Interactive Software for the Study of Electrical Circuits Using Graphs Theory

ANCA-ELENA IORDAN, MANUELA PANIOIU, IONEL MUSCALAGIU, RALUCA ROB
Polytechnic University of Timişoara, Engineering Faculty of Hunedoara,
Revoluţiei 5, 331128 Hunedoara ROMANIA
anca.iordan@fih.upt.ro

Abstract: - This paper presents the necessary steps required for object oriented implementation of new software used in the study of electrical circuits with assistance of graphs theory. The design of the software is achieved through specific UML diagrams representing the phases of analysis, design and implementation, the software thus being described in a comprehensible and concise mode. The new software is intended for teachers and students who teach, explore or evaluate electric circuits, because electrical engineering domain, especially electrical circuits, is difficult to understand for most students.

Key-Words: - Electrical Circuits, Graph Methods, UML, Java

1 Introduction
Circuit theory [1] is an important and the oldest side of electrical engineering. In engineering education it is considered significant to learn electric circuit theory. Grasping concepts and phenomena from circuit theory is important in understanding electric power systems [2,3], telecommunication, electronics and control theory.

One of the significant goals of a circuit theory teacher is to help students acquire a functional understanding of the subject.

2 Development phases of didactic software
UML [4] offers support for realizing of some high grade object orientated analysing and designing, that represent key elements in order to obtain solid, extensible and reusable software. Elaboration of the interactive environment was made in concordance with a specification that shows all desired elements.

These elements contain software requirements, making difference between obligatory and optional ones, and also contain the functionality and conditions that have to be accomplished in order to conform to the standards.

2.1 Analysis stage
Using Unified Modelling Language, didactic software analysis consists in realization of use case diagram and activity diagrams. To achieve diagrams was used the ArgoUML software [5].

The interactive software is described in a comprehensible and concise mode as representing the use cases [6]. Each case describes the interactions of user and software.

Use case diagram is created in an iterative manner. First, there were identified the actors, starting from problem formulating by identifying the role played by different persons and external resources that are implicated in interactions. Identifying the uses cases and the relations between them was based on the analysing the responsibilities accomplished by every actor and also the global specification that are referring to the functional requisites. Use case diagram representation is shown in figure 1.

Diagram advanced defines the domain of software, allowing visualization of the size and sphere of the action for the entire development process. This includes:
- One actor - the user that have external entities with which the software interacts;
- 15 use cases that describe the functionality of the interactive software;
- Relationships between user and use cases, relationships between use cases (dependency and generalization relationships) and relationships between actors.

After realizing the user requisites specifications, there appears the problem to look for details, using activity diagrams [7]. For each use case presented in the previous diagram is built an activity diagram. Each diagram shall specify the processes or algorithms that are behind use case analysis. Figure 2 shows the corresponding activity diagram for the use case Greedy algorithms, diagram which contains seven activities and a decision block.
2.2 Design phase
Object orientated methods introduced the representation of the static structure of software using classes and relations among them. This idea is succeeded from entity-relation diagram.

2.2.1 Class diagrams
Conceptual modelling allows identifying the most important objects for the interactive software [8]. Inheritance was not used only as a generalization device, which is when derived classes are specializations of the base class. 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’s algorithm has been implemented “Prim” class. To simulate Sollin’s algorithm has been implemented “Sollin” class.

In figure 3 are presented inheritance relationships and realization relationships. 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.
2.2.2 State diagram

Corresponding objects of the classes presented have behaviour and internal state, in other words, fulfil actions and hold information. To understand them we need to develop the state diagrams. UML state diagrams [9] describe the different states and it can find an object and the transitions between these states.

In figure 4 is presented the state diagram corresponding to an instance of Greedy class, a class which implements Runnable interface corresponding to the execution threads.

2.2.3. Sequence diagrams

Sequence diagram emphasizes the temporal aspect [10], being suitable for real-time specifications and complex scenarios. These diagrams determine the objects and classes involved in a scenario and sequence of messages sent between objects necessary to execute script functionality.

Diagram presented in figure 5 shows the interactions between objects that are aimed at determining minimum spanning tree using the Prim algorithm. We can observe that there are interactions between 17 objects of which the type of the object Proiect is already created, and objects of Prim, JButton, DesenGraf, DesenPrim, Graphics2D, Graf, Varf, Arc and Thread type will instantiate during interactions. At the beginning the execution control is taken by Proiect object that creates an instance to Roy class, this object takes the control of the interactions.

There will be instancing four JButton objects, a DesenGraf object and a DesenPrim object. Instancing the DesenGraf object has as effect creating a Graphics2D object and a Graf object. Instancing the DesenPrim object permits to create another Graphics2D object.
Fig. 5 Sequence diagram to determine minimum spanning tree using Prim algorithm
3 Graphical interface of the software

The interactive software was implemented in Java [11] as independent application. The application can easily convert in a Java applet. 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 environment 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 the mouse (figure 7).
- 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 (figure 8), 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 (figure 9) and determination of cospanning tree (figure 10). The submenu contains three options corresponding to the three greedy algorithms for the determination of the minimum spanning tree: Prim algorithm, Kruskal algorithm and Sollin algorithm.

Options of the next menu permits the electrical network analysis using two methods: 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.

4 Conclusion

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:
[5]. http://argouml.tigris.org