ISTANBUL UNIVERSITY ENGINEERING FACULTY
JOURNAL OF ELECTRICAL & ELECTRONICS

YEAR : 2001 (193-200)
VOLUME : 1
NUMBER : 2

DESIGNING AN OBJECT-ORIENTED APPLICATION MODEL BY BOOCH METHODOLOGY

BOOCH METODOLOJÝSÝYLE NESNEYE YÖNELÝK BÝR UYGULAMA MODELÝNÝN TASARLANMASI

Zerrin AYVAZ REÝS [1] **Mithat UYSAL** [2]

[1] University Of Istanbul, Educational Faculty Of Hasan Ali Yucel, Beyazit
[2]University Of Mimar Sinan, Department Of Enformatics, Findikli

[1]e-posta: ayvazzer@istanbul.edu.tr [2]e-posta:muysal@msu.edu.tr

ABSTRACT

Basically, good structures have object-oriented tendency which does not mean all object-oriented structures are good; on the other hand it does not mean that only object-oriented structures are good. An application based on the principles of object-oriented selection may provide the appearance of structures containing the characteristics of a complex system which has been organized. Sufficient software structures have many common characteristics: They are formed from abstracted levels which have been identified sufficiently. Every level having coherental & complete abstraction are possible only with well-identified and controlled interfaces. These levels are constructed over the lower abstraction levels which contain well-identified and controlled charecteristics.

There are many techniques to develop software for the object, one of which is Booch technique. Booch is the technical definition for the object to improve the design and analyse a system. There are different suggestions for Micro and Macro applications. Here, what we aim is to identify the steps to put a succesful project into practice by means of Booch methodology and to present it.

Keywords: *Object-Oriented Methodology, Object-Oriented design, Booch methodology, modelling, software engineering.*

ÖZET

Model, bir þeyin kurulmadan önce anlaþýlması amacýyla soyutlanmasýdýr. Modellere tüm ayrýntýlar dahil edilmediði için üzerinde deðiþiklik yapýlmasý orjinalinden daha kolaydýr. Çalýþanlar gereksinim duydukça bir tasarýmý gerçekleþtirmeden önce modeller yapmýþlardýr. Donaným ve yazýlým sistemlerinin geliþtirilmesi de farklý deðildir. Karmaþýk sistemler kurmak için, sistemin farklý yönleri soyutlanmalý, kesin bir gösterim kullanarak model oluþturulmalý, modelin sistemin gereklerini yerine getirdiði doðrulanmalý ve yavaþyavaþdetaylarý ekleyerek modelden uygulamaya geçiþyapýlmalýdýr.

Temel olarak, iyi yapýlar nesne-tabanlý olma eðilimindedir. Bu, sadece yada bütün nesne-tabanlý yapýlar iyidir demek deðildir. Nesne-tabanlý ayrýþtýrma prensiplerine dayalý bir uygulamanýn, *organize edilmiþ karmaþýk bir sistemin istenen özelliklerini içeren yapýlarýn ortaya çýkmasýný saðlayacaðý söylenebilir.*

Ýyi yazýlým yapýlarý genel olarak iyi tanýmlanmýþ soyutlama düzeylerinden oluþmuþtur. Bütünlük taþýyan ve uyumlu bir soyutlamayý ifade eden her düzey, iyi tanýmlanmýþ ve kontollü ara yüzlerle saðlanýr ve ayný þekilde iyi tanýmlanmýþ ve kontrollü özellikler içeren daha alt soyutlama seviyeleri üzerine inºa edilir.

Nesneye yönelik yazýlým geliþtirilmesi konusunda pek çok yöntem vardýr, bunlardan biri de Booch yöntemidir. Booch, bir sistemin analizi, tasarýmý ve geliþtirilmesi için nesneye yönelik yinelemeli bir teknik tanýmlamaktadýr. Bu konuda Mikro ve Makro uygulamalar için ayrý önerileri vardýr. Bu çalýþmada Booch metodolojisi ile baþarýlý bir projeyi ortaya koyabilmek için gereken adýmlarý belirtip, bir mikro uygulama ile bunu sunmayý amaçladýk.

Anahtar sözcükler: *Object-Oriented Metodoloji, Object-Oriented tasarým, Booch methodolojisi, modelleme yazýlým mühendisliði.*

1. INTRODUCTION

Developing a high quality software system is driven by a software development methodology. Different methods emphasize different aspects of the software development process. A software system consists of functions and data. Traditional methods are function oriented and divide the functions and the data. This approach causes some bottlenecks during maintenance. However object oriented methods integrates these two elements of software, encapsulates them and process them together.

2. MATERIAL AND METHOD

2.1. Modelling As A Design

Model is an abstraction of something before it is installed or put into practice. Since all details are not included it is easier for the model to be changed than the original itself. Those, who have had the need in a working environment, have formed models before they have developed the original design. The improvement of software&hardware systems are not different at all. To establish complex systems, differences in the system should be abstracted, a model should be made using a definite display, the fact that the model realizes the needs of the system should be confirmed, and by adding details gradually the transfer from the model to application must be done.

2.1.1.Modelling

Designers produce various models before they construct their product for various purposes. For example; architectural designs for the customers, aircraft models having different scales for speed tests, charcoal drawing for paintings, designs for machine pieces; designs for advertisements and books. They have different purposes.

Testing before constructing it physically:
Since ancient times people have formed models to see the result of what he is doing in this field. Recent developments in computer technology have enabled many physical structures to be simulated in computer before they are realized.

Simulation is not only cheaper, but provides information , which can not be reached and has short-life-span comparing to a real model. Both physical models and computer models are cheaper than constructing all the system and also enables to find mistakes and correct them.

Communicating with customers:
Architects and product designers form models for their customers. Copies are introductory products copying all or partial characteristics of a system.

Visualization:
Films, T.V. shows and commercials enable the producers to see their ideas. Before a detailed software begins irrelevant pieces, transition which is not appropriate and the edges which are not connected can be changed. Charcoal drawings are for changing their ideas before they transfer it on to stone or a frame.

Reducing complexity:
The reason for modelling is the necessity of working with complex systems which are very difficult to understand directly human brain have the capacity of fighting and overcoming which limit information in a limited time. Models reduce the complexity separating the necessary limited thing for a moment.

2.1.1.1. Abstraction

Abstraction is investigation of particular points of a problem. The main aim of the abstraction is seperating the important points for a particular purpose and suppress these which are not important. Abstraction must have a purpose because this purpose clarifies what is important what is not. The same thing can have different abstraction for different purposes.

All abstractions are incomplete or faulty. Words or languages used are incomplete definitions of the real world. But this does not change the situation that they are useful. The purpose of abstraction is the limitation of the universe to realize something. For this reason we must look for adequacy for the purpose not the pure reality. There is not only one model for one situation, but adequate or inadequate one.

A sufficient model holds the important facts for a problem while ignoring the others. For example; many of the computer languages are not sufficient to be used for modelling algorithms; because they need clasification for application details which are not related to algorithm. A model containing irrelevant details limits choosing model designs and removes attention from the real subject. There is not a correct way for classification of abstractions and also for constructing the structure of a definite system . While designing the solution for a structure in a field, sometimes clever methods, which have not been thought of before, can be found. In addition, how do we see the difference between a good structures and a lead one.

Basically, good structures have object-oriented tendency. This does not mean all object-oriented structures are good but only object-oriented structures are good. An application based on object-oriented separation principles enables the appearance of the structures containing the characteristics of a complex system which has been organized.

Good software structures have many common characteristics: They consist of abstraction levels which have been well-defined. Every level having coherent and complete abstraction are only provided by means of well-identified and controlled interface and constructed lower levels containing well-identified and controlled charecteristics. It is a fact that, between the interface and its application of the level without destructing the suppositions by the users.

The structure is simple: General behaviours; general abstractions and general mechanism enable this. A distinction between strategical and tactical decisions can be mentioned. A strategical decision carries more general and formal meanings and for this reason contains the organisation of the structures which have higher position. The mechanism of fault finding and correcting; user’s interface paradigm, memory administraction and object position politics, approachs of process syncronisation in application which are real-time strategical structural decisions are meant.

On the contrary tactical decision means only local structural decisions and generally contains application and interface details.

2.1.1.2. Circular And Developing Life-Cycle

Works enabling succesful object-oriented structures have not only circular but developing characteristics. This kind of work is circular that means the revision of object-oriented structure which is developing and advancing. The results of previous drive and transfering the collected experience to later analysis and design step are the basis of this revision. The work’s showing a continuous improvement in every move in analysis, design and cyclic process enable a solution; which supplies the real needs of the end-user and development of strategical and tactical decisions after controlling them, to occur. This solution is also simple, reliable and adaptable.

Cyclic and improving development process is the anti-thesis of waterfall model and its explains neither top, down or bottom-up style. According to the circumstance both are applicable.

Experiences show that object-oriented development is neither top-down nor bottom-up. Instead, as pruke suggests, complex systems which have been well-structured is best-created by using “round-trip-gestald design”. This design style underlines cyclic and improving structure for the system development. Doing this, the important thing is that the reneval and development of logical and physical appearances of the system should be perceived as a whole. Round-trip-gestald design is the basis of object-oriented design.

For some limited application fields, the problem which will be solved could have been defined with various applications in this field. Here, it is possible to code the improvement process as a whole. A new system designers in such field have already found out important abstractions, which mechanism should be used and expected move span for the system is generally known. Creation is still important for a process like this. But here, it is addressed to strategical decisions of the system. In this cases, since the risk of the improvement has already been eliminated, it is possible to reach very high production performance. The more we know about the problem that will be solved, the easier it becomes to solve it.

Industrial software problems are not always like this. They mostly require a balance of authentic set in which functional and performance needs form, and maximum performance of the developmemt team needs creativy.

All human activities basing on intelligence, experience and team work, which needs creativity require a cyclic and improving work.

2.2. Booch Methodology

This methodology approaches to software improvement process from two different points. These are ; Micro and Macro improvement process.

2.2.1. Mýcro Improvement Process

Micro processes of object-oriented improvement are defined by scenarios and structural products produced by macro processes. That means micro processes explain daily activities of an individual and a small development team. Micro processes software engineer is equal to software architect. Micro processes from the point of view of the engineer guide many decisions in the adaptation of the structure and daily product. On the other hand, from the architecture’s point of view it forms a frame for the alternative designs and the cycle of the structure.

In micro process, traditional analysis and design phases are blurred, and processes are carried out under an opportunist control (here substitution of the necessary ones) As Stroustrup has observed, there are not methods, prescriptions which will take the place of a software project.

Micro process have this tendency;

1. Identification of the classes and objects in the given abstraction level,
2. Identification of the semantics of this class and objects,
3. Identification of the relations between this class and objects,
4. Interface clarification of interface and application of these classes to objects,

2.2.2. Macro Improvement Process

Macro improvement process serves as a working frame which controls micro processes This expended process shows some activities and the number of the products then, can be measured. These activities inform the values which have been determined in case of risk to development team. (punishment & application and also informs the realization of approval of micro improvement processes at early stopes to the team) Macro improvement process shows the actvities of the whole development field in a week time scale in a month.

Many elements of macro process are experiments of simple software administration of the system which are not object-oriented and are object-oriented. These simple experiments include hardware administration, reliability of the quality, walkthrough and documentation.

Macro process, different from individual improvement team. Both are for supplying the needs of the customer by means of a quality software. Besides, end-users will have less information about the events because of polymorphic functions which have been clearly explained, and parametric classes used by those which have improved them.

For this reason, macro process is focused on risk and architectural look. These two elements,which can be administered, have the totality and quality that are the most effective points.

In macro process, analysis and traditional phases of the design are processes which have been well-classified and long-lasting in order to protect extra expansion. Macro processes are activities lined below

1) Providing the basic needs for the software (Conceptualization)
2) Developing behaviour tendency model for the system (Analysis)
3) Forming the architect for development (Design)
4) Development cycle of the details from one end to the other (Development)
5) Network admininstration for post development (Maintenance)

2.3. What Are The Development Steps For Booch Methodology

Booch Methodology is a technical definition for the analysis, design and development of a system. It has different suggestions for macro& micro applications.

Steps of macro application developments
1) Understanding the concepts- Clarifying the needs and placing them.
2) Analysis development method.
3) Creation of design architecture.
4) Adding value and realizing it.
5) Maintenance

Steps of micro developing process
1) Defining objects classes.
2) Defining the logic objects& classes.
3)Defining the relations between objects& classes.
4) Improving objects&classes.

Here we are going to illustrate the methodologies that have been investigated before. For this reason we have chosen a simple problem which is well-known in academic circle.

2.4. Case Study

Problem
A programme including grades of the students of a vocational high school having several departments, the results, after some required calculations by the system have been done, showing the success of the student,and also if desired this programme can also inform the teacher & student about the results, personal information such as military service(postponing it during university education), school fee,environment information e.t.c. When we apply the suggestions for developing and recurrent object-oriented tecnique by Booch Methodology , we see six different diagrams developed in the methodology for this reason.

(Class Diagram),

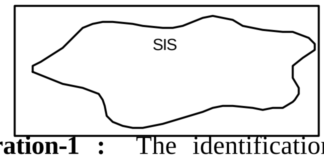
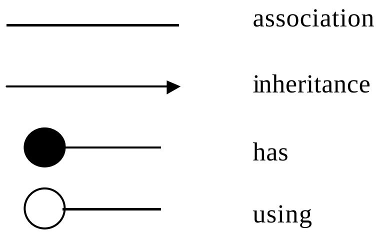


Illustration-1 : The identification symbol of Class in Booch Methodologie.

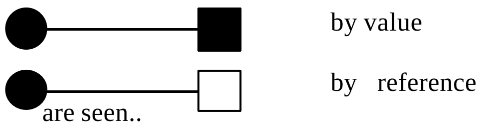
Class identification is realized with the symbol shown above in Booch Methodologie. We have already named our problem subject as SIS-Student Information System.

in the booch methodologie;
Symbol showing **class relations** are below



relationship adornments;
role
[key]
{constraint}
attributed class

containment adornments;



are seen..

Object Diagram;
The diagrams showing the relations between the classes and their definitions. If we give examples

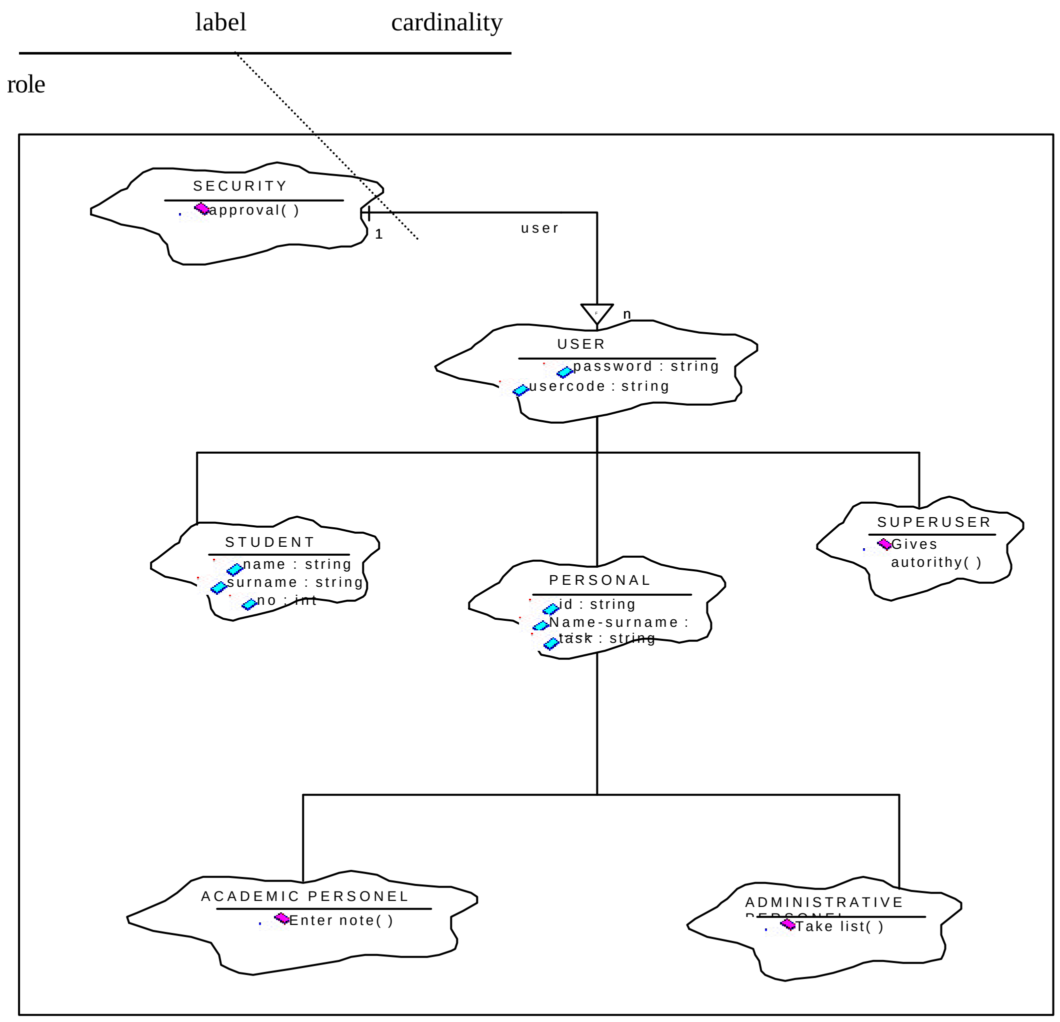


Illustration-2 : Class relation in Booch Methodologie.

SCENARIO : IDENTIFICATION of USER

PRECONDITION: User must have the responsibility to do this. (Superuser / Administrator)

* Superuser opens security system from the "programme".
* Chooses "New user identification" chart.
* Screen for new user identification opens.
* User's name-surname are coded.
* Key word is given to the user.
* The kind of responsibility.

i. Student,
ii. Academic personal,
iii. Administrative personal,
iv. Administrator approves.

* Confirms Superuser processes.

The diagram .below has been prepared for the scenario given above;

Interaction Diagram;

Diagrams showing how object classes influence each other. These classes are produced from the scenarios providing the solution according to the problem..

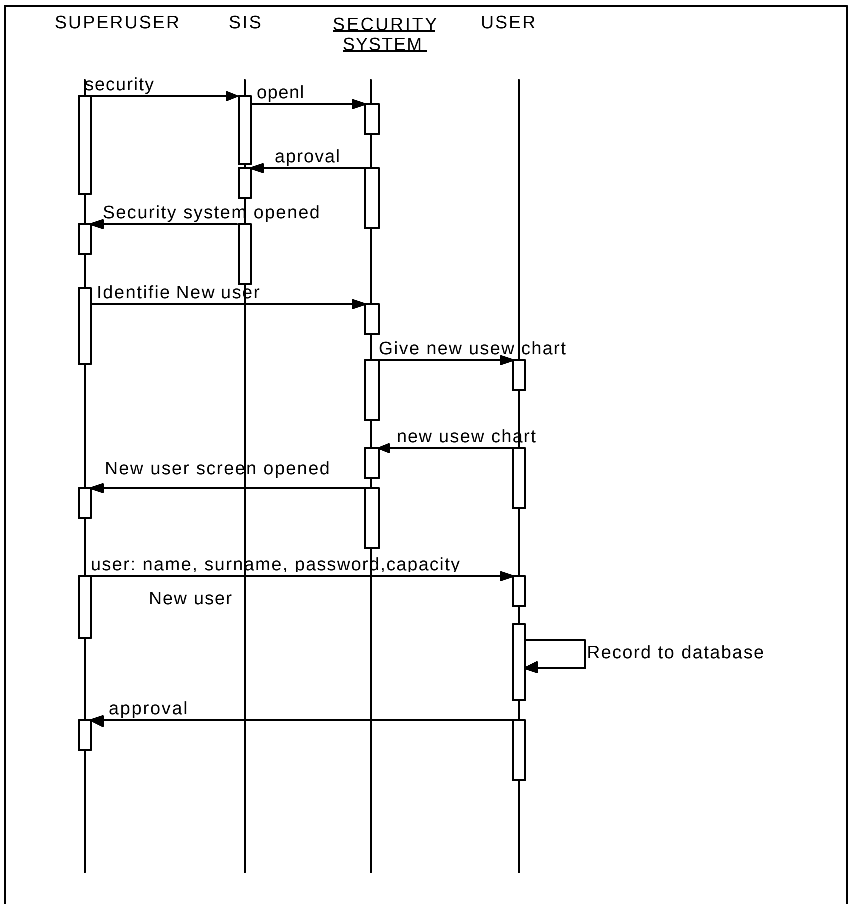


Illustration-3 : Interaction Diagram in Booch Methodologie.

Module Diagram; llustration of objects and classes of the defined system in general use with modular symbols. Symbols used are shown below.

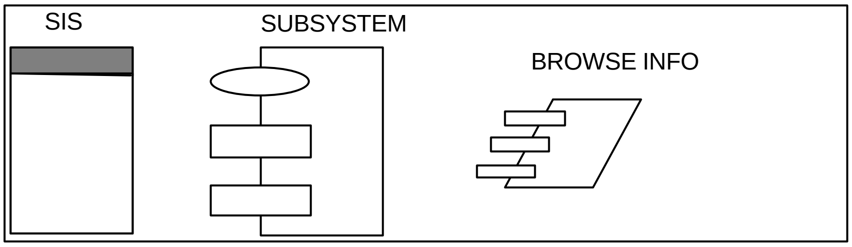


Illustration-4 : Modular Diagram in Booch Methodologie.

Process Diagram;
Process Symbols used are shown below.

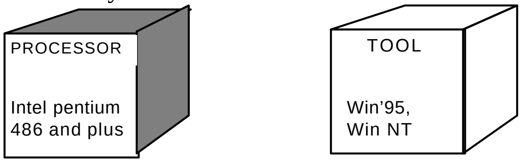


Illustration-5 : Process symbols in Booch Methodologie.

Shows placement of the process to the precessor which are physical appearences of the symbol.

State Diagram;

State diagrams are given with scenarios and they show statement transition between each other. The example for state diagram below has been given for the scenario before.

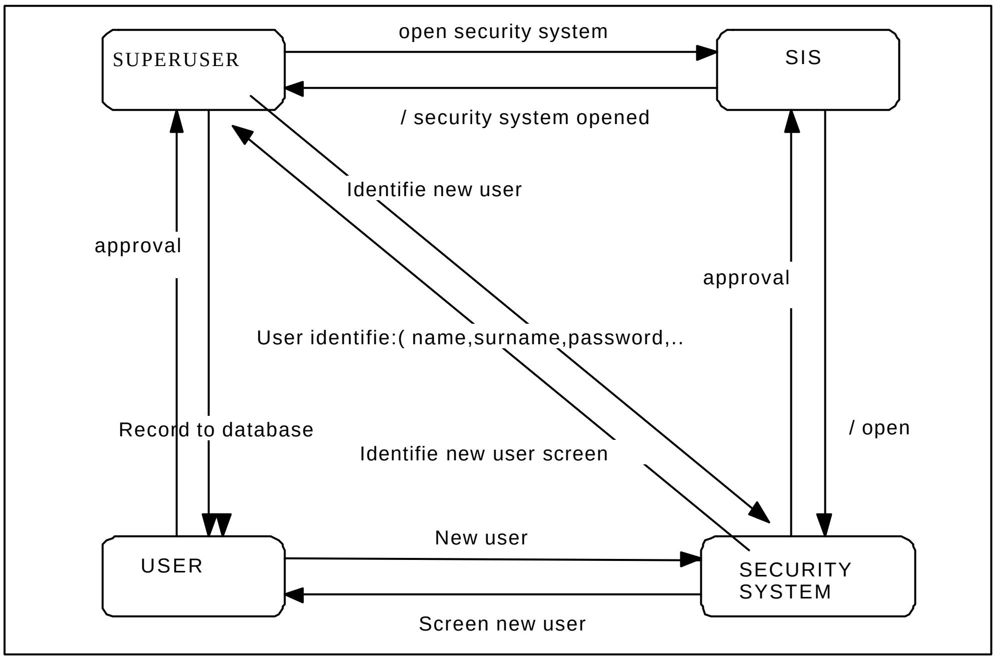


Illustration-6 : State Diagram in Booch Methodologie.

3. CONCLUSION

This case study is limited, so it can be progressed to compare different topics and developed other methodologies.

4. REFERENCES

[1] Brocks F., "The Mythical Man-Month", Addison-Wesley, 42, 1975

[2] Stroustrup B.; "The C+ Programming Language", Addison-Wesley, 362, 373, 1991

[3] Jones C, "Reuseability in Programming: A Survey of the State of The Art", IEEE Transaction on Software Engineering. Vol. SE-10(5), September-1984

[4] Humprey W.; "Managing The Software Process", Addison-Wesley, 5, 1989

[5] Parnas D., Clements P.;"A Rational Design Process; How and Why to Fake It". IEEE Transactions on Software Engineering. Vol. SE-12(2), 1986

[6] Boehm B., "A Spiral Model of Software Development and Enhancement. Software Engineering Notes", vol.11(4), .22, August 1986

[7] Booch G.;" Object-Oriented Analysis and Design", The Benjamin/Cummings, 236, 1994