Technological Features of Cross-Language Migration from PHP to Python of Software Products Working with Intensive Data

Barakhnin Vladimir[1], Kozhemyakina Olga[2], Revun Artem[3], Shashok Natalia[4]

Federal Research Center for Information and Computational Technologies, Novosibirsk, Russia

[1] bar@ict.nsc.ru
[2] olgakozhemyakina@mail.ru
[3] pm@artemrevun.ru
[4] n.shashok@g.nsu.ru

Abstract. This article describes the methods of cross-language migration from the PHP programming language to the Python programming language on the example of modernization of the search module of the system for complex analysis of poetic texts. The choice of language was determined by the need of integration of the module in the system, and by the fact that Python has a large number of open libraries that make it possible to work with machine learning technologies, what is an advantage in the intensive data processing. The main problems of cross-language migration are considered and the methods to solve these problems are proposed. The proposed solutions concern the data security, the separation of module representation from the data model, the work with database via declarative mapping. In the process of cross-language migration, the search module was rewritten and all problems were solved, what allowed to integrate it into the information system for complex analysis of poetic texts.

Keywords: cross-language migration, PHP, Python, intensive data, text analysis information system, natural language processing, analysis of poetic texts.

1 Introduction

In the practice of the development of the information systems, there are a number of problems associated with the need of the modernization of the source code of an application that is already in active usage. This can be caused, for example, by the migration of an information system into a new programming language. It is necessary to correctly reproduce the work of all parts of the old code so that there is no noticeable difference in the code execution time for the end user.

This article describes the features of migration of a software product that works with intensive data from the PHP programming language to Python.

The Python programming language, together with the Flask framework, is used to develop the applications in various scientific fields without any restrictions, for example, the creation of the framework for Covid-19 diagnostics [1] or the improvement of the user interface of the LHCb Software "control panel for nightly builds" [2]. In particular, the choice of language and framework for the improvement of the interface of LHCb Software was not chosen by chance: the developers needed to provide the system support and to have clear, readable code, as well as to use the tools which are already used for developments earlier [3]. Similarly, given that the other modules in the system of complex analysis of poetic texts are written in Python, the cross-language migration with the usage of the Flask framework can be carried out to modernize the search module in the system.

Let's describe the technology for solving this problem on the example of one of the modules of the information system for the automated complex analysis of poetic texts.

The data on poetic texts which are extracted in the process of their multilevel analysis (phonetic, metro-rhythmic, strophic, lexical, etc. characteristics) in accordance with the approach described in [4], refer to intensive data, since they are extracted directly from the source texts, and are not obtained through the usage of hypothetical patterns expected from the data, and are also planned to be placed in the public domain. In addition, according to [5], our system refers to the systems which are working with intensive data due to the constant expansion of the volume of accumulated data - both in connection with the expansion of the range of analyzed poetic texts, and in connection with the increase in the set of extracted characteristics.

In accordance with the modern realities of the software development, there was a need to modernize this module, which works with the corpuses of poetic texts of large volume.

To achieve this goal, the following tasks were set:

- to investigate the correctness of the considered software;
- to detect the problems that exist in the initial code;
- to substantiate the choice between updating and rewriting of the code;
- to substantiate the choice of the final programming language;
- to define the architecture of the application;
- to develop a new version of the software product.

2 Description of the Problem

The purpose of the development of the information system for the automated complex analysis of poetic texts [6], implemented at the FRC ICT, is to provide the philologists with a tool for the analysis of large-volume text corpuses.

The described information system (IS) is a complex of separate modules, each of which is an independent software product that performs a specific task and, at the same time, is integrated into the system.

One of the modules of the system, which is responsible for processing of big data in the database, was written using the PHP programming language, while the Bootstrap framework was used to organize the user interface. The code provided the solution to

all given tasks, namely, the interaction with the database and the generation of the user interface. PostgreSQL was chosen as the database management system. This relational DBMS, although not traditional when it is used in conjunction with PHP, has the ability to analyze the main linguistic objects (token/word/phrase/sentence/text), what is in demand when processing text information [7].

However, the further development of the system is carried out in the more universal programming language Python 3, what leads to the problem of the integration of the modules written in PHP with the new modules.

During the analysis of the initial code, the following problems were found:

- the complexity of the code, which causes a lot of internal dependencies, what leads to low scalability;
- the inseparability of the visual representation of data and the data model;
- critically low level of the security of IS.

Thus, despite the fact that the modules written in PHP are functional, they need to be improved: if the system is functioning, but it is difficult to maintain it or expand it, then, according to general theoretical guidelines [8], it cannot be considered correct.

2.1 Approaches to Improving the Initial Code

There are two approaches to improving the initial code: the refactoring and the complete rewriting.

The refactoring is the improvement of the initial code without changing the result of its execution [9]. The periodic refactoring can make it easier to understand how the program works, and in some cases it helps to find the various errors, although the error detection is not its main purpose. Thus, the refactoring leads to the optimization of the previously written program, improves the quality of code readability, and allows to add the new features. However, the refactoring is a very time-consuming process and can take a long time.

A complete rewriting of the code has the same effect as refactoring, but it is usually used either when the adding of new functionality within the old concept is unnecessarily time-consuming (for example, due to incompatible architectures), or when there is a need of the migration to another programming language. In addition, a complete rewriting carries the risk of adding the new errors that may not be obvious.

Also, regardless of the chosen approach, it is necessary to test the resulting product during the development process, what allows to find the errors in the operation of the modernized code and to solve them quickly, without having to return to the problem code after a while.

To choose between two different approaches, it is necessary to analyze the initial code of the application for problems and their severity. After that, it is possible to make a conclusion about the need for refactoring or rewriting of the code.

In addition, if there is a need to choose between different programming languages, the differences between the application deployment in probable programming languages should be determined.

2.2 Analysis of the Inseparability of the Data Representation and Data Model from Each Other in the Initial Code

Let's analyze the emerging problems in the initial code.

The user interface of the module is written in HTML using the Bootstrap framework, which includes CSS and the JavaScript programming language, while some of the search fields contain dynamic parameters that the interface requests from the database during the initialization.

Let's consider the part of the code under discussion that is responsible for the dynamic construction of the "Author" field. The program code presented below makes a request to the database, and, returning the values, generates a field of the select type using them, which is then displayed as a drop-down list of authors.

```
<div class="form-group">
 <label class="col-sm-2 form-check-label">
         <input type="checkbox" id='authorCheckBox'
name='authorCheckBox' class="form-check-input" on-
click='activateAuthorsFieid()'>Автор</label>
 <select class="selectpicker" multiple id='authorsList'
name='authorsList[]' data-live-search="true" data-max-op-
tions="10" data-actions-box="true" data-width="auto" ti-
tle="Ничего не выбрано..." data-size="5">
         <?php
             $select_authors = "SELECT \"LASTNAME\",
\"FIRSTNAME\", \"MIDDLENAME\" FROM \"AUTHOR\";";
             $authors = pg_query($select_athors);
             while ($authors_list = pg_fetch_array($au-
thors)) {
                 $author_string = $authors_list[0]."
".mb_substr($authors_list[1], 0, 1, UTF8).". ".mb_sub-
str($authors_list[2], 0, 1, UTF8).".";
                 echo '<option>'.$author_string.'</op-
tion>';
             }
         ?>
 </select><br><small class="col-sm-6 form-text text-
muted">Выберите автора из списка или воспользуйтесь
поиском</small>
</div>
```

This example shows that the work with the data is not separated from the formation of the interface. In particular, if there is a need to call the same query in another part of the system, for example, to further access to this data, it should be copied. Thus, the procedural architecture shown in the example leads to code redundancy, and, as a result, to the possible errors.

2.3 Analysis of the Security Level of the Search Module

Another problem concerns the analysis of the level of data protection.

The analysis of the protection level of the search module for poetic texts consists of the following actions:

- the analysis of the initial code of the application;
- the analysis of compiled SQL queries for the vulnerabilities such as SQL injection;
- the implementation of an SQL injection attack.

The SQL injection is an attack for which an attacker inserts the deleterious code into the text of a database request, what makes it possible to execute an unauthorized request. In this way, a hacker can get stored data or cause damage to the database.

The main reasons for the possibility of SQL injection are the incorrect processing of special characters and data types, as well as the dynamic query construction [10].

There are many ways to implement such attacks, as well as to protect against them. One of the main methods of leveling and preventing the potential threats is to check the correctness of the data which is entered from the client to the server. This method is implemented through the special characters escaping, the strict data type conversion, and the prepared or parameterized queries.

However, during the analysis of the application source code, it was found that there is no verification of the correctness of the entered data, requests are made dynamically, and the data is transmitted to the DBMS in the form in which they were sent by the user, without any processing of special characters or restrictions on data types.

To confirm the presence of the vulnerability, the attack was made by injecting "1 OR 1=1", entered in the filter field by metrorhythmic statistics, regardless of the selected metric. If the application works correctly, such a string must lead to no search results for all metrics, but instead, the entire corpus of poetic texts available in the database is displayed. Accordingly, the SQL injection attack is successful, and the user may receive incorrect data.

2.4 Differences between the Creation of Applications in PHP and Python

Today, the Python programming language is more popular than the PHP programming language. This is indicated by the TIOBE index, where Python ranks second and PHP ranks ninth [11]. The TIOBE index is calculated based on key queries in the most popular search engines: Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu. It is also worth noting that the TIOBE index does not indicate the best of the programming languages, but only reflects the activity of the global community that searches for information on each programming language.

Both languages offer a large number of frameworks for development, what allows to achieve fast, secure and well-scalable applications. However, Python has a large number of open libraries that let to work with machine learning technologies in a web environment, what is an advantage when working with big data.

Python is a language with strong implicit dynamic typing, and PHP is a language with weak implicit dynamic typing. [12] Therefore, in the tasks where PHP allows to mix the different data types in expressions, it's necessary to look for other solutions for Python.

Since Python is focused not only on Web development, for the interaction of a Python program with a server, it is required to use the WSGI interaction standard [13], unlike PHP, where such technologies are not necessary. This technology is an intermediate link between the web server and the application, what allows to process the information outside the application before sending it to the web server, what, in turn, can significantly improve the efficiency of the client-server application.

PHP was not created as a programming language and was originally a scripting language embedded in html-pages, so the separation of data from the interface in PHP is a non-trivial task. In turn, Python is a general-purpose programming language, so the separation of data and interface is achieved by simple methods.

3 Modernization of the Initial Code of the Discussed Software Product

Taking into account the above, the decision about the rewriting of the module in Python 3 was made.

The rewriting of the module separates the data model from the module representation, and also allows to solve the scalability problem and to use the data model in other tasks. The module retains its isolation, what improves the possibility of its usage in the further development of the system.

As the basis of the system, the Flask framework [14] was chosen, which, using WSGI, provides simple interaction between the server and the application, has a built-in html template engine for creating the user interface, allows to evaluate the efficiency of the application using internal tools, is suitable for building an application that has various modules, and is also the project with open initial code.

The strong side of the Python language is the number of libraries and tools available for it for both data processing and web programming, what distinguishes it from other languages, such as PHP. The strong side of the Flask web framework can be called its easiness, since it requires a small amount of code to get both a web application and a simple web-API from the functions and modules which are written in Python. [15]

3.1 Work of the Module with the Database

The database that the described module works with is described in detail in the article [6] and does not need any changes. The task when the module is being rewritten is to combine the working database with the Python SQLAlchemy module, which uses the ORM approach.

Let's consider the ER diagram of the poems database [16].

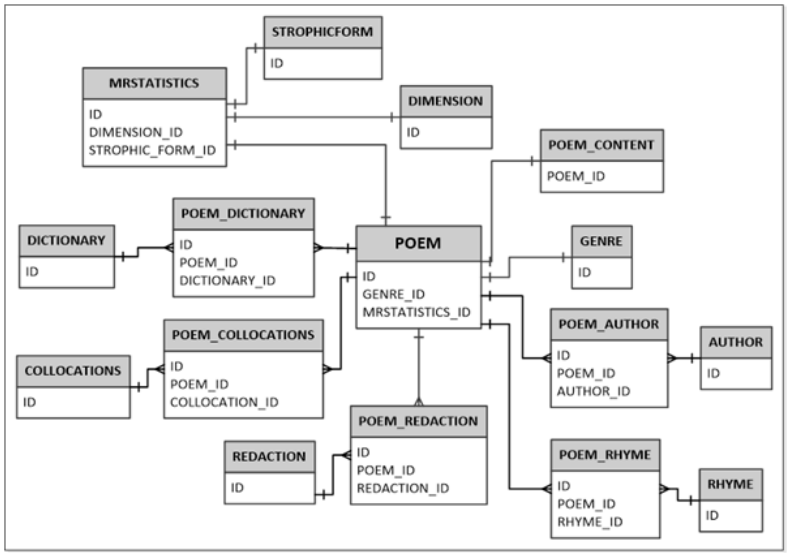

Fig. 1. ER-diagram of the poems database.

By initializing all the entities which are presented in the diagram as data models in the SQLAlchemy ORM, the universal tool for working with the database is get. This will allow to use the data from the database not only for search, but also for other purposes that may be required for the implementation of certain functions of the system.

3.2 Declarative Mapping of a Procedural Database in the Concepts of the SQLAlchemy Object-oriented Paradigm used in the Flask Software Package

Since the tables in the database already exist, the task is to declare each table as a class in accordance with the SQLAlchemy syntax [17]. A base class is created, from which all classes created on the basis of the tables shown in the ER-diagram are inherited (see Figure 1). Next, the attributes are defined for each class according to the attributes of the corresponding table.

The complexity of this declarative mapping is that for each author represented in the database, a set of tables is created according to its unique identifier, where the identifier serves as a postfix. In this case, the tables with the same structure store the data from different authors. The description of each such table as a class separately from each other is not an effective solution, because there is a code redundancy. It was decided to create a function with a numeric parameter that returns a class object depending on the specified parameter – the unique identifier of the author. This solution allows to avoid a redundancy, since there is no need to add a description of a new class to the database when a new table is being created, and it is possible to create the queries for each author dynamically without the need to modify the initial code of the program.

After declaring the classes, the tables can be used to display the information in the user interface, as well as to search the tables by the specified parameters.

3.3 User Interface Development

Since the Flask software environment uses the Jinja2 template engine as a tool, the interface templates should be prepared. All dynamic elements in the template use the rules adopted in Jinja2, and the static part is designed to match all the rules for creating the web documents.

During the template design process, at the analysis of the previous version of the interface, it was found that parts of the interface can be divided into separate templates. The separate documents were created, representing various parts that are common to most pages: the set of meta tags, two navigation panels (a general and for DB queries), and the error panel. If necessary, these templates are embedded in other html documents. In particular, the set of meta tags and the general navigation panel are embedded in any html-documents, while the navigation panel for database queries is embedded in the template for searching the database, the template for viewing saved user queries, and the template for viewing all the works available in the database. In addition, the pages for user registration and authorization were rewritten.

3.4 Development of the Initial Code of Application

The Flask software environment uses the concept of "blueprints" to create the various components of the application. The blueprints are used to expand the application, and they can also be applied to divide the code into logically related sets of operations.

The operations in the described module can be divided into two sets: the functions for working with the data of the corpus in the database and the functions for working with the user's personal data. Based on this, the blueprints 'main' and 'auth' are written, respectively.

The blueprint “main” is a set of the following operations:

- poems_list – the output of all the poetic texts available in the database;
- index – the output of the main page of the interface;
- search – the output of the form used to make a query in the database for data sampling from the corpus;
- search_result – the output of query results to the database;
- save_search – the saving a specific DB query for the user;
- saved_search – the output of all saved queries for the user.

The blueprint "auth" combines the following operations:

- signup_post – the creation of a user,
- login_post – the authorization,
- signup – the view of the user creation page,
- login – the view of the authorization page,
- signout – the exit application.

The save_search and saved_search operations are only available for authorized users, just like in the old version of the application. The difference is that when the code was rewritten, the Flask Login library was used, which automates the process of creating of a user session, and in the PHP version of the application, the session was created manually. At the attempt to open the pages corresponding to these operations without authorization, the user sees an error message and a request to log in.

The search_result operation has been split into the separate functions, since the processing of a user-filled form and the building of a query are different operations. The function of the building of a query to the database is also divided into component parts, depending on which part of a query is being built, since the determination of the columns of the table from which the selection is made and the determination of the data filtering parameters can also be understood as two different actions.

3.5 Results

The end result of cross-language migration of the application is a code which is completely separate from the presentation of html pages, and which dynamically builds the queries to the database depending on the parameters set by the user. The application structure has been rebuilt (Figure 2).

4 Conclusion

The technological features of cross-language migration between PHP and Python were identified on the base on the solved tasks.

Compared to the PHP language, Python does not allow does not allow to embed the code in html pages, so if there are similar pages in the project, it becomes necessary to isolate the logic inherent in the embedded code. The setting up of the correct and secure database queries is done in simpler ways than in PHP, where it may be non-trivial. The Python scripts and the related frameworks may not be supported by default on the using web server, and if there is no opportunity to use another web server, it is necessary to make additional settings.

There also may be the cases where the different data types are mixed in the initial code of a PHP application and the implicit but valid conversions occur. In such situations, when an analog of the application in Python is being written, it is necessary to have a clear idea of what data is used in a particular situation, and to apply a secure data type conversion. This imposes the certain limitations related to error processing: it is necessary to convert all the data sent by the user.

Acknowledgement. The study was carried with the support of the Russian Science Foundation (project No. 19-18-00466).

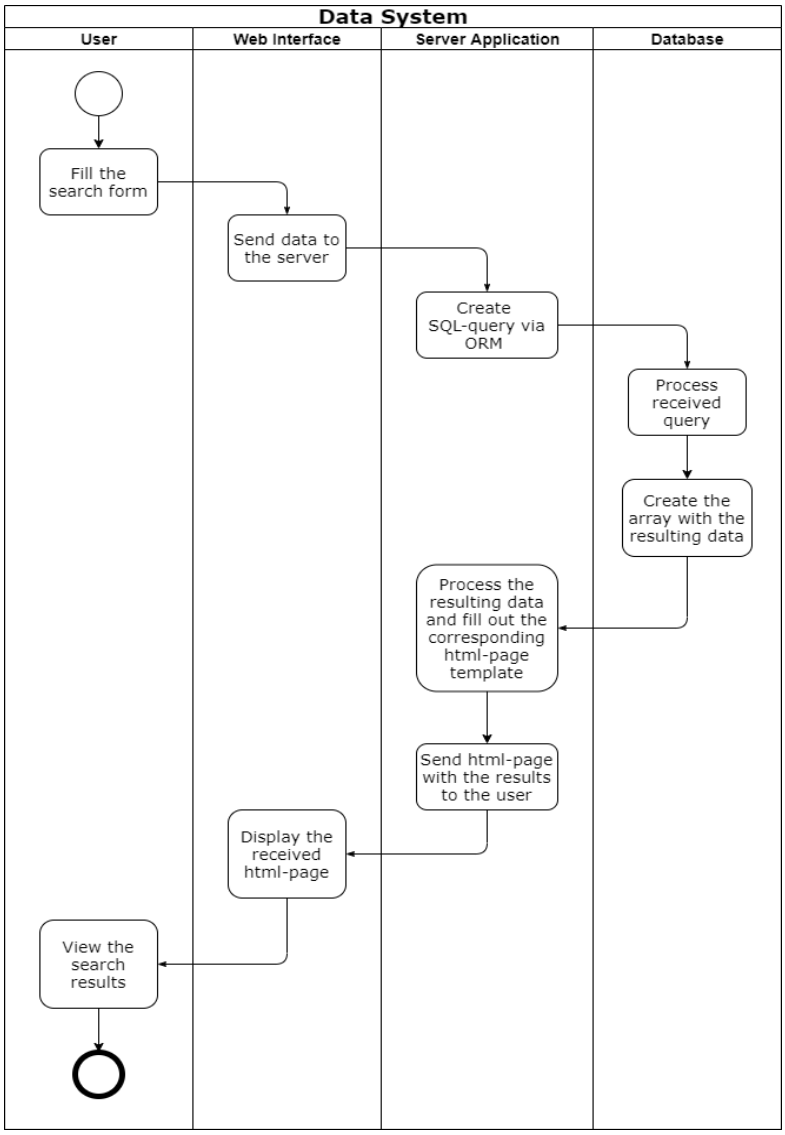

Fig. 2. The process of the work of the search module in the flow diagram after migration.

References

1. Palimote, J., Ebenezer, O., Atu, L.: A Framework for Diagnosis of Covid-19 Infection using Deep Learning Approach. International Journal of Computer Applications in Technology, 10(2), 93–98 (2021).
2. Clemencic, M., Couturier, B., Kyriazi, S.: Improvements to the User Interface for LHCb's Software continuous integration system. Journal of Physics: Conference Series, 664 (2015).

3. Clemencic, M., Couturier, B.: A New Nightly Build System for LHCb. Journal of Physics: Conference Series, 513 (2014).
4. Steve K., et al: Data-intensive Science: A New Paradigm for Biodiversity Studies. BioScience, 59(7), 613–620 (2009).
5. Michael F., Barbara R., Florian A.: On Testing Data-Intensive Software Systems. In: Security and Quality in Cyber-Physical Systems Engineering, pp. 129–148. Springer International Publishing, Cham (2019).
6. Barakhnin, V., Kozhemyakina, O., Borzilova, Yu.: The Development of the Information System of the Representation of the Complex Analysis Results for the Poetic Texts. Vestnik NSU, Series: Information Technologies, 17(1), 5–17 (2019).
7. Barakhnin, V., Kozhemyakina, O., Mukhamediev, R., Borzilova, Yu., Yakunin, K.: The design of the structure of the software system for processing text document corpus. Business Informatics, 13(4), 60–72 (2019).
8. Demeyer, S., Ducasse, S., Nierstrasz, O.: Object-Oriented Reengineering Patterns. 1st edn. Square Bracket Associates, Switzerland (2009).
9. Fowler, M., et al.: Refactoring: Improving the Design of Existing Code. 1st edn. Addison-Wesley, United States (1999).
10. Sokolin, D., Timokhovich, A.: Methods for Comprehensive Security of SQL Server Against SQL Injection Attacks. Academy, 3(19), 7–9 (2017). (in Russian)
11. TIOBE Index for May 2021, https://www.tiobe.com/tiobe-index/, last accessed 07 May 2021.
12. Purer, K.: PHP vs. Python vs. Ruby – The web scripting language shootout. In: Vienna University of Technology, Institute of Computer Languages, Compilers and Languages Group, 185.307 Seminar aus Programmiersprachen (2009).
13. Smorodsky, I.: Using the WSGI Standard When Developing Web Application in Python. STUDNET, 3(5), 348–357 (2020).
14. Flask Documentation (1.1.x), https://flask.palletsprojects.com/en/1.1.x/index.html/, last accessed 10 May 2021.
15. Aslam, F., Mohammed, H., Lokhande, P.: Efficient Way Of Web Development Using Python And Flask. International Journal of Advanced Research in Computer Science, 6(2), 54–57 (2015).
16. Barakhnin, V., Kozhemyakina, O., Borzilova, Yu.: The SQL Queries Optimization on the Example for the Search Module of the System for Complex Analysis of Literary Texts. Cloud of Science, 7(4), 749–763 (2020).
17. SQLAlchemy 1.4 Documentation, https://docs.sqlalchemy.org/en/14/, last accessed 10 May 2021.