Development of an application for streaming music using Larman's method for software development

Andreja Rudolf
Information Technology School
Beograd, Srbija
andreaja32215@its.edu.rs

Abstract— **Larman's method is a software engineering methodology for software development that enables the development of software that is modular, scalable, usable, reliable, and easy to maintain. Larman's simplified method encompasses all phases of the software lifecycle: collecting user's requests, analyzing requests, designing, implementing, and testing. This article presents the development of a web application for streaming music that allows users to listen to their favorite songs, artists, and albums as well as edit playlists of their choice.**

Keywords-component: Larman's method, software engineering, PHP, MySQL, JavaScript

III. INTRODUCTION

Object-oriented analysis and design enable all participants in the development of the application to gain insight into the analysis and implementation of a specific problem simply and comprehensively. Analyzing and solving problems in this way has many advantages. This article presents the development of a web application for streaming music that allows users to listen to their favorite songs, artists, and albums, as well as to form playlists of their choice. This article is given an explanation for the technologies used as well as the methodology by which the application was designed. The application is shown as well as examples of its use. A system that would meet all the above needs must be carefully developed using a software development method that would allow a detailed review and development of the most important segments of the software so that it is upgradeable and adaptable to changes in the environment that are constantly happening. For these reasons, we decided to use Larman's simplified software development method to develop a software system.

Besides choosing the appropriate method for software development, it is very important to choose the technology in which the system will be implemented. In this regard, we decided to develop the application in PHP/MySQL environment.

IV. AGILE METHODS OF SOFTWARE DEVELOPMENT

Software development uses a life cycle method that is a framework for structuring, planning, and controlling the software development process. Software development, in this regard, takes place through certain phases, from the requirements for software implementation to the final product. There are many methods of software development and each of them has certain advantages and disadvantages. The choice of method depends on the techniques used in software development, organization, and development strategy.

The most famous life cycle models are:

- Waterfall model,
- Spiral model,
- Iterative - incremental model,
- Prototype model.

The listed models have certain shortcomings that are successfully eliminated by applying agile methods of software development:

- Larman's software development method,
- Unifield Process,
- Extreme Programming,
- Scrum.

Larman's method stands out, which is characterized by simplicity and ensures the independence of use cases and system operations. This method is characterized in that it is very close to developers because the phases of analysis and design are emphasized within it. The positive side of this method is the creation of contracts for system operations. If the contract is not defined for system operations in the analysis phase, there is a danger of mixing "what" and "how" the system operation works, and in that case, the process of designing the system operation would start too early, which is not recommended. In addition to the many advantages of this method, there are certain disadvantages related to the lack of a description of the management activities of the development team and the entire

Ukoliko postoje sponzori, navesti ih ovdje. U protivnom, obrisati ovaj text box. (sponsors)

project, to which the unique software development process pays more attention. In addition, Larman's method lacks risk management, unlike e.g. a spiral model-based precisely on risk management.

V. LARMAN'S METHOD OF SOFTWARE DEVELOPENT

Software development uses a life cycle method that is a framework for structuring, planning, and controlling the software development process. Software development, in this regard, takes place through certain phases, from the requirements for software implementation to the final product. There are many methods of software development and each of them has certain advantages and disadvantages.

Larman's method is based on an iterative and incremental model of the software life cycle, it is presented in use cases, it uses an object-oriented design method where UML (Unified Modeling Language) diagrams are used. Object-oriented software development describes a system analogous to a real system in real life and defines the relations and behavior of objects in the system as well as in reality.

UML is a language that visually displays the conceptual (maybe concrete) structure and specifies the system in connected diagrams. UML is not a programming language, but a tool that accelerates software development. The iterative-incremental model in the software development process is applied through the increase and simultaneous improvement of the system through cycles of analysis, design, implementation, and testing.

The software development process according to this method includes the following stages [2]:

- Requirement specification phase
- Analysis phase,
- Design phase,
- Implementation phase,
- Testing phase.

B. Requirement specification phase

The specification of the requirements is a phase of intensive cooperation between the designer and experts who know the domain of the problem. Requirements are described using the Use-case Model, which describes the set of desired uses of the system by the user.

User requirements represent the properties and conditions that a system or broader view of a project must meet. User requirements can be functional or non-functional. Functional requirements define the required system functions, while non-functional requirements define all other requirements.

In Larman, user requirements are described using a verbal model and a system use case model. The verbal model is a verbal description of what the software is expected to do, a model that describes the purpose of the software's existence. The use case is the way the software is used. It consists of a scenario, basic and alternative. [2]

In Larman's software development method, requirements are described using a Use-Case Model UML, which consists of a set of use cases (UCs), actors, and links between UCs and actors [3].

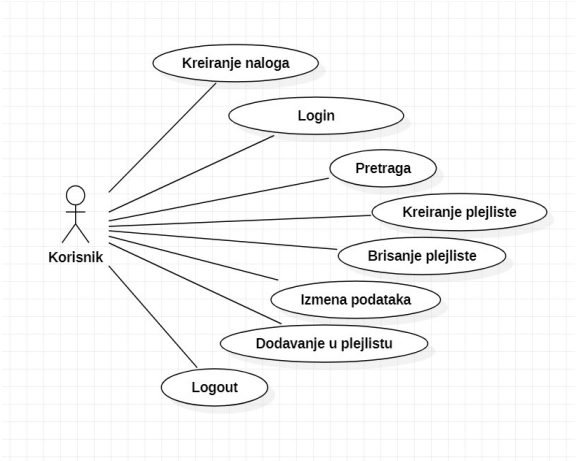


Figure 1. Use case digram

Text use cases have the following structure:

1. UC Name
2. UC Actors,
3. UC participants,
4. Pre-requirements that must be met for the UC to start being implemented,
5. Basic scenario of execution of UC
6. Alternative UC execution scenario.

During the execution of one use case, there are 5 types of possible events [3]:

1. AISO (Actor Invites the system to execute the System Operation)
2. APISO (Actor Prepares Input Parameters for the System Operation)
3. ANSO (Actor performs Non-System Operation)
4. SO (System executes System Operation)
5. OA (System sends Output Arguments to the actor)

The case of use with appropriate basic and alternative scenarios is presented textually as follows:

UC: Creating a playlist

Title: Creating a playlist

Actors: User

Participants: User and program - system

Prerequisites: The system is turned on, the user is logged in, and displays a page with all his playlists as well as a button to create a new one.

UC baseline scenario:

1. User invites the system to create a new playlist (AISO)
2. The system displays a pop-up (OA)
3. The user enters a name for the new playlist (APISO)
4. User confirms (AISO)
5. The system creates a new playlist (SO)
6. The system closes the window and displays the updated page (OA)

Alternative scenario:

The system cannot create an existing playlist and displays an error message (OA)

In the request specification phase, we should not use complex UML diagrams, but we must take care to specify requirements in a quality way because the choice of software and hardware components, as well as their configuration, depends on it.

C. Analysis phase

Based on the specification of the requirements, an analysis model is made whose main task is to describe the logical structure and behavior of the software system, ie. business logic software system.

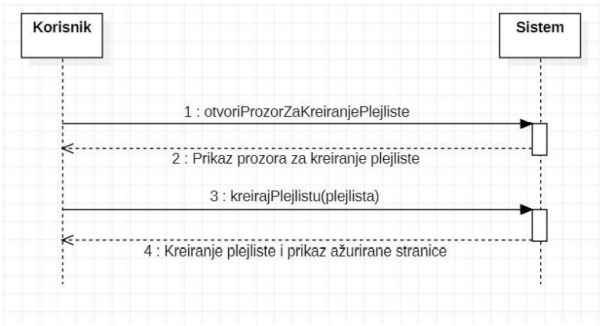
a) The behavior of a software system

The behavior of a software system is what the system does without an explanation of how it does it [5]. It is described using system sequence diagrams and system operations contracts. A system sequence diagram is made for each previously determined UC and it shows the events in a certain order, which establishes the interaction between the actors and the software system. The actor creates events that are an incentive to invite the SO, which means that the actor does not invite the direct execution of the SO but through an intermediary, ie. the recipient of the event. By the system sequence diagrams, only events that actors create within AISO actions and system responses to events as OA actions are represented. By identifying system events, the boundary between the software system and the actors is clearly defined. As a result of the analysis of system sequence diagrams, the requirements for the execution of the SO are obtained, ie the SOs that need to be designed are identified. The system sequence diagram for the case of using an individual entry is presented in the figure below.

DSUC: Creating a playlist

Baseline scenario:

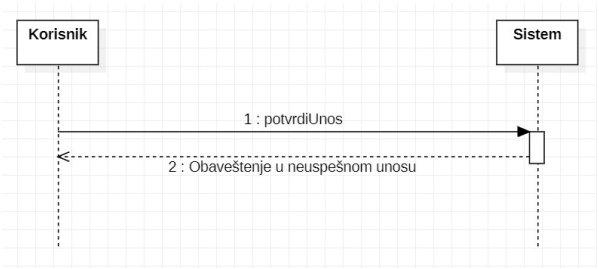
1. The user invites the system to open a window for creating a new playlist (AISO)
2. The system displays a pop-up (OA)
3. The user invites the system to create a playlist (AISO)
4. The system creates a playlist and displays an updated page (OA)



Slika 1. DSSK – kreiranje plejliste

Alternativni scenario:

Sistem ne može da kreira već postojeću plejlistu i prikazuje poruku (IA)



Slika 2. DSUS for alternative scenario – unsuccessful created playlist

Contracts are made for each of the observed SOs. The SO describes the behavior of the software system, so the contracts made for the SO describe the behavior of the system operation. The contracts describe what the SO should do, without explaining how it will do it. One contract is connected to one SO. Contracts consist of the following sections [5]:

1. Operation - the name of the operation and its input arguments.
2. Connection with UC - names of UC in which the SO is invited.
3. Pre-requirements - before the execution of the SO, certain pre-requirements must be met, ie the system must be in appropriate condition. The pre-requirements describe what needs to happen for the SO to take place, and it does not describe how it happened.
4. Post-requirements - after performing a system operation in the system, certain post-requirements must be met, which means that the system must be in the proper condition or the result of the operation is

canceled. Post-requirements describe the effects of SO execution.

UG contract: create a playlist

Operation: create Playlist

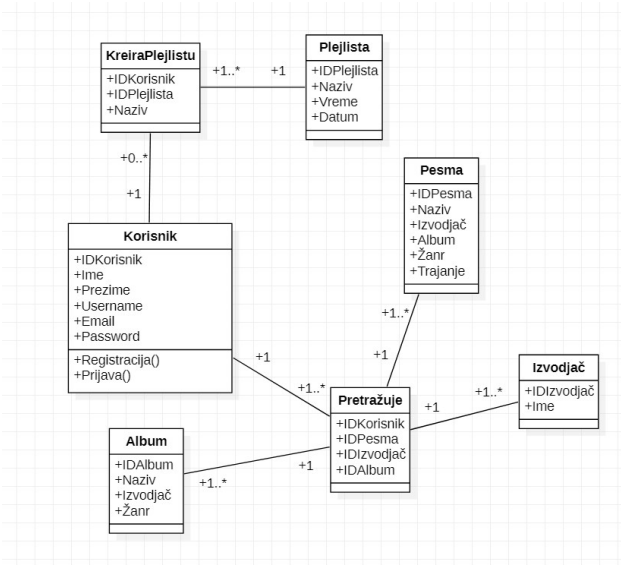
Pre-requirements: The user is logged in

Post-requirements: Create a playlist and display an updated page with all playlists

b) Software system structure

Unlike the definition of UC and the behavior of the software system where it was not necessary to know the object-oriented analysis at this stage, it is also very necessary.

The structure of a software system is described using a conceptual model that describes conceptual classes of problem domains, not software classes. The conceptual model contains conceptual classes, called domain objects, and associations between conceptual classes. Conceptual classes are identified based on attributes, which describe the properties of the classes, and which are entered by the user. Each attribute is related to a specific data type and has a specific value for a specific conceptual class occurrence. Association represents the connection between conceptual classes where each end of the association represents the role of a concept participating in the association. The role contains name, mapping, and navigation.



Slika 3. Diagram of a conceptual model

As a result of the analysis of the SC scenario and the creation of a conceptual model, the logical structure and behavior of the software system were obtained. Based on the conceptual model, by the simple mapping is created a relational model, based on which it is projected database, and in this case a relational database.

d) Relational model

User (IDUser, name, surname, username, email, password)

createsPlaylist (IDUser, IDPlaylist, Name)

Playlist (IDPlaylist, name, time, date)

Searching (IDSong, IDArtist, IDAlbum)

Song (IDSong, title, artist, album, genre, duration)

Album (IDAlbum, title, artist, genre)

Artist (IDArtist, name)

D. Design phase

Unlike the analysis phase, which describes the logical structure and behavior, the design phase describes the physical structure and behavior of the software system (software system architecture). Software system architecture design includes application logic, database, and user interface design. Within the application, logic is designed for the controller, business logic, and database. Business logic design involves designing the logical structure and behavior of a software system.

Designing the structure and behavior of a software system is preceded by defining the architectures of the software system. This article is used the classic three-level architecture of the software system.

Slika 4. Three-level architecture

The three-level architecture consists of the following levels:

1. User interface that represents the input-output representation of the software system.
2. Application logic that describes the structure and behavior of the software system.
3. Database that stores the state of software system attributes.

The user interface level is designed to be on the client-side, and the application logic and data warehouse are designed on the server-side.

Application logic consists of an application logic controller, business logic, and database broker. The controller is responsible for accepting the SO execution request from the client and forwarding it to the business logic responsible for the SO execution. Business logic is described by structure (domain classes) and behavior (SO). Domain classes must be on both the client and server sides. Database broker is responsible for communication between business logic and the database. As a result of the design, phase was obtained a detailed specification of the software system required for the implementation phase.

a) Sequence diagram and collaboration diagrams

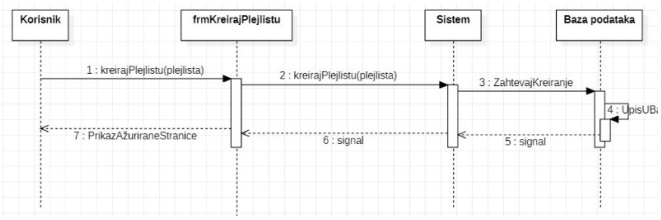
We will present a sequence diagram for the contract presented on the previous page

UG contract: create a playlist

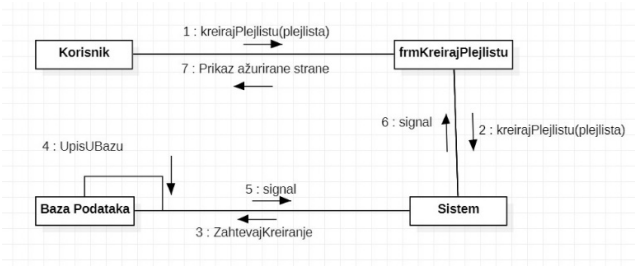
Operation: createPlaylist

Pre-requirements: The user is logged in

Post-requirements: A playlist has been created and a display of the updated page with all playlists



Slika 5. Sequence diagram UG – createPlaylist



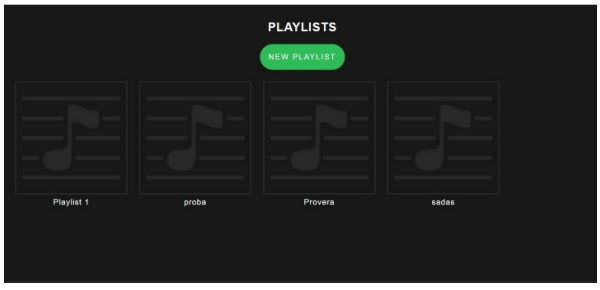
Slika 6. Collaboration diagram UG – createPlaylist

b) User interface design

The user interface represents the realization of the inputs and outputs of the software system. It consists of screen forms and a user interface controller.

UC playlist creation

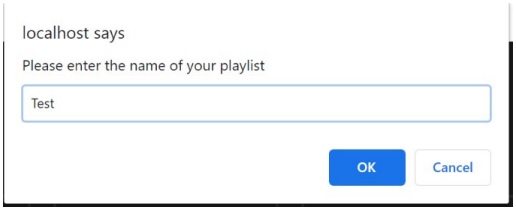
Pre-requirements: The system is turned on, the user is logged in, and displays a page with all his playlists as well as a button to create a new



Slika 7. Show all user playlists

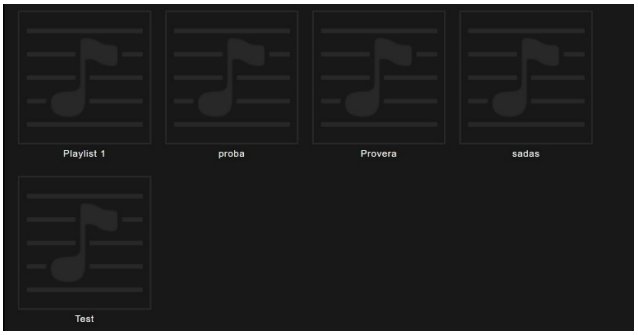
Basic scenario:

1. User invites the system to create a new playlist (AISO)
2. The system displays a pop-up (OA)
3. The user enters a name for the new playlist (APISO)



Slika 8. Confirmation of creating a playlist

1. User confirms (AISO)
2. The system creates a new playlist (SO)
3. The system closes the window and displays the updated page



Slika 9. Aurirana stranica sa plejlistama nakon kreiranja nove

Alternative scenario:

The system cannot create an existing playlist and display a message (OA)

Slika 10. Failed to create new playlists

E. Implementation phase

The implementation of the study example was performed in the PHP programming language. The database management system is MySQL. The graphical tool for database development is phpMyAdmin. The code editor is Visual Studio Code. The server is Apache and is installed together with PHP and MySQL using XAMPP packages. In the implementation phase, the designed software is divided into client-server architecture, so that all screen forms and the user interface controller are on the client-side (Tapestry frame), while on the server-side there is application logic and data warehouse, ie. software system.

So, we have two entities that are interconnected through society.

F. Testing phase

Constantly writing tests is one of the main features of creating quality software. Tests are written even before the methods of the classes being tested are implemented, to set the conditions that should be met by the implemented methods. Software testing, depending on the architecture of the software system, is performed through several independent testing units, where the testing units are software components obtained during the implementation phase of the software system. For each component, we make [3]:

1. Test cases that describe what the test should check,
2. Test procedures that describe how the test will be performed and
3. Test components that should automate test procedures if it is possible.

When the testing of all software components is completed, their integration is performed, ie tests of the integration of software components are performed.

CONCLUSION

This article is presented a web application that allows users to enjoy listening to music. Users can search for songs, albums, and artists at any time and create playlists that they can edit to their liking.

Larman's method provides a very efficient way to collect requests as well as the final product - software to be in line with user requirements. In addition, we get high-quality software, which is easy to maintain, and detailed and useful documentation. The illustrative documentation obtained by describing the scenarios of use cases using screen forms offers a user guide that will answer any question about the system and in this way facilitate our work with users. Various opportunities for improvement and changes are open:

1. Create a mobile application
2. Allow users to download songs
3. Allow users to listen to saved songs even without an active internet connection
4. Insert a display of music videos (video clips)
5. Take care of the display design of web applications for mobile devices and tablets
6. Introduce a subscription that will allow users to listen to the latest albums and songs from the same that are not available on other platforms (eg YouTube)

REFERENCES

[1] S. Pepić, Razvoj WEB aplikacija u PHP/MySQL okruženju, Beograd: Visoka škola strukovnih studija za informacione tehnologije, 2018.

[2] Svetlana Andjelić, Praktikum primenjenog programiranja (Skripta), I izdanje. Beograd: Visoka škola strukovnih studija za informacione tehnologije, 2013.

[3] S. Vlajić, Projektovanje softvera (skripta), FON, Beograd 2009.

[4] M. Fowler, UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd ed. Addison-Wesley Professional, 2003.

[5] C. Larman, Applying UML and Patterns: An Introduction to ObjectOriented Analysis and Design and the Unified Process, 2nd ed. Addison Wesley Professional, 2001.

[6] S. Vlajić, D. Savić, V. Stanojević, I. Antović i M. Milić, Projektovanje softvera – Napredne Java tehnologije, FON, Beograd 2008.

[7] Dž. Đonko i S. Omanović, Objektno orijentirana analiza i dizajn primjenom UML notacije, ETF Sarajevo, Sarajevo 2009.