

Using the object mapping approach from analysis to implementation for developing student registration system

Dhafer Abdulameer Abdulmonim¹, Zainab Hassan Muhamad², Bashar Alathari³

^{1,2}Department of Computer Science, Directorate of Education, Iraqi Ministry of Education, Iraq

³ITRDC, University Of Kufa, Iraq

Article Info

Article history:

Received Oct 15, 2018

Revised Nov 18, 2018

Accepted Jan 13, 2019

Keywords:

Object-oriented approach
Unified modeling language
Use-case diagram
Class diagram
Sequence diagram

ABSTRACT

The Unified Modeling Language (UML) is the effective standard language of an object-oriented based system analysis and design. The using of object-oriented approach based on UML specification can offer an understandable model, which can reduce the system's complexity. The use-case diagram, class diagram and sequence diagram are important models of the system development during early stages. The object mapping approach between UML class diagram in the analysis phase and source code in the implementation phase is a significant process in the software development life cycle. In order to ensure the consistency of system requirements from analysis to implementation. This paper presents the student registration system by using object mapping based on UML. The UML specification and the generation of Java source code influenced by specific features of class in object-oriented such as inheritance and the association between the classes.

Copyright © 2019 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Dhafer Abdulameer Abdulmonim,
Department of Computer Science, Directorate of Education,
Iraqi Ministry of Education, Najaf, Iraq.
Email: dhafer985@gmail.com

1. INTRODUCTION

In general, the software engineering has many of the software development methodologies that have been introduced [1]. The object-oriented approach is a significant approach, and it is one of the most popular approaches for systems development [2], [3]. In addition, it aims to specify the requirements of the organization and then uses diagrams for modelling and analysing the system requirements [4]. Unified Modeling Language (UML) is the modeling tool that used for describing the object-oriented based systems. UML is a visual language for constructing and modeling the structure and behavior of a system based on diagrammatic notation, which supported by syntax, and expressive semantics that are easy to understand [5], [6]. Moreover, the system requirements transformed to UML specification using UML diagrams during the system analysis process. Which are use-case diagram, class diagram, object diagram, sequence diagram, deployment diagram, component diagram, activity diagram, state diagram, and collaboration diagram. Furthermore, there are introduced many computer languages for object-oriented programming such as C, C++, and Java [7], [8].

A critical issue with systems modelling is that it becomes difficult to understand an entire design as consistent and congruent with system requirements from analysis phase into implementation phase. Although there are many researchers, who implemented studies related that had adopted the object-oriented approach. In [1], [3] the authors developed the management systems using object-oriented approach based on UML specifications. The UML diagrams used like use-case diagram, class diagram and activity diagram of the analysis phase. As well as, they mapped all the classes that had constructed in the system modelling to a source code at the implementation phase. By using the object-oriented programming languages C++. Hao & *et al.* [5] described the analysis, design and implementation of the experiment management system. The developed system via the Java language. Moreover, the system modeling based on UML diagrams like use-case diagram,

class diagram, communication diagram, and component diagram. A facilities management and information system had offered by [7] based on object-oriented approach. The UML diagrams used like use-case diagram, activity diagram, sequence diagram, object diagram, and state machines in system requirements analysis and design. In [8] the authors designed open-end fund investment of stock management system. Using UML diagrams: use-case diagram, class diagram, and sequence diagram. The developed system adopted UML specifications in the analysis and the design phases; also, it established the UML diagrams by the Rational Rose tool. In another work, Personnel management system investigated based on UML. The system requirements analyzed and structural system constructed as well. The author indicated that the UML improved efficiency and quality of the developed system. The UML diagrams used like use-case diagram, class diagram, and sequence diagram in [9]. In this respect, the studies [1], [3] are similar issues with our work as system analysis, UML specifications, class diagrams relationships analysis, and object mapping. Nevertheless, unlike with our work, including the sequence diagrams in UML specifications and the mapping of attributes and associations into Java programming language. In contrast, the essential difference to our work is that the studies [5], [7]-[9] mainly focused on the consistency between different kinds of diagrams, but not on the consistency between diagram and source code.

Therefore, our mapping method can handle system designs consistency problems, where UML diagrams corresponds to system requirements and source codes. Our goals in the generation of Java source code from design specifications is to facilitate the design and development process and ensuring consistency from design to implementation. On the other hand, the most of the colleges and universities use a manual procedure of registration management, which consumes a time and effort of the employee, students and lecturers. In addition, causes problems for accuracy and reliability of data. In order to solve the problem, development of the student registration system is in urgent need. The system will be providing accurate information, and minimize time and effort. Therefore, this paper describes the student registration system based on an object-oriented approach to define system specifications using UML diagrams. The use-case diagram, the class diagram, and the sequence diagram used in this paper. Moreover, this paper provides an analysis of the UML class diagram, and then converts the attributes, methods, and associations of the class diagram into the source code of the Java programming language.

2. RESEARCH METHOD

This section describes the adopted approach, in order to ensure the consistency of system requirements from analysis to implementation for the development of student registration system. The adopted method used an object-oriented approach to define system specifications using UML diagrams like: the use-case diagram, the class diagram, and the sequence diagram. Moreover, the UML class diagram analysed, and the attributes, methods, and associations of the class diagram converted into the Java source code. In addition, the object mapping method between UML specification and source code adopted as presented in [1] from analysis phase to implementation phase. As described in the following subsections:

2.1. System Requirement Analysis

In this paper, we present a student registration system. The system will enable the student to add or drop courses. Student may view registered courses. Professor must be able to login system to indicate which courses they will be teaching. They will also need to see which students signed up for their course offerings. This job is under the responsibility of the administrator who can create a new course. Administrator should provide an option to modify or remove courses.

2.2. UML Specification

In this paper, we will focus on three UML diagrams, which are use-case diagram, class diagram and sequence diagram for specifying the requirements of the student registration system. Rational Rose selected as UML modeling tool.

2.2.1 Use-Case Diagram

The use-case diagram has a main role in the system modelling. The use case diagrams provide interactions between actors and the system. It defines the final objectives in the system development to designers facilely. Therefore, it has helped solve the complexity of systems analysis and design [10], [11]. Figure 1 shows the use-case diagram for the student registration system. There are three actors involved in this system that are the administrator, the professor and the student. There are ten use cases/functions in the system, which consists of Login, DisplayCourses, AddCourse, DropCourse, ViewRegisteredCourses, Select Courses to Teach, CreateNewCourse, ModifyCourse and Remove Course.

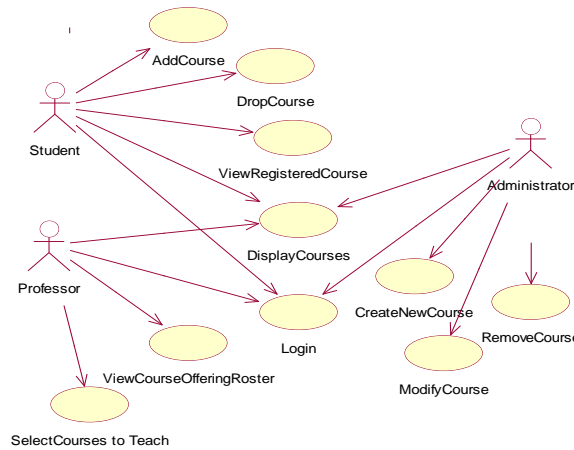


Figure 1. The use-case diagram for the student registration system

2.2.2 Class Diagram

The class diagram is widely used in object-oriented analysis and design; it describes the system architecture by classes and their relationship [12], [13]. Based on the use-case diagram in Figure 1, we constructed the class diagram for the student registration system; the class diagram divided into five classes created as shown in Figure 2. The classes are User, Professor, Student, Course and Registration. The attributes of class User are Name, Password, Email, Address, and ContactNumber. For each attribute, Set and Get methods for User class. The attributes of class Professor are StaffID, Salary and Department. For each attribute, Set and Get methods for Professor Class. The attribute of class Student is StdID and methods are SetStdID and GetStdID. The attributes of class Course are CourseCode, CourseName, FacultyCode, StartDate, End Date. For each attribute, Set and Get methods for Course class. The class User is a superclass with classes Professor and Student. The classes Professor, Student inherit attributes and methods from class User where all the methods and attributes are dependent to the class User. The class Registration uses the classes Professor, Student and Course. The methods of class Registration are Login (), DisplayCourses (), AddCourse (), DropCourse (), ViewRegisteredCourses (), ViewCourseOfferingRoster (), SelectCoursesToTeach (), Create New Course (), ModifyCourse (), and RemoveCourse (). There are no attributes in class Registration.

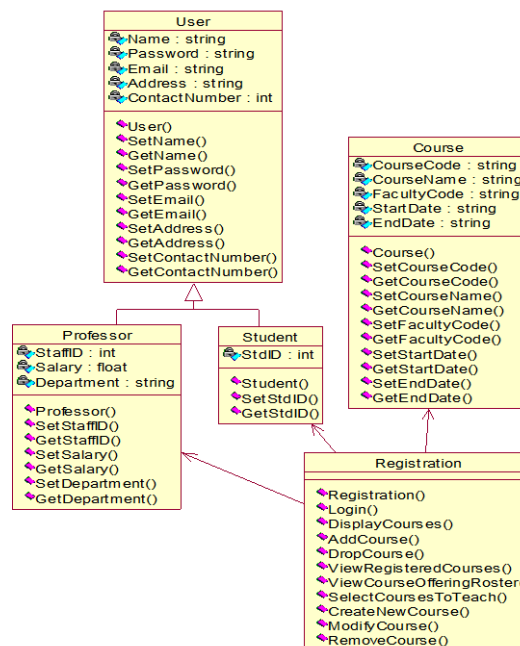


Figure 2. The class diagram for the student registration system

2.2.3 Sequence Diagram

The UML sequence diagram used, in order to understand objects interactions. The sequence diagrams are divided based on actors [14], [15]. In this paper, three of sequence diagrams created for the student registration system. The sequence diagrams express for the system administrators, the professor and the student. Figure 3 shows the sequence diagram for system administrator. Since there are many use cases regarding to the functions or activities of the system administrator, we will summarize the description of the messages depending on each use-case as follows: from one to five, the system administrator starts logging in if the password is right, the administrator can log in; otherwise, the administrator cannot enter. After the administrator entered, the admin page will be appear to interact with the system. From six to nine, the administrator can request to display a list of courses then the system return list of courses to display it. From ten to fourteen, the administrator can request to modify a course, after that the system will be update the course information in database and display update success message. From fifteen to nineteen, the administrator can request to delete a course, after that the system will be delete the course information in database and display delete success message. Figure 4 shows the sequence diagram for the Professor. The brief description of the messages depending on each use-case as follows: from one to five, the Professor starts logging in if the password is right, the Professor can log in; otherwise, the Professor cannot enter. After the Professor entered, the Professor page will be appear to interact with the system. From six to nine, the Professor can request to display a list of courses then the system return list of courses to display it. From ten to fourteen, the Professor can request to select courses to teach, after that, the system will be save the course specified in the database and display selecting success message. From fifteen to eighteen, the Professor can request to a list of course registered, after that the system will be display course offering roster.

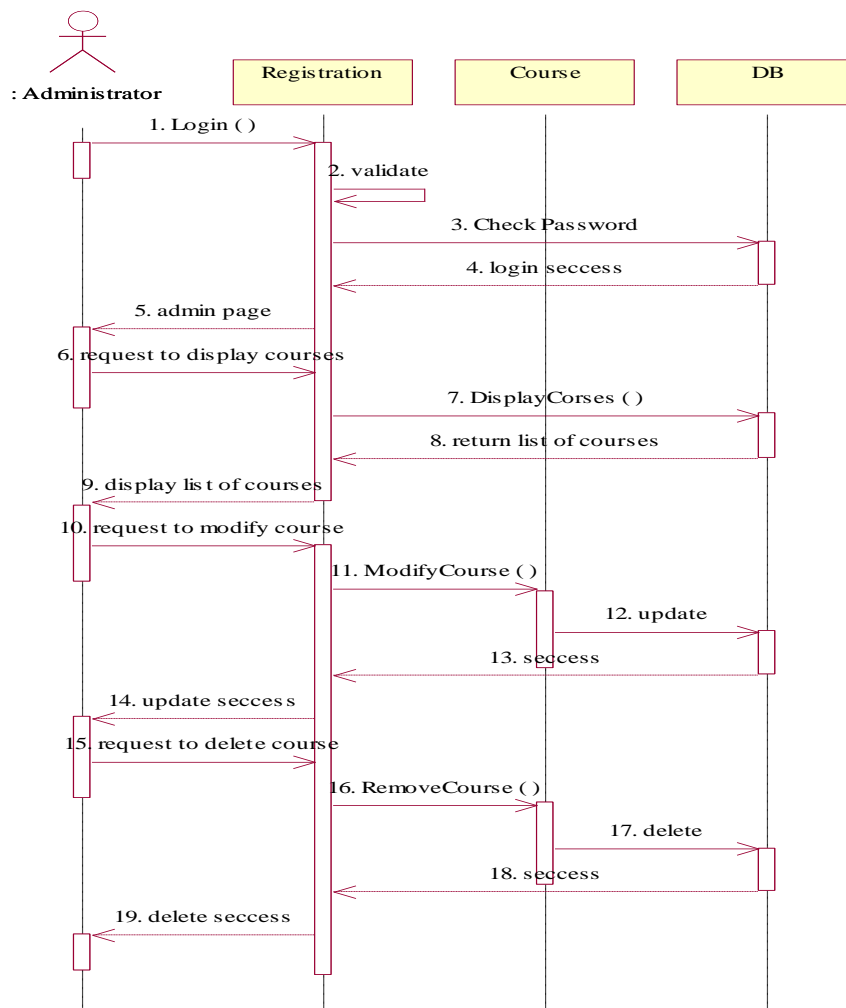


Figure 3. The sequence diagram for system administrator

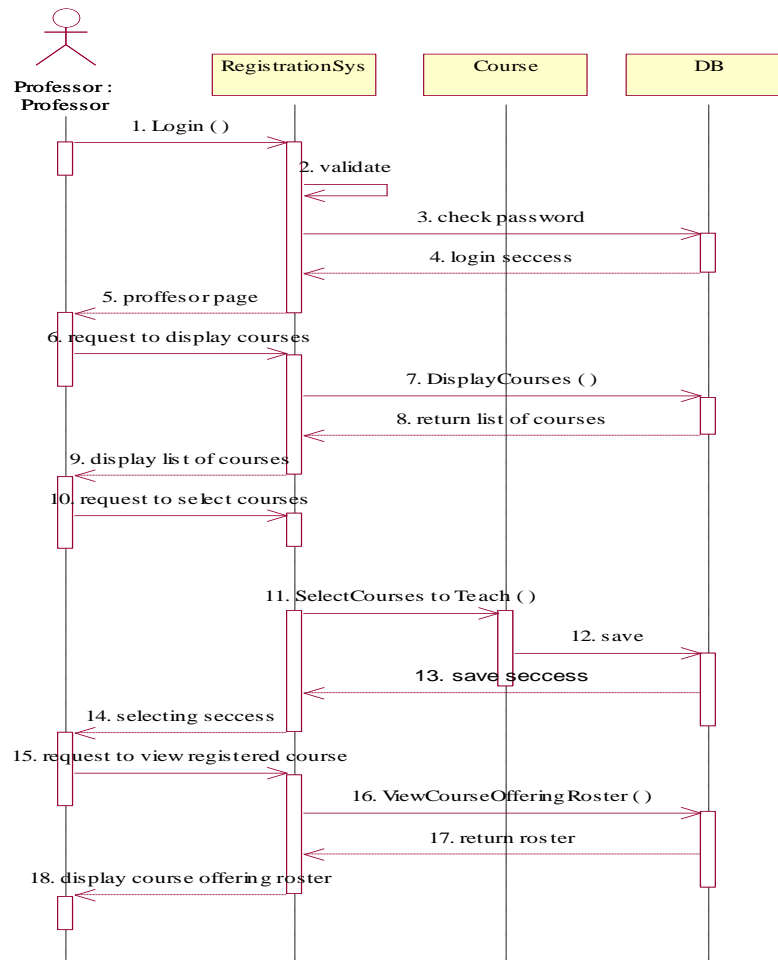


Figure 4. The sequence diagram for the Professor

Figure 5 shows the sequence diagram for the Student. The summarize description of the messages depending on each use-case as follows: from one to five, the Student starts logging in if the password is right, the Student can log in; otherwise, the Student cannot enter. After the Student entered, the Student page will be appear to interact with the system. From six to nine, the Student can request to display a list of courses then the system return list of courses to display it. From ten to fourteen, the Student can request to select courses to register, after that, the system will be save the course specified in the database and display registration success message. From fifteen to nineteen, the Student can request to drop a course; the system will be delete the course specified from the database and display drop success message. From twenty to twenty three, the Student can request to view a list of registered courses then the system return a list of registered courses to display it.

2.3. Object Mapping between Uml Specification and Source Codes

After the UML diagrams created in the design phase, the Java source code generated. In this paper, all the class diagrams and source code mapped prior the implementation phase. The mapping between UML specification and implementation involves translating UML class diagrams into Java source codes. Taking into consideration, the strictly commitment for the symmetry between class diagram and source code only the header files. The generated Java source code based on UML class diagram for the student registration system shows in the next section.

3. RESULTS and DISCUSSION

The results represent of objects mapping between the UML class diagrams and Java source codes for the student registration system as indicated from Table 1 to Table 4. Table 1 shows the mapping of class User with Java source code. The attributes and methods translated into Java source code using object-oriented

approach, all variables declared private and all methods declared public. Table 2 shows the mapping of class Professor and class Student into the Java source codes. Whereas, class Professor and class Student inherit from class User. In object mapping approach, the classes' inheritance concept translated into the Java source codes.

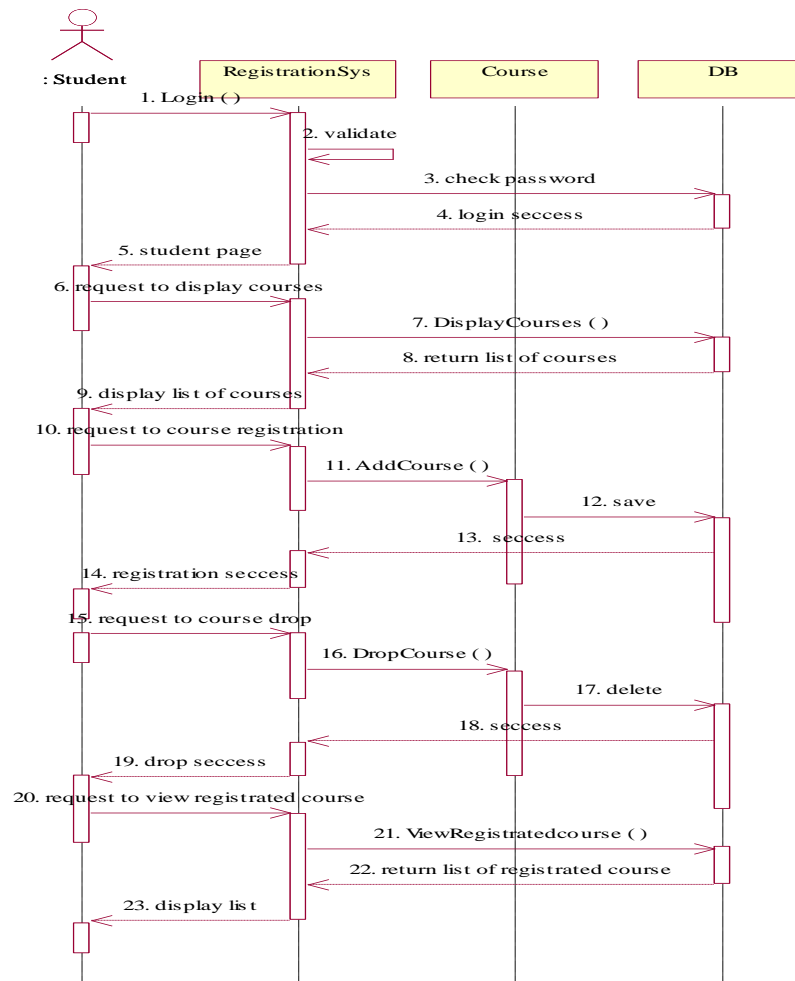


Figure 5. The sequence diagram for the Student

Table 1. Mapping of Class User into Implementation

UML class diagram	Java source codes
User	public class User {
• Name	//Instance variables (attributes)
• Password	private string Name;
• Email	private string Password;
• Address	private string Email;
• ContactNumber	private string Address;
	private int ContactNumber;
• User ()	// Method :
• SetName ()	public User () { } // Constructor – Initializes all instance variables
• GetName ()	public void SetName (string Name) { } // Set and get methods for all attributes
• SetPassword ()	public string GetName () { }
• GetPassword ()	public void SetPassword (string passsword) { }
• SetEmail ()	public string GetPassword () { }
• GetEmail ()	public void SetEmail (sting email) { }
• SetAddress ()	public string GetEmail () { }
• GetAddress ()	public void SetAddress (string address) { }
• SetContactNumber ()	public string GetAddress() { }
• GetContactNumber ()	public void SetContactNumber(int contactno) { }
	public int GetContactNumber() { } } // class User

Table 2. Mapping of Class Professor and Class Student using Inheritance into Implementation

UML class diagram	Java source codes
Professor	<pre>private class Professor extends User { // inheritance</pre>
<ul style="list-style-type: none"> • StaffID • Salary • Department 	<pre>//Instance variables (attributes) private int StaffID; private float Salary; private string Department;</pre>
<ul style="list-style-type: none"> • Professor() • SetStaffID () • GetStaffID () • SetSalary () • GetSalary () • SetDepartment () • GetDepartment() 	<pre>// Method : public Professor() { } // Constructor – Initializes all instance variables public void SetStaffID (int staffid) { } // Set and get methods for all attributes public int GetStaffID () { } public void SetSalary (int float salary) { } public float GetSalary () { } public void SetDepartment (int sting department) public sting GetDepartment() { } } // class Professor</pre>
Student	<pre>private class Student extends User { // inheritance</pre>
<ul style="list-style-type: none"> • StdID • SetStdID () • GetStdID () 	<pre>// Instance variables (attributes) private int StdID; // Method : public Student() { } // Constructor – Initializes all instance variables // Set and get methods for all attributes public void SetStdID (int stdid) { } public int GetStdID() { } } // class Student</pre>

Table 3. Mapping of Class Course into Implementation

UML class diagram	Java source codes
Course	<pre>private class Course {</pre>
<ul style="list-style-type: none"> • CourseCode • CourseName • FacultyCode • StartDate • EndDate 	<pre>// Instance variables (attributes) private string CourseCode; private string CourseName; private string FacultyCode; private string StartDate; private string EndDate;</pre>
<ul style="list-style-type: none"> • Course() • SetCourseCode () • GetCourseCode() • SetCourseName () • GetCourseName() • SetFacultyCode() • GetFacultyCode() • SetStartDate() • GetStartDate() • SetEndDate() • GetEndDate() 	<pre>// Method : public Course() { } // Constructor – Initializes all instance variables // Set and get methods for all attributes public void SetCourseCode (string coursecode) { } public string GetCourseCode() { } public void SetCourseName(string coursenam) { } public string GetCourseName() { } public void SetFacultyCode(string facultycode) { } public string GetFacultyCode() { } public void SetStartDate(string startdate) { } public string GetStartDate() { } public void SetEndDate(string enddate) { } public string GetEndDate() { } } // class Course</pre>

Table 4. Mapping of Class Registration Using Association into Implementation

UML class diagram	Java source codes
Registration	<pre>private class Registration {</pre>
<ul style="list-style-type: none"> • Registration() • Login() • DisplayCourses() • AddCourse() • DropCourse() • ViewRegisteredCourses() • ViewCourseOfferingRoster() • SelectCoursesToTeach() • CreateNewCourse() • ModifyCourse() • RemoveCourse() 	<pre>// Association with class "student" and class "course" and class "professor" private Professor theProfessor; private Student theStudent; private Course theCourse; public Registration() { } / Constructor – Initializes all instance variables // main methods public void Login() { } public void DisplayCourses() { } public void AddCourse() { } public void DropCourse() { } public void ViewRegisteredCourses() { } public void ViewCourseOfferingRoster() { } public void SelectCoursesToTeach() { } public void CreateNewCourse() { } public void ModifyCourse() { } public void RemoveCourse() { } } // class Registration</pre>

The class Course and class Registration mapped into Java source codes as shown in Table 3 and Table 4. Table 3 shows the mapping of class Course with Java source codes, and Table 4 shows the mapping of class Registration with Java source codes. In addition, Table 4 illustrated the classes association concept in object-oriented for class Registration with class Student, class Course, and class Professor. The classes association establishes through their objects to use functionality and services provided by that object.

The Object mapping method designed to support the software development process. It an effective method for generating Java source code depends on UML specifications. Depending on the results through development of a student registration system, the characteristics of the method are it allows generating Java source code that adapted to UML specification constructs such as inheritance and the association between the classes. In addition, it mapped the generated Java source code into UML class diagram specifications. That shield developers from the issues of the lack of consistency and congruence between the UML specification of system modelling and the generated Java source code implementation. Moreover, it supports the designed high-level model that is faithful to the system requirements from analysis phase into implementation phase. Thus, it refutes concerns by supporting the generated Java source code to related designs. Although, the generated Java source code from UML class diagram influenced by certain features of the Java language that is lack of multiple inheritance of implementations of classes.

4. CONCLUSION

The paper describes the analysis phase and the implementation phase of the student registration system by using a UML-based object-oriented approach. In addition, the object mapping from analysis into implementation. In accordance with the analysis the UML class diagrams and translate attributes and methods to the Java source code, we can consider the mapping between UML class diagram and Java source code have a significant impact on the software development process. In order to achieve consistency between requirements analysis phase and implementation phase.

ACKNOWLEDGEMENTS




The authors would like to thank the University of Kufa, Ministry of Higher Education and Scientific Research (MoHESR), Directorate of Education in Najaf and the Iraqi Ministry of Education for their support with collaboration and assistance in this research.

REFERENCES

- [1] R. Ibrahim "From UML Specification into Implementation using Object Mapping", *Canadian Center of Science and Education on Computer and Information Science*, Vol 3, No 3, pp.149-157, August 2010.
- [2] D. Kung and J. Lei, "An Object-Oriented Analysis and Design Environment", In *Applied 29th International Conference on Software Engineering Education and Training (CSEET)*, 2016, IEEE, pp. 91-100.
- [3] R. Ibrahim, *et al*, "Development of staff management system using UML-based object-oriented approach", In *Applied of the International Conference on Computing Technology and Information Management*, 2014, pp. 119-124.
- [4] F. Alhumaida and N. A. Zafar, "Possible improvements in UML behavior diagrams", 2014 *IEEE International Conference on Computational Science and Computational Intelligence (CSCI)*, Vol. 2, pp. 173-178, 2014.
- [5] S. Hao, *et al*, "Modelization of laboratory teaching management system based on UML". 2012 *IEEE Symposium on Robotics and Applications (ISRA)*, pp. 366-368, 2012.
- [6] J. Tomyim and A. Pohthong, "Requirements change management based on object-oriented software engineering with unified modeling language", In *Applied 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2016, IEEE, pp. 7-10.
- [7] L. Espinoza, *et al*, "Modeling a Facilities Management and Information System by UML", In *Applied Tenth International Conference on Information Technology: New Generations (ITNG)*, 2013, IEEE, (pp. 65-70).
- [8] L. Pang, *et al*, "Open-end fund investment stock management system modeling based on UML", In *Applied 6th International Conference on Information Management, Innovation Management and Industrial Engineering (ICIII)*, 2013, IEEE. Vol. 1, pp. 483-486.
- [9] X. Tang, and X. Zhou, "Analysis and design of personnel management system based on UML", In *Applied 6th International Forum on Strategic Technology (IFOST)*, 2011, IEEE, Vol. 2, pp. 1318-1320.
- [10] G. Bazydło, *et al*, "From UML State Machine Diagram into FPGA Implementation". In *Applied 12th International Federation of Automatic Control (IFAC) Conference on Programmable Devices and Embedded Systems*, 2013, Vol. 46(28), pp. 298-303.
- [11] G. Patel, and A. S. Priya, "Resolve the uncertainty in requirement specification to generate the UML diagram", In *Applied International Conference on Advances in Engineering and Technology (ICAET)*, 2014, IEEE, pp. 1-7.
- [12] F. Pecoraro, *et al*, "A methodology to identify health and social care web services on the basis of case stories", In *Applied E-Health and Bioengineering Conference (EHB)*, 2017, IEEE, pp. 217-220.

- [13] P. Sahoo and J. R. Mohanty, " Early Test Effort Prediction using UML Diagrams", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 5, No. 1, pp. 220 - 228, 2017.
- [14] H. Huang and D. Yin, "UML-based Requirements Analysis on Risk Pre-control System in Coal Enterprise". *Indonesian Journal of Electrical Engineering and Computer Science*. Vol. 11, No 7, pp. 4012-4019, 2013.
- [15] Z. Zhang, *et al*, "UML modeling for dynamic logistics system based on DEVS", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 11, No. 4, pp. 2168-2173, 2013.

BIOGRAPHIES OF AUTHORS

	<p>Dhafer Abdulameer Abdulmonim received the Bachelor's degree in Computer Science from University of Babylon, Babil, Iraq, in 2007, and the Master's degree in Software Engineering from Universiti Tun Hussein Onn Malaysia (UTHM), Batu Pahat, Malaysia, in 2015. He is currently a Lecturer of Computer Science at Directorate of Education, and Open Educational College, Najaf, Iraq. His research interests include software engineering, software testing and formal method.</p>
	<p>Zainab Hassan Muhamad received the Bachelor's degree in Computer Science from University of Babylon, Iraq, in 2007, and the Master's degree in Software Engineering from Universiti Tun Hussein Onn Malaysia (UTHM), Batu Pahat, Malaysia. She is currently a Lecturer of Computer Science at Directorate of Education, Iraqi Ministry of Education, Najaf, Iraq. Her research interests include software engineering, software testing and formal method.</p>
	<p>Bashar Alathari received a M.Sc. degree in software engineering from University of UTHM in 2015 from Malaysia. He is working as a lecturer and researcher in the university of Kufa, ITRDC. Among his research interests are research in software engineering and networking.</p>