ISSUES CONCERNING THE USE OF UML DIAGRAMS TO DEFINE THE UNDERLYING PROCESS MODEL SIMULATION

MIOARA UDRICĂ, TEODORA VĂTUIU, ADRIAN GHENCEA [*]

ABSTRACT: *Diagrams are a graphical representation of the information contained in a UML model, and are an essential feature of UML modelling. Each UML diagram is designed to let you view a software system from a different perspective and to varying levels of abstraction.*

KEY WORDS: *unified modelling language; model; process of simulation; diagrams; static model; dynamic model; functional model; structure; behaviour; system's cycle of life; analysis; design; implementation.*

JEL CLASSIFICATION: *D83, O32*

1. GENERAL CONSIDERATIONS

Unified Modelling Language (UML) succeeds an important wave of object oriented methods of analysis and design, used in the late 80's and early 90's. It is a unified modelling language, the result of a process of introducing standardization in object oriented design. It is the starting point in the future development of graphic languages.

Essential contributions in this field were brought by Grady Booch, Jim Rumbaugh and Ivar Jacobson. Based on their studies, on November 17, 1997, Object Management Group (OMG) announced the adoption of UML as a standard modelling language.

[*] *Ph.D., "Titu Maiorescu" University of Bucharest, Romania, udrica@gmail.com*
Assoc.Prof., Ph.D.,"Titu Maiorescu" University of Bucharest, Romania, v_teodora@yahoo.com
Lecturer, Ph.D., "Titu Maiorescu" University of Bucharest, Romania, adighencea@gmail.com

2. ADVANTAGES AND DISADVANTAGES OF UML UTILIZATION

Advantages of UML are:

▪ it is a frame for object oriented analysis, providing:
- different but complementary views of the system which guide the use of object concepts;
- Several levels of abstraction, that allows the control of the system's complexity through object solutions.

▪ it is a communication pylon
- graphic notation allows visual expression of an object solution
- formal aspect/appearance of its notation limits ambiguity
- visual features facilitate comparison and evaluation of solutions

▪ it is a formal and standardized language, which:
- gains precision/accuracy
- gains stability
- encourages the use of CASE tools

▪ ensures independence towards the implementation language and the domain of application

Disadvantages of UML are:

▪ the practice of UML language requires specialized training

▪ it allows the design of models, but it doesn't specify the process of the model's design. It is an iterative and incremental approach guided by the requirements/ needs of the system's users.

▪ it doesn't show/ not describe how the soft is developed/to develop the soft, but it can be used with any process.

The adoption of UML as a standard modeling language brings significant contributions in achieving complete, concise and intelligible models for real systems. Models contain abstractions representing the real world as a collection of entities (objects, instances) and of different connections between them. We can define constrains, by describing static, dynamic or temporal features of the entities.

The models are tested by the simulation process, viewed as a process of conducted experimentation in order to establish the level of adequacy in relation to the initial system.

Some of the shortcomings of the simulation process are:

▪ the support model of the simulation is a simplified one, following a single purpose. The solution offered is a punctual one, which does not always have a correspondent in the real system;

▪ the results can't be transferred to other problems because they are based on unique factors specific to particular problems.

▪ the results are difficult to interpret due to the fact that they are dependent of random factors. No matter how powerful the computer is, the optimal solution is difficult to obtain on a model which has many equations and a significant number of parameters.

These shortcomings have led to the use of simulation only when the interactions between components are complex, when random factors have a significant

influence and a great number of observations regarding data behavior is necessary, when the problem cannot be solved by using an algorithm or direct experiments. If we cannot apply direct optimization methods, the optimal solution is enabled by alternative experiments. We test the values of the different decision variables and highlight the consequences of some decisions regarding the values of the resulting variables.

The advantages of the simulation process:

- the model on which simulation is based provides a functional form of expression of the relationship between the phenomena studied. We can therefore test for unexpressed actions within the model;
- ensures a better structuring of the analyzed problem, allows exploration of the information flow and of operating procedures without interfering in the real system;
- uses the cybernetic control system which represents the basis of the decision-making process in practice;
- there are many software packages for the process of simulation;
- data used in the design of the model can be real observations (numerical values) or knowledge. These are translated into algorithms implemented by a computer system.

These have caused the consideration of simulation as one of the most powerful tools in the decision-making process. With the recognition of UML as a standard language for modelling, the process of simulation becomes a technique for coordinating computer-aided experiments and UML diagrams become universal visual tools for modelling elements. Each type of diagram shows a certain aspect of the modelled system: static structure, interactions between objects, physics components of an application, interactions between users and the system. Together they design a real world model, viewed from different perspectives and in different ways.

3. ROLE OF UML DIAGRAMS IN BUILDING MODELS UNDERLYING THE SIMULATION PROCESS

We present below how the UML diagrams participate in the design of models which represent the basis of the process of simulation.

3.1 Highlighting the role of UML diagrams in system structure and behaviour

The structure of the system is shown in the diagram of classes and the diagram of objects. They contain classes and relationships between them or objects and relationships between them, when the behaviour of individual objects require structural changes. In this case, the structure is further evidenced by collaboration diagrams.

For a class, **behaviour simulation** means to evidence the possible state and the events leading to transition from one state to another. This is done by using the state diagram, which can include the possible constraints emerging during the chaining state-event-state.

In most cases, state changes are determined by events occurring outside the object but to which it reacts. In case changes have an internal determination, caused by finishing a proper action of a state, to show the behaviour on define activities diagrams.

If the changes have an internal determination, resulting from the performing or ending a state's own actions, in order to represent the behaviour one needs to further define activity diagrams.

In the simulation process, UML can provide information about the collaboration between structure and behaviour, including in the same diagram activities and objects (fig.1)

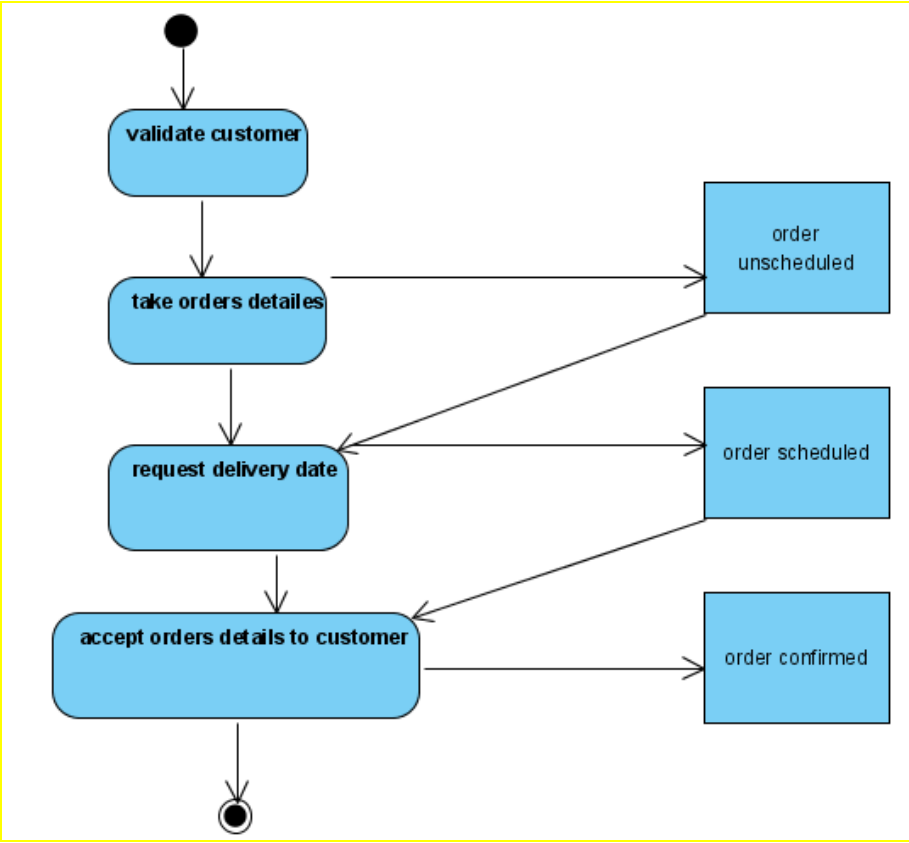

Figure 1.

For the system, the **simulation of behaviour** means showing the interactions between classes and objects. This can be realized using three types of diagrams: collaboration, sequences and activities diagrams.

In the collaboration diagram new links can occur between classes. In the sequences diagram, for every message, we have to have a correspondent operation in the destination class, as an appropriate response to the received message. While the activities diagram describes a case of usage, sequences and collaboration diagrams are

thought for scenarios of a case of usage, they show classes and their interactions during a scenario.

For classes with significant dynamic behaviour, the state diagram completes the case of usage.

3.2. Role of UML diagrams in highlighting system functionality and user requirements

The specification of the functional requirements of the system is express in cases of usage, which correspond to actions performed by a particular entity (actor) for a group of users. By showing cases of usage one can delimit the studied field and establish cooperation between users and the system's analysts. The entire functionality of the system is given by the set of usage cases, grouped in a usage case diagram.

The usage case diagram, which represents the interaction between actors and usage cases performed by them, represents a functional description of requirements, structured in relation to one or more actors. At this stage, there are no objects. Switching to an object structure is done by highlighting the objects which collaborate to obtain the functionality described by different cases of usage. In this respect, one follows different, possible scenarios, seen as instances of usage cases. One obtains in this way objects and sequences diagrams. If the algorithms are complex, one defines the state diagrams or activities diagrams.

Addressing how the system's functionality is performed using UML diagrams, many authors consider that there are two important diagrams: usage case diagram and classes' diagram. Usage case diagram expresses the requirements of the users. The system's functionality expressed by these requirements is transformed by the system's analysts into a model with classes and relations between them. The other diagrams are subordinated to the classes' diagram and use elements and tools of oriented object methodology to complete the classes' diagram:

▪ State and activities diagrams bring more details regarding the behaviour of objects within classes;

▪ Collaboration and sequences diagrams detail the interactions between classes;

▪ Activities diagrams bring elements necessary for the implementation of classes.

3.3 Role of UML diagrams in modelling static, dynamic and functional of the real system

Usage case diagrams ensure the correspondence between the user's requirements and their representation using abstracts defined and evidenced in different diagrams. These other diagrams show together a model of the real system, including static, dynamic and functional views. Grouped under this aspect, diagrams can be at a given moment a static, dynamic or functional model.

Between models there is a close connection:

- the dynamic model shows the order in which operations defined in the static model classes are made. Actions from the functional model

correspond to the operations from the static model, actors and usage cases become objects linked by functions from the static model.

- the actors are explicit objects from the objects model, which presents their structure. Data which flows from or towards the actors corresponds to operations for objects. For an actor object, the dynamic model shows when it functions. The dynamic model for these objects is necessary in order to determine the order of operation.
- the functional model reveals the meaning of the operations and of constrains from the object model and the significance of actions and activities from the dynamic model.

In all of the three models we have constrains, showing either relationships between two objects at the same moment in time, or relations between values of the same object at different moments in time. The constraints can regard objects, situation in which they shows partial or total dependence among objects, or they can regard the states from the dynamic model, or can even highlight restrictions upon operations from the functional model.

The static dimension regards the system's structure, its components and the relationships between them. It is shown with the help of classes and objects diagrams. These diagrams are modified across the system's cycle of life, leading finally to a model of the real system, completely implementable by using object oriented languages (Java, C++, Visual Basic).

The evidence of components and the relationship between them is not enough for understanding the system. In order to reveal the dynamic aspects, which depend of time, we need to define new diagrams: collaboration, state and sequences diagrams. The collaborative objects show what the object's states are and how they change and the events that cause transition in time from one state to another. Moreover, they highlight the sequences of operations which emerge as an answer to extern stimuli, without taking into account what the operation does and how it is implemented.

The dynamic dimension of the system is presented using events recognized by the system. Construction or destruction of objects, the change of the object's proprieties, updating the integrity constrains or the change occurred among the relations between objects are only a few changes generated in time by the appearance of some events.

The functional aspect refers to data flows that occur among different actors from the system. It describes what happens in the system and it is represented by the use case diagram and by activities and components diagrams.

In the situations when different department from the real system have to be emphasized in the functional model, we can allocate activities from the activity diagram in department.

For example, an activity diagram showing the process of obtaining payment (fig.2) can be used to denote some business area where activities take place (fig.3).

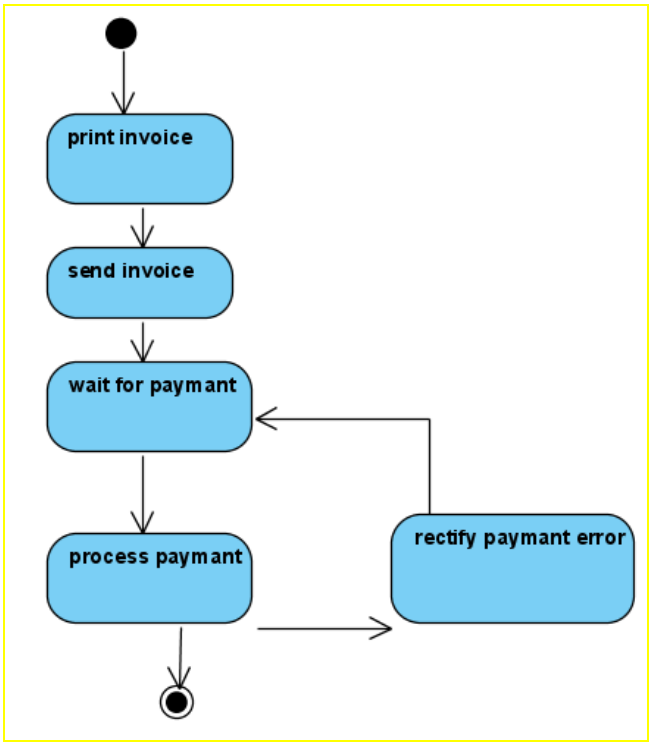

Figure 2.

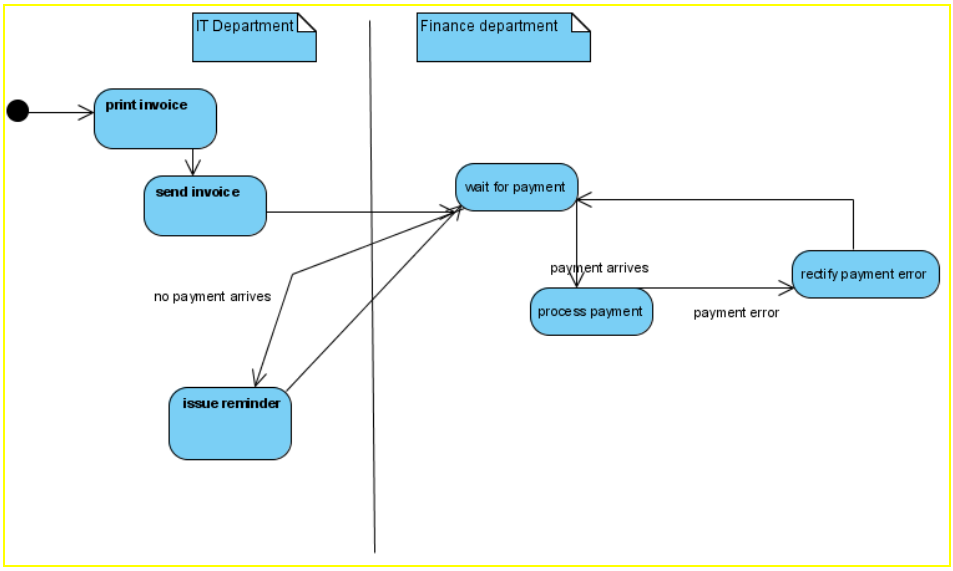

Figure 3.

In addition, highlighting the functional aspect involves the identification of input and output data, the definition of data processing procedures, the identification of constraints and the specification of the optimization criteria.

4. CONCLUSIONS

The design of an informatics system involves going through several stages. Every time, we find out the result into a correspondent model. The relationship between models remains, because we start from the elements of the domain from the real system and include details necessary for implementation during the design stage; the same elements are represented, but from different points of view.

Following the system's cycle of life means to revue, detail and complete the existing models at a moment in time. This confirms that object oriented models are developed in spiral.

Defined for new systems, or for allowing the development of existing systems, usage case diagrams are a common frame during analysis, design and implementation stage. They are the initial point in establishing the requirements of the design and they offer the basis for testing and verifying the obtained system.

The models obtained during different stages are represented by different types of diagrams, which are linked by momentary relations determined by the context. In successive stages, each model brings a different view on the system, adds new elements to the previous model, until, finally, we have a general view regarding the system.

Defining models is not a linear activity, since diagrams defined in one stage can be modified in another stage. Chaining and interaction of the models takes place throughout the definition of the system.

- functional requirements of the system with the help of the usage case diagram;
- the outcomes of the analysis stage lead to the creation of an object model and different scenarios which lead to behaviour diagrams (sequences and activities);

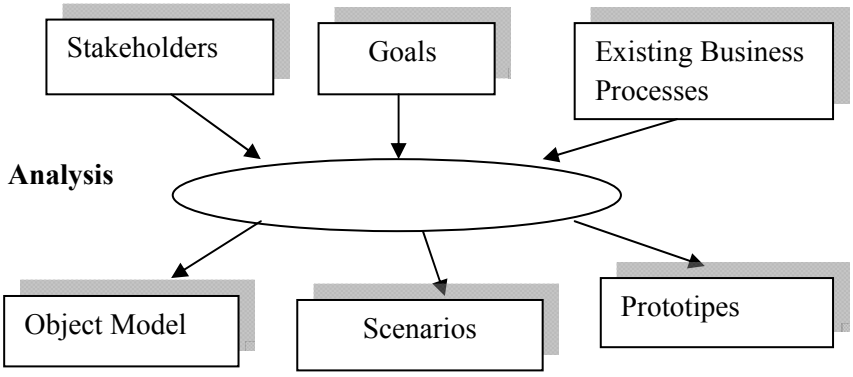

Figure 4.

During the design stage one defines new classes, eliminates the classes without relevance, or emphasizes new relations among the classes. Classes specific for different usage cases are integrated into a unique structure. One defines state diagrams, adds details necessary for the implementation into the model of classes (fig.5).

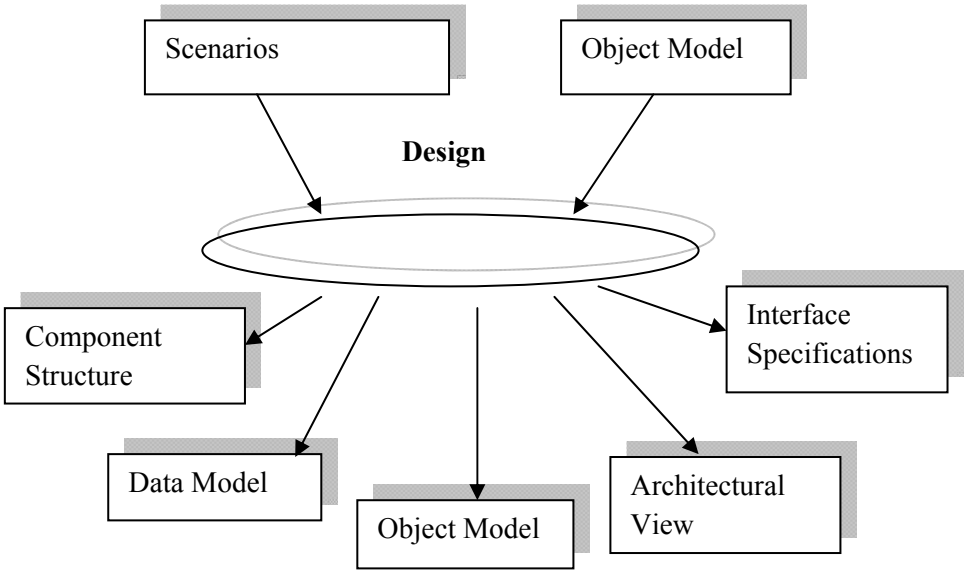

Figure 5.

Moreover, the same diagram is used with different purposes during the cycle of life:

- the activities diagram is attached to a usage case in the analysis stage and serves in the implementation stage to the detailed description of algorithms. Its components are translated into SQL phrases or in program instructions, and contribute to the defining of the structure of classes.
- object diagrams are used in the analysis stage for the abstraction of the model and in the design stage for adding details necessary for the implementation.

REFERENCES:

[1]. Booch, G.; Jacobson, I.; Rumbaugh, J. (2000) *OMG Unified Modeling Language Specification*, Version 1.3 First Edition: March 2000. Retrieved 12 August 2010

[2]. Gabay, J. (2002) *Merise vers OMT et UML.* InterEdition, Paris.

[3]. Ionita, D. (2010) *UML in Business Administration;* Journal of Knowledge Management, Economics and Information Technology; Issue 1; December

[4]. Vatuiu, V.E. (2010) *Dimensions and Perspectives for Knowledge Management and Information;* Journal of Knowledge Management, Economics and Information Technology; Issue 1; December

[5]. Lungu, I. (2005) *Metode de dezvoltare a sistemelor informatice,* Editura Universitas, Petrosani

[6]. Lungu, I. *(*2005) *Sisteme informatice executive* – Editura ASE, Bucuresti
[7]. Stanciu, V. (2003) *Proiectarea sistemelor informatice,* Editura DualTech, Bucureşti
[8]. Țarcă, **N.** (2005) *Informatică economică*, Editura Universității din Oradea, Oradea
[9]. Țarcă, **N.** (2008) *Bazele utilizării calculatoarelor*, Editura Universității din Oradea, Oradea
[10]. Udrică, M. si colectiv (2009) *UML prin aplicatii*, Editura Renaissance, Bucureşti.
[11]. Udrică, M. (2000) *Modelare orientată obiect.* Editura Cison, Bucureşti
[12]. Vătuiu, T.; Udrică, M. (2010) *Sisteme informatice. Eficiență prin analiză, proiectare, implementare.* Editura Renaissance, Bucureşti
[13]. Vătuiu, T. (2007) *Proiectarea sistemelor informatice, vol. 2,* curs universitar ID, Editura Academica Brâncuşi, Tg-Jiu
[14]. Vătuiu, T. (2006) *Proiectarea sistemelor informatice: aspecte conceptuale şi manageriale,* Editura Universității din Sibiu
[15]. Vătuiu, T. (2001) *Procesul de dezvoltare SOFTWARE UML*, Sesiunea națională de comunicări ştiințifice, ediția a-IX-a, Universitatea „Constantin Brâncuşi" , Facultatea de Ştiințe Economice, Tg-Jiu, ,Vol.4, Editura „Studii Economice", ISBN 973-85266-0-4, pag. 160
[16]. Vătuiu, T. (2001) *Analiza orientată pe obiecte în cazul metodologiei UML*, Sesiunea națională de comunicări ştiințifice, ediția a-IX-a, Universitatea „Constantin Brâncuşi" , Facultatea de Ştiințe Economice, Tg-Jiu, ,Vol.4, Editura „Studii Economice", ISBN 973-85266-0-4, pag. 164