BSP - S6 From knowledge graph databases to knowledge acquisition activities

Sunday 17th May, 2020 - 21:57

Stetsenko Olexandr
University of Luxembourg
Email: olexandr.stetsenko.001@student.uni.lu

Nicolas Guelfi
University of Luxembourg
Email: nicolas.guelfi@uni.lu

***Abstract*—**

1. Introduction

Technology has become very essential in our daily lives and continues to evolve continuously with the release of new technologies. Consequently, different sectors are affected and are forced to evolve. One of these sectors is the learning where new tools are developed for simplifying the acquisition of new skills, i.e. Languages.

Theses new tools are considered as online tools that can be assessed anywhere and by anyone who has an Internet connection. Each tool is dedicated for learning one or several skills that can be used in different sectors our lives. To acquire new skills, these tools are based on the traditional method such as learning through activities i.e. practical exercises, reading, listening, etc. The Bachelor Semester Project will focus to develop an online learning tool dedicated to learning words in the English language by doing quizzes. Each quiz is composed of several multi choice questions based on different aspects of the words i.e. definition.

The quizzes are generative which means that they are not created manually but automatically using generators. The role of generators is to create questions based on a graph from a graph database, called Neo4j. Additionally, the developed tool will give a freedom to the user for learning the words that he would like to learn.

As, the second part of the project, the state of the art of architecture modelling methods (I.e. 4+1 views with UML) for web applications will be explained and understood. Then, based on these architecture modelling methods, one or several methodologies will be selected and applied to our software in order to create different views based on UML diagrams. At the end, an explanation of created diagrams will be provided.

2. Project description

2.1. Domains

The Bachelor Semester Project is done in different domains of computer science such as Software Engineering, Web development and Software design.

2.1.1. Scientific.

Software Engineering is one of subfields of Computer Science. Software engineering is the application of engineering approaches for software development. During the development of the software, the developers deal with the requirements specification, design, development, testing and maintenance which are considered as the basic phases.

During the BSP, we will mainly focus on the design phase, also called Software Design.

Software Design is one of subfields of Software Engineering. Software design can be divided into three phases such interface design, architecture design and detailed design.

Interface design deals with the specification of the system with its environment. During the interface design, the internal system is considered as a black-box. The goal is to understand how the system will interact with its environment i.e. users.

Architecture design deals with the abstraction of overall structure of the system. The architecture design is used for describing globally the most important components, their properties, relationships, interfaces and interactions into the system.

Detailed design is the last phase of software design where all the components of the system are described in details in order to prepare for system implementation.At this stage, the properties, relationships, algorithms and data structures are defined in the design views.

2.1.2. Technical.

Web development consists to create web applications that are developed, deployed and accessed on Internet. Web development is divided in front-end and back-end development.

Front-end [4] development consists to design the graphical interface of a web application that allows the interaction between the system and the end users. Usually, the front-end is composed of several elements that are designed by CSS, HTML or JavaScript.

Back-end [4] development is hidden part of a web application. Back-end is composed of three parts such as functionalities of software, databases and server which allows to access and perform operations on the system.

2.2. Targeted Deliverables

2.2.1. Scientific deliverables.

During this project, several scientific deliverables are provided.

The first scientific deliverable is an explanation of architecture modelling methods based on ”4+1 architectural views”. In this deliverable, the views such logical, process, implementation, deployment and scenario views are described in order to understand how they can be used for modelling our developed software.

The second deliverable is the three UML diagrams based on the Logical, Process and Implementation views. The UML diagrams that are described are Class Diagrams, Process Diagram and Component Diagram which show the architecture of our software at abstract level. Additionally, an explanation on how theses diagrams are related between each other is also provided.

2.2.2. Technical deliverables.

The technical deliverable is a software dedicated to words learning in English Language. The developed software can be divided into two parts front-end and back-end.

The front-end is a user interface developed with React. The goal is to create a user interface where the user will be able to search words for learning and doing multi choice quizzes.

The back-end consists of a server which is used for handling the requests from the front-end. The back-end is composed of a server developed by Flask framework.

The server side allows the creation of a graph in a Neo4j graph database for a requested word which is then used for the generation of multi-choice question data that is send to the front-end. Additionally, a relational database is also used for storing the personal data of the user and the generated data about the questions into quizzes.

3. Pre-requisites

3.1. Scientific pre-requisites

3.1.1. Unified Modeling Language.

Unified Modeling Language (UML) is a modeling language dedicated to software architecture visualization. UML is based on different types of diagrams that are used for better understanding the software under development during the design phase.

It exists several types of diagrams in the UML that are dedicated for different purposes and shows the software architecture under different point of views. Generally, UML diagrams can be divided into two categories such Behavior and Structural. The behavior diagrams show how the software has to be behave under certain actions requested by the customers. The structural diagrams show the internal structure of the software, as example, relation between components, properties and interfaces.

3.2. Technical pre-requisites

3.2.1. Python.

Python is an interpreted, object-oriented, high-level and multi-platform programming language dedicated for web and application development. It has a dynamic semantics which define how and when the various constructions of a language should produce a program behavior. Python programming language allows a dynamic typing which gives the possibility to change type of value of a variable to another type without receiving an error.

The reason to use Python programming language in this project, is that it is easy to learn, to maintain and to develop.

3.2.2. Graph database (Neo4j).

Graph database is a database dedicated to store the data in the form of a graph. The graph is composed of nodes, edges and properties where properties are specific to each node in the graph and edges are used for indicating the relationships between nodes.

Additionally, the graph database provides the possibility to perform requests on the graph for information retrievement by using a Cypher querying language.

3.2.3. Relational database.

Relational database is a type of database where the data is organized into tables. A relational database can have one or several tables. A table is composed of rows and columns. Each column contains a label with a descriptive name and a data type that the stored data must fit. A row represent a single item of data. Each table has a defined structure and the stored data has to fit with this structure.

Additionally, as the database is composed of several tables, it can exist some relations between them in order to avoid redundancies.

3.2.4. Hypertext Markup Language. Hypertext Markup Language (HTML) is a standard markup language dedicated to documents design which are displayed in the web browser. HTML is only used for defining the structure of the document by using HTML elements such div,img,form, etc. Additionally, it is assisted by technologies such Cascading Style Sheets (CSS) and scripting languages such as JavaScript which are used for styling the document and improving the presentation by using colors, layouts and fonts.

4. A Scientific Deliverable

4.1. Requirements

In this section, the requirements of scientific deliverables are identified in order to understand the goals that must be achieved at the end of production. The scientific deliverables on which the requirements are focused are the UML diagrams which are based on the architecture of our software.

- R1: The produced UML diagrams should base on ”4+1 architectural view model” in order to provide an understandable design of our software from different views dedicated to different stakeholders such end-users, developers, system engineer and etc.

- R2: The created UML diagrams have to be consisted between each other. This mean that it should be possible to find parts from one diagram which can be related to another diagram in order to be able to assembly all the views in one piece for being able to develop the software. This can be achieved by using the same naming convention for components or parts, using colors and regrouping different components or patterns in the right groups.

- R3: The UML diagrams have to be also consistent with the implemented software. What means that all the components and operations provided in the diagrams have to be also present in in the software. The goal is to reduce the ambiguities between design part and development part of the software.

- R4: In order to understand better the software, the UML diagrams provided must not be of the same category. It exists two categories structural and behavior. The first allows to show the internal structure of the software and the second allows to show the behavior of the software. So, the final deliverables have to be composed of at least one diagram of behavior type and at least one diagram of structural type.

- R5: As, we are focused on the architecture design which is sub-field from the software design. The created UML diagrams should not be described into details and should only provide a global view of the software in order to stay at the right level of abstraction.

- R6: Before providing an explanation of the content of diagrams, an explanation of types of diagrams used for representing the software should be provided. Additionally, each type of diagrams used should be associated to one architectural view model.

4.2. Design

In this section, the explanations on how the UML diagrams are created are provided. Firstly, an explanation of ”4+1 architectural view model” is provided in order to understand the methodology used for software design. Secondly, some types of UML diagrams used in the final deliverables are explained.

4.2.1. 4+1 architectural view model.

Software architecture is one of the most important aspect in software engineering because it is the step before the implementation phase and it must be correctly done. When a software architecture is designed, sometimes some problems and questions may occur:

- A lot of things to express on one diagram/view dedicated to many stakeholders
- The designed architecture may not address all the customers
- To which part of software does the view belongs?
- What style to use for representing the view?

For solving theses questions and problems, the ”4+1 architectural view model” can be follow.

4+1 architectural view model is used for describing the software architecture based on multiple concurrent views. There are 4 basic views which are used for decomposing and representing the software architecture from different point of views. The 4 basic views are:

- Logical view
- Process view
- Physical view
- Development view

When all the four views are created, a new view called ”Scenario” can be produced based on the concepts described in the basic views.

Now, let’s look individually on each view in order to understand what have to be present in this views.

Logical view is used for representing the functional requirements of software in order to understand the services that must be provided by the system to the end-users. The system is decomposed and represent abstractly in the form of objects or object classes. They deals with the principles of abstraction, encapsulation and inheritance.

For designing the logical view, a class diagram or class templates diagram can be used. Both allow to represent the classes used in the software and the relationships which shows the relation between classes.

4.3. Production

4.4. Assessment

5. A Technical Deliverable 1

5.1. Requirements

5.2. Design

5.3. Production

5.4. Assessement

6. Conclusion

eND

References