SeeCode – A Code Review Plug-in for Eclipse

Moran Shochat, Orna Raz, Eitan Farchi

IBM Labs in Haifa

Outline

- Introduction
- SeeCode main features:
  - Code Review Perspective
  - Distributed Review
  - Review Comments
  - Statistics View
- Deployment
- Planned features

Introduction - Code Review

- Code Review – considered as the most effective technique for improving software quality
- Classical inspection (Michael Fagan):
  - All artifacts must be inspected
  - Meeting must be rescheduled if the participants are not well-prepared
- Selective Homeworkless Review (Eitan Farchi and Shmuel Ur):
  - The artifacts for review are selected according to quality concerns and cost-effectiveness
  - Use review techniques with little or no preparation

What do we require from an effective code review tool?

- Based on our experience with development teams across IBM, an effective review tool should:
  1. Enable the review of source files and provide a mechanism for handling and storing review comments
  2. Support the review meeting process and roles, possibly for a distributed team
  3. Provide syntax highlighting and easy code navigation (e.g., jumping from function usage to its definition)
  4. Maintain the association between comments and source lines, even when the code is changed
  5. Provide ongoing feedback on the effectiveness of the review process
  6. Allow version control of the review comments along with the source files
- Other tools exist, but each of them fulfills only part of these requirements
- SeeCode was designed to fulfill these requirements as well as to allow various review methodologies

SeeCode is not alone! What other code review tools exist?

- Any Integrated Development Environment (IDE)
- Jupiter - Eclipse plug-in (http://csdl.ics.hawaii.edu/Plone/research/jupiter)
  - Syntax highlighting, code navigation and association of markers to source lines
  - No review roles or team distribution support
- Codestriker - collaborative code reviewer (http://codestriker.sourceforge.net/)
  - distributed review environment
  - No substantial code navigation support
  - No association of review comments with the underlying code when the code changes
  - No review roles or review process support
- SourcePublisher – (http://www.scitools.com/products/sourcepublisher/)
  - Code printouts with syntax highlighting and links to other relevant parts
  - No comment handling mechanism
  - No review roles or review process support

Code Review Perspective

- SeeCode extends the Eclipse Integrated Development Environment (IDE) by adding a ‘Code Review’ perspective to the workbench
- The code review is performed in the same IDE, like any other programming task
- Syntax highlighting and code navigation - **there is such a thing as a free lunch**

Demo…

Distributed Review

- Three review roles:
  - Code Owner
  - Scribe
  - Reviewer
- Each SeeCode user chooses one of the above roles
- The Scribe and the Reviewers connect to the Owner over the network
- The Owner distributes the files for review, the review comments and the code navigation actions

Demo…

- Distributed review modes:
  - Team meeting – either face-to-face or virtual
  - Asynchronous review
    - The asynchronous review can be used as a preparation stage for the team meeting
- SeeCode supports the above roles and modes but does not enforce them!

Review Comments

- SeeCode uses Eclipse markers to store the review comments:
  - Marked by a SeeCode icon in the left ruler
  - persistent association of comments to source lines even when the source is changed by the author
- Two kinds of comment:
  - Local
  - Global
- Comment actions:
  - Add
  - Edit
  - Close
  - Delete

Demo…

Review Comments Storage

- Review comments are saved in an XML file
  - XML file for each project in the workspace
- Using any version control system, the comments file can be controlled together with the project source files
- Selective export of comments into text or XML file
- Selective import of comments from previously exported XML file
  - Reviewing of code without connecting to the distributed review

Quantitative Feedback – Statistics View

- Reports the effectiveness of the ongoing review effort
- Compares the updated number of issues for each file to the expected number of issues
- Expected number of issues:
  - Based on our past experience with several IBM teams
  - Average review rate of about 100 lines per hour
  - Average issue recording rate of 2-3 per person hour
- The quantitative feedback can be utilized by the review moderator to keep the review process on track

Demo…

Deployment of SeeCode

- SeeCode has been piloted by several IBM groups with good feedback
- User feedback:
  - Using the tool is intuitive and easy to learn (for developers who are used to modern IDEs)
  - Distributed review is very useful for teams that are spread over different locations and time zones
  - Syntax highlighting and code navigation features of Eclipse help the reviewers in understanding the code, making the review more efficient
  - Performing the review inside the IDE saves time and extra effort (e.g., no need to send files for review and no need to document the comments using a separate tool)
- SeeCode was designed to support the “selective homeworkless review”, but it does not enforce it – some teams use SeeCode differently
- Less useful for teams developing in a programming language that has no Eclipse plug-in

Planned Features

- Focus the review according to external data:
  - File change history
  - Code coverage report
  - Static analysis report
  - …

  Demo…
- Focus the review on a specific programming concern:
  - Performance
  - Reliability
  - Concurrency
  - …
- Focus the review on code changes by performing the review in Compare view

Planned Features – continued

- Quantitative feedback:
  - Ongoing issue detection rate
  - Review Quality Record

Acknowledgments

- SeeCode development:
  - Sergey Novikov
  - Nadav Steindler

Thank You!