Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.8, No.3, 2017
www.iiste.org

Development of Web Services for Computational and Analytical Processing Code of Software Metric

Haider Hassan Majeed AlKaraawi
College of Veterinary Medicine, University of Kerbala, Karbala, Iraq

Abstract
This study is to develop a web service, which allows to calculate and conduct analytical processing software metrics. To achieve the goal we need to an analytical overview of the software that allows to calculate code metrics, examine the existing metrics, learn the basic architectural approaches of building a web service, explore the applicability of fuzzy logic algorithms (West et.al., 2015) for the implementation of intelligent analysis software metrics, implement a web service that allows you to calculate and conduct analytical processing software metrics based on expert opinion and make testing design a web service. There are many approaches to solving the problem of testing and verification of software. During the development of this application was used unit testing. The purpose of unit testing - to isolate parts of the program and to show that these parts are operable individually. During the development process have been implemented unit tests, as a result of testing problems have been identified. The operation of the Web service is fully consistent with the previously defined functional requirements. Consequently, the task of determining the effectiveness of the programmer and the source code evaluation solved.
Keywords: fuzzy logic algorithms, SaaS, Object-oriented, cyclomatic complexity, Mamdani algorithm, RESTful, API, SVN-repository.

1. Introduction

Development of software (Hayes et.al., 2016), as well as any branch of production, associated with risks. Sometimes successful at first glance the project for unknown reasons closed. Over the past two decades, many companies used different methods of management of IT projects, some of them suggested the use of evaluation commonly used in material production sectors. Unlike most branches of material production, in matters of software projects create unacceptable simple approaches based on the complexity (Vovk et.al., 2015) of multiplication of the average labor productivity. This is due primarily to the fact that the economic indicators of the project nonlinearly depend on the scope and complexity of the calculation of allowed large error. Therefore, a comprehensive and rather complex technique (Zvezdin et.al., 2008) is used to solve this problem. An important part of these techniques is to assess the software quality (Knyazev et.al., 2007) (Solis et.al., 2016). Software quality is determined by various criteria. In order to be able to assess the extent to which the source code specified criteria, you must use the calculation software metrics. Software metric is a measure that allows to obtain a numerical value of some properties of the software or its specifications (Cohen et.al., 2011). Modern software market includes both paid and free tools to count the software metrics, but all the products display only metric values, but this is not enough to assess the state of the project, to assess the status of the project need to know the qualitative assessment of the resulting value metrics. The absence of applications that can assess on the basis of the calculated metrics, identifies the problem: the software market at the moment there are no means of analytical study of software metrics.

1.1 Practical & Techniques of Study

The practical significance of this work is caused by two factors: 1) In the last few years there was a steady tendency to migrate desktop solutions in the cloud services and application business - SaaS model (Software as a service) as the basic model of the sale and use of the software (Kavis et.al., 2014). At the moment, there is not a single SaaS-service that computes software metrics. 2)None of the existing distributions, realizing counting software metrics, does not allow them to carry out an analytical assessment. Study the applicability of fuzzy logic algorithms (West et.al., 2015) for the evaluation of analytical code metrics; Study the possibility of building metrics violations architectural principles (for example, the pattern Singleton); The implementation of a web service that implements an analytical assessment metrics using fuzzy logic and involving experts for an initial evaluation.

1.2 Review of Existing Solutions

Currently, the software market, which allows to evaluate the quality of the code (Zvezdin et.al., 2009), represented by the following software products. Rational ClearCase - a comprehensive solution for managing software configuration, which provides version control tools, control workspaces, support for parallel development and layout control. One of the functions provided by this package, is a function of building code metrics. Since the solution is based on a version control system, then the calculation of the metrics it is possible

to take into account the change of metrics for a particular file, as well as for the whole project. Visual Studio development environment, starting with the 2008 version, also provides a means of evaluating the Code Metrics PowerTool - a tool allows you to collect and display the code metrics in the project build process. The application calculates the following metrics: The depth of the inheritance; Clutch & The number of lines of code (LOC). The utility can be used as a plug-in as part of Visual Studio, as well as from the command line, in this case, the report will be stored in an xml file. Another tool, presented in the software market, is NDepend - application of static code analysis. NDepend counts a variety of complex metrics (for example, the rate of abstraction). The main advantage of this tool is a user-friendly interface that allows you to visualize the various metrics. The disadvantage is that NDepend requires the user's knowledge of architecture developed software, also used non-standard terminology. The special features of this tool include language CQL queries, which allows you to display only the desired metrics, and the ability to integrate with CI systems. FxCop is a tool that helps developers to follow the corporate software standards. FxCop code analysis to verify compliance with coding standards and naming conventions, and verify that the relevant rules are used in the written program. Net Reflector CodeMetrics Add-Ins - add-on .Net Reflector tool that calculates code metrics for multiple code assemblies and shows how much has changed or that the metric compared to the previous measurement. All results can be stored in a file. JDepend - analogue NDepend program - evaluates the code metrics, written in Java. Among the free solutions for Java JDepend remains the leader. The development of this tool has been discontinued in 2010. CMTJava - the tool supports the quantitative assessment of code metrics. Serenity Plugin - a plugin for CI Hudson, to evaluate code coverage, also has the possibility of constructing various types of metrics (McConnell et.al., 2004) such as complexity, the number of dependencies in the code. Unfortunately, this tool is a plug-in to the Hudson system, which limits the possibility of its expansion, besides the product requires a lot of time spent on provisioning, so is not suitable for solving the selected task. For convenience, add some of these instruments in the table (Table. 1) And carry out their analysis on the following criteria: Supported metrics; License; Costs; Choose the web interface & The supported programming languages.

Table. 1. Comparison analogues

№	Product	Metrics	License	Cost	Web Interface	Program
1.	IBM Rational ClearCase	Dimensionally - oriented metrics, complexity metrics, geometric models, metrics and clarity of style programs, object-oriented metrics (50)	proprietary	2500$	No	Java, C#, C++
2.	Visual Studio Code Metrics PowerTool 10.0	Dimensionally - oriented metrics, complexity metrics (10 metric)	proprietary	Included in the MS Visual Studio (Premium and higher) costs approximately $ 10,000	No	C#
3.	NDepend	Dimensionally - oriented metrics, complexity metrics (total 82 metric)	proprietary	400$	No	C#
4.	Serenity Hudson Plugin	Dimensionally - oriented metrics, complexity metrics Stability metrics (5 metric)	Atlassian Confluence Open Source Project License	free	Yes (using Hudson)	Java
5.	JDepend	Dimensionally - oriented metrics, complexity metrics (10 metric)	BSD License	free	No	Java

As can be seen from Table 1, the tools to compute and visualize metrics have greater value addition, among the products presented the only product that has a web interface, is Serenity Hudson Plugin. The main disadvantage of these systems is the lack of analytical processing elements computed code metrics. Among the studied software systems, only one displays the dynamics of changes in the code metrics compared to the previous measurement, that is not a sufficient level of solving the problem of analytical processing code metrics.

2. Defining the Functional Requirements

Developed service shall consist of the following subsystems: 1)The subsystem works with the repository;

2)Calculate Subsystem code metrics; 3)Analysis of subsystem efficiency. Requirements for the subsystem works with the repository: Data source to use the service must SVN- repository & for the normal operation of the service from the repository we need to receive: Information about the operation which performed; Username; Copy of the source code. The requirements for the calculation subsystem code metrics must calculate the following metrics (Novichkov et.al., 2007): 1) metrics program size: the number of lines of code - a metric of the software used to measure its volume by counting the number of lines of source code text. This index will be used to forecast labor costs for the development of a specific program, or for productivity assessment after the fact, as it is written; the number of classes; the number of blank lines; the number of rows that contain comments; the percentage of comments. The code should be documented. If the ratio of code to the comment is not 1: 4, then we assume that the code is poorly documented (Zvezdin et.al., 2008); the average number of lines for the classes (methods, files). 2)Object-oriented metrics: depth of inheritance tree - the length of the longest path of succession, ending on this module. The deeper the inheritance tree of the module, the harder it is to predict its behavior. On the other hand, increasing the depth provides a greater potential reuse module behavior data determined for the ancestor classes (Moses et.al., 2009). The boundaries of metric values are given by experts; children - number of modules directly inherit this module. Higher values of this metric indicate opportunities reuse; while too large a value may indicate a poorly chosen abstraction. The range of possible values of the metrics are given by experts; number of methods - a large number of methods to a violation of SOLID principles (Cohen et.al., 2011); the number of fields - a large number of fields also tells about violation SOLID principles; the number of implemented interfaces; connectivity of objects - the number of modules associated with the module as a customer or supplier. Excessive connection indicates weakness module encapsulation and can hinder code reuse; resistance - the distance from the main sequence. This index is an indicator of the balance between abstraction and build stability; percentage abstract classes - a large number of abstract classes complicates the perception of the system. 3)The complexity metrics: cyclomatic complexity - shows the structural complexity of the code, ie, the number of different branches to the code. The higher the score, the more tests should be written for the complete code coverage; method of weight class - represents a relative measure of the complexity of the class based on its cyclomatic complexity of each method (Geoffrey et.al., 2012). Class with more sophisticated techniques and a lot of techniques considered to be more difficult; lack of adhesion techniques - shows the degree of dependence of classes with each other. Good software design involves a small number of related classes. The more, the more difficult in the future reuse this class, as well as support. 4)Breach of the metrics architectural principles: the number of abusive pattern Singleton - even isolated cases of improper use of this pattern leads to problems in the product, which is very difficult to detect. To determine the correctness of the proposed use of annotations in the code. effectiveness analysis subsystem must provide: The possibility of the initial expert evaluation of projects; Data Mining based on Mamdani algorithm (Mohammed et.al., 2014) (Iancu et.al., 2012); Evaluation function programmer effectiveness on the basis of analytical processing of data mining.

2. Description Of The REST Architectural Style

For greater flexibility, the developed service will use the REST architectural style. REST(Representational State Transfer) - a style of software architecture for distributed systems such as the World Wide Web, often used for building web services. The term REST was introduced in 2000 by Roy Fielding (Richardson et.al., 2013). Systems that support the REST, RESTful-called systems. REST is a simple data management interface without the use of additional internal layers. Each piece of information is uniquely determined by a global identifier, such as a URL. Each URL in turn has a strictly specified format. Advantages of the approach: Extensibility applications; The common interfaces; The independence of the components; Components - allow intermediaries to ensure the security and encapsulation of legacy systems. When designing RESTful web services is important to remember CRUD (Create, Read, Update and Delete) methodology (Richardson et.al., 2013). In REST generally accepted the following standards: GET - read (read, get); POST- create (create); PUT - update (change, update, or create if does not exist); DELETE - delete (delete). It should be noted that the PUT and DELETE methods are difficult to implemented. Therefore, in practice, these methods are used by "overloading» POST method.

Typical architecture RESTful-application is shown in Figure 1.

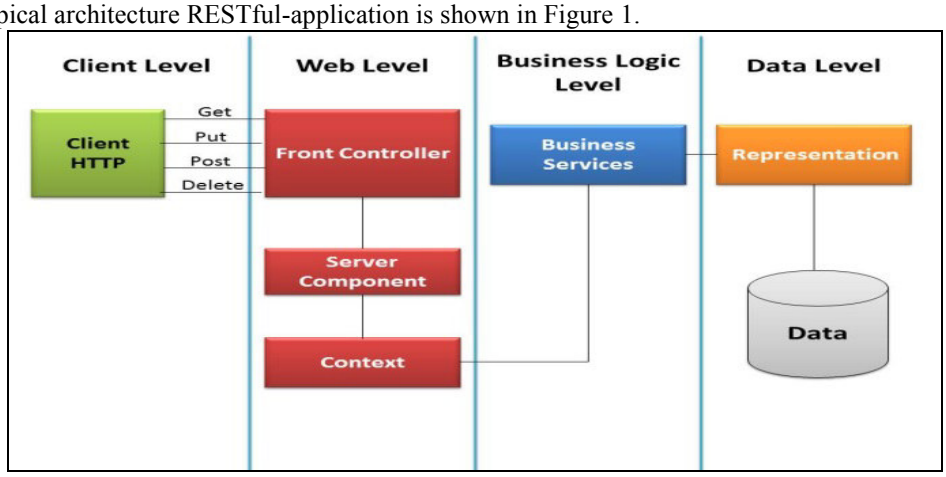

Fig. 1. Architecture RESTful application

Development of a web service involves the creation of components, manufacturing HTTP- requests. Through this approach, we are able to create a separate subsystems, each of which can be easily changed, the approach is well suited for the implementation of SaaS-platforms. To create and access to the components responsible for processing incoming requests from user agents, as well as sending responses, perform the following steps (Richardson et.al., 2013): Create a server connector and add to it the appropriate HTTP server component; Add or existing HTTP receive context corresponding to a given context server; To transfer an incoming request from the user agent to process the corresponding business service; Create a model for resource representation, generated business service, put into it and the data presented in the form of a representative state of the resource; Transfer the representative status of the resource back to the user agent. As a result, it becomes possible to realize various types of interfaces, such as a mobile interface, Web interface or desktop interface. In addition there is a possibility of an open API (Application Programming Interface) (Jacobson et.al., 2011) for third party developers.

3. Use Case Diagram

On the basis of the functional requirements will create a chart of options for using web service. The first step is to define the roles of all users of the system. The following user roles have been identified:

User - has the ability to work with information about the repository, to obtain information about the metrics and the effectiveness of the employees;

Expert - has the same capabilities as the user, but at the stage of learning has the ability to estimate the values of metrics values set boundaries. Description of use cases:

1) Repositories management includes functions: Add a new repository; Update information about the repository; Obtaining information about the repository; Repository removal.

2) Obtain information on the metrics of the project includes features: Calculation of project metrics; Determining the list of supported metrics for the project; Delete information of the old metrics.

3) Obtaining information on the effectiveness of: Intelligent analysis of metrics; Efficiency calculating metrics based on the analysis;

4) Evaluation of the code metrics include: Defining limit values for each metric for the test project.

Use Case diagram is shown in Figure 2.

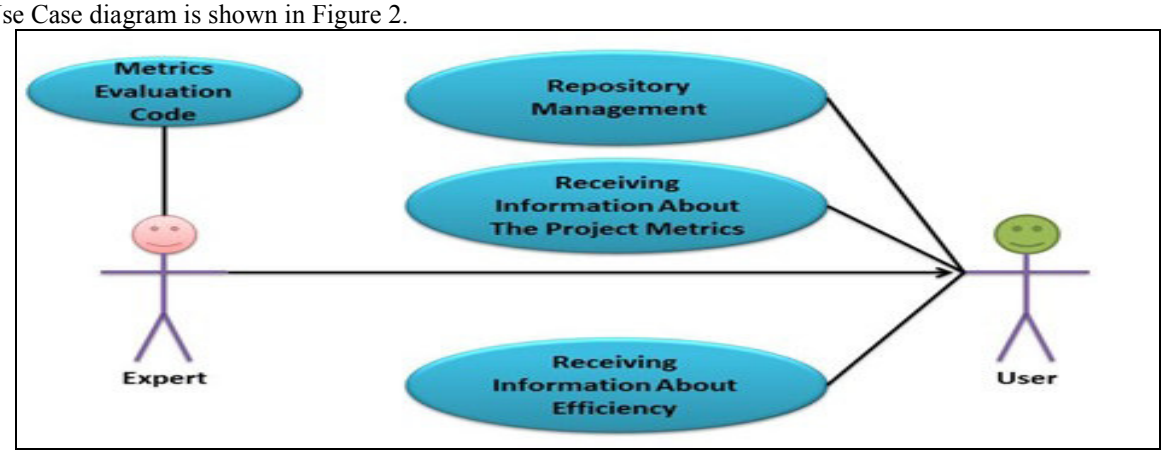

Fig. 2. Use Case Diagram

4. List of Information Entities

The repository stores information about the path to the repository in the URI (Uniform Resource Identifier) format, as well as the name of the repository that will be displayed in the system. The repository contains information about all the changes. **Information about changes (commit)** - stores information about all classes that affected the change, with the change date. **Author code** – contains information about the name of the author of the code. **Class** - the class name and stores information about the calculated metrics. **The calculated value** - contains information about the value of the calculated metrics for a given class. **Metric** - stores the metrics name and boundary values. **Expert evaluation** - contains the limit values for each metric.

5. Interface Design

Based on the diagram of use cases described in Fig. 2., will determine which screens should be available to the user interface design and run a Web service. interface should be designed for the expert system that meets the following requirements: Display of all project metrics; Display of brief information about each type of metrics; The ability to assign a rating for each metric. Initially, the service calculates the metric for the demonstration project, the expert can then put one of the three possible changes (good, excellent and bad) for the current value of the metric. For each metric expert can put its weight, thus showing the importance of this metric. In standard mode, the interface will: A list of all tracked repositories; Provide the ability to add a new repository. For each SVN-repository must be able to review the performance of each programmer with information about the latest commit (Küng et.al., 2016). For this interface has been developed.

6. Results of Implementation of the Design

The result of the implementation of the designed system is a web service that meets the requirements: 1) The service should be implemented in accordance with the requirements of the REST architectural. 2)Shall consist of three subsystems: Subsystem works with the repository; Subsystem metrics calculation; Subsystem of calculation of efficiency. 3)The counting of the subsystem should handle categories: Size metrics programs; Object-oriented metrics; Complexity metrics; Metric violations architectural principles. 4)Subsystem calculation of efficiency should include the following functions: Expert system functions; Analytical processing function metrics.

7. UML Class Diagram of the Expert System

An expert system is used only in a learning mode. Only an expert can work with it. Expert system allows you to: To receive the calculated metrics on the project; Enter the weight and possible range of values for all metrics. Given the above requirements, we construct a class diagram of the expert system Figure. 3.

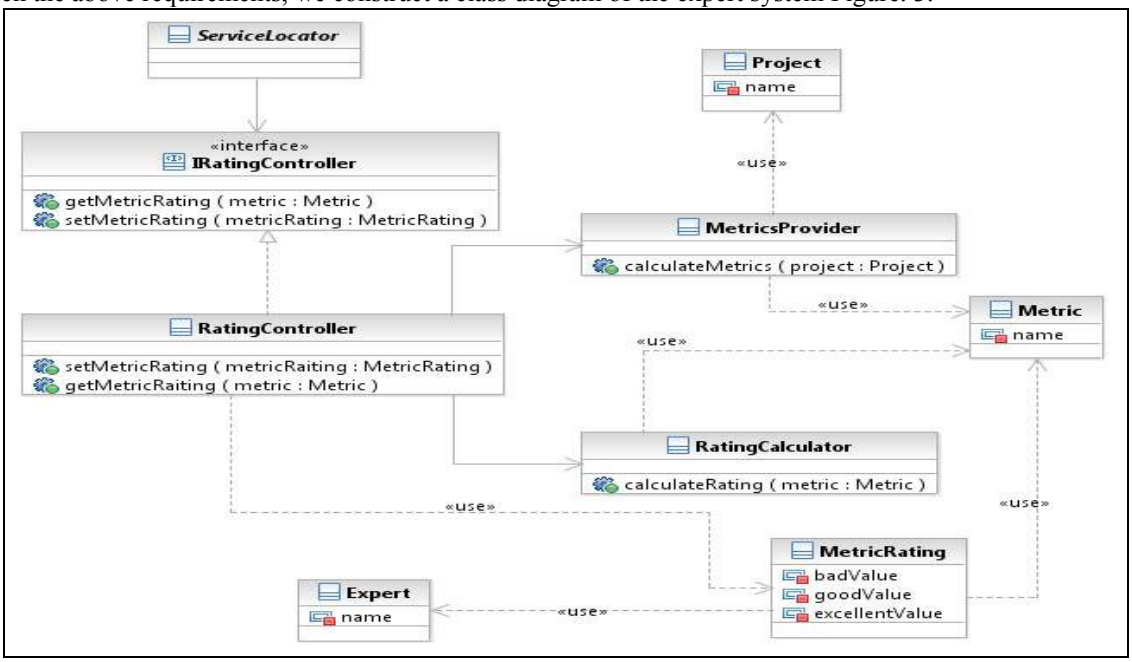

Fig. 3. The class diagram of the expert system

Description of classes expert system: RatingController - provides the ability to install third-party agents or request a new rating of the rating for a particular metric. getMetricRating - method takes as a parameter the metric description, returns a list calculated on the basis of expert estimations of boundary values for this metric. setMetricRaiting - method makes information about the expert evaluation metric for the metric in the database.

MetricProvider - the request to the subsystem calculate metrics, and processes the response. calculateMetrics method - receives as the parameter information of a reference example returns an array of calculated metrics. RatingCalculator - is responsible for the calculation of limit values of each metric. calculateRating - method as parameter information received metric is an agent which calculates the range of possible values for this metric, and returns the result as a set of three numbers (bad, normal and good metric values). IRaitingController and ServiceLocator - helper classes that are essential for the interaction with external agents. Project, Metric, Expert, MetricRating - basic information entities serving for communication between services. Class MetricProvider can call up a method in a class calulateMetric SingletonMetric, which, in turn, has delegated this operation SingletonInspection class. Class SingletonInspection build SingletonInspectionVisitor class that will pass all the classes and check the violation of the rules Figure. 4.

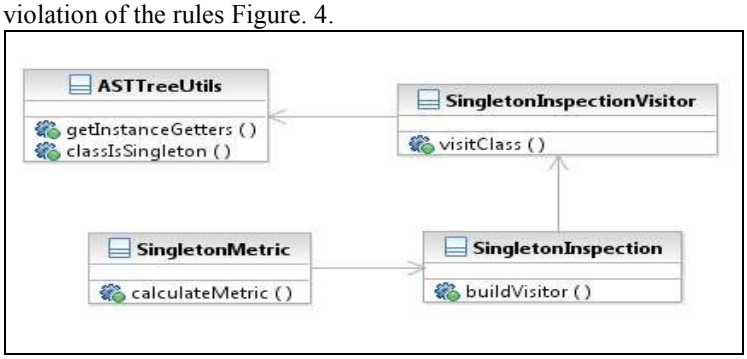

Fig. 4. Subsystems counting metric violation Singleton pattern

On the basis of UML-diagrams will build API, provided by an expert system. REST API expert system contains methods: **POST / expert / metric_rating** - allows you to add information about the peer review. The parameter adopted JSON- array with this structure:

```
{ "MetricId": "metric identifier"
  "ExpertId": "expert identifier"
  "BadValue": "Lower limit"
  "GoodValue": "Normal state"
  "ExcellentValue": "Upper limit" }
```

GET / expert / metric_rating – provides information on the ranges of metric values, structure JSON- array:

```
{  "MetricId": "metric identifier"
  "BadValue": "Lower limit"
  "GoodValue": "Normal state"
  "ExcellentValue": "Upper limit" }
```

This API is sufficient to complete the work of the expert system. Two variants of exceptions provided in the system: 1)There is no access to the SVN-repository. As a result, after three unsuccessful attempts to restore the connection will display an error message and stop repository monitored. 2)No information on expert assessments. In this case, the service cannot continue normal operation, a message will be displayed.

8. Implementation of Fuzzy Logic

In the role of the algorithmic component of this project is the algorithm of Mamdani. This algorithm is based on fuzzy logic conclusion (Vovk et.al., 2008), allow to avoid an excessive amount of computation, and has been appreciated by experts. This algorithm is currently achieved the most practical application of fuzzy modeling tasks. In the description of the algorithm the following definitions shall apply: Fuzzy variable - a tuple <α, X, A>, where: α - the name of the fuzzy variable; X - its scope; A - fuzzy set on the universe X. A linguistic variable is a tuple <β, T, X, G, M>, where: β - the name of the linguistic variable; T - set of its values (terms); X – universe fuzzy variables; G - syntactic procedure of formation of new terms; M - semantic procedure, forming fuzzy sets for each term of the linguistic variable. Fuzzy statement will be called the statement of the form «β IS α», where: β - linguistic variable; α - one of the terms of this variable. Rule fuzzy productions (hereinafter just the Regulation) will be called the classical rule of the form "if ... then ...", where the fuzzy statements will be used as the conditions and conclusions. Such rules are recorded as follows: IF (β1 IS α1) AND (β2 IS α2) THEN (β3 IS α3). Besides «AND» are also used logical connective «OR». But this entry is not used in this work, such rules are divided into several simpler (no «OR»). Sub-conditions - each of the fuzzy statements in the condition of any rule. Similarly, each of the statements in custody called subcontracted. The rule base - a set of rules, where each subcontract mapped to a specific weighting factor. For the implementation of the algorithm used by the object-

oriented approach (Dennis et.al., 2015). The source code written in the Java programming language. The chart shows the most significant connections and relationships between classes involved in the algorithm Fig. 5.

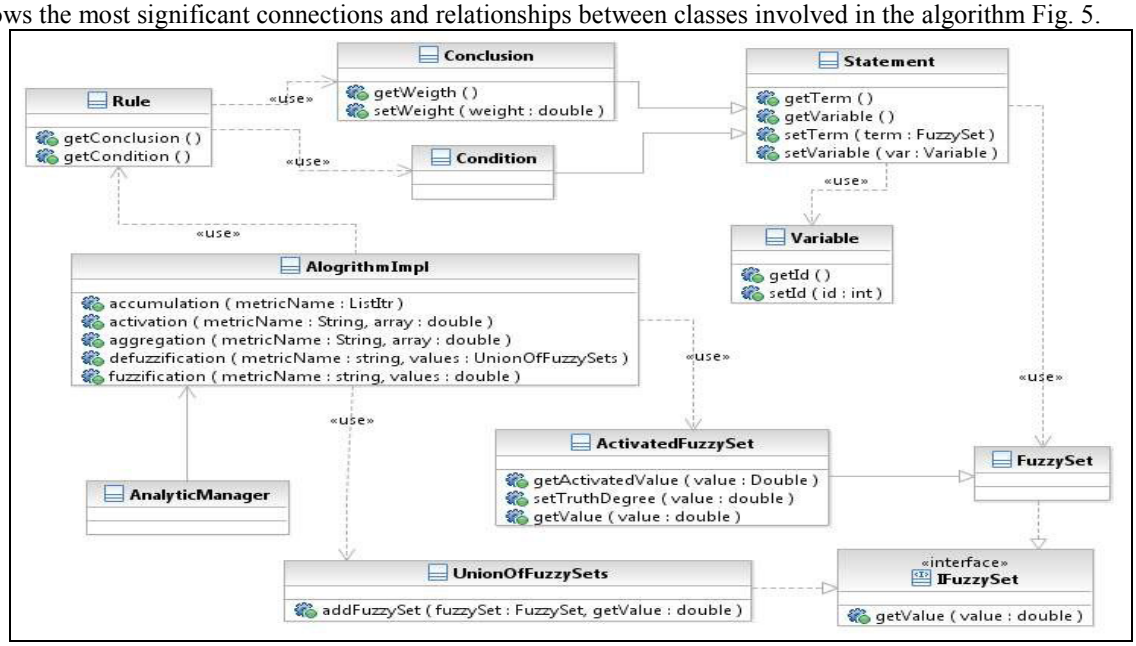

Fig. 5. The class diagram.

Rules (Rule) consist of the following conditions (Condition) and conclusions (Conclusion), which in turn are vague statements (Statement). Fuzzy statement includes linguistic variable (Variable) and a term which is represented by a fuzzy set (FuzzySet). On the fuzzy set membership function is defined, the value of which can be accessed using getValue () method. When the "activated" algorithm will need to use the fuzzy set (ActivatedFuzzySet), which in some way overrides function of fuzzy sets (FuzzySet). Also in the algorithm used by the union of fuzzy sets (UnionOfFuzzySets). The algorithm includes the steps and uses all the rule base (List <Rule>) as input. Also, the algorithm involves the use of "activated" fuzzy sets (ActivatedFuzzySet) and their associations (UnionOfFuzzySets). Defuzzification aim to obtain a quantitative value for each of the linguistic output variables. In this embodiment, the algorithm used by the center of gravity method, in which the output of the variable i-th value calculated by the formula:

$$y_i = \frac{\int_{Min}^{Max} x \cdot \mu_i(x)\,dx}{\int_{Min}^{Max} \mu_i(x)\,dx} \qquad (4)$$

Where: μi (x)- a membership function corresponding to a fuzzy set Ei; Min and Max - the boundaries of the universe fuzzy variables; yi - the result of defuzzification. This results in a group of numbers on which to draw conclusions about the status of the project, in particular, evaluation of the effectiveness of the programmer.

9. Testing The Web Service

There are many approaches to solving the problem of testing and verification of software. During the development of this application was used unit testing. The purpose of unit testing - to isolate parts of the program and to show that these parts are operable individually. During the development process have been implemented unit tests, providing approximately 80% coverage of test business logic code. Integration testing the core functionality was performed manually based on the state diagram. Initially the authorization window appears, If the user authentication has been established that it applies to the experts, the expert system interface Figure 6 will be opened.

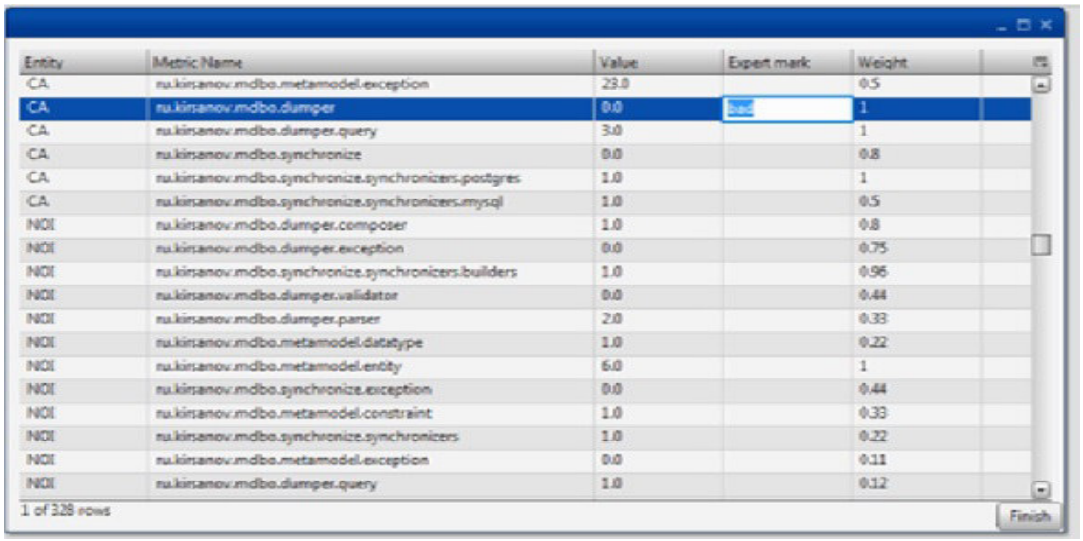

Fig. 6. The expert system interface

After at least one expert put down all of the estimates, the service starts to operate. In the process, there may be situations that the repository is unavailable, in which case the service marks it as such. In Figure 7, the service cannot access the repository Diploma project.

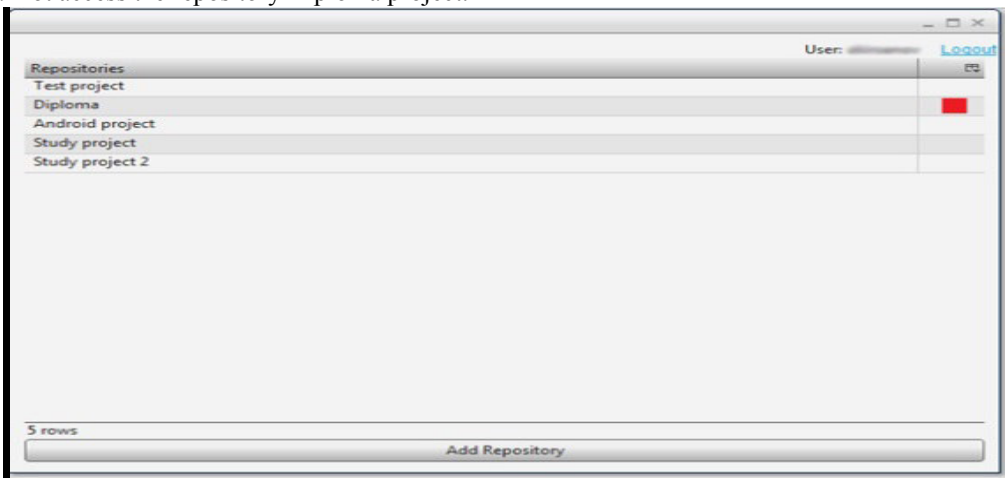

Fig. 7. Standard User Interface

When you click on the repository opens a page with information on the effectiveness of the programmer. According to the test scenario testuser2 programmer flooded low-quality code, consequently, its effectiveness should be reduced. Can be seen from the figure 8 that showed a decline in service efficiency.

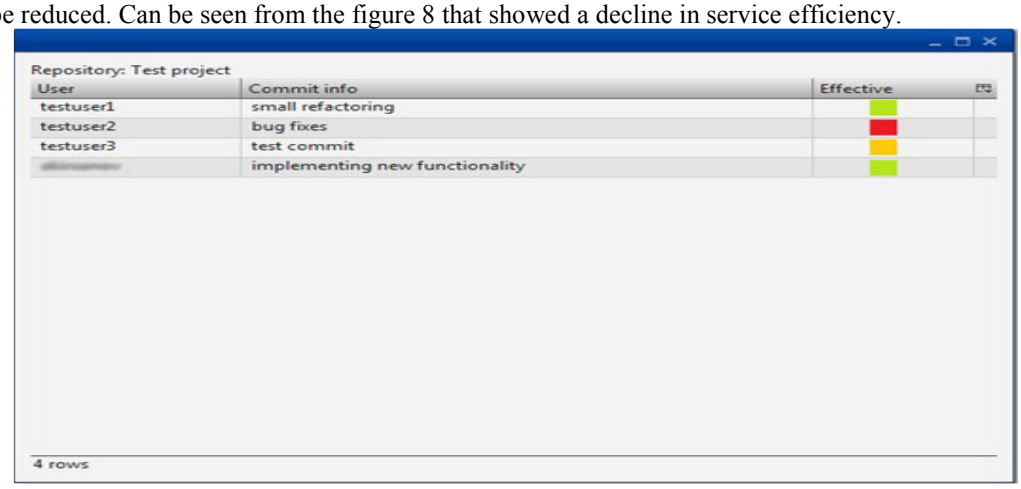

Fig. 8. Viewing information about the effectiveness of the programmer.

10. Result of Testing

As a result of testing problems have been identified. The operation of the Web service is fully consistent with the previously defined functional requirements. Consequently, the task of determining the effectiveness of the programmer and the source code evaluation solved.

11. Conclusion

The complexity of developed applications is growing every day, thus increasing the cost of the project. In such circumstances, the problem of analyzing code quality and efficiency of programmer's work is particularly acute. Currently existing tools do not provide an easy and convenient opportunity to assess the status of the project.

As a result of this work was developed SaaS-service that provides the ability to evaluate the effectiveness of the developer, and generally produce quality code analysis. Service carries out counting the different categories of metrics such as complexity metrics, quantitative metrics, introduced the metric violations of architectural principles. Specially designed application are:

1. Use of expert system at the stage of the system of education;
2. The use of fuzzy logic algorithms for analytical processing code metrics.

As a further development opportunities developed systems should be noted:

1. The use of neural networks in conjunction with fuzzy logic algorithms for more accurate analysis code metrics. Application of neural network algorithms will allow to realize the property of self-learning software system without the involvement of external experts;
2. The increase in the number of supported metrics, including enhanced support for metrics violations of architectural principles. The calculations are a number of metrics requires a significant amount of time and effort for implementation, and they are not critical to the task of evaluating the effectiveness of the programmer. Among the metrics violations architectural principles can be implemented SOLID violation metrics;
3. The introduction to the functionality of the service options for storing the history of evaluating the effectiveness of the programmer and determine its dynamics.

References

[1] West F., (2015). "Fuzzy Inference System".
[2] Hayes H., (2016). "Agile Project Management: The Ultimate Guide To Agile Project Management And Software Development".
[3] Vovk V., Papadopoulos H., (2015), "Measures of Complexity: Festschrift for Alexey Chervonenkis".
[4] Zvezdin S.V., (2008). "Formation of metrics software system code: Scientific and technical statements.
[5] Knyazev E.G., Shopyrin D.G., (2007). "Analysis of code changes in the method of clustering metrics".
[6] Solis T., (2016). "Quality Assurance: Software Quality Assurance Made Easy".
[7] Cohen M. B., Cinneide M. O., (2011). "Search Based Software Engineering: Third International Symposium", (2011th ed.).
[8] Kavis M., (2014). "Architecting the Cloud: Design Decisions for Cloud Computing Service Models (SaaS, PaaS, and IaaS)".
[9] Zvezdin S.V., (2009). "Metrics as a quality management tool: Open systems", - Chelyabinsk: SUSU, Vol. 8. S. 51- 54.
[10] McConnell S., (2004). "Code Complete: A Practical Handbook of Software Construction", (2nd ed.).
[11] Novichkov A., (2007). "Code metrics and the practical realization of their collection and analysis".
[12] Moses M., Karpenko A., (2009). "Software reliability metrics calculation based on static analysis", Scientific and technical sheets of St. Petersburg State Polytechnic University. - St. Petersburg: STU,. Vol. 80. S.139- 147.
[13] Geoffrey K., (2012). "Cyclomatic Complexity Metrics Revisited: An Empirical Study Of Software Development And Maintenance".
[14] Mohammed J., Wagner M. (2014). "Data Mining and Analysis: Fundamental Concepts and Algorithms".
[15] Iancu I., (2012). "A Mamdani Type Fuzzy Logic Controller".
[16] Richardson L., Amundsen M., (2013). "RESTful Web APIs: Services for a Changing World".
[17] Vovk O., (2008). "Study of fuzzy inference algorithms of Difficulty in facility management systems" Research in information technology. - St. Petersburg: STU, Vol. 30. S. 83- 88.
[18] Jacobson D., Brail G., (2011). "APIs: A Strategy Guide: Creating Channels with Application Programming Interfaces".
[19] Küng S., Onken L., Large S., (2016). "TortoiseSVN a Subversion client for Windows: Chapter 3. The Repository ".
[20] Dennis A., Wixom B., (2015). "Systems Analysis and Design: An Object-Oriented Approach with UML".