technical information;
- the automated account and conducting operations of economic activities;
- increases productivity and reliability of calculations and great volumes of the information processing;
- conducting electronic document circulation;
- constructions of computer networks and communication networks of the general access;
- information resources creations;
- creation and conducting databases and knowledge
- information security maintenance;
- transition to electronic forms of service in bank sphere;
- condition of processes and objects monitoring.

This list of problems which can effectively be solved by using modern information technology and DS – is not completed, and the near future will open for DS application wide opportunities.

In DS it is necessary to provide collaborative work of the devices removed from each other and elements, systems of such organization have architecture difficult enough and a considerable quantity of components.

Thereupon it would be expedient to define what a DS is, however, it is impossible to give unequivocal, accurate and only right definition of DS.

It is necessary to classify DS to various signs taking into account their special-purpose designation.

So, it is important to consider hardware, program systems and their combinations by physical realizations. Certainly, it is necessary to carry the majority to last type. So, for example, the distributed file system should be carried to program realization DS, and computer networks can be considered as set DS of the combined type. Operating systems, applied software, control systems, calculations, information and database processing also can be distributed. The network «Internet» is also typical representative of DS where other DS can be find as components, for example, system of domain names (DNS – Domain Name System).

One of the main DS properties is possibility to acceptance the essentially of other management principle – management without the uniform center which has received the name «the decentralized management». Depending on appointment and structural realization DS distribution of operating resources in it can be various, therefore DS should be considered depending on degree of decentralization of management and knots quantity of the system having operating functions. As an example of high level decentralization system network «Internet» can also serve.

The term «distributed» means a certain division in time, sequence or space. To speak about DS it is necessary to mark territorial and functional distribution

The territorially-distributed systems represent systems, which parts (knots) are carried in space, for example, it can be one electronic system of document circulation of the enterprise which cases are in different cities or even countries. Certainly, the network «Internet» concerns this type also.

The functional-distributed systems represent systems where various parts (knots) solve strictly defined problems or functions.

So such systems can be:

- territorially-functionally distributed in which both kinds of distribution are observed: all knots of system are carried in space and each of them carries out functions strictly defined for it;
- territorially-distributed in which all knots carried in space carry out identical functions;
- functional-distributed in which all knots carry out strictly certain (target) functions, but actually, they can be in one place or even in space.

The example of the territorially-distributed and territorially-is functionally distributed systems are modern automation systems of technological processes on industrial productions. Territorial distribution pursues the aim to arrange the control means as close to objects of automation and objects of management on the industrial process equipment as possible that leads to the big economic gain as the total length of the demanded power and information cables which cost can reach 30 % from a total cost of automation system and management.

Each control mean of territorially-functional distributed control systems of technological processes carries out only the group of problems and according to certain units of the process equipment. Types of carried out problems are rather various: gathering, processing and data storage, mathematical calculations and development of operating influences.

Structural realization of such systems as much as possible correlates with structural realization of automation objects. Algorithmic programming maintenance of such systems essentially differs from the concentrated systems which were the cores before active distribution of the first. Modern software of designing and programming for DS automation and management are based on object-oriented programming languages application.

As a whole DS can be defined as multicomponent (multimodular, multinodal) system with functional and-or territorial division of components independent and cooperating among themselves. As system knots control means (in automation and management systems), terminals, devices information input-outputs, etc. can act as the COMPUTER.

The higher self-sufficiency of each knot of system is, the less is required their interaction among themselves and higher is efficiency of such system.

There are specific features of DS:

- the high reliability caused by partial duplication of separate parts functions of system and application of subsystems reservation;
- the adaptability to manufacture expressed in possibility of configuration change and system modernization;
- the high speed caused by functional distribution and the usage of several or even set of the COMPUTER;
- high flexibility, i.e. possibilities of fast systems readjustment (change-over, a reconfiguration);
- possibility of using real time operating systems;
- convenience of designing due to wide application of information technology, imitation means, modeling, the analysis and synthesis.

With reference to a design stage it is possible to note following kinds of models DS:

- the general model;
- the physical model;
- the resource model;
- the program model.

We can divide DS to the continuous (which consist of a large quantity of knots, and each knot of such system has the next related knots-analogs) and discrete (in which the final central structure and each traced knot is self-sufficient).

Functions and the problems which DS knots may solve can depend on various factors (which are considered when designing such system):

- environment properties of functioning system;
- a kind and sources of the entering information;
- the form of the target information and the requirement to it;
- the purposes and the problems solved by system using.

Speaking about the purposes, it is necessary to understand that any purposes can be reached in this or that degrees. Thereupon this degree of purposes achievement can be characterized numerically by means of the function which parameters are entrance and target sizes, criteria of an estimation and time:

$$Z = f(X_i, Y_i, C, t) \quad (1)$$

As the world experiment shows the greatest prospects of development and the brightest life cycle is observed at open type systems. By general understanding, open systems are such systems which are compatible to set of related objects, i.e. for software shipping, and for equipment rooms – compatibility and functional interchangeability is provided. If to look at open systems wider, the specified properties don't provide a high-grade openness. Users (first of all – professional) haven't enough these possibilities, as they don't give all possibilities functional system expansion at its completion and adaptation level. Therefore it would be desirable to see strengthening of expansion tendency of open systems concept which specifications are in open access for all comers. Here the specification is understood as all necessary complex of the documentation opening the device, a principle of action and the system organization, and sufficient for its perfection, change, modernization and adaptation. So, with reference to the software the full disclosure principle will mean availability of algorithm and the program text. So, open systems involve the greatest quantity of users, and it allows to assume that in the long term the DS of opened and completely opened types will occupy maximum share. One vivid example is the stunning success of firm IBM in the market of personal COMPUTERS. Firm IBM has given in open access to the specification of their working out that has caused the big interest of experts and fast further development of this architecture.

When constructing DS we use various sets of technical methods and the means named paradigms [3].

One of this paradigms is a system of the distributed objects which are understood as DS services and resources to which it is possible to have access.

On this paradigm are based:

- industrial standard DS CORBA;
- DS intermediate level DCOM of firm Microsoft, cooperating with OS Windows;
- the distributed objective model specification of the program interface of a remote methods call from virtual car Java RMI (Remote Method Invocation).

Standard CORBA has been developed for the purpose of intermediate program creation level with a view of teamwork software maintenance of various manufacturers.

System DCOM has been developed for preservation of the existing software compatibility with new operating systems.

The main DS requirement is a possibility of sharing programs and information by different knots and users. Thereupon in DS file systems (FS) with corresponding functionality – distributed FS are applied, most known of which are network FS NFS companies Sun Microsystem and distributed FS Coda, developed in Carnegie Mellon University. FS NFS originally intended for platform UNIX using, but then has passed to other platforms. Coda it is applied in platforms UNIX and Linux and provides continuation of normal work at rupture connection with the server.

As in DS the multiuser access to files distributed FS should define criteria of a file change. So, in NFS definitive change of a file is defined by last knot (client) who has closed a file opened on record. In distributed FS Coda for display of last file change it is necessary to rediscover.

Now program functional model DS is represented in the form of program levels, or the layers going from the lowest (machine) level to highest (user or man-machine-interface) level. Each level is the successor of functionality of higher level. The level model of DS software allows to consider each level irrespective of others as functionally complete part of system. But there are restrictions on incapsulation in the object-oriented approach and differentiation of level functionality. And this is the greatest problem in DS constructed on a principle «client-server».

It is necessary to notice nowadays within the limits of structure «client-server» the most widespread is a problem of information systems construction working with databases (DBMS), and with Web-interface use at client level. The given problem arises according to Internet, and Intranet.

Here are the most extended «client-server» types:

- one server – a lot of clients;
- many servers and many clients;
- many clients – an intermediate proxy-server – servers.

Dialogue modes between clients and servers can be constructed by following principles:

- communication of the client initiative;
- communication of the server initiative;
- both client and server initiative.

Interactions between clients and servers can be following types:

- the client forms a problem to the server, the server solves it and sends result to the client (typically for databases and knowledge);
- the server forms a problem to the client, the client solves it and sends the result to the server (typically for the distributed calculations of high complexity);
- the mobile agent: the client informs the server about the arisen problem, and the server sends to the client the program module for its decision in the answer;
- the thin client: the client has no enough resources for perform any problems, and works as the remote terminal of input-output, handing over the entering information for continuous processing to the server, and reproducing only the information received from the server and of the same kind;
- Peer to Peer: the client and the server practically aren't distinguishable in system, have the equal rights and possibilities, and can act in that or other quality.

The information interchange problem in a network has five basic levels:

- physical – the interface of communication of the transferring environment with the sending device;
- network – information interchange between the server and a network;
- gateway – an information transfer to other network;
- transport – management of information transfer and control;
- level of applications – realization of the user program.

At gateway level report Internet Protocol (IP), and on transport – report Transmission Control Protocol (TCP) is applied.

For working out the distributed applications under Internet with Web-interface use at client level there are many possibilities, however recently the increasing popularity is earned by a package java.net [4].

Effective DS working out should be under construction on the basis of accurate statement of the purposes and problems, formation of requirements to DS, definitions of its use conditions, and also necessity of support and service. Considering DS expansibility, additional reserve resources and possibilities should be put in them.

Making DS types, components and also their modeling, considerably simplifies processes of their analysis and synthesis.

One of the most responsible moments defining DS conception, its structure and the basic functionality, is a motivated and well-founded choice of its architecture, modes and work reports.

Program DS realizations are represented by the multilevel scheme, and definition of levels quantity, their functional division, interaction and encapsulation represents a serious technical problem, and the single algorithm to such a decision doesn't exist.

The most preferable are the decisions based on technologies of open systems, as having statistically longer life cycle.

References

1. Sirotskiy A.A. Application of modern microprocessor control systems for automation of technological processes of foundry manufacture.//Magazine «Founder of Russia», 2005, №3, p. 40-44.
2. Sirotskiy A.A. Problems of automation of technological processes of foundry manufacture and means of their decision. The collection of the selected reports of 49th MNTK AAE of Russia «Priorities of development domestic autotractorbuilding and preparations of engineering and scientific shots of 4th international scientific symposium «Modern autotractorbuilding and the higher school of Russia». Moscow, MGTU «MAMI», 2005. The book 5. p. 67-77.
3. Tanenbaum E.S, Van Steen of M. The Distributed systems. Principles and paradigms. SPb.: Peter, 2007.
4. Svistunov A.N. Construction of the distributed program systems on Java. BINOM. Laboratory of knowledge, 2010.

A COMPARISON STUDY OF OBJECT-ORIENTED DATABASE MANAGEMENT SYSTEMS

Euclid Keramopoulos, Ph.D., Lecturer, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki, euclid@it.teithe.gr

Michael Zounaropoulos, Student, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki, mzounar@it.teithe.gr

George Kourouleas, Student, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece, Thessaloniki, gkourou@it.teithe.gr

Abstract. In recent years Object-Oriented Database Systems have a remarkable growth and obtain a larger share of the market of database systems. This is a consequence of the advantages and faster performance of Object-Oriented Database Systems compared with Relational Database Systems. In this paper, we present the features that an Object-Oriented Database System should support. Moreover we compare the eight more powerful Object-Oriented Database Systems and we present a comparison table between those systems according to the features we introduce.

Introduction

Object-Oriented Database Management Systems (OODBMS) are originated from Object-Oriented Programming Languages (OOPL). Thus, OODBMS support the characteristics that made OOPLs successful and furthermore OODBMS integrate OOPLs by offering the possibility of storing objects permanently as in every database management system.

The last decade many Object-Oriented Database Management Systems (OODBMS) have been developed. The most known of them are: Gemstone (Smalltalk) [1], Jdostruments (objectDB) [2], Versant Object Database [3], ObjectStore [4], Jade [5], Matisse [6], db4objects [7], Objectivity [8], Progress Software [9], EyeDB Object Oriented Database Management System [10], McObject Perst [11], Cache [12], ODABA [13], Eloquera [14], Generic Object Oriented Database System (GOODS) [15], JODB (Java Objects Database) [16], MyOODB [17], NeoDatis ODB [18], Orient ODBMS [19], Ozone Database Project [20], Statice [21], VOSS (Virtual Object Storage System) [22], Obsidian Dynamics (DTS/S1) [23].

In this paper we organize the features that an OODBMS should support and we draw a comparison table between the eight most important API for Object Persistence and OODBMS. The structure of this paper is as follows. Next, in section II, we introduce the features that an OODBMS should support. In section III, we briefly analyze the eight more powerful API for Object Persistence and OODBMS and in section IV, we present a comparison analysis between those APIs and OODBMSs. Finally, in section V we draw our conclusion.

1. Features of OODBMS

In this section we organize the features that an OODBMS should support as a result of our review. First of all, it is very important for an OODBMS to support the ODMG 3.0 standard [24]. This includes the basic components of the ODMG 3.0 standard, namely Object Definition Language (ODL), Object Query Language (OQL) and furthermore powerful types like collections and complex types. In the ODMG 3.0 standard are supported five types of collections, i.e. set, bag, list, array and dictionary, which means that it is expected from an OODBMS to support most of them. Complex data types are stated for user-defined data types like classes and structures. Moreover, a big issue in ODMG 3.0 is the type of inheritance that is supported. In our review, all of the OODBMS support single inheritance except of only two, which support multiple-inheritance.

OODBMS is also important to support traditional database features. Thus, an OODBMS should support concurrency control, recovery and indexing. On the other hand it is useful for an OODBMS to provide a query facility. In our review it was found that OODBMS use three types of query structure, i.e. ODMG 3.0 OQL [24], NoSQL [26] and LINQ [27].

Another aspect that it was studied in this review is the software portability. Thus, it was checked the platform that those OODBMS were compatible (Windows, Android , Linux, Mac). Moreover, another important feature is the usage of the OODBMS in three layer architecture, i.e. if the OODBMS can installed in a database server and accessed by a client.

The user interface that an OODBMS supports it's another category of features that are significant because the OODBMS becomes more efficient when it provides a user-friendly user interface. Thus, three types of user interfaces are met in a OODBMS: (a) a Graphical User Interface, in order to manipulate all the services of the OODBMS, (b) a database user interface, in order to design and present the structure of the database and the actual data and (c) the data explorer, in order to represent in a tabular form the data of objects. Also, it is advantage for an OODBMS the possibility to export the data in XML format, which is the standard format of data from a lot of web applications. Finally, if an OODBMS provides JDBC [28] type metadata of object data then it is possible data dynamically applications to be created.

2. Presentation of most representative OODBMSs

Two OODB APIs, JDO and JPA along with the six more successful OODBMS are presented, namely, db4o, ObjectDB, Objectivity/DB, EyeDB, Perst and ObjectStore. We include in our review JDO and JPA because a programmer can use those APIs in order to manage object persistence.

Java Persistence API (JPA) [29] is an API that can be used in J2EE and J2SE applications. It is based on leading persistence frameworks like Toplink [30] and Hibernate [31]. JPA offers POJO (Plain Old Java Object) standard [32] and object relational mapping (OR mapping) [33] for data persistence among applications.

Java Data Objects (JDO 2.0) [34] is an API regarding data persistence that now is developed inside Apache JDO open-source project [35]. JDO 2.0 uses POJO to represent persistent data. This approach divides data manipulation (accessing Java data members in the Java domain objects) from database manipulation (calling the JDO interface methods). This leads to a high degree of independence of the Java view of data from the database view of the data.

db4o [7] is an open source OODBMS that enables Java and .NET developers to store and retrieve any object. In db4o objects are stored exactly in the same way that are defined in Java or .NET applications. Moreover, db4o supports the persistence of any object without concerning of its complexity. It is designed to be embedded in clients invisible to the end user. Client/server version allows db4o to communicate between client and server-side applications.

ObjectDB [2] is a powerful OODBMS, which is reliable, easy to use and extremely fast. It provides all the standard database management services (storage and retrieval, transactions, lock management, query processing, etc.). Many features of ObjectDB based on JPA and JDO APIs. Moreover, features that are common in relational databases (e.g. primary keys etc) are also supported by ObjectDB.

Objectivity/DB [8] is a commercial OODBMS produced by Objectivity. It allows applications to make standard C++, Java, Python or Smalltalk objects persistent. Objectivity/DB supports also SQL/ODBC and XML. It runs on Linux, LynxOS, UNIX and Windows platforms. Objectivity/DB is a distributed database that provides a single logical view across a federation of databases.

EyeDB [10] is an OODBMS based on the ODMG 3.0 standard, which developed and supported by the French company SYSRA. EyeDB provides an advanced object model (inheritance, collections, methods, triggers, constraints), an object definition language based on ODMG ODL, an object query language based on ODMG OQL and programming interfaces for C++ and Java. EyeDB distributed under the terms of the GNU Lesser General Public License.

Perst [11] is an Open Source OODBMS developed by McObject LLC. Perst stores data directly in Java objects, which boosts runtime performance. It is compact, with a core that consists of only 5000 lines of code and thus, it demands minimal resources due to this small footprint. It is reliable and it supports transactions with the ACID properties. Finally, it provides schema evolution, XML import/export, database replication, and support for large databases.

ObjectStore [4] is an OODBMS that supports traditional DBMS features such as persistent storage, transaction management, distributed data access, and associative queries. ObjectStore was designed to provide a programmatic interface to persistently allocated data (data that lives beyond the execution of an application program) and transiently allocated data (data that does not survive beyond an application's execution).

3. Comparison Analysis

In table 1 a comparison analysis is introduced based on our review on the two APIs and six OODBMSs that briefly presented in the previous section. In table 1, it is shown the features that every OODBMS of our review supports. The symbol “√” is used to present that the feature is supported by the OODBMS and the symbol “?” is used to present that it is not clear if the feature is supported. An empty cell shows that the OODBMS does not support the corresponding feature.

Table 1. Comparison Table of OODBMS [1]

Features / OODBMS's		JPA	JDO	db4o	ObjectDB	Objectivity	EyeDB	Perst	Objestore
Support of ODMG 3.0		√	√		√	?	√	?	
Collections	Set	√	√	√	√	√	√	√	√
	Bag						√		
	List	√	√	√	√	√		√	√
	Array	√	√	√	√		√	√	√
	Dictionary	?	?	?	?	?	?	?	?
Complex Types		√	√	√	√	√	√		
Multiple Inheritance		√	√						
Concurrency Control		√	√	√		√	√	√	√
Recovery		√	√	√	√	√	√	√	
Indexing			√		√	√	√	√	√
Query Facility	OQL	√	√	√	√	√	√	√	
	LINQ	√	√	√	√	√		√	
	NoSQL								
XML export	Internal					√	?	√	
	Third Party Program			√			?		
Metadata JDBC type		√	√		√		√		
Platform	Windows	√	√	√	√	√	√	√	√
	Android			√				√	
	Linux	√	√	√	√	√	√	√	√
	Mac	√	√	√	√	√		√	
Client-Server Edition		√	√	√	√		√	√	
Interface	GUI					√			
	DBMS UI				√		√		
	Data Explorer			√	√	√	√		

From the above table it concluded that the reviewed OODBMSs support most of the features but no one all of them. More particularly, ODMG 3.0 standard supported by JPA, JDO, ObjectDB and EyeDB but OQL supported also by db4o, Objectivity and Perst. The collection type supported by most of the OODBMS except of the bag type, which supported only by EyeDB, and for the dictionary type is not clear if any OODBMS supports it. In the six OODBMSs a complex type can be created except of Perst and Objectivity. Only the two APIs allows multiple inheritance when all the OODBMS limited to single inheritance. The traditional DBMS features are supported by the vast majority of the OODBMSs. No one of the reviewed OODBMS provides the possibility of defining a NoSQL query. Only db4o, Objectivity and Perst can export the object data into XML data. The JDBC type metadata can be used only from applications that use as Object-oriented persistence the two APIs or ObjectDB and EyeDB. Only, db4o and Perst are portable to the four platforms whereas the most of the OODBMS can be used in three-tier architecture. Finally, only db4o, ObjectDB, Objectivity and EyeDB supports a user-friendly interface whereas all the others only a textual one.

Conclusions

In this paper we presented a list of features that an OODBMS should support. Also, we briefly outlined two APIs that can be used by object-oriented applications for object persistence and six

well-known OODBMSs. Moreover, we draw a comparison analysis table where it is shown which feature supported by every OODBMS. From the comparison analysis it was concluded that no OODBMS supports all the desirable features that an OODBMS should support. In our future plans included the design and development of a prototype that will map an object-oriented database into an equivalent semi-structured XML database.

Acknowledgements

The work presented in this paper has been supported by the Research Committee of Alexander Technology Educational Institute of Thessaloniki under the Research Support Program 2009 (Π.Ε.Ε. 2009).

References

1. Gemstone (Smalltalk). Web site: <http://www.gemstone.com/>, Accessed 2011.
2. Jdostruments (objectDB). Web site: <http://www.jdostruments.org/>, Accessed 2011.
3. Versant Object Database. Web Site: <http://www.versant.com/>, Accessed 2011.
4. ObjectStore. Web Site: <http://web.progress.com/en/objectstore/>, Accessed 2011.
5. Jade. Web Site: <http://www.jadeworld.com/>, Accessed 2011.
6. Matisse. Web Site: <http://www.matisse.com/>, Accessed 2011.
7. db4objects. Web Site: www.db4o.com/, Accessed 2011.
8. Objectivity. Web Site: <http://www.objectivity.com/>, Accessed 2011.
9. Progress Software. Web Site: <http://web.progress.com/en/index.html>, Accessed 2011.
10. EyeDB Object Oriented Database Management System. Web Site: <http://www.eyedb.org/>, Accessed 2011.
11. McObject Perst. Web Site: <http://www.mcobject.com/perst>, Accessed 2011.
12. Cache. Web Site: <http://www.intersystems.com/cache/>, Accessed 2011.
13. ODABA. Web Site: <http://odaba.com/content/start/>, Accessed 2011.
14. Eloquera. Web site: <http://eloquera.com/page/home.aspx>, Accessed 2011.
15. Generic Object Oriented Database System (GOODS). Web site: <http://www.garret.ru/goods.html>, Accessed 2011.
16. JODB (Java Objects Database). Web site: <http://www.java-objects-database.com/>, Accessed 2011.
17. MyOODB. Web site: <http://www.myoodb.org/>, Accessed 2011.
18. NeoDatis ODB. Web site: <http://www.neodatis.org/>, Accessed 2011.
19. Orient ODBMS. Web site: <http://www.orienttechnologies.com/cms/>, Accessed 2011.
20. Ozone Database Project. Web site: <http://www.ozone-db.org/frames/home/what.html>, Accessed 2011.
21. Stalice. Web site: <http://www.sts.tu-harburg.de/~r.f.moeller/symbolics-info/statice.html>, Accessed 2011.
22. VOSS (Virtual Object Storage System). Web site: <http://voss.logicarts.com/>, Accessed 2011.
23. Obsidian Dynamics (DTS/S1). Web site: <http://obsidiandynamics.com/dts/index.html>, Accessed 2011.
24. Cattell R.G., Barry D.K.. The Object Data Standard: ODMG 3.0 (Morgan Kaufmann Publishers) 1999.
25. Strozzi, Carlo: NoSQL - A Relational Database Management System. Web Site: http://www.strozzi.it/cgi-bin/CSA/tw7/1/en_US/nosql/Home%20Page, Accessed 2010.
26. LINQ. Web Site: <http://msdn.microsoft.com/en-us/netframework/aa904594>, Accessed 2011.
27. Maydene Fisher, Jon Ellis, Jonathan Bruce. JDBC™ API Tutorial and Reference (3rd Edition). The Java Series, Sun, 2003.
28. Daoqi Yang. Java(TM) Persistence with JPA. Outskirts Press, 2010.
29. Oracle Toplink. Website: <http://www.oracle.com/technetwork/middleware/toplink/tl-grid-097210.html>, Accessed 2011.
30. Boss Hibernate. Web Site: <http://www.hibernate.org/>, Accessed 2011.
31. Brian Sam-Bodden. Beginning POJOs From Novice to Professional. Apress Media LLC, 2006.
32. M.L. Fussell. Foundations of object relational mapping. White Paper, <http://www.chimu.com>, 1997
33. Oracle JDO. Web Site: www.oracle.com/technetwork/java/index-jsp-135919.html, Accessed 2011.
34. Java Data Objects. Web Site: <http://db.apache.org/jdo/>, Accessed 2011.

SCALING FROM OOP TO SOA AND ENTERPRISE ARCHITECTURES. COMPLEXITY CONTROL AND EVOLUTION SUPPORT IN SOFTWARE DESIGN

Ilya E. Ermakov, lecturer, Polycarpov Institute of Technology at the State University - Educational-Scientific-Industrial Center, CTO, NPO «Tesla», Russia, Orel, ilya@ermakov.net.ru

1. Reliability and Evolvability

Essentially, reliability and evolvability of systems are two sacred goals ahead software engineers. What is the main gap in reliability? Of course, it's complexity of systems. In coherence with this fact, a lot of programming tools, abstractions and principles may be considered as **control complexity facilities**. Control complexity requires, first of all, abstractions, and, secondly, some strict discipline in system design and implementation. Such discipline helps us to overcome new large problems, but it may be strait for future life of a system because of flexibility lack.

Abstractions and Discipline => Complexity Control => Reliability
Reliability and Flexibility => Evolvability

Fig. 1 - Main Forces of Effective Design

Evolvability is the second basic property of system design — to be ready for a long life in volatile world. Complexity control is the first condition for evolvability because unreliable system can't have long life. But not all reliable systems are evolvable. Squeeze structure is a cost of reliability. Therefore flexible areas should be established during design if we need some freedom for system evolution. We introduce reliable skeletons over a homogeneous set of elements (for example, strict typing over plain memory) and after than we'll be forced to introduce facilities for flexibility over the squeeze layer.

Such contraries should not be perceived as disappointing. True system design demands well-balanced thinking from an engineer. There are many well-known things in IT history (LISP, Forth and some of modern dynamic languages), that have beautiful abstractions and homogeneous structure. But abstractions without a discipline (that need to be implemented as some concrete facilities) don't achieve true complexity control and reliability. And flexibility without reliability don't achieve evolvability. All languages used for large system design (Ada, Java, Component Pascal, C#, chronologically, and others) are balanced for all sides of described above.

2. Modular and Object-Oriented (OO) Facilities

Modular and object-oriented facilities were introduced in programming languages independently one of another, but because of similar problem's vision. Actually, two these approaches may be considered in overlaped notions — let's try to do it.

We'll tell about so called «true modules» based on D. Parnas conceptions and supported by special language constructions in modular languages (Mesa, Modula-2, Ada, Object Pascal, Oberon, chronologically, and others). True module is:

1. a part of division system;
2. a unit of hiding internals (hiding statical information);
3. a unit of hiding dynamical state (by an interface consisting of abstract operations; and it's nothing like encapsulation).

There are some other practical flavors of true modules: they are units of separate (but not independent!) compilation and dynamical loading and linking (as it's prevalent in the Oberon's family).

More than, we should behold that a module may be a unit of variation several implementations of one interface (several implementations for one definition). And it's nothing like polymorphism.

So, true modular programming helps us to create complex and easily modifiable systems. But modification is not as yet evolution generally.

Native OO languages (Simula and its derivatives) and «total OO» languages (Smalltalk etc.) support (2) and (3) on a fine-grained level: for data types and data instances. Such abstract data types and their instances have been named classes and objects. OOP in software design introduces explicit expression of classification relations between entity types. Facilities of inheritance have long history — from wars between single and multiple inheritance to modern principle of pure interface extension, without implementation inheritance (and without problem of fragile base classes[1]). Since OOP is fine-grained modularization of dynamical data instances, it opens way to dynamic reconfiguration and extension of software (well-known fantastic flexibility in Smalltalk environments).

As we discussed above, real engineering requires not so flexibility, but also large-scale design and reliable ribs. Native OOP has powerful facilities for extension, but true modular programming supports reliable middle-scale organization of systems. Its main cause for incorporation of modular facilities in OO languages (classical modules similar Modula-2 etc. have been offered in Java SE 7). Concept «module is a singleton object» is shallow because these facilities live on different layer of system architecture.

Another way is to incorporate OO facilities in modular languages (Oberon, Ada-95, Component Pascal). Class-object notion is superfluous in this case. Modularity, data type extension and strong typing establish base for reliable and evolvable design (as in component-oriented paradigm [1]). Some modern languages such as Google Go use this approach too.

3. SOA and REST essentials

Let's remember what we have before SOA. Broadly speaking, there were modular and OOP architectures and component programming for control complexity at a middle level, there were relational databases for separation data-model level in enterprise systems and there was multitiered structure for separation information processing from interactions with external world.

A lot of approaches (OO-DBMS, CORBA, RMI, DCOM etc.) have been concentrated on scaling facilities of OO languages for needs of large distributed systems. There are two problems: first of all, fine-grain facilities of OO languages are meticulous on this level — it may implicate vague, unreliable and inefficient design; secondly, very successful clean separation between data level and other tiers is lost. It may be posited that any new approach shouldn't lose division lines introduced for complexity reduction formerly.

An initial idea of SOA was simple, natural and implementation-independent: to keep horizontal lines of tier separation and to force vertical lines between different group of functions. Every such group should be encapsulated in enough independent service. A service encapsulates responsibility for some database partition and offers set of actions for its clients. We see same logic as it have been described above for modularity. And, of course, ability to vary several implementations of service is attractive for enterprise system's engineers. For such variation (and dynamical binding and reconfiguration) some middleware was suggested by several vendors. Enterprise bus for service communication is example of simple and effective facility.

There are more and more critique of SOA today. Such critique is because of complexity, low flexibility and high use cost of all favourite SOA middleware. Total interface specifications don't add measurable growing of reliability, but greatly decrease evolvability. A lot of trash features cover clean idea of SOA. We shouldn't forget, that approaches such SOA are oriented not to implementation, but to design, so we need nothing fat tools to support them.

And so, what is the REST approach — new fashion or an objective trend? It is clearly on base of comparison modular and OO facilities, that has been done above. Let's rely on SOA gains, introduce explicit identification of enterprise informational resources and apply fine-grained service-oriented interfaces to each resource — and we have REST, that binds data resources with services same as OOP binds data structures with procedures. It's meritoriously that REST is based on unused potential of simple HTTP protocol. Well-known principle of hiding real processing implementation under requests to abstract URIs is yet another example of modular and object-oriented (polymorphic) facility.

Summary

There is some trend in evolution from modular and OO programming to modern enterprise architectures like SOA&REST. This trend is oriented to control complexity and support evolvability in large systems. Essentials of enterprise architectures are simple and implementation-independent.

References

1. C. Szyperski. Component Software. Beyond Object-Oriented Programming. Addison-Wesley Longman, 1998.

MODELING DIGITAL LIBRARY MANAGEMENT SYSTEM BASED ON THE METHODS OF QUEUING THEORY

Elena I. Astasheva, Graduate, Voronezh State Technical University, Russia, Voronezh,
astasheva_elena@mail.ru, astasheva@vorstu.ru

Constantine A. Razinkin (supervisor), Doctor of Technical Sciences, Professor of technology and automated systems of Electronic Engineering, Voronezh State Technical University, Russia, Voronezh

In today's rapidly changing and dynamic world of e-science a library should be represented by a system of distributed servers to be able to provide relevant information. This system has long proven itself on the positive side, because to achieve high performance is required high concentrations of productive capacity in the same place and in a single device, as well as a repository of information (which is true for the electronic library) can be increased almost indefinitely.

As part of my research system of distributed servers is an integral part of effective implementation of the constructed models of flow control applications in the electronic library. We turn to these models.

Let's abstract from the electronic library's functions and represent it as a system (Figure 1) with one input and one output (the result of processing the application: a positive or negative).

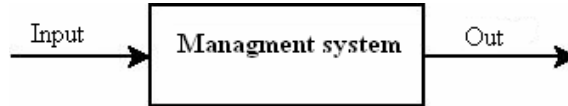


Fig. 1 - Management system

To create an effective management system it's important to create its model. One method of constructing models of the system is a method of queuing theory. Here, this method will be used to service the flow of requests.

Let's imagine that our "black box" in Figure 1 - a multi-channel queuing system (the number of channels is n). The scheme of this kind of system is shown in Figure 2.

Set up a system of differential equations for the probabilities of system states. Because a n -channel, our system has a queue with an arbitrarily large number of applications s , the system of differential equations takes the form:

$$\begin{aligned}
 \frac{dp_0(t)}{dt} &= \sum_{i=1}^n \mu_{0i} p_0(t) - \lambda p_0(t) \\
 \frac{dp_{01}(t)}{dt} &= f(1)\lambda p_0(t) - (\mu_{01} + f(1)\lambda)p_{01}(t) + \mu_{01} p_{n+1}(t) \\
 \frac{dp_{02}(t)}{dt} &= f(2)\lambda p_0(t) - (\mu_{02} + f(2)\lambda)p_{02}(t) + \mu_{02} p_{n+1}(t) \\
 &\dots \\
 \frac{dp_{0n}(t)}{dt} &= f(n)\lambda p_0(t) - (\mu_{0n} + f(n)\lambda)p_{0n}(t) + \mu_{0n} p_{n+1}(t) \\
 &\dots \\
 \frac{dp_{n+1}(t)}{dt} &= \lambda \sum_{i=1}^n f(i)p_{0i}(t) - \left(\sum_{i=1}^n f(i)\lambda + \sum_{i=1}^n \mu_{0i} \right) p_{n+1}(t) + \sum_{i=1}^n \mu_{0i} p_{n+2}(t) \\
 \frac{dp_{n+2}(t)}{dt} &= \lambda \sum_{i=1}^n f(i)p_{n+1}(t) - \left(\sum_{i=1}^n f(i)\lambda + \sum_{i=1}^n \mu_{0i} \right) p_{n+2}(t) + \sum_{i=1}^n \mu_{0i} p_{n+3}(t) \\
 &\dots \\
 \frac{dp_{n+s}(t)}{dt} &= \lambda \sum_{i=1}^n f(i)p_{n+s-1}(t) - \left(\sum_{i=1}^n f(i)\lambda + \sum_{i=1}^n \mu_{0i} \right) p_{n+s}(t) + \sum_{i=1}^n \mu_{0i} p_{n+s+1}(t)
 \end{aligned} \tag{1}$$

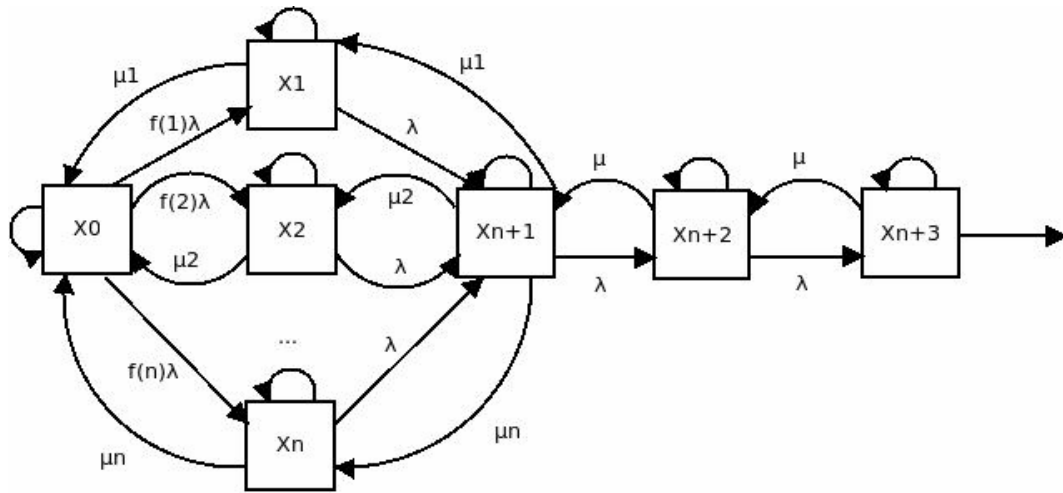


Fig. 2 - Multi-channel queuing system

We simplify the system of differential equations and obtain an expression for the probability of link-state, with $1 \leq k \leq n$:

$$P_{0k} = \frac{f(k)\lambda \left[\sum_{s=1}^{\infty} \frac{\lambda^{s+1}}{\mu_{np}^{s+1}} + \frac{\lambda}{\mu_{np}} \right]^{-1} + \frac{\lambda^2 \mu_{0k} \left[\sum_{s=1}^{\infty} \frac{\lambda^{s+1}}{\mu_{np}^{s+1}} + \frac{\lambda}{\mu_{np}} \right]^{-1}}{\mu_{0k} + f(k)\lambda} \quad (2)$$

continue to turn states for $s \geq 1$:

$$P_{n+s} = \frac{\lambda^{s+1} \left[\sum_{i=1}^{\infty} \frac{\lambda^{i+1}}{\mu_{np}^{i+1}} + \frac{\lambda}{\mu_{np}} \right]^{-1}}{\mu_{np}^{s+1}} \quad (3)$$

As a result of these expressions, we can derive the formula for calculating the average number of requests in the queue:

$$m_s = M[S] = \sum_{s=1}^{\infty} s P_{n+s} = \left[\sum_{i=1}^{\infty} \frac{\lambda^{i+1}}{\mu_{np}^{i+1}} + \frac{\lambda}{\mu_{np}} \right]^{-1} \cdot \sum_{s=1}^{\infty} \frac{s \lambda^{s+1}}{\mu_{np}^{s+1}} \quad (4)$$

As a result of these expressions, we can derive presented formulas to give a limiting law for the number of busy channels, depending on the flow of applications and performance management of electronic research library [1, 2, 3, 4]. By using the above expression for the probability of the system states we can analyze the work of the overall management system, a model of the system close to real operating conditions, to determine the effectiveness of the system and offers options for optimizing the management of e-science library to achieve the most desired service option applications.

The distribution function of the flow of requests will be used to test the mechanism of distribution of applications in an n-channel system. The distribution function will be as follows:

$$f(i) = \frac{\lambda^2 \varphi}{j \sum_{i=1}^n \frac{\lambda}{i}} + \frac{\lambda^2 (1 - \varphi)}{(m - j) \sum_{i=1}^n \frac{\lambda}{i}} \quad (5)$$

where $j - \{1 \dots n\}$ the sequence number of the channel maintenance; $m = n + 1$; n - number of channels.

Let's analyze the operation of control, using the graphing function of flow distribution of applications for different values of φ (the probability of occupation of a channel, we use the

boundary values $\varphi = 0$, $\varphi = 0,5$ and $\varphi = 1$). Baseline data: $j = 1 \dots 20$; $n = 20$; $m = 21$; $\varphi = 0$, $\varphi = 0,5$ or $\varphi = 1$; $\lambda = 50$.

The distribution of the flow of requests between service channels in graphical form shown in Figure 3.

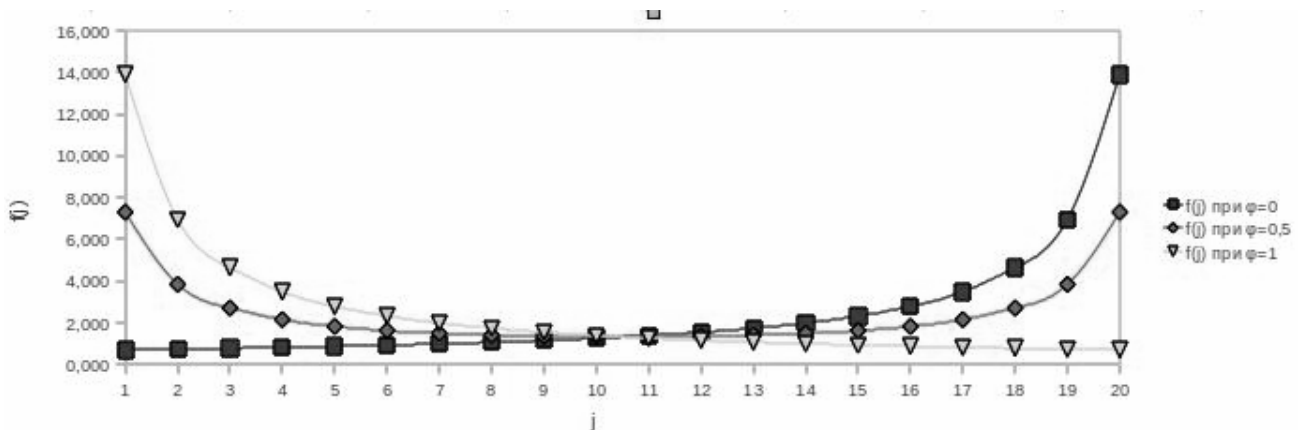


Fig. 3 - The distribution of the flow of requests between service channels

The results of calculations of the basic system parameters:

- Load factor of the system = 0,125;
- The average service time of the application = 0.05;
- Probability of no queue = 0.99;
- Average number of requests in the queue for service = 0,013;
- Average number of requests that are in the system = 2.5;
- The mean residence time of the application in the system = 0.063;
- The mean residence time of the application in the queue = 0.00025;
- The absolute capacity of the system = 47.

All these values are independent of the channel service rate, since the calculated from the average productivities of channels. Consequently, these features give a general idea about the parameters of the control system.

The study can establish that the use of the resulting model of a control system efficiency is high and almost all of the application (99%) will be served by the system.

References

1. Wentzel E.S. Probability theory / ES Wentzel. Moscow: State Publishing House of Physical-Mathematical Literature, 1962. - 564.
2. Wentzel E.S. The tasks and exercises in probability theory / ES Wentzell, LA Ovcharov. Moscow: Higher School, 2000. - 366 with.
3. Cox, DR Queuing theory / DR Cox, WL Smith, Lane. from English. Springer-Verlag, 1966 - 218 with.
4. A. Kofman queuing. Theory and applications / A. Kaufman, R. Kryuon. Springer-Verlag, 1065. - 302.
5. Olzoeva S. Modelling and calculation of distributed information systems / SI Olzoeva. Ulan-Ude: Publishing VSGTU, 2004. - 67.
6. Careful E.V. Mathematical methods for modeling economic systems: A Handbook. Manual / EV Gentle, V. Gentle. 2nd ed. Rev. and add. Moscow: Finances and Statistics, 2006. - 432.

TEACHING OBJECT-ORIENTED ANALYSIS AND DESIGN THROUGH COMPUTER GAMES

Elvira-Maria Arvanitou, Alexander Technological Institute of Thessaloniki, Greece, Thessaloniki
Apostolos Ampatzoglou, Software Engineer, AynSoft, Greece, Thessaloniki
Ignatios Deligiannis, Ph.D., Associate Professor, Alexander Technological Institute of Thessaloniki, Greece, Thessaloniki, ignatios@it.teithe.gr

Abstract. During the last years there is a trend of introducing modern technologies, such as game based learning, in the educational process. This paper describes the analysis and design phase of an educational game, namely “The Game of Sweng”, which aims at teaching students how to analyze and design software products using the object-oriented paradigm. The game was analyzed and design during the BSc thesis of the first author and is currently under development by a Greek software development company.

Introduction

Computer games have been used for educational purposes, since they were introduced, in the late 1960s [17]. In [9 and 13] the authors argue that games are a fundamental part of the evolving human experience and the way we learn, providing the opportunity to practice and explore in a safe environment. Advocates of computer game-based learning argue that computer games have the potential to transform the way that students learn, motivate and engage a new generation of learners in a way that traditional education does not [16]. On the other hand, Dondi and Prensky [5 and 12] oppose against using computer games in didactic approaches, because present educational software is not considered of high quality.

The software development process is a very complicated task, whose management requires various skills. The development process in its simpler form, i.e. Waterfall model, includes five steps, (a) Requirements, (b) Design, (c) Implementation, (d) Verification and (e) Maintenance. In a typical software engineering company, many employees are working in parallel in order to achieve the close deadlines and expected products’ time to market. In order for the communication of the employees to be feasible, a variety of software design documents needs to be created. One of the most well-known formalization methods is the Unified Modeling Language (UML) which graphically depicts customer requirements and the software architectural design. The above mentioned procedure is the learning outcome of various modules, such as software engineering, information systems, requirements engineering, object-oriented analysis, etc.

In this paper we present the analysis and design phase of a game called “The Game of Sweng”. This game has been documented in the BSc thesis of the first author, and is currently under development by a Greek software engineering company¹. The evaluation of the learning objectives of the system will take place in the next six months in two undergraduate courses of Greek Universities. In section 2 we present some background on game-based learning and on other attempts of teaching software engineering through computer games. Additionally, section 3 describes the game and documents of a planned forthcoming controlled experiment. Finally, in section 4 we present some conclusions of our research.

1. Background Information

This section of the paper presents a literature review on the effects of game based learning. According to previous work, game based learning provides various benefits concerning students’ education, character and socialization. In [4], the authors suggest that computer games strengthen children self-confidence and socialization, in the sense that game playing often includes sentimental missions that require mutual respect among peers. In [6], the authors attempt to introduce a model that predicts user enjoyment in educational games.

¹ <http://www.aynsoft.gr>

In [16], the motivation for learning that computer games offer to children is being discussed. During the study, the authors have contacted regular gamers and non-gamers, through interviews, so as to investigate the research goals. The results suggested that computer games are not motivational for all individuals, but the perception on game based learning, under certain circumstances, has been positive. Moreover, in [14], the authors performed an experiment on a sample of over one thousand students. The main aim of the research has been to evaluate the influence of educational videogames with respect to learning and motivation. The teacher reports and student observations confirmed an improvement in the motivation for learning and a positive attitude towards the technology because of the game. More specifically, high levels of attention and concentration, self-esteem increase, growth of technological faculties and enhanced collaboration between students have been observed. Thus, the authors propose the incorporation of computer games in educational plans. Additionally, in [1], the authors propose a card game which simulates the software engineering process. Furthermore, in [3] uses games development in order to increase the interest of students in software development tasks. Finally, [7 and 11] use games examples in teaching design patterns for similar reasons.

2. Game Description

In this section we describe the proposed educational game. *The Game of Sweng* is an educational online game that simulates the complete development process, in order to enhance its users' knowledge on object-oriented analysis, design, implementation, testing and maintenance. In section 3.1 we textually describe the game. In section 3.2 we provide selected analysis features, such as use case diagrams and use case descriptions. Finally, section 3.3 describes the methodology that will be used during the evaluation of the system.

2.1 Textual Description

The game supposes that the student is the manager of a software engineering company. The starting budget of the company will be 50K €. The project manager will inspect several available projects that his/her company can select from. Each project is accompanied by an extended textual description and the amount of money that the company will get if the project is successfully delivered to market. The company can work on only one project at the same time. However, if the manager aborts or fails the project during the game period (usually a semester of 4 months), he/she can select from the remaining projects with his remaining budget.

The first action that the manager will do is to select the personnel of the company. Every candidate employee will be marked in four tasks (a) analysis, (b) design, (c) implementation, and (d) testing. Additionally, the personality of the employee will be characterized according to MBTI [10]. The frequency of available personality types in the database is similar to the one described in [2]. Furthermore, the effect of personality on each development stage has been simulated according to [2, 8 and 15]. Finally, every employee is accompanied by a specified hourly wage and overtime wage.

Next, the manager has to create the time schedule for the software development. More specifically, the manager will be asked to create an extended GANTT chart that will provide references to predefined development milestones. These milestones are:

- Creating the use case diagram
- Creating the use case descriptions
- Creating system sequence diagrams
- Creating domain model
- Creating collaboration diagrams
- Creating sequence diagrams
- Creating class diagram
- Implementing code
- Testing code

In the abovementioned GANTT chart the manager will have to assign tasks to his/her staff. After completing time scheduling, the game proceeds in simulating the project development.

Table 1. “Create Professor” UC Description

Use Case ID	UC-A17		
Primary Actor	Administrator		
Secondary Actor(s)			
Brief Description	Create professor from administrator.		
Preconditions	The educational institution has to be registered in the system which belongs to the teacher.		
Flow of Events		Actor Input	System Response
	1	The administrator asks to create a new professor.	
	2		The system presents the form “Create Professor”.
	3	The administrator inserts the professor’s data into the system.	
	4		The data is correct. The system checks if the professor is registered.
	5		The professor is not registered. The system creates and shows the username, the password, the license expiration date and the email of professor.
	5		The system asks for the number of students who attending the course.
	6	The administrator inserts the number of students and one file with data from registered students.	
	7		The system creates and presents into a form all the usernames, passwords from students.
8		The system sends email for the data, called the UC: “Send Email”	
Alternative Flow:			
4a		Actor Input	System Response
	1		The data is wrong. The administrator returns to the step “2” and asks to professor again his data.
5a		Actor Input	System Response
	1		The system presents the message: “He is already registered”
	2		The administrator returns to the step “2” and asks to professor again his data.

Table 2. “Creating Sequence Diagram from Company Employees” UC Description

Use Case ID	UC-A11		
Secondary Actor(s)	System Production and Diagrams Evaluation		
Brief Description	The employees create Sequence Diagram to see the students.		
Preconditions	The professor has completed the UC “Send new grade” for Conceptual Model.		
Flow of Events		Actor Input	System Response
	1	The system asks to employees to create Sequence Diagram.	
	2		The system calls the UC: “Evaluation Diagram”.
	3		The system returns to student the Sequence Diagram which creates in step “2”.

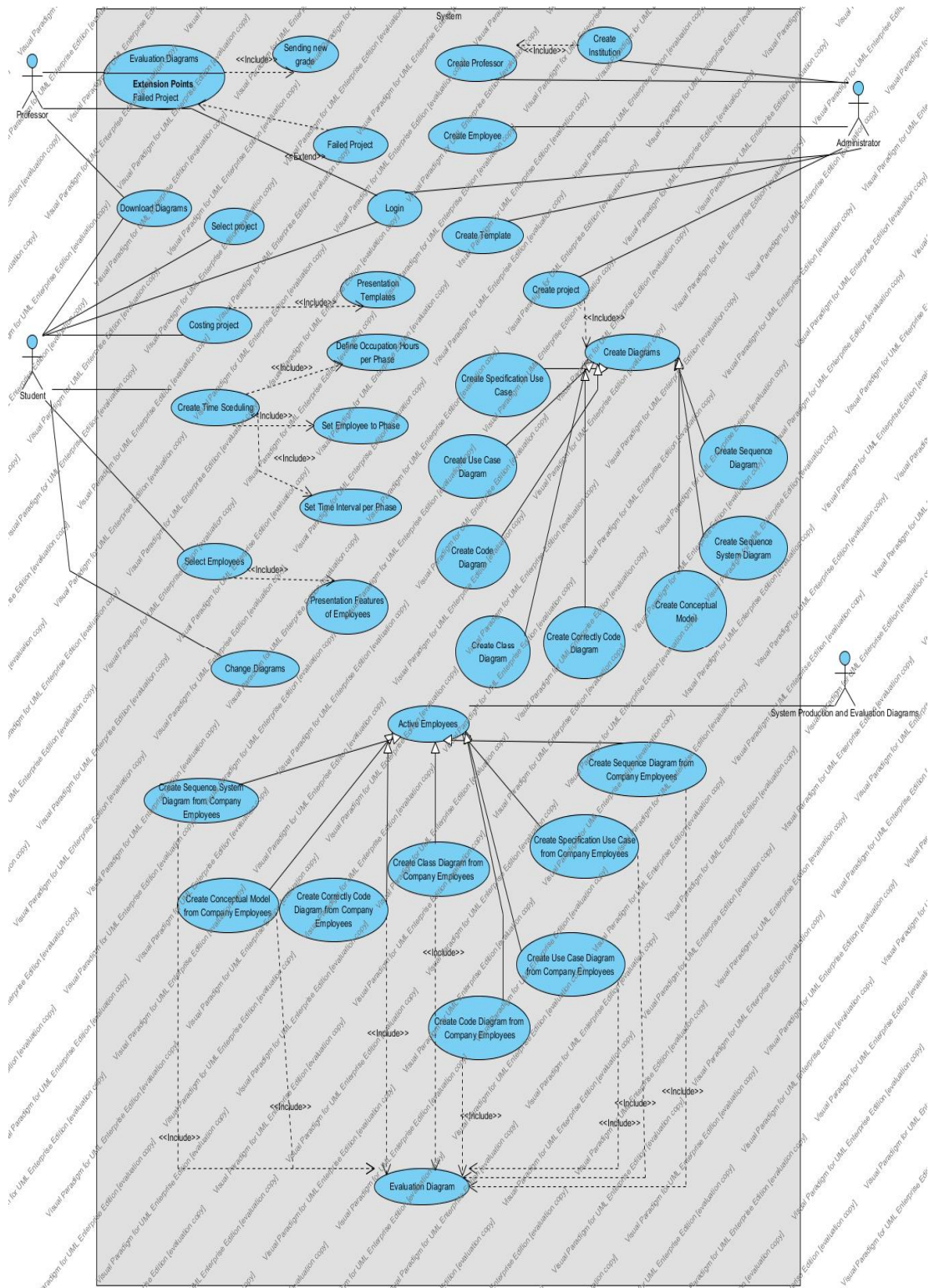


Fig. 1 – System Use Case Diagram

The procedure for handling each milestone is similar. Suppose that the project manager has assigned analysis tasks to three employees, and that he/she sets the “Create Use Case Diagram Milestone” in the end of the 2nd week of the development process. The system will evaluate the productivity of the analysis team, the collaboration among team members, the intensity of their work, and the time interval that was assigned to this task, and provide an evaluation mark for this stage. The corresponding evaluation mark will be assigned to a Use Case Diagram that will be sent to the project manager, on behalf of the development team. The project manager will correct the Use Case Diagram and resubmit it to the system. The system will automatically evaluate the changes. The mark on the previous milestone is taken under consideration on the evaluation of the

current activity. The abovementioned procedure is replicated until the project fails, or until the successful completion of the project.

The learning outcome of the game is expected to be the familiarization of the student with project scheduling, resource handling, UML diagrams, analysis techniques, design techniques, implementation techniques and testing techniques.

2.2 System Description

In this section we provide the use case diagram of the system and the description of two use cases, namely (a) Creating Sequence Diagram (Table 2) from Company Employees and (b) Create Professor (Table 1).

2.3 Empirical Plan

In order to evaluate the proposed software, we plan to use *The Game of Sweng* in two Greek universities during the next semester. In order to validate the learning outcomes of the game, we will perform the following procedure:

- Give all students a software engineering test
- Divide the students into balanced groups, with respect to their test scores. The outcome of this procedure will be the creation of two groups, NGG (non game group), and GG (game group)
- Give electronic material to NGG students and usernames and passwords to GG students.
- After a four month period, all students will be given a software engineering test.
- Students will be asked to evaluate their satisfaction from the procedure and the time they spent on it.
- Statistically analyze the results

Conclusions

This paper is a work in progress research that demonstrates the use of an online educational game called "*The game of sweng*", that aims at teaching software project management, object-oriented analysis, object-oriented design and object-oriented development to undergraduate IT students. The game is currently under development while is planned to be evaluated during the next six months. In the paper we present the rationale of the game and several requirements specifications and design documents that have been produced during its development. Up to now, many studies suggest that computer games can be used in the educational process in order to increase student satisfaction and learning interest. On the other hand, up to now, there is a lack on educational online computer games that could be taught in software engineering sources. Consequently, we believe that the proposed game will help academics who are teaching object-oriented software development methods.

References

1. Baker, E. Oh Navarro, and A. van der Hoek, "An experimental card game for teaching software engineering processes". *Journal of Systems and Software*, Elsevier, 75(1-2), pages 3-16, February 2005
2. L.F. Capretz, "Personality Types in Software Engineering," *Int'l J. Human-Computer Studies*, 58 (2), pages 207–214, 2003.
3. K. Claypool and M. Claypool, "Teaching software engineering through game design", *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '05)*, ACM, pages 123-127, 2005.
4. H. Cunningham, "Gender and computer games", *Media Education Journal*, 17, pages. 13–15, 1994.
5. Dondi and M. Moretti, "A methodological proposal for learning games selection and quality assessment", *British Journal of Educational Technology*, 38 (3), pages 502–512, 2007.
6. F.L. Fu, R.C. Su, S.C. Yu, "EGame Flow: A scale to measure learners' enjoyment of e-learning games", *Computers & Education*, Elsevier, 52 (1), pages 101-112. January 2009.
7. P. Gestwicki and F. S. Sun, "Teaching Design Patterns Through Computer Game Development", *Journal of Education Resources Computing*, 8 (1), March 2008).

8. Narasimhaiah and Y. W. Lam, "Who should work with whom? Building effective software project teams", *Communications of the ACM*, 47 (6), pages 79-82, June 2004.
9. R. Koster, "A Theory of Fun for Game Design. Scottsdale", *Paraglyph Press*, Arizona, 2005.
10. Myers, "The Myers-Briggs type indicator", *Consulting Psychologists Press*, Palo Alto, California, 1962.
11. Z.. Nguyen, S.B. Wong, "Design Patterns for Games", *Proceedings of the 33rd SIGCSE technical symposium on Computer science education*, Cincinnati, Kentucky, pages 126 – 130, 2002.
12. M. Prensky, "Types of learning and possible game styles, digital game-based learning", *McGraw-Hill*, USA, 2001.
13. Rieber, "Seriously considering play: designing interactive learning environments based on the blending of microworlds, simulations and games", *Education and Training Resource & Development*, 44, pages 42–58, June 1996.
14. R. Rosas, M. Nussbaum, P. Cumsille, V. Marianov, M. Correa and P. Flores et al., "Beyond Nintendo: design and assessment of educational video games for 1st and 2nd grade students", *Computers & Education*, Elsevier, 40(1), pages 71–94, 2003.
15. P. Sfetsos, I. Stamelos, L. Angelis and I. Deligiannis, "An experimental investigation of personality types impact on pair effectiveness in pair programming", *Empirical Software Engineering, Computer Science*, Springer Netherlands, 14, pages 187-226, 2009.
16. Whitton, "Motivation and computer game based learning", *Proceedings of 27th Australasian Society for Computers in Learning in Tertiary Education (ASCILITE 2007)*, pages 1063-1067, 2-5 December 2007, Singapore.
17. J. Wolfe and D. Crookall, "Developing a scientific knowledge of simulation/gaming", *Simulation and Gaming*, 29 (1), pages 7–19, March 1998.

THE IMPLEMENTATION OF THE FRAMEWORK TO CREATE AND MANAGE EXCHANGE TRADING ROBOTS

Pavel P. Oleynik, Ph.D., System Architect, Aston, Russia, Rostov-on-Don, xsl@list.ru

Introduction

Currently, there are many software implementations that automate activities of any application domain. No exception is exchange trading. Now there are many different software products to simplify the process of adding/moving/deleting orders in the trading system.

Many of them allow you to develop your own trading strategies based on different algorithms. Such programs are called "exchange robots" to the advantages of which are:

1. The robot, as opposed to a man is not prone to stress and therefore will follow the implemented algorithm, regardless of the current market situation.
2. Trader does not want total control over software application. That is, there is no need to control constantly by computer and monitor the progress of trade.

Despite the presence of the above advantages, there is a serious drawback: writing an algorithm may contain errors that can lead to financial losses. Currently, however, there are many methodologies and tools to minimize the number of errors in the application.

1. Optimal criteria for developing framework

The development of any software starts with the formation of the tasks that must be solved by it. Therefore it is necessary to form a number of requirements (optimality criterion), which will meet complete solution:

1. The system must receive data from the various libraries provided by brokers or exchanges. Since each exchange, and many brokers provide the library, allowing to obtain the data, it is unacceptable to have a rigid adherence to any external interfaces or classes. Instead, the need to write a number of support classes in the system, to a more unified approach using third-party libraries.
2. Active use of multithreading. Modern processors are multi-core and support a range of technologies to ensure the parallel execution of multiple pieces of code simultaneously. That's why it makes sense in the development of the platform to include the selection of individual threads within which to perform operations such as loading data from the exchange/broker as well as implementing the various steps of the exchange robots.
3. Need to implement advanced graphical user interface. The presence of graphic forms that display information in a user-friendly way is the basic requirement for any stock exchange software. This is especially true with respect to the described system, because the trader has to monitor the current situation and the process of executing of the robot and in the event of unforeseen situation (for example, usually in the early stages of debugging algorithm) in time to stop the trade.
4. The possibility of implementing a number of different exchange robots in a single framework. This implies the existence of a single hierarchy of base classes, which are used for creating robots. That is, classes that implement financial strategies of exchange robots will be inherited from existing base classes.

2. The structural diagram of a typical implementation

In Fig. 1 shows a structural diagram of the implementation of the framework to create and manage exchange trading robots. Consider the picture in more detail. Currently, there is the most popular relational database management system (RDBMS) [1]. Therefore, developed implementation uses a database of this class, namely Microsoft SQL Server 2008 R2. In the development of many software products object-oriented programming languages are used so it is one of these used to implement the system. In our case we used the programming language C # 4.0, and the development itself was performed in an integrated environment Microsoft Visual Studio

2010 [3]. Since the relational model operates on the concept of "relation" and the C # language concepts of "class", then there is an object-relational impedance mismatch, which was decided by the use of patterns object-relational mapping, which were implemented in their respective libraries [4-5].

The developed system is closely integrated with the software supplied by the stock exchange/broker. It uses a hierarchy of classes, you can load data from multiple streams (quotes, orders, deals, etc.) in the objects of the domain classes. Thus the association implemented is between different objects and collections of relevant (for example, the association between financial instruments and orders for it).

Exchange robots in this implementation are objects (instances of classes) that subscribe to event handlers that are declared in the classes to load the data. That is, each step of the financing strategy is executed after new data from software libraries exchange/broker. The result is the creation of domain objects and sending commands to the library of the exchange of software for creating/moving/deleting of orders for purchase/sale contracts trading instrument.

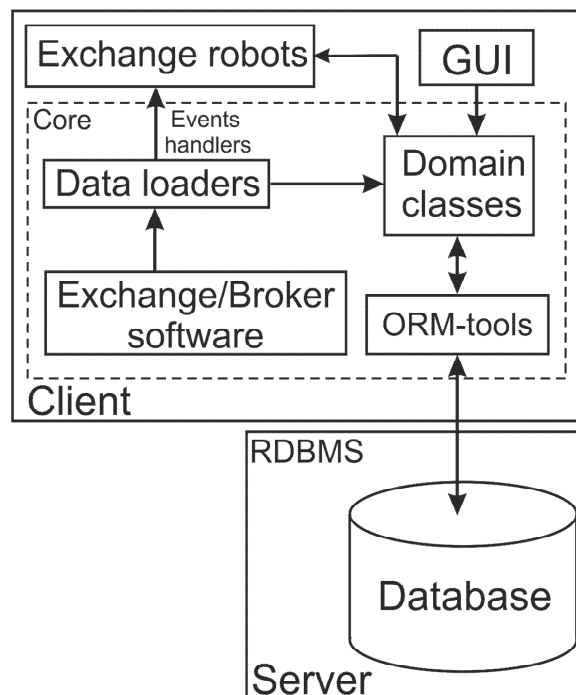


Fig. 1 - Structural diagram of implementation of the platform to create and manage exchange trading robot

In the development of active use multithreading. At the same time operations such as retrieving data from libraries broker and financial strategy execution steps that form the basis of the algorithm the robot performed in a separate child threads. In the main thread running refreshing a graphical user interface that allows traders to control the trading process. This feature is related to the implementation of the principles of graphic library on Windows in general and at the platform, .NET Framework in particular [3].

From the description it is clear that the architecture of the proposed solutions presented in the structural diagram (Fig. 1) meets the criteria of optimality, selected earlier.

3. The basic domain classes

Consider the basic domain classes, shown in Fig. 2. The base functionality common to all classes implemented as methods declared in the class BaseTradingObject, which is the root of the own hierarchy. This general approach used in designing and implementing software for object-oriented programming language that supports only single inheritance (for example, in C #) [3]. It should be noted that exchange software transmits data in the form of streams, which are structurally similar to flat tables. At the same time for implementations references uses an approach similar to that used in relational theory. That is, applied fields, which are primary and foreign keys of relations [1]. Therefore, when loading data reference organization, loads all transmitted codes, which

declared in abstract base class CodedObject from which inherited many others. Many of the data are relevant only for a specific period of time, just for themselves is declared RefreshableCodedObject class which has a property that allows you to store the date and time. Many of the objects must have a reference to trading instrument. For example, an order for purchase/sale, deal, position, and deep of market rows are relevant only for a particular instrument. It is for these classes of base abstract class introduced TradingInstrumentObject, and derived from it to implement a class are listed objects.

For objects that are named, introduced the class NamedObject. If in addition you must specify the current date, then it must be derived from RefreshableNamedObject, and if additional states and the full name (eg, Futures), then declare it derives from RefreshableFullNamedObject. All financial instruments are derived from common abstract parent FinancialInstrument, in this case may be those which are executed trades (eg, Derivatives) and those for which tenders are not executed (for example, Trade indexes). To declare a first common parent class used TradingInstrument, and root for the second abstract class stands NonTradingInstrument.

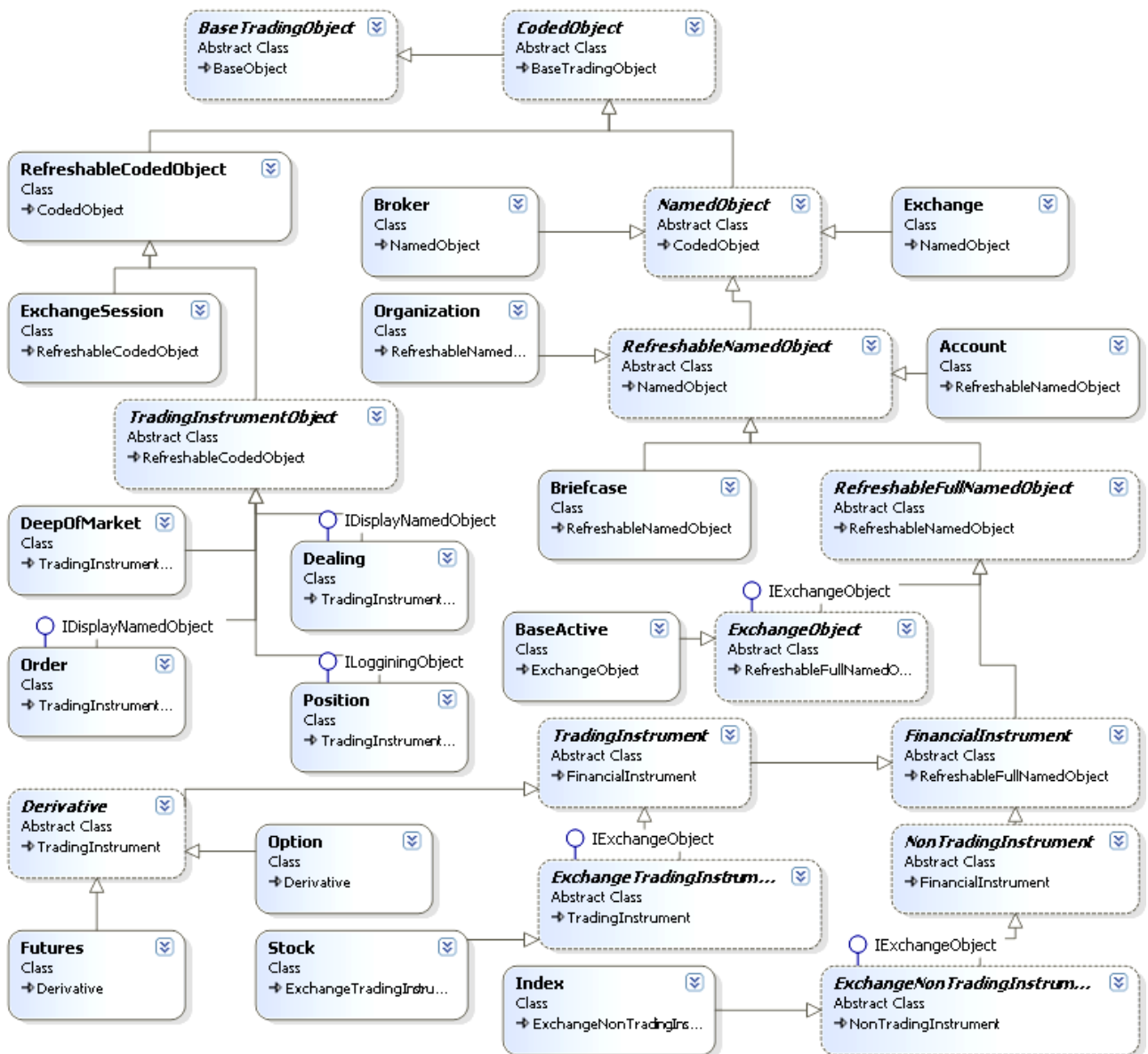


Fig. 2 - The domain classes

From the presented above shows that the declared base class is sufficient to describe any object that can be obtained from streams, transmitted exchange/brokerage software. Consequently, the developed platform, you can create practically any exchange trading robot.

Conclusions

Currently for implemented platform graphical forms application are developing that will display data in a convenient form for the trader, as well as the creation of methods and techniques to filter the data and transition from one object to other referencing objects.

The article highlighted the criteria of optimality, which requires a modern platform for creating and managing exchange robots. A structural diagram of a typical solution is implemented, as well as detailed basic domain classes that have been declared for implementing the system.

References

1. Date C.J. An Introduction to Database Systems, 7 Sub edition, Addison Wesley Longman, 2000, 938 p.
2. Mistry R., Misner S. Introducing Microsoft SQL Server 2008 R2, Microsoft Press, Redmond, Washington, 2010, 216 p.
3. Troelsen A. Pro C# 2010 and the .NET 4 Platform, 5 edition, Apress, 2010, 1752 p.
4. Fowler M. Patterns of Enterprise Application Architecture, Addison-Wesley Professional, 2002, 560 p.
5. Ambler S.W. Agile Database Techniques—Effective Strategies for the Agile Software, John Wiley & Sons, 2003, 373p.

Printed 10.12.2011

Makeup – **Edward G. Galiaskarov**

Cover design – **Pavel P. Oleynik**

Press-correctors – **Valery V. Laptev, Ilya E. Ermakov**

Responsible for edition – **Edward G. Galiaskarov**

Signed in print 05.12.2011. A4 size

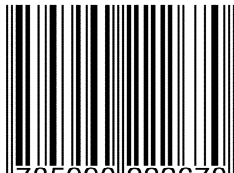
Offset paper. Printing digital

Con.prin.sh. **6,8**

Circulation 300 copies

Order № 735

ISBN 978-5-9902226-7-0



9 785990 222670