Modelling using UML diagrams of an Intelligent System for the Automatic Demonstration of Geometry Theorems

ANCA IORDAN, MANUELA PĂNOIU, IOAN BACIU, CORINA DANIELA CUNȚAN
Technical University of Timişoara, Engineering Faculty of Hunedoara,
Revoluției 5, 331128 Hunedoara
ROMANIA
anca.iordan@fih.upt.ro

Abstract: - In this work will be presented the design of an intelligent system destined for development process of demonstrating abilities for geometry theorems. This system will make available to user a proof assistant which will allow interactive vizualization of several demonstrations for the same theorem, demonstrations that have been generated by using three specific methods for automatic demonstration of theorems: area method, full-angle method and inferences accomplishment. For the implementation of the component used to represent knowledge and proof mechanisms will be used Prolog language and for the achievement of geometric construction associated to the theorem will be used Java language.

Key-Words: - Intelligent Software, Geometry, Java, Prolog, Automatic Demonstration Theorems.

1 Introduction

Dynamic geometry software and computer algebra software are the most widely used software for mathematics in education. Dynamic geometry software allow the user to create geometric constructions and to animate these constructions by dragging the free objects. All the depending objects are updated in real time. As the proving activity is central in mathematics, it is surprising that most software systems which are in widespread use to teach mathematics can not deal with this activity. Software which are specialized in proofs do exist, they are called proof assistants. Adapting proof assistants to be used in the classroom is an active research field [1].

The education community has studied the impact of the use of Dynamic Geometry Software on the proving activity [2,3]. Dynamic Geometry Software are mainly used for two activities:

- to make the student create geometric constructions;
- to make the student explore the figure, invent conjectures and check facts.

We believe that these software systems should also be used to help the student in the proving activity itself. Work has been performed in this direction and several Dynamic Geometry Software with proof related features have been produced.

1.1 Analysis of the existing dynamic geometry software

There are quite many dynamic geometry software: Cabri Euclide, Cabri Geometer, Chypre, Cinderella, Dr. Geo, Euclid, Euklid DynaGeo, Eukleides, GeoLabo, Geometria, Geometrix, Geometry Explorer, Geometry Tutor. But few can deal with proofs: Cabri Euclide, Chypre, Cinderella, Geometrix, Geometry Explorer, Geometry Tutor.

These systems can be roughly classified into two categories:

- the systems which permit to build proofs;
- the systems which permit to check facts using an automated theorem prover.

The Geometry Tutor [4], Chypre [5], Cabri-Euclide [6] and Geometrix [7] systems belongs to the first category. Using these systems the student can produce proofs interactively using a set of known theorems. In most of these systems the student can not invent a proof very different from what the program had pre-computed using automated theorem proving methods.

Geometry Explorer [8] and Cinderella [9, 10] belongs to the second category. Geometry Explorer provides a diagrammatic visualization of proofs generated automatically by a Prolog implementation of Chou's full angle method [11]. Cinderella allows to export the description of the figure to computer algebra software systems to perform algebraic proofs.

1.2 Approaches to proving geometry theorems

There are two approaches to proving geometry theorems using computers: the artificial intelligence

approach and the algebraic computation approach. The earliest work in geometry theorem proving by computer programs was done by Gelernter and his collaborators [12].

It was based on the human simulation approach and has been considered a landmark in the artificial intelligence area for its time. In the area of algebraic computation approach, the earliest work dates back to Hilbert. In his classic book [13], Hilbert outlined a decision method for a class of constructive geometry statements in affine geometry.

A breakthrough in automated geometry theorem proving is made by Wu. Restricting himself to a class of geometry statements of equality type, Wu introduced a method in 1977 which can be used to prove quite difficult geometry theorems efficiently [14]. Ko and Hussain [15], Wang and Hu [16], Gao [17], Kapur and Wan [18] also succeeded in implementing theorem provers based on various modified version of Wu's method.

The success of Wu's method has revived interest in proving geometry theorems by computers. In particular, the application of the Gröbner basis method [19] to the same class of geometry theorems that Wu's method addresses has been investigated.

More recently, the artificial intelligence approach has been revived to such an extent that it can solve hundreds of difficult geometry problems and produce multiple and shortest proofs for geometry theorems efficiently [20].

The artificial intelligence approach is also used for automated generation of construction steps of geometric diagrams and successfully applied to many difficult geometric problems. Methods of automated reasoning in geometry have a wide range of applications, including kinetic analysis of robotics, linkage design, computer vision, etc.

1.2.1 Bracket algebra methods

One of the earliest effort to develop coordinate free methods of geometric reasoning is to use techniques from the bracket algebra such as Cayley factorization [21]. The bracket algebra is a non-commutative algebra. There is still no decision method similar to that of the Gröbner basis. Therefore, bracket algebra can only be used to do "computer-aided geometric reasoning".

In [22] an algorithm based on bracket algebra for proving projective geometry theorems was given. The basic idea is to represent geometric hypotheses and conclusions as algebraic relations and use simple algebraic computation to deduce the conclusion from the hypotheses. The proofs thus generated are very short. Based on this technique, a program called *Cinderella* has been developed [23].

1.2.2 Area method

The area method uses high-level geometric lemmas about geometry invariants such as the area and the Pythagorean difference as the basic tool of proving geometry theorems [24]. Zhang found many elegant ad hoc methods based on areas of triangles to solve geometric problems.

These ad hoc methods have been developed into a complete method of AGTP, which are surprisingly powerful in than it has been used to proved hundreds of geometry theorems of constructive type and the proofs are generally short and elegant. This method seems to be the first to produce human-readable proofs for hard geometry theorems efficiently.

1.2.3 Full-Angle Method

The full-angle method [11] has been demonstrated to prove hundreds of geometry theorems automatically whilst producing proofs which are both short and human-readable [24, 25].

The full-angle method relies on a single high-level geometric invariant called the full-angle to prove theorems.

The full-angle method is a rule based method and is not a decision procedure. But this method also has its advantages: all the proofs produced by the method are very short and it has been used to prove several theorems that all the other methods fail to prove.

1.2.4 Wu's method

Wu's method is the most powerful method in terms of proving difficult geometry theorems. Wu's method is a coordinate-based method [14]. It first transfers geometry conditions into polynomial equations then deals with the polynomial equations with the characteristic set method. This method has been used to prove more than 600 geometry theorems.

Following Wu the mechanization problem of geometry theorem proving consists mainly of three steps:

- The first is the algebraization of geometry, namely, reducing the problem of geometry theorem proving to purely algebraic problem. Then the hypothesis and conclusion of a theorem can be expressed as sets of algebraic relations.
- The second step is to well order those algebraic relations which correspond to the hypothesis of the theorem and decide whether the algebraic relations corresponding to the conclusion can be deduced from the well-ordered algebraic relations according to a certain procedure. This is called the mechanization of geometry, while the

procedure for determining the inference relation is called a mechanization method.

- The final step is to implement the method on computers in order to achieve the proof of each theorem.

Algebraic methods, though powerful, generally can only tell whether a statement is true or not. After Wu's method, several researchers tried to develop automated geometry theorem proving (AGTP) methods based on vector calculation in the mid-80s in order to find simpler proofs [26].

1.2.5 Automated diagram generating

Most work on automated geometry reasoning focused on theorem proving and discovering. In [11] a global propagation method for automated generation of construction steps of diagrams was presented. This method uses a forward chaining to find the information needed in the construction and uses a backward chaining to find the construction sequence. For a diagram described declaratively with geometric constraints, the method may be used to find a sequence of constructing steps of drawing the diagrams with ruler and compass.

2 Presentation of the proposed informatics systems

The system combines these features: dynamic geometry, automatic theorem proving and interactive theorem proving.

The advantages of this system are:

- The use of a proof assistant provides a way to combine geometrical proofs with larger proofs.
- There are facts than can not be visualized graphically and there are facts that are difficult to understand without being visualized.
- We should have both the ability to make arbitrarily complex proofs and use a base of known lemmas.
- The verification of the proofs by the proof assistant provides a very high level of confidence.

After designing the system functional architecture, that should allow the support of the didactic activity, is proposed a model for the graphic interface that should satisfy the specific needs of a didactic environment from the teachers, students and pupils viewpoint. Designing of the user graphic interfaces represents the most important phase of the process for implementing an intelligent training system. The user interface should provide all the necessary facilities to a student or pupil, in order to navigate intuitively within the application and as transparent possible.

The user-oriented designing process imposes the interface the characteristics that allow the user to control the training process. A usable interface must be studied, achieved and repeatedly tested in order to maximize the efficiency and minimize the time necessary for the teaching and training processes.

2.1 System's analysis

The informatics system will be described in a clear and concise manner by presenting the use-cases, using the UML unified modelling language [27].

Representation of the use-cases diagram is shown in figure 1. Each case describes interactions between the user and the system.

For each use case presented in the previous diagram we'll build activity diagram. Each diagram will specify the processes or algorithms which are behind the analysed use-case.

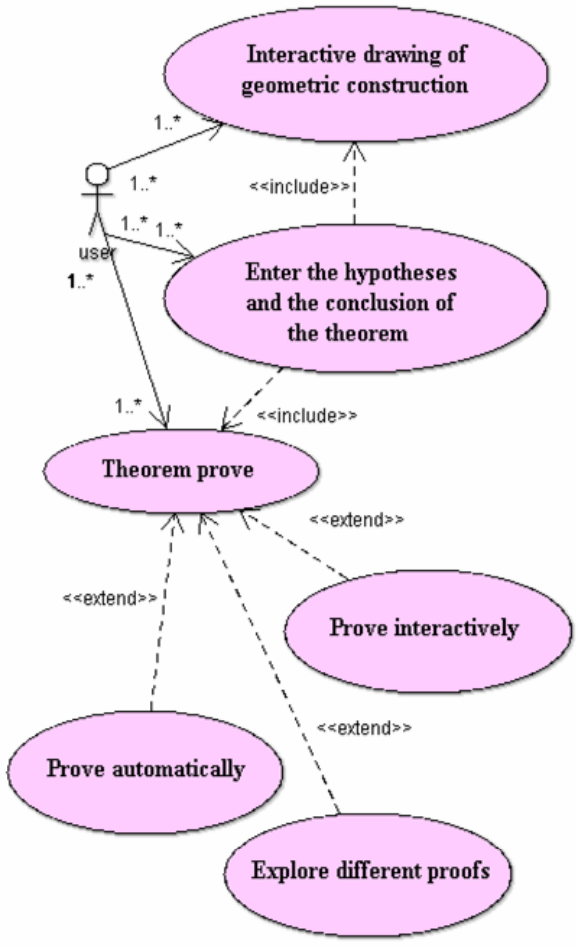

Fig.1 Use-cases diagram

The implementation activity for the goal „Interactive drawing of geometric construction" was finalized by the following personal contributions [28, 29, 30, 31, 32]:

- efficient implementation of the classes corresponding to the 2D and 3D geometric elements by achieving of inheritance, aggregation and composition relationships;
- implementation of some instantiation algorithms of the elements, of intersection or tangency, using special programming methods, algorithms which contribute to the rapid and accuracy of the desired geometric elements graphic representation;
- description of the interactions between objects in different contexts, as well as visualizing the mode in which is divided the system and the dependencies between modules;
- introduction of options non-existent in case of other such systems for achieving accurate 2D geometric constructions;
- possibility to achieve 3D geometric constructions;
- visualization of the 3D geometric elements as two-dimensional images in the projection plan, using the desired orthogonal projection.

The use-case "Introduction of hypothesises and the conclusion of the theorem" will allow the reception of data about the problem that is to be resolved and the translation of this data obtaining this way the facts base. This use-case will interact with the use-case that allows the drawing of geometric figures.

Starting with the facts base, the intelligent system will offer the option of drawing automatically the geometric figure corresponding to the theorem but there will exist also the variant of accomplishing the geometric figure interactively.

The achievement of the use-case "Theorem demonstration" will be made by traversing the next stages:

- Accomplishment of the representation of geometric knowledge component;
- Implementation of demonstrating mecha-nism;
- Accomplishment of design assistant.

Representation of geometric knowledge component will consist from a number of declaring modules, each of them adding to the knowledge base new geometry concepts and theorems.

Geometric objects will be represented as facts in the facts base and for definitions and theorems will be used rules. The programming language used to represent knowledge will be Prolog.

For implementing the demonstration mechanism in Prolog language will be used three methods that are specific to geometry theorems demonstration:

- area method;
- full-angle method;
- combining the forward and backward chaining techniques.

The activities deployed for implementing the component for interfaces accomplishment are:

- Representing solution by AND/OR trees;
- Combining the forward and backward chaining techniques;
- Optimizing these techniques by choosing the optimal heuristic information for obtainig the solution;
- Determination of the efficiency for this method by comparing it with the demonstrations generated through utilising the area method and the full-angle method.

For the accomplishment of demonstrating assistant the next steps are going to be followed:

- Fruition of the graphical window with it's corresponding menu and instruments;
- Obtaining corresponding stages of the algorithm used for the demonstration of the theorem and the accomplishment of the corresponding diagram;
- Graphical representation of the transited steps in demonstration through using the prius created diagram.

The intelligent system will offer the possibility of comparing the demonstrations of the same theorem using the three methods, thus developping the users abbilities to demonstrate theorems.

2.2 System's designing

Designing of data structures is crucial for the stability and performance of the entire intelligent system. If we think to hierarchies of classes for an object-oriented system for dynamic geometry, we'll end by representing the inheritance as in the trees from [33,34].

The basic class for all 2D geometric objects is *Element2D*. This class contain all the methods and attributes which are used for all geometric elements, such as, for example, the colour and drawing style of a geometric element, and its drawing methods.

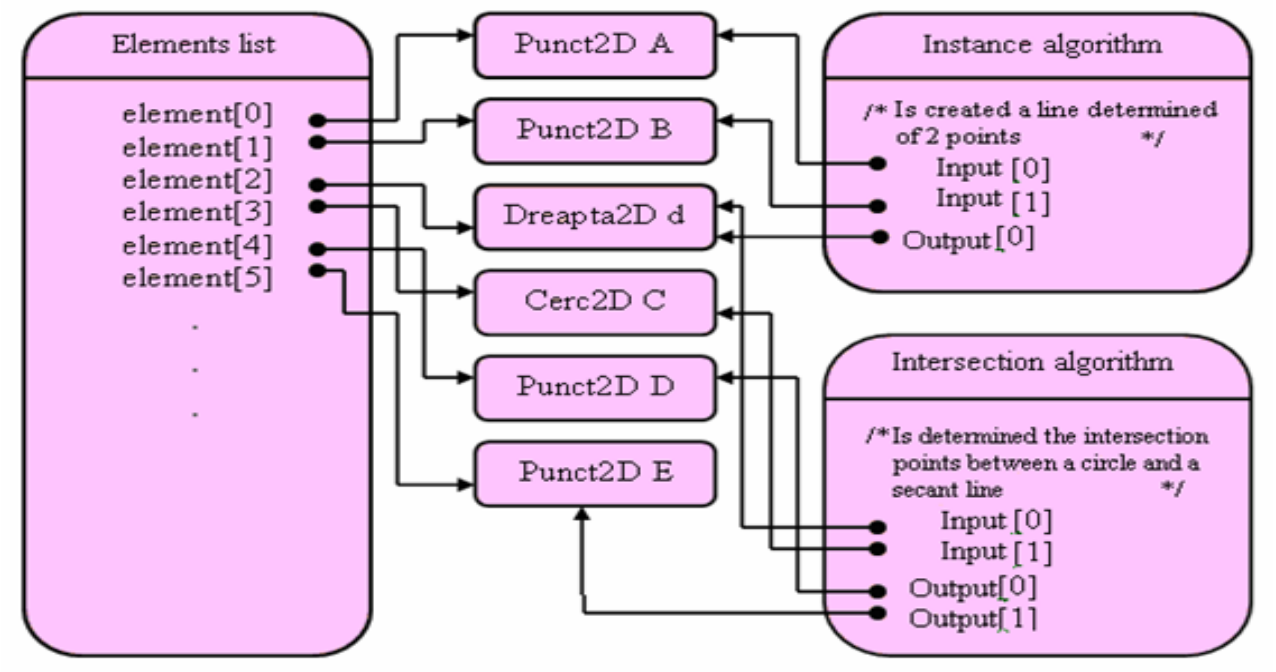

Fig.2 Part from structure of an instance of the *Desen2D* class

Subclasses of the *Element2D* class, such as: *Point2D*, *Line2D*, *Segment2D*, *Vector2D, Patrulater2D, Triunghi2D, Elipsa2D, Hiperbola2D, Parabola2D* contain additional data which are necessary for the respective geometric element, such as: coordinates of a point, extremities of a segment, origin and extremity of a vector, coefficients from the equation of a line.

Beside these classes corresponding to the geometric concepts, there are classes that contain algorithms for representation both of the independent geometric elements, and the dependent ones, such as: intersection of two unparallel lines, circle circumscribed to a triangle, tangent and normal to a conical in a point.

The class *Desen2D* contains algorithms for representation of 2D geometric constructions.

In figure 2 is presented a part of the structure for the *Desen2D* class that contains a list with all the geometric elements that have been created and which are instances of the *Element2D* class. Algorithms operate with data that is stored through attributes of instances.

The first presented algorithm receives as input data two *Punct2D* elements and returns an instance of the *Dreapta2D* class determined through the input data. The intersection algorithm receives also two input data: one instance of the *Dreapta2D* class, previously created, and one instance of *Cerc2D* class. If the line is secant to the circle then the algorithm will return, as output data, two instances of the *Punct2D* class that represents the intersection points.

Figure 3 presents the inheritance and achievement relationships used. It may be noted that all attributes and methods of the *JPanel* class will apply to the derived class *Desen2D*, which implements *MouseInputListener* interface.

2.3 Sequence diagrams

The sequence diagram [35] is used primarily to show the interactions between objects in the sequential order that those interactions occur. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them.

The sequence diagrams for this software are made with ArgoUML-0.28. The diagram illustrate in figure 4 shows the interactions between objects, which have as purpose the drawing the triangle determined by three points. One can notice that there are interactions between twelve objects, out of which the objects of *Desen2D*, *Vector<Punct2D>*, *Vector<Element2D>* and *Graphics2D* type are already created, and the objects of *Nume, Punct2D, Triunghi2D* and *MouseEvent* type will instantiate during the interactions.

These objects are represented on Ox axis and on Oy axis are represented the mesages ordered increasingly in time. At the beginning, the execution's control is undertaken by the object of *Desen2D* type which creates an instance of the *Vector<Punct2D>* class. Giving back the control to the object of *Desen2D* type, further will be instantiated the object of *Nume* type.

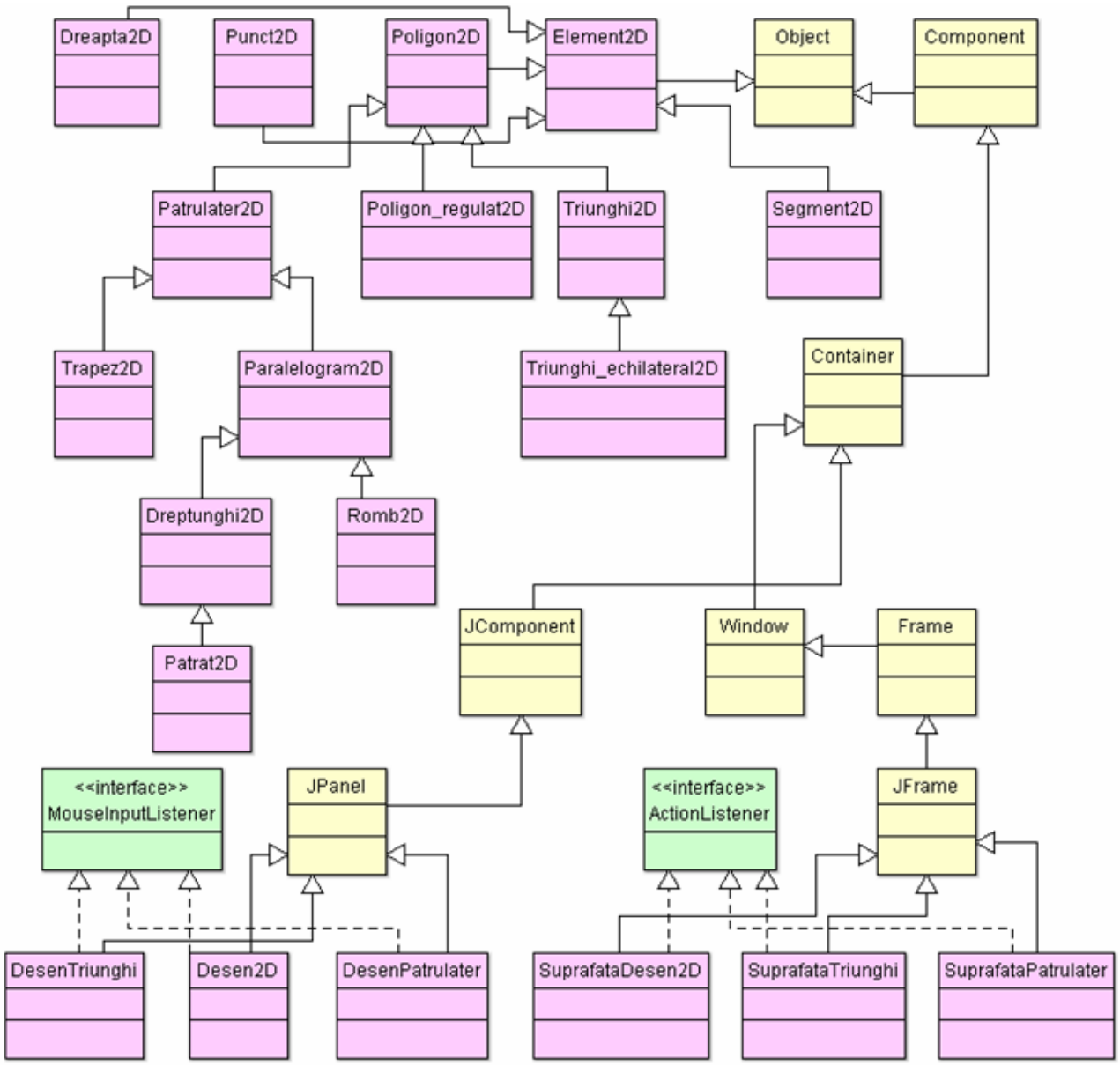

Fig. 3 Class Diagram

Now, the control is undertaken by this newly created instance, that will allow the display of a window where will be introduced name for the new point.

Further, the execution's control is transmitted to the object of *MouseEvent* type which will lead to instantiate an object of *Punct2D* type. The execution's control is transmitted to the object of *Vector<Punct2D>* type, in order to add the point previously created in the list of triangle vertices and then will be destroyed the instances of the *Nume* class and of the *Punct2D* class.

Further, will be instantiated the object of *Triunghi2D* type, representing the triangle, and then will be destroyed the object of *MouseEvent* type. The execution's control is transmitted to the object of *Vector<Element2D>* type, in order to add the triangle previously created in the list of 2D elements of the geometric construction, and then will be destroyed the instance of the *Triunghi2D* class. Finally, will be redrawn the geometric construction, which will include now also the triangle, by using the object of *Graphics2D* type.

The diagram illustrate in figure 5 shows the interactions between objects, which have as purpose the drawing the triangle's centroid and the medial triangle. One can notice that there are interactions between five objects, out of which the objects of *DesenTriunghi*, *Vector<Element2D>* and *Graphics2D* type are already created, and the objects of *Element2D* and *Triunghi2D* type will instantiate during the interactions.

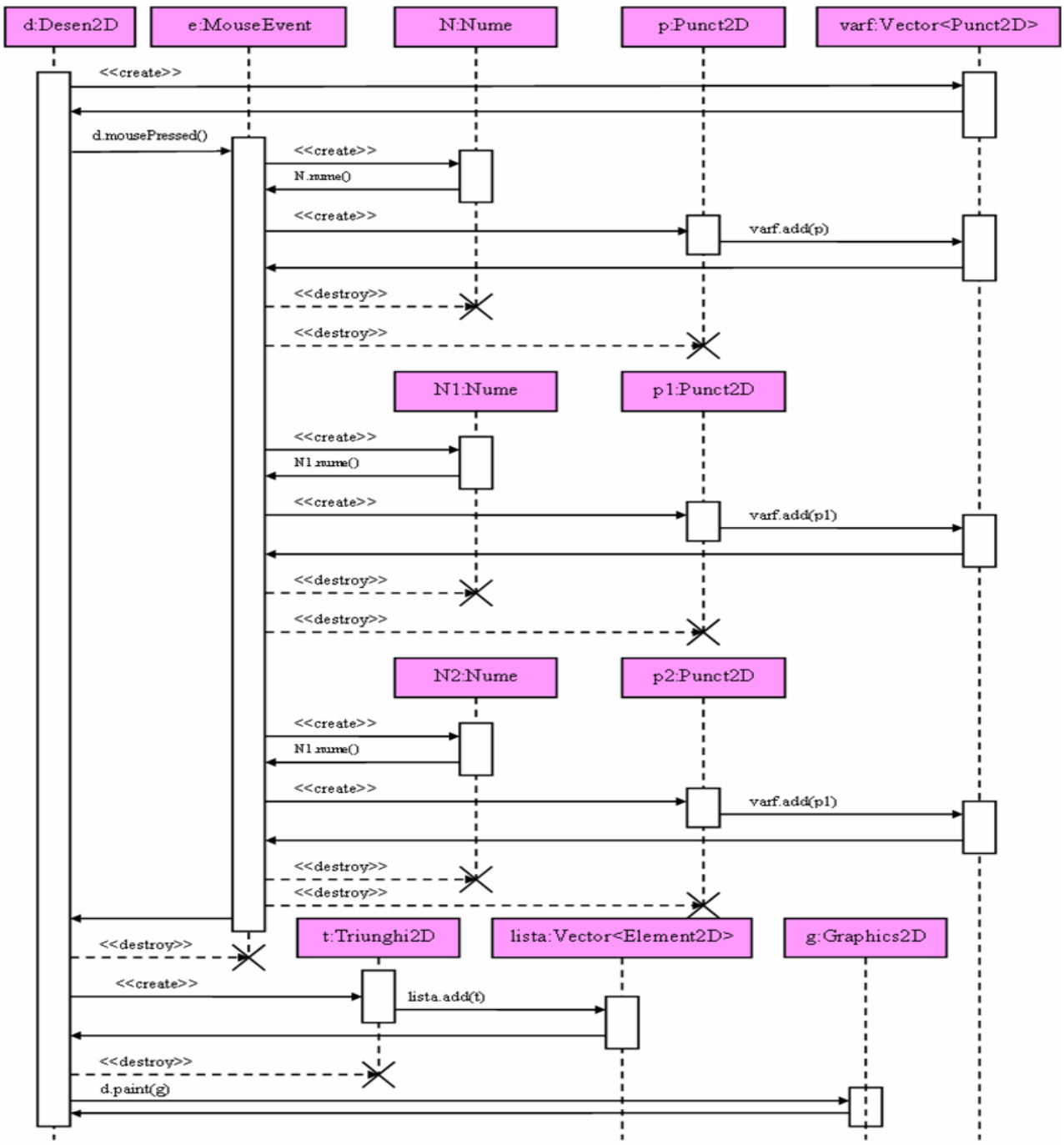

Fig. 4 Sequence diagram for drawing the triangle determined by three points

At the beginning, the execution's control is undertaken by the object of *DesenTriunghi* type which appeals an instance of the *Vector<Element2D>* class in order to obtained the triangle's centroid.

The execution's control is transmitted to the object of *DesenTriunghi* type and will be redrawn the geometric construction, which will include now also the triangle's centroid by using the object of *Graphics2D* type.

Giving back the control to the object of *DesenTriunghi* type, will be instantiated the object of *Triunghi2D* type, which represents the medial triangle.

Further, the execution's control is transmitted to the object of *Vector<Element2D>* type, in order to add the triangle previously created in the list of 2D elements of the geometric construction, and then will be destroyed the instance of the *Triunghi2D* class.

Finally, will be redrawn the geometric construction, which will include now also the medial triangle by using the object of *Graphics2D* type.

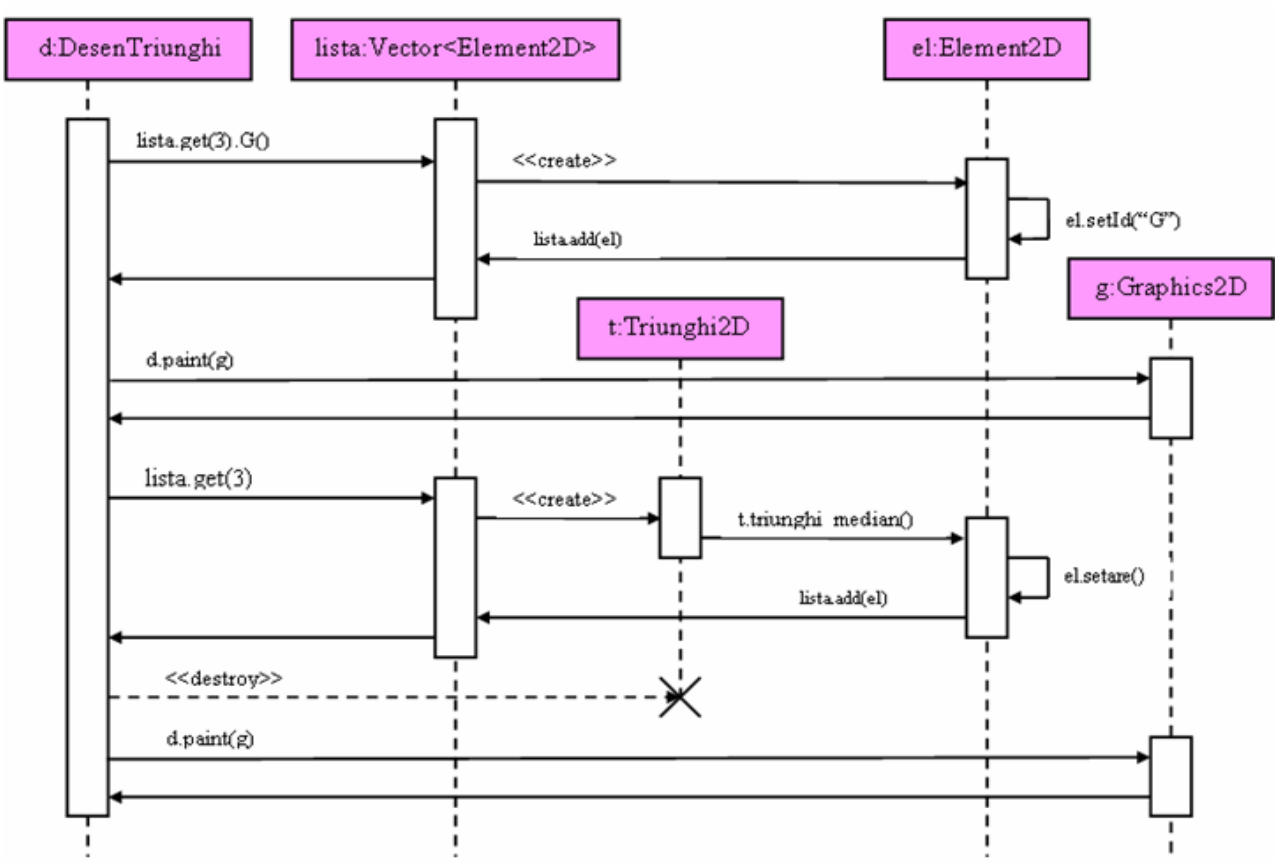

Fig. 5 Sequence diagram for drawing the triangle centroid and the medial triangle

The diagram illustrate in figure 6 shows the interactions between objects, which have as purpose the drawing the quadrilateral's bimedians. One can notice that there are interactions between six objects, out of which the objects of *DesenPatrulater*, *Vector<Element2D>* and *Graphics2D* type are already created, and the objects of *Element2D, Segment2D* and *Patrulater2D* type will instantiate during the interactions.

At the beginning, the execution's control is undertaken by the object of *DesenPatrulater* type which give the execution's control to the object of *Vector<Element2D>* type.

Now is created an instance of *Patrulater2D* type which permits the creation of three instances of *Segment2D* class corresponsive to the three bimedians. Finally, will be redrawn the geometric construction, which will include now also the quadrilateral's bimedians by using the object of *Graphics2D* type.

The diagram illustrate in figure 7 shows the interactions between objects, which have as purpose the drawing the triangle's circumcircle and the triangle's circumcenter.

One can notice that there are interactions between five objects, out of which the objects of *DesenTriunghi*, *Vector<Element2D>* and *Graphics2D* type are already created, and the objects of *Element2D* and *Triunghi2D* type will instantiate during the interactions.

At the beginning, the execution's control is undertaken by the object of *DesenTriunghi* type which appeals an instance of the *Vector<Element2D>* class in order to obtained the triangle's circumcenter.

The execution's control is transmitted to the object of *DesenTriunghi* type and will be redrawn the geometric construction, which will include now also the triangle's circumcenter by using the object of *Graphics2D* type.

Giving back the control to the object of *DesenTriunghi* type, will be instantiated the object of *Element2D* type, which represents the triangle's circumcircle.

Further, the execution's control is transmitted to the object of *Vector<Element2D>* type, in order to add the circle previously created in the list of 2D elements of the geometric construction.

Finally, will be redrawn the geometric construction, which will include now also the triangle's circumcircle by using the object of *Graphics2D* type.

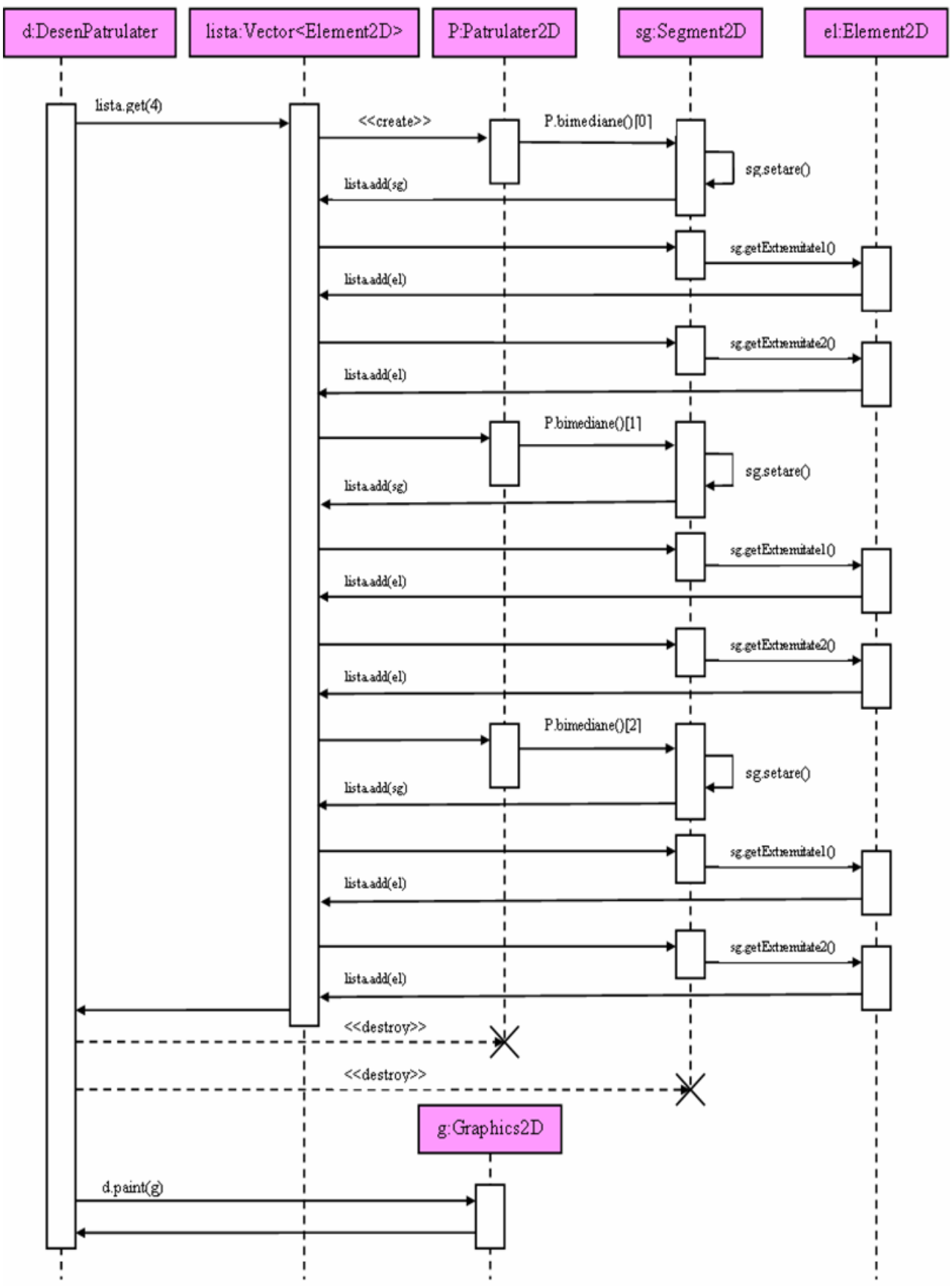

Fig. 6 Sequence diagram for drawing the quadrilateral bimedians

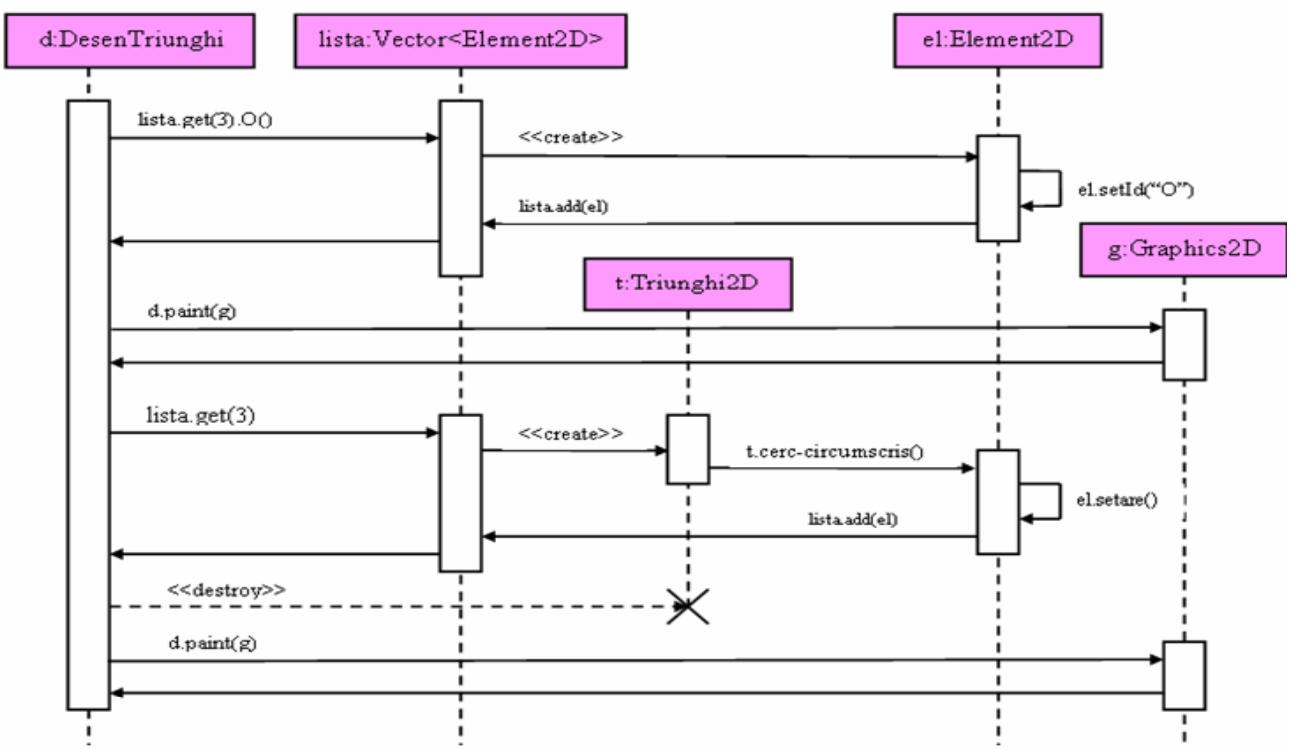

Fig. 7 Sequence diagram for the triangle circumcircle and the triangle circumcenter

2.4 The degree of originality of the intelligent system

The originality of this intelligent informatic system consists of:

- Automatic or interactive achievement of the geometric figure specific to the theorem that is going to be demonstrated.
- Using several methods to demonstrate theorems: with this methods for the same theorem it is going to obtain demonstrations in different styles. This aspect is important for using the intelligent system in the process of computer assisted training at geometry because different methods allow pupils and students to explore different demonstrations. Secondly, for a certain class of geometry theorems, a particular method can produce shorter demonstrations than other methods.
- Using of a demonstrating assistant: it will allow the combination of automatic theorem demonstration with the interactive one, offering a high level of trust in generating demonstrations.

3 Conclusion

Using the intelligent informatic system in studying geometry will contribute to forming and developping informatic culture of the pupils. Computer assisted training in the study process of geometry elements is also an efficient increasing method for the motivation of learning this discipline and the quality of it's assimilation.

The theme treated in this paper is of great actuality, geometry being an important component in forming young mathematicians, engineers or architects. The informatics application is aggregating in this great actuality and major issue that it approaches in a new multidisciplinary manner through the prism of modern educational technologies.

References:

[1] G. Mayrhofer, S. Saminger, W. Windsteiger, CreaComp: Experimental Formal Mathematics for the Classroom, *Proceedings of ICTMT8*, 2007

[2] O. Yevdokimov, About a constructivist approach for stimulating students' thinking to produce conjectures and their proving in active learning of geometry, *Fourth Congress of the European Society for Research in Mathematics Education*, 2004

[3] F. Furinghetti, P. Domingo, To produce conjectures and to prove them within a dynamic geometry environment: a case study, *Proceeding of Psychology of Mathematics 27th international Conference*, 2003

[4] J. Anderson, C. Boyle, G. Yost, The Geometry Tutor, *IJCAI Proceedings*, 1985

[5] P. Bernat, Chypre: Un logiciel d'aide au raisonnement, *Technical Report 10*, IREM, 1993

[6] V. Luengo, Cabri-Euclide: Un micromonde de Preuve intégrant la réfutation, *PhD thesis, Université Joseph Fourier*, 1997
[7] J. Gressier, Geometrix, 1988-1998
[8] S. Wilson, J. Fleuriot, Combining dynamic geometry, automated geometry theorem proving and diagrammatic proofs, *ETAPS Satellite Workshop on User Interfaces for Theorem Provers*, Edinburgh, 2005
[9] U. Kortenkamp, Foundations of Dynamic Geometry, *PhD thesis*, ETH Zürich, 1999
[10] J. Schwartz, Probabilistic algorithms for verification of polynomial identities, Symbolic and algebraic computation, *Lecture Notes in Computer Science*, vol. 72, 1979, Springer-Verlag
[11] S. Chou, X. Gao, J. Zhang, Automated generation of readable proofs with geometric invariants, theorem proving with full angle, *Journal of Automated Reasoning*, 1996
[12] H. Gelernter, Realization of a geometry theorem machine, *Proceedings of International Conference in Info Procces*, Paris, 1959
[13] D. Hilbert, Foundations of Geometry, *Open Court Publishing Company*, Illinois, 1971
[14] W. Wu, Basic Principles of Mechanical Theorem Proving in Geometries, *Science Press*, Beijing, 1984, English version, Springer Verlag, 1993
[15] H. Ko, M. Hussain, Geometry theorem proving by decomposition of quasi-algebraic sets, *Artificial Intelligence*, 1988
[16] D. Wang, S. Hu, Mechanical proving system for constructible theorems in elementary geometry, 1987
[17] X. Gao, Transcendental functions and mechanical theorem proving in elementary geometries, *Journal of Automated Reasoning*, 1990
[18] D. Kapur, H. Wan, Refutational proofs of geometry theorems via characteristic set computation, *Proceedings of ISSAC*, Tokyo, 1990
[19] J. Ritt, Differential Algebra, *AMS Colloquium Publications*, New York, 1980
[20] S. Chou, X. Gao, J. Zhang, A deductive database approach to automated geometry theorem proving and discovering, *Journal of Automated Reasoning*, 1996
[21] N. White, T. McMillan, Cayley factorization, *Proceedings of ISSAC*, ACM Press, New York, 1988
[22] J. Richter-Gebert, Mechanical theorem proving in projective geometry, *Annals of Mathematical and Artificial Intelligence*, 1999
[23] J. Richter-Gebert, U. Kortenkamp, Cinderella, *Springer-Verlag*, Berlin, 1999
[24] S. Chou, X. Gao, J. Zhang, Machine Proofs in Geometry, *World Scientific*, Singapore, 1994
[25] S. Chou, X. Gao, J. Zhang, A collection of 110 geometry theorems and their machine proofs based on full-angles, *Technical Report, Department of Computer Science*, The Wichita State University, 1994
[26] T. Havel, Some examples of the use of distances as coordinates for Euclidean geometry', *Journal of Symbolic Computation*, 1991
[27] M. Fowler, K. Scott, UML Distilled: A Brief Guide to the Standard Object Modeling Language, *Addison Wesley*, Readings MA, USA, 2000
[28] A. Iordan, M. Panoiu, C. Panoiu, Development New Dynamical Methods for the Study of the Euclidian Geometry, *WSEAS International Conference on Education and Educational Technology*, Italy, 2007
[29] A. Iordan, G. Savii, M. Panoiu, C. Panoiu, Development of a Dynamical Software for Teaching Plane Analytical Geometry, *WSEAS International Conference on Engineering Education*, Heraklion, Crete Island, Greece, 2008
[30] A. Iordan, G. Savii, M. Panoiu, C. Panoiu, Multimedia Interactive Environment for Study the Plane Analytical Geometry, *WSEAS Transactions on Computers*, Vol. 7, 2008
[31] A. Iordan, M. Panoiu, I. Muscalagiu, R. Rob, Realization of an interactive informatical system for the quadric surfaces study, *WSEAS International Conference on Computers*, Rhodes Island, Greece, 2009
[32] A. Iordan, Development of Interactive Software for Teaching Three-Dimensional Analytic Geometry, *WSEAS International Conference on Distance Learning and Web Engineering*, Hungary, Budapest, 2009
[33] A. Iordan, G. Savii, M. Panoiu, C. Panoiu, Development of a dynamical software for doing geometrical constructions, *WSEAS International Conference on Applied Informatics and Communications*, Rhodes Island, Greece, 2008
[34] A. Iordan, G. Savii, M. Panoiu, C. Panoiu, Visual interactive environment for doing geometrical constructions, *WSEAS Transactions on Computers*, Vol. 8, 2009
[35] J. Odell, Advanced Object Oriented Analysis& Design using UML, *Cambrige University Press*, 1998