A new approach in the design of an interactive environment for teaching Hamiltonian digraphs

A E Iordan[1], **M Panoiu**[1]

[1] Electrical Engineering and Industrial Informatics Department, Engineering Faculty of Hunedoara, Polytechnic University of Timisoara

E-mail: anca.iordan@fih.upt.ro

Abstract. In this article the authors present the necessary steps in object orientated design of an interactive environment that is dedicated to the process of acquaintances assimilation in Hamiltonian graphs theory domain, especially for the simulation of algorithms which determine the Hamiltonian trails and circuits. The modelling of the interactive environment is achieved through specific UML diagrams representing the steps of analysis, design and implementation. This interactive environment is very useful for both students and professors, because computer programming domain, especially digraphs theory domain is comprehended and assimilated with difficulty by students.

1. Introduction

Graphs theory [1] represents the mathematic study of abstract relationship structure between the objects. Although graphs are themselves purely theoretical, their ability to model pair-wise relationships in systems of arbitrary complexity yields abundant direct correspondence with numerous important physical systems in the real world. More than this, the fundamental graph structure permits a geometrical view of them in many ways. If they are taken together, these two properties suggest that graph theory teaching and also researching in this domain can obtain benefits using an interactive environment addressed Hamiltonian graphs study.

Starting with this necessity, we designed an interactive environment that can be used by students and also by graphs theory beginners, but strong enough to help the researchers in graphs theory domain. In educational scope, the interface and the visual components will lead the students and the beginners in this domain to a better comprehending of the next algorithms using for Hamiltonian trails and circuits [2] determination: Chen algorithm [3], Kaufmann algorithm [4], Foulkes algorithm [5] and Albertson algorithm [6].

2. Interactive Environment Development

Using UML, interactive environment analysis consists in accomplishment of use case diagram [7]. The interactive environment is described in a comprehensible and concise manner as representing the use cases. Each case specifies the interactions among users and environment. UML uses case diagram is created in an iterative approach (figure 1). Diagram performed defines the environment domain, allowing visualization of the size and sphere of the action for the entire realization process. This includes:

- Three actors - the users that have external entities with which the environment interacts;
- Seven use cases that describe the functionality of the interactive environment;
- Relationships between users and use cases (association relationships), relationships between use cases (dependency and generalization relationships), and relationships between actors.

Oriented object methodologies [8] introduced the representation of the static structure of application using classes and relations between them. Through oriented object modelling of the Java application have been identified classes and relationships of inheritance and realization which are presented in figure 2.

- To memorize the vertices of the graph has been implemented “Vertex” class.
- To memorize the arcs of the graph has been implemented “Arc” class.
- To memorize a graph has been implemented “Graph” class.
- For the drawing of a Hamiltonian trail or circuit has been implemented “Hamiltonian” class.
- For the drawing of a graph has been implemented “GraphRepresentation” class.
- For the determination of the Hamiltonian trail using Chen algorithm has been implemented “HamiltonianChen” class.
- For the determination of the Hamiltonian trails or circuits using Albertson algorithm has been implemented “HamiltonianAlbertson” class.

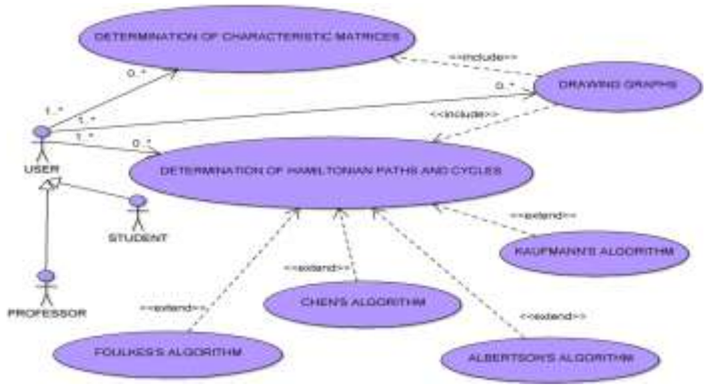


Figure 1. Use cases diagram

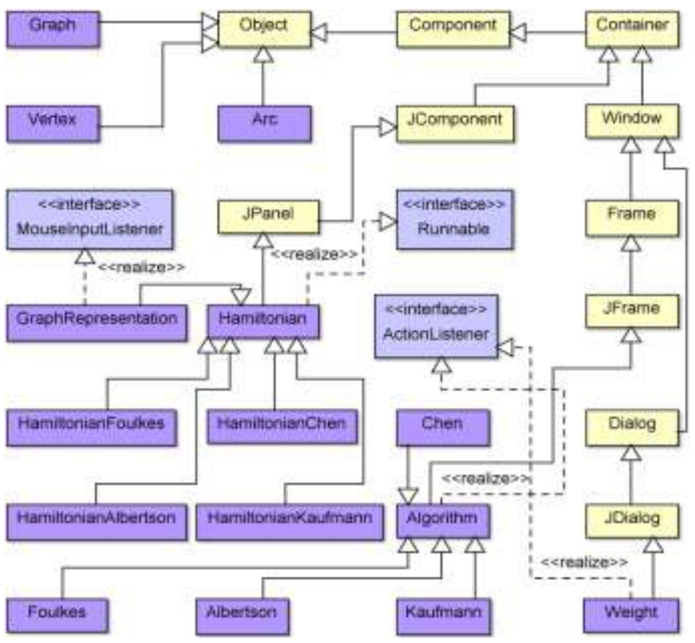


Figure 2. Diagram that presents relationships between classes

- For the determination of the Hamiltonian trails or circuits using Kaufmann algorithm has been implemented “HamiltonianKaufmann” class.
- For the determination of the Hamiltonian trails or circuits using Foulkes algorithm has been implemented “HamiltonianFoulkes” class.
- In order to achieve the window that will form graphical user interface of the application has been implemented “Algorithm” class.
- To simulate Chen algorithm has been implemented “Chen” class.
- To simulate Albertson algorithm has been implemented “Albertson” class.
- To simulate Foulkes algorithm has been implemented “Foulkes” class.
- To simulate Kaufmann algorithm has been implemented “Kaufmann” class.
- To introduce the weight of an arc has been implemented “Weight” class.

We can observe that the “Algorithm” class inherit attributes and methods of the “JFrame” class, but implements the “ActionListener” interface. “Weight” class inherits attributes and methods of the “JDialog” class, but implements the interface “ActionListener”. “Hamiltonian” class inherit attributes and methods of the “Jpanel” class, but implements “Runnable” interface, and “GraphRepresentation” class inherits attributes and methods of the “Hamiltonian” class and implements the “MouseInputListener” interface.

Between instances of the classes presented in figure 2 there are especially composition and aggregation relationships. In the composition relationship, unlike the aggregation relationship, the instance cannot exist without the party objects. Analysing figure 3 we can observe that an instance of “Hamiltonian” type consists in two objects of “Graph” type, one of “Graphics2D” type and the other of “Thread” type. Aggregation relationship is an association where it’s specified who is integer and who is a part. For example, an object of “Arc” type or represent a part from an object of “Graph” type.

Collaboration diagrams describe the behaviour of a set of objects in a certain context with an emphasis on organizing the objects involved in the interaction. Diagram presented in figure 4 renders the interactions between objects that allow determining the Hamiltonian trail using Foulkes algorithm.

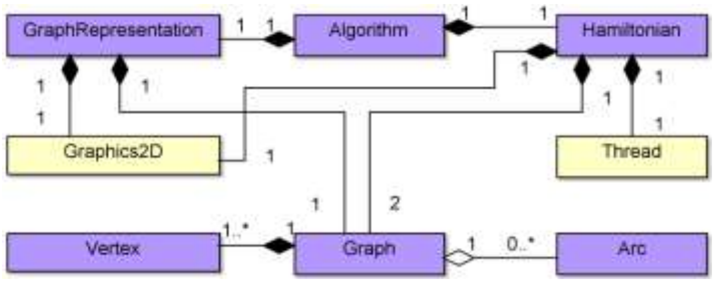


Figure 3. Diagram that presents relationships between instances of classes

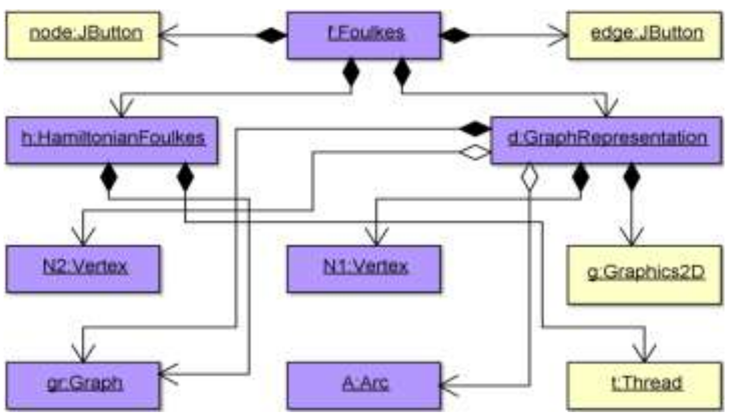


Figure 4. Collaboration diagram to determine the Hamiltonian trail using Foulkes algorithm

Component diagram [9] is analogue to packages diagram, allowing visualization of how the interactive environment is divided and the dependencies among class modules. The diagram presented in figure 5 describes the set of components that together provide environment performance.

Central component of the diagram is “Algorithm.class”, a component obtained by transforming by the Java compiler into executable code of the “Algorithm.java” component. As can be seen that component interacts directly with components “GraphRepresentation.class”, “Hamiltonian.class”, “Kaufmann.class”, “Foulkes.class”, “Chen.class” and “Albertson.class”.

Component “Hamiltonian.class” that is obtained by Java compiler from component “Hamiltonian.java” in executable code interactions directly with components “Graph.class”, “GraphRepresentation.class”, “HamiltonianAlbertson.class”, “HamiltonianKaufmann.class”, “HamiltonianFoulkes.class” and “HamiltonianChen.class”.

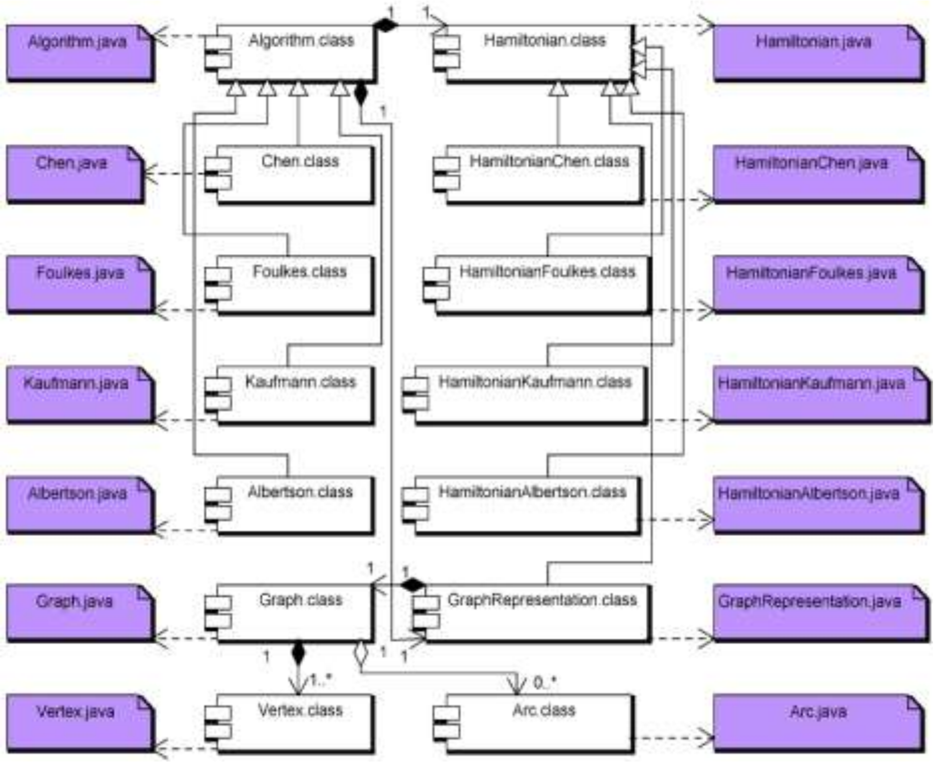


Figure 5. Component diagram

3. Interactive environment graphical interface

The interactive environment was implemented in Java [10] 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 particularized requirements in use case diagram (figure 1) it was designed graphical user interface of the interactive environment that contains a bar with three menus as in figure 6. In the same figure is represented the Hamiltonian trail obtained into a digraph with twelve vertices and four strongly connected components.

First menu contains four options:

- New graph – permits creating a new directed graph by specifying the vertices and arcs using the 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 digraph being lost.

The second menu contains five options that will permit determination of adjacency matrix, incidence matrix, distance matrix, Latin matrix and reduced Latin matrix for the current graph. The final menu, “Hamiltonian Trails”, contains four commands and permits determination of the Hamiltonian trails and circuits in current graph only if this one exists, using one of the following four algorithms: Chen algorithm, Kaufmann algorithm, Albertson algorithm or Foulkes algorithm.

For the exemplifications of the simulation, is presented Chen algorithm. In figure 7 are presented steps that are made in determination of the Hamiltonian trail in a digraph with 8 vertices without circuits. Corresponding to Chen theorem [3], if the Hamiltonian trail exists, then it is unique. Figure 8 presents the obtained Hamiltonian trail for the digraph used for exemplification.

Hereinafter is simulated Kaufmann algorithm [4], also called Latin multiplication algorithm. This algorithm uses two matrices: the Latin matrix and the reduced Latin matrix.

For the digraph with 5 vertices which is drawn in window presented in figure 9, in which can be observed these matrices by selecting corresponding options of the second menu. In figure 10 are presented one Hamiltonian trail and five Hamiltonian circuits obtained for the digraph with 5 vertices. The window in which are presented these results is represented in figure 11.

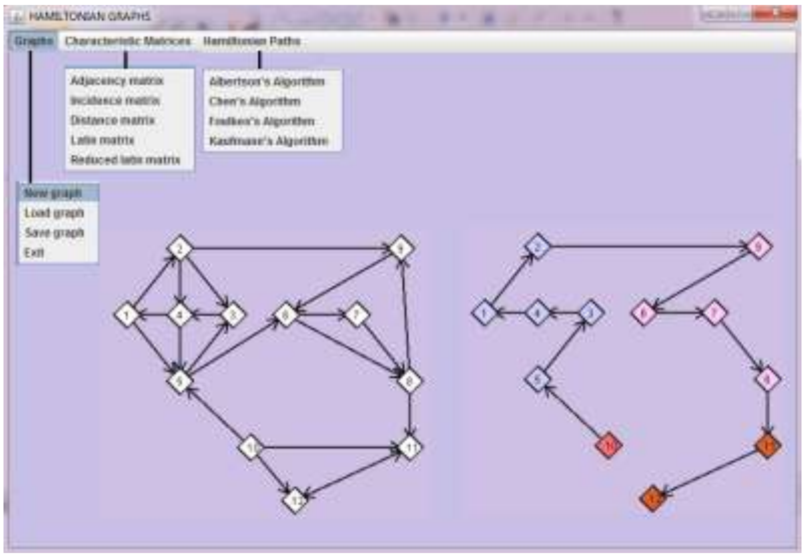


Figure 6. Graphical user interface

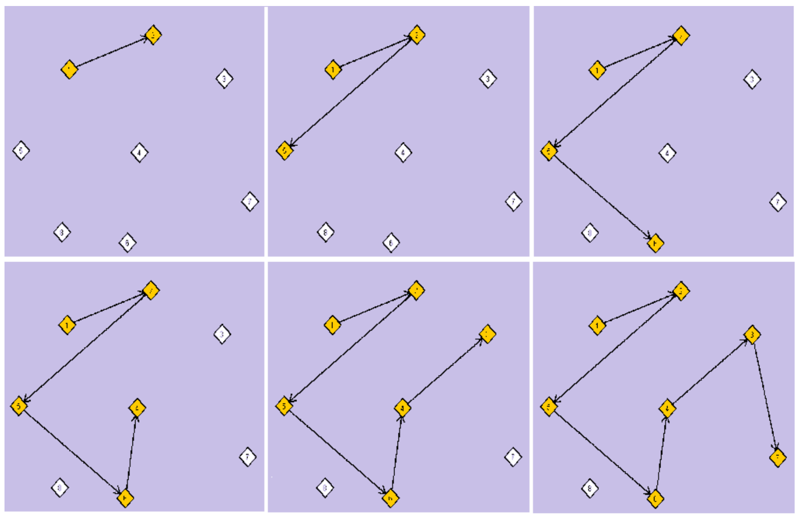


Figure 7. Implementation steps in determining the Hamiltonian trail

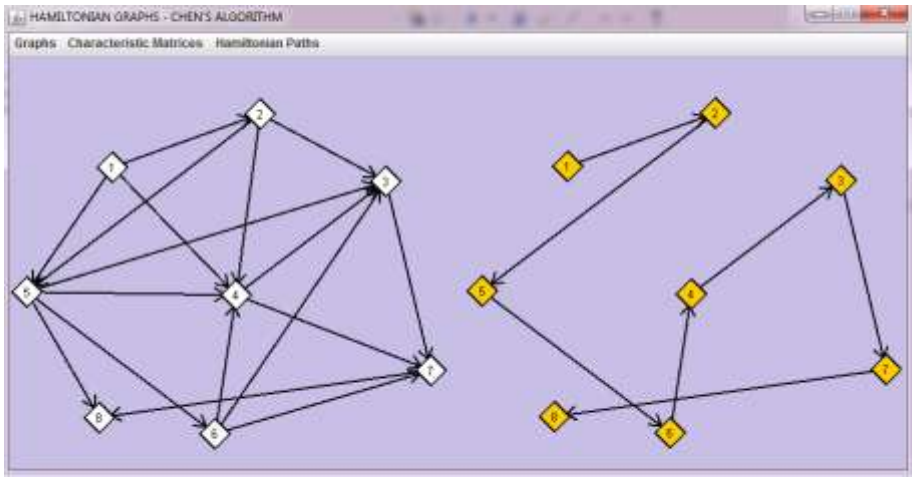
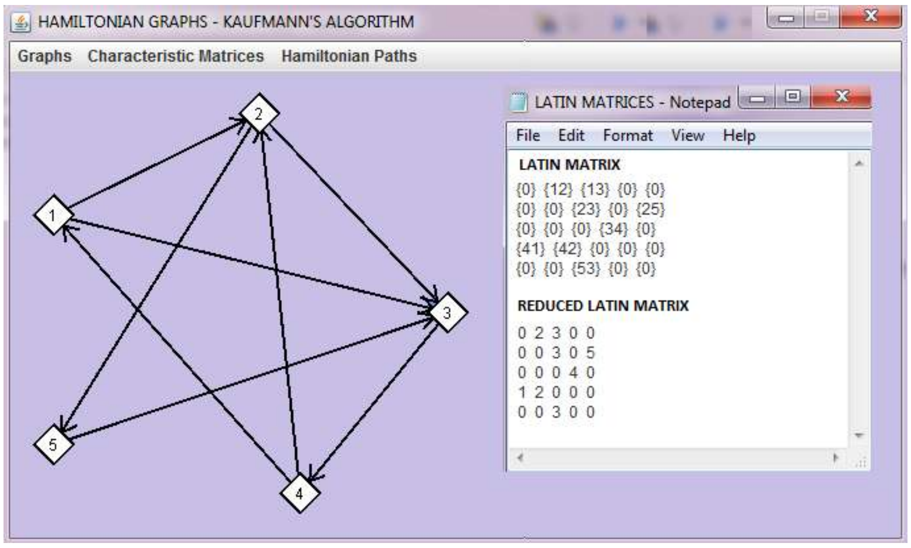

HAMILTONIAN GRAPHS - CHEN'S ALGORITHM
Graphs Characteristic Matrices Hamiltonian Paths

Figure 8. Hamiltonian trail using Chen algorithm


HAMILTONIAN GRAPHS - KAUFMANN'S ALGORITHM
Graphs Characteristic Matrices Hamiltonian Paths
LATIN MATRICES - Notepad
File Edit Format View Help

```
LATIN MATRIX
{0} {12} {13} {0} {0}
{0} {0} {23} {0} {25}
{0} {0} {0} {34} {0}
{41} {42} {0} {0} {0}
{0} {0} {53} {0} {0}

REDUCED LATIN MATRIX
0 2 3 0 0
0 0 3 0 5
0 0 0 4 0
1 2 0 0 0
0 0 3 0 0
```

Figure 9. Latin matrix and reduced Latin matrix associate to a digraph

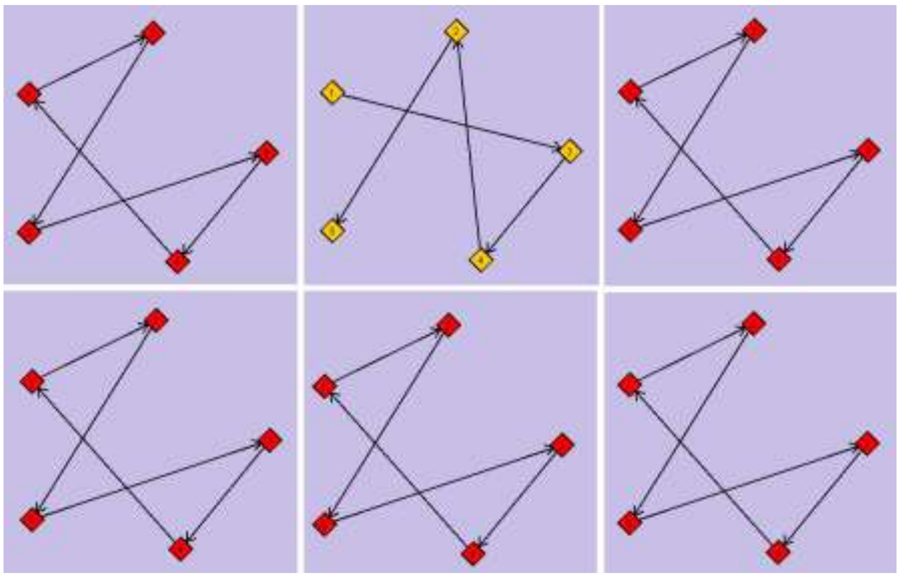
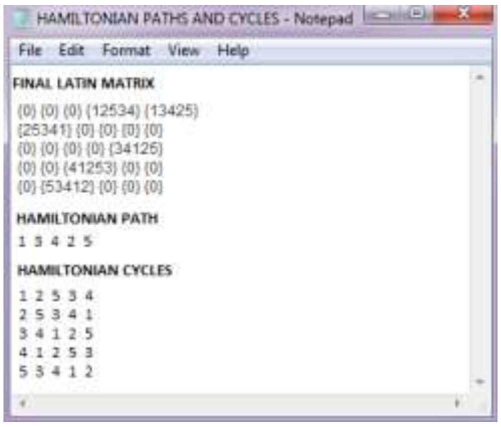


Figure 10. Hamiltonian trail and circuits using Kaufmann algorithm


HAMILTONIAN PATHS AND CYCLES - Notepad
File Edit Format View Help

```
FINAL LATIN MATRIX
 {0} {0} {0} {12534} {13425}
 {25341} {0} {0} {0} {0}
 {0} {0} {0} {0} {34125}
 {0} {0} {41253} {0} {0}
 {0} {53412} {0} {0} {0}

 HAMILTONIAN PATH
 1 3 4 2 5

 HAMILTONIAN CYCLES
 1 2 5 3 4
 2 5 3 4 1
 3 4 1 2 5
 4 1 2 5 3
 5 3 4 1 2
```

Figure 11. The list of Hamiltonian trail and circuits

4. Conclusions

The main goal in realizing of this article is designing and implementation of an interactive environment that has to lead the user to accomplish experience in comprehending and controlling acquaintances in Hamiltonian digraphs theory domain and to offer efficient and convenient access to the latest information.

The theme that is treated in this article is actually; graphs theory is an important component in developing the experience of the young mathematicians and engineers. The informatics application follows these actual problems and boards it in a new manner using modern educational technologies.

References

[1] Chartrand G, Lesniak L and Zhang P 2010 *Graphs & Digraphs* (London: Chapman and Hall/CRC)
[2] Gould R 2003 Advances on the Hamiltonian Problem – A Survey *Graphs and Combinatorics* **19** 7
[3] Chen C and Quimpo N 1979 On some classes of Hamiltonian graphs *Southeast Asian Bulletin of Mathematics* **2** 252
[4] Kaufman A and Malgrange Y 1963 Recherche des chemins et des circuits Hamiltoniens d'un graphe *Revue Française de Recherche Opérationnelle* **26** 61
[5] Foulkes J 1960 Directed Graphs and Assembly Schedules *Proceedings of Symposia in Applied Mathematics* **10** 218
[6] Albertson M 1987 Finding Hamiltonian Cycles in Ore Graphs *Congr. Numer.* **58** 25
[7] Fowler M and Scott K 2000 *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (Reading: Addison-Wesley)
[8] Dennis A, Wixom B and Tegarden D 2012 *System Analysis and Design with UML* (San Francisco: Wiley)
[9] Gomaa H 2011 *Software Modelling and Design: UML, Use Cases, Patterns and Software Architectures* (New York: Cambridge University Press)
[10] Bloch J 2008 *Effective Java* (Boston: Addison-Wesley)