Nabil Arman et al. / (IJCSE) International Journal on Computer Science and Engineering
Vol. 02, No. 03, 2010, 607-613

A Systematic Approach for Constructing Static Class Diagrams from Software Requirements

Nabil Arman
Department of Mathematics and Computer Science
Palestine Polytechnic University
Hebron, Palestine

Khalid Daghameen
Department of Computer and Electrical Engineering
Palestine Polytechnic University
Hebron, Palestine

***Abstract*—The trend towards the use of object-oriented methods for software systems development has made it necessary for the use of object-oriented approaches in object-oriented software systems development. Class diagrams represent an essential component in any object-oriented system design. The development of such class diagrams in a systematic way is very crucial in an object-oriented development methodology. The main principles used in obtaining these class diagrams in a systematic way are described since class diagrams are very essential in object-oriented development practice.**

Keywords- Class diagrams, software requirements, object-oriented methods

I. INTRODUCTION

Many object-oriented development methodologies have been developed in the past two decades. These methodologies include Object Modeling Technique (OMT), OOSE and OPEN [1-3]. Commercial methodologies that can be purchased are available today, including Rational Rose and Rational Unified process[4,5]. A System Development Methodology (SDM) has been defined as "…a systematic approach to conducting at least one complete phase (e.g., requirements analysis, design) of system development, consisting of a set of guidelines, activities, techniques and tools, based on a particular philosophy of system development and the target system [6]. From this definition, it is generally recognized that there is a great emphasis on "being systematic" for any phase of system development methodology.

The requirements engineering practice using object-oriented development methodologies has been well-documented in the literature [7-9]. In [7], a number of case studies were presented. In these case studies, the use of a systematic approach for obtaining static class diagrams was absent. In [8], the role of object-oriented process modeling in requirements engineering phase of information systems development is presented. In [9], object-oriented requirements engineering and management is explained. These approaches are categorized into two main categories according to Edward Yourdon. The two categories are evolutionary to Object-Oriented analysis and Design and revolutionary to Object-Oriented analysis and Design. Rambuagh and Martin-Odell are two approaches of evolutionary approach. Booch and Wrifs-Brock are two approaches of revolutionary approach [10-12].

Some approaches to Object-Oriented analysis and design were based on use cases after the introduction of UML. Dough Rosenberg and Matt Stephen proposed an approach for Object modeling based on use cases [13]. Their approach includes the static and dynamic models of Object-Oriented. While our approach in this paper is concerned with static models, their approach depends mainly on software engineers/practitioners. Our new approach is less dependent on humans.

This paper presents a systematic approach consisting of a set of guidelines and techniques for obtaining static class diagrams from software requirements. The guidelines are explained in detail and two case studies are presented to illustrate the approach. Research into a systematic approach for obtaining class diagrams practices is therefore needed to improve understanding of object-oriented system development methodologies in general and class diagrams in particular.

II. STATIC CLASS DIAGRAMS CONSTRUCTION

A static class diagram is constructed from software requirements and problem statements by following the steps below [14]:

1. Identify the list of all nouns and noun phrases from the problem statement. These nouns and noun phrases represent classes and adjectives represent attributes. A software engineer/practitioner obtains such a list from the problem statement, use cases, actor-goal list or application narrative description.
2. Identify the list of verbs which represent actions in the software requirements. These actions represent methods in the static class diagram.
3. Make a refinement for the nouns determined in step 1. Such nouns represent values for attributes, or nouns that are identical to other nouns. This step is important to ease the process of having the static diagram, but it is not necessary; nouns that are not related to the problem statement are isolated classes in the static diagram. This means that they are not

part of the system. Nouns that are identical according to the problem statement appear in the static diagram with the same attributes and associations as other nouns. From that, one also needs to pick one of such classes. For example, in a problem statement concerning a university, a teacher may be called a teacher, a professor or an instructor. Such nouns mean the same thing in this problem statement .

4. Arrange the nouns from step 2 into a matrix keeping the order of such nouns the same in the rows and columns. The nouns that appear in the left column is called "left cell" in the paper and the ones that appear in the top row is called "top row" In this early step, one keeps the nouns and noun phrases at the top of the list and the adjectives at the bottom of the list.
5. Fill up the table with the letters (I, H, P, U or keep it empty) by answering YES/NO questions. The four questions are: Is the "left cell" a "top row"?, Has a "left cell" a "top row"?, Is a "left cell" a "top row"? and Does the "left cell" use the "top row"? consecutively. It is important to realize that once a cell is occupied by a letter, one doesn't need to do the rest of the questions for that particular cell.
6. The answers of the four questions must be according to the problem statement, use cases, actor-goals or application narrative description. In a university problem statement, the answer for a question such as "Is the graduate-student a student?" is yes. The answer for "Has student information a date?" is yes. On the other hand, an answer for the question such as "Is a student a graduate-student?" is no, since there is a student who is not a graduate student for that problem domain.
7. Identify those nouns which are classes and those which are attributes. Any row that has an "H" letter is a class, as for those rows which don't have an "H" letter is left to the software practitioner to identify. Rows that have more than one "I" is a class. The row which has a single "I" is also a class. As for the adjectives, all of them are attributes.
8. Rearrange the nouns again in the table taking into account that all candidate classes must be sorted according to the number of "H" letters in each row and the others are kept to the end of the list for the classes, while the attributes and methods should occupy the tail of the list. Fill up the lower half of the matrix with letters by asking the simple four questions again. In other words, the arrangement takes into account that the diagonal contains "I's" and all the cells that are above the diagonal should have the "H" and the lower part has the "P's".
9. Each identified class in the table should have a corresponding class in the class diagram. If one is using UML models, a standalone class is drawn by the name of the class.
10. For those rows that contain the same letter, distinguish between cells if they mean different things, even if the difference is less significant. You can use any identification symbol such as the prime "'".
11. Identify the hierarchy by connecting classes using the cells containing "I" letters. An L-shape is formed between two candidate classes. For the hierarchy stair shape like is obtained by forming more than one L-shape from those "I" cells.
12. An UML static diagram is obtained from the information obtained from the previous steps.
13. For the "P" cells, arrange them into groups according to the groups of "I's". For example, a stand alone "P" in a row means that this is an attribute; "left cell" which is an attribute in the class "top row". For two "P" cells in a row, find the equivalent of "I's" that form the lower part of L-shape from a row in the top. Those are part of the lower L-shape and this attribute is part of the class which is in the higher L-shape of "I's". A group of three P's is the same and you need to find the Upper L-shaped class upon identifying the Attribute position of an attribute.
14. For those "P" cells, which are classes, or those classes that have "has-a" associations with another class. The first step is to locate the position of the class which should be connected to another class. This is done using the same as step 11.

III. CASE STUDIES

Two case studies are presented here to explain the main guidelines of our approach:

Case study I: Assume that the following text/problem statement is taken from a software requirements document:

> Vessels are one kind of containers that contain liquids, vessels are different in shapes and sizes. Each vessel has a capacity which is the maximum quantity of the substance that is contained using a vessel. The amount of substance in a vessels wobbles between zero to the maximum of the capacity of a vessel. All types of vessels have a capacity computed differently. When someone fills up a vessel, he/she is going to add some substance to the existing substance in the vessel.
>
> A lot of shapes of vessels are exist, but we are concerned with few of them, some of them which are rectangular tanks, this type has a rectangular base and a height, the rectangular base is identified by its length and width. A cubic tank which is the same as a rectangular one except that the base is a square base and both length and width have the same value. A cylindrical tanks has a circular base and height. A circular base is identified by its radius.

To construct a static class diagram form the above problem statement, the steps are applied as follows:

- The first step in analyzing the problem statement is to identify a list of nouns and noun phrases, such as vessel, container, size, capacity, tank, rectangle, square, height, length and width.
- The next step is to take some of those nouns out of the list. Such nouns are those which are identical or they refer to the same thing. In the problem statement, tank, vessel and container are synonyms to the same thing, so vessel is picked. Some nouns are not part of the problem although they are mentioned in the problem statement. A software engineer/practitioner must take them out of the list such as substance and liquid. In this problem, liquid is not part of the problem. Even if the software engineer/practitioner choose not to take them out. Such classes will be an isolated classes in the static class diagram.
- All names that are generated from the previous step are arranged in an adjacent matrix as in Figure 1.
- Fill up the table (below the diagonal) with letters (I, H, P). To have the letter "I", The answer for the following question must be yes, Is “Left cell” a “top row”? For example “Is a rectangular tank a vessel? “. The answer is yes. Therefore, the cell that corresponds to rectangular tank with vessel should have an “I”.
- If the answer for a question is no, the cell remains empty. Like “Is a capacity a vessel?” The answer is no, therefore, the cell remains empty.
- A question is generated to have an answer either yes or no. The question is “Has (left row) a (top cell)?” or “Is the (left row) part of a (top cell)?” If the answer to any of them is yes, then the corresponding cell is marked with a “P”.
- The next question to fill up the table is “Has (left row) a (top row)?” If the answer to this is yes, write an “H” in the cell corresponding to the “left row” and “top cell”. In the example, some cells are changed to “H”.
- The matrix becomes like Figure 1. Any row that has a P letter means that this is an attribute. A row which has an one “I” is a class. Others would be classes if the row is an adjective, that means it is an attribute. If it is a verb, then it is a method.
- Group each of the "P" cells together using the prime notation; for example, the capacity of the rectangular vessel and cubic vessel is computed by the same way, while the capacity of the cylindrical vessel is different, thus, the "P" cells of the rectangular and the cubic should have one prime, while the cylindrical "P" cell has a double prime.
- With a pencil, form the English letter L by connecting I-cells together. You need to form a stair-shape so you can have the hierarchy form of the class diagram. The first path is (Vessel→Rectangular Tank, Rectangular Tank→Cubic Tank) and the next path is (Vessel→cylindrical Tank) as shown in Figure 2.

	Vessel	Amount	Rectangular tank	Cubic tank	Cylindrical tank	Length	Height	Width	Capacity	Radius	
Vessel	I										C
Amount	P	I									A
Rectangular tank	I	H	I						H		C
Cubic tank	I	H	I	I					H		C
Cylindrical tank	I	H			I				H		C
Length			P	P		I					A
Height			P	P	P		I				A
Width			P					I			A
Capacity	P		P’	P’	P’’				I		A
Radius					P					I	A

Figure 1. Building Object-Oriented information for case study I

	Vessel	Rectangular tank	Cubic tank	Cylindrical tank	Length	Height	Width	Capacity	Amount	Radius	
Vessel	I										C
Rectangular tank	I	I						H	H		C
Cubic tank	I	I	I					H	H		C
Cylindrical tank	I			I				H	H		C
Length		P	P		I						A
Height		P	P	P		I					A
Width		P					I				A
Capacity	P	P’’	P’’	P’				I			A
Amount	P	P	P	P					I		A
Radius				P						I	A

Figure 2. Class relation for case study I

- Draw UML shapes of the hierarchy. This is shown in Figure 3 for this case study.
- In another table, rearrange the table so that the classes are at the top and the attributes are at the bottom and apply the same for all of them as shown in Figure 2. The rearrangement should take into consideration the number of “I’s” within each row, and they should be sorted in ascending order.
- For the "P" cells, form a region of one cell, two cells, three cells, four cells and so on.
- The two attributes capacity and amount can be in the top of the class hierarchy as you form a rectangular region of four cells and that region intersects with the top hierarchy which is vessel. This region covers the lower part of all L-shape. Such attribute must be at the top of the hierarchy
- Length has two boxes and should be in the Rectangular tank class as in Figure 2.
- Height has two regions, one with two boxes and the other with one as we can’t form a region with three boxes that forms a lower part of L-shape that are related in the hierarchy. From this, one can notice that one is in the rectangular tank class and the other in the cylindrical tank class.

- For the capacity, which is an attribute (method) because we are talking about a process of computing the capacity, the number of the prime notation determines the number of the groups one can form in the "P" cells. In the example there are three. The double primes are together, which means that the computation of the capacity should be in the rectangular tank class, while the other two should be in the cylindrical and the abstract class (vessel).

Case study II: Assume that the following text/problem statement is taken from a software requirements document:

Simple electrical software is used to draw the basic components of the electrical network such as resistor, capacitor, inductor as well as the voltage source. Each component is drawn on a specific location of the drawing sheet. It occupies a rectangular area of the drawing sheet. The drawing sheet has a width and a height. The software will not draw any component outside the drawing sheet. The location is determined through the x and y coordinates.

Each and every component has a caption. It is used to name that component, which means that the caption is a text property. The caption is drawn on the drawing sheet, so, it requires a location and it occupies and area as well as the component itself. The component is placed on the drawing sheet either vertically or horizontally; this is called the orientation of a component. The orientation is applied on the caption as well as the value. The value of the component is shown also on the drawing sheet. Therefore, it needs a location it occupies an area. The unit of the value determined through the type of the object which is associated with; it means the component is a resister, the value unit is Ohms and the same applies for the others.

The drawing program allows the user to move the component from one place of the drawing sheet to another. When it moves the component, it moves with the value and caption of that component. While if the move is for the caption, only the caption is moved.

Each electrical component has two connection points. They are named nodes. Each node has its own voltage value. Each node has its own caption and a value for the voltage. The electrical network consists of a list of components mentioned above.

To construct a static class diagram form the above problem statement, the steps are performed as follows:

- The first step is to construct a list of all nouns and noun-phrases. The verbs which represent methods are also determined. For this case study, the nouns are electrical software, component, electrical network, resistor, capacitor, inductor, voltage source, draw, location, drawing sheet, occupy, rectangular area, width, height, caption, vertical, horizontal, value, unit, move, connection point, node, x-coordinate and y-coordinate.
- The next step is to refine the nouns, noun phrases, verbs, values and others. For this case study, it is clear that horizontal and vertical are values for an attribute called orientation. Connection point and node refer to the same thing. The verb “occupy” means that it occupies a rectangular area on the drawing sheet; which means we should keep only one. The rectangular area is identified by a width and height, which is also in the list. From this, a number of items of the list should be taken out.
- Fill up the table using the four questions mentioned in step 5. The result of this process is shown in Figure 4.
- Identify all the entities in the software requirements and classify them into classes, attributes or methods. The draw is a method, so, it is identified as a method (M). Drawing

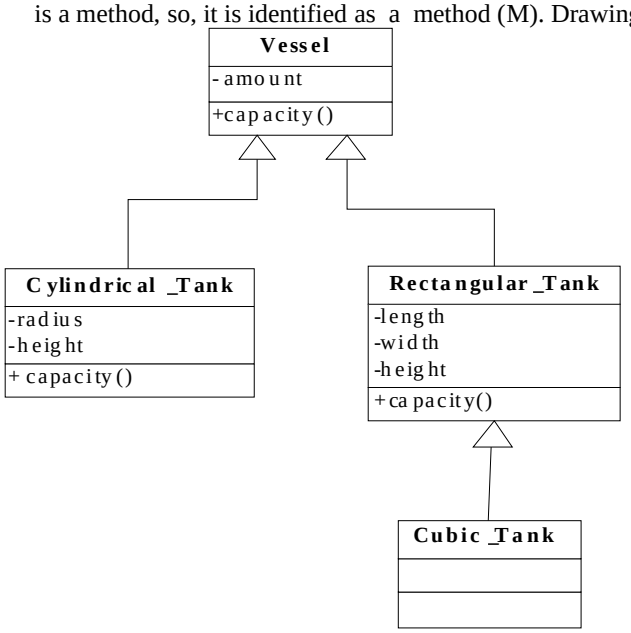

Figure 3. Static class diagram for case study I

Sheet is a class since it has "H" cells. Component is a class since it has "H" cells as well as a "P" cell.

- All entities in the table are sorted and the sorting should be applied to the rows and columns.
- Find the identical attributes and methods. Most of the properties are identical except the draw method. In the draw method the process of drawing a resister is different from the process of drawing a capacitor. And the same applies for the other entities.
- Rearrange the columns and rows in order to keep the diagonal with "I" cells in the table. This rearrangement is performed in such a way to keep the upper half above the diagonal with "H" cells and the lower half below the diagonal with the "P" cells.
- Form L-Shape from the "I" cells. There are a number of L-shapes in the table. Those L-shapes represent the inheritance. Figure 5 shows that resistor, capacitor, inductor and voltage inherit from component class.
- Group the "P" cells into groups of one, two, three and so on. The grouping process should match the lower part of the L-shape or should match the union of the lower part of some of the L-shapes. Figure 5 shows that location and node have 5 "P" cells that can be grouped together and

	electrical software	component	electrical Network	resistor	capacitor	inductor	voltage source	draw	location	drawing sheet	width	height	caption	value	move	node	x-coordinate	y-coordinate	Orientation
electrical software	I																		
component		I						H	H		H	H	H	H	H	H			
electrical Network			I																
resistor		I		I				H	H		H	H	H	H	H	H			H
capacitor		I			I			H	H		H	H	H	H	H	H			H
inductor		I				I		H	H		H	H	H	H	H	H			H
voltage source		I					I	H	H		H	H	H	H	H	H			H
draw		P		P	P	P	P	I								P			
location		P		P	P	P	P		I								H	H	
drawing sheet										I	H	H							
width		P		P	P	P	P			P	I		P	P					
height		P		P	P	P	P			P		I	P	P					
caption		P		P	P	P	P				H	H	I			P			H
value		P		P	P	P	P				H	H		I		P			H
move		P		P	P	P	P								I				
node		P		P	P	P	P						H	H		I			
x-coordinate									P								I		
y-coordinate									P									I	
Orientation				P	P	P	P						P	P					I

Figure 4. Building object-oriented information for case study II

	Electrical Software	electrical Network	drawing sheet	component	resistor	capacitor	inductor	voltage source	location	node	caption	value	move	draw	width	height	x-coordinate	Y-coordinate	Orientation	
Electrical Software	I																			
electrical Network		I		H															H	C
drawing sheet			I												H	H				C
component		P		I					H	H	H	H	H	H	H	H				C
resistor				I	I				H	H	H	H	H	H	H	H			H	C
capacitor				I		I			H	H	H	H	H	H	H	H			H	C
inductor				I			I		H	H	H	H	H	H	H	H			H	C
voltage source				I				I	H	H	H	H	H	H	H	H			H	C
location				P	P	P	P	P	I								H	H		C
node				P	P	P	P	P		I	H	H								C
caption				P	P	P	P	P		P	I				H	H			H	C
value				P	P	P	P	P		P		I			H	H			H	C
move				P	P	P	P	P					I							M
draw				P	P'	P'	P''	P'''		P				I						M
width			P	P	P	P	P	P			P	P			I					A
height			P	P	P	P	P	P			P	P				I				A
x-coordinate									P								I			A
Y-coordinate									P									I		A
Orientation				P	P	P	P	P			P	P							I	A

Figure 5. Class relation for case study II

match a lower L-shape of a group of L-shapes. Even they are classes, the relation between them is a "has-a" relation. This means that the Component class is the class that makes a relation with the location and node. Caption and Value are classes and they have 6 "P" cells, which can be grouped together and they can match the lower part of a group of L-shape. This means that this class must have a "has-a" relation with the component class. The other "P" can't be grouped with any "P" cell, which means the class has a "has-a" relation with Caption and value classes.

- Draw the static class diagram using UML notation. In this case study, there are eleven classes named according to the names in the table in Figure 5.
- Draw the inheritance relation among the classes. In this case study, resistor, capacitor, inductor and voltage source are subclasses from the component class.
- The attributes and methods are identified in previous steps. Their location is determined using the grouping of P-Cells. In this case study, the orientation attribute exists in three classes: component, caption and value classes.
- The electrical software class is a standalone class and it doesn't contain any attribute. Therefore, it can be deleted from the class diagram. Figure 6 shows the static class diagram of the software requirements in this case study.

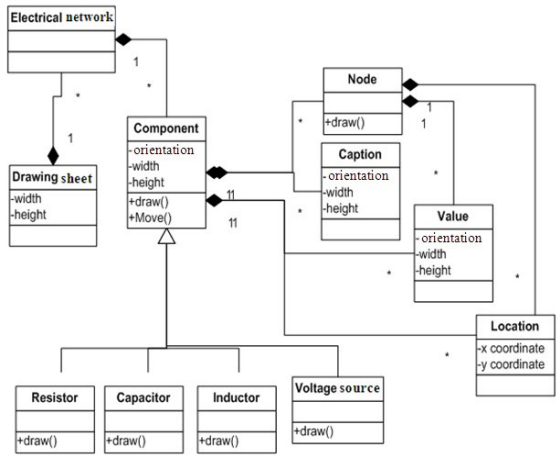

Figure 6. Static class diagram for case study II

IV. RESULTS AND CONCLUSIONS

The proposed approach of developing static class diagrams in a systematic way is very essential in the practice of object-oriented software engineering. This approach can be implemented and incorporated in any Integrated CASE (Computer Aided Software Engineering) Tool to aid in the process of obtaining the class diagrams from the software requirements. The approach has the main advantage of being systematic, which is very helpful in software systems development. The development of static class diagrams is very essential in the practice of object-oriented software engineering. For a long time, software engineers have suffered from the lack of a systematic approach for constructing these class diagrams. This paper presents such an approach which will enable the software engineers to develop these static class diagrams without going through the software requirement many times to construct such class diagrams. Our approach might seem a lengthy process. However, the approach can be implemented as a CASE Tool, which is the next step in this research.

REFERENCES

[1] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy and W. Lorensen, 1991. Object-Oriented Modeling and Design. 3rd Edn., Prentice-Hall, Egnlewood Cliffs, New Jersey, ISBN: 0136298419, pp: 500.

[2] Jacobson, I., M. Christerson, P. Jonsson and G. Overgaard, 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley ACM., New York, ISBN: 0201544350, pp: 524.

[3] Henderson-Sellers, B. and A. Simons, 2000. The Open software engineering process architecture: From activities to techniques. J. Res. Pract. Inform. Technol., 32: 47-68. http://www.acs.org.au/jrpit/abstracts/32_47.pdf

[4] Quatrani, T., 1998. Visual Modeling with Rational Rose and UML. Addison-Wesley, Reading, MA., ISBN: 0201310163, pp: 222.

[5] Jacobson, I., G. Booch and J. Rambaugh, 1999. The Unified Software Development Process. Addison-Wesley Longman, Reading, MA., ISBN: 0201571692, pp: 463.

[6] Wynekoop, J. and N. Russo, 1997. Studying system development methodologies: An examination of research methods. Inform. Syst. J., 7: 47-65. DOI: 10.1046/j.1365-2575.1997.00004.x

[7] Dawson, L. and P. Darke, 2002. The adoption and adaptation of object-oriented methodologies in requirements engineering practice. Proceeding of the ECIS 2002, June 6-8, Gdansk, Poland, pp: 406-414. http://is2.lse.ac.uk/asp/aspecis/20020083.pdf

[8] Kasser, J., 2003. Object-oriented requirements engineering and management. Proceeding of the Systems Engineering Test and Evaluation (SETE) Conference, Oct. 2003, Canberra, pp: 1-15. http://www.seecforum.unisa.edu.au/PETS/Object%20oriented%20requirements%20engineering%204.pdf

[9] Knott, R., V. Merunka and J. Polak, 2003. The role of object-oriented process modeling in requirements engineering phase of information systems development. Proceeding of the EFITA 2003 Conference, July 5-9, Debrecen, Hungary, pp: 300-306.

[10] Pressman, R.S., 2000. Software Engineering: A Practitioner's Approach. 5th Edn., Mcgraw Hill, ISBN: 0073655783, pp: 860.

[11] Pfleeger, S. and J. Atlee, 2006. Software Engineering: Theory and Practice. 3rd Edn., Pearson Prentice Hill, ISBN: 0131469134, pp: 716.

[12] Bennett, S., S. McRobb and R. Framer, 2006. Object-Oriented Systems Analysis and Design Using UML. 3rd Edn., McGraw Hill, ISBN: 0077110005, pp: 698.

[13] Rosenberg, D. and M. Stephens, 2007. Use Case Driven in Object Modeling with UML: Theory and Practice. Apress, ISBN: 1590597745, pp: 438.

[14] Arman, N. and Daghameen, K., 2007. "A Systematic Approach for Constructing Static Class Diagrams from Software Requirements," Proceedings of the 8th International Arab Conference on Information Technology (ACIT'2007), November 26-28, Academy for Science & Technology and Maritime Transports, Syria.

AUTHORS PROFILE

Nabil Arman received his BS in Computer Science with high honors from Yarmouk University, Jordan in 1990, an MS in Computer Science from The American University of Washington, DC USA in 1997, and a Ph.D. from the School of Information Technology and Engineering, George Mason

University, Virginia, USA in 2000. He is an Associate Professor of Computer Science at Palestine Polytechnic University, Hebron, Palestine. Dr. Arman is interested in Database and Knowledge-Base Systems, and Algorithms.

Khalid Daghameen received his BS in Computer Systems Engineering in 1996 from Palestine Polytechnic University/Palestine. He worked for two years as a teaching assistant at Palestine Polytechnic University. In 2001, he completed his master's degree in Computer Engineering from University of Detroit Mercy/USA. Then he worked for two years in Caterpillar Inc as a computer engineer. Currently, he is working at Palestine Polytechnic University/College of Engineering and Technology as an instructor. His main interests are algorithms, image processing and software Engineering.