MetricMiner: uma ferramenta web de apoio à mineração de repositórios de software

Francisco Zigmund Sokol[1], **Mauricio Finavaro Aniche**[1], **Marco Aurélio Gerosa**[1]

[1]Instituto de Matemática e Estatística – Universidade de São Paulo (USP)
Rua do Matão, 1010 – 05508-090 – São Paulo – SP – Brasil

csokol@linux.ime.usp.br, {aniche,gerosa}@ime.usp.br

Abstract. *This paper presents MetricMiner, a web application that provides support for researchers to mine software repositories. This tool stores data from repositories and calculates source code metrics over the history of the projects. Using MetricMiner, all the interaction is done by means of a web interface, differently from others tools current available, which demand that researchers install and configure the tool and download the projects in which they want to calculate metrics. Furthermore, MetricMiner provides a database with several open source projects in which researchers may extract and relate their data, in addition to submit new metrics to be calculated over these projects.*

1. Introdução

Minerar repositórios de software é uma tarefa custosa. Para desenvolver uma pesquisa na área, o pesquisador é obrigado a baixar diversos projetos em sua estação de trabalho e realizar uma série de cálculos sobre o código dos projetos e sobre os metadados de seu repositório. Esse processo a instalação de diversas ferramentas e bibliotecas localmente para reaproveitar as ferramentas desenvolvidas nessa área, tornando o processo trabalhoso e demorado.

Além de ser um processo complexo, esse tipo de pesquisa consome muitos recursos computacionais. Baixar os repositórios a serem minerados consome um volume considerável de banda. Depois, os dados devem ser processados e persistidos em um banco de dados, ocupando um grande volume de disco. Só então o pesquisador pode calcular métricas sobre esses dados, além de extrair relações entre os metadados do histórico do sistema de controle de versão, gastando uma quantidade grande de processamento de CPU. Só depois de passar por todas essas etapas, é possível extrair dados e avaliar hipóteses por meio de análises estatísticas.

Dessas dificuldades surgiu a motivação para o desenvolvimento do MetricMiner, uma aplicação web que realiza todas as etapas da mineração de um repositório de software. A ferramenta pode ser acessada em `http://metricminer.org.br`.

Na próxima seção é exibida a arquitetura da ferramenta desenvolvida para solucionar o problema descrito. Na Seção 3 são apresentadas brevemente as principais funcionalidades implementadas no MetricMiner. Na Seção 4, algumas ferramentas relacionadas ao software desenvolvido são exibidas e analisadas. A última seção traz a conclusão do artigo, mostrando como a ferramenta desenvolvida contribui na área de mineração de repositórios de software.

2. Arquitetura

Para o desenvolvimento da aplicação web, foi utilizado o VRaptor[1], um arcabouço para desenvolvimento web em Java baseado no padrão de projeto *MVC* (*Model-View-Controller*). Para a armazenar os dados minerados foi utilizado o banco de dados MySQL e o arcabouço de mapeamento objeto-relacional Hibernate[2].

O MetricMiner surgiu a partir do rEvolution[3], uma ferramenta de linha de comando que extrai dados de um repositório local e persiste em banco de dados relacional. Grande parte do código do rEvolution pôde ser reutilizado no MetricMiner, como o componente que realiza a interface com o sistema de controle de versão.

Algumas etapas da mineração de dados são demoradas, como clonar repositórios, persistir suas informações no banco de dados e calcular métricas sobre o código fonte. Com isso, optou-se por executar essas tarefas de maneira assíncrona, através da simulação de uma fila de execução. Cada tarefa é armazenada no banco de dados e processada por um componente que constrói as dependências de cada tarefa e as executa na ordem em que foram cadastradas.

Ao final desse processo, todas as informações do projeto estão persistidas no banco de dados e disponíveis para serem extraídas pelo pesquisador. É importante ressaltar que após a extração dos dados do sistema de controle de versão, é possível implementar novas métricas para serem calculadas sobre o código, sem precisar executar todos os passos anteriores novamente. A Figura 1 descreve as tarefas envolvidas no processo de mineração sobre um repositório de software cadastrado na ferramenta.

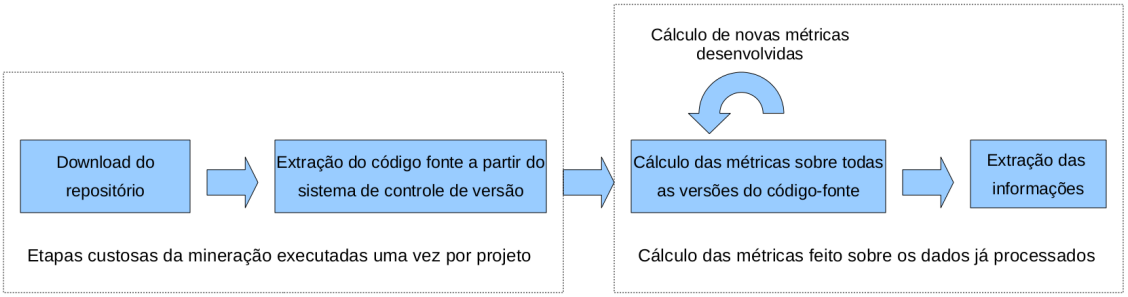

Figura 1. Diagrama do processo de mineração realizado pelo MetricMiner

A arquitetura é flexível na definição de novas tarefas, é necessário apenas que sejam desenvolvidas duas classes implementando duas interfaces. As classes que definem as tarefas do MetricMiner podem ser observadas no diagrama de classes simplificado da Figura 2. Cada tarefa implementada tem acesso ao projeto sobre o qual a tarefa será executada e uma conexão com o banco de dados. Dessa forma, cada tarefa pode extrair e relacionar qualquer informação do projeto e persistir os seus resultados no banco de dados.

O componente que realiza a interface com o sistema de controle de versão é flexível em relação ao sistema utilizado pelo projeto que será minerado. Para dar suporte a um novo sistema de controle de versão, é necessário apenas a implementação da

[1]http://vraptor.caelum.com.br/
[2]http://www.hibernate.org/
[3]http://github.com/mauricioaniche/rEvolution

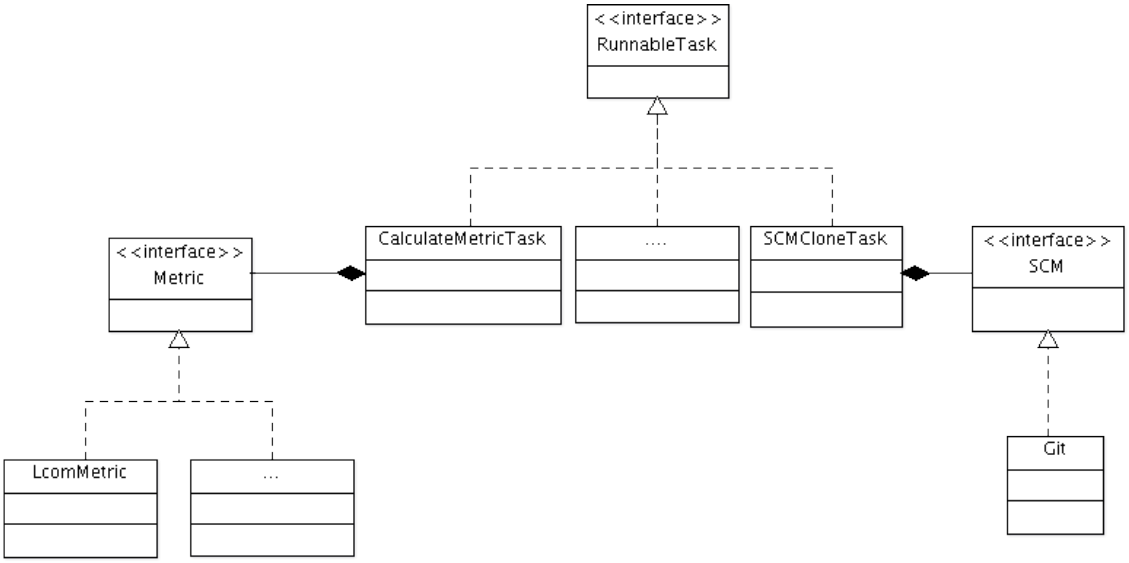

Figura 2. Diagrama com as principais tarefas implementadas no MetricMiner

interface SCM, que contém operações básicas sobre os metadados do sistema de controle de versão. Até o momento, já foi implementado suporte ao GIT.

A arquitetura também é flexível em relação a criação de novas métricas de código. Para criar uma nova métrica, deve ser definida uma classe para representar os resultados do cálculo da métrica que serão armazenados no banco de dados e uma classe que implemente a interface Metric. O cálculo das métricas sobre o código dos projetos cadastrados é feito por meio da tarefa definida na classe CalculateMetricTask. Ao ser executada na fila de execução, essa tarefa calcula uma determinada métrica para todas as versões de todos os códigos fonte de um projeto. As classes que calculam as métricas implementadas podem ser observadas no diagrama da Figura 2. A seguir encontra-se a lista das métricas implementadas acompanhadas de uma breve explicação:

- **Complexidade Ciclomática:** mede a complexidade do código, quanto maior o número de instruções como *if*, *while*, *case*, *&&*, ||, ou *?*, mais complexo é o código e, portanto, maior o valor da métrica [McCabe 1976].
- **Falta de Coesão dos Métodos (*LCOM*):** mede a coesão, considerando que uma classe é coesa se todos os métodos acessam os mesmos atributos da classe em comum [Henderson-Sellers 1996].
- ***Fan-out*:** mede o acoplamento de uma classe, contando o número de invocações de métodos de outras classes [Lorenz 1994].
- **Quantidade de Linhas de Código (*LOC*):** a contagem de linhas por método [Chidamber 1994].
- **Quantidade de Invocações de Métodos:** a contagem de invocações por método [Henry 1994].

3. Principais funcionalidades

Para iniciar o processo de mineração, o usuário deve cadastrar um projeto no sistema, fornecendo o nome e a *url* do repositório do código. Depois de salvar, são inseridas novas tarefas na fila de execução que irão baixar o código do respositório e persistir os dados

do sistema de controle de versão do projeto no banco de dados. Ao final dessas tarefas, o usuário pode entrar na tela de detalhes do projeto cadastrado e vizualizar alguns dados simples.

A Figura 3 exibe a tela de vizualização no MetricMiner do projeto Ant, software de código aberto da fundação Apache com mais seis mil *commits*. Nessa tela, são exibidas informações básicas sobre o projeto como número de *commits*, número de *commiters*, *url* do repositório, data do primeiro e último *commit*. São exibidos também dois gráficos simples nos quais pode ser vizualizado o número de *commits* nos últimos doze meses e o número de arquivos modificados em cada *commit* nos últimos seis meses.

Mais abaixo, na seção "*Available Metrics to Calculate*", são exibidas as métricas disponíveis no MetricMiner que ainda não foram calculadas para o projeto. O usuário pode selecionar as métricas que deseja calcular e as tarefas para a realização dos cálculos serão inseridas à fila de execução. Na seção "*Scheduled Tasks*", as tarefas que já foram executadas podem ser vizualizadas. Nesse caso, podemos ver que as métricas de complexidade ciclomática e a *fan-out* já foram calculadas, enquanto a *LCOM* e a quantidade de invocações de métodos estão na fila de execução para serem calculadas.

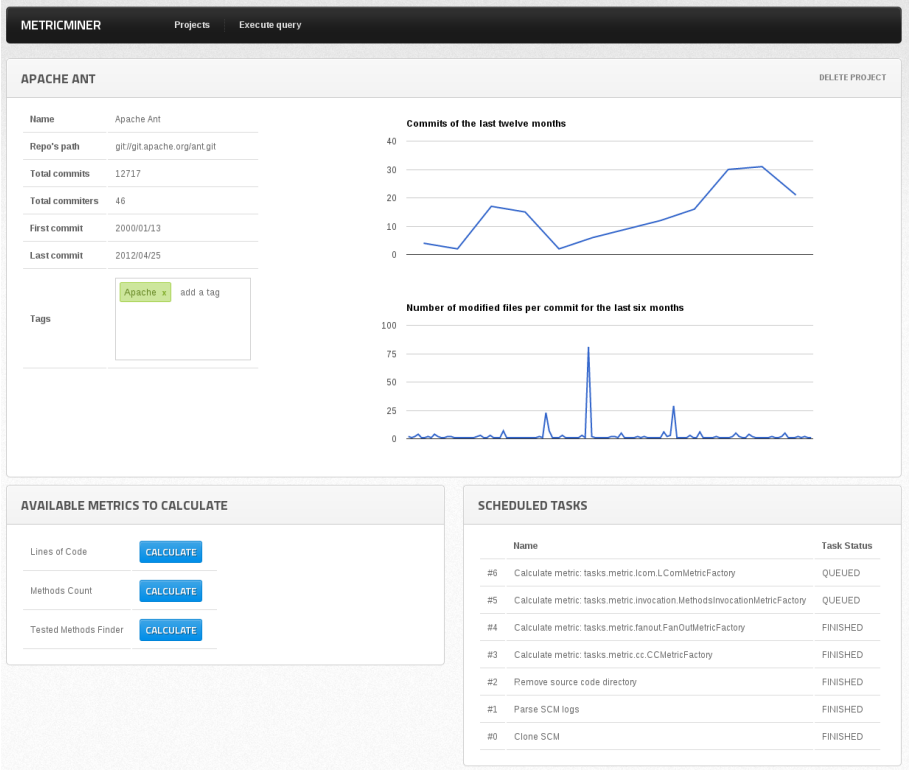

Figura 3. Tela de vizualização de projeto

Após o cálculo das métricas, o pesquisador pode realizar consultas aos dados calculados pela ferramenta. A Figura 4 exibe a tela na qual é possível inserir uma consulta em SQL ao banco de dados do MetricMiner. Depois de salvar a consulta, uma nova tarefa é adicionada à fila de execução e, ao final dessa tarefa, o pesquisador pode baixar o resultado de sua métrica em um arquivo CSV (*Comma Separated Values*).

Até o momento, o MetricMiner conta com uma base com dados de mais de 20

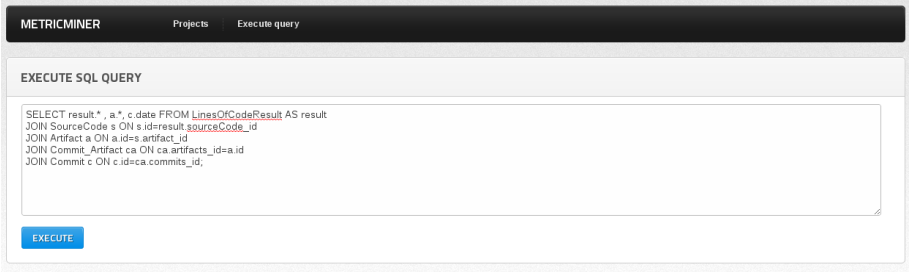

Figura 4. Tela de consultas aos dados calculados

projetos de código aberto sobre os quais é possível calcular métricas e extrair qualquer informação por meio de sua interface ou da implementação de novas tarefas. O código do projeto está disponível publicamente[4], de forma que qualquer colaborador pode submeter novas tarefas e métricas que serão acrescentadas à ferramenta e executadas sobre os projetos consolidados na base de dados.

4. Ferramentas relacionadas

Nesta seção serão apresentadas ferramentas que se relacionam ao MetricMiner.

4.1. Sonar

Não se conhece ferramentas web de suporte a mineração de repositórios de software. Uma ferramenta que se assemelha ao MetricMiner, é o Sonar [5], uma aplicação web que analisa o código fonte e extrai uma variedade de relatórios sobre o sistema, como resultados de métricas de código e dependências estruturais entre as classes. O foco desta ferramenta é apoiar a equipe de desenvolvimento e acompanhar a qualidade do código escrito. O Sonar não armazena metadados do sistema de controle versão e não permite que se extraia dados dos projetos armazenados, mas fornece uma interface de vizualização com muitos recursos. Por esses motivo, o Sonar é muito utilizado na indústria e pouco utilizado para fins acadêmicos.

Diferentemente do Sonar, o MetricMiner tem o foco na área acadêmica, possibilitando que os usuários extraiam dados calculados pelo sistema para realizar a análise que desejar, sem focar tanto na vizualização dos dados armazenados.

4.2. Eclipse Metrics

O Eclipse Metrics[6] é um *plugin* para o Eclipse que calcula uma variedade de métricas de código no ambiente do desenvolvedor. Dessa forma, para que se acompanhe a evolução do código do projeto é necessário executar manualmente o *plugin* para os *releases* que se deseja analisar. No MetricMiner, o cálculo das métricas é realizado sobre todo o histórico de versões do projeto analisado, sem que o usuário precise selecionar as versões manualmente. Além disso, não é necessário que o usuário configure nada em seu ambiente e nem que mantenha as versões que deseja analisar localmente.

[4]http://github.com/csokol/MetricMiner
[5]http://www.sonarsource.org/
[6]http://metrics.sourceforge.net/

4.3. Kalibro

Desenvolvida no Brasil, o Kalibro[7] é uma ferramenta que calcula as principais métricas de código fonte. O foco do Kalibro é dar suporte ao desenvolvedor, sugerindo valores de referência para as métricas calculadas, apontando possíveis problemas no projeto analisado. A ferramenta permite que os valores de referência sejam configurados por projeto. Através de código JavaScript, é possível compor as métricas calculadas pelo Kalibro, permitindo que o usuário crie novas métricas. Assim como o Eclipse Metrics, o Kalibro não analisa todo o histório de versões, de forma que o usuário precisaria selecionar as versões e recalcular as métricas manualmente.

5. Conclusão

O MetricMiner é uma ferramenta que agrega valor à área de mineração de respositórios de software, armazenando um grande volume de dados de repositórios de projetos de código aberto e possibilitando que os usuários extraiam essas informações. Por ser uma aplicação web, suas funcionalidades são de fácil acesso ao pesquisador, ao contrário das ferramentas existentes.

Através do MetricMiner, pesquisadores podem extrair dados pré-calculados, poupando tempo e recursos computacionais necessários nos trabalhos realizados com as ferramentas disponíveis atualmente. Além disso, por ser um projeto de código aberto com uma arquitetura flexível, a ferramenta permite que colaboradores desenvolvam novos componentes que podem ser acrescentados ao sistema. Através dessa colaboração, pesquisadores poderão, por exemplo, submeter novas métricas que serão calculadas sobre todos os projetos já consolidados na base de dados do MetricMiner.

Futuramente, pretende-se implementar a interface com outros sistemas de controle de versão, paralelizar a implementação de alguns algoritmos para tornar o processo de mineração mais eficiente e desenvolver novas métricas de código. Além disso, pretende-se desenvolver uma *API* para que componentes externos ao MetricMiner implementem interfaces de vizualização dos dados minerados.

Referências

Chidamber, S.; Kemerer, C. (1994). A metrics suite for object oriented design. pages 476–493. IEEE TSE, Vol. 20 (6).

Henderson-Sellers, B. (1996). *Object-oriented metrics: measures of complexity*. Prentice-Hall.

Henry, W. L. S. (1994). Object-oriented metrics that predict maintainability. J. Systems and Software, vol. 23, no. 2.

Lorenz, M.; Kidd, J. (1994). *Object-oriented metrics: A Practical Guide*. Prentice-Hall.

McCabe, T. (1976). A complexity measure. pages 308–320. IEEE TSE, SE-2, Vol. 4.

[7] http://www.kalibro.org/