Crab: Uma Ferramenta de Configuração e Interpretação de Métricas de Software para Avaliação de Qualidade de Código*

Paulo R. M. Meirelles, Raphael Cóbe, Simone Hanazumi, Paulo Nunes, Geiser Challco, Straus Martins, Eduardo Morais, Fabio Kon

[1] Departamento de Ciência da Computação -IME/USP

{paulormm,rmcobe,hanazumi,prnunes,geiser,straus,emorais,kon}@ime.usp.br

Abstract. *This paper presents Crab, a tool designed to be incorporated into Java source code metric tools, extending them to provide easy to understand evaluation of the analyzed software quality. This is achieved by letting an expert user specify a set of acceptance ranges to each metric provided by the base tool and enabling the definition of compound metrics.*

Resumo. *Este artigo apresenta a Crab, uma ferramenta que estende programas de cálculo de métricas de código-fonte Java. A Crab permite a configuração, por um especialista, de um conjunto de intervalos para os valores das métricas disponibilizadas e a definição de métricas compostas, possibilitando uma apresentação de fácil entendimento dos resultados da avaliação do código.*

1. Introdução

A crescente adoção de programas de código aberto [Benkler 2006] e de métodos ágeis pela indústria de software promove o código-fonte a um dos artefatos mais importante para se medir a qualidade de software. Com isso, as métricas de qualidade de código-fonte são mecanismos importantes para avaliação desses sistemas.

A utilização de métricas de qualidade de código-fonte como critério para a avaliação da qualidade de um software é motivada por estudos [Henderson-Sellers 1996, Sato et al. 2007] que indicam ser viável analisar algumas das principais características para a aceitação de um software, tais como: flexibilidade, complexidade e manutenibilidade a partir do código-fonte. Entretanto, observou-se que as ferramentas de avaliação de código-fonte ainda são deficientes da perspectiva de um usuário não especialista em métricas de software, pois geralmente apresentam resultados na forma de valores isolados para cada métrica. Isso exige que os usuários possuam a capacidade de interpretar esses valores, dificultando a tomada de decisões a respeito da qualidade do software.

Uma deficiência, muitas vezes ignorada durante o projeto de ferramentas de métricas, é não levar em conta o fato de que a interpretação livre de contexto de uma métrica pode não retratar características consideradas importantes na decisão de adotar um determinado software [Mills 1988]. Este artigo apresenta uma ferramenta denominada Crab, que tem como objetivo lidar com esses problemas existentes em ferramentas de métricas, tornando possível a definição de um contexto para a avaliação dos valores coletados.

*Os autores agradecem ao CNPQ, CAPES, FAPESP, Qualipso e CCSL-USP pelo apoio a este trabalho. Também agradecem ao professor Alfredo Goldman por permitir parte do seu desenvolvimento dentro da disciplina Laboratório de Programação eXtrema do IME-USP.

A Crab recebe como entrada, a definição de um conjunto de métricas, que deve ser obtida a partir de uma ferramenta específica de métricas de software, chamada aqui de *ferramenta base*. Como saída, produz um resultado da avaliação da qualidade do software em um formato de mais fácil entendimento. A ferramenta possibillita que, um usuário com conhecimento em métricas, possa cadastrar intervalos de valores de julgamento para cada métrica, juntamente com uma avaliação qualitativa, ou ainda carregar configurações definidas por outros especialistas.

2. Ferramentas de Métricas de Software

Por meio de métricas específicas, pode-se avaliar o quão adequado a uma propriedade desejada está um determinado sistema [Lanza and Marinescu 2006]. Baseado nesses dados, um desenvolvedor pode controlar a qualidade de seu projeto no que diz respeito a boas práticas de escrita do código-fonte.

Existem ferramentas disponíveis que permitem calcular métricas a partir do código-fonte, possibilitando aferir que determinadas características estejam presentes ou não em um software. Entre essas ferramentas, pode-se citar o *Metrics* (metrics.sourceforge.net): um plugin para a *IDE Eclipse*, que é capaz de calcular 22 métricas em códigos Java, classificando o resultado obtido de acordo com cores que indicam valores aceitáveis ou não para as métricas. Outra ferramenta amplamente utilizada é o *Checkstyle* (checkstyle.sourceforge.net), também um plugin para a *IDE Eclipse*, que avalia o estilo do código desenvolvido, auxiliando a padronização do código-fonte e indicando como o software pode ficar mais claro e simples. Uma outra alternativa é a JaBUTi (*Java Bytecode Understanding and Testing*), que através de análise do *bytecode*, extrai métricas de qualidade e de complexidade do código [Vincenzi et al. 2003]. Essas ferramentas seguem a tendência de expor ao usuário dezenas de valores numéricos que são de pouco valor para não especialistas em métricas de código-fonte.

A Crab se diferencia das demais ferramentas de métrica de software, pois possibilita a combinação de um conjunto de métricas e a apresentação dos resultados em um formato de mais fácil entendimento por usuários menos familiarizados com métricas de software. Ela permite também que especialistas em métricas configurem intervalos qualitativos de valores para cada métrica. Assim, os valores obtidos nas medições de um determinado código-fonte são classificados automaticamente, de acordo com os intervalos predefinidos pelo especialista, auxiliando não especialistas na avaliação do código-fonte.

3. A Ferramenta Crab

Com objetivo de potencializar o uso das métricas de código-fonte, a Crab permite que um especialista defina intervalos de valores para cada métrica, associando valores qualitativos a esses intervalos. Cada métrica é calculada para cada classe e, em seguida, sumarizada para a avaliação do software como um todo. A sumarização pode se dar de 4 formas: soma, onde os valores coletados em cada classe são somados; máximo, onde apenas o valor máximo é exibido; mínimo, onde somente o valor mínimo é exibido; e média, onde o valor exibido é fruto da média aritmética dos valores em cada classe.

A crab também possibilita a definição de métricas compostas, definidas através de expressões que podem utilizar os valores coletados nas outras métricas. Essa funcionalidade torna possível expressar melhor uma determinada característica do software.

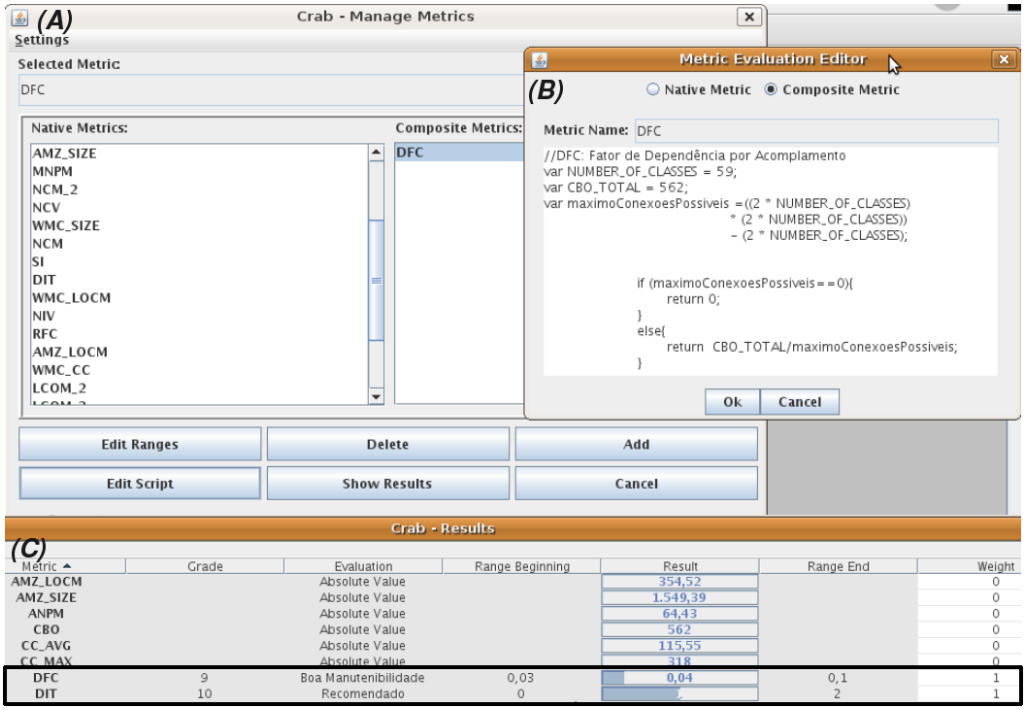

Figura 1. Telas da Crab

A Crab foi projetada para ser utilizada em conjunto com qualquer ferramenta de coleta de métricas escrita em Java. Ao ser integrada a uma ferramenta base, a Crab passa a reutilizar suas métricas. As funcionalidades da Crab são as seguintes:

- Carregar métricas implementadas na ferramenta base e listá-las na tela da Crab, permitindo que o usuário edite (configure) ou exclua métricas (ver Figura 1.A);

- Composição de métricas através de um código Javascript, possibilitando a definição de uma nova métrica. Esse script pode reutilizar valores de métricas providas pela ferramenta base e outras métricas previamente compostas (ver Figura 1.B);

- Configuração das métricas, isto é, definir a descrição, a categoria de contabilização (soma, média, máximo, mínimo) e os intervalos qualitativos das métricas;

- Definição qualitativa do intervalo, indicando nome (ex.: péssimo, ruim, regular, bom, ótimo), os valores inicial (fechado) e final (aberto), uma nota, comentário e recomendação;

- Contabilização das análises de todas as classes carregadas para a medição e geração de um resultado sobre a qualidade do código. A Crab chama os métodos que executam os algoritmos de cálculo das métricas da ferramenta base, executa o Javascript das métricas compostas e, de posse desses dados, verifica em que intervalo configurado o resultado da métrica é classificado, apresentando ao usuário uma nota para intervalo e o valor absoluto em comparação ao valor mínimo e máximo do intervalo;

- Construção de um resultado unificado, totalizando os valores das métricas das classes (ver Figura 1.C). O cálculo efetuado na totalização dos valores de cada métrica é feito de acordo com a categoria em que a métrica foi classificada: soma, média, máximo ou mínimo. Por exemplo, linhas de código é soma, métodos por classes é média e profun-

didade na hierarquia de classe é máximo. Além disso, pesos para cada métrica podem ser estipulados de forma que seja possível, ao final, calcular uma nota para o código avaliado, obtida pela média ponderada dos valores medidos.

- Permite trabalhar com diferentes configurações para os valores das métricas. Esses valores podem ser salvos em um arquivo de propriedades Java. Isso possibilita definir valores de referência para as métricas em diversos contextos, estabelecidos por diferentes especialistas em métricas, de tal forma que o usuário final poderá carregar a configuração de sua escolha.

3.1. Arquitetura da Crab

A Figura 2 apresenta a arquitetura da Crab. Para simplificar a incorporação da Crab a diferentes ferramentas base, a arquitetura é composta de poucos componentes com responsabilidades bem definidas.

O componente central da arquitetura é a classe MetricEvaluationServiceFacade, implementada seguindo o padrão de projeto Façade. Esse fachada é utilizada como ponto de acesso único às funcionalidades da Crab e suas operações são largamente utilizadas pela interface com o usuário. Ela é responsável pelas chamadas de métodos dos demais componentes, denominados de Service. Da mesma forma, um serviço apenas acessa outro serviço através da fachada. A arquitetura da Crab é composta por 4 serviços:

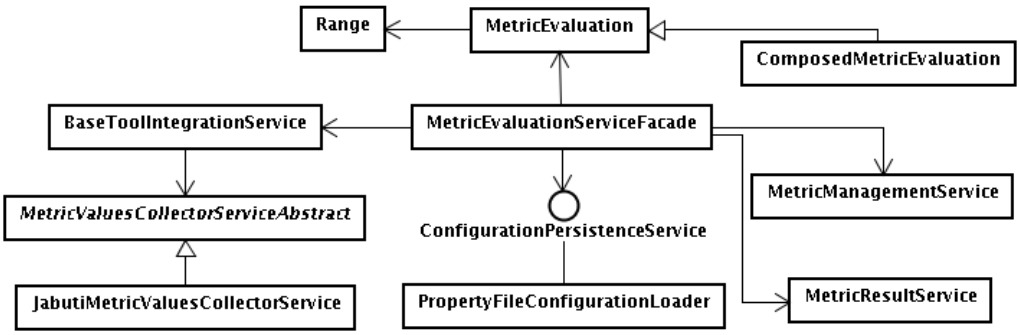

Figura 2. Arquitetura da Crab

- Gerenciamento de Métricas: tem como principal atribuição o cadastro dos valores utilizados nas avaliações e como são utilizados para classificar o código que está sendo avaliado, de acordo com os intervalos qualitativos definidos. A classe responsável por essas funções é a MetricManagementService. Essa classe possui métodos responsáveis por cadastrar os dados das métricas, criar efetivamente o código de uma métrica composta definida e gerenciar os intervalos qualitativos para cada métrica, utilizando as classes MetricEvaluation, ComposedMetricEvaluation e Range.

- Carregar métricas disponíveis e seus respectivos valores: esta responsabilidade se refere ao mecanismo de integração entre a Crab e a ferramenta base. Seu principal componente é o BaseToolIntegrationService, uma classe de serviço que acessa a classe abstrata MetricValuesCollectorServiceAbstract, utilizada para realizar requisições à ferramenta base, recolhendo os valores das medições, bem como as métricas disponíveis. Ferramentas base devem implementar essa classe abstrata e seus respectivos métodos: collectMetrics(), responsável por fornecer os valores das medições e getAvailableMetrics() que fornece uma lista das métricas disponíveis na ferramenta base.

- Carregar configuração: a Crab é capaz de persistir os intervalos e avaliações nela cadastrados. O componente responsável por essa funcionalidade, o ConfigurationPersistenceService, provê a capacidade de armazenar e recuperar essas informações. A Crab permite a exportação e importação de arquivos de propriedades Java. Para isso, a classe PropertyFileConfigurationLoader implementa a interface ConfigurationPersistenceService.

- Cálculo dos resultados: a classe MetricResultService é responsável por contabilizar a avaliação do código, coletar os valores para as métricas obtidas na ferramenta base e validar e executar o código Javascript de métricas compostas. Os métodos dessa classe sumarizam os valores de todas as métricas por classe e também totalizam a avaliação geral do código do software, calculando o resultado final de cada métrica de acordo com a categoria definida.

4. Estudo de Caso: Integração com a JaBUTi

Para iniciar a validação da Crab, ela foi integrada à ferramenta de testes e métricas JaBUTi [Vincenzi et al. 2003], que foi escolhida por ser de código aberto e por seus desenvolvedores estarem disponíveis para tirar qualquer eventual dúvida. A JaBUTi contém um módulo que calcula um conjunto de 27 métricas de código para programas escritos em Java. A integração com a Crab tornou possível estender as análises realizadas pela JaBUTi.

A JaBUTi, por ser uma aplicação *desktop*, apresentou um cenário de desenvolvimento mais simples de ser explorado como primeiro estudo de caso da Crab. A integração foi realizada pela implementação da classe abstrata referente ao serviço de carregar as métricas disponíveis na Crab, i.e., a definição da classe JabutiMetricValuesCollectorService que estende MetricValuesCollectorServiceAbstract (ver Figura 2), sobrescrevendo os métodos getAvailableMetrics() e collectMetrics(), responsáveir por obter as métricas nela disponíveis e invocar os métodos responsáveis por fornecer os valores coletados pelas métricas para a Crab.

Com a finalidade de avaliar as funcionalidades da Crab dentro da JaBUTi, analisou-se o código-fonte de um sistema com 5751 linhas e 59 classes. Nesse simples exemplo de uso, além de definir intervalos qualitativos para as métricas nativas da JaBUTi, foram compostas novas métricas. Para ilustrar, a Figura 1.B apresenta a composição de uma nova métrica, denominada DFC (Dependece Factor by Coupling), obtida pela razão do valor da métrica CBO (Coupling Between Object) pelo número máximo de conexões possíveis entre as classes do software, baseado no conceito de conectividade apresentado em [Ferreira et al. 2008]. O resultado da avaliação do software utilizando essa métrica, bem como os intervalos qualitativos definidos para ela, podem ser vistos na Figura 1.C.

5. Considerações Finais

Este artigo apresentou a ferramenta Crab para avaliação automática da qualidade de software através de métricas de qualidade de código-fonte. Essa ferramenta foi concebida com o objetivo de ser integrada a outras, reutilizando suas estratégias de cálculo das métricas. Com ela, é possível configurar métricas e definir novas métricas compostas por outras. Com isso, a Crab torna mais amigável a análise dos valores das medições, possibilitando a criação de faixas qualitativas para métricas, permitindo ainda a escrita, de forma dissertativa, de comentários e recomendações para os valores obtidos nas métricas.

O principal diferencial da Crab é o mecanismo de construção de novas métricas, compostas por outras métricas, representando características num nível abstrato mais próximo aos conceitos da engenharia de software, i.e., características como flexibilidade e manutenibilidade. Com essa funcionalidade, a Crab provê a ampliação do uso de métricas de código-fonte, potencializando a análise de qualidade do código.

Desenvolveu-se um estudo de caso, demonstrando que o projeto da Crab facilita a integração com outra ferramenta (a JaBUTi). O esforço para a concretização dessa integração se restringiu a entender como e quais são as chamadas para execução dos algoritmos das métricas na JaBUTi. Também, com um simples exemplo de uso, foi confirmado o que se espera inicialmente da Crab: tornar possível classificar o código-fonte de um software a partir de intervalos qualitativos definidos por especialistas. Experimentos mais completos serão realizados no contexto da pesquisa que envolve a evoluação da Crab.

A Crab continuará sendo desenvolvida para a geração de relatórios mais informativos e de leitura ainda mais simples. Sua interface gráfica será melhorada e repensada observando alguns conceitos de usabilidade. Outra inovação pretendida, e já prototipada, é possibilitar a importação de novas métricas, carregando pela Crab um arquivo .class com o código de uma métrica, que a ferramenta base não possua. Além disso, ela será acoplada a outras ferramentas para coleta de métricas, como o Metrics, a fim de validar sua generalidade. Pretende-se formar uma comunidade de desenvolvimento da Crab e de compartilhamento de configuração de valores de referências para as métricas, por isso seu código está disponível sob a licença BSD, no repositório: svn://ccsl.ime.usp.br/jabutimetrics. Na página do projeto: ccsl.ime.usp.br/mangue/crab, há uma versão .jar da Crab integrada na JaBUTi e um tutorial de uso da mesma.

Referências

Benkler, Y. (2006). *The Wealth of Networks: How Social Production Transforms Markets and Freedom*. Yale University Press.

Ferreira, K. A. M., Bigonha, M. A. S., and Bigonha, R. S. (2008). Re-estruturação de software dirigida por conectividade para redução de custo de manutenção. In *Revista de Informática Teórica e Aplicada*, volume 15, pages 155–180.

Henderson-Sellers, B. (1996). *Object-Oriented Complexity*. Prentice-Hall.

Lanza, M. and Marinescu, R. (2006). *Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems*. Springer.

Mills, E. E. (1988). Software metrics. Technical report, Software Engineering Institute (SEI), Carnegie Mellon University.

Sato, D., Goldman, A., and Kon, F. (2007). Tracking the evolution of object oriented quality metrics. In *Proceedings of the 8th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2007)*, pages 84–92.

Vincenzi, A., Wong, W., Delamaro, M., and Maldonado, J. (2003). Jabuti: A coverage analysis tool for java programs. In *XVII SBES - Brazilian Symposium on Software Engineering*, pages 79–84.