X Simpósio Brasileiro de Sistemas de Informação
27 a 30 de Maio de 2014
Londrina - PR - Brasil
SBSI 2014
Ecossistemas Digitais e Estratégicos

Anexo: Mapeamento sistemático do artigo: "*GiveMe Metrics* - Um Framework Conceitual para Extração de Dados Históricos sobre Evolução de Software".

Sumário

Mapeamento Sistemático

O mapeamento sistemático é um mecanismo que permite ao pesquisador fornecer uma visão geral sobre a área de pesquisa que está investigando, levando em conta fatores como a quantidade de pesquisas publicadas sobre a área pesquisada, e os tipos de pesquisa disponíveis [Steinmacher e Gerosa, 2012]. Já um protocolo de mapeamento é o método que o pesquisador define na formalização de buscas sobre a área de pesquisa alvo de investigação [Kitchenham, 2004].

Planejamento do Mapeamento Sistemático.

O protocolo desenvolvido foi constituído de objetivo, critérios para avaliação das ferramentas, questões de pesquisa, *string* de busca e bases de dados de publicações. Detalhamos a seguir cada um deles.

Objetivo: levantar quais são as ferramentas capazes de reportar dados sobre evolução de software, provenientes de três categorias de repositórios: Categoria 1 - Repositórios de Código Fonte; Categoria 2 - Repositórios de Defeitos de Software; Categoria 3 - Repositórios de Dados sobre o Processo de Desenvolvimento de Software.

Critérios para a avaliação e aceitação das ferramentas: foram definidos critérios para essa tarefa, estabelecidos com base em uma reunião com pesquisadores, tendo sido apresentadas as características dos repositórios (de código fonte, de defeitos e de processos de desenvolvimento de software), incluindo características utilizadas por outras instituições parceiras, além do SINAPAD. Assim, os critérios foram definidos a partir dos principais critérios levantados na literatura e revisados nessa reunião.

Cada categoria possui um conjunto específico de critérios. Ferramentas da Categoria 1 - Repositórios de Código Fonte, devem possuir as seguintes características:

- (C1): capacidade de se conectar a repositórios de código fonte SVN ou GitHub com suporte a SVN, mesmo sob autenticação;
- (C2): capacidade de analisar código Java ou C#;
- (C3): capacidade de extrair automaticamente um conjunto de métricas por cada *release* ou *commit*, ou por projeto de um software no repositório. Uma ferramenta será considerada válida se extrair pelo menos uma das métricas de cada uma das características de software definidas em [Araújo, 2009], que são Tamanho[1], Periodicidade[2] e Complexidade[3];

[1] Tamanho: caracterizado pela quantidade de artefatos produzidos em cada etapa do ciclo de vida do software e medido por uma das métricas a seguir: número de pontos de função, números de pontos de caso de uso, número de requisitos, número de classes, número de métodos por classe, número de classes de domínio, número de classes de suporte, número de subsistemas, número de linhas de código fonte.

[2] Periodicidade: representa o intervalo de tempo decorrido entre cada versão produzida de um artefato e medida pelo intervalo de tempo entre versões de um produto (diz respeito à versão do software a partir da qual as métricas são extraídas).

[3] Complexidade: identificada através de elementos que podem medir a complexidade estrutural de um artefato e medida por uma das métricas a seguir: número de casos de uso, número de diagramas de classes, número de diagramas de sequência, número de diagramas de estados, número de diagramas de empacotamento, número de diagramas de atividades, profundidade de herança por classe, número de filhos por classe, acoplamento entre objetos, resposta de uma classe, falta de coesão entre métodos, complexidade ciclomática por método.

- (C4): capacidade de exportar os resultados nos seguintes formatos: .xls, .csv, .txt, .xml ou gravá-los em um banco de dados.

Já as ferramentas da Categoria 2 - Repositórios de Defeitos de Software, devem possuir as seguintes características:

- (C1): capacidade de se conectar a repositórios de registro de defeitos, mesmo sob autenticação;
- (C2): capacidade de retornar automaticamente um conjunto de dados históricos por *release, commit* ou por projeto de um software no repositório. Devido à diversidade de tipos de dados que podem ser extraídos sobre um defeito, não será limitado aqui o tipo de defeito registrado. Entretanto, ferramentas que extraiam defeitos de código fonte devem considerar apenas softwares desenvolvidos em Java ou C#;
- (C3): capacidade de exportar os resultados obtidos nos seguintes formatos: .xls, .csv, .txt, .xml, .html ou gravá-los em um banco de dados.

Por fim, as ferramentas na Categoria 3 - Repositórios de Dados sobre o Processo de Desenvolvimento de Software, devem possuir as seguintes características:

- (C1): capacidade de se conectar a repositórios de registro de dados de processos de desenvolvimento de software, mesmo sob autenticação;
- (C2): capacidade de retornar um conjunto de dados históricos por *release, commit* ou projeto de um software no repositório. Uma ferramenta será considerada válida se retornar qualquer tipo de informação sobre o processo de desenvolvimento de software. Uma resalva está no fato de que, caso a ferramenta considere projetos de código para extrair informações, somente serão aceitas ferramentas que manipulem código Java ou C#;
- (C3): Capacidade de exportar os resultados obtidos nos seguintes formatos: .xls, .csv, .txt, .xml, .html ou gravá-los em um banco de dados.

Questões de Pesquisa: mediante o objetivo apresentado e os critérios para avaliação das ferramentas, foram definidas questões de pesquisa, com o objetivo de guiar a execução do mapeamento sistemático. As questões de pesquisa definidas são as seguintes:

- (Q1) Quais são as ferramentas disponíveis em cada uma das categorias de repositórios?
- (Q2) Em quais critérios pré-definidos para cada uma das categorias de repositórios as ferramentas se encaixam?

Strings de busca: para cada uma das categorias de repositórios, foram definidas as seguintes *strings* de busca:

- Categoria 1**:** ((ferramenta OR tool OR plugin) AND (extração OR extract OR mining OR mineração) AND (métricas OR métrica OR metric OR metrics) AND (repositório OR repositórios OR repository OR repositories) AND ((código AND fonte) OR (source AND code)))

- Categoria 2: ((ferramenta OR tool OR plugin) AND (extração OR extract OR mining OR mineração) AND (métricas OR métrica OR metric OR metrics) AND (repositório OR repositórios OR repository OR repositories) AND (defeitos OR defeito OR defect OR defects OR bug OR bugs OR "bug tracker"))

- Categoria 3:((ferramenta OR tool OR plugin) AND (extração OR extract OR mining OR mineração) AND (métricas OR métrica OR metric OR metrics) AND (repositório OR repositórios OR repository OR repositories) AND (software AND development AND process))

Bases de dados de publicações: para executar as *strings* de busca, foram consideradas cinco bases de dados eletrônicas de publicações com acesso livre para alunos da Universidade Federal de Juiz de Fora (UFJF). As bases usadas foram:

- Science@Direct (http://www.sciencedirect.com)
- El Compendex (http://www.engineeringvillage.com)
- IEEE Digital Library (http://ieeexplore.ieee.org)
- ISI Web of Science (www.isiknowledge.com)
- Scopus (http://www.scopus.com)

O protocolo do mapeamento sistemático definido nesta seção permite uma busca formal pelas ferramentas, diferentemente de uma abordagem de pesquisa *ad-hoc* que não possui uma metodologia rigorosa para obter resultados através de investigação [Kitchenham, 2004].

Execução do Mapeamento Sistemático.

Após a definição do protocolo do mapeamento sistemático as *strings* de busca definidas foram executadas nas bases de publicações e o retorno das buscas foram analisados. Adicionalmente, as mesmas strings foram executadas na máquina de busca do Google para recuperar ferramentas que porventura não tenham sido publicadas, como ferramentas proprietárias (disponíveis para download através de sites). Isto se torna necessário porque há ferramentas comerciais que não foram alvos de publicações, então não aparecerão em bases de dados de publicações. Além disso, a experiência dos pesquisadores que definiram os critérios para aceitação das ferramentas, mostra que há ferramentas que não foram alvos de publicação e que possuem as características desejadas e descritas nos critérios definidos.

As buscas ocorreram ao longo do mês de novembro de 2013 e os idiomas considerados foram Inglês e Português. Na primeira categoria, Repositórios de Código Fonte, foram obtidas 12 ferramentas, sendo que destas, apenas 2 foram aceitas em todos os critérios definidos no protocolo. Na segunda categoria, Repositórios de Defeitos de Software, foram obtidas 22 ferramentas, sendo que destas, apenas 7 foram aceitas em todos os critérios definidos no protocolo. Já na terceira e última categoria, Repositórios de Dados sobre o Processo de Desenvolvimento de Software, foram obtidas 5 ferramentas, com apenas 3 aceitas em todos os critérios.

Resultados.

As ferramentas obtidas foram avaliadas e catalogadas, objetivando responder a primeira questão de pesquisa definida no protocolo. Além de dividi-las por categoria de repositório de dados históricos, foi possível ainda identificar grupos de ferramentas em cada uma das categorias. Esta informação foi útil para auxiliar na definição do GMM. Para responder a segunda questão de pesquisa, as características de cada uma das ferramentas foram avaliadas, levando em consideração os critérios pré-estabelecidos para cada uma das categorias de repositórios de dados.

Resultados da questão de pesquisa (Q1).

As ferramentas levantadas na Categoria 1 são listadas a seguir. Os grupos identificados são (i) ferramentas que possuem interface via console para linha de comando e (ii) as que possuem interface gráfica do usuário (GUI).

Grupo (i):

- MetricMiner [Sokol, et. al, 2013] é uma ferramenta web de mineração de repositórios de código fonte. Com ela é possível acessar repositórios públicos de código fonte e extrair métricas de código como linhas de código e complexidade ciclomática, dentre outras. MetricMiner possui um banco de dados onde o usuário da ferramenta pode consultar via comandos SQL as métricas dos projetos minerados.
- Search Engine [Search Engine, 2013] é uma ferramenta proprietária, com capacidade de fornecer métricas de código extraídas de projetos desenvolvidos em linguagens como Java e C# (suporte a outras linguagens também disponíveis).
- Sourcerer [Sourcerer, 2013] permite a extração de métricas de código de projetos desenvolvidos na linguagem Java. A ferramenta em si é um projeto Java que pode ser executado pelo usuário. As métricas extraídas são armazenadas em um banco de dados MySQL.
- Kalibro Metrics [Morais, et. al, 2013] é uma ferramenta com versão desktop chamada Kalibro Desktop. Esta ferramenta trabalha de forma integrada com a ferramenta Analizo[Terceiro, et. al, 2013]. Com ela é possível extrair métricas de código fonte Java, C e C++. Há suporte para acesso a repositórios SVN, de onde é possível extrair um conjunto de métricas de código fonte.
- SourceMiner [SourceMiner, 2013] Torna possível extrair um conjunto de cinco métricas de código fonte. Ela está disponível como um plugin do Eclipse, onde fornece múltiplas visões sobre as métricas extraídas.
- EvoJava [Oosterman, et. al, 2011] se difere das demais ferramentas apresentadas até o momento por utilizar recursos semânticos para analisar o código fonte, exclusivamente de projetos desenvolvidos na linguagem Java, e retornar um conjunto de métricas que podem ser exportadas para o Microsoft Excel.
- EvolTrack [Werner, et. al, 2011] trata-se de um plugin para o ambiente de desenvolvimento Eclipse capaz de acessar repositórios SVN e criar visualizações Eclispe/UML2Tools sobre as análises realizadas no código. Ela possui uma característica interessante que permite a escolha de uma versão do projeto no repositório ou um range de versões para analisar, fornecendo uma barra de navegabilidade entre as versões selecionadas.
- MASU [Higo, et. al, 2011] foi desenvolvida para dar suporte a extração de métricas de código fonte Java, C# e VB, entretanto o suporte a extração de métricas C# ainda está em desenvolvimento. Esta ferramenta não permite o acesso a repositórios como SVN, o que inviabiliza a extração automática de métricas por cada uma das versões do software presente no repositório de código.
- MinerAll [Silva, et. al, 2013] é framework criado para suportar a extração de artefatos de software presentes em um repositório de dados, para fins de reuso, e a mineração de informações provenientes da análise dos artefatos também armazenados no repositório. A mineração de informações contará com a extração de

métricas de código através da utilização de técnicas de *Data Mining* como associação, classificação, clusterização e regressão.

- SO-MSR [Matsumoto and Nakamura, 2011] é um framework para mineração de repositórios de dados capaz de extrair, dentre outras coisas, métricas de código fonte em repositórios CVS (em desenvolvimento) e Subversion. Segundo [Matsumoto and Nakamura, 2011] é possível extrair métricas de qualquer tipo de linguagem de programação.

Grupo (ii):

- Analizo [Terceiro, et. al, 2013] diferentemente de outras ferramentas citadas até o momento, como MetricMiner, Epona e Sourcerer, esta permite a análise de códigos fontes desenvolvidos nas linguagens Java, C e C++. Seu suporte a conexões com repositórios SVN e o grande número de métricas que ela permite extrair, podem ser explorados somente em ambientes Debian Linux, via linha de comando.
- Epona [Junior and Juvenal, 2012] é uma ferramenta que permite o *download*, para uma pasta local, do projeto escolhido diretamente em repositórios de código fonte como *GitHub*[GitHub, 2013]. Com ela é possível a extração de métricas de código fonte exclusivamente de projetos Java.

Na Categoria 2 foram detectados cinco diferentes grupos de ferramentas. O primeiro deles é o grupo das (i) capazes de detectar defeitos em código fonte e reportá-los. O segundo grupo é o (ii) das ferramentas de gerenciamento de defeitos, que também permitem reportar dados sobre os defeitos armazenados. O terceiro grupo engloba (iii) as ferramentas para automação de *build*, mas que possuem recursos que permitem a identificação e detalhamento dos defeitos encontrados. Já o quarto conjunto, que possui apenas uma ferramenta, é o das (iv) capazes de analisar dados de processos de software e reportar os defeitos detectados. O quinto grupo possui as ferramentas (v) capazes de minerar bases de dados de defeitos e retornar informações. A seguir são apresentadas as ferramentas levantadas.

Grupo (i):

- FindBugs [FindBugs, 2013] é uma ferramenta *free* que permite a detecção de defeitos em código fonte de projetos desenvolvidos na linguagem Java. Com ela é possível categorizar os defeitos detectados de acordo com o nível de criticidade que eles possuem (assustadores, preocupantes, ou com motivo para se preocupar). Além disso, possui recursos como possibilidade de compartilhamento de informações entre desenvolvedores sobre erros detectados pela primeira vez e que podem ser motivo de alerta; checagem de atualizações online e implementação em forma de plugin que permite a integração com outras ferramentas e ambientes de desenvolvimento.
- PMD [PMD, 2013] permite a detecção de defeitos em código fonte. Sua lista de defeitos detectáveis incluem, a exemplo, objetos e que foram instanciados e variáveis declaradas, porem não usadas e blocos *try/catch* vazios. Ela possui ainda um recurso para a detecção de código fonte duplicado, chamado de *Copy-Past Detector*. Este recurso está disponível para linguagens Java, C, C + + , C # , PHP, Ruby, Fortran e JavaScript. PMD é uma ferramenta que, assim como FindBugs, pode ser integrada a ambientes de desenvolvimento.
- FxCop [FxCop, 2013] é uma ferramenta que atua no ciclo de desenvolvimento de software, atuando sobre projetos desenvolvidos para a plataforma .NET da Microsoft, encontrando e fornecendo ao desenvolvedor uma lista das violações de

regras de programação que estão ocorrendo no projeto em desenvolvimento. FxCop pertence a categoria de ferramentas que detectam defeitos no código fonte.

- QJ Pro [QJ Pro, 2013] pertence à categoria de ferramentas que detectam defeitos em código fonte, limitando-se a linguagem Java. O diferencial desta ferramenta é que ela se baseia nas melhores práticas de desenvolvimento Java para identificar defeitos e ainda é capaz de exibir exemplos para auxiliar na resolução do problema detectado.
- BugMaps [BugMaps, 2013] permite a análise estática de código fonte visando encontrar defeitos. Com ela é possível acessar repositórios de código fonte para analisar diferentes projetos. Os defeitos detectados são visíveis através dos mecanismos gráficos que a ferramenta oferece (visualização gráfica de informações).
- Slicing Metrics [Kula, 2010] é uma ferramenta capaz de se conectar a repositórios de código fonte e extrair métricas sobre defeitos no código. Esta ferramenta, entretanto, analisa apenas código implementado na linguagem C.
- ChangeMiner [ChangeMiner, 2013] é um plugin que pode ser integrado ao ambiente de desenvolvimento de software da Microsoft, o Visual Studio. Através desse plugin pode-se conectar a repositórios de código fonte e analisar erros. Um dos objetivos de ChangeMiner é apoiar desenvolvedores na tarefa de manutenir o código fonte, que pode ser melhorado após a remoção dos erros detectados.
- AQtime Pro [AQtime, 2013] é uma ferramenta proprietária que pode ser usada como um plugin para o ambiente de desenvolvimento Microsoft Visual Studio ou como uma ferramenta isolada. Com ela é possível analisar projetos de código desenvolvidos em linguagens como C/C++, Delphi, .NET, Silverlight, Java, VBScript e Jscript com o objetivo de encontrar pontos que estão comprometendo a qualidade do código fonte, detectando problemas como gargalos de desempenho e falhas de memória.
- MinerAll [Silva, et. al, 2013] foi desenvolvida para dar suporte na tarefa de minerar informações provenientes da análise histórica de repositórios de código fonte aberto. Trata-se de um projeto em desenvolvimento que contempla módulos acopláveis de acesso a diferentes tipos de sistemas de controle de versão, como SVN e CVS objetivando, dentre outras coisas, permitir a análise da dependência existente entre artefatos de software e na prevenção de defeitos e falhas que possam vir a ocorrer no projeto.
- DynaMine [Livshits and Zimmermann, 2005] foi desenvolvida para auxiliar na detecção de defeitos em código fonte, como problemas na API utilizada e violações de regras de codificação. Ela atua na detecção de padrões de violações que podem surgir no código, identificando-os com auxilio de técnicas de mineração de dados. A detecção de padrões leva em consideração a análise histórica do software, ou seja, considerando versões anteriores existentes no repositório de código fonte.

Grupo (ii):

- Bugzilla [Bugzilla, 2013] é uma ferramenta para gestão de defeitos que permite que desenvolvedores relatem os defeitos encontrados no desenvolvimento do software. Com ela é possível cadastrar novos defeitos e acompanhá-los até que sejam resolvidos. Bugzilla também é conhecida como *Defect Tracking System* (sistema de rastreamento de defeitos).

- Bug Track [Bug Track, 2013] é uma ferramenta proprietária, mas com versão *trial* disponível, para gestão de defeitos. Ela permite ao desenvolvedor criar uma linha histórica em ordem cronológica do defeito cadastrado. Possui o diferencial de permitir que equipes distribuídas possam gerenciar seus defeitos através da interface web da ferramenta e serem notificados via *email* sobre alterações realizadas, como mudanças de estado (aberto para resolvido, por exemplo).
- Trac com Subversion [Martins, et. al, 2013] é uma associação de ferramentas que permite ao desenvolvedor associar defeitos detectados a um *commit*. Trac pertence a categoria de ferramentas para gestão de defeitos, assim como o Bugzilla. A definição de um ticket permite que o desenvolvedor identifique e relate um defeito encontrado, fornecendo parâmetros como, por exemplo, *Summary* (resumo do defeito), *Reporter* (quem está cadastrando o defeito) e *Description* (descrição textual do defeito).
- Redmine [Redmine, 2013] assim como Bugzilla, pertence a categoria de ferramentas para gerenciamento de defeitos, onde é possível cadastrar e mantê-los, alterando, quando necessário, o seu estado atual. Dentre as *features* que ele possui estão a capacidade de suportar a gestão de defeitos de vários projetos, notificações via *email* e integração com repositórios como (SVN, CVS, Git, Mercurial, Bazaar e Darcs).
- Mantis [Mantis, 2013] é uma ferramenta que assim como o Bugzilla, se encaixa na categoria *Bug Tracker*, que permitem o gerenciamento de defeitos e a criação de relatórios para exportar os dados.

Grupo (iii):

- Hudson [Hudson, 2013] é uma ferramenta para automação de *build*. Com ela é possível atuar de forma contínua na construção (*Building*) e teste (*Testing*) de um software. Esta ferramenta se encaixa no escopo deste trabalho por permitir que seu utilizador trabalhe de forma integrada com outras ferramentas como PMD e Findbugs, dentre outras, que são responsáveis pela detecção de defeitos em software.
- Jenkins [Jenkins, 2013] assim como Hudson, suporta atividades de integração contínua, como construção (*Building*) e teste (*Testing*). Trata-se de um projeto código aberto que permite a integração de outras ferramentas para a detecção de defeitos.

Grupo (iv):

- FRASR [Poncin et al., 2011] (*Framework for Analyzing Software Repositories*) é um framework que permite a análise de processos de desenvolvimento de software armazenados em repositórios de dados. Dentre as características que ele possui destacam-se a capacidade de extrair informações de sistemas de controle de versão, *bug trackers* e listas de *emails*. Especialmente, ele é capaz de contabilizar os defeitos encontrados e verificar o estado em que se encontram, como criado, fechado, comentado ou reaberto.

Grupo (v):

- BuCo Reporter [Ligu, et. al, 2013] leva em conta o histórico de versões disponíveis de um projeto em um repositório de defeitos. Ela possui um módulo que permite o acesso a sistemas de rastreamento de defeitos e exporta em formato XML um conjunto de informações como defeitos resolvidos, defeitos não resolvidos, tempo médio de correção, dentre outros.

- JGitMiner Web [Roma, 2013] permite a mineração de dados de defeitos, dentre outras coisas, provenientes de repositórios, como o GitHub, a fim de proporcionar a usuários (que podem ser pesquisadores ou donos de algum projeto de software) um ambiente em que eles possam implementar métricas importantes para o contexto das(os) suas(eus) pesquisas/softwares.
- Mozkito [Mozkito, 2013] é um framework de propósito geral capaz de permitir a desenvolvedores minerar informações provenientes de relatórios de defeitos associados ao código fonte. Ele é constituído de diferentes módulos com propósitos específicos, que permitem estendê-lo conforme as necessidades do utilizador.
- BugMiner [Leon Wu, et. al, 2011] atua na mineração de bases de dados de defeitos minerando-as com o objetivo de extrair informações sobre redundâncias encontradas nos defeito cadastrados. O desenvolvimento da ferramenta conta com recursos de mineração de dados e análise estatística.

Na Categoria 3 foram detectados dois diferentes grupos de ferramentas. O primeiro deles é o das (i) ferramentas para gerenciamento de processos de desenvolvimento de software e que também são capazes de reportar dados sobre o processo de desenvolvimento. Já o segundo grupo possui as (ii) ferramentas capazes de minerar informações em repositórios e reportá-las. As seguintes ferramentas foram levantadas:

Grupo (i):

- IssuePlayer [Garousi and Leitch, 2010] é uma ferramenta capaz de permitir a gerentes de projetos gerenciar fatores como desempenho da equipe de desenvolvimento, pontualidade na entrega de resultados, gerenciar o quão eficiente a equipe tem sido e ainda gerenciar a qualidade do que está sendo produzido. A gerência é possível graças a análise de artefatos contidos em repositórios SourceForge.

Grupo (ii):

- DIG [Scott, et. al, 2005] é uma ferramenta capaz de extrair informações de variados repositórios de dados objetivando auxiliar no controle dos processos de desenvolvimento de software, através da análise dos relatórios gerados por ela. Esta ferramenta ainda auxilia a compreensão da variação dos dados nos repositórios.
- Disco [Disco, 2013] é uma ferramenta proprietária capaz de exportar diferentes relatórios sobre o processo de desenvolvimento de software que são fornecidos para análise. Trata-se de uma ferramenta para gerenciamento e descoberta de informações sobre processos de desenvolvimento de software. Alguns exemplos de informações que podem ser obtidas são: duração total do projeto, duração média e máxima dos projetos.
- FRASR [Poncin et al., 2011] como descrito nas ferramentas da Categoria 2, é capaz de extrair dados sobre defeitos. Entretanto, ele também foi recuperado como uma ferramenta capaz de extrair dados sobre processo de software, mais precisamente dados armazenados em repositórios de código fonte. Manipulando tais repositórios, é possível extrair dados como quem mais realizou *commits*, quem são os autores e quando as atividades foram finalizadas pelos desenvolvedores. Pertence o conjunto de ferramentas para mineração de informações sobre processos de desenvolvimento.

- *Information Display With Multiple Views* [Hilst and Huang, 2013] suporta a extração de informações sobre o processo de desenvolvimento de software focando na identificação de *logs* como: quem alterou um dado artefato, quando foi alterado, quais alterações foram realizadas. A ideia é manipular repositórios de controle de versão e de acompanhamento de tarefas para minerar informações. Pertence o conjunto de ferramentas para mineração de informações sobre processos de desenvolvimento.

Resultados da questão de pesquisa (Q2)

Três diferentes tabelas foram criadas para auxiliar na visualização da aderência das ferramentas aos critérios definidos na subseção Planejamento do Mapeamento Sistemático. A Tabela 1 mostra a aderência das ferramentas da Categoria 1 com os critérios definidos para a Categoria 1. A Tabela 2 mostra a aderência das ferramentas da Categoria 2 com os critérios definidos para a Categoria 2. Já a terceira e última tabela, Tabela 3, mostra a aderência das ferramentas da Categoria 3 com os critérios definidos para a Categoria 3. Células marcadas com Sim definem que a ferramenta em questão se encaixa no critério relacionado. Células marcadas com Não revelam que a ferramenta não se encaixa em algum ponto do critério.

Tabela 1. Ferramentas da Categoria 1

Ferramenta	Critérios				Ferramenta	Critérios			
	C1	C2	C3	C4		C1	C2	C3	C4
Analizo	Sim	Sim	Sim	Sim	*Source Miner*	Não	Sim	Sim	Não
Kalibro Metrics	Sim	Sim	Sim	Sim	*EvoJava*	Sim	Sim	Não	Sim
Metric Miner	Não	Sim	Não	sim	*EvoTrack*	Sim	Sim	Sim	Não
SearchEngine	Não	Sim	Não	Sim	*MASU*	Não	Sim	Não	Não
Epona	Não	Sim	Não	Sim	*MinerAll*	Sim	Não	Não	Não
Sourcerer	Sim	Sim	Não	Sim	*SO-MSR*	Sim	Sim	Sim	Não

Tabela 2. Ferramentas da Categoria 2.

Ferramenta	Critérios			Ferramenta	Critérios		
	C1	C2	C3		C1	C2	C3
Bugzilla	Sim	Sim	Sim	*BugMaps*	Sim	Sim	Não
BuCo Reporter	Sim	Sim	Sim	*Slicing Metrics*	Sim	Não	Sim
Bug Track	Sim	Sim	Sim	*MinerAll*	Sim	Não	Não
Redmine	Sim	Sim	Sim	*JGitMiner*	Não	Não	Não
Hudson	Sim	Sim	Sim	*ChangeMiner*	Sim	Não	Sim
Jenkins	Sim	Sim	Sim	*Trac + Subversion*	Sim	Sim	Não
Mantis	Sim	Sim	Sim	*AQtimePro*	Não	Não	Sim
Findbugs	Não	Não	Sim	*DynaMine*	Sim	Sim	Não
PMD	Não	Não	Sim	*FRASR*	Sim	Sim	Não
FxCop	Não	Não	Sim	*Mozkito*	Sim	Não	Sim

QJ Pro	Não	Não	Sim	*BugMiner*	Não	Sim	Não

Tabela 3. Ferramentas da Categoria 3.

Ferramenta	Critérios		
	C1	C2	C3
IssuePlayer	Sim	Sim	Sim
DIG	Sim	Sim	Sim
Disco	Sim	Sim	Sim
FRASR	Sim	Sim	Não
Information Display With Multiple Views	Não	Não	Não

A Tabela 4 apresenta o resultado final do levantamento das ferramentas que estão de acordo com todos os critérios pré-definidos no protocolo. Nela é possível ver a divisão das ferramentas pelas categorias de repositório de dados históricos, bem como os grupos que foram identificados na questão de pesquisa Q1.

Tabela 4. Ferramentas divididas por categoria de repositório e grupos.

Categoria 1	
Ferramenta	Grupo Pertencente
Analizo	Grupo (i)
Kalibro Metrics	Grupo (ii)
Categoria 2	
Ferramenta	Grupo Pertencente
Bugzilla	Grupo (ii)
Redmine	Grupo (ii)
Bug Track	Grupo (ii)
Mantis	Grupo (ii)
Hudson	Grupo (iii)
Jenkins	Grupo (iii)
BuCo Reporter	Grupo (v)
Categoria 3	
IssuePlayer	Grupo (i)
DIG	Grupo (ii)
Disco	Grupo (ii)

Todas as ferramentas contidas na Tabela 4 foram utilizadas na definição do framework conceitual *GiveMe Metrics.*

REFERÊNCIAS

Steinmacher, I. Chaves, A.P. Gerosa, M.A. (2012), Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature. Journal of Computer Supported Cooperative Work.

Kitchenham, Barbara. (2004), Procedures for performing systematic reviews. Technical Report, TR/SE-0401, Department of Computer Science, Keele University, Keele,UK.

Araújo, M.A.P., Travassos, G.H., Kitchenham, B., (2005), Evolutive Maintenance: Observing Object-Oriented Software Decay, Technical Report ES-672/05 - COPPE/UFRJ.

Sokol, F.Z.; Finavaro Aniche, M.; Gerosa, M.A., (2013), "MetricMiner: Supporting researchers in mining software repositories," *Source Code Analysis and Manipulation (SCAM), 2013 IEEE 13th International Working Conference on* , vol., no., pp.142,146, 22-23 Sept.

Search Engine: http://www.semanticdesigns.com/Products/SearchEngine/. Acessado em: 22 de novembro de 2013.

Sourcerer: http://sourcerer.ics.uci.edu/. Acessado em: 22 de novembro de 2013.

Morais, Carlos. Meirelles, Paulo. Morais, Eduardo, (2013), *Kalibro Metrics: um serviço para monitoramento e interpretação de métricas de código fonte. C*entro de Competência em Software Livre. Instituto de Matemática e Estatística. Universidade de São Paulo.

Terceiro, *Antonio. Costa, Joenio. Meirelles, Paulo. Miranda, João, Rios, Luiz Romário Santana. Almeida, Lucianna. Chavez, Christina. Kon, Fabio.* (2013), Analizo: an Extensible Multi-Language Source Code Analysis and Visualization Toolkit**.** Universidade Federal da Bahia.

SourceMiner: http://www.sourceminer.org/index.html. Acessado em: 24 de novembro de 2013.

Oosterman , Joshua. IRWIN, Warwick. Churcher , Neville. (2011), EvoJava: a tool for measuring evolving software. In *Proceedings of the Thirty-Fourth Australasian Computer Science Conference - Volume 113* (ACSC '11), Mark Reynolds (Ed.), Vol. 113. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 117-126.

Werner, Cláudia., L. Murta, M. Schots, A. Magdaleno, M. Silva, R. Cepeda, C. Vahia, (2011), EvolTrack: A Plug-in-Based Infrastructure for Visualizing Software Evolution. I Workshop Brasileiro de Visualização de Software/II Congresso Brasileiro de Software: Teoria e Prática (CBSoft), São Paulo, Brazil, September 2011, pp. 1-8.

Higo, Yoshiki. Saitoh, Akira. Yamada , Goro. Miyake, Tatsuya. Kusumoto, Shinji. Inoue, Katsuro. (2011), A Pluggable Tool for Measuring Software Metrics from Source Code. In *Proceedings of the 2011 Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement* (IWSM-MENSURA '11). IEEE Computer Society, Washington, DC, USA, 3-12.

Silva, José Teodoro. Wiese, Igor S. Steinmacher, Igor. Gerosa, Marco Aurélio. (2013), MinerAll: Uma ferramenta para extração e mineração de dados de repositórios de software livre. Coordenação de Informática – Universidade Tecnológica Federal do Paraná (UTFPR). Departamento de Ciência da Computação – Universidade de São Paulo (USP). Acessado em: 18 novembro 2013.

Matsumoto, Shinsuke. Nakamura, Masahide, (2011), Service Oriented Framework for Mining Software Repository, In The Joint Conference of the 21st International

Workshop on Software Measurement (IWSM) and the 6th International Conference on Software Process and Product Measurement (Mensura), pp.13-19, November 2011. (Nara, Japan).

Junior, Silva. Juvenal, Flávio. (2012), Ferramenta para coleta de dados em projetos open source. Centro de Informática da Universidade Federal de Pernambuco. Trabalho de conclusão de curso.

GitHub: https://github.com/. Acessado em 26 de novembro de 2013.

FindBugs: http://findbugs.sourceforge.net/. Acessado em: 22 de novembro de 2013.

PMD: http://pmd.sourceforge.net/. Acessado em 24 de novembro de 2013.

FxCop: http://msdn.microsoft.com/en-us/library/bb429476%28v=vs.80%29.aspx. Acessado em: 18 de novembro de 2013.

QJ Pro: http://qjpro.sourceforge.net/concepts.html. Acessado em: 22 de novembro de 2013.

BugMaps : http://java.labsoft.dcc.ufmg.br/bugmaps/index.html. Acessado em: 19 de novembro de 2013.

Kula, Raula., (2010), Using Program Slicing Metrics for the Analysis of Bug Fixing Processes – Nara Institute of Science and Technology.

ChangeMiner : https://sites.google.com/site/frchico/changeminer. Acessado em: 26 de novembro de 2013.

AQtime Pro : http://smartbear.com/products/qa-tools/application-performance-profiling/. Acessado em: 17 de novembro de 2013.

Livshits, Benjamin. Zimmermann ,Thomas. (2005), DynaMine: finding common error patterns by mining software revision histories. *SIGSOFT Softw. Eng. Notes* 30, 5.

Bugzilla: http://www.bugzilla.org/features/. Acessado em 25 de novembro de 2013.

Bug Track : http://www.bugtrack.net/tour.html. Acessado em: 25 de novembro de 2013.

Martins, Daves Marcio Silva. Classe, Tadeu Moreira de. Palmeira, Guilherme de Jorge. DORNELAS, Eduardo Leandro Pinto. Integração das ferramentas Trac e Subversion. Engenharia de Software Magazine, Edição 26. Acessado em: 18 novembro de 2013.

Redmine: http://www.redmine.org/. Acessado em: 25 de novembro de 2013.

Mantis: http://www.mantisbt.org/. Acessado em: 25 de novembro de 2013.

Hudson Plugins: http://hudson-ci.org/PluginCentral3/. Acessado em: 25 de novembro de 2013.

Jenkins Plugins: https://wiki.jenkins-ci.org/display/JENKINS/Plugins. Acessado em 25 de novembro de 2013.

Poncin, W. Serebrenik, A. Van den Brand, M., (2011), "Process Mining Software Repositories,".*Software Maintenance and Reengineering (CSMR), 15th European Conference on* , vol., no., pp.5,14, 1-4

Ligu, Elvis. Chaikalis, Theodoros. Chatzigeorgiou, Alexander, (2013), BuCo Reporter: Mining Software and Bug Repositories. Acessado em: 26 novembro de 2013.

Roma, Douglas N, (2013), Uma Ferramenta para Mineração de Dados de Projetos de Software, Criação de Redes e Cálculo de Métricas Sócio -Técnicas. 59 f. Trabalho

de Conclusão de Curso – Curso Superior de Tecnologia em Sistemas para Internet, Universidade Tecnológica Federal do Paraná – UTFPR. Campo Mourão.

Mozkito : https://mozkito.org/. Acessado em: 21 de novembro de 2013.

Leon Wu, Boyi Xie, Gail E. Kaiser, Rebecca J. Passonneau, (2011), BugMiner: Software Reliability Analysis Via Data Mining of Bug Reports. In proceeding of: Proceedings of the 23rd International Conference on Software Engineering & Knowledge Engineering (SEKE'2011), Eden Roc Renaissance, Miami Beach, USA, July 7-9, 2011.

Garousi, Vahid. Leitch, James, (2010), IssuePlayer: An extensible framework for visual assessment of issue management in software development projects. *J. Vis. Lang. Comput.* 21, 3 (June 2010), 121-135.

Scott D. Miller, Aditya P. Mathur, Raymond A. DeCarlo, (2005), DIG: A Tool for Software Process Data Extraction and Grooming. In *Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01* (COMPSAC '05), Vol. 1. IEEE Computer Society, Washington, DC, USA, 35-40.

Disco: http://www.fluxicon.com/disco/. Acessado em: 21 de novembro de 2013.

Hilst, Van M. Huang, Shihong, (2013), Repository views for rapid exploration and developer insight, *Southeastcon, 2013 Proceedings of IEEE* , vol., no., pp.1,6, 4-7