Code review

I. Introduction

Code review is systematic examination of computer source code, sometimes referred to as peer review. It is intended to find mistakes overlooked in the initial development phase, improvifng the overall quality of software. Automated code reviewing software lessens the task of reviewing large chunks of code on the developer by systematically checking source code for known vulnerabilities. A 2012 study by VDC Research reports that 17.6% of the embedded software engineers surveyed currently use automated tools for peer code review and 23.7% expect to use them within 2 years.

After CR are achieved following objectives:

- improving the quality of the code
- style errors are detected
- increases the degree of co-ownership of the code
- localized errors at an early development stage

Many companies are using the CR, as the young specialists learning tool, which will appear in the required skills; in the process there is knowledge exchange between the more experienced and less experienced colleagues. There are several ways of the CR, for example, the direct personal contact with the developer of the code. However, this method does not allow for the CR, if for some reason the developer and the inspector are at a great distance from one another.

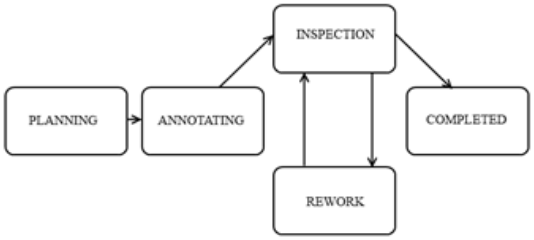
Therefore, the most convenient variant considered to be using special programs that allow for Code review to remotely. The important aspect is the integration of these tools with the environments in which the development of software is carried out.

II. METHODS, RESULTS, DISCUSSION

On the market of software in recent times create a large the number of solutions for CR. Basically, all the solutions have the server part that is installed at the office of, as well as the client applications, located directly on users' computers. Many solutions support to view and reviewing code a web interface, some decisions are based entirely on the web interface.

Collaborator project supports the expansion of facilities for the Visual Studio, and the IDE Eclipse, and a web-based interface. The program odeollaborator implemented support the majority popular revision control systems.

By default, all added and changed files in the local working copy will be added to the inspection. This tool supports the convenient role model that is when creating inspection code, the developer himself chooses, who will conduct the inspection, and who can monitor this.



The life cycle of a code inspection

The illustration shows the standard life cycle of the CR in odeollaborator environment. This life-cycle phase divided into the following:

- Planning - the creation of the inspection and selection with a program code files
- Annotation - selection an inspector and observers add, and also possible to add a comment to the inspection
- Inspection - inspection of program code
- Rework - the correction of errors identified by the at the stage Inspection
- Completed - the completion of the inspection

The peculiarity of this the life cycle is that the inspection is not completed, until all the defects will be corrected, i.e. transitions between the Inspection and Rework can be an unlimited amount.

AgileReview - a popular extension of the Eclipse development environment, enabling you to create and carry out an inspection within the Eclipse development environment. From the obvious merits of this solution can be identified Refactoring stability, which allows you to edit the code, without losing touch with the code inspections.

Upsource - is a tool to review of the source code and navigate through code repositories. Upsource supports version control system Git, Mercurial, Subversion, and Perforce.

The Java-supported projects Upsource further code analysis similar to how it is presented in IntelliJ IDEA environment. For the second version there is a plugin for IntelliJ IDE development. This leaves many other environments, for which there are no corresponding expansion. Of the key features can be identified discussing and reviewing the code.

Based on the of the study has been formulated a set of key requirements to develop tools for enhancing the development environment:

- Use of a role model for the division of responsibilities in the program code and the creator of the reviewer;

- binding information about the defect is not a line number, and directly to the program code;
- Creation of a database of common programming errors in the language Java;
- add the ability to offer a turnkey solution fixes the defect;
- Create a server application that stores the inspection and information about them;
- Using a the life cycle illustrated in Figure 1

Since the code is can be changed during operation, it was created binding defect information is not to the number of lines of code, but directly to the substantive elements of the source code. To reduce the time for the code inspection IBExpert a database, this stores the most common programming errors in Java.

Very often there is a situation where, instead of describing the defect developer offers an alternative solution to a specific area program code. The relevant script is supported in my application. In the discussed solutions support the following roles:

- Developer - creates the code, sends it to the inspection, then gets to inspect the code, corrects defects and sends an inspector until you have corrected all defects;
- Inspector - receives code written by the developer finds the defects and comments on them, and then sends the code to inspect the developer for revision, after correcting the defects of the developer, the inspector again receives the code to verify the correction of all defects. In the case where one or more defects were not corrected, or by repeated viewing new defects inspector can send code developer was found on completion.
- Observer - a person who can see all the stages of the inspection, to view the information on all the identified defects.

The server component interacts with the database, and provides storage of inspection results. The following are possibilities the server side:

-Storage Error information;

-Getting file for inspection;

-Vydacha file for inspection;

-Storage Information about users.

Interaction with the client part of the program carried out by the developed for this network protocol.

III. Conclusion

As a result of the project the following tasks were solved:

- Developed a means of extending the NetBeans IDE that supports the creation and conduct of an inspection of the code, as well as monitoring the process.
- A network protocol to communicate with the means of expanding the NetBeans IDE.
- Developed a database for storing information on inspections and defects.

Realized means of extending the NetBeans IDE supports a custom script code inspection organization and implements basic operations, such as sending a code inspection, an inspection, review inspection results, sending the corrected code, and completion of the inspection.

At the moment, it was not possible to obtain adequate information about how to use tools designed to speed up test code inspection. Conducting appropriate research is a separate task, which forms the possibility of further work on the project.