Avaliação da Ferramenta SonarQube para a Melhoria da Qualidade de Software por Desenvolvedores Durante a Graduação

Guthyelvis dos Santos Xavier[1]

Departamento de Ciências Exatas (DCX) – Universidade Federal da Paraíba (UFPB) - Campus IV
Cep 58297-000 – Rio Tinto – PB – Brasil

guthyelvis.santos@dcx.ufpb.br

Abstract. *Seeking to improve the learning process related to software development in undergraduate courses in Bachelor of Information Systems and Degree in Computer Science at Campus IV at UFPB, functionalities of the SonarQube tool, which is a web platform for management and control of quality of software projects, have been analysed. During the execution of this work, the tool was presented to some students at different levels of the special undergraduate program to improve training based on direct continuous feedback through reports from SonarQube itself, with the objective of verifying the acceptance of the tool by students in undergraduate courses. Some characteristics were selected considering the different levels of students. According to the results obtained, an efficiency of the tool can be obtained in generating quick feedbacks, encouraging the user to seek ways to solve or understand the problems indicated by the platform, which can be considered an extremely important factor for the learning process and improving software quality. From these results it was possible to observe that the tool and the characteristics in question have great value for the academic community where the study was carried out.*

Resumo. *Buscando melhorar o processo de aprendizado relacionado ao desenvolvimento de software nos cursos de graduação em Bacharelado em Sistemas de Informação e Licenciatura em Ciência da Computação do Campus IV da UFPB, foram analisadas funcionalidades da ferramenta SonarQube, que é uma plataforma web para gerenciamento e controle de qualidade de projetos de software. Durante a execução deste trabalho, a ferramenta foi apresentada para alguns estudantes em diferentes níveis da graduação visando a melhoria contínua do aprendizado a partir de feedbacks diretos gerados através de relatórios do próprio SonarQube com o objetivo de verificar a aceitação da ferramenta por alunos em disciplinas de graduação. Foram selecionadas algumas funcionalidades ponderando os diferentes níveis dos estudantes. De acordo com os resultados obtidos, pôde ser observada a eficiência da ferramenta em gerar feedbacks rápidos, estimulando o usuário a buscar formas de resolver ou entender os*

[1] Trabalho de conclusão de curso, sob orientação da professora Ayla Débora Dantas de Souza Rebouças submetido ao Curso de Bacharelado de Sistemas de Informação do Centro de Ciências Aplicadas e Educação (CCAE) da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de BACHAREL EM SISTEMAS DE INFORMAÇÃO.

problemas indicados pela plataforma, o que pode ser considerado um fator de extrema importância para o processo de aprendizagem e melhoria da qualidade de software. A partir de tais resultados pôde-se observar que a ferramenta e as funcionalidades em questão apresentam grande valor para a comunidade acadêmica onde o estudo foi realizado.

1. Introdução

Nos últimos anos a demanda por profissionais de desenvolvimento de software capacitados tem aumentado cada vez mais, segundo a Associação Brasileira de Empresas de Software(ABES), o setor nacional de TI cresceu em cerca de 9,8%. Apesar de tal crescimento a demanda por profissionais não têm sido atendida devido ao tempo elevado para que esses profissionais se qualifiquem para o trabalho e adquiram experiência suficiente para produzir software com qualidade e uma maior visão de produto. Considerando este cenário, surgiram práticas com o intuito de tornar mais produtivo e eficiente o processo de desenvolvimento de software com qualidade. A análise de código, por exemplo, começou a ganhar força visando garantir a qualidade dos produtos de software e como forma de observar se certos padrões de qualidade estão sendo garantidos, mesmo mantendo-se a agilidade dos times. Em vários momentos, os times de desenvolvimento estão preocupados em apresentar uma solução funcional, mas muitas vezes deixam a desejar na qualidade do projeto. Porém, acredita-se que com ferramentas adequadas se pode apoiar o desenvolvimento de software para se obter maior qualidade nos produtos.

Considerando estes aspectos apresentados, este artigo visa avaliar como a ferramenta SonarQube pode auxiliar o processo de capacitação de desenvolvedores que estão na graduação a melhorar a qualidade de um produto de software. Pretende-se fazer uma análise da ferramenta SonarQube e das funcionalidades utilizadas para dar suporte ao processo de aprendizado, melhorando a qualidade de código e a viabilidade de uso desta ferramenta por desenvolvedores que ainda estão na graduação. Para isso, pretende-se realizar um estudo de caso inicial com alguns alunos de diferentes semestres dos cursos de Licenciatura em Ciência da Computação (LCC) e Bacharelado em Sistemas de Informação (BSI) da UFPB - Campus IV verificando a aceitação da ferramenta pelos estudantes e suas impressões.

Os critérios que levaram à seleção da ferramenta analisada foram o fato de estar disponível gratuitamente, possuir comunidade ativa e documentação robusta, que são fatores de extrema importância para a resolução de eventuais dificuldades de uso com a ferramenta. Além disso, buscou-se identificar uma ferramenta que pudesse ser introduzida com mais facilidade para alunos de diferentes períodos.

As demais seções deste artigo estão organizadas conforme descrito a seguir: a Seção 2 apresenta a fundamentação teórica; a Seção 3 apresenta a metodologia utilizada no trabalho; a Seção 4 apresenta a ferramenta SonarQube; a Seção 5 apresenta a análise de resultados; a Seção 6 apresenta as conclusões e propostas de trabalhos futuros; e, por fim, a seção 7 apresenta as referências utilizadas para o desenvolvimento do projeto.

2. Fundamentação Teórica e Trabalhos Relacionados

Durante o processo de desenvolvimento de um software, frequentemente são encontrados defeitos, falhas de segurança, *code smells,* entre outros problemas**. S**egundo Fowler (1999), *code smells* é um indicador de possível problema estrutural em código fonte, que pode ser melhorado via refatoração, que é uma melhoria no código sem alteração de sua funcionalidade. Considerando possíveis problemas que códigos podem apresentar, áreas

como qualidade de software ganham força, principalmente ao explorarem práticas como análise de código, testes automatizados e softwares externos que visam a melhoria contínua, gerando assim um produto de alta qualidade.

De acordo com Pressman (2001), qualidade de software é a conformidade entre requisitos funcionais e não funcionais que têm sido explicitamente declarados, padrões de desenvolvimento que tenham sido claramente documentados e características implicitamente esperadas de todo software a ser desenvolvido. Com algumas práticas da área de qualidade de software, os problemas são identificados de forma mais rápida e ágil, podendo diminuir prejuízos futuros. Partindo desse pressuposto, a área de gestão da qualidade de software se torna mais importante, uma vez que por meio dela pode ser entregue um produto mais confiável ao nível de usuário.

Criar um software de qualidade é um dos objetivos básicos da engenharia de software. Porém, o termo "qualidade" se torna genérico uma vez que ele depende diretamente do ponto de vista de quem o "observa". Segundo Gomes (2007), um software de qualidade é fácil de usar, funciona corretamente, é de fácil manutenção e mantém a integridade dos dados em falhas do ambiente ou outras fora do seu controle.

Uma das formas de se tentar conseguir qualidade de código é por meio da análise estática. A análise estática de código é uma técnica que consiste em buscar identificar problemas antes mesmo da execução do código para que sejam rapidamente resolvidos. A primeira ferramenta de análise estática surgiu em meados dos anos 1970 [Chelf; Chou 2007], e ficou conhecida como Lint. A criação desta ferramenta ocorreu logo após a revisão formal e as inspeções de código terem sido reconhecidas como peças-chaves para a produtividade e a qualidade final do produto [Gomes et al. 2009].

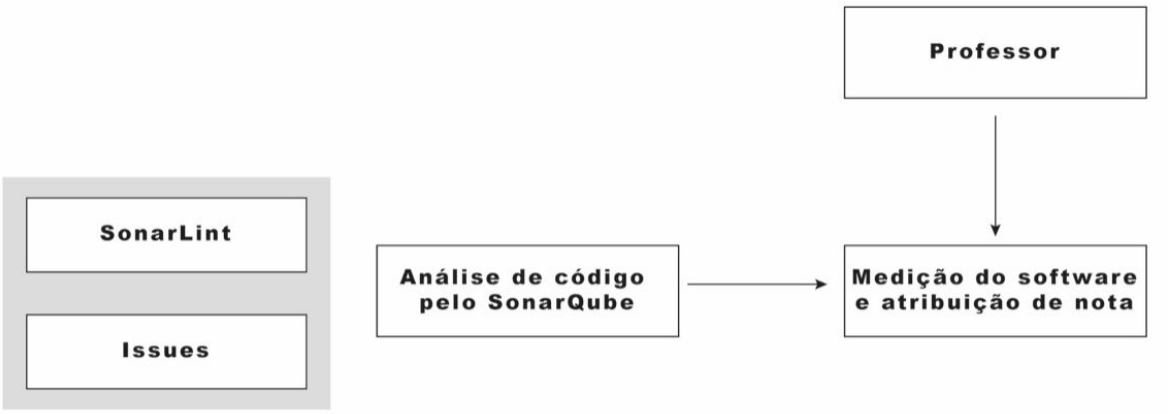
Uma das formas de realizar análise estática é por meio de ferramentas de análise de código, que são ferramentas que executam varreduras no código fonte em busca de potenciais erros, e em alguns casos sem a necessidade da execução. Essas ferramentas são caracterizadas por funções padrão com a finalidade de melhorar a qualidade do código a partir de métricas pré-estabelecidas. Algumas dessas funções são: exibição de alertas de fácil entendimento, localização de *bugs*, *code smells*, falhas de segurança, entre outras. Alguns exemplos destas ferramentas são: SonarQube[2], PMD[3], FindBugs[4] e Checkstyle[5].

No contexto acadêmico, segundo Yulianto e Liem (2014), estas ferramentas também podem ser utilizadas no contexto educacional para facilitar o processo de avaliação de atividades práticas de programação de alunos. Além de ferramentas deste gênero poderem ser consideradas úteis no processo de fixação do conteúdo, elas podem ser abordadas também como uma estratégia de avaliação do professor. Segundo Alves (2019), quando realizada de forma manual pelo professor, a análise do código-fonte é um processo trabalhoso e repetitivo, pois exige que o professor realize a mesma análise minuciosa para cada programa individualmente. Nas disciplinas de programação, normalmente é utilizada a técnica *learning by doing* que de acordo com Lesgold (2001), consiste em ministrar conceitos teóricos junto a aplicação de exercícios. Sinthsirimana e Panjaburee (2013), propõem um modelo arquitetural de um analisador de código adaptado ao contexto da pesquisa apresentado na Figura 1.

---

[2] SONARSOURCE. SonarQube. Disponível em: https://www.sonarqube.org/. Acesso em: 22 de ago. de 2020.
[3] SOURCEFORGE. PMD. Disponível em: https://pmd.github.io/. Acesso em: 22 de ago. de 2020.
[4] PUGH, William; DAVID, Homemeyer. Disponível em: http://findbugs.sourceforge.net/ Acesso em: 22 de ago. de 2020.
[5] BURN, Oliver. Disponível em: https://checkstyle.sourceforge.io/ Acesso em: 22 de ago. de 2020.

Figura 1. - Arquitetura de um analisador de código voltado ao contexto educacional.



Fonte: Adaptado de Sinthsirimana e Panjaburee (2013), versão traduzida.

Sendo assim, uma ferramenta deste tipo auxilia o processo de ensino e aprendizagem já que é capaz de avaliar vários programas em um curto período de tempo de forma coerente e fornecer feedback detalhado de forma imediata aos alunos (Yulianto; Liem, 2014).

Além de ferramentas de verificação de código, é importante destacar ainda que outra forma de buscar qualidade de software é por meio de abordagens como inspeções e testes de software, e que fazem parte do processo de Verificação e Validação (V&V), conforme destaca Sommerville. Os testes de software consistem em executar uma implementação do software com dados de teste para examinar as saídas e o comportamento operacional. Já as inspeções de software analisam e verificam representações de sistema como documentos de requisitos, diagramas de projeto e código fonte do programa de forma estática, a partir do método de inspeção. No entanto, o presente trabalho não focará em inspeções e testes como estratégia para garantir a qualidade de software.

Um trabalho relacionado ao presente artigo é o dos pesquisadores Schneider e Jaques (2016), que utilizaram um ambiente virtual de aprendizagem para apoiar as disciplinas de programação, chamado Feeper. Foram utilizados questionários seguindo a escala Likert de concordância para uma amostra de treze usuários que utilizaram a ferramenta por um período de três semanas, resultando no aumento da qualidade do auxílio dado pela ferramenta aos alunos durante a programação de suas soluções.

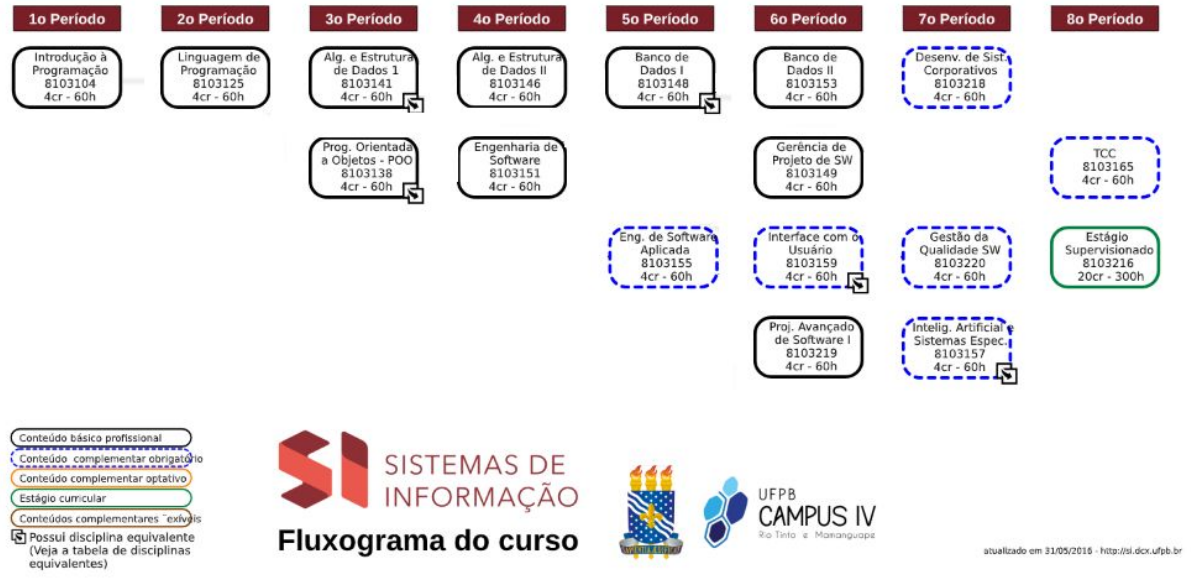
Outro trabalho relacionado é o de Raabe et al. (2015), onde os autores realizaram avaliação sobre como o feedback fornecido por um corretor automático de algoritmos impactou nas atitudes subsequentes dos estudantes. O artigo indicou que o corretor teve um impacto positivo na dinâmica de resolução e depuração das soluções pelos estudantes, mas apontou a necessidade de ampliação no número de testes estáticos a serem realizados.

De forma complementar a tais estudos, o presente trabalho pretende apresentar uma Avaliação da Ferramenta SonarQube para a melhoria da qualidade de software por desenvolvedores durante a graduação fazendo um estudo de caso com a comunidade acadêmica dos cursos de Bacharelado em Sistemas de Informação (BSI) e Licenciatura em Ciência da Computação (LCC) da UFPB-Campus IV avaliando a aceitação da ferramenta por tais alunos e observando como a ferramenta poderá auxiliar o processo de aprendizado.

3. Metodologia

Através de uma pesquisa exploratória, o presente artigo visou identificar inicialmente ferramentas de análise de código que pudessem auxiliar no processo de aprendizado sobre qualidade de software nas disciplinas envolvendo desenvolvimento de software dos cursos de Bacharelado em Sistemas de Informação (BSI) e Licenciatura em Ciência da Computação (LCC) da UFPB-Campus IV, aumentando a qualidade de softwares produzidos por tais alunos de graduação. Foi então selecionada uma ferramenta, o SonarQube, e foram analisadas as funcionalidades *SonarLint* e *Issues* utilizadas com o propósito de elevar a qualidade de produtos de software produzidos e auxiliar o processo de aprendizado. Uma vez identificados os principais aspectos da ferramenta, foi realizado um estudo de caso inicial com estudantes dos cursos de BSI e LCC. Para o estudo de caso foram realizadas demonstrações sobre o potencial da ferramenta SonarQube e suas funcionalidades, utilizando os relatórios gerados pela própria ferramenta a partir do projeto "Caderneta - Backend"[6], desenvolvido na linguagem Java. As demonstrações foram feitas individualmente através da plataforma Google Hangouts, para 8 alunos entre o terceiro e décimo período e posteriormente foram realizadas entrevistas semiestruturadas para detalhar a experiência dos alunos e como a ferramenta pode/poderia impactar a experiência de cada um dos entrevistados em disciplinas de seus cursos como as disciplinas apresentadas respectivamente nas Figuras 2 e 3.

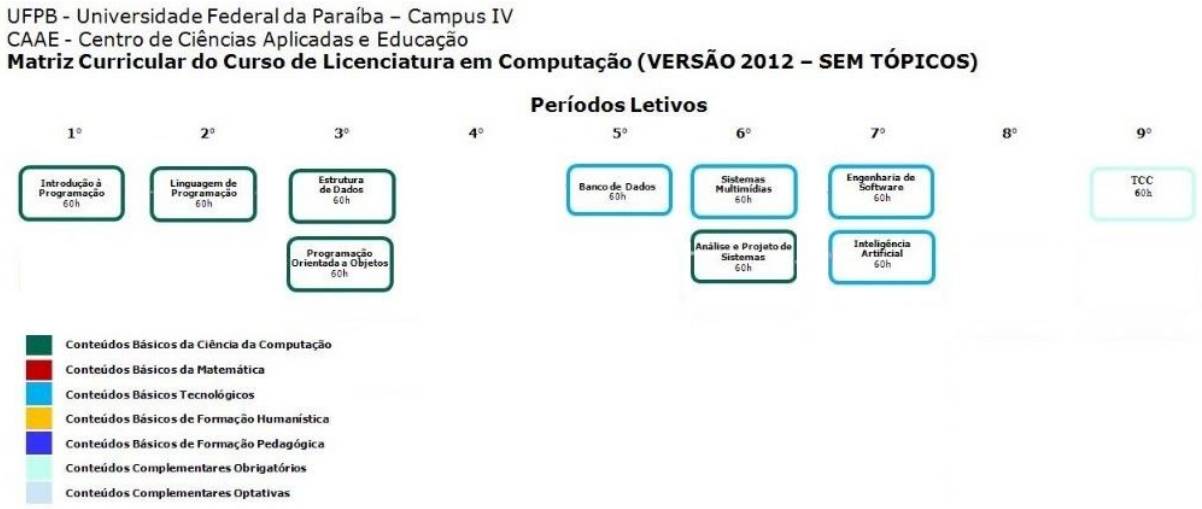
Figura 2. Disciplinas da matriz curricular do curso Bacharelado em Sistemas de Informação.



Fonte: Adaptado de https://si.dcx.ufpb.br/fluxograma/

[6] Caderneta - Backend. AIRON, Osvaldo; COUTINHO, Vinicius; XAVIER, Guthyelvis; TAVARES, Alberis. Disponível em: https://drive.google.com/drive/folders/1I511fECNnKfqk7bSiaOSVSI3wlajEpuh?usp=sharing Acesso em: 07/10/2020.

Figura 3. Disciplinas da matriz curricular do curso de Licenciatura em ciência da computação



Fonte: Adaptado de https://www.ufpb.br/lcc/contents/menu/PPC

3.1 Estudo de caso

A demonstração da ferramenta aconteceu a partir da leitura e exploração dos relatórios gerados a partir da execução do SonarQube utilizando o projeto Caderneta - Backend. Inicialmente foi explorada a seção "*Code Smells*" a fim de observar possíveis notificações de problemas de código correspondentes a erros que fossem familiares para os voluntários demonstrando as causas para cada *code smell* envolvido e o potencial de detecção da ferramenta. Para cada "*Code Smell"* detectado, foram observadas a análise realizada pela ferramenta, o exemplo do código apresentando, a inconformidade para o trecho de código em questão e a(s) solução(ões) propostas pela ferramenta. Em seguida, foi explorada a seção "*Bugs*" repetindo os passos executados para a seção anterior. Após a demonstração da ferramenta, foram realizadas as entrevistas seguindo o roteiro proposto (Apêndice A). Em seguida foram realizados os processos de Análise dos dados coletados e classificação dos voluntários (Descritos na seção 3.2) e por fim foram levantados os resultados do projeto com base nos dados coletados.

3.2. Coleta e Análise de Dados

Foram coletados dados de 8 alunos cursando diferentes semestres do curso através de reuniões por meio da plataforma Google Hangouts. As entrevistas foram compostas de perguntas abertas com o objetivo de avaliar possíveis pontos positivos e negativos relacionados à experiência do aluno com a ferramenta SonarQube e a experiência dos mesmos com os seus respectivos cursos. Para cada entrevista, o entrevistador realizou uma classificação dos entrevistados, associando a cada um deles um identificador, o seu curso e semestre atual conforme ilustrado no Quadro 1. Em seguida, as entrevistas foram transcritas de modo a ter resultados baseados na experiência dos alunos com a ferramenta utilizada e a relevância que poderia ter para disciplinas do curso.

Quadro 1. Perfis dos alunos que participaram do estudo

ID	Curso	Semestre
AL01	BSI	Terceiro
AL02	BSI	Quarto
AL 03	BSI	Nono
AL04	BSI	Décimo
AL05	LCC	Sexto
AL06	LCC	Sexto
AL07	BSI	Nono
AL08	BSI	Nono

4. Ferramenta e funcionalidades analisadas

4.1. SonarQube

O SonarQube "é uma plataforma Web aberta para gerenciar a qualidade de código, oferecendo indicadores e métricas sobre a qualidade dos códigos-fonte de seus projetos" [Barros 2010]. A ferramenta executa varreduras no código fonte em busca de potenciais erros. Um dos maiores benefícios da utilização da ferramenta como um detector de defeitos durante a fase de codificação é a facilitação da correção com menor custo.

4.2 Ferramentas oferecidas pelo SonarQube

O SonarQube fornece ferramentas integradas que podem auxiliar diretamente o desenvolvedor durante o processo de codificação. Como objeto de estudo, foram selecionadas duas funções das cinco listadas pela documentação do próprio SonarQube, e que estão descritas no Quadro 2. Outras funcionalidades oferecidas foram removidas do escopo deste artigo devido à complexidade ou não adaptabilidade à realidade dos cursos em questão.

Quadro 2. Ferramentas integradas selecionadas como objeto de estudo

SonarLint	SonarLint é um produto complementar que funciona em seu editor, fornecendo *feedback* imediato para que você possa detectar e corrigir problemas antes de enviar o código ao repositório.
Issues	A ferramenta levanta problemas sempre que uma parte do seu código quebra uma regra de codificação, seja um erro que irá quebrar o seu código (*bug*), um ponto no seu código aberto a ataques (vulnerabilidade) ou um problema de manutenção (*code smell*).

4.3. Detecção de *code smells* pelo SonarLint

O SonarLint é uma ferramenta muito eficiente na detecção de *code smells* e a partir das métricas de detecção também traz possíveis refatorações para ajudar o desenvolvedor a corrigir e avaliar o impacto do problema. Alguns dos principais exemplos estão dispostos no Quadro 3.

Quadro 3. Exemplos de *code smells* e refatorações sugeridas pelo SonarQube

Code smells	Refatorações sugeridas
Duplicate Code: código idêntico ou muito semelhante existente.	Mover o código para sua própria unidade e chamar a unidade nos locais onde foi originalmente usada.
Large Class / God Class: Uma classe que faz coisa demais no sistema.	*Extract Class*: dividir a classe *Extract Subclass*: criar uma subclasse para a classe
Long Method / God Method: Métodos que centralizam a funcionalidade da classe	*Extract Method*: dividir o método em dois *Replace Method with Method Object*: transformar um método em uma classe
Feature Envy: um método de uma classe usa atributos somente da outra classe	*Move Method e Move Field*: mover métodos e atributos entre classes *Inline Class*: juntar duas classes em uma
Divergent Change: Ocorre quando uma classe pode mudar frequentemente de diferentes formas e por razões distintas	*Extract Class*: extrair a classe.
Shotgun Surgery: Toda vez que uma classe é alterada, você tem que fazer várias pequenas mudanças em outras classes diferentes	*Move Method e Move Field*: mover métodos e atributos entre classes. *Inline Class*: juntar duas classes em uma.
Refused Bequest: Uma classe herda atributos e métodos de outra classe, mas não os usa	*Push Down Method*: mover o método ou atributo da superclasse para a subclasse. *Replace Inheritance with Delegation*: substituir o relacionamento de herança por associação
Comments: Comentários não é uma coisa ruim. Entretanto, excesso de comentários pode indicar que os nomes de métodos/atributos não estão suficientemente expressivos	*Extract Method*: quebrar um método em dois *Rename Method/Field*: renomear

Fonte: Adaptado de https://pt.slideshare.net/annuvinayak/code-smells-and-its-type-with-example, versão traduzida

5. Resultados

Os resultados obtidos foram divididos em tópicos, referenciando os dados coletados dos alunos que participaram da pesquisa seguindo os passos metodológicos descritos na Seção 3 por meio de entrevistas realizadas através da plataforma Google Hangouts.

5.1. Experiências anteriores com ferramentas de análise de código

Do total de entrevistados, apenas 3 alunos informaram que passaram por experiências anteriores com ferramentas do gênero, mas relataram que tais experiências foram limitadas.

- *"Serviu apenas para saber que a ferramenta existia...." (AL08)*

5.2. Dificuldades e/ou impedimentos associados ao processo de aprendizagem relacionado às disciplinas de programação

Esta seção faz referência a pergunta "*P3*" do roteiro de entrevista do Apêndice A e foi dividida em dois tópicos baseados nas perspectivas gerais dos alunos e um aspecto relatado especificamente pelos alunos do curso de LCC.

5.2.1. Dificuldades e/ou impedimentos associados ao processo de aprendizagem nos cursos de BSI e LCC

Dos alunos que apresentaram dificuldades com programação, todos relataram a dificuldade de entender como a programação funciona e a dificuldade em relacionar teoria e prática.

- *"É complicado entender como funciona por baixo dos panos" (AL01)*

Um impedimento relatado por alguns alunos foi a falta de dedicação nos períodos iniciais que ocasionaram um problema ainda maior quando somado à velocidade em que o mercado evolui e a gama de tecnologias que surgem periodicamente. Segundo Lemos et al. (2019), o processo de ensino-aprendizagem é composto pela colaboração mútua entre aluno e professor. Nesse sentido, independente da metodologia adotada pelo professor, caso não haja dedicação do aluno, o processo de apreensão e construção do conhecimento ficará comprometido. O insucesso nas disciplinas iniciais podem ser associados à dificuldade em compreender e aplicar conceitos básicos de programação e relação de como um algoritmo poderá resolver um problema real.

5.2.2. Dificuldades e/ou impedimentos associados ao processo de aprendizado no curso de LCC

Os alunos entrevistados relataram que o fluxograma das disciplinas do curso em questão apresentado na Figura 2 é defasado durante o quarto e quinto período, uma vez que as disciplinas não focam em programação de fato naqueles semestres.

- *"Acabei deixando de lado para focar nas disciplinas do período." (AL05)*

5.3. Recursos oferecidos pela universidade para motivar o aluno

Quando questionados sobre formas de estimular o aprendizado oferecidas pela universidade ("P4"), todos os alunos associaram o estímulo ao aprendizado diretamente à atuação profissional de alguns professores e seus respectivos métodos de ensino.

- *"A empolgação dele contagia qualquer um!". (AL06)*

Outro questionamento importante foi associado às ferramentas disponibilizadas pela universidade, especificamente pelo Departamento de Ciências Exatas (DCX), onde as mesmas possuem pouca ou nenhuma integração com as disciplinas e a necessidade de possíveis novas ferramentas que estejam mais alinhadas com as práticas do mercado de trabalho como o SonarQube por exemplo. Alguns dos entrevistados não tinham conhecimento sobre a existência do Software Hub do departamento, portal onde as ferramentas são disponibilizadas.

5.4. Principais problemas identificados no uso da ferramenta na graduação

Durante a demonstração da ferramenta, foram levantados três principais problemas relacionados ao uso da ferramenta e discutidos nas subseções a seguir.

5.4.1 Detecção de bugs

Dos oito entrevistados, apenas 2 questionaram a relevância da funcionalidade de detecção de bugs alegando que para as demonstrações de projetos realizadas na maioria das disciplinas, o fluxo principal sempre é testado durante o processo de desenvolvimento, então a utilização da ferramenta deveria ser focada no aprimoramento das "*skills*" dos alunos.

5.4.2 Detecção de "*code smells*"

Um aluno avaliou que a funcionalidade detecção de "*code smells*" pode ser prejudicial nos dois primeiros períodos de ambos os cursos uma vez que os alunos que não possuem conhecimento prévio estarão focados em entender os conceitos básicos e ao ser submetido a uma quantidade maior de informação poderá perder o foco no objetivo da disciplina.

- *"Não sei se é tão importante para um aluno aprender boas práticas se ele ainda está aprendendo o que é programação e os conceitos básicos de uma linguagem." (AL07)*

Tal visão pode ter sido enviesada devido à funcionalidade ter sido demonstrada em sua totalidade. Contudo todas as funcionalidades em questão podem ter uma definição de regras de uso personalizadas, levando em consideração especificamente a experiência do aluno que irá utilizá-la. Este recurso poderá ser utilizado pelo professor para definir um nível de cobrança para a turma em questão.

5.4.3 Utilização do SonarQube em disciplinas

Do total, 75% dos alunos preocuparam-se com a forma na qual a ferramenta poderia ser utilizada por professores como parte do escopo de um projeto, enxergando a ferramenta como uma forma de aumentar a cobrança por parte dos professores.

- *"Na maioria das vezes não consigo entregar nem o escopo planejado…" (AL06)*

A partir de tais questionamentos, foi possível observar que os alunos enxergaram a ferramenta apenas como um analisador a fim de atingir um padrão de qualidade pré estabelecido, quando na verdade a utilização da ferramenta pode trazer uma maior dinamicidade ao desenvolvimento do projeto, visto que o professor poderá avaliar um projeto além das funcionalidades entregues, e a qualidade faria parte de um escopo planejado assim como foi citado pelo *AL06.*

5.5 Feedback gerado pela ferramenta

De modo geral, a maioria dos alunos reagiram positivamente aos feedbacks gerados pela ferramenta. Suas funcionalidades exploradas no estudo foram consideradas pela maioria dos participantes do estudo como muito importantes tanto no processo de aprendizado como para os alunos de maior experiência, uma vez que a ferramenta acelera o processo de análise para corrigir um problema.

5.6 Detecção de *code smells*

Das funções exploradas, a detecção de *code smells* foi considerada a mais importante por dois motivos. Para desenvolvedores que se encontram no início do curso, a funcionalidade sempre detecta e reforça trechos básicos que podem ser refatorados no código, reforçando assim as boas práticas de programação. A segunda motivação parte de desenvolvedores com mais experiência, que seria reavaliar os produtos gerados e dar a oportunidade de refatoração a partir de uma análise rápida, permitindo a correção de pequenos hábitos pessoais na codificação que futuramente podem influenciar de forma negativa no processo de manutenção do código.

- *"É muito simples a forma como a ferramenta trata os problemas no código e além do resultado de como resolver". (AL01)*
- *"Quando o projeto já está avançado é complicado parar para observar o que pode ser melhorado".(AL06)*

6. Conclusões e Trabalhos Futuros

Este projeto teve como objetivo realizar uma avaliação da ferramenta SonarQube como uma auxiliadora no processo de aprendizagem e melhoria da qualidade de código por alunos de graduação dos cursos de Bacharelado em Sistemas de Informação(BSI) e Licenciatura em Ciência da Computação(LCC) da UFPB-Campus IV.

Foi realizada uma demonstração da ferramenta de modo a coletar feedbacks com base na experiência dos voluntários do estudo. Com base nos resultados ponderados a partir da transcrição de entrevistas, foi possível aferir resultados positivos e negativos relacionados à aceitação da ferramenta por tais alunos.

A partir das informações coletadas, pôde-se concluir que a ferramenta e as funcionalidades apresentadas neste artigo apresentam grande valor para a comunidade acadêmica em questão, visto que todos os voluntários reagiram de forma positiva em relação ao objeto de estudo, porém demonstraram receio sobre o uso da ferramenta como meio de gerar maior cobrança por parte dos docentes. Sendo assim, acredita-se que ela pode ser sugerida aos alunos de disciplinas envolvendo projetos como forma de gerar feedback rápido sobre melhorias que podem ser realizadas em seus códigos, desde que o professor em questão consiga desenvolver uma metodologia de uso da ferramenta buscando sanar possíveis *gaps* no conhecimento que os alunos possuem e tentando adaptar regras de uso da ferramenta ao nível de cobrança da disciplina na qual será aplicada.

Como trabalho futuro, pretende-se realizar a aplicação da ferramenta em uma ou mais disciplinas durante um semestre, de modo que seja possível coletar dados referentes à experiência dos alunos familiarizados com a ferramenta utilizando a mesma durante o desenvolvimento de seus projetos e coletando também dados de outros *stakeholders*, como professores e empresas. Possíveis disciplinas em que a ferramenta pode ser analisada são as disciplina de Análise e Projeto de Sistemas (em LCC) e a disciplina de Programação Orientada a Objetos (LCC e SI) já que nestas disciplinas são discutidas boas práticas de programação orientada a objetos e que são atacadas pelo SonarQube

7. Referências

ABES (Associação Brasileira de Empresas de Software). "Mercado Brasileiro de Software – Panorama e Tendência 2019"

Alves, N. C. (2019). “Codemaster: Um Modelo de Avaliação do Pensamento Computacional na Educação Básica Através da Análise de Código de Linguagem de Programação Visual”.

Barros, Bruno. Evoluindo o design e arquitetura através das métricas do sonar. **MUNDOJ.** n. p.14-29 - 09/2010. Disponível: http://www.univale.com.br/unisite/mundo-j/artigos/43sonar.pdf

Chelf, B. and Chou, A. (2007). The next generation of static analysis.

Gomes. (2007). An overview on the Static Code Analysis approach in SoftwareDevelopment

Gomes. and André Luis. (2009). Qualidade de código com SonarQube - Métricas.

Lemos, W. L.; Cunha, J. A. O.; Saraiva, J. A. (2019). “Ensino de Engenharia de Software em um Curso de Sistemas de Informação: Uma Análise dos Problemas e Soluções na Perspectiva de Professores e Alunos”.

Lesgold, A. M. “The nature and methods of learning by doing”. American Psychologist, American Psychological Association, Washington, v. 56, n. 11, p. 964 - 973, 2001.

Martin Fowler. (1999). Refactoring: Improving the Design of Existing Code. AddisonWesley Professional, 1st edition.

Pressman, R.S. (2001). “Software Engineering: A Practitioner’s Approach”, Fifth Edition, McGraw Hill.

Raabe, A.; Jesus, E.; Hodecker, A.; Pelz, F. (2015). Avaliação do Feedback Gerado Por Um Corretor Automático de Algoritmos.

R. Terra. and R. S. Bigonha. (2008). Ferramentas para Análise Estática de Códigos Java.

Schneider, G.; Jaques, P. A.; “Combinando técnicas de análise estática e avaliação dinâmica para avaliação de código em ambientes de aprendizagem de programação”

Sinthsirimana, S.; Panjaburee, P. “A Development of Computer Programming Analyzer: A Case Study on PHP Programming Language”. Journal of Industrial and Intelligent Information, v. 1, n. 3, 2013”.

Sommerville. (2007). Engenharia de Software. Pearson Addison-Wesley, São Paulo, 8 edition, caps. 22–23, p. 341–373.

SonarQube. Disponível em: <https://docs.sonarqube.org/latest/> Acesso em: 22 ago. 2020.

Yulianto, S. V.; Liem, I. “Automatic Grader for Programming Assignment Using Source Code Analyzer”. International Conference on Data and Software Engineering, Bandung, Indonesia 2014.

Apêndice A - Roteiro da entrevista com os alunos

Apresentação:

- Auto-apresentação
- Agradecimento ao participante

Objetivo da Entrevista:
O objetivo desta pesquisa é avaliar a experiência e possíveis pontos positivos e negativos relacionados à experiência do aluno com a ferramenta SonarQube.

P1. Fale um pouco de você: sua formação, idade, experiência com programação.

Sondagem: Como foi a experiência?

P2. Você possui experiências anteriores com ferramentas de análise de código?

Sondagem: Se sim, poderia descrever em linhas gerais?

P3. Existe/Existiu algo que dificultou/impediu seu processo de aprendizado relacionado às disciplinas de programação?

Sondagem: Se sim, poderia descrever?

P4. [Opinião] O que a sua universidade oferece ou faz para estimular o processo de aprendizado?

P5. [Opinião] Quais funcionalidades demonstradas/utilizadas você não considera importantes para o processo de aprendizado e/ou melhoramento da sua codificação atualmente.

Sondagem: Por que não a(s) considera importante(s)?

P6. [Opinião] Quais funcionalidades demonstradas/utilizadas você considera importantes para o processo de aprendizado e/ou melhoramento da sua codificação atualmente?

P7. Qual a possibilidade de você utilizar/indicar a ferramenta para o seu ambiente de trabalho e/ou estudos?

Apêndice B - Termo de consentimento

Dados de identificação

Título do Projeto: **Avaliação da Ferramenta SonarQube para a Melhoria da Qualidade de Software por Desenvolvedores Durante a Graduação.**
Pesquisador Responsável: **Guthyelvis dos Santos Xavier**

Você está sendo convidado (a) para participar, como voluntário, do projeto de pesquisa "*Avaliação da Ferramenta SonarQube para a Melhoria da Qualidade de Software por Desenvolvedores durante a graduação*", de responsabilidade do pesquisador Guthyelvis dos Santos Xavier.
Leia cuidadosamente o que segue e me pergunte sobre qualquer dúvida que você tiver. Após ser esclarecido (a) sobre as informações a seguir, no caso aceite fazer parte do estudo, assine ao final deste documento, que consta em duas vias. Uma via pertence a você e a outra ao pesquisador responsável. Em caso de recusa você não sofrerá nenhuma penalidade.

Declaro ter sido esclarecido sobre os seguintes pontos:

1. O objetivo deste trabalho é avaliar como a ferramenta SonarQube pode auxiliar o processo de capacitação de desenvolvedores que estão na graduação a melhorar a qualidade de um produto de software.

2. A minha participação nesta pesquisa consistirá em fornecer dados baseados na minha experiência com a ferramenta e como a mesma pode/poderia ter impactado a minha experiência com as disciplinas e/ou com os cursos abordados pela pesquisa.

3. Ao participar desse trabalho estarei contribuindo para a exploração de uso da ferramenta em questão de modo a propor uma metodologia que possa auxiliar e/ou melhorar a experiência dos alunos dos cursos em questão.

4. Não terei nenhuma despesa ao participar da pesquisa e poderei deixar de participar ou retirar meu consentimento a qualquer momento, sem precisar justificar, e não sofrerei nenhum prejuízo.

5. Fui informado e estou ciente que há nenhum valor econômico, a receber ou a pagar, por minha participação, no entanto, caso eu tenha qualquer despesa decorrente da participação na pesquisa, serei ressarcido.

6. Meu nome será mantido em sigilo, assegurando assim a minha privacidade, e se eu desejar terei livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo o que eu queira saber antes, durante e depois da minha participação.

7. Fui informado que os dados coletados serão utilizados, única e exclusivamente, para fins desta pesquisa, e que os resultados poderão ser publicados.

Eu, _____, declaro ter sido informado e concordo em participar, como voluntário, do projeto de pesquisa acima descrito.

Cidade, _____ de _____ de 2020.

_____
Assinatura do participante