УДК 004.415.2

ПРИМЕНЕНИЕ ИНСТРУМЕНТА КОНТРОЛЯ КАЧЕСТВА КОДА SONARQUBE В КРУПНЫХ ПРОЕКТАХ

Ефимов А.А.

магистрант,

Уральский государственный экономический университет,

Екатеринбург, Россия

Аннотация

Зачастую перед крупными IT-компаниями, занимающимися разработкой программного обеспечения, возникает проблема производительности изготавливаемого продукта. Поскольку большинство специалистов, принимающих непосредственное участие в разработке начальной версии продукта, изначально имели менее конкретную задачу и набор требований к продукту и ввиду этого могли допускать набор ошибок касательно архитектуры проекта – возникают ошибки, которые существуют в проекте чрезвычайно длительный период времени и о существовании которых могут и не догадываться новые разработчики, что были подключены к разработке программного продукта в период поддержки и доработки уже готового решения. Как же в таком случае производится поиск уязвимостей проекта? В статье рассмотрены основные концепции и возможности инструмента контроля качества кода SonarQube. Изучен интерфейс применяемого инструмента и набор задач, которые можно решить с его помощью.

Ключевые слова: SonarQube, анализ кода, аналитика, оптимизация, производительность, чистый код.

USING SONARQUBE SOURCE CODE QUALITY CONTROL TOOL IN LARGE PROJECTS

Efimov A.A.
master student,
Ural State University of Economics
Yekaterinburg, Russia

Annotation

Often, large IT companies involved in software development have a problem with the productivity of the manufactured product. Since the majority of specialists directly involved in the development of the initial version of the product initially had a less specific task and a set of product requirements, and therefore could have made a set of errors regarding the project architecture, there are errors that exist in the project for an extremely long period of time and which may exist and do not guess the new developers that were connected to the development of the software product during the period of support and refinement of the ready-made solution. How, then, is a project vulnerability search performed? The article discusses the basic concepts and capabilities of the SonarQube code quality control tool. The interface of the used tool and a set of tasks that can be solved with its help are studied.

Key words: SonarQube, code analysis, analytics, optimization, performance, clean code

SonarQube является продуктом компании SonarSource, основной офис которой расположен на территории Швейцарии. Её исходный код написан на языке Java и регулярно самоанализируется данным продуктом, что может

свидетельствовать об отсутствии каких-либо критических ошибок и серьёзных уязвимостей в продукте.

С технической стороны SonarQube представляет собой платформу непрерывной проверки исходного кода, основными задачами которой являются:

- Хранение информации о проектах
- Обработка вносимых изменений (pull-request) в анализируемые проекты
- Объединение полученной по результатам анализа информации в единую структуру
- Предоставление общей информации, замечаний и технического долга по проекту - конечному пользователю в удобном для него виде

SonarQube поддерживает широкий набор анализируемых языков программирования, среди которых присутствуют:

- ABAP
- C/C++
- C#
- COBOL
- CSS/Less/SCSS
- Cucumber Gherkin
- Erlang
- Flex/ActionScript
- Groovy
- Java + Java Properties
- JavaScript
- JSON

— Lua

— Objective-C

— PHP

— PL/I

— PL/SQL

— Puppet

— Python

— RPG

— Swift

— VB.NET

— Visual Basic

— Web

— XML

Будучи платформой, SonarQube предоставляет обширный функционал для написания плагинов: абстрактную модель, в рамках которой реализуются проверки. Иными словами, абсолютно любой человек может реализовать свой язык программирования, а затем написать плагин на базе SonarQube для его поддержки.

Помимо возможности серверного развертывания, SonarQube может быть применен локально на виртуальной машине для проверки указанных локально расположенных каталогов с исходным кодом.

Для наиболее раннего предотвращения возможных проблем компания SonarSource выпустило новый продукт SonarLint, представляющий собой расширение для сред разработки. Его главными особенностями являются возможность применения используемых в развёрнутой версии SonarQube правил и уведомления пользователя о их нарушении непосредственно в среде разработки. В случае, если продукт SonarQube не был развёрнут и SonarLint не удаётся получить набор применяемых правил, применяется набор

стандартных правил для проведения общего анализа исходного кода. Данный режим применяется для любого из поддерживаемых языков программирования.

На текущий момент расширение SonarLint поддерживает следующие среды разработки:

— Eclipse

— IntelliJ IDEA

— Visual Studio

— VS Code

Главной страницей SonarQube является список проектов, добавленных в систему. На ней отображается краткая характеристика по каждому анализируемому проекту: количество строк кода, версия сборки, количество уязвимостей, багов и мест применения неоптимальных решений, а также дата последнего анализа.

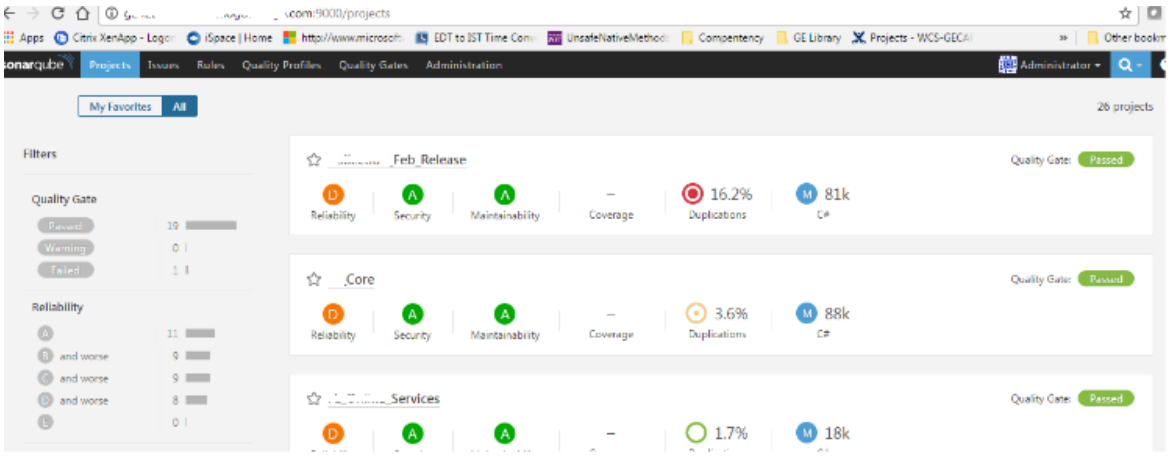

Рис. 1 - Главная страница[1]

Возможность настройки отображаемого содержимого главной страницы при помощи набора встроенных виджетов позволяет представлять

[1] Составлено автором

состояние кода анализируемых проектов в максимально удобном для пользователя виде.

С целью приобретения более детальной информации о результатах последнего анализа проекта можно осуществить переход на страницу метрик проекта.

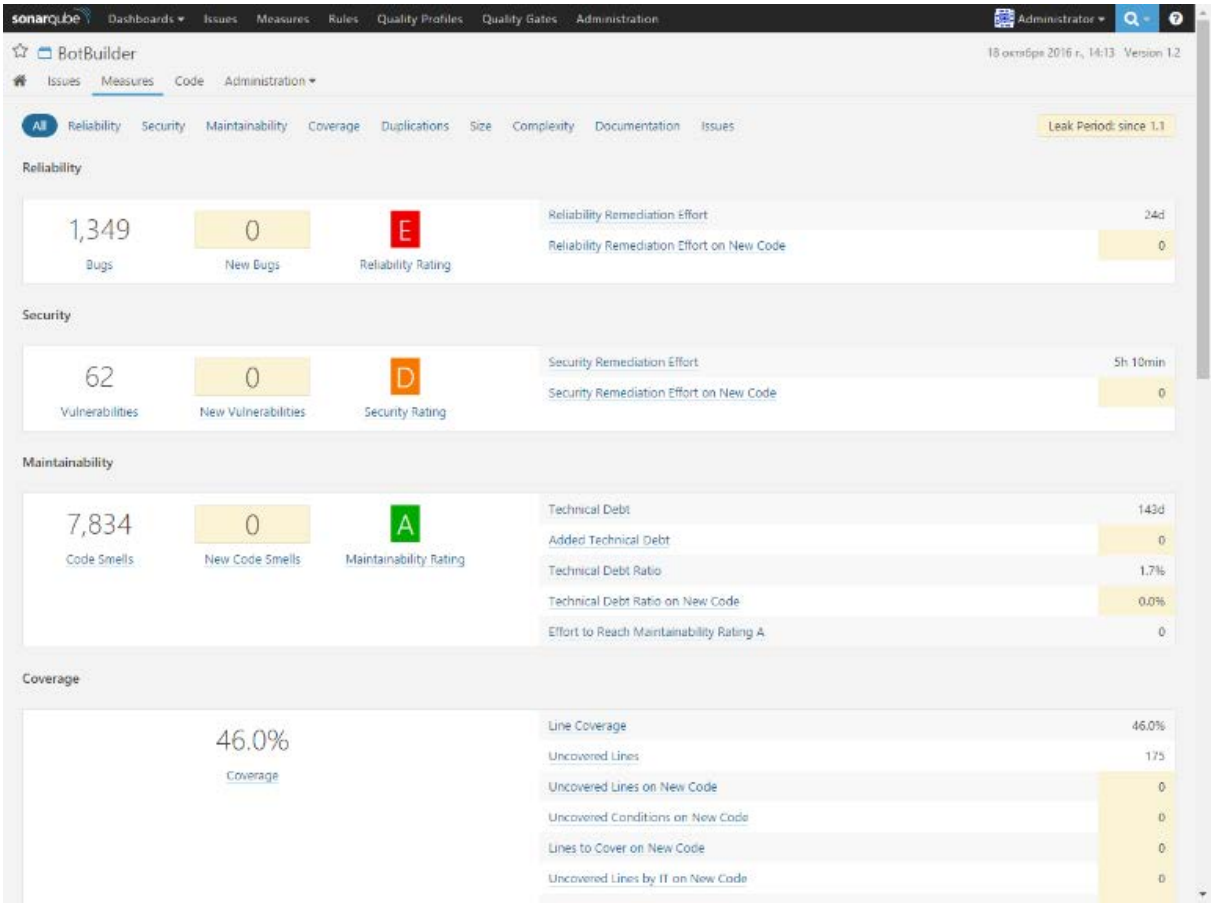

Рис. 2 - Обзор результатов анализа проекта[2]

На данной странице представлена информация о следующих параметрах:

— Надежность (Reliability)

— Безопасность (Security)

— Поддерживаемость (Maintainability)

[2] Составлено автором

— Покрытие тестами (Coverage)

— Дублирование (Duplications)

— Размер кодовой базы (Size)

— Цикломатическая сложность (Complexity)

— Документирование кода (Documentation)

— Ошибки (Issues)

При переходе к просмотру параметра «Надежность» (Reliability) имеется возможность изучить более подробную информацию о следующих аспектах:

— Недавно обнаруженные дефекты

— Общее число дефектов в проекте

— Общее число трудового долга, требуемого для устранения всех выявленных проблем

— Рейтинг надежности проекта по шкале от A до E

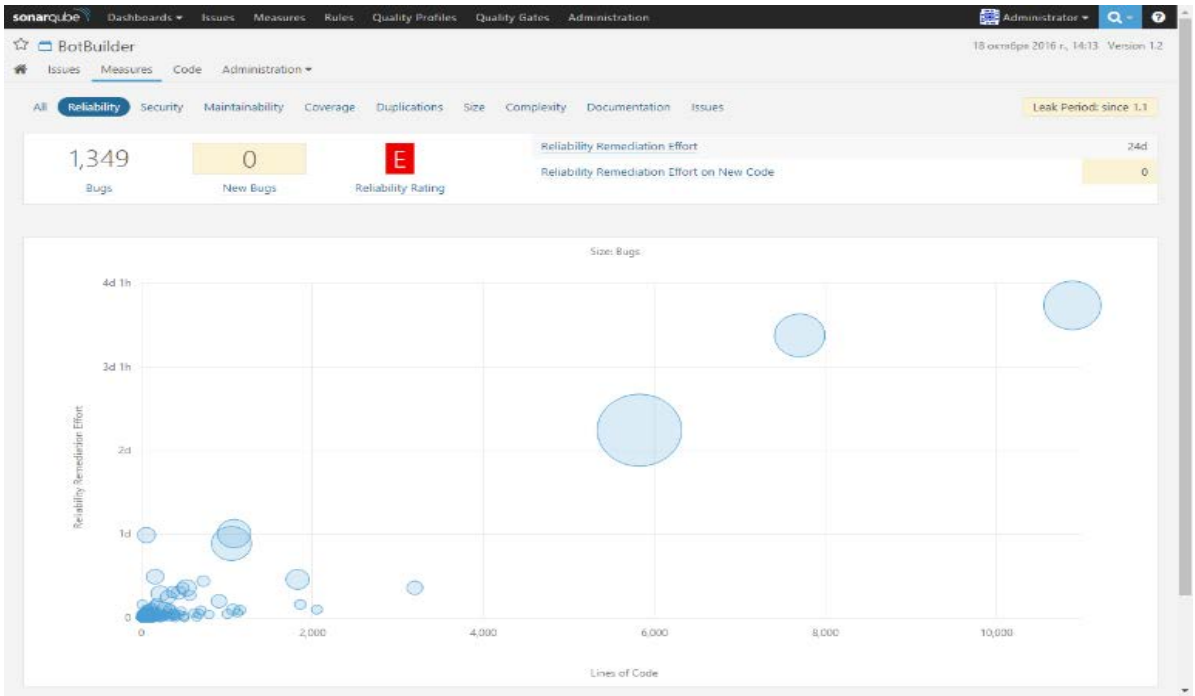

Рис. 3 - Обзор параметра «Надежность» (Reliability)[3]

[3] Составлено автором

Помимо этого, SonarQube позволяет проанализировать все наиболее важные характеристики кода, начиная отдельными файлами и заканчивая уровнем проекта в целом.

Для получения информации о наиболее проблемных участках кода необходимо переключиться на просмотр рейтинга надёжности (Reliability Rating) и отсортировать полученный список файлов проекта по возрастанию рейтинга надёжности.

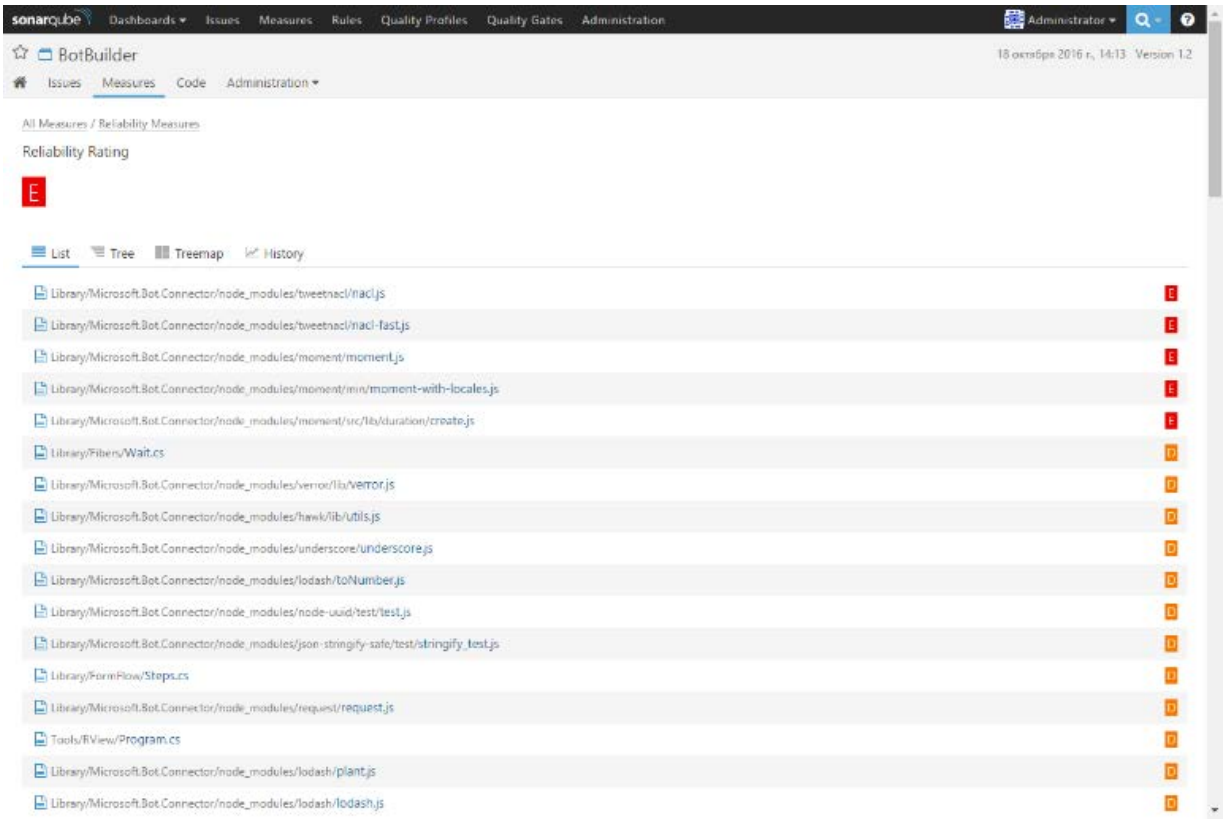

Рис. 4 – Обзор файлов проекта с применением сортировки по возрастанию рейтинга надёжности (Reliability Rating)[4]

Для просмотра исходного кода, в котором был обнаружен дефект, требуется выбрать требуемую запись из полученного списка.

[4] Составлено автором

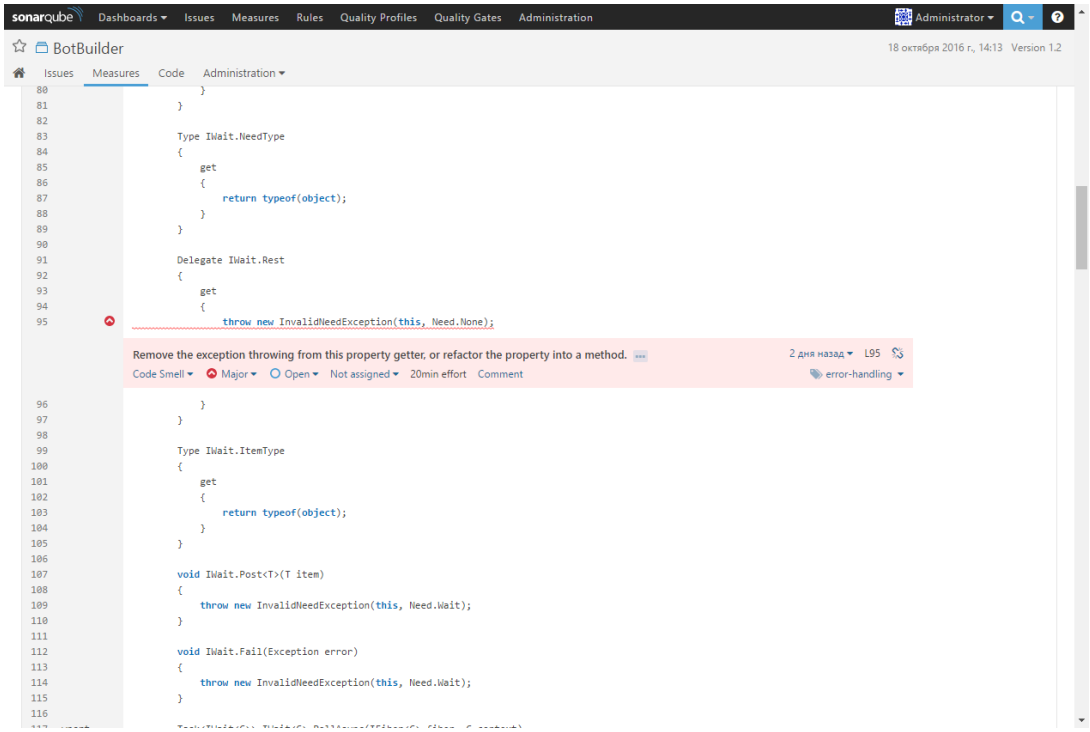

Рис. 5 – Обзор исходного кода выбранного файла проекта[5]

Стоит отметить, что подобный подход к навигации встречается повсеместно для каждого из доступных параметров.

При просмотре вкладки «Безопасность» (Security) можно получить информацию о рейтинге безопасности проекта, общем числе найденных уязвимостей, количестве уязвимостей, введённых в проект с момента последнего релиза, а также общую оценку трудового долга.

[5] Составлено автором

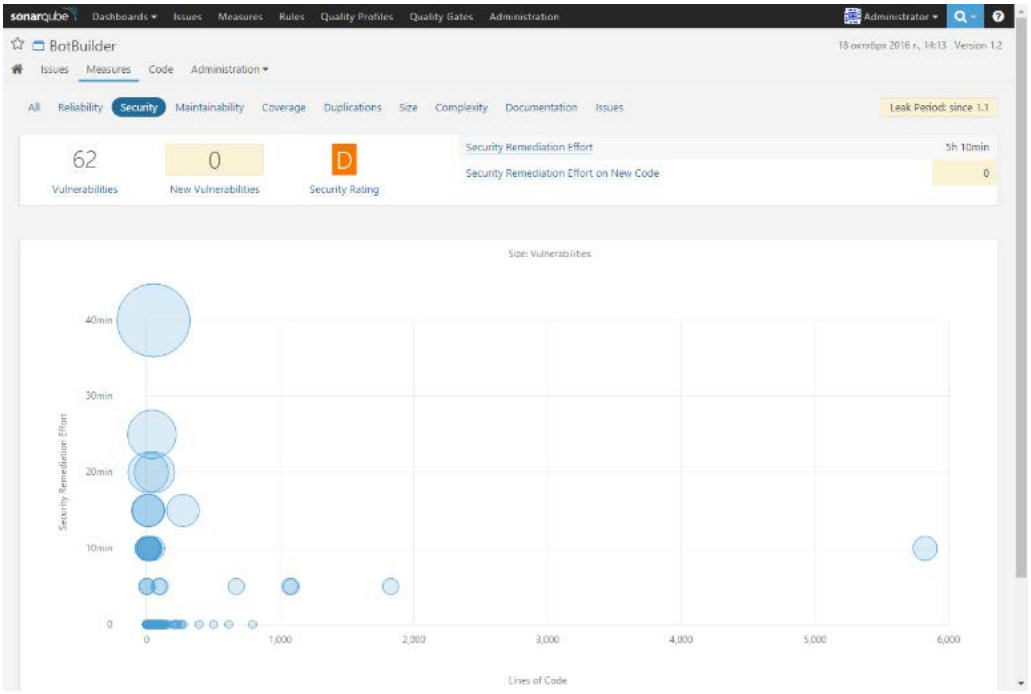

Рис. 6 – Обзор параметра «Безопасность» (Security)[6]

На странице «Поддерживаемость» (Maintainability) можно получить более полную информацию о техническом долге проекта.

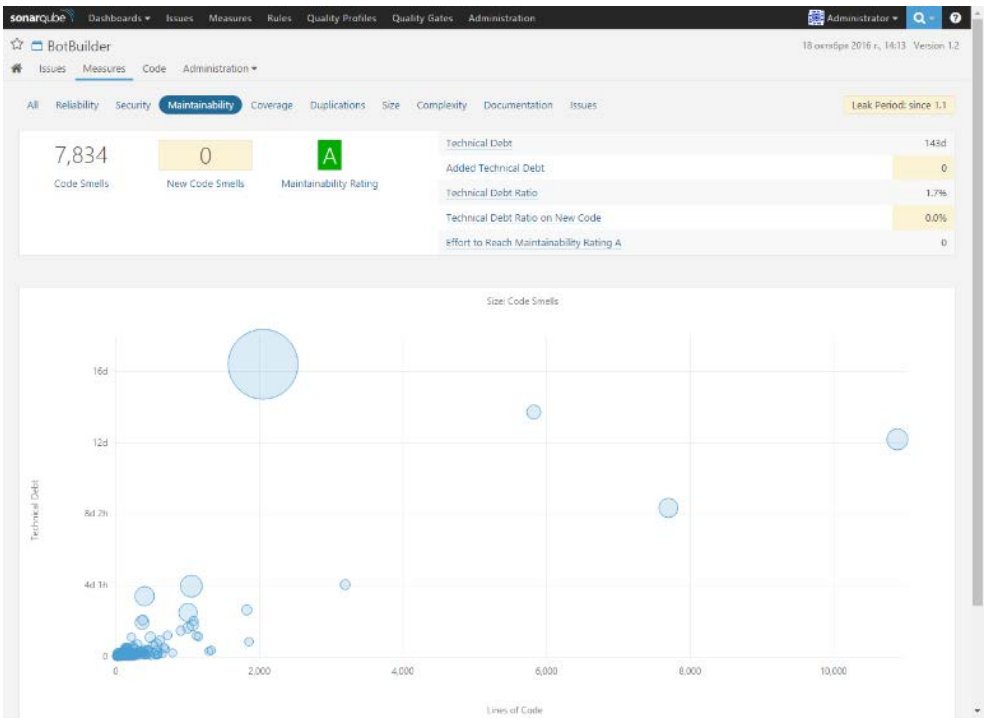

Рис. 7 – Обзор параметра «Поддерживаемость» (Maintainability)[7]

[6] Составлено автором

Перейти к списку отсортированных по количеству обнаруженных мест применения неоптимальных решений можно при помощи интерактивного меню навигации.

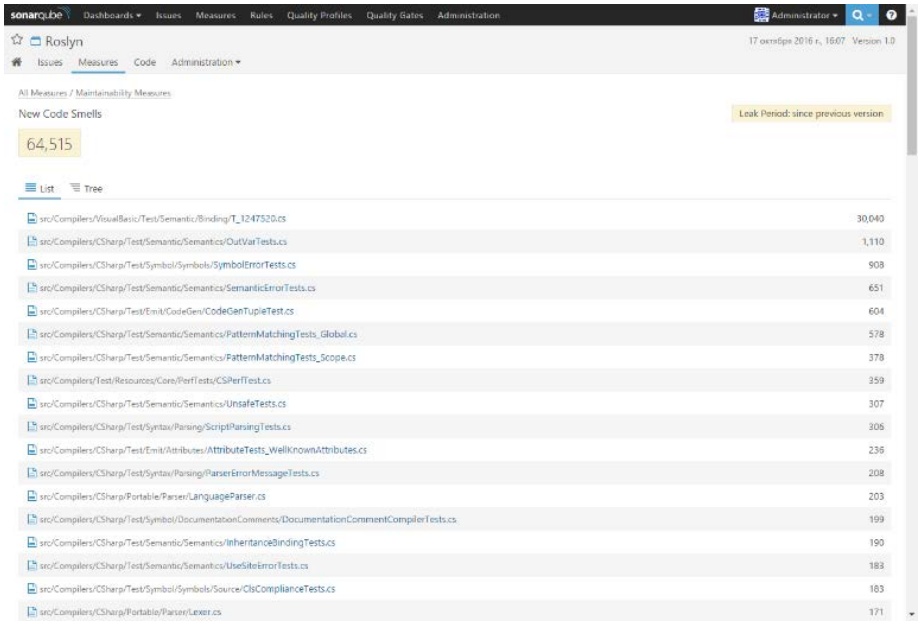

Рис. 8 – Обзор файлов проекта с применением сортировки по возрастанию числа применения неоптимальных решений[8]

Переключение на просмотр найденного участка кода с применением неоптимального решения осуществляется при помощи выбора любого значения из ранее полученного списка.

[7] Составлено автором
[8] Составлено автором

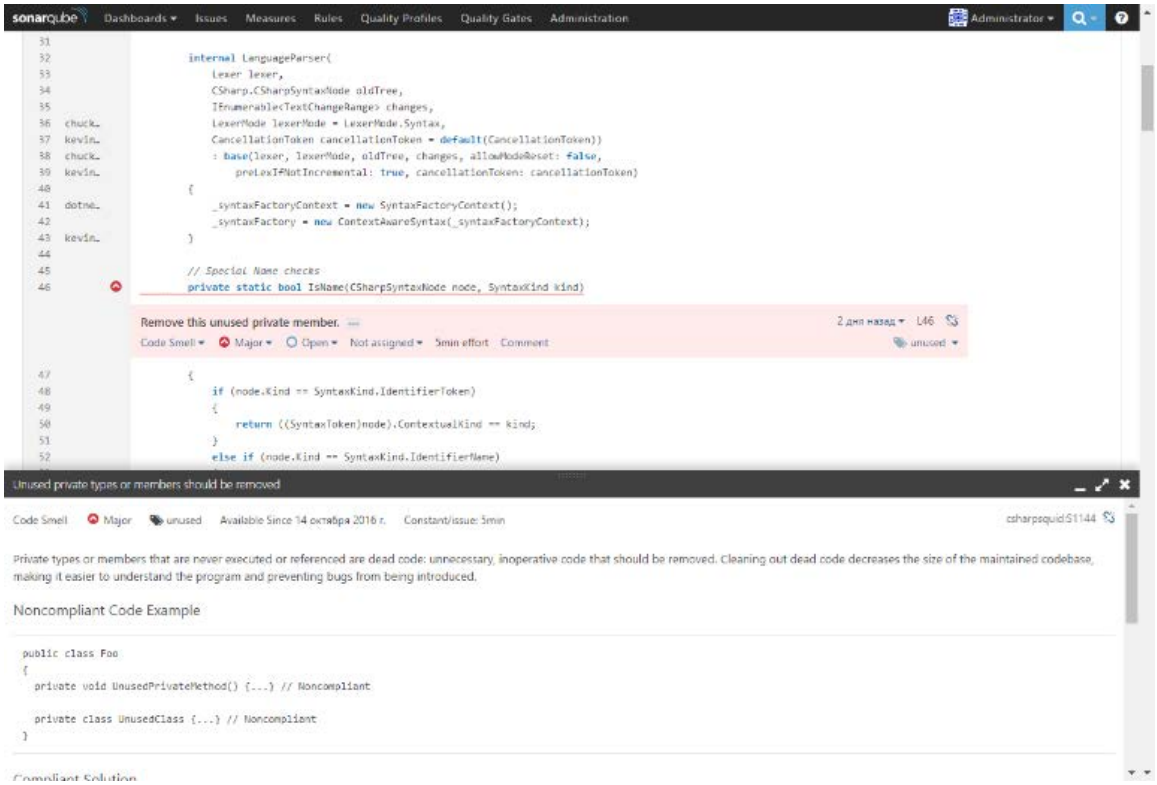

Рис. 9 – Обзор исходного кода выбранного файла проекта[9]

Количество строк кода, покрытых тестами, а также их процент от общего числа строк кода в проекте можно увидеть на странице «Покрытие тестами» (Coverage)

[9] Составлено автором

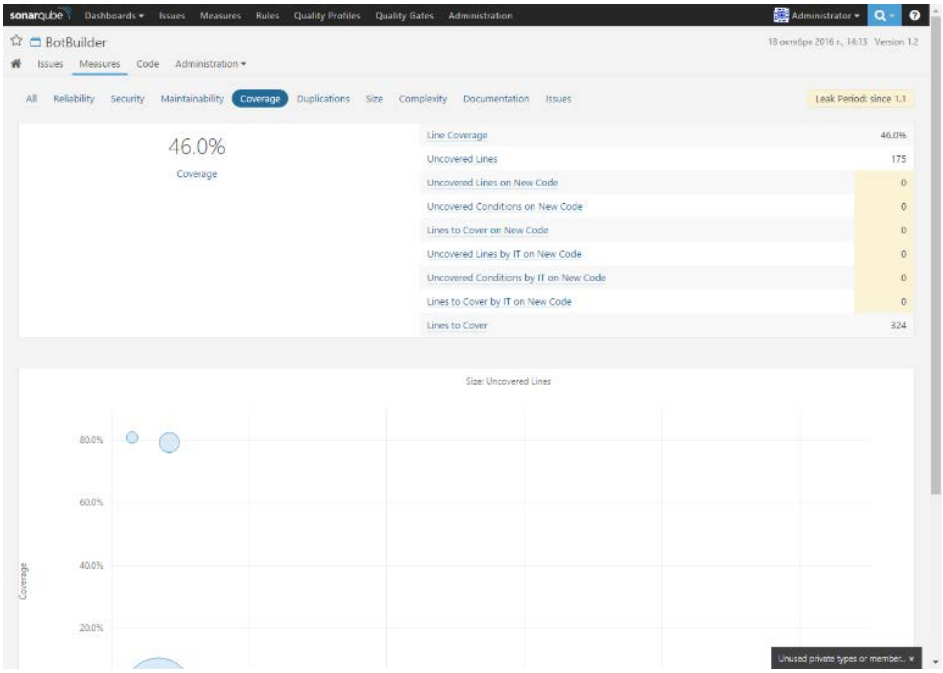

Рис. 10 – Обзор параметра «Покрытие тестами» (Coverage)[10]

Вся информация о классах, в которых наблюдается дублирование исходного кода представлена во вкладке «Дублирование» (Duplications)

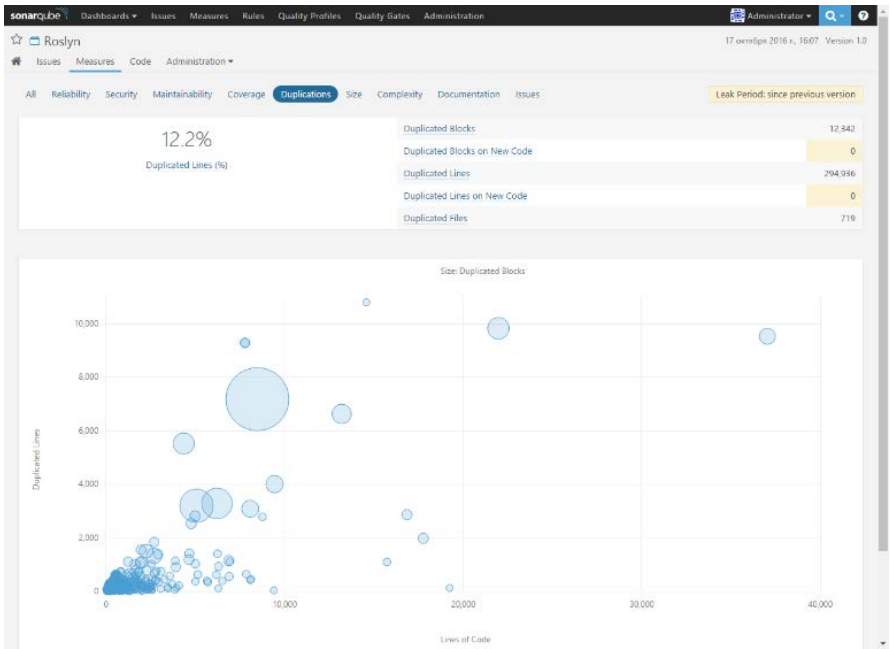

Рис. 11 – Обзор параметра «Дублирование» (Duplications)[11]

[10] Составлено автором

Благодаря данной вкладке можно оценить общее число дублирования строк/блоков/файлов исходного кода.

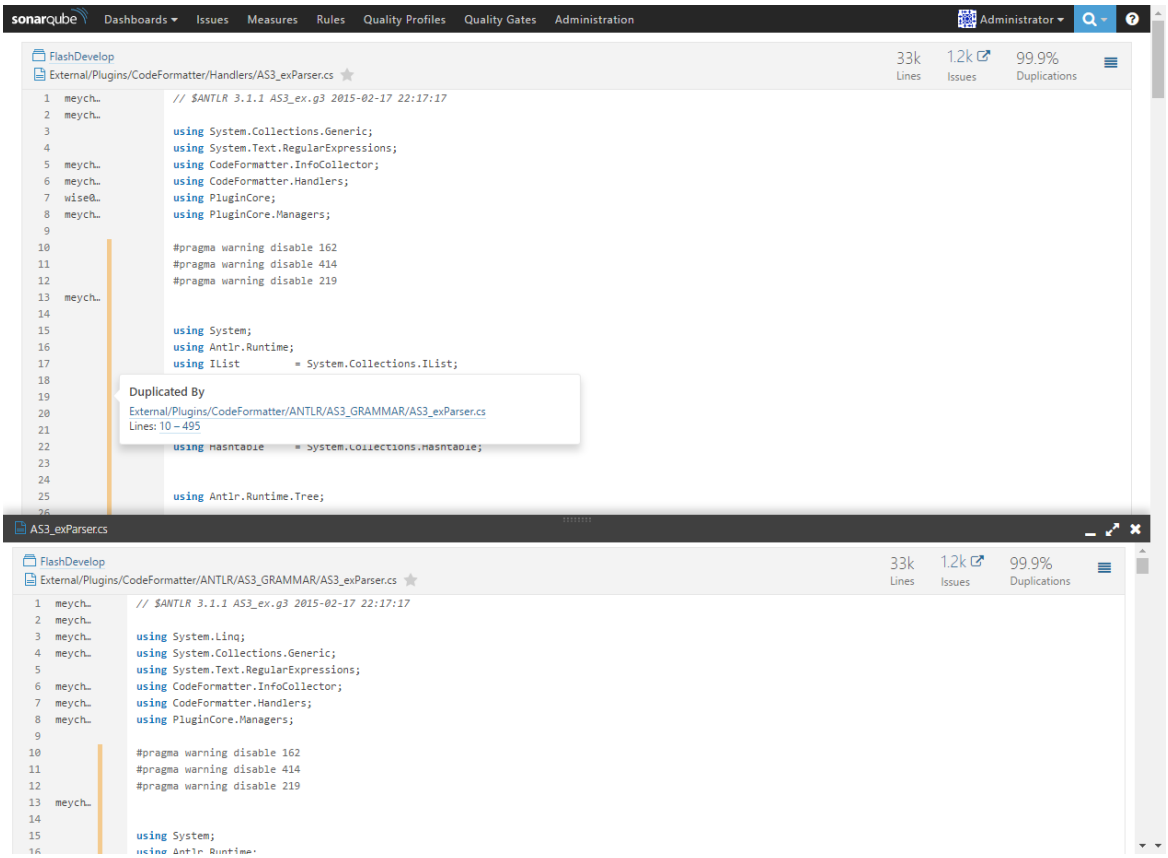

Рис. 12 – Обзор исходного кода выбранного файла проекта[12]

Информация о используемых директориях/файлах/классах/функциях/выражениях и их размерах содержится во вкладке «Размер кодовой базы» (Size)

[11] Составлено автором
[12] Составлено автором

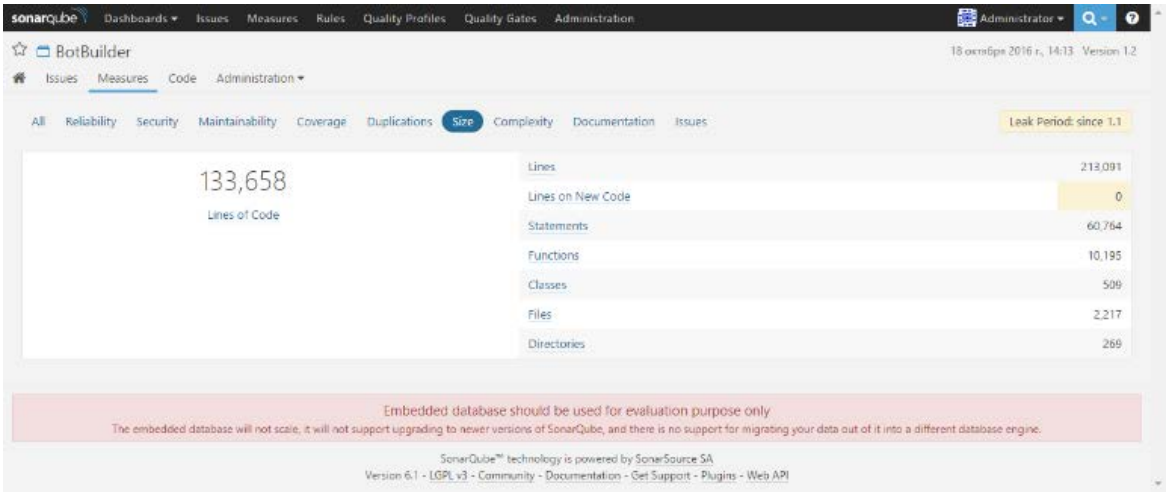

Рис. 13 – Обзор параметра «Размер кодовой базы» (Size)[13]

Вкладка «Цикломатическая сложность» (Complexity) обеспечивает пользователя о средней сложности используемых файлов и функций внутри проекта, а также производит расчет суммарной цикломатической сложности.

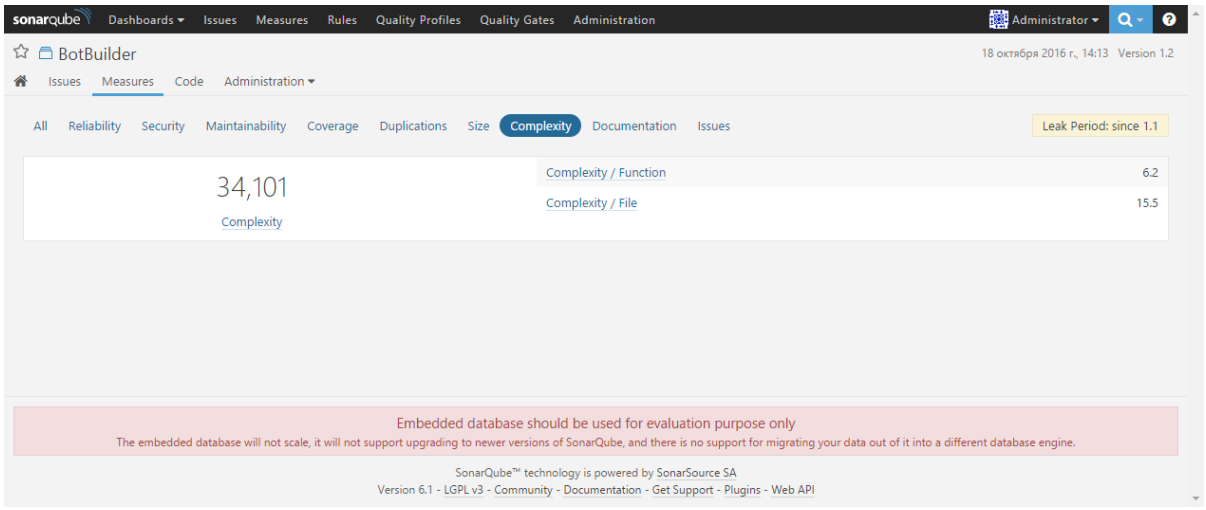

Рис. 14 – Обзор параметра «Цикломатическая сложность» (Complexity)[14]

[13] Составлено автором
[14] Составлено автором

Вкладка «Документирование кода» (Documentation) обладает обширным набором данных относительно комментариев в исходном коде проекта и их применении. Здесь также учитывается количество и уровень документирования публичных API.

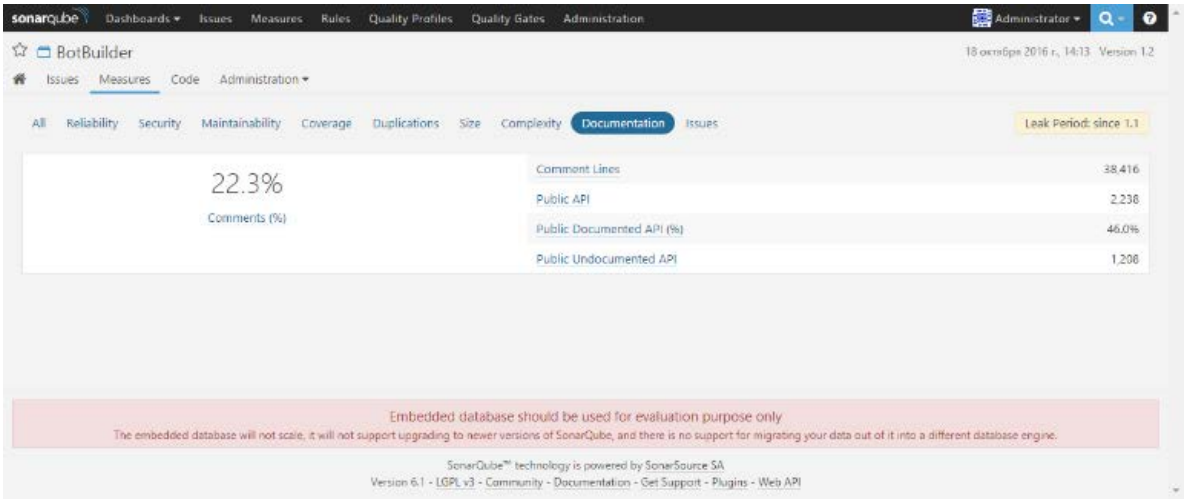

Рис. 15 – Обзор параметра «Документирование кода» (Documentation)[15]

Информация о всех проблемах проанализированного проекта содержится во вкладке «Ошибки» (Issues).

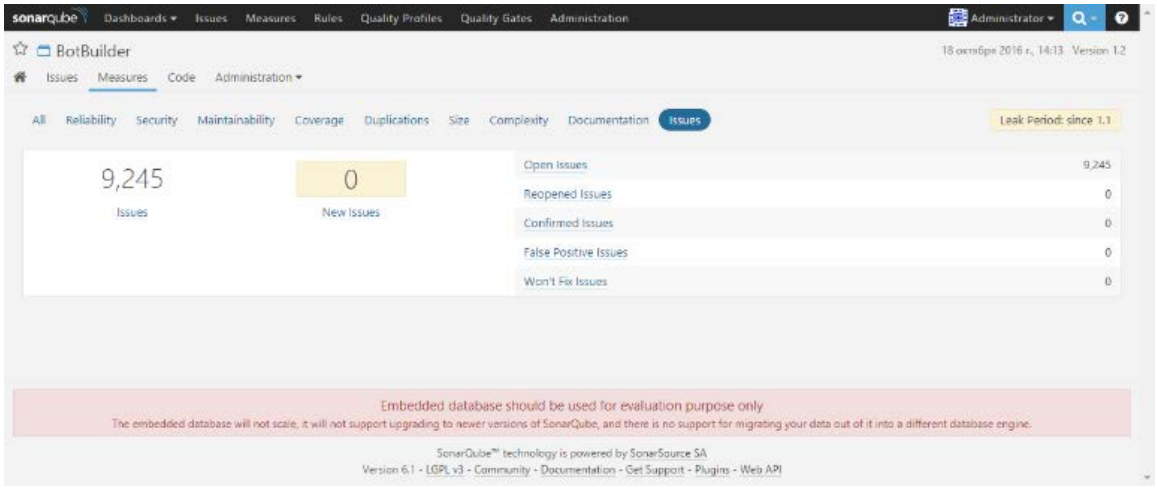

Рис. 16 – Обзор параметра «Ошибки» (Issues)[16]

[15] Составлено автором
[16] Составлено автором

Следует выделить немаловажные особенности основных концепций рассматриваемого продукта:

- SonarQube представляет собой продукт с открытым исходным кодом
- SonarQube обладает низким порогом входа и максимально интуитивным интерфейсом
- SonarQube способен обеспечить функционирование даже в условиях локального использования
- SonarQube приводит всех заинтересованных в процессе разработки к применению единой единицы измерения – технический долг.
- SonarQube предоставляет средства для регулирования числа правил написания кода и количества проводимых проверок.
- SonarQube не выставляет высоких требований к инфраструктуре.

Ввиду того, что применение данной технологии способно гарантировать значительный рост качества разрабатываемого приложения и обладает минимальным набором требований для внедрения, применение данной технологии считается целесообразным в крупных проектах.

Библиографический список

1. Сивохин А.В., Хмелевской Б.Г. Технология разработки программного обеспечения. Описание лабораторных работ. - ПГУ: Пенза, 2002. – 410 с.
2. Липаев, В. Надежность программных средств. / В.В. Липаев. – Москва: СИНТЕГ, 1998. – 358 с.
3. Кролл П., Кратчен Ф. RationalUnifiedProcess - это легко. - М.: «Кудиц-Образ», 2004. – 180 с.

4. Савкин В. Принципы управления качеством программ [Электронный ресурс]. Электрон. текстовые дан. – М.: Открытые системы, 2008-2018. – Режим доступа: http://www.osp.ru/os/2008/06/5344965
5. Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес. Приемы объектно-ориентированного программирования. Паттерны проектирования. -- СПб: Питер, 2001. -- 368 с.:
6. Collaborator. What is Code Review? – [Electronic resource] – URL: https://smartbear.com/learn/code-review/what-is-code-review/
7. SonarQube Docs – [Electronic resource] – URL: https://docs.sonarqube.org/latest/
8. SonarLint Docs – [Electronic resource] – URL: https://www.sonarlint.org/

Оригинальность 96%