Ferramentas de Medição de Software: Um Estudo Comparativo

Gustavo Vale, Hudson Borges, Eduardo Figueiredo, Clarindo Pádua

Laboratório de Engenharia de Software (LabSoft), Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil

{gustavo, hsborges, figueiredo, clarindo}@dcc.ufmg.br

Resumo. Métricas de software permitem medir, avaliar, controlar e melhorar produtos e processos de software. O suporte automatizado no formato de ferramentas para extração de métricas é crucial. Contudo, para que tais ferramentas sejam realmente efetivas, é necessário que elas atendam às necessidades do usuário/desenvolvedor. Visando identificar boas e más características de usabilidade, esse trabalho apresenta um experimento controlado com 31 sujeitos que utilizaram e avaliaram as ferramentas CodePro Analytix e Metrics. Os resultados demonstram que as duas ferramentas possuem boas características de usabilidade. Contudo, a estratégia de apresentação dos dados adotada por cada uma das ferramentas é favorável a determinadas atividades e, no geral, a ferramenta CodePro Analytix se saiu melhor na avaliação.

Palavras-chave: usabilidade, ferramentas para medição de software, experimento controlado.

1 Introdução

Métricas de software permitem medir, avaliar, controlar e melhorar produtos de software. Muitas pesquisas exploram o tópico de métricas de software, pois métricas ajudam no controle do processo de desenvolvimento e manutenção de software [1][6]. Por exemplo, métricas podem ser combinadas para identificar anomalias de código ou projeto, comumente conhecidas como *Bad Smells* [8]. Métricas também podem ser usadas para fornecer uma base para critérios de aceitação entre o responsável pela produção e o cliente no desenvolvimento de software [3]. Portanto, promover o uso de métricas é essencial para assegurar ou estimar a qualidade de um software. Contudo, a extração de tais métricas sem apoio ferramental pode ser uma tarefa árdua. Sendo assim, ferramentas para medição de software, como Metrics e CodePro Analytix, foram desenvolvidas com o objetivo de facilitar o processo de extração e visualização de métricas de códigos orientados a objetos.

Muitas vezes ferramentas desenvolvidas para tecnologias comuns no ambiente industrial, como Programação Orientada a Objetos (POO), não são capazes de medir adequadamente métricas em códigos desenvolvidos em outras tecnologias, tais como Programação Orientada a Aspectos (POA) [11] e Programação Orientada a Características (POC) [2]. Considerando que técnicas com alta modularidade estão

ganhando mais espaço na indústria [5], ferramentas para auxiliar na medição dessas tecnologias são necessárias. Baseado nesta tendência, este trabalho analisou a usabilidade de duas ferramentas de medição de código com objetivo de identificar boas e más características de usabilidade presentes atualmente nessas ferramentas.

Os resultados dessa análise visam oferecer uma referência *(baseline)* para evolução e/ou criação de ferramentas dessa natureza. Um experimento controlado foi realizado com 31 alunos de graduação e pós-graduação em Ciência da Computação. Neste experimento, a usabilidade das ferramentas CodePro Analytix e Metrics foi avaliada. O experimento foi dividido em três fases e os resultados demonstram que ambas as ferramentas possuem aspectos de usabilidade que dependem do contexto de uso, mas a ferramenta CodePro Analytix, de acordo com os critérios definidos, obteve melhor resultado na comparação.

O restante deste artigo esta dividido da seguinte forma. Na Seção 2 são apresentados alguns trabalhos relacionados. A metodologia adotada no experimento é apresentada na Seção 3. Os resultados obtidos são apresentados na Seção 4. Em seguida, na Seção 5 são discutidos alguns aspectos de usabilidade das ferramentas avaliadas. Na Seção 6 as ameaças à validade do trabalho realizado são brevemente descritas. Por fim, as conclusões e trabalhos futuros são descritos na Seção 7.

2 Trabalhos Relacionados

No melhor de nosso conhecimento, não foram encontrados estudos que avaliam e/ou apresentem boas características de usabilidade para ferramentas de medição de software a ambientes de desenvolvimento integrados (IDE - *Integrated Development Environment*) como o Eclipse[1]. Portanto, os trabalhos relacionados apresentados neste trabalho estão relacionados ao ambiente de desenvolvimento Eclipse.

Chen e Max reportam sua experiência de uso do Eclipse em um curso de programação em Java [4]. Eles observaram que a maioria dos problemas enfrentados pelos participantes estão relacionados à falta de habilidade em resolver problemas de programação e não propriamente à usabilidade da IDE.

Storey *et. al.*, apresentam uma ferramenta chamada Gild que tem por objetivo oferecer um simples e poderoso ambiente de desenvolvimento para ajudar estudantes e instrutores a partir de visões da IDE Eclipse [15]. Essa ferramenta foi implementada na forma de *plug-in* e foi construída a partir da análise de necessidades de estudantes e instrutores para a linguagem de programação Java. Além disso, os autores realizaram um experimento e os resultados indicaram que a ferramenta conseguiu prover uma interface mais fácil de usar que a tradicional do Eclipse.

Por fim, Mealy *et. al.*, identificaram 81 requisitos de usabilidade para ferramentas de refatoração de software a partir da análise das tarefas definidas na ISO 9241-11 [12]. A partir disso, os autores analisaram o Eclipse para verificar se os requisitos identificados estavam implementados. Apesar do Eclipse não implementar todos os requisitos, este ambiente apresentou melhores resultados na comparação com outras IDEs.

[1] https://eclipse.org/

3 Metodologia

Com objetivo de identificar boas e más características de usabilidade das ferramentas de medição de código atuais, um experimento controlado foi conduzido seguindo as etapas propostas por Wohlin *et al.* [16]. Mais especificamente, esse experimento objetivou identificar qual ferramenta apresenta melhor usabilidade. As hipóteses H0 (nula) e H1(alternativa) são descritas da seguinte forma:

H0: Usabilidade CodeProAnalytix > Usabilidade Metrics

H1: Usabilidade CodeProAnalytix ≤ Usabilidade Metrics

A variável dependente que se deseja medir é usabilidade. Foram utilizados cinco atributos como variáveis independentes, sendo eles: produtividade do usuário; facilidade de aprendizado; retenção do aprendizado com uso intermitente; prevenção de erros do usuário; e, satisfação (Atributos de Nielsen [13]). Esses atributos são avaliados conforme descrito na Seção 3.3.

3.1 Sistema Alvo

Duas ferramentas para medição de código fonte orientado a objetos foram escolhidas, sendo elas, CodePro Analytix[2] e Metrics[3]; ambas são *plug-ins* populares para a IDE Eclipse. O CodePro Analytix é uma ferramenta gratuita desenvolvida pela Google e, além da função medição, diversas outras funções como audição de código e identificação de código clonado estão disponíveis para uso. O Metrics é uma ferramenta gratuita e de código aberto, desenvolvido em ambiente acadêmico e está na versão 1.3.8. Além de medição, outras funções também estão disponíveis, tal como análise de dependências.

3.2 Configuração do Experimento

O experimento iniciou-se com 36 sujeitos que são alunos de graduação ou pós-graduação em Ciência da Computação pela Universidade Federal de Minas Gerais (UFMG). Duas sessões, com mesmo formato, foram realizadas no decorrer do experimento em diferentes datas. A primeira sessão foi realizada com três sujeitos (experimento piloto). O principal objetivo do experimento piloto era identificar problemas na aplicação para corrigi-los na sessão seguinte. A segunda sessão foi realizada com 33 sujeitos; no entanto, dois deles deixaram alguns campos chave em branco e seus dados tiveram que ser descartados. Assim, na análise dos resultados foram considerados os dados de 31 sujeitos. Cada sessão foi organizada em três fases sequenciais: (i) caracterização de conhecimentos prévios dos sujeitos do experimento; (ii) execução de tarefas pelos sujeitos do experimento; e, (iii) avaliação qualitativa das ferramentas.

Caracterização de conhecimentos prévios dos sujeitos do experimento – O objetivo dessa fase é conhecer os sujeitos do experimento e a experiência deles com as ferramentas de medição. Os sujeitos foram divididos em dois grupos, GA (sujeitos

[2] https://developers.google.com/java-dev-tools/download-codepro/
[3] http://metrics.sourceforge.net/

que utilizaram a ferramenta CodePro Analytix) e GM (sujeitos que utilizaram a ferramenta Metrics). À medida que os sujeitos terminavam a primeira fase, os aplicadores do experimento definiam qual grupo cada sujeito iria integrar de forma a manter um equilíbrio no número de sujeitos em ambos os grupos. Como a definição do grupo dos sujeitos é depende dos resultados da primeira fase, a descrição detalhada de como os sujeitos foram divididos está na Seção 4.1.

Execução de tarefas pelos sujeitos do experimento – Nessa fase do experimento os sujeitos receberam um formulário com dois conjuntos de tarefas. Os conjuntos de tarefas continham as mesmas tarefas para ambos os grupos (GA e GM). No primeiro conjunto, o sujeito deveria coletar o valor de uma métrica especifica para uma classe especifica. No segundo conjunto de tarefas, o sujeito deveria obter o valor de dez métricas para diferentes artefatos de código, como classes, pacotes ou projeto. Duas métricas (para auxiliar na avaliação da usabilidade) foram coletadas pelos sujeitos ao final de cada conjunto de tarefas: o tempo inicial e final (medido em segundos); e, o número de cliques (medido pela ferramenta OdoPlus[4]).

Avaliação qualitativa das ferramentas – Nessa fase os sujeitos receberam um formulário com perguntas, principalmente, voltadas para detectar satisfação. Além das perguntas para medir satisfação foram utilizados outros tipos de perguntas para colher informações sobre as expectativas dos sujeitos, como uso de determinadas métrica e forma de trabalho. No caso da forma de trabalho, o objetivo era verificar se os sujeitos preferem realizar o processo de medição junto com a atividade de desenvolvimento ou em uma atividade diferente própria para identificar anomalias.

3.3 Avaliação da Usabilidade

As normas ISO/IEC 9126 [10] e ISO/IEC 25000 [9] apresentam algumas sub-características do que se entende por usabilidade de produtos de software. No entanto, essas sub-características são colocadas em termos mais genéricos do que os cincos principais atributos de usabilidade propostos por Nielsen [13]. Sendo assim, por serem mais específicos e objetivos, neste trabalho, optou-se pelos cinco atributos de Nielsen como referência nas avaliações de usabilidade.

O método de avaliação utilizado faz uso de dados de desempenho e de satisfação dos usuários obtidos por meio da realização de tarefas envolvendo os produtos de software sob avaliação. Foram propostos dois conjuntos de tarefas similares, nos quais, o usuário participa e registra o valor de algumas métricas geradas pelas ferramentas em avaliação.

Os cinco atributos principais de Nielsen para avaliar a usabilidade são: (i) produtividade do usuário, (ii) facilidade de aprendizado, (iii) retenção do aprendizado com uso intermitente, (iv) prevenção de erros do usuário e (v) satisfação. Nesta avaliação, não foram atribuídos pesos aos atributos, uma vez que a importância de cada atributo depende do contexto específico de uso do produto de software pelos usuários. Dessa forma, deixa-se livre a interpretação dos resultados com qualquer viés em relação à importância relativa dos atributos de Nielsen.

[4] http://www.fridgesoft.de/odoplus.php/

Cada um dos cinco atributos é medido de uma forma diferente. De forma a evitar medidas absolutas, como o índice de manutenibilidade [7], propõe-se a comparação dos valores obtidos em cada atributo. Em uma comparação entre dois sistemas, compara-se cada atributo, determina-se em qual sistema cada um apresentou melhor resultado. O critério adotado para definição do sistema com melhor usabilidade foi o número de atributos no qual cada um apresentou melhores resultados: o sistema com um maior número de atributos melhor avaliados é considerado o sistema com maior usabilidade. A forma como os cinco atributos devem ser medidos é explicada a seguir.

Produtividade do Usuário – A produtividade do usuário é calculada pelo número de atividades corretas, dividido pelo tempo gasto na tarefa. Para esta medida valores maiores são melhores. Como todos os sujeitos do experimento realizam as mesmas tarefas, os valores que variam de sujeito a sujeito são o tempo gasto e a proporção de acerto por tarefa. Portanto, usuários que fizerem mais tarefas corretas em menos tempo possuem maior produtividade.

Facilidade de Aprendizado – A facilidade de aprendizado é medida por duas métricas: (i) diferença percentual da quantidade de sujeitos que fizeram uso da documentação do CodePro Analytix e do Metrics; e, (ii) diferença da média do número de cliques do primeiro e do segundo conjunto de tarefas para a mesma ferramenta.

O uso frequente da documentação pode indicar problemas de usabilidade, logo a ferramenta em que não é necessário consultar sua documentação tende a ser mais intuitiva que outras que necessitam ter sua documentação consultada a todo momento. No caso da métrica número de cliques, acredita-se que quando o usuário aprender a fazer uma tarefa ou fizer mais tarefas ele tende a fazer um caminho direto. No caso do item (i) quanto menor o número de sujeitos que utilizaram a documentação melhor. No caso do item (ii) quanto maior e positiva a diferença melhor, pois significa que os sujeitos na média por tarefas aprenderam a utilizar o sistema de forma mais eficiente no segundo conjunto de tarefas. Observe que são comparadas as diferenças entre o número de cliques dos sujeitos utilizando uma mesma ferramenta e não o número de cliques dos sujeitos utilizando ferramentas diferentes.

Retenção do Aprendizado com Uso Intermitente – Para medir esse atributo, é realizada uma comparação entre usuários com perfis e conhecimento parecidos que estão utilizando o sistema pela primeira vez (usuários novatos) e, usuários que não utilizam há um período considerado significativamente longo (usuários antigos). O prazo para se considerar uso intermitente (período considerado significativamente longo) pode variar, no entanto, neste trabalho, será utilizado o período de seis meses.

Este atributo é medido pelos três conjuntos de medidas descritos a seguir: (i) comparação entre as porcentagens do número de acertos de tarefas de usuários antigos e dos usuários novatos; (ii) diferença das médias de produtividade das tarefas dos usuários antigos e novatos; e (iii) diferença entre as médias do número de cliques das tarefas dos usuários antigos e novatos. Supõe-se que, nos três casos, se o usuário antigo obtiver uma diferença positiva em relação ao usuário novato, foi possível reter algum aprendizado com uso intermitente.

Prevenção de Erros do Usuário – Acredita-se que um bom método para avaliar atributos ou características de qualidade de produto de software, entre eles usabilidade, tem como resultado um grau, nível ou comparação das características

entre sistemas e, além disso, deve-se apontar em quais partes do sistema erros foram identificados para que eles possam ser corrigidos. Assim, o primeiro passo, voltado para prevenção de erros de usabilidade é obter os atributos em que o sistema apresenta fraquezas. O segundo passo é corrigir os erros identificados e reavaliar o sistema. A avaliação desse atributo no primeiro passo será feita pela comparação das tarefas realizadas pelos usuários. Os critérios utilizados neste trabalho para avaliar os erros de usabilidade cometidos pelos sujeitos são: (i) caso nenhum, um ou dois usuários errem a mesma questão, deve-se computar como uma falha ou esquecimento do usuário; (ii) se três ou mais usuários erram a mesma tarefa utilizando a mesma ferramenta, pressupõe-se que o problema é da ferramenta e, a causa desses erros (de usabilidade) deve ser investigada.

Satisfação – Esse atributo é talvez o mais subjetivo dos atributos de Nielsen, pois trata-se de algo qualitativo. Para medi-lo, escolheu-se um conjunto de perguntas. Segundo a norma ISO/IEC 25010 [9] satisfação pode ser medida por quatro sub-características, sendo elas: utilidade, confiança, prazer e conforto. Dessa forma cada sub-característica foi medida com uma ou mais perguntas baseadas em outro trabalho [14]. Cada sub-característica possui peso igual independente do número de perguntas. Assim, a ferramenta que possuir maiores índices (0-100) em mais sub-características será avaliada com maior satisfação.

As quatro sub-características relacionadas à satisfação foram medidas segundo os critérios a seguir. *Utilidade* – medida por pergunta única e binária: Você considera o sistema útil para sua finalidade? *Confiança* – medida por pergunta única e escalável em seis níveis: Você confia nos resultados retornados pelo sistema avaliado? *Prazer* – medido por perguntas similares e escaláveis em seis níveis, após realizar cada conjunto de tarefas e após todas as tarefas: Qual a dificuldade para realizar a tarefa? *Conforto* – medido por duas perguntas a primeira genérica e escalável em seis níveis e a segunda binária: Quão satisfeito o usuário ficou de utilizar o sistema? Você recomendaria o sistema?

4 Resultados

4.1 Conhecimentos Prévios dos Sujeitos do Experimento

Na Figura 1 é apresentada a sumarização do perfil dos 31 sujeitos do experimento. A maioria deles são alunos de graduação (26 – 84%) (Figura 1a). Em relação a IDE Eclipse, 12 sujeitos (39%) consideram que sabem utilizá-la bem, 13 sujeitos (42%) haviam utilizado e, somente 6 sujeitos (19%) declararam nunca ter utilizado (Figura 1b). O conhecimento sobre métricas de software dos sujeitos foi, em sua maioria, nenhum (12 sujeitos – 39%) ou baixo (13 sujeitos – 42%) (Figura 1c). Apesar de 28 sujeitos terem cursado ou estarem cursando a disciplina de Engenharia de Usabilidade 16 deles (52%) declararam ter baixo conhecimento em usabilidade de sistemas (Figura 1d), os restantes declararam ter médio conhecimento (48%). Por fim, 29% dos participantes declararam ter experiência profissional de mais de 3 anos, 23% experiência entre 1 e três anos, 13% tiveram menos de 1 ano de experiência e 35% nunca trabalharam na indústria (Figura 1e).

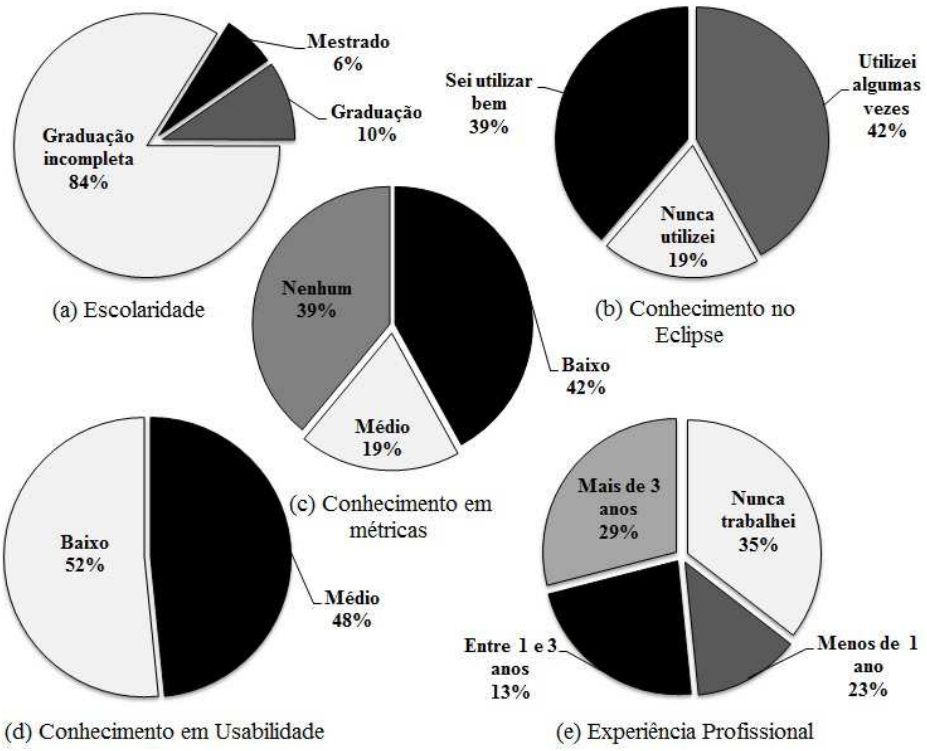

Figura 1. Perfil dos Sujeitos do Experimento

Além das características apresentadas na Figura 1 os sujeitos foram questionados sobre a utilização das ferramentas de medição sob avaliação nesta pesquisa, os *plug-ins* CodePro Analytix e Metrics (Figura 2a e 2b, respectivamente). Os resultados mostraram que a maioria dos sujeitos não conhecia nenhuma das duas ferramentas 84% e 74%, respectivamente para as Figuras 2a e 2b. Apenas um sujeito (3%) havia utilizado a ferramenta CodePro Analytix (Figura 2a). Quatro sujeitos (13%) já haviam utilizado a ferramenta Metrics (Figura 2b). O único sujeito que havia utilizado o CodePro Analytix também havia utilizado o Metrics.

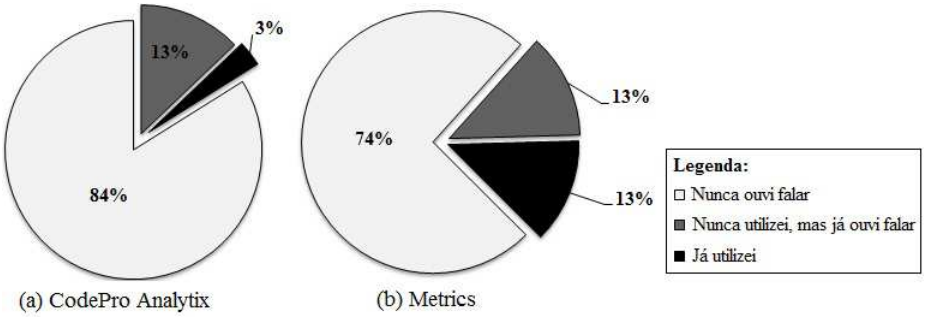

Figura 2. Conhecimentos das Ferramentas CodePro Analytix e Metrics

A divisão dos sujeitos nos grupos foi feita da seguinte forma:

- Dentre os quatro sujeitos com experiência prévia nas ferramentas, o sujeito com experiência com o CodePro Analytix foi escolhido para realizar o experimento com essa ferramenta, e além dele um dos outros três sujeitos que já havia utilizado o Metrics foi escolhido. Neste, caso o sujeito com maior nível de escolaridade. Esse critério foi utilizado para não prejudicar a ferramenta em sua análise, uma vez que os demais conhecimentos eram equivalentes entre os sujeitos que haviam utilizado o Metrics. Assim, os outros dois sujeitos com experiência prévia com o Metrics realizaram o experimento com essa ferramenta.
- Os demais sujeitos foram escolhidos aleatoriamente de forma a deixarem os grupos, GA e GM, homogêneos ou próximos disso.

Ao final da divisão o grupo GA (CodePro Analytix) continha 16 sujeitos e o grupo GM (Metrics) continha 15 sujeitos.

4.2 Análise da Usabilidade das Ferramentas

Nesta seção são apresentados os resultados da avaliação da usabilidade utilizando os cinco atributos de Nielsen.

Produtividade do Usuário – Como descrito na Seção 3.3, a produtividade dos usuários foi avaliada considerando a razão da quantidade de tarefas concluídas com sucesso (acertos) e o tempo gasto por cada sujeito (medidos em segundos). Na Tabela 1 está sumarizado o número de acertos, o tempo gasto e a produtividade total e média que os sujeitos tiveram com cada uma das duas ferramentas analisadas. Pode-se ver que o número de acertos com as ferramentas foram 154 e 143, para os grupos GA e GM, respectivamente. Mesmo o número de sujeitos de GM sendo menor que o número de sujeitos de GA (15 contra 16) – logo o número de acertos seria menor – a média do número acertos, proporcionalmente, também foi menor (9,625 para GA e 9,533 para GM). Em relação ao tempo total e médio gasto na realização das tarefas foram menores para os sujeitos de GA. A produtividade média foi maior para a ferramenta CodePro Analytix comparada à ferramenta Metrics, sendo 0,011779 e 0,007715, respectivamente.

Tabela 1: Produtividade Total por Ferramenta

GA	*Total*	*Média*
Acertos	154	9,625
Tempo	14884	930,25
Produtividade	0,188465	0,011779

GM	*Total*	*Média*
Acertos	143	9,533
Tempo	22073	1471,533
Produtividade	0,115722	0,007715

Facilidade de Aprendizado – Como descrito na Seção 3.3, partiu-se da premissa que ferramentas com objetivos semelhantes que exigem relativamente menos consultas às suas respectivas documentações tendem a ser de utilização mais fácil. Analisando o uso da documentação dos sujeitos, pode-se observar que dos sujeitos que usaram a ferramenta CodePro Analytix somente 50% (8 de 16) deles necessitaram consultar a documentação, enquanto 86,6% (13 de 15) dos sujeitos que realizaram o experimento com o Metrics necessitaram da documentação. Nenhum

sujeito utilizou documentação para realizar o segundo conjunto de tarefas. O Metrics teve um desempenho pior segundo este critério, pois mais sujeitos proporcionalmente utilizaram a documentação.

Os resultados do total e da média por sujeito do número de cliques do primeiro conjunto de tarefas para o segundo para cada ferramenta estão resumidos na Tabela 2. Em relação ao CodePro Analytix, para realizar o primeiro conjunto foram gastos 2.004 cliques enquanto para realizar o segundo conjunto de tarefas foram gastos 3.761 cliques, sendo uma média por sujeito de 125,25 e 23,51 cliques para realizar cada tarefa do primeiro e segundo conjunto de tarefas. Seguindo o mesmo raciocínio para realizar o primeiro e segundo conjunto de tarefas com o Metrics foram necessários 5.979 e 3.133 cliques. Isso representa uma média de 398,6 e 20,89 cliques para cada sujeito realizar uma tarefa do primeiro e segundo conjunto de tarefas, respectivamente. Em linhas gerais, foram necessários menos cliques para realizar todas as tarefas com o CodePro Analytix. No entanto, como deseja-se avaliar a facilidade de aprendizado, a diferença entre a média do número de cliques do primeiro conjunto de tarefas para o segundo conjunto de tarefas foi maior e positiva para a ferramenta Metrics (377,71 contra 101,74).

Tabela 2: Sumarização do Número de Cliques

CodePro Analytix (GA)	***Total***	***Média***
Conjunto 1	2.004	125,25
Conjunto 2	3.761	23,51
Diferença das médias		101,74

Metrics (GM)	***Total***	***Média***
Conjunto 1	5.979	398,60
Conjunto 2	3.133	20,89
Diferença das médias		377,71

Assim, os resultados sugerem que os sujeitos que realizaram o experimento com o Metrics tiveram dificuldade inicial maior, mas foram mais eficientes em relação ao número de cliques que os sujeitos que realizaram o experimento com o CodePro Analytix. Isso resulta em um empate na avaliação do atributo Facilidade de Aprendizado, CodePro Analytix melhor no primeiro conjunto de métricas e Metrics melhor no segundo.

Retenção do Aprendizado com Uso Intermitente – Como descrito na Seção 3.3, para avaliar este atributo, foram comparados os dados dos 4 sujeitos que haviam utilizado pelo menos uma das duas ferramentas há seis meses (usuários antigos) com os dados dos sujeitos que nunca haviam utilizado as ferramentas (usuários novatos). Três itens compõe a comparação: (i) diferença das proporções (%) do número de acertos do total de tarefas; (ii) a diferença das médias da produtividade; e, (iii) as diferenças entre as médias do número de cliques de todas as tarefas. Em todos os três casos compara-se o segundo conjunto de tarefas com o primeiro.

Os sujeitos antigos tiveram 8 e 11 acertos para o CodePro Analyix, 9 e 10 para o Metrics. Isso equivale a uma taxa de acerto média de 86,36% para ambas as ferramentas. Os demais sujeitos tiveram uma taxa de acerto de 87,36% para o CodePro Analytix e 86,71% para o Metrics. Nenhuma das ferramentas se sobressaiu, uma vez que os sujeitos que já haviam usado as ferramentas tiveram um desempenho levemente inferior aos sujeitos que não haviam usado em seus respectivos grupos.

Os sujeitos que haviam utilizado o CodePro Analytix previamente tiveram produtividade média de 0,014833; os demais sujeitos que utilizaram a mesma ferramenta tiveram uma produtividade média de 0,011343. Logo, os sujeitos que

haviam utilizado o CodePro Analytix previamente foram mais produtivos que os sujeitos de primeira viagem. No caso do Metrics, os sujeitos que haviam utilizado essa ferramenta tiveram uma produtividade média de 0,006365 e os demais sujeitos que utilizaram essa mesma ferramenta obtiveram uma produtividade de 0,007984. Assim, os sujeitos que utilizaram a ferramenta previamente tiveram um desempenho pior em questão de produtividade que os sujeitos noviços para o Metrics. Assim, nesse segundo item (ii) o CodePro Analytix foi melhor;

Os sujeitos que haviam utilizado o CodePro Analytix realizaram 142 cliques na média para realizar todas as 11 tarefas, os demais sujeitos realizaram 391,5 cliques na média para realizar as mesmas 11 tarefas. No caso do Metrics, os sujeitos que haviam utilizado previamente essa ferramenta precisaram de 83 cliques na média, os demais sujeitos necessitaram de 688,15 cliques na média. Com isso, conclui-se que os sujeitos de ambas as ferramentas que utilizaram uma delas previamente obtiveram melhores resultados que os sujeitos que estavam usando pela primeira vez. No entanto, quando comparado as diferenças entre o número de cliques dos sujeitos que utilizaram pela primeira vez com os sujeitos que utilizaram as ferramentas previamente (249,5 para o CodePro Analytix e 605,15 para o Metrics) a diferença do Metrics é bem mais expressiva que o da outra ferramenta. Assim, o Metrics obteve melhor desempenho nesse item (iii). Em resumo, no item (i) nenhuma das ferramentas se sobressaiu e nos itens (ii) e (iii) os sujeitos de cada uma das duas ferramentas obtiveram melhores resultados. Assim, nenhuma ferramenta se sobressaiu na avaliação deste atributo (Retenção do Aprendizado com Uso Intermitente).

Prevenção de Erros do Usuário – Medir esse atributo em um experimento único é muito difícil por causa dos dois passos descritos na Seção 3.3, assim o foco deste trabalho foi identificar quais erros foram cometidos por falhas dos sujeitos e quais foram cometidos por problemas de usabilidade das ferramentas (passo 1). Na Tabela 3 é apresentada a quantidade de erros nas atividades realizadas pelos sujeitos do experimento, ressaltando que as atividades de 2 a 11 são referentes ao segundo conjunto de tarefas.

Tabela 3: Quantidade de Erros por Atividade

Conjunto de tarefas	**1**	**2**									
Tarefa	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
GA	2	9	2	0	0	0	3	1	1	0	3
GM	6	1	4	0	0	0	1	1	0	2	6

As atividades em que os usuários cometeram mais de dois erros foram consideradas como problemas de usabilidade (Seção 3.3). Portanto, as seguintes observações podem ser feitas: (i) a ferramenta CodePro Analytix apresentou problemas (erros) de usabilidade nas atividades 2, 7 e 11. Com destaque para a atividade 2 na qual 9 sujeitos (56%) falharam na execução e coletaram valores errados. As atividades 7 e 11 tiveram uma taxa de erro levemente superior ao mínimo definido, no qual apenas três sujeitos erraram (18,7%); (ii) a ferramenta Metrics apresentou problemas (erros) de usabilidade nas atividades 1, 3 e 11. As atividades 1 e 11 tiveram uma taxa de erro de 40%, valor pouco abaixo do máximo registrado pela CodePro Analytix. A atividade 3 teve uma taxa de erro de 26,6%.

De forma ampla, ambas as ferramentas apresentaram o mesmo número de erros de usabilidade segundo os critérios definidos. A principal diferença é que os

problemas de usabilidade identificados no CodePro Analytix tiverem uma maior concentração em uma atividade, enquanto os problemas de usabilidade do Metrics foram mais distribuídos. Acredita-se que o fato dos artefatos de código (classes e métodos) não estarem em ordem alfabética pode ter contribuído com alguns erros dos usuários na ferramenta Metrics. O nome parecido de algumas classes do sistema medido também podem ter instigado erros dos sujeitos, por exemplo, os nove sujeitos que erraram a tarefa 2 na ferramenta CodePro Analytix responderam igual. Os 4 e 6 sujeitos que erraram as tarefas 3 e 11 também colocaram as mesmas respostas para as respectivas questões correspondendo a erros de usabilidade.

O número total de erros ou falhas cometidos pelos sujeitos é igual a 21 para as duas ferramentas; no entanto, o número de sujeitos que realizaram o experimento com o CodePro Analytix é maior (16 sujeitos contra 15 do Metrics), talvez se mais um sujeito realizasse o experimento com o Metrics igualando o número de sujeitos que realizaram o experimento no CodePro Analytix, este sujeito cometesse algum erro. Logo, os sujeitos teriam cometidos mais erros com o Metrics. Além disso, dentre erros que mais se repetiram (erros de usabilidade) o número de erros que os sujeitos cometeram com Metrics é levemente superior aos do CodePro Analytix (16 contra 15 erros). Dessa forma, o CodePro Analytix será considerado melhor neste atributo.

Satisfação – Este atributo foi medido por um conjunto de perguntas, que inicialmente avaliaram quatro sub-características para então avaliar o atributo satisfação. As perguntas estão descritas na Seção 3.3, aqui são apresentados apenas os resultados provindos das respostas dos sujeitos do experimento para cada uma das quatro sub-características e, ao final, o resultado da avaliação do atributo satisfação. Este atributo foi medido por quatro sub-características: (i) *Utilidade*: 100% dos sujeitos disseram que acham ambas as ferramentas úteis; (ii) *Confiança*: As ferramentas receberam índices de confiança equivalentes a 78,75% e 81,33%, neste caso o Metrics foi melhor; (iii) *Prazer*: Em ambas as ferramentas o nível de dificuldade foi baixo, sendo 32,08% e 43,56% para o CodePro Analytix e Metrics, respectivamente; neste caso, foi mais prazeroso o uso da ferramenta CodePro Analytix, pois os sujeitos tiveram menor dificuldade em sua utilização; (iv) *Conforto*: Essa sub-característica também foi avaliada de maneira similar para as duas ferramentas, por exemplo, todos os sujeitos (100%) recomendaram o uso da ferramenta. Os índices finais dessa sub-característica são 83,13% e 82,67% para as ferramentas CodePro Analytix e Metrics, respectivamente, neste caso o CodePro Analytix foi melhor.

Esse atributo obteve resultados semelhantes para ambas às ferramentas, no entanto, a ferramenta CodePro Analytix foi melhor em duas sub-características (Prazer e Conforto); igual em uma sub-característica (Utilidade). Assim, o CodePro Analytix obteve melhores resultados no atributo Satisfação.

5 Discussão

Nessa seção são apresentadas algumas reflexões com o intuito de destacar aspectos de usabilidade ou não que tornaram as ferramentas melhores ou piores de acordo com os resultados do experimento. Contudo, os aspectos a serem discutidos não devem ser

considerados como pré-requisitos para novas ferramentas, mas sim como um guia de usabilidade para ferramentas dessa categoria.

O primeiro aspecto considerado, neste trabalho, consiste na forma como ambas as ferramentas adotam para a coleta e apresentação das métricas. Na ferramenta CodePro Analytix é implementada uma interface de apresentação mais detalhada, fazendo uso de *menus* de rápido acesso e gráficos que acompanham tabelas de dados. Já na ferramenta Metrics é implementada uma interface de apresentação minimalista cujo objetivo é apresentar somente o mínimo necessário (tabelas de dados).

Nesse sentido de coleta a apresentação das métricas foi observado que interfaces com mais detalhes apresentaram um nível de usabilidade melhor para usuários que fazem uso rápido para coleta de poucas métricas. Ao contrario de interfaces minimalistas que favorecem usuários que necessitam coletar muitas métricas de uma única vez. Além disso, o experimento também mostrou que a interface com mais detalhes (da ferramenta CodePro Analytix) exigiu que menos sujeitos acessassem a documentação, indicando que ela é mais intuitiva que a interface do Metrics.

Outro aspecto importante analisado refere-se à facilidade de acesso aos dados. O CodePro Analytix oferece uma opção de acesso mais rápida e intuitiva a partir de uma opção no menu que aparece quando o usuário clicar com o botão direito sobre a pasta do projeto na visão (*package explorer*) padrão do Eclipse. Desta forma o usuário pode facilmente optar por calcular as métricas do projeto selecionado e uma visão é apresentada com os resultados. Por outro lado, a ferramenta Metrics exige que o usuário execute dois passos para habilitar o uso do *plug-in*: (i) habilitar o Metrics nas propriedades do projeto; e, (ii) deixar a visão do Metrics visível no Eclipse para posteriormente selecionar o artefato do projeto (habilitado anteriormente). Esse ponto foi observado principalmente no primeiro conjunto de tarefas, no qual os sujeitos que utilizaram o CodePro Analytix tiveram um tempo bastante inferior aos sujeitos que usaram o Metrics.

Ao final do experimento, na fase de avaliação qualitativa das ferramentas, os sujeitos receberam quatro perguntas: (i) sobre a relevância das métricas que podem ser coletadas pelas ferramentas; (ii) quais métricas os sujeitos sentiram falta; (iii) o que o sujeito mais e menos gostou na ferramenta; e, (iv) se o sujeito acha relevante destacar valores discrepantes de métricas e, caso positivo, se ele prefere que esses valores sejam destacados logo na atividade de desenvolvimento, ou seja, a medida que o código é desenvolvido ou; em uma atividade separada, como por exemplo, uma atividade para medir e refatorar o código.

Sobre a relevância das métricas todos os sujeitos informaram que as métricas são relevantes (i). 28 dos 31 sujeitos informaram que faltam algumas métricas, no entanto, a grande maioria dos sujeitos não informou quais métricas. Os sujeitos que informaram foram generalistas dizendo apenas métricas tradicionais e métricas orientadas a aspectos (ii). Entre os pontos que os sujeitos mais e menos gostaram na ferramenta estão, por exemplo, a interface atraente das ferramentas que plotam gráficos instantaneamente, principalmente, para a ferramenta CodePro Analyitx e como pontos que os sujeitos menos gostaram está em ter que habilitar o *plug-in* (Metrics) depois de instalado (iii). Quanto a última questão (iv) 100% dos usuários acham interessante destacar valores discrepantes de alguma forma, sendo que 19 dos 31 sujeitos (61,29%) acham que os valores discrepantes devem ser destacados logo na

atividade de desenvolvimento, contra 12 sujeitos (38,7%) acham que os valores discrepantes devem ser destacados em uma atividade separada.

Dado que na comparação realizada os sujeitos que realizaram o experimento com a ferramenta CodePro Analytix obtiveram melhores resultados para os três atributos de usabilidade de Nielsen, Produtividade, Prevenção de Erros do Usuário e Satisfação). Nos outros dois atributos (Facilidade de Aprendizado e Retenção do Aprendizado com Uso Intermitente) as ferramentas tiveram desempenho equivalente. Além disso, com a discussão realizada nesta seção, conclui-se que a ferramenta CodePro Analytix possui maior usabilidade que a ferramenta Metrics aceitando a hipótese H0 e rejeitando a hipótese H1.

6 Ameaças a Validade

Mesmo com o planejamento minucioso deste trabalho, alguns fatores devem ser considerados na validação dos resultados e discussões. O experimento foi realizado com estudantes em uma classe e os sujeitos podem não representar corretamente uma companhia. Os sujeitos realizaram o experimento individualmente, assim, eles podem ter obtido um desempenho melhor devido à experiência prévia. Caso, sujeitos experientes se unissem aos menos experientes o conhecimento se nivelaria. No entanto, o número de avaliações seria menor podendo limitar o escopo do trabalho.

Tentou-se medir a experiência dos sujeitos, no entanto, podemos não ter obtido uma resposta real, pois, geralmente, os sujeitos dizem serem menos experientes do que realmente são. O número de sujeitos limita os resultados, talvez se o número de sujeitos fosse maior ou o ambiente fosse outro os resultados poderiam ser diferentes. As conclusões são válidas para um grupo de sujeitos e para as ferramentas avaliadas. Alguns sujeitos podem gostar mais da outra ferramenta e não realizar um bom experimento para prejudicar a ferramenta que ele está avaliando.

7 Conclusões e Trabalhos Futuros

A avaliação da usabilidade de sistemas de software, na maioria das vezes, é dependente de métricas e de como as métricas são combinadas para expressar pontos coerentes no processo de medição e avaliação. A usabilidade e as diversas formas de avaliá-la têm sido estudadas em diversos trabalhos. No entanto, muitas vezes os métodos são custosos, genéricos ou estão distantes da maioria dos pesquisadores e empresários. Para tornar o processo de medição de software mais próximo de estudantes e pesquisadores, novas ferramentas têm sido desenvolvidas e a demanda por novas tecnologias tem forçado o desenvolvimento de novas ferramentas.

Este trabalho apresenta um experimento controlado que teve como objetivo: (i) avaliar a usabilidade de duas ferramentas de extração de métricas de código, e (ii) apresentar boas e más características de usabilidade presentes em tais ferramentas. Foi constatado que a ferramenta CodePro Analytix possui maior usabilidade; que as métricas implementadas nestas ferramentas são importantes na avaliação de software; e, que os sujeitos preferem que valores discrepantes sejam destacados logo na implementação do sistema. Espera-se que os resultados e a metodologia apresentados

neste artigo sejam usados como referência (*baseline*) para novas ferramentas e comparações desse tipo. Além disso, esse trabalho também contribui com um método alternativo de avaliação e comparação de sistemas de modo geral, não só ferramentas.

Como trabalhos futuros pretende-se generalizar o método proposto neste trabalho, realizar uma comparação com outros métodos para avaliar a usabilidade de sistemas e realizar uma comparação com ferramentas que medem código de outras tecnologias como programação orientada a características e aspectos.

Agradecimentos. Este trabalho recebeu recursos financeiros do CNPq (Processo 485907/2013-5), CAPES e FAPEMIG (Processos APQ-02532-12 e PPM-00382-14).

Referências Bibliográficas

1. Alves, T. L., Ypma, C., Visser, J.: Deriving metric thresholds from benchmark data. In: Software Maintenance (ICSM), 2010 IEEE Int. Conf. on. Pp. 1-10. IEEE (2010).
2. D. Batory, S. O'Malley, The Design and Implementation of Hierarchial Software Systems with Reusable Components, ACM Transactions on Software Engineering and Methodology (TOSEM), 1(4), pp. 355-398, 1992.
3. Bevan, N.: Quality in use: Meeting user needs for quality. Journal of Systems and Software 49(1), pp. 89-96 (1999).
4. Chen, Z., Marx, D.: Experiences with Eclipse IDE in programming courses. Journal of Computing Sciences in Colleges 21(2), 104-112 (Dec 2005).
5. Ferreira G. C., Gaia, F. N., Figueiredo, E., Maia, M. A.: On the Use of Feature-Oriented Programming for Evolving Software Product Lines – A Comparative Study, Science of Computer Programming, pp. 65-85, 2014.
6. Ferreira,K. A., Bigonha, M. A., Bigonha, R. S., Mendes, L. F., Almeida, H. C.: Identifying thresholds for object-oriented software metrics. Jour. Sof. Systm., 85(2), 244-257 (2012).
7. Foreman, J., Brune, K., McMillan, P., Rosenstein, R.: Software technology reference guide-a prototype (cmu/sei-97-hb-001, ada 305472). Pittsburgh, pa: SEI (1997).
8. Fowler, M.: Refactoring: improving the design of existing code. Pear. Educ. India (1999).
9. ISO/IEC 25000 – Software Engineering – Software Product Requirements and Evaluation (SQuaRE) – Guide to SQuaRE. International Standards Organization. Tech. rep. (2005).
10. ISO/IEC 9126: Software Product Evaluation – Quality Characteristics and Guidelines for the User, Geneva, International Organization for Standardization. Tech. rep. (2001).
11. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier J. M., Irwin, J.: Aspect-oriented programming. In: 11th Eur. Conf. on Obj.O. Prog., pp.220-242, 1997.
12. Mealy, E., Carrington, D., Strooper, P., Wyeth, P.: Improving Usability of Software Refactoring Tools. In: 2007 Australian Software Engineering Conference (ASWEC'07). pp. 307 318. IEEE (Apr 2007), http://ieeexplore.iee.org/articleDetails.jsp?arnumber=4159683.
13. Nielsen, J.: Usability Engineering. Elsevier (1994).
14. Padilha, J., Pereira, J., Figueiredo, E., Almeida, J., Garcia, A., Sant'Anna, C.: On the effectveness of concern metrics to detect code smells: An empirical study. In: Advanced Information Systems Engineering. pp.656-671. Springer (2014).
15. Storey, M. A., Damian, D., Michaud, J., Myers, D., Mindel, M., German, D., Sanseverino, M., Hargreaves, E.: Improving the usability of Eclipse for novice programmers. In: Proceedings of the 2003 OOPSLA workshop on eclipse technology eXchange – eclipse'03. pp. 35-39. ACM Press, New York, USA (Oct 2003).
16. Wohlin, C., Runeson, P., Host, M., Ohlsson, M. C., Regnell, B., Wesslén, A.: Experimentation in software engineering. Springer (2012).