FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE – FANESE
MBA EM ADMINISTRAÇÃO DE BANCO DE DADOS

MARLOS ALÃ SANTOS FIAIS

MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

Aracaju – SE
2016.1

MARLOS ALÃ SANTOS FIAIS

MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE, da Faculdade de Administração de Negócios de Sergipe – FANESE, como requisito para a obtenção do título de Especialista em Banco de Dados.

_____

Nome completo do Avaliador

_____

Nome completo do Coordenador de Curso

_____

Nome completo do Aluno

Aprovado (a) com média: _____

Aracaju (SE), ____ de _____ de 2016.

MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

Marlos Alã Santos Fiais

RESUMO

No ambiente da programação alguns programadores se deparam com códigos difíceis de serem entendidos no momento de uma manutenção, já que muitos desses códigos não são mantidos por quem os programou. Muitas vezes, esses programadores não fazem mais parte da equipe. Existem ferramentas sendo desenvolvidas para auxiliar o processo de manutenção de *softwares*, facilitando a vida dos programadores. A mineração de dados é utilizada pela Engenharia de Software na busca de padrões e tendências que possam auxiliar na construção, manutenção e evolução de *softwares* através de repositórios de *software* que guardam informações como: código-fonte, histórico de versões (logs), relatórios de erros (rastreio de defeitos), entre outros. Nesse artigo científico, qualitativo e bibliográfico, iremos revelar como o uso da mineração de dados pode auxiliar no desenvolvimento e manutenção de *softwares*.

Palavras-chave: Evolução do *Software,* Mineração de Repositórios de *Software,* Repositórios de *Software*.

INTRODUÇÃO

Os avanços obtidos nas áreas de *software* e *hardware* possibilitaram a criação de aplicações comerciais e científicas capazes de processar grandes volumes de dados. Por isso, são necessárias novas ferramentas e técnicas capazes de analisar automaticamente o volume de dados produzidos, fornecendo o conhecimento necessário para auxiliar nos processos de decisão dentro das corporações.

Do que adiantaria ter um grande volume de dados se eles não transmitem informação útil?

Os projetos de *software* possuem também à sua disposição uma grande quantidade de dados que são produzidos durante as atividades de desenvolvimento e de apoio. São milhares de linhas de código escritas, defeitos são relatados e resolvidos, discussões técnicas acontecem nas listas discussões e de e-mails, etc. Analisar tais dados contribui no avanço do entendimento sobre o projeto e possibilita que decisões sejam tomadas de forma mais fundamentada.

A Mineração de Dados aliada aos grandes repositórios de dados, vem como uma ferramenta capaz de transformar os dados coletados de diversas fontes em informações úteis que serão utilizadas pelos gestores das corporações para tomarem decisões importantes.

Da mesma forma, a Mineração de Dados aliada a Engenharia de *Software* tem auxiliado os desenvolvedores de *software.* Utilizando-se dos repositórios de históricos de *software* é possível descobrir padrões e regras que podem melhorar a qualidade do software e aumentar a sua produtividade.

Assim, iremos trazer nesse artigo, um pouco da evolução do *software*, definindo repositórios de *softwares*, e finalmente trazendo algo sobre a Mineração desses repositórios no auxílio aos desenvolvedores de *software*.

2 A EVOLUÇÃO DO SOFTWARE

A evolução do *software* é um tema que tem despertado bastante interesse em meio aos profissionais de desenvolvimento de *software*. Essa evolução está diretamente ligada às mudanças frequentes que acontecem no mundo real. Assim como no mundo real, os *softwares* passam por várias mudanças ao longo do tempo. Essas mudanças vão de correções de pequenos erros gerados por inconsistências no código, atualizações que podem ser para melhorias detectadas ou pela solicitação de um determinado cliente.

Compreender o processo de evolução de um *software* é uma tarefa complexa. Os sistemas de *software* grandes possuem um longo histórico de desenvolvimento com diversos desenvolvedores trabalhando em conjunto ou partes separadas do sistema.

É comum que nenhum desenvolvedor conheça o código-fonte do sistema por completo por conta da sua complexidade ou pelo fato de que integrantes que iniciaram o desenvolvimento do projeto já não fazem mais parte da equipe.

Assim, analisar os dados históricos do desenvolvimento de um *software* grande manualmente é impraticável. Portanto, a Mineração de Repositórios de *Software* (MRS) analisa a evolução de *software* de forma automatizada aplicando técnicas da Mineração de Dados sobre o histórico do desenvolvimento de sistemas de *software*.

Segundo proposta de Lehman (citado por SILVA, 2013, p. 7), existe um conjunto de oito leis, apresentadas no Quadro demonstrativo 1, que segundo o autor se aplicavam à evolução de todos os sistemas de *software*.

Quadro Demonstrativo 1

LEIS DE LEHMAN

1. Mudança contínua	Um *software* deve ser continuamente atualizado, caso contrário tornar-se-á progressivamente menos satisfatório.
2. Complexidade Crescente	À medida que um *software* é alterado verifica-se um acréscimo da sua complexidade, a não ser que seja feito trabalho para mantê-la ou reduzi-la.
3. Auto regulação	O processo de evolução de *software* é autorregulado próximo à distribuição normal em relação às medidas dos atributos de produtos e dos processos.
4. Conservação da Estabilidade Organizacional	A taxa média de atividade global efetiva num *software* em evolução tende a ser constante durante o tempo de vida do produto.
5. Conservação da Familiaridade	Regra geral, a taxa de crescimento incremental e a taxa de crescimento a longo prazo tendem a declinar.
6. Crescimento Contínuo	O conteúdo funcional de um *software* deve ser aumentado continuamente durante o seu tempo de vida, a fim de ir ao encontro das necessidades dos utilizadores.
7. Qualidade Decrescente	A qualidade do *software* será entendida como declinante a não ser que o *software* seja adaptado rigorosamente às mudanças no ambiente operacional.
8. Sistema de Retroalimentação	Os processos de evolução de *software* são sistemas de retroalimentação em múltiplos níveis, em múltiplos *loops* e envolvendo múltiplos agentes.

Fonte: Silva
Data: 2013, p. 7.

Ainda não existe um consenso do valor dessas leis para sistemas de *software* de código-fonte aberto. Em FERNÁNDEZ-RAMIL et al. 2008 (citado por SOKOL, p. 7), alguns autores discutem a validade dessas leis a partir da análise de diversos estudos realizados sobre projetos de código aberto concluindo que algumas das leis se aplicam, enquanto outras ainda não foram evidenciadas cientificamente.

3 REPOSITÓRIOS DE SOFTWARE

O Repositório de *Software* é toda ferramenta que apoie a execução de processo de desenvolvimento de *software* e que sustente informações sobre as atividades realizadas pelos

agentes do processo. Os repositórios de *software*, na prática, apenas armazenam informações, e pouco são usados para dar suporte ao processo de decisão. Os dados armazenados nos repositórios poderão ser recuperados e tratados para gerarem informações importantes na melhoria do processo de desenvolvimento de *softwares*.

Esse termo abrange todos os artefatos produzidos durante o desenvolvimento de um sistema de *software*, que vai desde os arquivos com o código-fonte do sistema que podem estar armazenados em um sistema de controle de versão, até mensagens em listas de discursão e de e-mails trocadas entre os desenvolvedores. Tais repositórios contêm informações valiosas, que podem ser exploradas para compreender a evolução de *software* e contribuir com o desenvolvimento do projeto.

Um sistema de controle de versão é uma ferramenta utilizada no cotidiano de desenvolvimento de *software*. Por meio dessa ferramenta, um desenvolvedor controla as alterações sobre o código que está modificando. Com essa ferramenta, o desenvolvedor agrupa as mudanças feitas no código em *commits* e o sistema de controle de versão armazena esses grupos de alterações. Essa é uma fonte de dados muito importante no contexto de Mineração de Repositórios.

Em D'AMBROS et al. 2008 (citado por SOKOL, p. 8), os autores destacam os seguintes tópicos estudados na análise de repositórios de software:

- Concentração do trabalho dos desenvolvedores e análise de redes sociais: descobrir em quantos e em quais pontos do software os desenvolvedores estão dedicando mais seus esforços e como eles se comunicam, para buscar soluções de problemas no processo de desenvolvimento e na estrutura da equipe.
- Impacto e propagação de alterações: busca entender o efeito das mudanças feitas em certa parte do sistema sobre o resto do código do projeto. Compreendendo melhor o efeito dessas alterações, a equipe estima melhor os custos das tarefas. Além disso, um desenvolvedor pode ser informado de quais arquivos ele precisará checar após ter feito certa alteração.
- Análise de tendências e hotspots: são pontos do software que sofrem alterações frequentemente. Encontrar tais pontos do sistema ajuda a levantar deficiências na arquitetura do projeto e sugerir possíveis ajustes para aprimorar sua manutenção.
- Previsão de falhas e defeitos: os dados disponíveis em repositórios de software podem ser usados como entrada para algoritmos de aprendizagem de máquina, visando criar modelos preditivos de possíveis falhas no sistema. Com esse modelo, a equipe toma ações para prevenir falhas em versões futuras.

Assim, a área de MRS usa esses repositórios para extrair os dados disponíveis e descobrir informações importantes que auxiliem nos projetos de *software*, para que os *softwares* não se tornem obsoletos, sem novas funcionalidades e tenha sua vida útil comprometida.

4 MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

As grandes corporações buscam sempre a tomada de decisões analisando dados históricos que são coletados de diversas fontes. Os projetos de *software* possuem à sua disposição uma grande quantidade de dados que são produzidos durante as atividades de desenvolvimento e de manutenção. São diversas linhas de código escritas ao longo do tempo, muitos defeitos são relatados e corrigidos em listas de discussões e de e-mails, sistemas de gerenciamento de tarefas, fóruns e documentação do *software*, etc. A análise de tais dados aumenta de forma significativa o entendimento sobre o projeto, os desenvolvedores e os processos que governam a evolução e manutenção do *software* e possibilita que decisões sejam tomadas de forma mais fundamentada.

Antes de tratarmos da MRS, vamos falar um pouco sobre a mineração de dados (no inglês, *Data Mining*). A Mineração de dados é uma área da computação que consiste, com o uso de inteligência artificial, em extrair dados armazenados em diferentes fontes e analisa-los, transformando-os em informações úteis que possam ser utilizadas na tomada de decisão por parte dos gestores das corporações.

A mineração de dados é apenas uma parte do processo de Descoberta do Conhecimento em Bases de Dados (KDD - *knowledge discovery in databases*).

O KDD é todo processo de descoberta de conhecimento útil nos dados, já o *Data Mining* refere-se à aplicação de algoritmos para extração de modelos dos dados. Alguns autores consideram os dois termos sinônimos, assim, o *Data Mining* se torna a principal etapa do KDD, etapas essas são citadas a seguir:

1- Limpeza de dados: eliminação de ruídos e dados inconsistentes.
2- Integração de dados: combinação de diferentes fontes de dados.
3- Seleção: seleção de atributos de interesse do usuário.
4- Transformação dos dados: transformação dos dados em um formato reconhecido pelo algoritmo de mineração.
5- Mineração: aplicação de técnicas inteligentes para extrair padrões de interesse.
6- Avaliação ou Pós-processamento: identificação de padrões interessantes de acordo com os critérios do usuário.

7- Visualização dos resultados: utilização de técnicas de representação de conhecimento com a finalidade de apresentar o conhecimento minerado ao usuário.

Os algoritmos de mineração de dados deverão ser capazes de encontrar padrões que possam auxiliar os gestores na tomada de decisão.

A mineração de dados pode ser utilizada para descobrir tendências e padrões em diferentes repositórios de dados, entre eles, aqueles oriundos da construção e manutenção de *softwares*.

> As técnicas de mineração precisam ser desenvolvidas ou adaptadas para tarefas e domínios específicos, e na Engenharia de *Software* não é diferente, necessita-se de abordagens específicas para se minerar os dados que são provenientes da construção de *software*.
> Entre as fontes de dados passíveis de análise podemos citar: bases de código estático, histórico de versões do *software*, rastros de execução de programas, relatórios de erros, listas de discussão e logs de implantação de sistemas. COLAÇO (2010, v. 31)

Várias tarefas da Engenharia de *Software* podem ser auxiliadas com o uso da MRS, entre elas temos:

- Programação: em nível de construção de *software*, o programador pode utilizar-se das melhores práticas de programação identificando características de uso de uma API (*Application Program Interface*) ou *framework* automaticamente; mantendo os padrões de uso atualizados, baseando-se sempre na mais nova versão de uma API ou *framework*; identificando padrões que abranjam casos de herança em *frameworks*.
- Detecção de defeitos: todo sistema segue algumas regras para se manter correto, mas na maioria das vezes essas regras não são documentadas. A derivação dessas regras requer um conhecimento prévio muito grande e as técnicas de mineração de dados podem ajudar a inferi-las a partir do código-fonte.
- Depuração: diversas aplicações, principalmente as de código aberto, possuem repositórios de erros, os quais possuem relatórios dos erros e possíveis soluções. Estes relatórios consomem tempo de desenvolvimento e muitas vezes são duplicados, contudo, são um compendio valioso de informações. Assim, as técnicas de mineração podem ser utilizadas para tentar predizer se há probabilidade da geração de um erro dado um estado S de um programa e um evento E.
- Manutenção: toda manutenção de *software* se inicia em um ponto específico do programa. Minerar dados do histórico de alterações armazenado em sistemas de

> controle de versão pode ajudar programadores com sugestões do tipo: “Programadores que alteraram esse método também alteraram o(s) método(s) ...”. (COLAÇO, 2010)

Nesse contexto, a MRS é uma área de pesquisa que tem o seu foco na recuperação, interligação e análise dos dados históricos que são produzidos durante o desenvolvimento de *softwares* e armazenados em repositórios. Suponha que um desenvolvedor fez uma mudança em uma determinada parte de um sistema. O que mais ele tem que mudar?

Baseado na ideia de que os artefatos que mudaram juntos no passado têm uma forte tendência a mudar juntos no futuro, pesquisadores têm criado ferramentas que auxiliam na disseminação de mudanças. A mineração também é utilizada para entender como o *software* evoluiu ao longo do tempo, compreender os efeitos da interação social entre os desenvolvedores no processo de desenvolvimento e prever defeitos e dificuldades. A MRS é um campo de pesquisa relativamente novo e tem atraído o interesse de diversos pesquisadores.

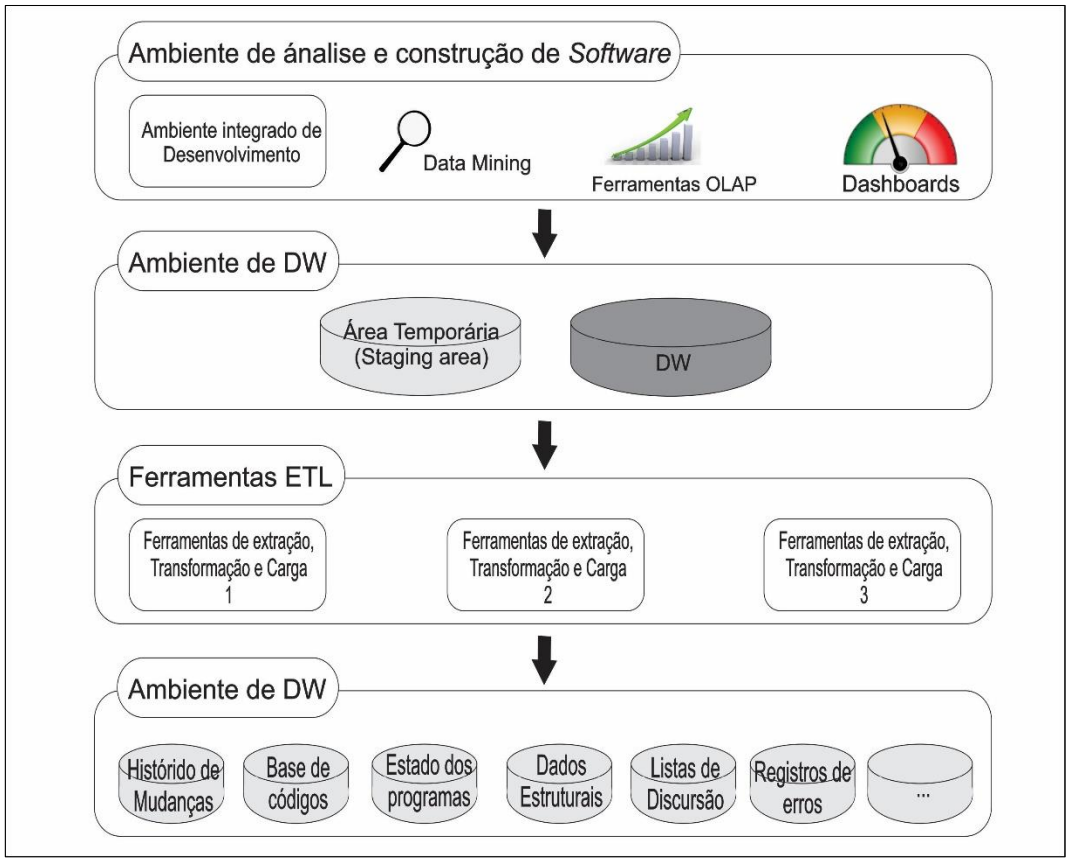
Grande parte do sucesso da área de mineração de repositórios de *software* deve-se à abundante disponibilização de dados cedidos pelo movimento de *software* livre. Projetos de grande visibilidade, como o Linux, a Eclipse IDE e vários outros da Apache Software Foundation e da Free Software Foundation, assim como projetos mais recentes hospedados no GitHub (Serviço de Web Hosting compartilhado para projetos), são frequentemente escolhidos para realização de estudos científicos.

Existem várias ferramentas de mineração de repositórios de *software* encontradas na literatura. Entre elas temos:

- ChangeMiner: é um plug-in gratuito para Visual Studio que usa regras de associação para minerar o histórico de versões e orientar programadores para prováveis pontos de manutenção. Dado um conjunto de mudanças existentes, regras sugerem e predizem mudanças prováveis, identificam acoplamentos e previnem erros provenientes de mudanças incompletas. No momento da alteração do código-fonte, baseado no histórico pregresso de alteração conjunta de módulos, o *plug-in* reporta ao desenvolvedor os próximos pontos prováveis de manutenção tanto de maneira textual quanto gráfica, com objetivo de aumentar a eficiência da manutenção e diminuir a inserção de erros.

Figura 1

ARQUITETURA DO CHANGEMINER



Fonte: Santos, Colaço, Mendonça
Data: 2014, p. 4.

- Neurominer: é uma ferramenta de análise em Programação Neolinguística para extrair das listas de discussão de um projeto o canal cognitivo mais usado pelos desenvolvedores (Sistema de Representação Preferencial Contextual - SRP). Isso ajuda na tomada de decisão de alocação de desenvolvedores, bem como traça o perfil psicológico de cada colaborador ou cliente da corporação, fazendo-se uso de qualquer texto que represente uma manifestação dos mesmos. A base do Neurominer consiste em um dicionário neuro-linguístico formado pelas palavras que detectam o sistema de representação usado e por conceitos de Engenharia de *Software*. O Neurominer combina mineração de dados, técnicas de inteligência estatísticos e artificiais.
- MetricMiner: é um sistema com interface web, onde os usuários cadastram projetos de *software* para serem minerados. Nesse cadastro devem ser fornecidos um nome e a url pública para o repositório de código do projeto. Os sistemas de controle de versão atuais, com suporte ao MetricMiner são o Git e o SVN. Como a maioria das etapas da mineração de dados são demoradas, como clonar repositórios, persistir suas

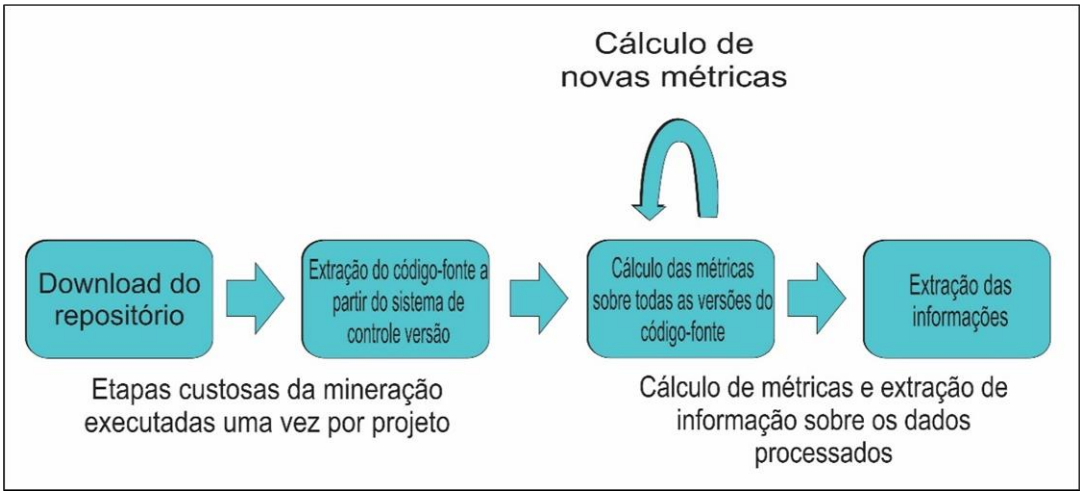
informações no banco de dados e calcular métricas sobre o código-fonte, essas tarefas são executadas por meio de uma fila de execução armazenada no banco de dados. Cada tarefa é registrada no sistema e processada por um componente que constrói as dependências de cada tarefa e as executa na ordem em que foram cadastradas.

Após o cadastro do projeto são registradas quatro tarefas relativas ao mesmo: carregamento do repositório de código, processamento de todas as versões dos arquivos do repositório e armazenamento dos dados, remoção dos arquivos carregados e cálculo das métricas de código sobre todas as versões do código-fonte do projeto.

Finalizando o processo, todas as informações do repositório do projeto e métricas de código estão gravadas no banco de dados e disponíveis para serem extraídas pelo pesquisador. A Figura 2 descreve as tarefas envolvidas no processo de mineração sobre um repositório de software cadastrado na ferramenta.

Figura 2

TAREFAS ENVOLVIDAS NO PROCESSO DE MINERAÇÃO SOBRE UM REPOSITÓRIO DE SOFTWARE CADASTRADO NA FERRAMENTA METRICMINER



Fonte: Sokol
Data: 2012, p. 16.

- MinnerAll: é uma ferramenta utilizada atualmente para pesquisas relacionadas à busca e reconhecimento de padrões para predição de bugs e falhas em versões. A precisão destes tipos de informações é de grande valia para gerentes e membros de equipes de desenvolvimento de projetos. A construção destes mineradores fornece dados que ampliam a riqueza das informações que podem ser extraídas. Um módulo responsável pelo levantamento de métricas e visualizações está em fase de análise e projeto, e

contará com algumas representações gráficas, tais como, participação de desenvolvedores, frequência de correções, acoplamento físico e lógico entre módulos. Uma vez que a arquitetura da ferramenta possibilita extensões, podem ser desenvolvidos e adicionados: novos tipos de repositórios, métricas de software, formas de visualização e interpretadores. Com esta ferramenta é possível estudar o comportamento de desenvolvedores e dos projetos de software livre. Ela fornece informações importantes para tomada de decisões dos líderes de projetos, como a participação dos membros por exemplo. Atividades futuras preveem a ampliação dos mineradores para analisar dados de repositórios e de listas de discussão simultaneamente, bem como a construção de um serviço web que permita executar consultas remotamente sobre os dados minerados.

- Odyssey-WI: é uma ferramenta que tem como objetivo, detectar rastros de modificação utilizando repositório de sistema de controle de versões. O sistema de controle de versão utilizado pela ferramenta é o Odyssey-VCS, e do sistema de controle de modificações, o Odyssey-CCS. Odyssey-VCS é um sistema de controle de versões que atua sobre modelos UML, fazendo uso de uma política configurável para a unidade de versionamento. A unidade de versionamento representa os elementos que serão versionados e, portanto, irão se tornar itens de configuração quando enviados para o repositório. Neste cenário, a unidade de versionamento pode ser casos de uso, classes, atributos, operações, etc. O versionamento proposto pelo sistema permite o acompanhamento da evolução individual de cada elemento do modelo. Odyssey-CCS é um sistema configurável e flexível para o processo de controle de modificações. Com ele o gerente de configuração modela o processo e define as atividades.

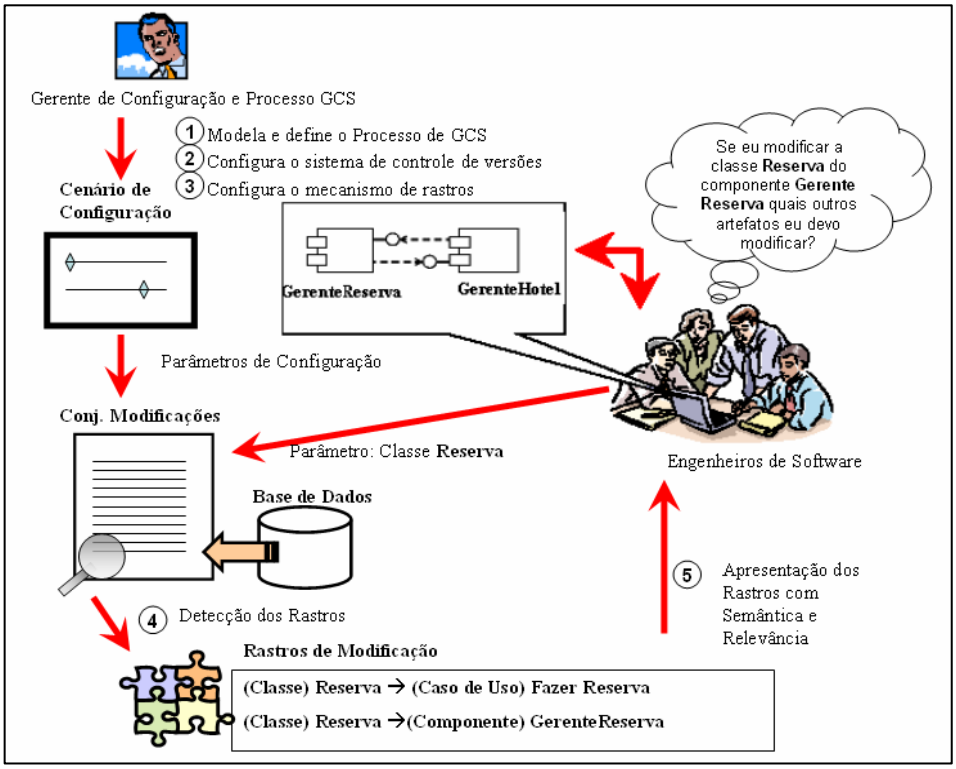
  Este sistema é utilizado, principalmente, na etapa inicial de preparação do ambiente e na execução do processo de controle de modificações. Durante o processo de desenvolvimento, o engenheiro de software efetua *checkout* dos artefatos, realiza as modificações necessárias, e ao final, armazena os artefatos no repositório, através da operação de *check-in*. Com isso, uma nova versão dos artefatos é gerada. Esses artefatos são identificados no sistema de controle de versões como itens de configuração.

  A ferramenta Odyssey-WI deve ser utilizada sempre que o engenheiro de software necessite de orientação sobre artefatos que deve modificar. A ferramenta, que detecta os pares de artefatos UML que sofreram modificações juntos durante o processo de

desenvolvimento e manutenção, extrai do repositório as modificações realizadas no *software* e realiza sobre esse conjunto a mineração dos dados.

Figura 3

CENÁRIO DE UTILIZAÇÃO DA FERRAMENTA ODYSSEY-WI



Fonte: Dantas, Murta, Werner
Data: 2016. P 5.

- *SourceMiner:* é um *plug-in* para a Plataforma Eclipse que auxilia seus usuários no processo de visualização e compreensão de *software*. Ele disponibiliza algumas formas visuais de inspecionar o código-fonte de um *software*, provendo visualizações em árvore, por dependência de classes e pacotes, filtro para seleção de recursos a serem analisados, entre outros.

  O *SourceMiner* integra diferentes paradigmas de visualização embutidos em quatro diferentes visões: *TreeMap, Polymetric, Dependency e GridCoupling.* Abaixo seguem as principais características de cada visão:

  - *TreeMap*: representa a estrutura pacote-classe-método, ou seja, a estrutura dos pacotes, classes e métodos em si e seus nomes e tamanhos.
  - *Polymetric:* é uma representação bidimensional que usa retângulos para representar entidades do *software*, tais como classes e interfaces, bem como linhas para

representar o relacionamento de herança entre elas. As dimensões dos retângulos são usadas para representar propriedades das entidades.

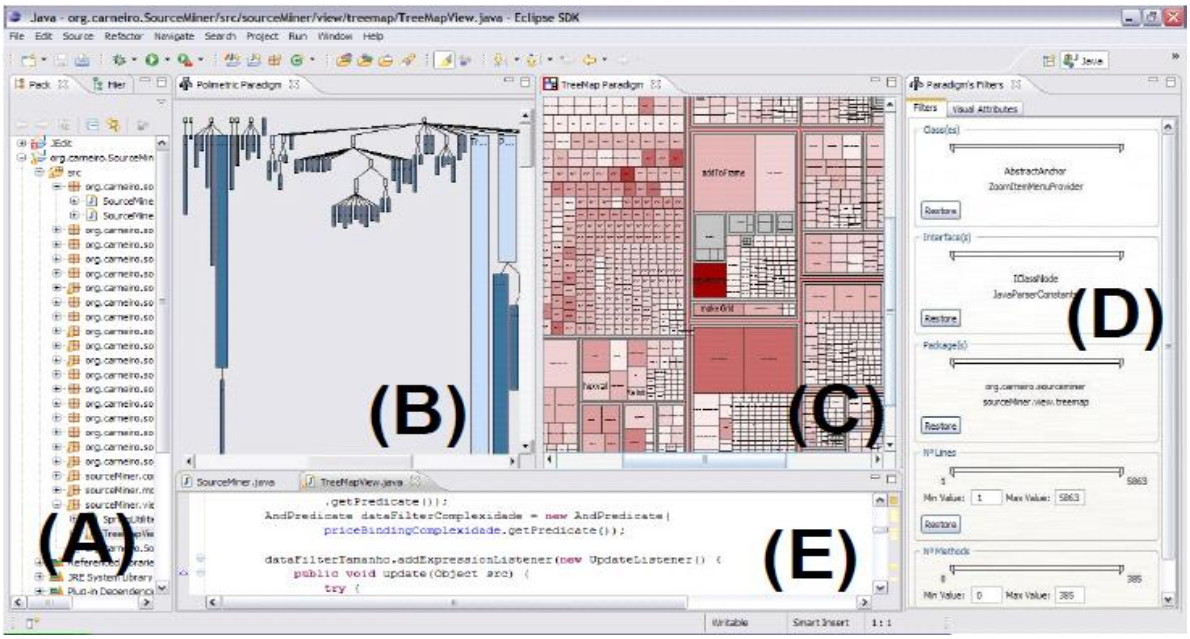
- *Dependency:* representa a dependência entre classes ou pacotes através de um grafo.
- *GridCoupling:* possui dois objetivos: o primeiro é representar uma visão geral sobre o grau de acoplamento e todos os módulos que compõem o projeto do *software*. O segundo objetivo é representar o grau de acoplamento do nó selecionado com outros.

A Figura 4 apresenta uma imagem do *SourceMiner*. Ela exibe um possível cenário que inclui duas das visões do *plug-in*, a *Polymetric* (B) e *TreeMap* (C).

As visões são organizadas lado-a-lado, assim como podem ser complementadas por outras visões, como o Editor do Eclipse (E) e o *Package Explorer* (A). O usuário pode também filtrar (D) a informação apresentada simultaneamente pelas visões.

Figura 4

IMAGEM DO *SOURCEMINER*



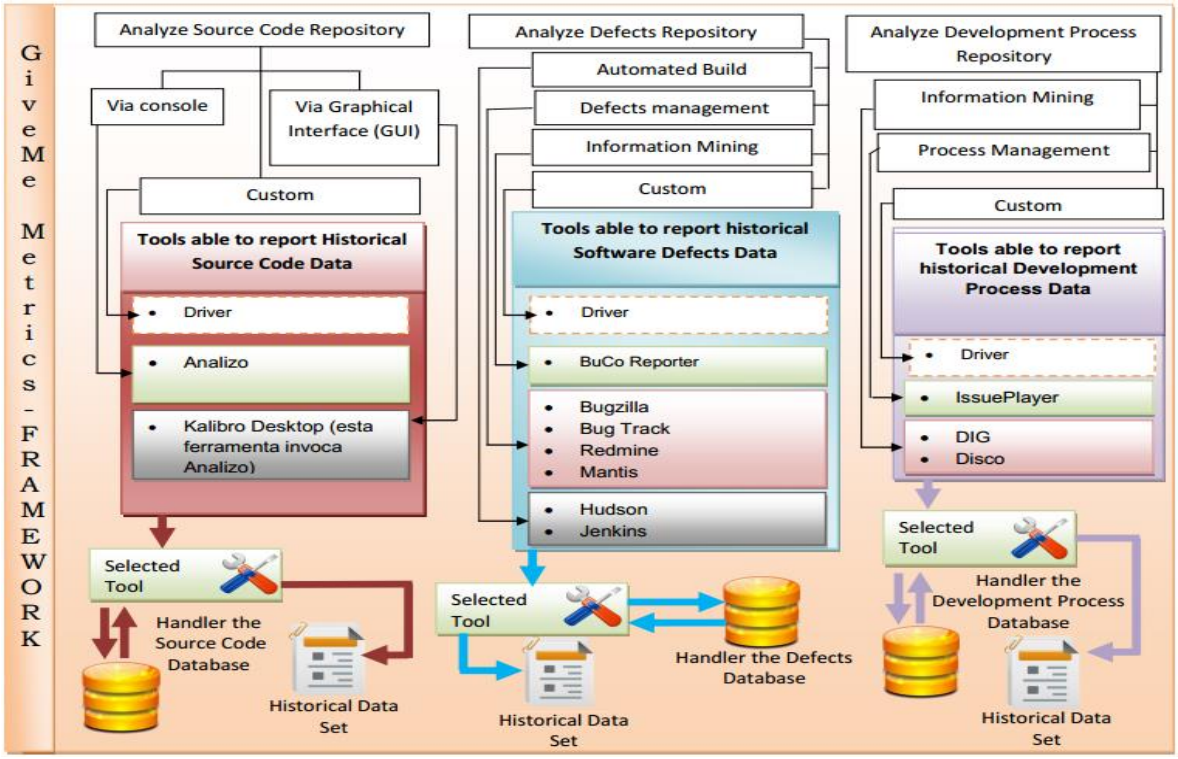
Fonte: (http://www.sourceminer.org/)
Data: 2010.

- GiveMe Metrics: é uma ferramenta que tem como proposta suprir a necessidade de extrair dados históricos de três diferentes tipos de repositórios, repositório de código-fonte, repositório de defeitos ou repositório de dados sobre o processo de desenvolvimento de software, seguindo critérios específicos. A maior vantagem na sua

utilização está nos parâmetros fornecidos para a escolha de ferramentas, dadas as características do repositório que se deseja manipular visando simplificar a escolha da que melhor se encaixa nas necessidades da tarefa de manipulação do repositório. A utilização do *framework* pode ser dividida em três diferentes cenários, permitindo ao usuário escolher entre três diferentes tipos de repositórios de dados para analisar.

Figura 5

IMAGEM DO *GIVEME METRICS*



Fonte: (http://givemeinfra.com.br/givememetrics.html)
Data: 2014.

CONCLUSÃO

Nos últimos anos, a importância da mineração de repositórios tem aumentado com a necessidade das empresas de tornarem-se mais orientadas a dados. Gerentes de projetos de *software* estão se interessando cada vez mais em transformar os dados que antes eram apenas armazenados nos diferentes repositórios de *software* em informações relevantes para fundamentar seu processo de tomada de decisão.

Do ponto de vista acadêmico, a mineração de repositórios provê dados empíricos para avaliar o uso de técnicas e tecnologias do ponto de vista histórico e vêm sendo cada vez mais usada pelos principais grupos de pesquisa em Engenharia de *Software*. A principal gama de dados de projetos de software tem surgido dos *softwares* livres.

Por se tratar de uma ferramenta nova, ainda existem muitos desafios para a mineração de repositórios de *software*. Dentre alguns desafios, destacam-se a grande quantidade de dados não estruturados e de fontes heterogêneas de informação. Outro ponto que podemos destacar está relacionado à escala. Alinhado ao fenômeno de *BigData* (termo que descreve o imenso volume de dados – estruturados e não estruturados), estudos atuais têm demandado escalabilidade das técnicas de coleta e análise dos dados para grande quantidade de informações. Podemos citar também, o desafio de facilitar a reprodução das pesquisas realizadas para aplicação em contextos/projetos diferentes. No que diz respeito à aplicação, o assunto tem sido foco de discussões na comunidade. O desafio é transformar pesquisas empíricas em soluções aplicáveis no dia a dia dos engenheiros de *software*, auxiliando o processo de desenvolvimento do *software* e no contínuo aumento de qualidade.

> O grupo de pesquisa LAPESSC (http://lapessc.ime.usp.br) do Departamento de Ciência da Computação do IME/USP realiza pesquisas de mineração de repositórios focando em estudos de evolução, manutenção e design de software, dependências de artefatos, métricas de software, análise da colaboração de desenvolvedores e software social.
> Atualmente o grupo vem desenvolvendo uma ferramenta chamada MetricMiner (http://www.metricminer.org.br), citada acima, cuja ideia é facilitar a vida de pesquisadores que precisam extrair e ligar informações oriundas de diversos repositórios de código.
> A mineração de repositórios de código veio para ficar e vai ajudar gerentes e desenvolvedores a tomarem decisões no dia a dia e pesquisadores a entenderem melhor a Engenharia de Software, acabando com muitos de seus mitos. (GEROSA, WIESE, OLIVA, ANICHE, 2015, p 23)

ABSTRACT

In the programming environment some developers are faced with difficult codes to be understood at the time of maintenance, since many of these codes are not maintained by those who programmed them. Often, these programmers are no longer part of the team. Tools are being developed to assist the software maintenance process, making life easier for programmers. Data mining is used by the Software Engineering in search patterns and trends that can assist in the construction, maintenance and evolution of software through software repositories that store information such as source code, version history (logs), reports errors (defects screening), among others. In this scientific, qualitative and bibliographical article, we will reveal how the use of data mining can assist in the development and maintenance of software.

Keywords: *Software Development, Software Repositories Mining Software Repositories.*

REFERÊNCIAS BIBLIOGRÁFICAS

AMO. Sandra de. **Técnicas de Mineração de Dados.** Uberlândia. (2004).

COLAÇO JR, Methanias. **Mineração de Repositórios de Software:** A Computação ajudando à Computação. SQL Magazine, v. 31, 2010. <http://www.devmedia.com.br/mineracao-de-repositorios-de-software-a-computacao-ajudando-a-computacao/17632#ixzz46qnbiJ40>.

COLAÇO JR, Methanias. **Identificação e Validação do Perfil Neurolinguístico de Programadores Através da Mineração de Repositórios de Engenharia de Software**. Salvador, 2011.

COSTA. Daniel Alencar da. **Avaliação da contribuição de desenvolvedores para projetos de software usando mineração de repositórios de softwares e mineração de processos.** Natal, 2013.

DANTAS, Cristine; MURTA, Leonardo; WERNER, Cláudia. **Odyssey-WI:** Uma Ferramenta para Mineração de Rastros de Modificação em Modelos UML Versionados. Simpósio Brasileiro de Engenharia de Software (SBES), Sessão de Ferramentas, Florianópolis, 2006.

GEROSA. Marco Aurélio. WIESE. Igor Scaliante; OLIVA. Gustavo Ansaldi; ANICHE. Maurício Finavaro, (2015), **Mineração de Repositórios de Software Livre.** Revista da Sociedade Brasileira de Computação. Ed 27, p19-24.

MACIEL, Igor Iure Santos. **Mineração de dados para descoberta de padrões de uso de uma ferramenta de visualização de *software*.** Itabaiana, 2010.

QUILICI-GINZALES. José Artur, ZAMPIROLLI. Francisco de Assis. **Sistemas inteligentes e Mineração de Dados.** Santo André, 2014.

SANTOS, Francisco R.; COLAÇO JR, Methanias; MENDONÇA, Manoel. **ChangeMiner:** um plug-in visual para auxiliar a manutenção de software. 2014.

SILVA, José Teodoro da. MinerAll: **Uma ferramenta para extração e mineração de dados de repositórios de software livre.** Campo Mourão, 2012.

SILVA. Maria Goreti Simão da**. Mineração de Repositórios de Software, Modelos de Previsão de Pedidos de Evolução de Software.** Lisboa, 2013.

SOKOL. Francisco. **MetricMiner:** uma ferramenta web de apoio à mineração de repositórios de software. São Paulo, 2012.

TAVARES. Jacimar F; DAVID. José Maria N.; ARAÚJO. Marco Antônio P.; BRAGA, Regina; CAMPOS. Fernanda. **GiveMe Metrics:** Um framework conceitual para extração de dados históricos sobre evolução de software, 2014.