ANNALS of Faculty Engineering Hunedoara
– International Journal of Engineering

Tome XII [2014] – Fascicule 3 [August]
ISSN: 1584-2673 [CD-Rom, online]

a free-access multidisciplinary publication
of the Faculty of Engineering Hunedoara

[1.] Anca-Elena IORDAN

DESIGN WITH UML OF AN INTERACTIVE SOFTWARE FOR THE STUDY OF THE TWO SURFACES OF REVOLUTION

[1.] Technical University of Timisoara, Engineering Faculty of Hunedoara, ROMANIA

Abstract: This article illustrates the required phases for the achieving of the interactive software used for the study of two surfaces of revolution: sphere and pseudosfere. The modelling of the interactive software is accomplished by specific UML diagrams representing the phases of analysis, design and implementation. The analysis phase is characterized by two types of UML diagrams: use-case diagram and activity diagrams. The design phase is characterized by three types of diagrams: class diagram, state diagrams and interaction diagrams. Implementation phase it corresponds the component diagram. By achieving these types of diagrams, the interactive software thus is described in an obvious and objective manner, without ambiguity.
Keywords: Interactive Software, Java, UML, Sphere, Pseudosphere

1. INTRODUCTION

A surface of revolution [1] is a surface in Euclidean space created by rotating a curve (the generatrix) around a straight line (axis) in its plane. The parametric equations of a surface of revolution are given by the next relations:

$$\begin{cases} x(u,v) = \varphi(u)\cdot\cos v \\ y(u,v) = \varphi(u)\cdot\sin v \\ z(u,v) = \psi(u) \end{cases} \quad (1)$$

In this paper will be analysed two types of surfaces of revolution: sphere and pseudosphere. A sphere is defined mathematically as the set of points that are all the same distance *r* from a given point in three-dimensional space [2]. Considering orthogonal coordinate system, the sphere (the two-dimensional surface) can be expressed by the following parametric equations.

$$\begin{cases} x(u,v) = x_0 + r\cdot\sin\theta\cdot\cos\varphi \\ y(u,v) = y_0 + r\cdot\sin\theta\cdot\sin\varphi, 0 \le \theta \le \pi, 0 \le \varphi \le 2\pi \\ z(u,v) = z_0 + r\cdot\cos\theta \end{cases} \quad (2)$$

The pseudosphere is sometimes also called the antisphere or tractricoid [3]. This surface is the result of a tractix rotation around its axis. The name "pseudosphere" comes about because it is a two-dimensional surface of constant negative curvature just like a sphere with positive Gauss curvature. The cartesian parametric equations are:

$$\begin{cases} x(u,v) = a + r\cdot\sin\theta\cdot\cos\varphi \\ y(u,v) = b + r\cdot\sin\theta\cdot\sin\varphi \\ z(u,v) = c + r\cdot\left(\ln tg\frac{\theta}{2} + \cos\theta\right) \end{cases}, 0 < \theta < \pi, 0 \le \varphi \le 2\pi \quad (3)$$

2. SOFTWARE ANALYSIS

From the perspective of unified modelling language, interactive software analysis includes the representation of two types of diagrams: the use-case diagram and activity diagrams.

The use-case diagram [4] offers simplified and graphical representation of what the interactive software must really do. Use-case diagram is based upon functionality and thus will focus on the

"what" offers the interactive software and not the "how" will be realized. The use case diagram corresponding to this software, presented in figure 1, includes: an actor, six use cases that describe the functionality of the software and relationships among them.

For every use-case of the previous diagram was achieved an activity diagram. Each activity diagram specifies processes and algorithms used to achieve the purpose specified by the use-case. In terms of UML, activity diagrams [5] are meant to model both computational processes and those of the organization. Activity diagrams show the general flow of control. In figure 2 is rendered the activity diagram corresponding to the use-case "Surface representation obtained by isometry".

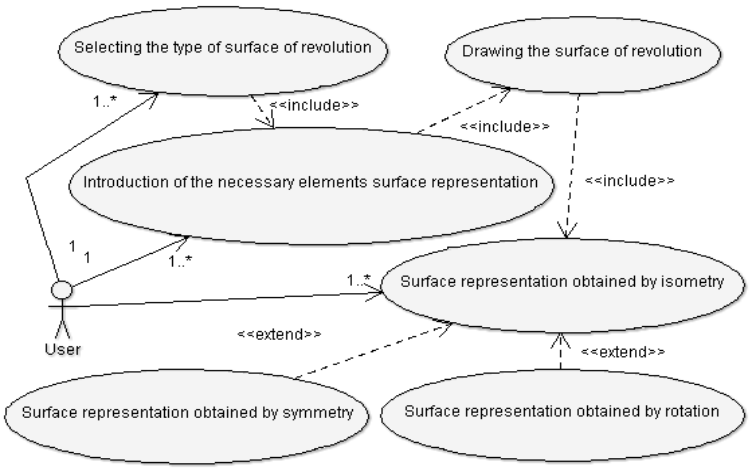

Figure 1. Use-case diagram

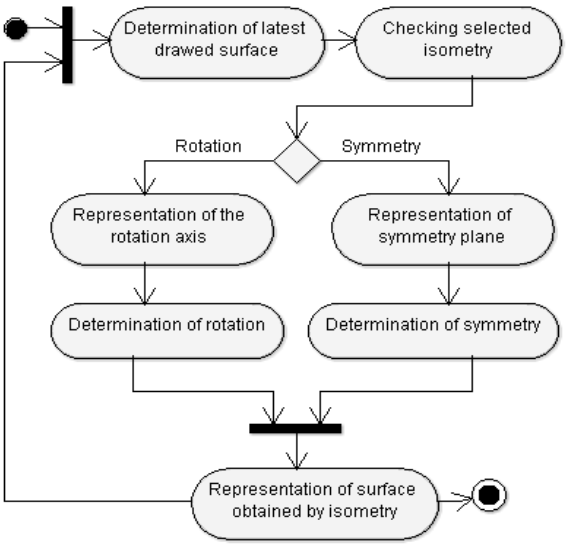

Figure 2. Activity diagram

3. SOFTWARE DESIGNING

Class diagram [6] represents the main block of object-oriented modeling. It is used both for static conceptual modeling software as well as detailed modeling aimed at translating in programming source code. For achieving the objectives of software were identified required classes and the relationships between them.

Figure 3 presents the inheritance, composition and aggregation relationships used. It may be noted that all attributes and methods of the *Element3D* class will apply to the all its derived classes: *Dreapta3D, Plan3D, Punct3D, Sfera* and *Pseudosfera*. Similarly, that all attributes and methods of the *Izometrie3D* class will apply to the all its derived classes: *Rotatie3D* and *Simetrie3D*. This class diagram shows contain specific classes to the interactive software as well as existing classes and interfaces from Java.

A sequence diagram [7] presents the interactions between objects arranged in a temporary sequence. He describes the objects involved in interaction and the sequence of messages exchanged between the needed objects to achieve the desired purpose. Sequence diagrams are usually associated with a use-case.

The diagram illustrates in figure 4 shows the interactions between objects, which have as purpose the drawing a pseudosphere. One can notice that there are interactions between five objects, out of which the objects of *Suprafata3D*, *Vector<Element3D>* and *Graphics2D* type are already created, and the other two objects: *Parametru* and *Pseudosphera* type will be instantiate during the interactions.

At first the execution control is taken by the object of *Suprafta3D* type. Following an event that interacts with the *Suprafta3D* object allows the creation of an instance of *Parametru* class. After getting the parameters that characterize the pseudosphere, control is returned to the *Suprafata3D* object. Following is created an instance of *Pseudosfera* class. The control will be given to the object of *Suprafta3D* type that will destroy the object of *Parametru*. We can observe that lifeline of the

Parametru object is interrupt, by marking an X, the message appears bearing the stereotype <<*destroy*>>. Control is transmitted to the object of *Vector<Element3D>* type to add the object previously created.

The control will be given to the object of *Suprafta3D* type that will destroy the object of *Pseudosfera* type. Through interaction with *Graphics2D* object will redraw the pseudosphere.

Collaboration diagrams, on the other side, concentrate on the relationships among the objects [8]. They are very useful for visualizing the way several objects collaborate to achieve the intended purpose and for comparing a dynamic model with a static model (figure 5). Collaboration and sequence diagrams render the same information, and can be transformed into one another without difficulty. The selection of the two types of diagrams depends upon what the designer wants to make visually apparent.

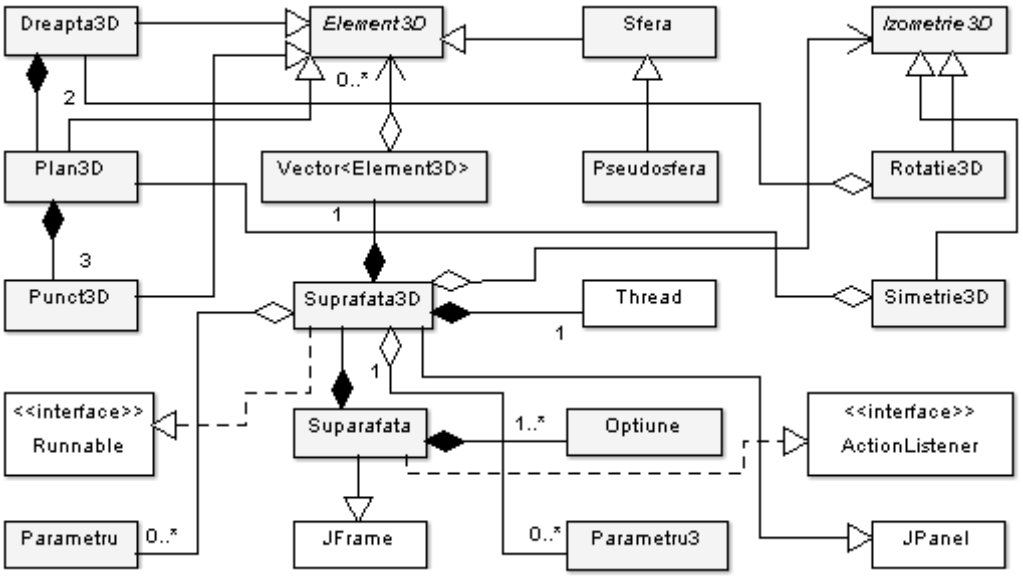

Figure 3. Class diagram

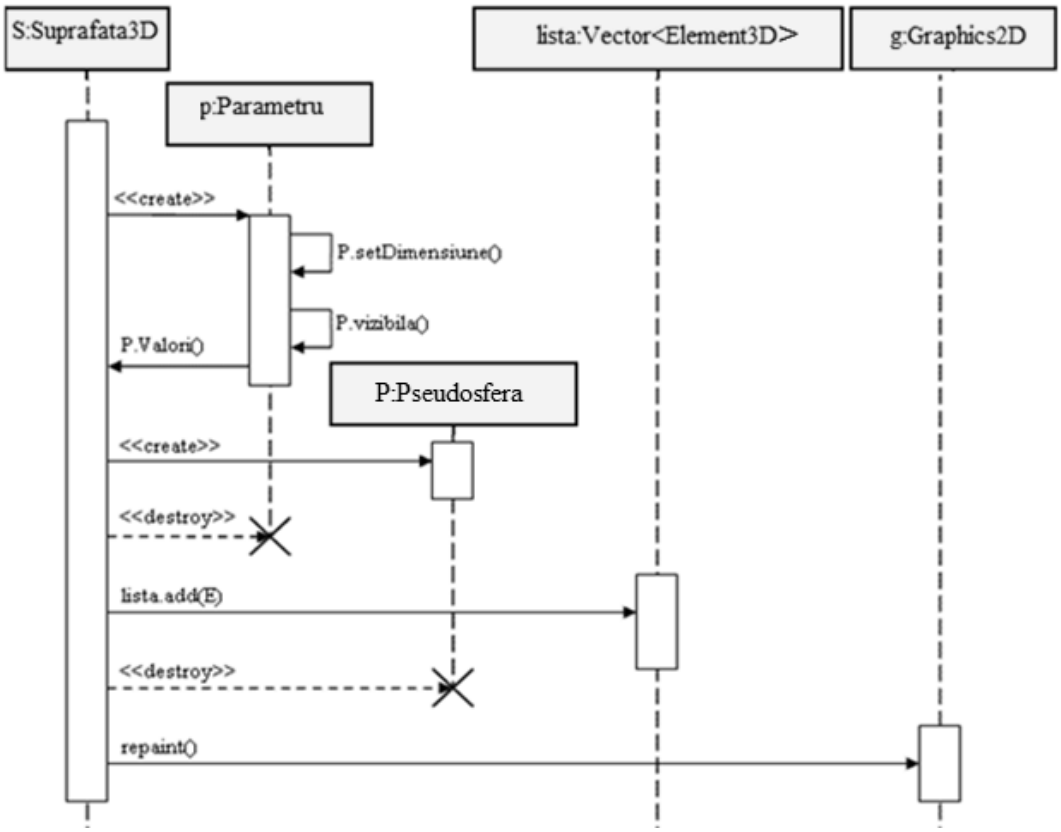

Figure 4. Sequence diagram

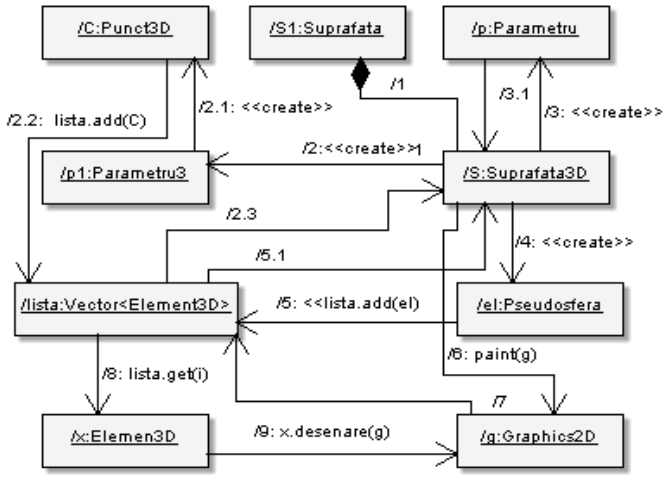

Figure 5. Collaboration diagram

4. SOFTWARE IMPLEMENTATION

The component diagram [9] enables the visualization modules that compose the software and the dependencies between them. The diagram that is shown in figure 6 describes the collection of components that all together ensure the functionality of the interactive software.

Central component of the diagram is *Suprafata.class*, a component obtained by transforming by the Java compiler into executable code of the *Suprafata.java* component. As can be seen that component interacts directly with components *Suparfata3D.class*. This component interacts with the next three components: *Element3D.class*, *Izometrie3D. class* and *Parametru.class*.

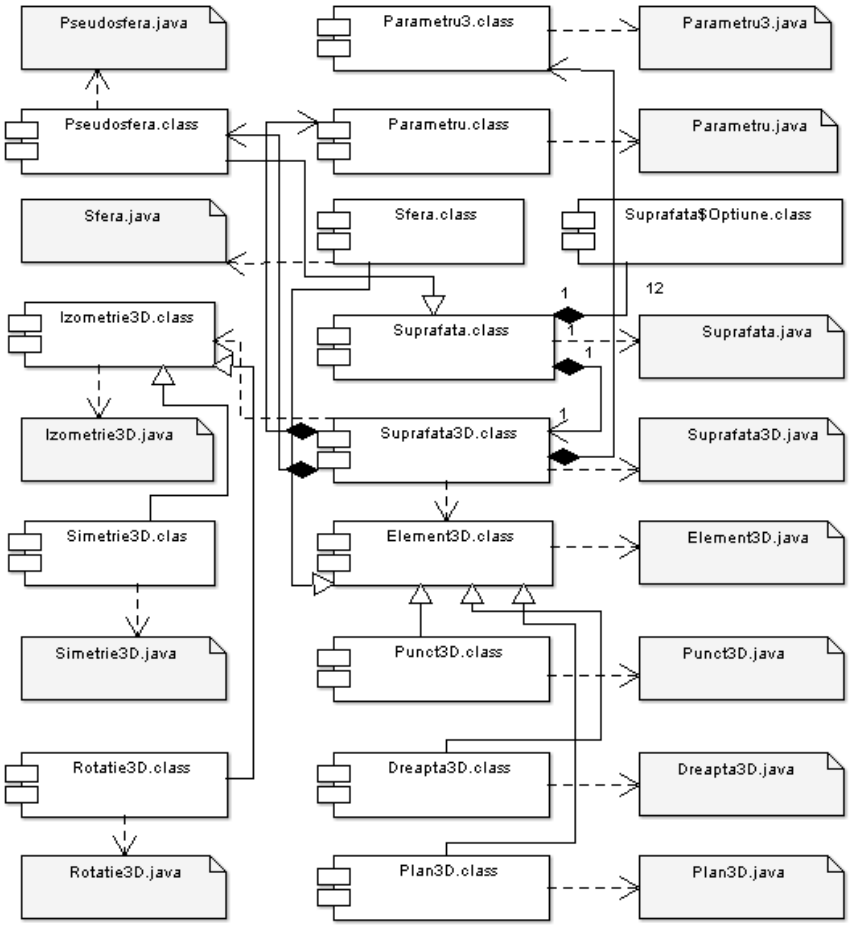

Figure 6. Component diagram

5. GRAPHICAL USER INTERFACE

The interactive software is accomplished using the Java object-oriented programming language [10]. From the application window it can selected by a main menu the following options:

- ✓ the graphic representation of sphere;
- ✓ the graphic representation of pseudosphere;
- ✓ the plan tangent and the normal to the sphere into a point;
- ✓ the surface of revolution obtained by applying an isometry: rotation or symmetry.

In the figure 7 is represented a sphere obtained by applying a symmetry about a plane, and in the figure 8 is represented a pseudosphere obtained by applying a symmetry about a plane.

The interactive system allows the utilization of any orthogonal projection for representing the 3D geometric elements as two-dimensional images in the projection plan.

Specification of the desired projection is made by means of the three angles made by the three axis of the orthonormal benchmark from space with Ox axis from the projection plan. After achievement of a 3D geometric construction there is the possibility to change the observation point by rotations around the Ox, Oy and Oz axis.

In the figure 9 is presented a sequence from the animation of a sphere and a pseudosphere, animation achieved by rotation around the Ox axis. The presented sequence includes 6 two-dimensional images obtained by using the following projections: (α=-140, β=-40, γ=70), (α=-140, β=20, γ=130), (α=-140, β=80, γ=-170), (α=-140, β=140, γ=-110), (α=-140, β=-160, γ=-50) and (α=-140, β=-100, γ=10).

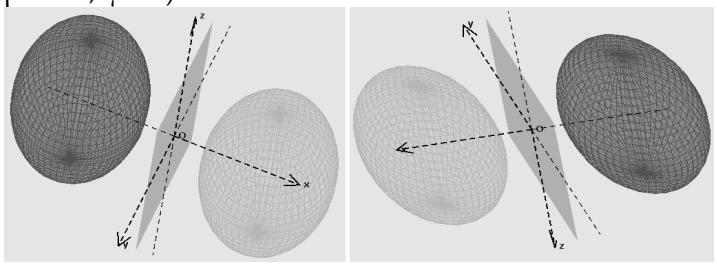

Figure 7. Sphere obtained by symmetry

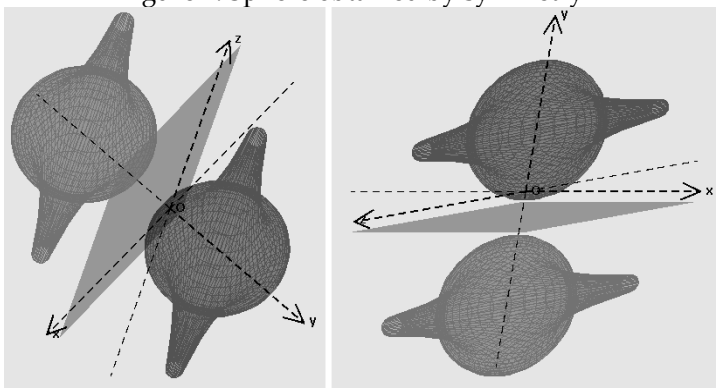

Figure 8. Pseudosphere obtained by symmetry

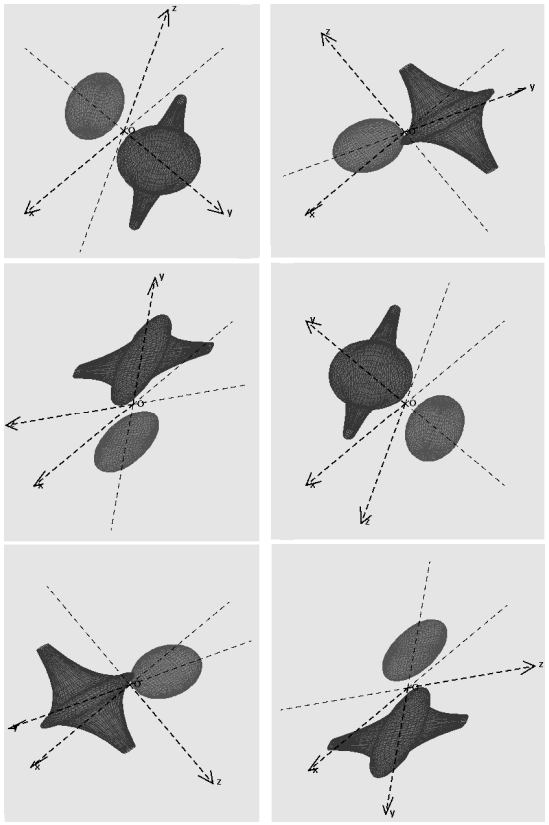

Figure 9. Rotation of a sphere and a pseudosphere around the Ox axis

6. CONCLUSIONS

Because the representation of the diagrams corresponding to all three phases: analysis, design and implementation, the interactive software has been described in an obvious and objective manner, without ambiguity. The use of the unified modelling language for the achievement of the diagrams is characterized by rigorous syntactic, rich semantic and visual modelling support.

The diagrams have been achieved using a new approach, multidisciplinary of the interactive software, grouping both modern pedagogy methods and discipline-specific components. The connection between teaching activities and scientific goals and objectives was established through the development of the new methods and the assimilation of new means, capable of enhancing school performance, enabling students to acquire the knowledge and techniques required and apply them in optimum conditions.

REFERENCES

[1.] G. Salmon, *A Treatise on the Analytic Geometry of Three Dimensions*, 2002

[2.] W. McCrea, *Analytical Geometry of Three Dimensions*, Dover Publications, 2006

[3.] A. Church, *Elements of Analytical Geometry*, 2014

[4.] M. Fowler, K. Scott, *UML Distilled: A Brief Guide to the Standard Object Modelling Language*, Pearson Education, 2003
[5.] L. Craig, *Appliyng UML and Patterns*, Pearson Education, 2008
[6.] K. Lunn, J. Skelton, S. Bennett, *Schaum's Outline of UML*, McGraw-Hill Education, 2004
[7.] B. Henderson-Sellers, B. Unhelkar, *Open Modelling with UML*, Addison Wesley, 2000
[8.] B. Oestereich, *Developing Software with UML*, Addison Wesley, 2000
[9.] P. Jalote, *A Concise Introduction to Software Engineering*, Springer-Verlag, 2008
[10.] J. Bloch, *Effective Java*, Pearson Education, 2008

ANNALS of Faculty Engineering Hunedoara – International Journal of Engineering