Journal of Theoretical & Applied Information Technology, Islamabad PAKISTAN

31st May 2010. Vol.15. No 2.

www.jatit.org

REQUIREMENTS BASED STATIC CLASS DIAGRAM CONSTRUCTOR (SCDC) CASE TOOL

[1]KHALID DAGHAMEEN, [2]NABIL ARMAN

[1]Instructor, Department of Electrical and Computer Engineering, Palestine Polytechnic University, Palestine

[2]Associate Professor, Department of Mathematics and Computer Science, Palestine Polytechnic University, Palestine

E-mail: dkhalid@ppu.edu , narman@ppu.edu

ABSTRACT

Object-Oriented development methodology is currently the main trend in software industry. Many tools were introduced to aid in the analysis and design phases of the object-oriented development methodology. However, tools that can aid in the requirements phase of the object-oriented development methodology and that can generate class diagrams as a major step in the design phase have attracted little attention and efforts. This paper introduces a tool to automate the process of constructing a static class diagram from software requirements. The CASE tool takes the static class diagram a step further and generates a skeleton code for popular object-oriented languages.

Keywords: *Software requirements, Object-oriented development, CASE tools.*

1. INTRODUCTION

Object-oriented software development has been a major trend in software industry. To reduce the high costs of developing software companies have been trying to automate as many phases as they can in the object-oriented development methodology. Obtaining static class diagrams from software requirements is a major activity in object-oriented development methodology. A systematic approach for constructing static class diagrams from software requirements is presented in [1]. This paper presents a CASE tool that is based on that approach of constructing class diagrams. Many tools were introduced to aid in the analysis and design phases. It is well-known that there are three generic phases for software development regardless of the methodology being used. These phases are definition, development and deployment [2]. For these different stages, many CASE tools were introduced that can aid in the automation of some phases of the software development of the software. These tools, which automate part of the development process, include UML Skeleton tool [2,3]. The tool converts the design of the class diagram into a skeleton of code based on the chosen language. The programmer must do the rest of the work. Other tools were introduced to help in cost estimation and software tracking process. These are managerial processes rather than software development. An example of those tools is COCOMO model [2,4]. One of the main tools in Object-Oriented development is the Rational Rose. It uses the UML for the design of the software development. This tool can even have a skeleton code for different languages after obtaining the static class diagram, which is the role of the developer. One main thing about Rational Rose is that it works perfectly after design [5].

The role of use cases in the construction of class diagrams was presented in [6]. However, the focus was on using the use cases rather than the software requirements to construct class diagrams as presented in our approach. Therefore, one should consider developing a tool to automate the process of generating use cases from software requirements represented by narrative description of a problem statement for that tool to perform what our tool is capable of.

A tool to implement a system that automates the building of class diagrams from free-text

ISSN: 1817-3195 / **E-ISSN**: 1992-8615

31st May 2010. Vol.15. No 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

Table 1. Building Object-Oriented information

requirement documents was presented in [7]. This approach first applies natural language processing techniques to understanding of the written requirements, and then uses domain knowledge represented by domain ontology to improve the performance of class identification. However, this approach has the overhead of handling the domain ontology to identify classes of the class diagram. These tools perform part of the development phase. There is no tool that performs the automation of the entire process. Another important issue is that there is no tool that automates the process between requirement analysis and design phases. The CASE tool in this paper introduces a semi-automated process to perform that. Human intervention is required at some point to answer few questions or confirm certain assumptions.

2. SCDC CASE TOOL DESCRIPTION

The CASE tool presented in this paper uses the systematic approach/algorithm presented in [1]. The algorithm consists of a number of iterative steps to accomplish the construction of the static class diagram form the software requirements of a given narrative description of a problem statement. The algorithm needs to build a matrix to represent the relationships between candidate classes and the candidate's members of these classes. The matrix must be filled with letters such as I, P and H. These letters represent inheritance, part of relationship and "has-a" relationship. The matrix is then used to extract the static class diagram from the matrix. A sample matrix of certain requirements is given in Table 1.

ISSN: 1817-3195 / E-ISSN: 1992-8615

31st May 2010. Vol.15. No 2.

The CASE tool works to develop the static class diagram after analyzing the requirements input to the tool as a paragraph. Another tool for natural language processing is used to extract all the nouns, noun-phrases and verbs from the problem statement narrative description. This tool used the English language. The lack of a natural processing tool for Arabic language represents an obstacle of allowing problem statement description in Arabic to be used in SCDC CASE tool. The approach that is presented in [1] has a number of steps to automate the process of obtaining the static class diagram from the matrix filled with the letters I, H, and P using the algorithm:

```
for each column of the matrix
 for each row of the matrix
    Begin
        Matrix[row, column]= “”
        Construct an “IS_A” questions for
        that column, row
        if the Answer of the question is Yes
          Matrix [row, column]=”I”
     else
        Construct a “HAS_A” question for
        that column, row
      if the answer of the question is Yes
        Matrix [row, column]=”H”
      else
         Construct a “part-of” question for
         that column , row
       if the answer of the question is Yes
          Matrix [row, column]=”P”
   End
```

Determining the inheritance in the matrix is performed by finding a cell that contains an “I” between a row and a column that are not equal and that the column must be less than the row is given by the algorithm:

```
for each row in the matrix

 for column from 0 to row
   if Type of the that row is class
     if Matrix [row, column] =”I”
          Relation between classes [row]
           and classes[column]  is inheritance.
```

The tool output is an XML file that describes the candidate classes, classes contents, structure and relationships between classes. In addition, the tool provides a skeleton code for two languages: VB.NET as well as C# language. SCDC CASE tool can be extended easily to support other object-oriented programming languages like Java and C++.

3. SCDC CASE TOOL DEMONSTRATION

SCDC is a CASE Tool that has a GUI interface, that is very simple and easy to use. After launching the application, the program shows up like Figure 1. The user can enter the problem statement by either specifying the text file of the problem statement or by copying it from any other text program and paste it into the specified box. In order to get all the nouns and noun phrases as well as verbs, the user must click on the button start analyzing. The CASE tool fills up a list of nouns and noun phrases as a well as another list for the verbs and verb phrases.

The SCDC gives the opportunity to the developer to change, delete or even add new nouns or verbs. If the user feels there are more nouns or verbs that must be there. he/she can add them to the lists using the GUI components as shown in Figure 2. The second way has a default answer and it is faster than the message box. Figure 2 shows the problem statement of the example as well as the noun and noun phrases that were extracted from the problem statement. The next step is to perform a series of yes/no questions. This step is too lengthy, but the tool works to decrease the number of questions. The questions can be performed in two ways:

(1) The first one uses the message boxes in which each message box would fill one cell in the matrix. Figure 3 shows a sample of the message box questions.

(2) The second one is to use a grid of questions; the tool shows all the questions in one form, so the developer could pick the answer of any question. Figure 4 shows a sample of the grid that has the questions. The second way has a default answer and it is faster than the message box. Figures 3 and 4 shows part of the questions that the CASE tool construct and that can generally be answered by the software engineer. The questions refer to the same previous problem.

The developer does not need to worry about losing the data. In fact he/she can save all what he/she does until the point then continue at some other time. The tool saves all the information in a

ISSN: 1817-3195 / E-ISSN: 1992-8615

31st May 2010. Vol.15. No 2.

project file in an XML format. After completing the questions, the tool would generate the static class structure, the output would be an XML format. Figure 5 shows a sample of the output file of the previous example.

As mentioned before, at some stages human intervention is required to refine those nouns and noun phrases as well as verbs and verb phrases.

The tool creates a matrix according to the number of nouns and verbs. The matrix initially would be empty. Note that the nouns and verbs are sorted and the nouns are kept at the beginning of the list. It is clear that filling the matrix with appropriate letters require extensive human intervention, although the tool has some optimization to ease this step and make the human intervention less, but still there is a work that must be done by the software engineer. The tool performs this by a series of yes/no questions to fill up the cells of the matrix, or a chart that contains radio buttons to select the correct answer.

The tool searches each row to identify its type. For example any row that has more than one cell that contains an "I" means that this element is a class. The same thing applies if one cell in a row that has one cell that contains an "H". The verbs which are at the rear of the list are always methods.

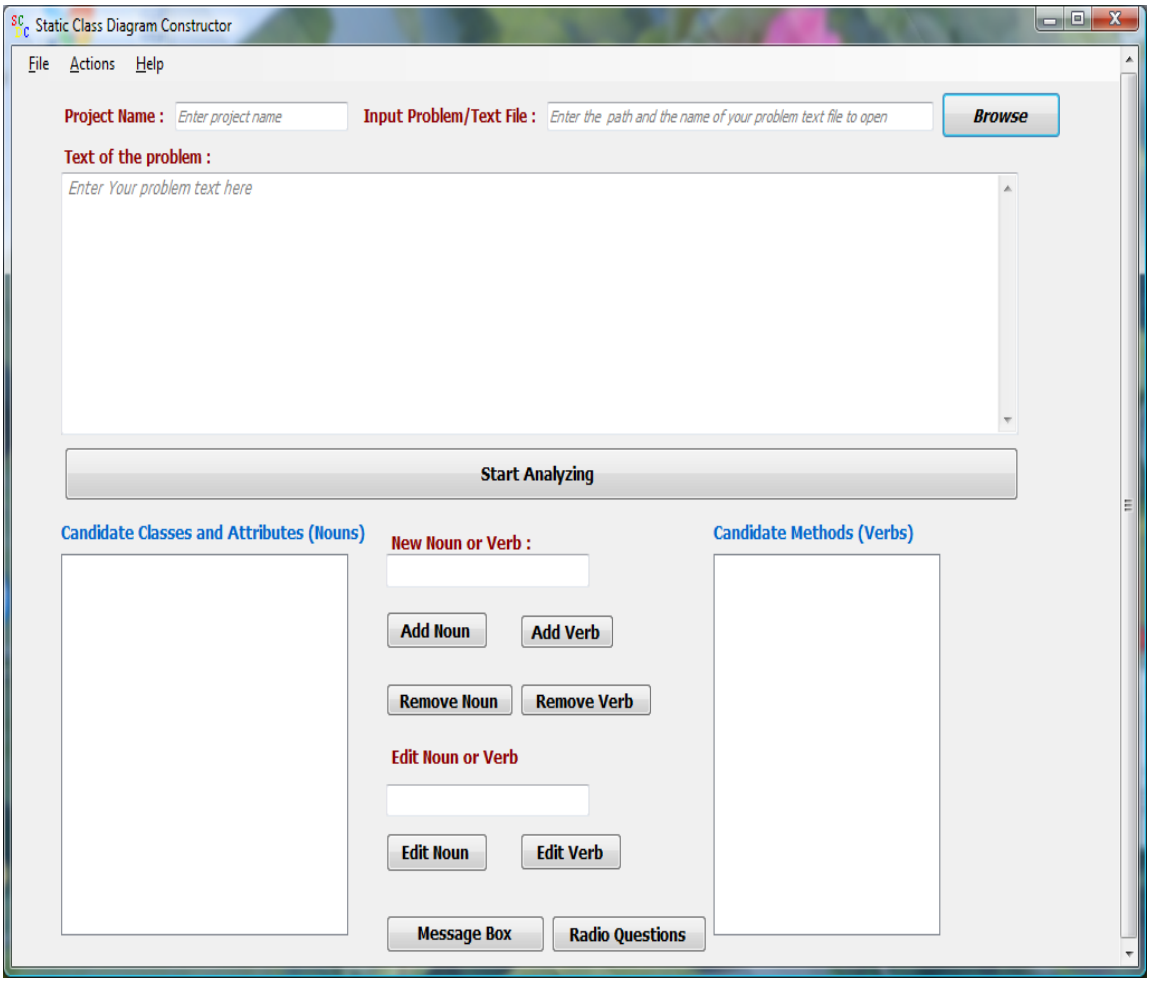

Figure 1. SCDC Main Window

ISSN: 1817-3195 / E-ISSN: 1992-8615

31st May 2010. Vol.15. No 2.

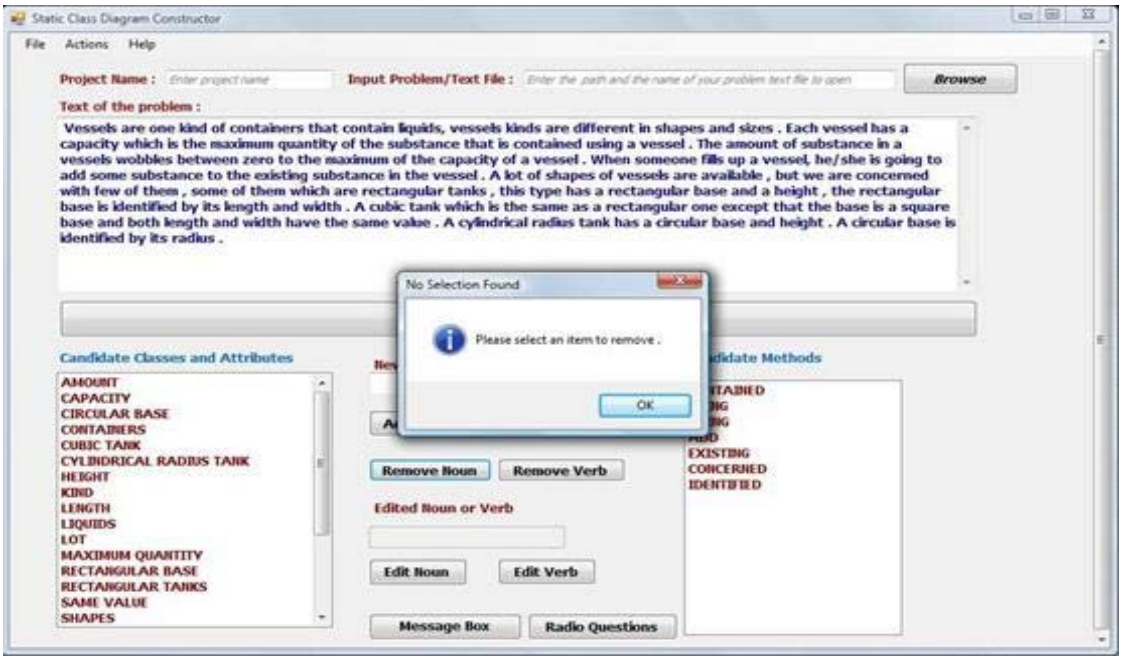

Figure 2. SCDC after analyzing the problem statement

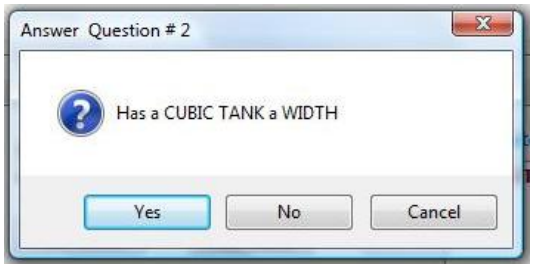

Figure 3. Sample of the Message box questions

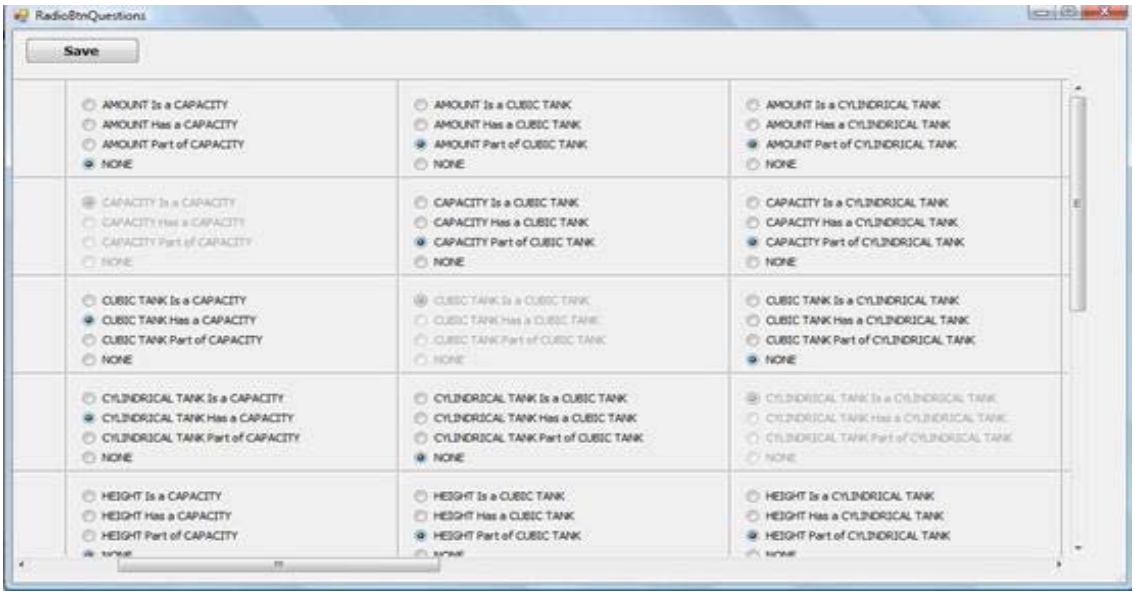

Figure 4. Grid view of the Questions

ISSN: 1817-3195 / E-ISSN: 1992-8615

31st May 2010. Vol.15. No 2.

www.jatit.org

```
<?xml version="1.0" ?>
- <Package>
  <ProjectName>VESSELS_SHAPES</ProjectName>
- <Class>
  <ClaasName>VESSELS</ClaasName>
- <Attributes>
  <Item>AMOUNT</Item>
  </Attributes>
- <Operations>
  <Item>CAPACITY</Item>
  </Operations>
  <Super>Object</Super>
- <Sub>
  <Item>CYLINDRICAL_TANK</Item>
  <Item>RECTANGULAR_TANKS</Item>
  </Sub>
  </Class>
- <Class>
  <ClaasName>CYLINDRICAL_TANK</ClaasName>
- <Attributes>
  <Item>HEIGHT</Item>
  <Item>RADIUS</Item>
  </Attributes>
  <Operations />
  <Super>VESSELS</Super>
  <Sub />
  </Class>
- <Class>
  <ClaasName>RECTANGULAR_TANKS</ClaasName>
- <Attributes>
  <Item>HEIGHT</Item>
  <Item>LENGTH</Item>
  <Item>WIDTH</Item>
  </Attributes>
  <Operations />
  <Super>VESSELS</Super>
- <Sub>
  <Item>CUBIC_TANK</Item>
  </Sub>
  </Class>
- <Class>
  <ClaasName>CUBIC_TANK</ClaasName>
  <Attributes />
  <Operations />
  <Super>RECTANGULAR_TANKS</Super>
  <Sub />
  </Class>
  </Package>
```

Figure 5. Sample of the output file

Generating the class hierarchy is also performed by the CASE tool. It looks for any row that has more than one cell that contains an "I" and finds that the corresponding column , which means the relation between the left-cell and the top-cell is an inheritance. If the row has more than twice of the cells that contain an "I". This means multiple inheritance between the left-cell and the top-cells.

4. CONCLUSIONS

In this paper, the systematic approach for constructing static class diagrams from software requirements is used to develop SCDC CASE tool. SCDC CASE tool can aid in the automation of the construction of static class diagrams from software requirements expressed as a problem statement narrative description with little human intervention to confirm certain assumptions about the requirements. SCDC CASE tool uses a simple and user friendly GUI interface. The tool is very useful for software engineers who are responsible for generating the static class diagrams from software requirements and who are interested in generating the skeleton code for popular object-oriented programming languages. In addition, SCDC CASE tool doesn't involve a great deal of preprocessing steps like other tools that involve much overhead in determining the appropriate classes from the software requirements and in constructing the class

ISSN: 1817-3195 / E-ISSN: 1992-8615

31st May 2010. Vol.15. No 2.

diagrams. Finally, these tools don't generate skeleton code for popular object-oriented programming languages to facilitate the tasks of software engineers.

ACKNOWLEDGEMENTS

The authors would like to give deep appreciation to the students Ibrahim Saraheen, Sari Jabareen and Sari Adam for implementing the tool.

REFRENCES:

[1] Arman, N. and Daghameen, K., "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, 2007, Academy for Science & Technology and Maritime Transports, Syria.

[2] Pressman, R., Software Engineering, McGraw-hill, 6th edition, 2005. ISBN :007-123840-9.

[3] Rosenberg, D. and Stephens, M. , "Use Case Driven Object Modeling with UML", Apress, 2007, ISBN-10 (pbk): 1-59059-774-5.

[4] Galorath, D. and Evans, M., "Software Sizing, Estimation, and Risk management", Auroback publication, 2006, ISBN-10(pbk): 10: 0-8493-3593-0.

[5] Boggs, M, and Boggs, W. Mastering UML with Rational Rose, 2002, sybex, ISBN-10: 0782140173.

[6] Anda, B. and Sjoberg, D., "Investigating the Role of Use Cases in the Construction of Class Diagrams," ESE, Vol. 10, No. 3, 2005.

[7] Zhou, X., Zhou, N., "Auto-generation of Class Diagram from Free-text Functional Specifications and Domain Ontology," 2008, downloaded 1/2/2010 from http://www.daviszhou.net/Research/INFO626Prj.pdf.

AUTHOR PROFILES:

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.

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, Algorithms and Software Engineering.

ISSN: 1817-3195 / **E-ISSN**: 1992-8615