

Building the Payroll Information System for High Education Institution Using UML – Master Thesis Work

Stefan Pitulić^{1*}, Siniša Ilić¹, Julijana Lekić¹

¹University of Priština/Kosovska Mitrovica, Faculty of Technical Sciences, Kosovska Mitrovica, Serbia

*stefan.pitulic@pr.ac.rs

Abstract: *The use of Students' Team work in developing the information system using the UML method to properly model the specific business processes in calculation the payroll of a high education institution and its implementation is described in this paper. The system has been developed by the book, following the lessons from the courses of the study programme Computer Science and Informatics. The existing legacy software was studied, gaps were identified and using the UML diagrams the new system was modelled. Based on the such model and using the free software components, the system was developed, tested and implemented by the students of master studies. Using the UML for initial modelling and updating the model according to the changes implemented during the system development, led to the fact that the model became the documentation based on which the system could be maintained even by developers who didn't participate in initial system development.*

Keywords: *UML, Development of Information Systems, Students' projects*

1. INTRODUCTION

In modern society, it is inconceivable to manually perform processes which require the repetition of relatively easy data processing, such as recording credit cards transactions, perform bookkeeping, recording of article sales, etc.

In order to educate students whose interests are in Information Technologies, a number of courses were designed within the different study programmes. The knowledge the students gain through the education process in the area of Information Technology (IT) and Computer Sciences (CS) is rather demonstrated through the students' practical work. In some High Education Institutions, the students of IT and CS have to work in the student team projects, with the task to develop software [1, 2], to build a solution [3], to test software [4] or to estimate the quality and usefulness of UML software modelling tools [5].

The team work is usually introduced for the students because this is how they will work once they enter the software development industry. It is rare for software engineers to work in isolation. As software engineers our students will also be assessed on their performance by others in their workplace, so they need to understand the appraisal process and how it can help to further develop their skills as practitioners [1]. In order to perform development tasks, the students had to pass programming modules covering Problem

Solving, Program Design and Implementation, and Object Oriented Program Design Development. The assessments of the students' project works are sometimes performed using the different software quality models (modularization, reusability, updatability, etc.) [2], or in number and severity of software bugs found [4].

The objective of our work was to assign to the group of the best students of the Computer Science Master study module a software development and implementation project that can be implemented and really used in commercial environment. The project life cycle had to fulfil all development and implementation steps (by the book) as it would have in commercial development company with the special emphasis on modelling the system before development and implementation. The specific task was to replace old one-user system for payroll calculation (with limited number of functionalities) with the modern Web based multiple users system. The work in the project was divided between the students: collecting the user requirements, building the model, developing the Web application, developing the database, implementing the hardware and software. During the work initially created models were updated by all parties.

At the end, the work of each student was incorporated in his/her master thesis work and they later defended their master theses.

When multiple users of different profiles need to perform individual business processes on their workstations using the system, when the architecture of the system is complex, and when the system is to be developed using the object oriented approach, then it is clear that the complex modelling tool must be selected during the development. Also, a developing team with clear shared tasks must be created and more complex server – client architecture must be designed. Our team was built from two teachers (Managers and Architecture Designers) and three students (UML designer, Java Web Application designer and MS SQL Server Database Designer). Documenting the business processes, modelling the classes of the application and database and designing the system architecture was performed using the UML [6, 7] tool; and a three-tier client - server architecture was selected.

2. LEGACY SYSTEM

The information system to be replaced was developed in Visual Basic programming language, it was compiled and the source code was missing.

Access to the information system was limited to only one user at a time. Although the business processes required the parallel input of data by users of different profiles (financial department to insert the actual monthly workloads of employees, legal department to update the positions (ranks) of employees and their work experience, head of study programme department to update teaching staff semester workload, etc.), only one user (from financial department) used to log in and compile data from the paper to the GUI. He/she used to do it on particular workstation where the application and a MS Access database were installed.

The GUI was designed in the form of MS Excel tables where a user could move between the cells of two different types: enabled and disabled for modifications by user. User could modify names of employees, the number of working hours, type of employees (teaching/non-teaching staff, academic rank of teachers), working experience, etc., and based on these inputs, the application could calculate values of cells disabled for modification by user (gross salary, net salary, deductions, etc.).

The codebooks (like academic ranks of teachers, names of the banks, bank accounts of employees, codes and names of municipalities where employees live in, etc.) didn't exist, all data were stored in many redundant databases (one database for one payroll calculation) with non-normalized tables with many columns and repeating data. For each Payroll calculation a new database used to be opened and the data from the database of the previous period was initially copied into it, thus allowing modifications for the new payroll calculation.

A simple MS Access database security was implemented for default user with a user name and a password. But, unfortunately, that protection could be easily overridden by using the free tools to find or delete a MS Access user's password. This database also lacked a built-in backup service.

3. MODELING A NEW SYSTEM

For the development of the new system, the UML method was selected and the system was modelled using the diagrams of: requirements, activities, use cases, user interfaces, communication, sequences, classes, as well as diagrams of data, software components and deployment.

3.1. Diagram of user requirements

The first step in creating the new or improving the existing software is to collect user requirements as well as business rules. In the UML, they are recorded in diagrams of user requirements. The requirements are usually grouped by the larger business processes. The requirements are collected by reading the existing documents (rulebooks, reports, etc.) and on clarification meetings with stakeholders (financial department, legal department, heads of study program departments, etc.). In order to document all requirements, for each one is assigned a unique label and relations between them is documented. Figure 1 shows user requirements related to employee master data.

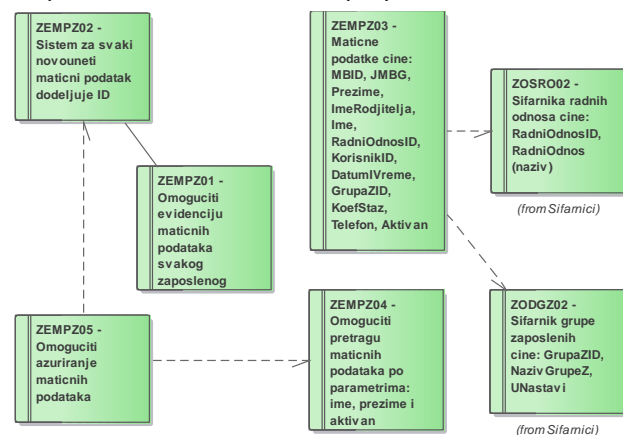


Figure 1. Diagram of user requirements related to employee master data.

3.2. Activity diagram

The activity diagram is diagram composed of activities, control flows, objects, object flows, decisions and partitions (swimming lanes) that show who perform activities. Activity diagrams can be used to model high-level business processes, but they can also be used to model each individual operation. These operations can be performed sequentially and in parallel. In Figure 2 is shown, a sequential flow of business activities for payroll preparation and calculation. As it can be seen there are three partitions (stakeholders - user groups): Legal department. Financial department and the Payroll system.

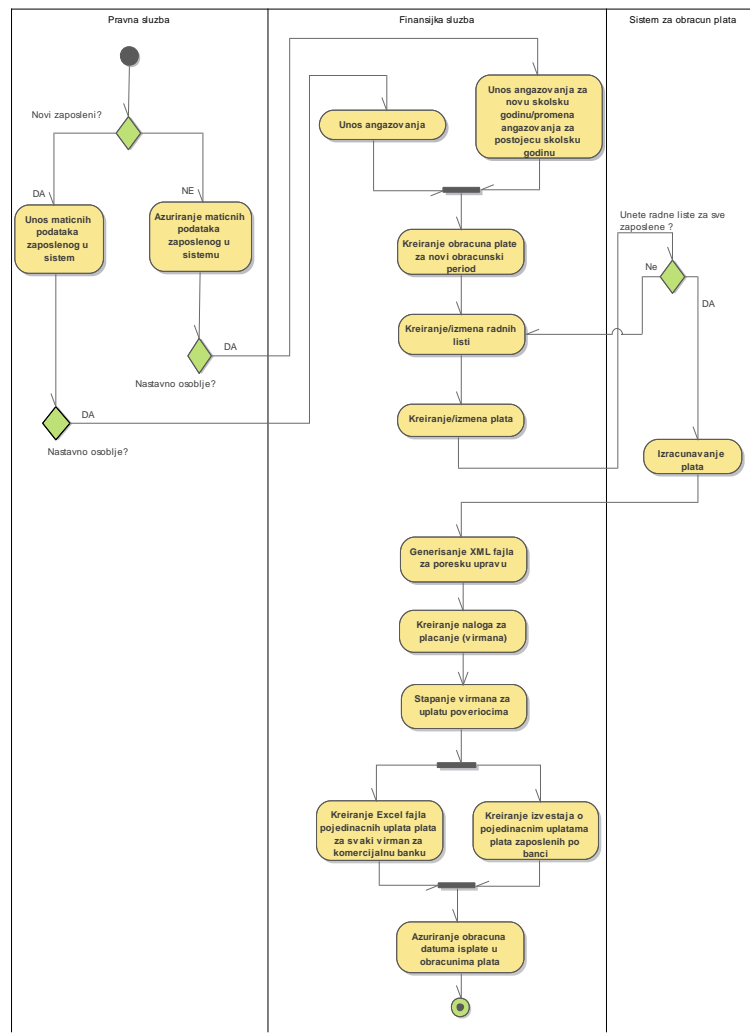



Figure 2. Activity diagram for Payroll Calculation system.

3.3. Use case diagram

The use case diagram describes how a software user can perform a process. The use case diagram is used to show how the system will respond to an action by the user, subsystem or external system. Each use case is described via message sequence between the system and one or more participants. The essence of this diagram is to define participants, use cases, as well as to define connections between participants and use cases.

A global (high level view for the whole system, not for individual functionalities) use case diagram is shown in Figure 3. It shows that the user must log into the system to perform any group of functionality, and depending on the role assigned to him/her, he/she will be enabled to perform only certain operations in the information system. Each user group has specific rights. There are three groups of users defined in our information system: administrator, financial and legal department user, and five high level use cases: Log in, Maintenance of the code tables and users, Evidence of the Employees, norms and workloads, Preparation and payroll calculation and Report generation.

In the UML tool we used, the icon  in a use case indicates that this use case has a more detailed diagram (decomposition) related to it.

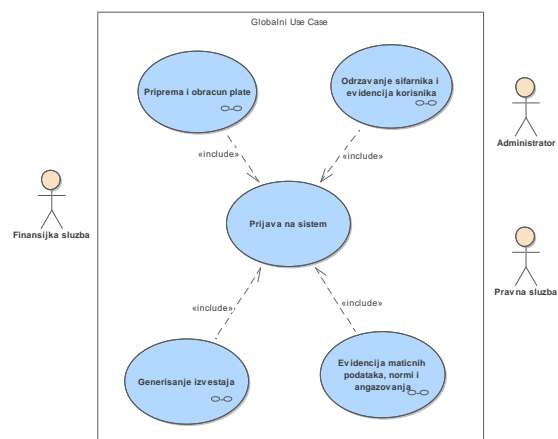


Figure 3. Global Use Case Diagram.

In Figure 4, the detailed diagram of the high level use case "Evidence of the Employees, norms and workloads" is presented. The user from Legal department is in charge for maintaining the basic data of employees, positions of employees, additional functions of employees (dean, vice dean,

heads of study programs, etc.), court decisions about suspensions of employees (alimony, other payments based on lost court cases, etc.), bank accounts of employees, the expected workload of employees according to the contract, etc.

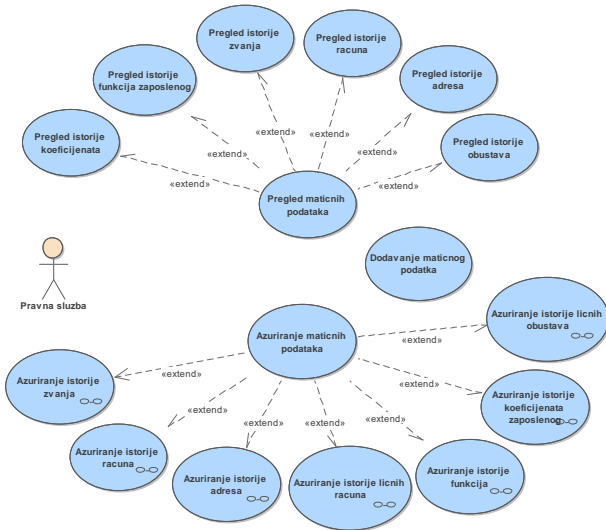


Figure 4. Use Case Diagram of Evidence of the Employees.

The use case is properly documented only when a **scenario** of using the use case is precisely described. The scenario consists of clear ordered messages (interactions) between user and a system. In order to client understand how future use cases will work, a system designer must assign the scenario for each use case (basic and

alternative scenarios), and show to a client the set of all use cases of the future system with their descriptions. For example, the diagram of use cases for maintaining codebook of banks (where employees have their bank accounts) with a description of the basic and alternative scenarios is shown in Figure 5.

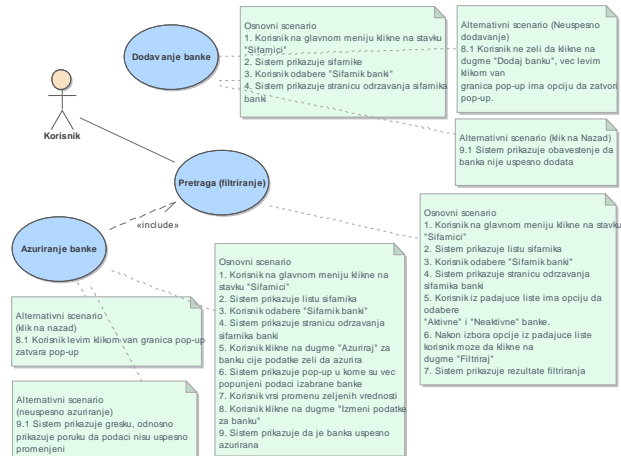


Figure 5. The diagram of use cases "maintenance of the banks".

A user may not understand such messages because most of them describe user actions on a GUI. That is why the scenario must have the reference on a proper user form. Usually in UML a rough GUI is drawn with its components: text boxes, buttons, drop down lists, etc. Because we have implemented the new information system, we show the real web forms in Figure 6.

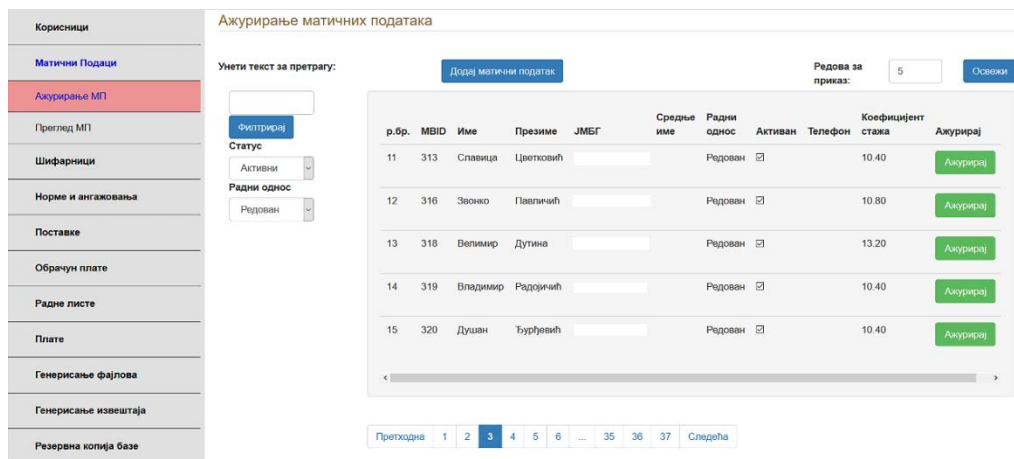


Figure 6. User interface for selection of an employee to update his master data.

4. APPLICATION MODELING

4.1. Communication diagram

After the client agrees to the proposed functionalities of the future system shown through the use case diagrams and the user interface layouts, the next step is to design a working structure (classes) of the system application. The first step is to create communication diagrams where for each or the groups of use cases the classes that must handle functionality of use

case(s) will be identified. This diagram also shows the connections between different types of classes and entities involved in messaging. The classes are of types: Boundary classes (in our case JSP pages), Entity classes (in our case Beans – the classes which objects keep the data) and Control classes (the classes where the system logic is implemented – Servlet, Data Manager, Peer classes). In Figure 7 is shown the communication diagram with the classes needed to implement functionality of the use cases related to

maintenance of the banks. One can notice the scenario of the use case in the left upper corner.

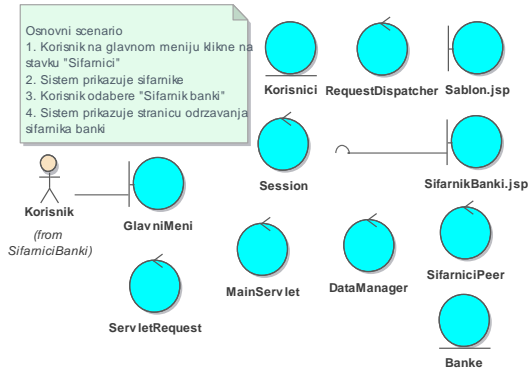


Figure 7. Communication diagram for use case: "search for the bank".

4.2. Sequence diagrams

Based on the proposed classes in communication diagrams, messages that are exchanged between those classes are entered into the sequence diagrams in order to implement the steps described in the use case scenarios. This type of diagram is easy for understanding as it shows the sequence of the messages between classes in time ordered from the top to the bottom. The decisions (if clauses) and loops are shown in rectangles with key words alt and loop. One can also notice the moments of creating objects of particular classes (objects appear somewhere in the middle of diagram) and

the references to execution of the other sequence diagrams (rectangles with a key word "ref"). The messages in the diagram must contain exactly the structure of methods or functions calls. The message must have a clear name with input parameters and return value type, because the user who looks at the diagram (in most cases the developer) must know how to capture those parameters, send a call to a function or a method and collects back the return value of the function. One example of the sequence diagram is presented in Figure 8.

4.3. Class diagram

Class diagrams are created based on the messages posted in the sequence diagrams. Messages that exchange data between different classes become public functions and belong to the classes receiving them (to which the arrow in the sequence diagrams is oriented). The attributes of the classes are mostly private. In Control class the attributes are usually pointers to the other classes (to which current class is communicating to) or pointers of data been received from other classes; in boundary class they are GUI controls (text boxes, drop down lists, buttons, etc.) and in Entity classes they are attributes that keep data (i.e. name, address, phone number, email address, etc. for the entity Employee).

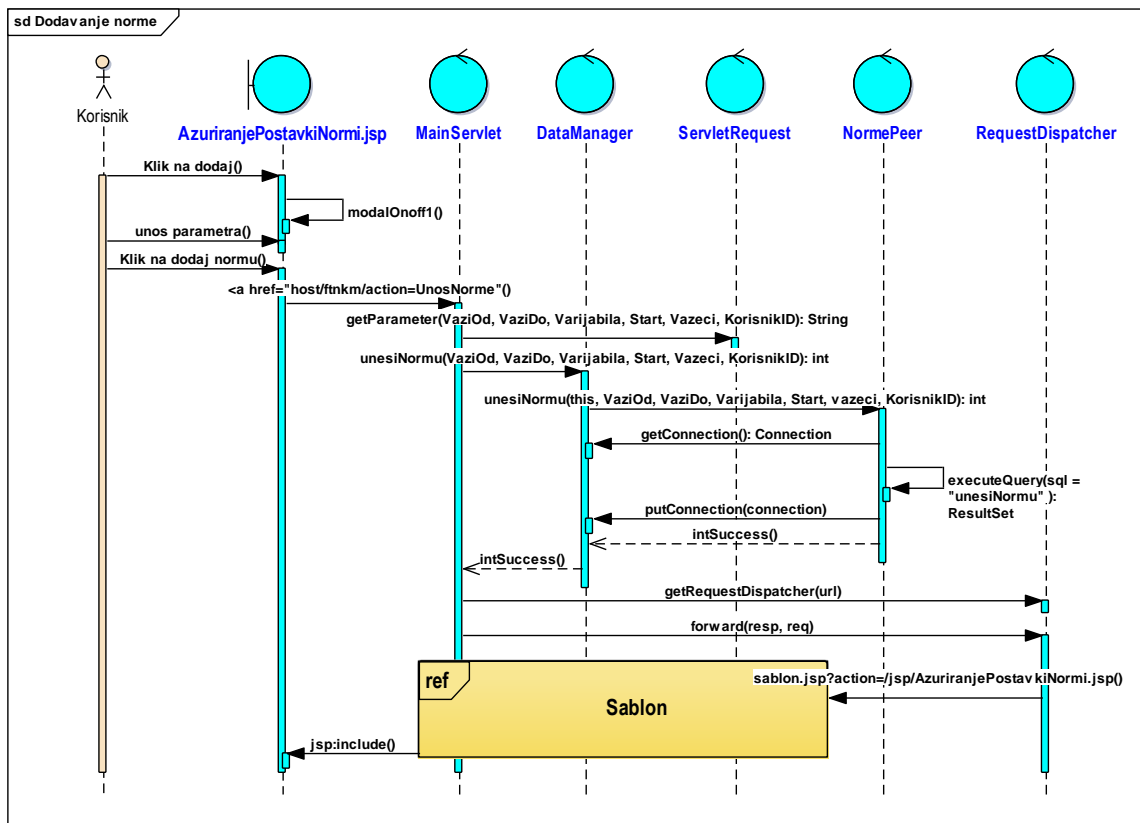


Figure 8. Sequence diagram for use case: "Adding the work norm".

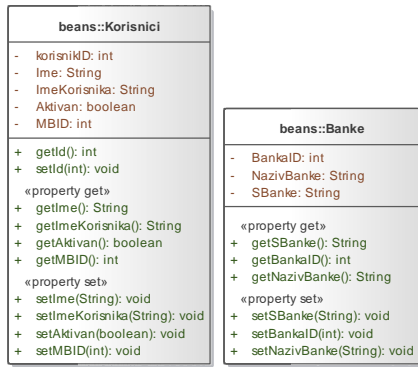


Figure 9. Class diagram of users and bank codebook.

Based on the messages on the sequence diagram in Figure 8, the partial class diagram with only entity classes (because of the space needed) is shown in Figure 9. UML tool can generate source code for each class in the diagram. The created class source code will have the constructor(s), list of attributes, and class functions or properties declarations with all parameters and return values. The source code will miss the implementation (a

code how each function is executed) and it remains as a task for the developer. Up to this stage the functioning (the dynamic) and the classes (the structure) of an application is modelled.

5. DATABASE DEVELOPMENT

From the entity classes (by involving only attributes of the classes) an initial database diagram can be created. Through the several iterations, some attributes will be suppressed (like pointers to other classes) and relations between tables will be established. In cases where entities are related with cardinality many-to-many new tables will be created.

Our database consists of several table groups: users and roles, employees and their properties, norms and worksheets, payroll and payment orders. The database scheme of group of tables "employees and their properties" is presented in Figure 10.

Because of the paper limitation, the diagrams of database stored procedures, triggers and other objects are not presented.

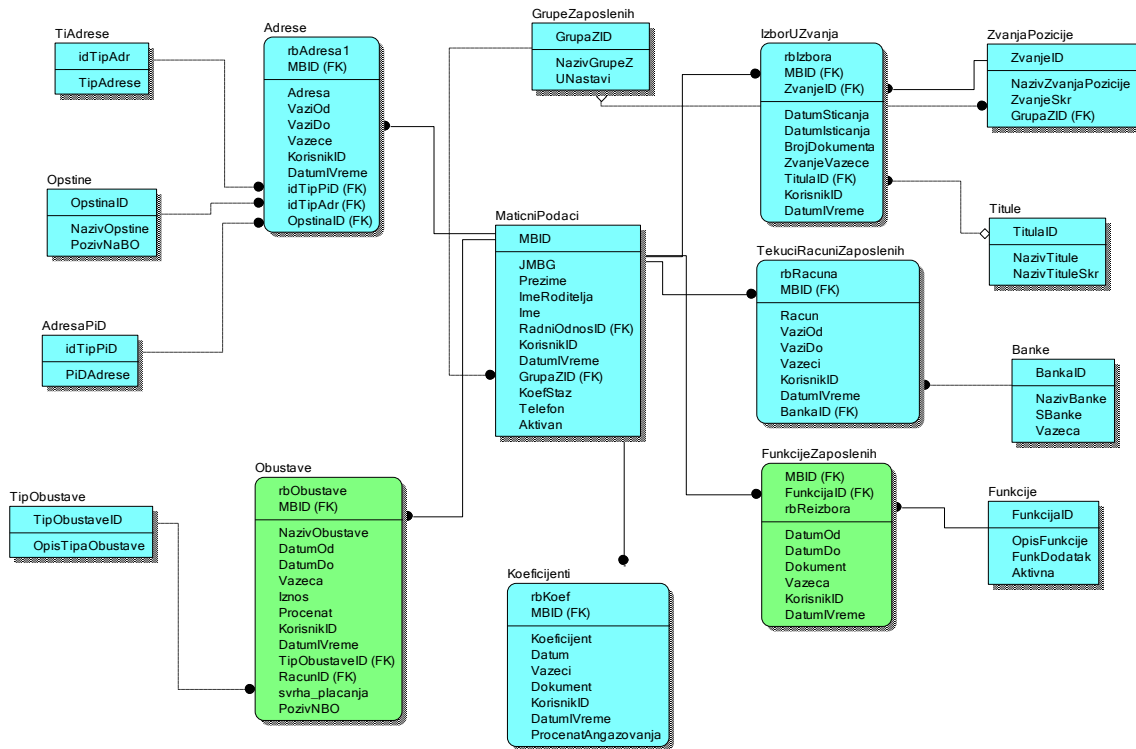


Figure 10. Schematic illustration of database entities related to "employees and their properties".

6. IMPLEMENTATION MODELLING

In UML modelling, it is not enough just to describe the building and dynamic blocks for application and database development, but it is also necessary to design an information system implementation model. It is important to document the software packages to be installed on computers in the production environment, and also to document how those computers are connected and configured.

6.1. Component diagrams

One of the most important features in the process of developing complex software solutions is their architecture, i.e. organisation of components that communicate with each other. Our application uses a three-tier architecture, which is composed of three logically independent tiers (presentation layer – business logic – database server) that communicate through the interfaces.

In our case, the first tier, the presentation tier, was made using JSP (Java server page) technology [8] with the HTML5 web pages [9] for presenting the content on a web browser of a client, combined with Java and JavaScript code. The standard computer network is a required medium that connects an application user to a second tier server in a computer centre.

Business logic is written in a Java programming language using a NetBeans development environment that uses a GlassFish application server to develop and run Web applications [10]. In order to application run on a server in a production environment, the following software components must be installed on the server side tier 2: Java Development Kit v8.x, GlassFish v4.0 application server configured to enable SSL, compiled application in WAR format to be attached to the GlassFish server, Jasper Reports v5.5.0 for reporting, DOM (The Document Object Model) parser for generating XML files, and for tier 3: Microsoft SQL Server Express 2014 SUBP and the payroll database to be attached to the SUBP.

The business logic tier retrieves the requested data from the data tier (using the JDBC interface [11]) and sends them to the presentation tier. The presented components are shown on Components diagram in Figure 11.

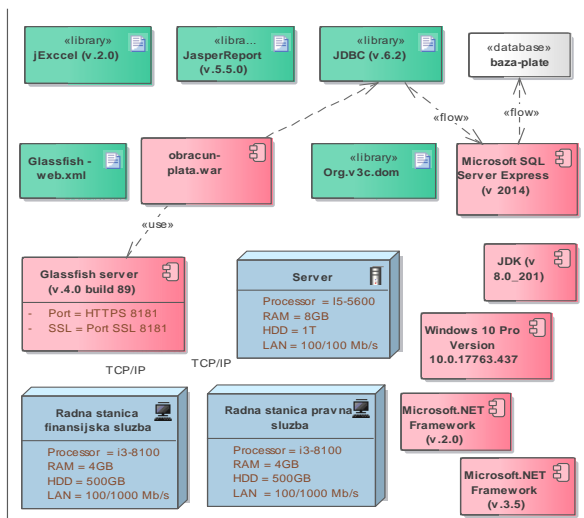


Figure 11. Component Diagram for payroll calculation software.

6.2. Deployment diagram

One of the tasks of a deployment diagram is to show a network architecture of the hardware components.

The server and workstations in the financial and legal departments are connected to the *Switch Linksys SE2800*, which is connected to a *Cisco* router, thus providing Internet access and communication between workstations and servers as it is a web-based three-tier application.

7. IMPLEMENTATION OF THE PROJECT

Working on the described project assignment is an example of development and implementation of a real software project through the teamwork. Students who participated on a team that implemented the project, applied the knowledge gained mostly from the following courses: Object Oriented Programming, Databases, Information Systems, Software Design, and Principles of Software Engineering in their master theses.

In addition, it was the opportunity for students to gain significant experience and to overcome in the most efficient way the problems of applying theoretical and abstract knowledge to solving practical problems. Students used the acquired knowledge, skills, tools and techniques to implement project activities to meet client requirements.

Compared to the research results presented in papers [1-5], where the main indicators were: involvement of students in tasks during the project life-cycle via contribution matrices, students' feedback obtained by questionnaires, estimation of quality of software related to modularization, reusability, updatability, etc. obtaining the number of bugs during test session, in our team work the main output was satisfaction of the customer after installing the software.

After first installation of the software for payroll calculation in very short time, thanks to modularization and reusability the system functionalities were extended to process calculation of compensations for contracted teachers from external institutions and compensations for work on scientific and commercial projects. Thanks to the proper UML documentation, the software is continuously updating with adding more functionalities by other students.

By continuously monitoring the results obtained and evaluating students' work, the teachers gained a real insight into the students' ability to participate in the real-world tasks. In the process of monitoring of students' acquired knowledge in the implementation of the project, the teachers (as part of the team) actively participated in order to project be developed and implemented in an efficient and effective manner. The teachers were controlling the UML diagrams during the analysis and project design and were giving tips for minor corrections of them. Also, teachers were assisting in the development of the application and in the development of the database, as well as in adjusting the architecture of the system in the production environment.

The key to successful customer satisfaction, and thus enhancing a business value, is good project management. During the team work on this assignment, students were able to learn how to manage the project: through the organisation,

planning and control of activities, through the division of the project into phases and tasks, through the rational alignment of all necessary resources and the coordination of performing the required activities.

As an integral part of this assignment, the students learned how to gather the necessary data, analyse customer requirements and model the future system so that the system satisfies the clients, and how to manage the risks, quality and time during the development of the system. One of the essential features of the project solution presented is using the UML (Unified Modelling Language) for initial modelling of the software and emerging it into the detailed documentation that allows easy maintenance and updating of the application and makes the system flexible for upgrading.

The core functionalities of the project were completed within the expected period of three months, and remained functionalities were developed and implemented in next three months, within which the full functionality of the system was obtained.

8. CONCLUSION

This paper describes the practical development of the real modern three-tier Information System by students of the master study programme using the UML as a modelling tool. By creating different UML diagrams, an object-oriented software is designed in a well-defined order. Using the UML diagrams, the other members of the developing team developed the web based application and the database. During the development some elements in diagrams are altered and these diagrams represent now the detailed development documentation needed to further upgrade the software.

The diagrams used confirmed the premise that:

- activity diagrams, user interface diagrams and use case diagrams can be used for making the proposal of functioning of the system to the client negotiations with the client (since any client can easily read the use case scenario and understand work on the offered user interface),
- by creating communication, sequence and class diagrams, the designer maps business rules into program building blocks in an object-oriented programming language, which are classes with their attributes and functions. Based on these diagrams a developer can easily obtain the source code of the classes and implement already identified functions and properties,
- data diagrams with a database scheme and a list of procedures, triggers and other objects can facilitate the database developer to properly design the database,
- component and deployment diagrams document the real setup of software and hardware components and enable system architect to

properly configure, connect and run information system

The implementation of a project management concept has led to the fulfilment of the project goals respecting the completion of the project in the planned time and with the anticipated resources and quality in accordance with the customer requirements and later users' satisfaction.

ACKNOWLEDGEMENTS

This paper was financially supported by the Min. of Education, Science and Technological Development of the R. of Serbia under project number TR-35026.

REFERENCES

- [1] M. Devlin, L.F. Marshall, and C. Phillips, "Fair Assessment of Contribution and Process in Student Team Projects" in Proceedings of the Informing Science + Information Technology Education Conference, July 31 - August 5 2017, Ho Chi Minh City (Saigon), Vietnam
- [2] F. Koetter, M. Kochanowski, M. Kintz, B. Kersjes, I. Bogicevic, and S. Wagner, "Assessing Software Quality of Agile Student Projects by Data-mining Software Repositories" in Proceedings of the 11th International Conference on Computer Supported Education (CSEDU 2019), pp. 244-251.
- [3] P.B. Cranwell, E.M. Page, and A.M. Squires, "Assessing Final-Year Practical Work Through Group Projects", Practice and Evidence of Scholarship of Teaching and Learning in Higher Education, Vol. 12, No.3, 2016
- [4] D. E. Krutz, S. A. Malachowsky, and T. Reichlmayr, "Using a Real World Project in a Software Testing Course" in Proceedings of SIGCSE'14, March 3-8, 2014, Atlanta, GA, USA
- [5] I. Silva, B. Alturas and A. Carneiro, "UML modeling tools: Assessment in perspective of users," 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), Lisbon, 2017, pp. 1-6, doi: 10.23919/CISTI.2017.7975818.
- [6] G. Booch, J. Rumbaugh, I. Jacobson (1998), Unified Modelling Language User Guide, Addison Wesley.
- [7] S. Ilić, A. Veljović (2017), Database Software Design at UML, University of Pristina, FTN K.Mitrovica, University of Kragujevac, FTN Čačak, Kosovska Mitrovica and Čačak.
- [8] JavaServer Pages, retrieved from: <https://www.oracle.com/technetwork/java/javaee/jsp/index.html>, last access: June 13th 2020.
- [9] C. Murphy, R. Clark and O. Studholme, "Beginning HTML5 and CSS3", Apress, 2012.
- [10] GlassFish, retrieved from: <https://javaee.github.io/glassfish/> last access: June 13th 2020.
- [11] JDBC Basics, retrieved from: <https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>, last access: June 13th 2020.