ANNALS of Faculty Engineering Hunedoara
– International Journal of Engineering
Tome XIII [2015] – Fascicule 3 [August]
ISSN: 1584-2673 [CD-Rom; online]
a free-access multidisciplinary publication of the Faculty of Engineering Hunedoara

[1.] Anca-Elena IORDAN

INTERACTIVE SOFTWARE DESIGN FOR SHAPING ELECTRICAL CIRCUITS UTILIZING GRAPHS

[1.] University Politechnica Timisoara, Engineering Faculty Hunedoara, ROMANIA

Abstract: This work illustrates the necessary phases for the object oriented development of new software dedicated to the study of electrical circuits by means of graphs theory. The modelling of the interactive software is accomplished by specific UML diagrams representing the phases of analysis, design and implementation. The analysis phaseis characterizedby two types of UMLdiagrams:use-case diagramandactivity diagrams. The design phaseis characterized bythree types ofdiagrams:class diagram, state diagramsandinteractiondiagrams. Implementation phase it corresponds the component diagram.This software is meant for teachers and students that teach, explore or evaluate electric circuits, because electrical engineering domain, especially electrical circuits, is complicated to master for the great majority of students.
Keywords: Electrical Circuits, Graph Algorithms, UML Diagrams, Java

1. INTRODUCTION

Electrical circuit theory [1] is the most significant and the oldest branch of electrical engineering. In engineering education it is deemed meaningful the study of electrical circuit theory. Mastering the concepts and phenomena specific to the theory of electrical circuits is important in the understanding of electric power systems [2,3], telecommunication, electronics [4] and control theory.
One of the major purposes in the teaching the theory of electrical circuits consists in helping the students to better understanding and mastery of the subject.

2. ANALYSIS PHASE

UML [5] provides support for the achievement of object oriented analysis and design of high quality, which is an essential paradigm to get a robust, expandable and reusability software. The use-case diagram [6] 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. To achieve the diagram was used the ArgoUML software [7].

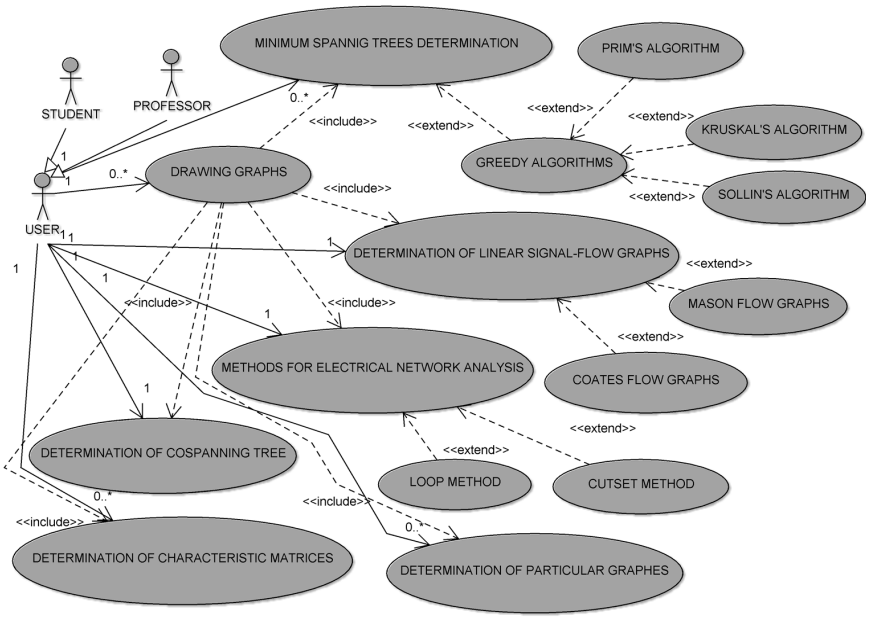

Figure 1. Use-case diagram

Use case diagram is created in an iterative manner. Use case diagram, which is shown in figure 1, includes:

- three actor - the users which represent outside parties with which the software interacts;
- 15 use cases that describe the purposes of the interactive software;
- Relations between actor and use cases, relations between use cases and relations between actors.

For every use caseofthe previousdiagramwas achievedanactivity diagram. Eachactivity diagram specifies processes and algorithms used to achieve the purpose specified bythe use-case.

3. DESIGN PHASE

Class diagram [8] representsthe main blockofobject-orientedmodeling. 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.

To memorize the vertices of the graph has been implemented "Varf" class. To memorize the arcs of the graph has been implemented "Arc" class. To memorize a graph has been implemented "Graf" class. For the drawing of a minimum spanning tree has been implemented "DesenGreedy" class. For the drawing of a graph has been implemented "DesenGraf" class.

In order to achieve the window that will form graphical user interface of the application has been implemented "Proiect" class. To simulate Prim algorithm has been implemented "Prim" class. To simulate Sollin algorithm has been implemented "Sollin" class.

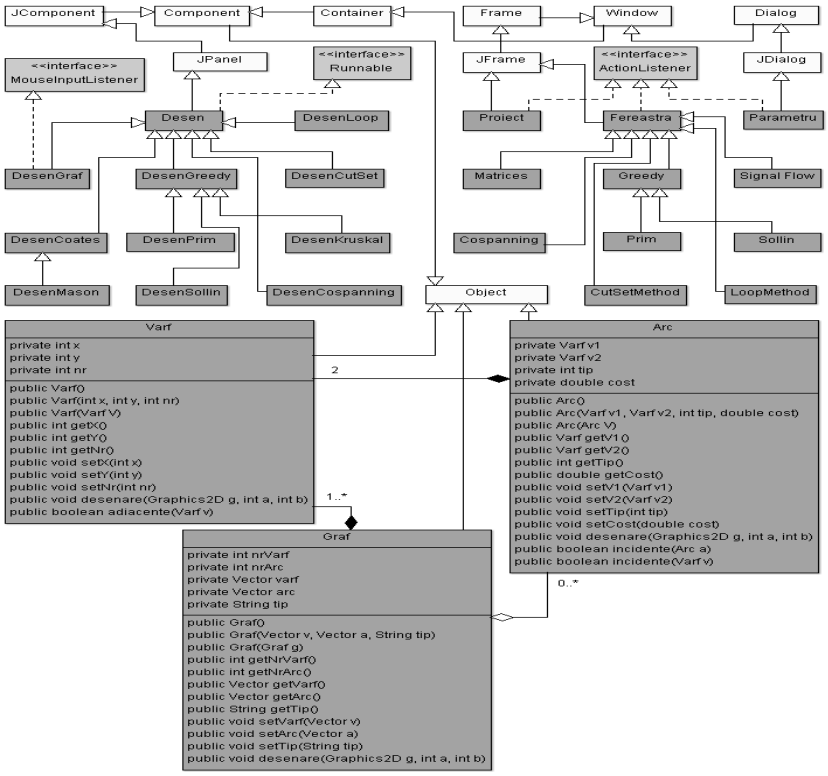

Figure 2. Class diagram

In figure 2 are presented inheritance, composition, aggregation, and realization relations. We can observe that the "Proiect" class inherit attributes and methods of the "JFrame" class, but implements the "ActionListener" interface.

"Parametru" class inherits attributes and methods of the "JDialog" class, but implements the interface "ActionListener". "Desen" class inherit attributes and methods of the "JPanel" class, but implements "Runnable" interface, and "DesenGraf" class inherits attributes and methods of the "Desen" class and implements the "MouseInputListener" interface.

In the composition relation, unlike the aggregation relation, the instance cannot exist without the party objects. Analysing figure 2 we can observe that an instance of "Arc" type consists in two objects of "Varf" type. Aggregation relation is an association where it is specified who is integer and who is a part. For example, an object of "Arc" type represents a part from an object of "Graf" type.

4. IMPLEMENTATION PHASE

The component diagram [9] enables the visualizationmodulesthat composethe softwareandthedependenciesbetween them. The diagram that is shown in figure 3 describes the collection of components that all together ensure the functionality of the interactive software.

The central component of the diagram is "Proiect.class", a component obtained by transforming using the Java compiler into executable code the "Proiect.java" component. As it can be seen the component interacts directly with component "Fereastra.class".

The component "Fereastra.class" that is obtained by transforming using Java compiler into executable code the component "Fereastra.java"interactionsdirectlywithcomponents "DesenGraf.class", "Desen.class", "Matrices.class", "Cospanning.class", "CutSetMethod.class", "LoopMethod.class", "SignalFlow.class", "Parametru.class" and"Greedy.class". Component "Greedy.class" is obtained by transforming into executable code using Java compiler the component "Greedy.java" interactions directly with components "Prim.class", "Kruskal.class"and "Sollin.class".

The component "Desen.class"thatisobtainedbytransformingintoexecutable code using Java compiler the component "Desen.java "interactions directly with components "DesenGraf.class", "DesenGreedy.class", "DesenCoates.class", "DesenCospanning.class", "DesenCutSet.class", "DesenLoop.class"and "DesenMason.class".

The component "DesenGreedy.class" that is obtained by transforming into executable code using Java compiler the component "DesenGreedy.java" interactions directly with components "Graf.class", "DesenKruskal.class", "DesenSollin.class" and "DesenPrim.class". The component "Graf.class" that is obtained by transforming into executable code using Java compiler the component "Graf.java", interactions directly with two components: "Arc.class" and "Varf.class".

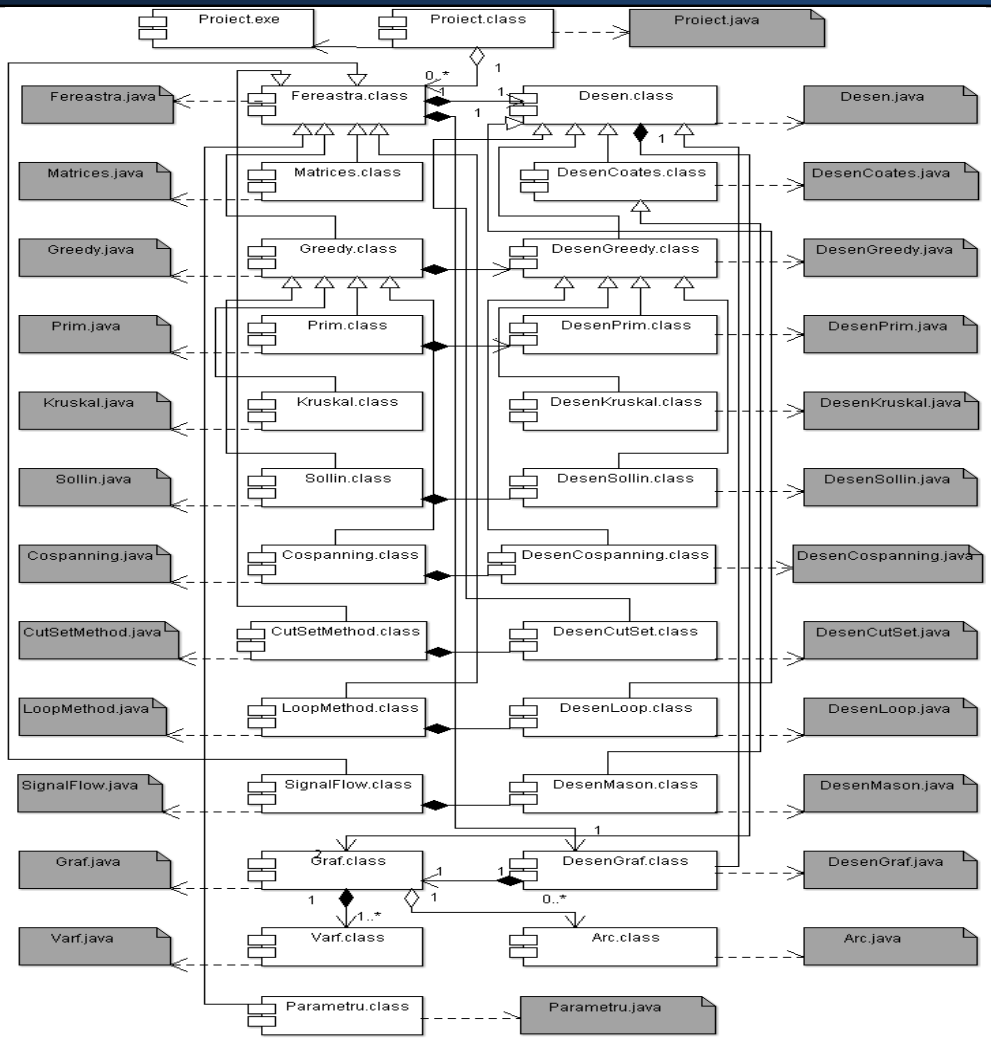

Figure 3. Component diagram

5. GRAPHICAL INTERFACE

The interactive software was implemented in Java [10,11] as independent application. By using visual simulations in computer assisted learning the efficiency of learning is increased.

Starting from specified requisites in use cases diagram (figure 1) it was designed graphical user interface of the interactive software that contains a bar with five menus.

First menu contains the following options:

- New graph – permits creating a new graph associated to an electrical circuit by specifying the vertices and arcs using mouse.
- Load graph – permits graphical representation of a graph read from an existing file.
- Save graph – permits saving the information about a current graph.
- Exit – permits to exit from an application, any unsaved graph is being lost.

The second menu contains four options that will permit the determination of the adjacency matrix, the incidence matrix, the cut matrix and the circuit matrix of the current graph associated to an electrical circuit.

Options of *Tree* menu permits the determination of the spanning tree corresponding to the current graph by selecting an algorithm and de determination of cospanning tree. The submenu contains three options corresponding to the three greedy algorithms for the determination of the minimum spanning tree: Prim algorithm [12], Kruskal algorithm [13] and Sollin algorithm[14].

Options of the next menu permits the electrical network analysis using two methods [15]: loop method and cutset method.

The last menu offers the possibility to resolve over problems corresponding to electrical circuits using Mason graphs or Coates graphs. The Coates graph [16] associated with matrix A is a weighted directed graph whose adjacency matrix is the transpose of the matrix A. For a matrix A, given in the first relation, the transpose matrix is presented in the second relation and the corresponding Coates graph is shown in figure 4.

The Coates graph associated with matrix $A+I_n$, where I_n is identity matrix, is called the Mason graph [17] associated with matrix A. For the Coates graph, presented in figure 4, the corresponding Mason graph is shown in figure 5.

$$\begin{pmatrix} 0 & -20 & 10 & -30 \\ -10 & 0 & 0 & -10 \\ 30 & -20 & 0 & 20 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad (1)$$

$$\begin{pmatrix} 0 & -10 & 30 & 0 \\ -20 & 0 & -20 & 0 \\ 10 & 0 & 0 & 0 \\ -30 & -10 & 20 & 0 \end{pmatrix} \quad (2)$$

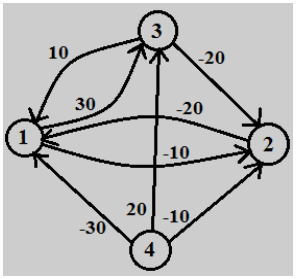

Figure 4. Coates graph

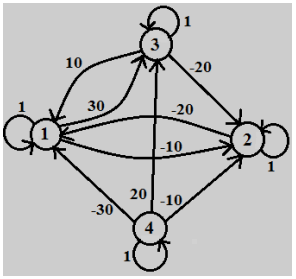

Figure 5. Mason graph

6. CONCLUSIONS

Through representation of diagrams for all three phases: analysis, design and implementation, the interactive software has been presented in a comprehensible and concise mode. The use of the Unified Modelling Language for the realization of the diagrams is characterized by rigorous syntactic, rich semantic and visual modelling support. The diagrams were made using a new onset, multidisciplinary of the informatics application, encompassing both modern pedagogy methodologies and discipline specific components. The nexus of teaching activities and scientific aims and objectives was established through the development of the new methods and the assimilation of new ways, capable of enhancing school performance, enabling students to acquire the knowledge and techniques required and apply them in optimum conditions.

REFERENCES

[1.] C. Pănoiu, R. Rob, I. Baciu, M. Pănoiu, Shunt Active Filter Command Designed in LabVIEW, International Journal of Circuits, Systems and Signal Processing,Vol. 5, pp. 513-520, 2011
[2.] C. Pănoiu, R. Rob, M. Pănoiu, Memorizing, Playing and Editing Songs Using LabVIEW Environment, AWER Procedia Information Technology and Computer Science,Vol. 2, pp. 222-227, 2012
[3.] C. Panoiu, I. Baciu, M. Pănoiu, C. Cuntan, *Simulation Results on the Currents Harmonics Mitigation on the Railway Station Line Feed Using a Data Acquisition System,* WSEAS Transaction on Electronics, Vol. 4, pp. 227-236, 2007
[4.] I. Baciu, C. Cuntan, R. Rob, C. Pănoiu, Accentuating of the Resulting Effects after Connecting Power Active Filters on Supplying Line of the Electric Traction System, International Journal of Circuits, Systems and Signal Processing, Vol. 5, pp. 505-512,2011
[5.] J. Odell, Advanced Object Oriented Analysis& Design using UML, Cambrige University Press, 1998
[6.] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, 2001
[7.] http://argouml.tigris.org
[8.] G. Booch, J. Rumbaugh, I. Jacobson, "The Unified Modeling Language User Guide", Addison Wesley, 1999
[9.] J. Rumbaugh, I. Jacobson, G. Booch, The Unified Modeling Language Reference Manual, Addison Wesley, 1999
[10.] J. Skansholm, Java from the Beginning, Addison Wesley Press, 2005
[11.] J. Bloch, Effective Java, Pearson Education, 2008
[12.] S. Even, Graph Algprithms, Cambridge University Press, 2012
[13.] J. Harris, J. Hirst, M. Mossinghoff, Combinatorics and Graph Teory, Springer-Verlag Press, 2008
[14.] M. Daniel, Graph Theory, Mathematical Association of America, 2008
[15.] J. Gross, Graph Theory and its Applications, Taylor & Francis Ltd, 2005
[16.] L. Surhone, M. Timpledon, S. Marseken, Coates Graph, VDM Publishing, 2010
[17.] W. C. Chen, Graph Theory and its Engineering Applications, Advanced Series in Electrical and Computer Engineering, World Scientific Publishing, 1997