GiveMe Views: uma ferramenta de suporte a evolução de software baseada na análise de dados históricos

Alternate Title: *GiveMe Views*: a support tool to software evolution based on historical data analysis

Jacimar F. Tavares
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
jacimar.tavares@ice.ufjf.br

José Maria N. David
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
jose.david@ufjf.edu.br

Marco Antônio P. Araújo
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF) / IF Sudeste MG
marco.araujo@ice.ufjf.br

Regina Braga
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
regina.braga@ufjf.edu.br

Fernanda Campos
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Juiz de Fora (UFJF)
fernanda.campos@ufjf.edu.br

Glauco Carneiro
Programa de Pós Graduação em Sistemas e Computação
Universidade Salvador (UNIFACS)
glauco.carneiro@unifacs.br

RESUMO

Este artigo apresenta a ferramenta *GiveMe Views*, capaz de indicar possíveis módulos e componentes que poderão ser impactados quando uma manutenção corretiva, adaptativa ou evolutiva é realizada. Fornece visualizações que permitem ao usuário obter informações sobre as relações existentes entre módulos e componentes baseando-se em análises estatísticas realizadas por um motor SAE (*Statistical Analysis Engine*) sobre dados históricos de software. Uma prova de conceitos foi realizada com o intuito de verificar a viabilidade da ferramenta em atividades de evolução de software em um contexto real de utilização. Ao final, os resultados da implantação da ferramenta em uma empresa de desenvolvimento de software são apresentados.

Palavras Chave

Evolução de Software, Visualização de Software.

ABSTRACT

This paper presents Give Me Views tool that can indicate possible modules and components that may be impacted when a corrective maintenance, adaptive or evolutionary is performed. Provides views that allow user to obtain information about the relationships between modules and components based on statistical analyzes performed by an SAE (Statistical Analysis Engine) on historical software data. A proof of concept was performed in order to verify the feasibility of the tool in software development activities in a real context of use. The tool deployment results in a software development company are presented.

Categories and Subject Descriptors

K.6.3 [Software Management]: *Software maintenance.*

General Terms

Management, Measurement, Design, Experimentation.

Keywords

Software Evolution, Software Visualization.

1. INTRODUÇÃO

Durante a evolução do software, alterações provenientes de manutenções realizadas podem ser observadas. Em [17] é definido que alterações no software constituem um ciclo de manutenção formado pelas fases:

- Requisição de mudanças: onde são definidas as mudanças a serem realizadas;
- Planejamento: se relaciona às atividades de (i) Compreensão do programa: na qual o software a ser alterado é analisado, buscando conhecimento sobre ele; e (ii) Análise do impacto: visa conhecer os riscos da efetivação da mudança, bem como o que será necessário alterar;
- Implementação da alteração: diz respeito à execução da alteração planejada. É formada pelas seguintes etapas: (i) Reestruturação para a mudança: contempla as alterações necessárias para reestruturar o sistema de modo a acomodar as novas implementações; (ii) Propagação da alteração: teoria fundamental neste trabalho, pois diz respeito a análise de componentes vizinhos a um componente alterado. Simplificando, cada vez que um componente é alterado, é dito que os componentes que possuem relação com ele (ou seja, vizinhos) devem ser verificados a fim de descobrir se alterações também serão necessárias neles. Alteração em um componente vizinho é denominada alteração secundária;
- Verificação e validação: verificação da alteração efetuada;

- Re-documentação: corresponde à adaptação da documentação existente ou à definição de nova documentação, objetivando registrar as alterações realizadas.

Este trabalho apresenta o relato do desenvolvimento da ferramenta *GiveMe Views*, um *plugin* para o ambiente de desenvolvimento *Eclipse*, que atua sobre projetos desenvolvidos em *Java*, visando apoiar a fase Implementação da alteração [17].

Criada para apoiar a análise visual de mudanças (apesar de não ser o objetivo deste artigo tratar sobre o tema visualização de software) realizadas em projetos de software da indústria, auxilia desenvolvedores nas tomadas de decisão sobre mudanças futuras. Os dados utilizados neste artigo, e para testes com o *GiveMe Views*, foram obtidos como resultado de parcerias estabelecidas com duas empresas de desenvolvimento de software de gestão empresarial. Uma delas utiliza um software próprio para a gestão de solicitações de mudança chamado *Service Center*. A outra utiliza o software *Mantis* [12]. Por questões de confidencialidade, a empresa que utiliza o *Service Center* será denominada neste artigo como empresa parceira 1. Já a que utiliza o *Mantis*, empresa parceira 2. Ambas as bases de dados dos citados softwares serão chamadas de repositórios de mudanças.

A motivação para o desenvolvimento da ferramenta surgiu a partir (i) da identificação de um problema no processo de manutenção da empresa parceira 1 e, (ii) uma particularidade encontrada no processo de registro e controle das solicitações de mudança, também na empresa parceira 1, ambos apresentados a seguir:

O problema: a empresa estava insatisfeita com o número de solicitações de mudanças reprovadas pelo setor de qualidade. As reprovações se davam (i) pela falta de controle sobre a evolução do software e (ii) a complexidade do projeto de código dos produtos. Ao longo dos anos e dos vários ciclos de manutenção, os produtos se afastaram de suas arquiteturas iniciais dado que não havia um controle sobre a evolução. Alterações eram realizadas sem o devido planejamento da evolução do projeto de código. Isso fez com que os projetos de códigos dos softwares se tornassem confusos e de difícil manutenção. Mediante isso, desenvolvedores que realizam as solicitações de mudanças têm dificuldade para identificar, por exemplo, quais componentes devem ser alterados quando outros componentes também forem alterados. Isso acontecia pois eles não possuíam nenhum tipo de solução (técnica, metodologia, ferramenta, *framework*, ou modelo) que fornecesse esse tipo de indicação.

A particularidade: está relacionada às informações de rastreabilidade que são adicionadas sempre que uma solicitação de mudança era finalizada, ou seja: módulos e componentes alterados. O termo módulo é designado para classificar: (i) softwares (produtos que a empresa comercializa), (ii) módulos (como módulo financeiro, módulo fiscal, entre outros) e (iii) pacotes de código (*packages*, tais como os criados em ambientes como *Netbeans* e *Eclipse*). Já o termo componente refere-se a: (i) classes ou arquivos de código alterados, (ii) formulários (tais como *jFrame* e *jDialog*), (iii) *DLLs* (bibliotecas de código e dados) [7] e (iv) *scripts* de banco de dados. A rastreabilidade é dita como as ligações existentes entre módulos e componentes que sofreram algum tipo de manutenção ao longo da resolução da solicitação de mudança. Permite que *links* sejam criados e mantidos entre artefatos gerados e alterados durante o ciclo de vida de um software [13]. Os módulos e componentes alterados são obtidos através dos sistemas de controle de versão que, tal como dito em [14], podem fornecer informações úteis para entender a evolução do software.

A empresa parceira 2 possuía a mesma dificuldade da empresa parceira 1 para estimar módulos e componentes que poderiam ser alterados sempre que um outro dado módulo ou componente fosse alterado. Em contrapartida, não possuía informações de rastreabilidade associadas às suas solicitações de mudança.

As informações de rastreabilidade, adicionadas às respectivas solicitações de mudanças, na empresa parceira 1, não eram usadas para ajudar na tomada de decisões futuras, mas somente para registro do que foi alterado pelo desenvolvedor, caso necessária uma consulta futura.

O problema a ser tratado neste artigo é a falta de uma ferramenta que apoie a análise da evolução de software utilizando dados históricos de solicitações de mudanças e informações de rastreabilidade de software e que atue como um *plugin* para o ambiente Eclipse. A principal contribuição está na definição de uma ferramenta livre que possibilite que desenvolvedores possam estimar os impactos que alterações no código fonte podem ocassionar, considerando para isso, dados históricos de todas as versões de um software analisado e cálculos estatísticos automaticamente gerados pela ferramenta, diferentemente de outras ferramentas de mesmo propósito que levam em consideração somente análises realizadas sobre o projeto de código, estaticamente e sobre uma única versão.

1.1 Metodologia

Inicialmente foi utilizado o *framework GiveMe Metrics* [9], parte inicial desta pesquisa, para conduzir a extração dos dados históricos de solicitações de mudanças dos repositórios de mudanças das empresas parceiras 1 e 2, respectivamente. Com os dados em mãos, foi conduzido um processo de análise junto ao Departamento de Estatística da Universidade Federal de Juiz de Fora (reunindo pesquisadores das áreas de Estatística e Ciência da Computação). Esse processo permitiu a implementação de um motor estatístico, chamado SAE - *Statistical Analysis Engine* (subseção 3.3). O SAE possibilitou ao *GiveMe Views* indicar quais as chances estatísticas de um dado módulo, ou componente, ser impactado quando outro módulo ou componente fosse alterado. Posteriormente, foi desenvolvida a ferramenta *GiveMe Views* que foi avaliada junto a empresa parceira 1. O objetivo deste artigo é apresentar a ferramenta *GiveMe Views*, bem como sua atuação na resolução do problema encontrado nas empresas parceiras 1 e 2.

Este artigo está dividido da seguinte forma: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 introduz os princípios e o projeto da *GiveMe Views*. A Seção 4 apresenta a avaliação da primeira versão da ferramenta, realizada através de uma prova de conceitos. A seção 5 apresenta o cenário atual do *GiveMe Views* e suas principais funcionalidades implementadas. A seção 6 apresenta as considerações finais, os resultados obtidos e os trabalhos futuros.

2. TRABALHOS RELACIONADOS

Ferramentas para apoiar o processo de manutenção de software têm sido propostas na literatura técnica com frequência. Nesse contexto, algumas tratam de técnicas de visualização [3] como uma forma de tornar esse processo mais efetivo ao apresentar uma abordagem semiestruturada que permite a geração de visões gráficas, em forma de diagramas de classes, que foram diretamente geradas a partir da análise do projeto de código fonte de um software . O objetivo é obter a visão geral da arquitetura e dos relacionamentos entre as entidades envolvidas. Entretanto, os

dados históricos de software não são utilizados para o entendimento das relações entre as entidades, através apenas da geração de visões. Já em [4] é argumentado que, devido às mudanças que um software sofre durante seu ciclo de vida, é comum que, em alguns casos, suas arquiteturas se afastem do projeto original e até mesmo se tornem degradadas. Nesse sentido, é apresentada a ferramenta *Buch* que permite a geração de visualizações em forma de grafos a partir da arquitetura existente, utilizando, para isso, técnicas de agrupamento. Os grafos de relacionamento gerados por B*uch*, no entanto, não dão significância estatística à relação existente entre as entidades, se atendo apenas em apresentá-los.

SQuAVisiT [16] é um *framework* constituído por ferramentas e *plugins* que se conectam a repositórios e geram diferentes tipos de visualizações mediante a realização de análises. Essa proposta se difere das apresentadas em outros trabalhos relacionados por permitir a análise de código de diferentes linguagens. Entretanto, repositórios de dados históricos não são analisados no intuito de se obter informações sobre os relacionamentos entre as entidades em um contexto histórico.

3. *GIVEME VIEWS*: PRINCÍPIOS E PROJETO

A premissa para utilização do *GiveMe Views* é possuir dados históricos de software que possam ser extraídos de três diferentes tipos de repositórios. São eles: (i) de código fonte, (ii) de solicitação de mudanças e (iii) de processos de desenvolvimento de software. Por enquanto, *GiveMe Views* é capaz de analisar apenas dados provenientes dos repositórios (i) e (ii), como por exemplo dados de solicitações de mudanças e dados de *logs* de repositórios de código fonte.

Para que os dados pudessem ser extraídos das empresas parceiras, foi utilizado o *GiveMe Metrics Framework* [9]. Trata-se de um *framework* conceitual para extração de métricas e dados históricos sobre evolução de software em três diferentes tipos de repositórios.

A Figura 1 apresenta uma visão geral sobre a proposta de pesquisa, exibindo onde a ferramenta *GiveMe Views* atua. Mostra as parcerias realizadas (Empresas parceiras 1 e 2) que forneceram acesso aos seus repositórios de mudanças que foram analisados. O *framework* GiveMe Metrics foi utilizado e o conjunto de dados de saída da extração foram armazenados no repositório de dados históricos da ferramenta *GiveMe Views*, denominado *GiveMe Repository*.

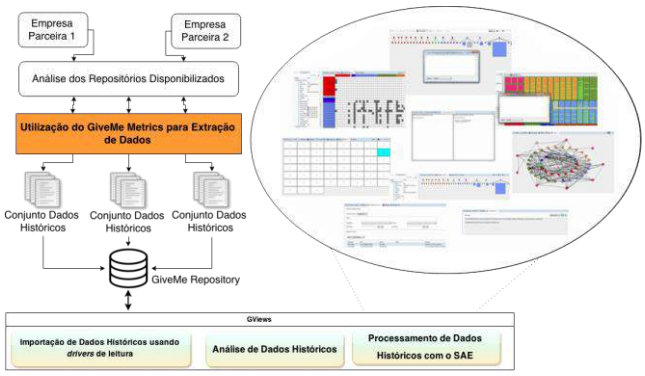

Figura 1. Visão geral da solução

O conjunto de dados de saída é a entrada para a *GiveMe Views*, que o analisa, processa e mantém as informações resultantes para serem acessadas pelas visualizações, como ilustrado na elipse da figura.

Para que os dados mantidos no *GiveMe Repository* possam ser manipulados pelo *GiveMe Views*, foi necessária a implementação de *drivers* de leitura (subseção 3.2) que atuam na importação.

Os dados históricos, extraídos das empresas, foram analisados por um conjunto de pesquisadores. Através desta análise observou-se que os dados referentes a uma solicitação de mudança continham informações de manipulação de um ou mais componentes, no caso dos dados da empresa parceira 1. Também foi observado que um dado componente do software era sempre modificado quando outro componente também era modificado. Isso pode ser um indício de um acoplamento entre eles, de forma que alterações em um poderiam impactar diretamente no outro [15]. Observando essa questão, foi desenvolvido o motor SAE (subseção 3.3) e posteriormente o *GiveMe Views*.

3.1 *Drivers* de Leitura

Um importante conceito implementado no *GiveMe Views* é o de *driver*. São implementações que permitem a manipulação dos dados históricos independente do formato, que está restrito aos suportados pelo *GiveMe Metrics*: *.xml*, *.txt*, *.html*, *.csv* e *.xls*. Um conjunto de dados históricos pode ser formado, por exemplo, por um arquivo *.csv* e um *.txt*, ou somente um *.xls*.

Um *driver* é implementado para ler conjuntos de dados exportados por cada uma das ferramentas que compõem o *framework GiveMe Metrics*. Na versão atual do *GiveMe Views*, há suporte para leitura de dados exportados pela ferramenta Mantis [12] (como é o caso da empresa parceria 2), e dados exportados do repositório customizado da empresa parceira 1, o *Service Center*. É importante destacar que *GiveMe Views* suporta o desenvolvimento de *drivers* para leitura de dados históricos provenientes de outras fontes, e não somente para um conjunto restrito de ferramentas suportadas pelo *framework GiveMe Metrics*.

3.2 *GiveMe Repository*

É um repositório de dados históricos, e também o nome do *plugin* desenvolvido para automatizar a sua construção. A função do *plugin* é eliminar a responsabilidade do usuário na criação da estrutura de pastas do repositório. Além disso, implementa toda a lógica de manipulação do *GiveMe Repository*. A Figura 2 ilustra a estrutura do repositório *GiveMe Repository* gerada pelo *plugin*.

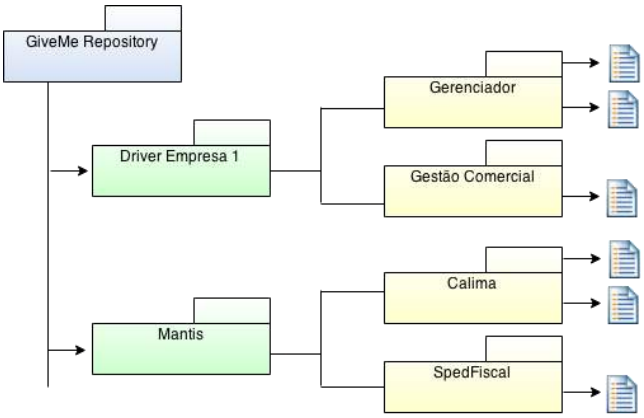

Figura 2. Estrutura do repositório gerado pelo *plugin GiveMe Repository*

Conforme a Figura 2 o espaço para armazenamento de dados para dois *drivers* diferentes são destacados: o Driver Empresa 1 e o *driver* desenvolvido para a ferramenta *Mantis*.

Dentro de cada espaço, há outros espaços que são específicos para cada projeto que se tenha dados históricos para analisar. No espaço para o *Driver Empresa 1* há dados históricos de dois projetos: Gerenciador e Gestão Comercial. No espaço *Mantis* há dados históricos de outros dois projetos: Calima e *SpedFiscal*. Todos os quatro projetos mencionados são reais e foram disponibilizados através da parceria com as empresas 1 e 2, respectivamente.

Para gerar a estrutura do *GiveMe Repository*, o usuário deve apenas informar o caminho físico de sua escolha para que ele seja criado. Como resultado, todas as vezes que *GiveMe Views* necessitar analisar um conjunto de dados históricos, pertencentes a um projeto, o repositório será localizado.

3.3 SAE - *Statistical Analysis Engine*

O motor SAE foi concebido basicamente a partir de uma questão a ser investigada: historicamente, qual o percentual em que um dado componente foi modificado sempre que outro componente também foi modificado? Para responder esta questão foram consideradas teorias estatísticas de Distribuição de Frequência Simples e Conjunta [5]. A Figura 3 ilustra a definição do SAE.

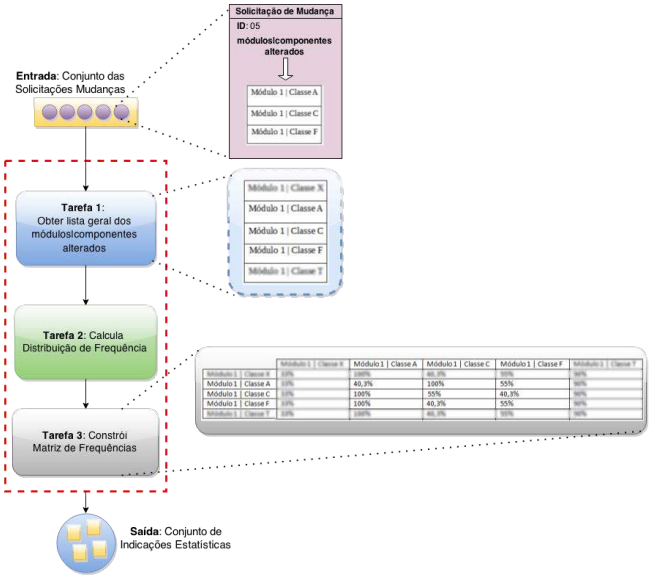

Figura 3. Sequência de tarefas do SAE

Os módulos da Figura 3 que estão dentro da linha pontilhada representam o SAE. Como entrada, tem-se um conjunto de solicitações de mudanças. Cada solicitação é única, e possui um conjunto de informações, como classes e módulos alterados para resolução daquela solicitação de mudança, como pode ser visto no *zoom* aplicado na solicitação de mudança de ID 05.

A primeira tarefa do SAE é obter a lista geral dos módulos e componentes alterados, sem repetições entre todos os conjuntos de informações. Em seguida, são executados os cálculos estatísticos (Tarefa 2) e construída a matriz que possui as frequências calculadas (Tarefa 3).

O *zoom* na Tarefa 3 mostra um exemplo de matriz com as estatísticas obtidas. Como saída do SAE, tem-se um conjunto de estatísticas sobre qualquer método de um software, através do qual é possível avaliar, por exemplo, o impacto de uma alteração em um dado método que se deseja dar manutenção.

Para o caso da empresa parceira 2, (e de outras empresas que, porventura, não tenham informações de rastreabilidade associadas diretamente às solicitações de mudança), a ferramenta *GiveMe Views* prevê a leitura de outro conjunto de dados como entrada, que são arquivos de *log* de histórico de *commits* obtidos de repositórios de código fonte. Atualmente há suporte para repositórios SVN (*SubVersion*) [10] e GIT [11].

Após o desenvolvimento do SAE, foi construída a ferramenta *GiveMe Views*, a qual implementará o SAE em um módulo específico, com suas respectivas visualizações. A subseção seguinte apresenta a arquitetura do *GiveMe Views*.

3.4 Arquitetura do *GiveMe Views*

A arquitetura foi projetada com dois objetivos principais: (i) permitir a execução do processamento estatístico dos dados históricos ao invocar o motor SAE, e (ii) apoiar a persistência de informações originadas do processamento estatístico com o SAE, permitindo que elas sejam acessadas por visualizações ou por outras ferramentas, posteriormente integradas à *GiveMe Views.* A Figura 4 apresenta a arquitetura, e a relação entre os módulos implementados.

Uma entrada para a ferramenta *GiveMe Views* é a indicação de um módulo/componente que se deseja dar manutenção. Para isto, basta selecionar o componente que a ferramenta capturará o seu nome e o módulo ao qual ele pertence, concatenando essas informações separadas pelo caracter *pipe* " | ", conforme ilustrado na Figura 5.

Em seguida, os dados históricos referentes ao projeto alvo da manutenção são importados do *GiveMe Repository* e filtrados, para a análise estatística, bem como para o cálculo das métricas disponíveis. A informação gerada é então persistida, disponibilizada para exportação ou para acesso de visualizações e de ferramentas, via provedor. Para executar a ferramenta *GiveMe Views*, basta selecionar o projeto que se deseja dar manutenção e, com o botão direito do mouse, acessar a opção *Visualizer with GiveMe Views.*

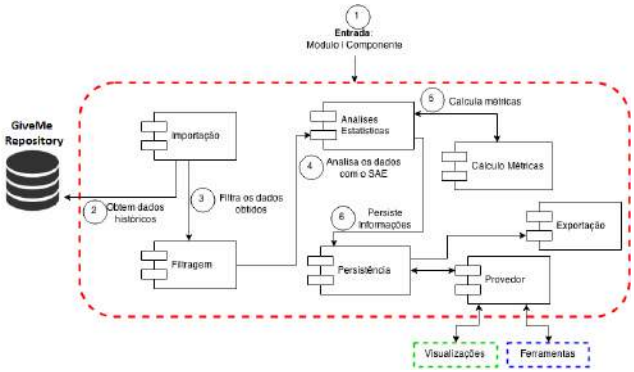

Figura 4. Arquitetura *GiveMe Views*

- Importação: responsável por selecionar no repositório o conjunto de dados históricos referentes ao projeto em manutenção;
- Filtragem: realiza a filtragem dos dados importados, desconsiderando, por exemplo, dados de solicitações de mudanças que não foram concluídas e, portanto, não afetaram

nenhum trecho de código do projeto em manutenção. Portanto, são irrelevantes no processamento estatístico;

- Análises Estatísticas: responsável por implementar o SAE. Além disso, é responsável por formatar as informações nos formatos necessários para os diferentes tipos de visualização disponíveis no *GiveMe Views*;
- Cálculo Métricas: calcula as métricas sobre dados históricos que são disponíveis no *GiveMe Views*. A lista completa das métricas está disponível em [2].
- Persistência: módulo responsável por manter em memória todas as informações estatísticas processadas, bem como as métricas e as informações gerenciais da ferramenta;
- Provedor: único ponto pelo qual visualizações e ferramentas a serem integradas ao *GiveMe Views* podem obter informações sobre os dados históricos;
- Exportação: módulo responsável por implementar os relatórios disponíveis para exportação.

4. AVALIAÇÃO DA FERRAMENTA

Esta seção descreve uma prova de conceitos com o objetivo de analisar a viabilidade da ferramenta *GiveMe Views* para apoiar a avaliação de impacto em atividades de manutenção de software. O objetivo é apresentado segundo a abordagem *Goal/Question/Metric* (GQM) [1]: "***Analisar*** a ferramenta *GiveMe Views* ***com a finalidade de*** verificar a viabilidade de uso na tarefa de identificação dos possíveis módulos e componentes que serão impactados quando uma dada manutenção corretiva, adaptativa ou evolutiva for efetuada em um componente de um módulo ***com respeito*** a alterações no projeto de **código *do ponto de vista do*** desenvolvedor que realizará a manutenção ***no contexto de*** evolução de software".

A prova de conceito contou com a participação do gerente de projetos da empresa parceira 1. Após a utilização do *GiveMe Metrics* [9] para extrair os dados do repositório de solicitação de mudanças, a ferramenta *GiveMe Views foi* apresentada por meio de um treinamento realizado em um único dia. Posteriormente, o gerente analisou o repositório de mudanças e selecionou uma que envolvia uma manutenção corretiva já concluída. O objetivo é usar o *GiveMe Views* para verificar a sua capacidade de indicar os mesmos módulos e componentes alterados na solicitação de mudança selecionada. A solicitação selecionada foi a manutenção corretiva realizada no Componente *clsCalculoNota*, pertencente ao módulo *DLL – Calculos*, uma vez que se tratava de uma solicitação concluída recentemente pelo próprio gerente de projetos. Nesse contexto, as seguintes hipóteses foram definidas:

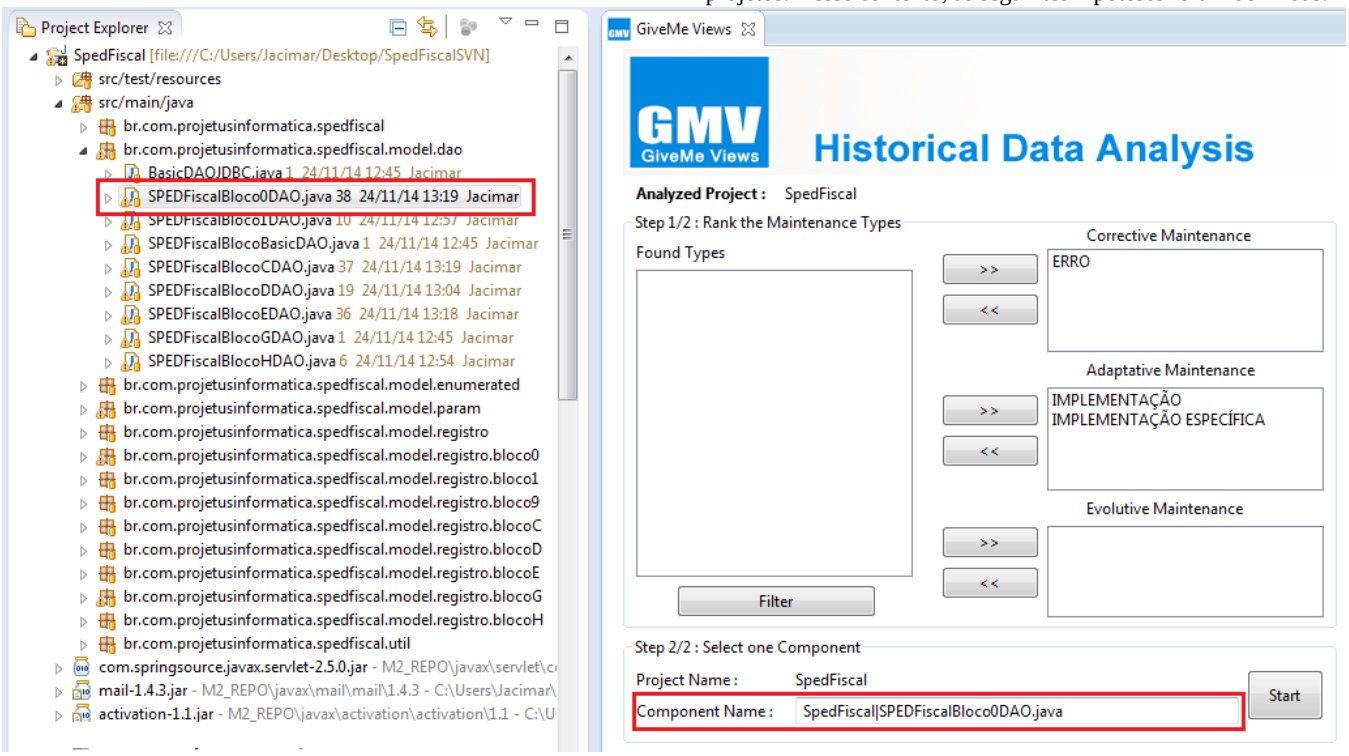

Figura 5. Tela principal do GiveMe Views e a seleção de um componente para verificação do impacto de mudança

- H_{nula}: a ferramenta *GiveMe Views* não é capaz de apoiar a identificação de quais componentes e módulos poderão ser impactados quando uma dada manutenção corretiva for realizada no componente *clsCalculoNota*;
- $H_{alternativa}$: a ferramenta *GiveMe Views* é capaz apoiar a identificação de quais componentes e módulos poderão ser impactados quando uma dada manutenção corretiva for realizada no componente *clsCalculoNota*.

A Figura 6 mostra a visualização *Tree View* [6] gerada a partir da execução da ferramenta *GiveMe Views*, considerando a necessidade de verificar quais componentes poderão ser impactados quando uma manutenção for realizada no componente *clsCalculoNota*, pertencente ao módulo *DLL - Cálculos.*

A porcentagem entre parênteses é o resultado do cálculo estatístico realizado pela ferramenta *GiveMe Views*. Assim, *fNotaEntrada*, componente pertencente ao módulo Gerenciador, foi o componente com maiores chances estatísticas de ser impactado quando uma manutenção corretiva for realizada no componente *clsCalculoNota*, pertencente ao módulo DLL – Cálculos (12,12% de chance).

Ainda, através da Figura 6, é possível perceber que a ferramenta pode indicar impactos não somente em componentes de um mesmo módulo, mas também entre outros módulos que são interligados. Como exemplo, há mais chances estatísticas do componente *fNotaEntrada*, pertencente ao Módulo *Gerenciador*, ser impactado do que o componente *mdlTipos* (10,60%) pertencente ao mesmo módulo que se deseja alterar.

A prova de conceito apresentou evidências iniciais de que a ferramenta pode apoiar a identificação de componentes que de fato foram modificados, no caso selecionado pelo gerente de projetos da empresa parceira, aceitando-se a hipótese alternativa da questão de pesquisa. Entretanto, avaliações adicionais devem ser realizadas no sentido de evidenciar situações nas quais um único experimento não permitiu revelar.

Como ameaças à validade do experimento podemos citar o fato da avaliação ter se baseado em uma manutenção realizada pela empresa parceira 1. Caso ela não tenha impactado os componentes de fato necessários, a indicação dada pelo *GiveMe Views* poderia não corresponder à realidade.

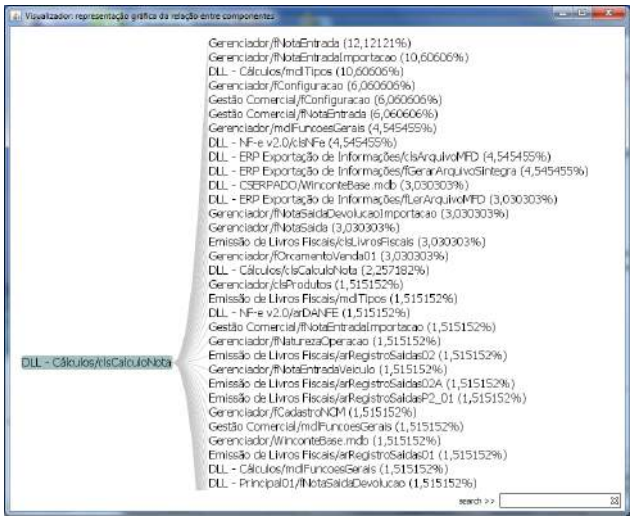

Figura 6. Visualização *Tree View*

5. GIVEME VIEWS: CENÁRIO ATUAL

Após a condução da prova de conceitos, foram realizadas implementações e adaptações nas funcionalidades do *GiveMe Views* visando adequar a ferramenta a uma nova necessidade: fornecer indicações de métodos que poderão ser impactados quando um dado método de uma classe fosse alterado, e não somente indicar classes ou arquivos de código, formulários (tais como *jFrame* e *jDialog*), *DLLs* (bibliotecas de código e dados) e *scripts* de banco de dados, que podem ser impactados quando uma dada manutenção fosse realizada.

A adaptação se tornou necessária porque, em alguns projetos que serão alterados, indicar apenas as classes que poderão ser impactadas com mudanças em classes relacionadas pode não ser suficiente, por exemplo. A existência de classes, como algumas encontradas em projetos na empresa parceira 1, que possuem mais de 15.000 linhas de código evidenciaram esta necessidade. Indicar que uma classe dessas possa ser impactada se uma alteração for realizada em outra dada classe pode ser uma informação muito vaga. As novas implementações e adaptações realizadas visaram apenas integrar a ferramenta *GiveMe Views* com a ferramenta *GiveMe Trace*, apresentada na seção 5.1.

5.1 Integração *GiveMe Views* e *GiveMe Trace*

GiveMe Trace [15] é uma ferramenta de apoio a rastreabilidade de software. Trata-se de um *plugin* para o ambiente *Eclipse*. Atua gerando dados históricos sobre a rastreabilidade de software ao se conectar automaticamente a repositórios SVN e GIT. É capaz de reportar informações em dois formatos diferentes, basicamente:

- *Arrays* de *strings:* dado um número de *commit* (nome dado ao envio alterações de código ao repositório de código fonte) ou um intervalo de *commits* (ou seja, um conjunto de números que representam diferentes *commits* realizados para um projeto), *GiveMe Trace* retorna uma lista dos métodos alterados por *commit*. Cada *string* retornada segue o mesmo padrão: nomeDaClasse.java|nomeDoMetodo (nome da classe é concatenado com um *pipe* e com o nome do método alterado). Um *array* retornado contém um conjunto de métodos de todas as classes de código fonte que foram modificadas num dado *commit*, ou um intervalo deles.
- Arquivos texto (txt): dado um número ou intervalo de *commits*, *GiveMe Trace* retorna um arquivo de *log* de alterações realizadas no repositório de código fonte, idêntico ao fornecido pelo próprio gerenciador de código fonte. A diferença consiste na granularidade das informações. A Figura 7, extraída de [15], mostra um exemplo de arquivo de *log* gerado na análise dos *commits* de número 19 e 20 (informados pelo usuário da ferramenta). Nela é possível ver que todas a modificações realizadas estão registradas no nível de métodos alterados, separados das classes as quais pertencem por um *pipe*.

As informações geradas pela ferramenta *GiveMe Trace* não podem ser geradas nativamente pelos sistemas de controle de versões de código fonte (SVN e GIT). Através dessas informações a ferramenta *GiveMe Views* passou a realizar o processamento estatístico com o SAE, considerando um nível de granularidade mais baixa, ou seja, no nível de método.

```
Commit: 19
Autor: USUARIO
Date: Tue Nov 11 13:35:37 BRST 2014

 Mensagem: final da implementação dos Ranges;
Resolvi warnings;

Fechamento do projeto;

M /givemetrace/src/givemetrace/implementations/TraceGit.java|preCoreToLogMethod
M /givemetrace/src/givemetrace/implementations/TraceGit.java|preCoreToLogClass
M /givemetrace/src/givemetrace/implementations/TraceGit.java|preCoreToArrayMethod
M /givemetrace/src/givemetrace/implementations/TraceGit.java|preCoreToArrayClass
M /givemetrace/src/givemetrace/implementations/TraceGit.java|getCountCommit

Commit: 20
Autor: USUARIO
Date: Tue Nov 18 18:25:43 BRST 2014

 Mensagem: tratamento dos url nos métodos do provider para evitar exception por
 não reconhecer que o url é um repositorio

M /givemetrace/src/givemetrace/views/vpGiveMeTraceMain.java|createPartControl
```

Figura 7. Trecho do arquivo de rastreabilidade

A seção 5.2 apresenta o conjunto de recursos e visualizações desenvolvido para o *GiveMe Views*, já considerando a integração com a ferramenta *GiveMe Trace.*

5.2 Recursos e visualizações

Atualmente, *GiveMe Views* disponibiliza recursos e visualizações que visam apoiar desenvolvedores na tomada de decisão em atividades de evolução de software. Nesta subseção serão apresentadas as três principais visualizações disponíveis. Para

executar o *GiveMe Views*, foram utilizados dados históricos do projeto *SpedFiscal*, cedido pela empresa parceira 2, que é um sistema para apoiar a gestão empresarial.

A Figura 8 mostra a *Tree View* criada após a seleção do método *getRegistro0200*, (pertencente à classe SPEDFiscalBloco0DAO. java), para análise de impacto, bem com os métodos que se relacionam com ele e as chances estatísticas, historicamente falando, de sofrerem impacto com uma manutenção.

Além de exibir os métodos que podem ser impactados quando o método selecionado for modificado, a visualização *Tree View* ainda exibe um número que representa a porcentagem de chances de isso ocorrer considerando o histórico de alterações, calculado pelo SAE.

Figura 8. Visualização *Tree View*

Assim como a *Tree View*, a visualização *Deep View* também exibe os métodos que podem ser impactados quando um dado método for alterado e a porcentagem estatística de um impactar o outro, historicamente, de forma ordenada pela maior porcentagem estatística. Entretanto, *Deep View* possui um diferencial: é capaz de analisar os impactos em diferentes níveis de profundidade. A Figura 9 mostra os métodos que podem ser impactados caso uma alteração seja feita no método *getRegistro0200*, pertencente a classe SPEDFiscalBloco0DAO. Numa atividade de manutenção, o desenvolvedor que modificasse o método *getRegistro0200* (método pai), teria que verificar, em todos os métodos que se relacionam com ele (filhos), se alguma alteração será necessária.

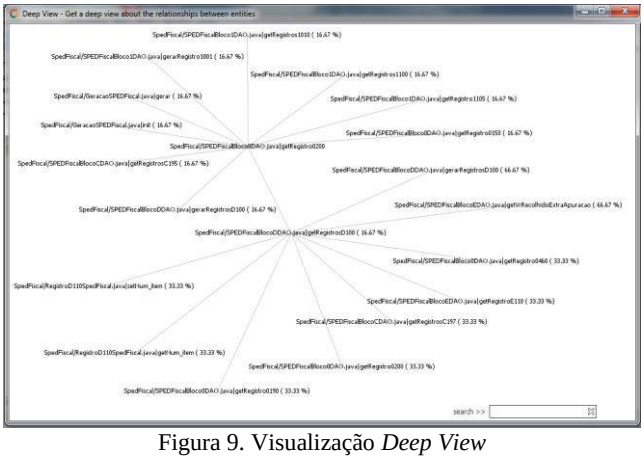

Figura 9. Visualização *Deep View*

Caso um dos filhos seja alterado, um novo ciclo se iniciará, sendo necessário verificar se os filhos do filho modificado foram impactados. Para analisar os métodos que podem ser impactados quando um método filho for alterado, basta clicar com o botão direito do mouse sobre um método que a visualização *Deep View* automaticamente irá exibir os filhos.

Durante a realização de uma atividade de manutenção, o desenvolvedor pode se deparar com questões de tempo que o impeça de verificar todos os métodos indicados pelo *GiveMe Views* como possíveis alvos de manutenção. Nesse contexto, o que se pode fazer é considerar aqueles com maiores chances estatísticas, e começar por eles. Além das porcentagens estatísticas, tem-se outro recurso que pode ser considerado, que está disponível na visualização *Modules and Components View* (Figura 10). A primeira informação que se tem é que o projeto *SpedFiscal* foi alvo de 468 manutenções, sendo que delas, 6 afetaram o componente *getRegistro0200*.

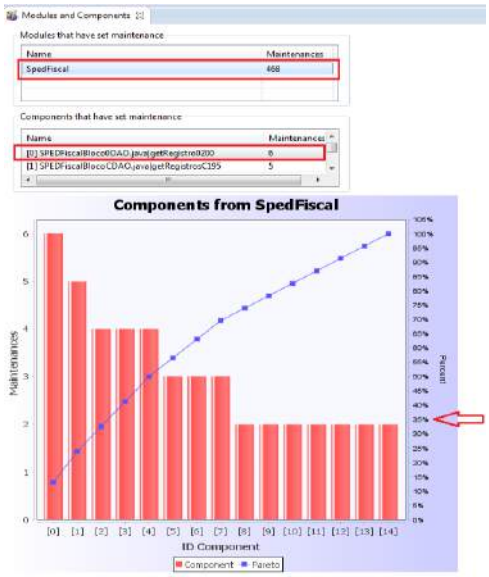

Figura 10: Visualização *Modules and Components View*

O gráfico exibido à direita mostra o número de manutenções por componente e uma curva de Pareto, que indica que 35% das manutenções ocorreram nos métodos representados pelos *ids* 0, 1 e 2, tornando-os boas opções de verificação de manutenção inicial.

5.3 Resultados Obtidos

Como resultados da implantação da ferramenta na empresa parceira 1 tem-se:

Cálculo de métricas: um dos resultados deste trabalho é que *GiveMe Views* fornece um meio para calcular um conjunto de métricas, cujos valores antes eram desconhecidos, proporcionando acesso a informações que podem auxiliar no processo de tomada de decisão, por exemplo: foi possível à empresa parceira 1 perceber que o número de desdobramentos (reaberturas de solicitações de mudança) abertos é maior que o número de solicitações de mudanças. Isso, na prática, indica que a empresa gasta mais recursos para solucionar problemas que surgem após a modificação de um componente se comparado aos recursos gastos com solicitações provenientes de usuários. Isso pode mostrar que o nível de acoplamento do sistema compromete a manutenção, indicando a necessidade de efetuar melhorias no projeto dos seus produtos.

Atualização da equipe de suporte: a métrica Porcentagem Acerto de Estimativa de um Módulo permitiu à empresa descobrir que o setor de suporte não estava sendo eficiente na tarefa de identificação do trecho que deverá ser alterado, mediante a

solicitação de mudança do cliente. Assim que um chamado é aberto, a equipe de suporte cadastra o possível módulo e o componente que deverá ser alterado. *GiveMe Views* mostrou que, em três anos de manutenções, apenas 35% das vezes essa informação foi corretamente cadastrada. Nesse caso, a equipe de suporte necessitará passar por novo treinamento para conhecer a frequência de modificação dos produtos ao longo do tempo. Cabe ressaltar que, tal métrica pode não se aplicar a outros contextos de outras empresas, dado a necessidade de se possuir dados históricos que permitam o cálculo estatístico.

Apoio ao setor de qualidade: o processo de mudanças da empresa parceira 1 consiste na abertura de uma solicitação de mudança, na sua execução e verificação da mesma. A etapa de verificação é de responsabilidade do setor de qualidade. Ao receber uma solicitação de mudança para a correção de um defeito, por exemplo, deve-se, além de verificar se o defeito foi corretamente solucionado, verificar se novos defeitos não foram inseridos em outros componentes acoplados. Com *GiveMe Views*, o setor de qualidade passou a ter uma referência que indica os pontos que devem ser verificados, facilitando assim o trabalho antes feito manualmente e baseado na experiência de profissionais do setor de qualidade.

Percepção de uso da ferramenta: inicialmente, *GiveMe Views* foi testada pelo gerente de projetos da empresa parceira 1, que assimilou de imediato suas funcionalidades e utilização. Foi reportado que a ferramenta é de fato útil na identificação de possíveis pontos que poderão ser impactados dada uma solicitação de mudança, e que seu uso ajudará a reduzir o tempo atualmente gasto na identificação de possíveis pontos que serão impactados, bem como a avaliação do que foi alterado.

6. CONSIDERAÇÕES FINAIS

O problema abordado neste artigo não se aplica apenas ao contexto das empresas parceiras 1 e 2. De uma forma geral, empresas que não possuem documentação atualizada da arquitetura de seus produtos, bem como informações sobre seu projeto de código, podem ser potenciais usuárias da ferramenta *GiveMe Views*. Como contribuições tem-se o apoio prestado pela ferramenta *GiveMe Views* à manutenção de software, dado que ela proporciona uma visão gráfica e geral sobre os relacionamentos entre módulos e componentes, sem que seja necessário acesso ao código fonte, pois a ferramenta leva em consideração somente dados históricos em suas análises.

As limitações da ferramenta *GiveMe Views* são aquelas impostas pelo conjunto de dados históricos que são analisados. Caso o conjunto de dados de entrada seja ruim, as análises podem não ser precisas e as indicações podem ser irrelevantes. Um conjunto de dados é dito ruim se as modificações realizadas no código fonte de um software em manutenções anteriores não tiverem impactado métodos que realmente têm algum tipo de relação. Nesse caso, *GiveMe Views* indicaria pontos a serem alterados que podem não ter relação de fato com o componente a ser alterado, mesmo que com chances estatísticas pequenas de impacto.

Como trabalhos futuros, *GiveMe Views* será integrada à ferramenta *SourceMiner* [8] que provê a visão dos relacionamentos considerando o contexto do código fonte fornecendo, portanto, a visão do código no momento atual do software, isto é, considerando sua versão mais recente. A integração proporcionará ao usuário uma troca de contexto em tempo de execução, permitindo analisar a relação entre módulos e componentes do ponto de vista do código fonte e do ponto de vista histórico, baseado em estatística, como o fornecido pelo *GiveMe Views.* Através desta integração busca-se avaliar a forma pela qual as atividades de compreensão podem auxiliar na Evolução de Software.

7. AGRADECIMENTOS

À CAPES, à FAPEMIG e ao CNPq pelo apoio financeiro, ao professor José Antônio Teodoro, do Departamento de Estatística da Universidade Federal de Juiz de Fora e às empresas parceiras.

8. REFERÊNCIAS

[1] Basili, Victor R. Caldiera, Gianluigi H. Rombach, Dieter. (1994). “The Goal Question Metric Approach”. Chapter in Encyclopedia of Software Engineering, Wiley, 1994.

[2] *GiveMe Views:* http://givemeinfra.com.br/givemeviews. Acessado em jun de 2014.

[3] Haitzer, Thomas. Uwe Zdun. “Semi-automated architectural abstraction specifications for supporting software evolution”. Science of Computer Programming, Vol. 90, Part B, 2014.

[4] Mancoridis, S.; Mitchell, B.S.; Chen, Y.; Gansner, E.R. "Bunch: a clustering tool for the recovery and maintenance of software system structures," (ICSM '99) Proc. IEEE International Conference on Software Maintenance, 1999.

[5] Meyer, Paul L. “Probabilidade – Aplicações à Estatística”. 2ª ed. LTC – Livros Técnicos e Científicos Editora Ltda. 1993.

[6] Prefuse: http://www.prefuse.org/. Acessado em jan de 2014.

[7] Microsoft DLL: http://support.microsoft.com/kb/815065/pt-br. Acessado em 06 fev de 2015..

[8] Carneiro, G. F. “Sourceminer: um ambiente integrado para Visualização multi-perspectiva de software”. 230 f. Tese (doutorado) – Universidade Federal da Bahia, Inst. Matemática, Doutorado em Ciência da Computação. 2011.

[9] Tavares, J., David, J. M. N., Araújo, M. A.P., Braga, R., Campos, F. C. A. (2014), “GiveMe Metrics – Um framework conceitual para extração de dados históricos sobre a evolução do software”. X Simpósio Brasileiro de Sistemas de Informação (SBSI), Londrina/PR.

[10] SUBVERSION: https://subversion.apache.org/ Acessado em 10 de fev de 2015.

[11] GIT: http://git-scm.com/. Acessado em 10 de fev de 2015.

[12] Mantis Bugtracker: https://www.mantisbt.org/. Acessado em 06 de fev de 2015.

[13] Walters, B.; Shaffer, T.; Sharif, b; Kagdi, H. Capturing software traceability links from developers' eye gazes. Proc. 22nd Int. Conf. on Program Comprehension. ACM, 2014.

[14] Mohan, K.; Xu, P.; Cao, L.; Ramesh, B. Improving change management in software development: Integrating traceability and software configuration management. Decision Support Systems, v.45, n.4, p. 922 - 936, 2008.

[15] Lélis, C. A. S, Araújo, M. P, David, J. M. N. *GiveMe Trace*: Uma ferramenta para apoio a rastreabilidade de software. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Universidade Federal de Juiz de Fora, 2014.

[16] Van Den Brand, M.; Roubtsov, S.; Serebrenik, A. (2009). "SQuAVisiT: A Flexible Tool for Visual Software Analytics," Software Maintenance and Reengineering, 2009. CSMR '09. 13th European Conference on, pp. 331- 332.GIT: http://git-scm.com/. Acessado em 10 de fev de 2015.

[17] Sneed, H.M. “Software evolution: A roadmap”. Software Maintenance. Proc. IEEE International Conference, 2001.