CodeKeySearch: Aliando Reutilização de Software com Métodos Ágeis.

Ester C. de Lima, Frederico T. de Oliveira, Leonardo Murta, Cláudia Werner

COPPE/UFRJ, Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro (UFRJ)
{esterlima, ftoliveira, murta , werner }@cos.ufrj.br

Abstract. *This paper describes an approach that aims at supporting software reuse in the context of agile methods, more specifically Extreme Programming (XP). This approach is automated by CodeKeySearch, which is a class and method-search mechanism based on keywords. A requirements engineering tool serves as a knowledge repository, favoring the reuse in other projects, and supporting the link between documentation and source code. Therefore, CodeKeySearch together with its tool for requirements engineering in tandem support search, evaluation, and adaptation of the artifact found for reuse.*

Resumo. *Este trabalho descreve uma abordagem que visa apoiar a reutilização de* software *em sistemas desenvolvidos com métodos ágeis, especificamente o Extreme Programming (XP). Essa abordagem é automatizada pelo CodeKeySearch, que é um mecanismo de busca de classes e métodos através de palavras-chave. Uma ferramenta de levantamento de requisitos serve de repositório de conhecimento, favorecendo também a reutilização em outros projetos, e apoiando a ligação entre a documentação e o código fonte gerado. Assim, o CodeKeySearch e a sua ferramenta de cadastro de requisitos em conjunto apóiam a busca, compreensão e adaptação do artefato encontrado para a reutilização.*

1. Introdução

Desenvolver sistemas de forma rápida, mas mantendo a qualidade, é hoje uma necessidade comum nas empresas que anseiam por sistemas capazes de automatizar suas atividades diárias. Além disso, mudanças de requisitos, mudanças na equipe de desenvolvimento e outros, também são constantes e de grande impacto no processo de desenvolvimento de software. Os métodos ágeis representam uma estratégia para atender à dinâmica desses projetos, no qual a equipe está preparada para atender às mudanças e estas acabam sendo bem aceitas, uma vez que já são esperadas. Os aspectos principais nesses métodos são a simplicidade e a velocidade [Highsmith and Cockburn, 2001]. Numa outra vertente, temos a reutilização de software, que também vem contribuir para agili-

zar o processo de desenvolvimento, favorecendo o aumento da produtividade e da qualidade do software desenvolvido.

No entanto, a prática da reutilização ainda não se tornou comum nos métodos ágeis. Alguns projetos foram desenvolvidos para facilitar a reutilização em métodos tradicionais [Werner et al., 2003], mas que são de difícil aplicação nos métodos ágeis em função das divergências nas práticas adotadas. As abordagens de reutilização, por exemplo, requerem uma boa documentação de seus artefatos para que seja possível buscá-los, compreendê-los e adaptá-los para a reutilização. Por outro lado, os métodos ágeis não são centrados na documentação. Eles optam por uma documentação apropriada para auxiliar a implementação do sistema, porém sem excessos, o que pode constituir um fator de impacto negativo para a busca e compreensão de artefatos, caso não seja bem explorado.

Dessa forma, este trabalho tem como objetivo apoiar a "reutilização ágil" [McCarey et al., 2005]. Ou seja, aplicar a prática da reutilização de software no desenvolvimento com uso de métodos ágeis. Para isso, foi feito um levantamento da sistemática que deve ser adotada para a reutilização e um estudo dos diversos métodos ágeis. O Extreme Programming (XP) foi o método ágil que se apresentou mais favorável para ser utilizado no contexto de reutilização de software na medida em que foi facilmente adaptada para incorporar atividades relacionada à reutilização de software.

Para atingir esse objetivo, foi construída a ferramenta CodeKeySearch que, em sua concepção, é capaz de auxiliar a adoção da prática de reutilização de software no XP, permitindo buscar classes e métodos para reutilização e contribuindo também na agilidade do desenvolvimento com XP. A ferramenta requer que os projetos sejam desenvolvidos especificamente para este fim, atendendo a apenas dois esforços complementares às práticas do desenvolvimento em XP: um na fase de elaboração das metáforas e um outro na fase de desenvolvimento.

As seções seguintes estão divididas da seguinte forma: a Seção 2 apresenta a fundamentação teórica necessária para introduzir a reutilização no desenvolvimento de software com métodos ágeis. A Seção 3 expõe o fluxo das atividades nesse novo processo e a Seção 4 apresenta a descrição da ferramenta CodeKeySearch, proposta para auxiliar a reutilização em XP. A Seção 5 apresenta alguns trabalhos relacionados. Finalmente, a Seção 6 apresenta algumas discussões acerca do CodeKeySearch e perspectivas futuras.

2. Fundamentação Teórica

Nessa seção apresentamos os conceitos relevantes para introduzir a reutilização no processo de desenvolvimento baseado em métodos ágeis.

2.1 Reutilização de Software

Reutilização de Software é o processo de criação de sistemas de software a partir de software preexistente [Krueger at al., 1992]. Em uma organização, o desenvolvimento pode ser **para** reutilização ou **com** reutilização.

O desenvolvimento de software **para** reutilização tem como fim produzir os artefatos para que sejam reutilizados futuramente. No desenvolvimento de software **com** reutilização, artefatos são incorporados ao sistema. Entende-se por artefatos reutilizáveis qualquer produto resultante das atividades de desenvolvimento de software, como, por exemplo, códigos, componentes, especificações de requisitos, conhecimentos adquiridos com o desenvolvimento de um software, etc. Dessa forma, os artefatos reutilizáveis devem ser desenvolvidos especificamente para a reutilização, representando uma forma compreensível e útil para a reutilização por aqueles que não o criaram [Decker et al., 2005]. Em ambos os casos, a reutilização de software vem somar benefícios ao desenvolvimento de sistemas, tais como [Frakes et al., 2005]: aumento na produtividade e redução do tempo de entrega; aumento da qualidade, nos quais sistemas e produtos de software são mais confiáveis e consistentes; padronização do código que facilita a manutenção e evolução do mesmo; e interoperabilidade e compatibilidade.

Num processo onde há reutilização, é preciso identificar o que será reutilizado, empregar tempo na busca, na compreensão e, eventualmente, na modificação para adaptar o artefato encontrado, de forma a atender às necessidades do que está sendo desenvolvido. Ainda hoje, a reutilização sistemática enfrenta uma série de desafios, causados por um suporte insuficiente nos passos básicos da reutilização [Decker et al., 2005]: **buscar**, **compreender** e **adaptar**. Nem sempre as pessoas conseguem encontrar aquilo que precisam ou mesmo nem iniciam o processo de busca por saberem das dificuldades que existem nesse processo e do tempo que deverão empregar para esse fim, considerando mais ágil implementar do zero o que se necessita. As dificuldades da compreensão se dão pelo fato da pouca ou ausente documentação capaz de auxiliar nessa etapa.

2.2 Métodos Ágeis

Em projetos nos quais são constantes as mudanças em escopo e requisitos, a equipe é pequena e o tempo de entrega do software é curto, é preciso que o desenvolvimento seja simples e ágil. Para esses tipos de projetos, os métodos tradicionais podem se tornar inviáveis, e para evitar que não se adote nenhum método, os métodos ágeis tornam-se um recurso valioso [Soares, 2004].

Highsmith e Cockburn [2001] definem desenvolvimento ágil de software como uma abordagem de desenvolvimento que trata os problemas das mudanças rápidas: mudanças nas forças de mercado, requisitos de sistemas, tecnologia de implementação e equipes de projeto dentro do período de desenvolvimento. Uma vez que eliminar tais mudanças é inviável, a estratégia é estar preparado para atendê-las, sem a necessidade de retrabalho, custos adicionais ou inserção de defeitos no desenvolvimento. Assim, mudanças de prioridades ou requisitos são bem administradas e absorvidas pela equipe e pelo cliente, desde que essas mudanças não ultrapassem o escopo limite, ou prazo de entrega e o custo acordado do projeto.

Diversos métodos ágeis emergiram nos últimos anos e, em 2001, foi publicado o Manifesto Ágil, que é baseado nos seguintes valores chave: indivíduos e interações acima de processos e ferramentas; software executável acima de documentação; colaboração do cliente acima de negociação de contratos; e respostas rápidas a mudanças acima

dos planos a serem seguidos. Esses valores mostram que processos, ferramentas, documentação, contratos e planos de projetos são necessários e úteis, porém não são o foco principal nos métodos de desenvolvimento ágil [Manifesto Ágil, 2001].

O XP, criado por Kent Beck e Ward Cunningham em 1999, é o mais popular dos métodos ágeis. Ele é indicado para equipes pequenas e médias, com até dez integrantes. O XP define um conjunto de doze práticas, escolhidas com base em quatro valores: comunicação, simplicidade, *feedback* e coragem [Teles, 2004]. As práticas do XP apóiam umas às outras e devem ser usadas em conjunto para se ter agilidade no processo. Aplicar as práticas de forma isolada pode não produzir a agilidade desejada.

3. A Abordagem Proposta

A abordagem proposta visa atender aos desenvolvedores que utilizam XP como método ágil e Java como linguagem de programação. A filosofia por trás da abordagem consiste em iniciar o processo de desenvolvimento para reutilização, para que, futuramente, os projetos desenvolvidos forneçam artefatos para o desenvolvimento com reutilização de outros projetos. Para permitir a aplicação dos três passos básicos da reutilização (i.e., busca, compreensão e adaptação) foi necessário complementar algumas das práticas do desenvolvimento com XP.

Assim como nos outros métodos, o XP também passa pelo levantamento de requisitos, que é feito através de metáforas, onde são anotadas em cartões as descrições simples da funcionalidade em questão. O levantamento dessas metáforas é feito junto ao cliente, uma vez que a interação com o usuário representa a base do método. Desses cartões são derivadas as funcionalidades que compõem o sistema como um todo.

Em nossa abordagem, as descrições das funcionalidades devem ser anotadas em cartões virtuais semelhantes ao do XP**.** Para cada metáfora, deve ser anotada pelo menos uma palavra-chave que represente a metáfora em questão. Essa palavra-chave deve fazer parte de um vocabulário representativo do domínio em questão, que será criado ao longo do processo de desenvolvimento. As palavras-chave associadas a cada metáfora, também, devem aparecer no código que está sendo desenvolvido para atender à metáfora. Cada classe ou método pode conter uma ou mais palavras-chave associadas.

Ao final de cada projeto, este deve ser incluído no repositório para reutilização, de onde será derivado um relacionamento entre as palavras-chave.

Quando os métodos de uma classe possuem uma ou mais palavras-chave diferentes entre si, é possível criar um relacionamento entre elas, aumentando o escopo das buscas. Por exemplo: Uma classe possui a palavra-chave antena, porém um de seus métodos possui a palavra-chave rádio e receptor. Com isso, é feito um relacionamento entre antena:rádio, antena:receptor e rádio:receptor. Assim, sempre que a palavra-chave pesquisada for rádio, será indicado antena e receptor como palavras-chave associadas.

A busca na abordagem proposta é feita através das palavras-chave retiradas das metáforas previamente cadastradas, dando uma maior semântica à mesma.

Em resumo, o desenvolvimento **para** reutilização será composto pelas seguintes etapas: Primeiramente, os requisitos são todos levantados e registrados, com a colabora-

ção de todos da equipe, na forma de metáforas. Em cada metáfora é adicionada uma ou mais palavras-chave que representam o requisito. A etapa seguinte ao processo de levantamento de requisito é o desenvolvimento. Este deve ser feito acrescentando-se a palavra-chave, que se encontra na metáfora que está sendo codificada, nas classes e métodos. Ao final do desenvolvimento, o desenvolvedor executa a ferramenta sobre o projeto e este passa a compor o repositório de reutilização, contendo o relacionamento entre as palavras-chave e o relacionamento entre métodos/classes e as palavras, para apoiar os desenvolvimentos futuros.

O desenvolvimento **com** reutilização ocorre da seguinte forma: O cadastro de estórias serve de apoio para a busca de metáforas. A pesquisa é realizada sobre o texto da metáfora, que contém informações suficientes para entender o requisito e os métodos e classes que o implementam. Encontrada a metáfora desejada para reutilização, seleciona-se a palavra-chave associada a esta para busca de código. A busca pela palavra-chave retorna os artefatos (classes e métodos) que estão relacionados à palavra-chave em questão. Serão também sugeridas outras palavras que estejam associadas à palavra pesquisada, assim como seus artefatos relacionados. A sugestão de palavras-chave visa aumentar o escopo da busca, retornando mais artefatos relevantes para o desenvolvedor. Através do acesso ao código fonte, é possível reutilizar ou adaptar o código já desenvolvido para o projeto atual. Ao final do projeto, este também passa a compor o repositório.

Bosch [2000] define diferentes formas de adaptação de um artefato reutilizável recuperado. Em nossa abordagem, utilizamos a técnica de copiar e colar. Esta técnica permite uma reutilização rápida, porém nos traz algumas limitações no que tange a evolução do código copiado.

4. A Ferramenta CodeKeySearch

Para automatizar a abordagem proposta, a ferramenta CodeKeySearch foi inicialmente desenvolvida como uma aplicação Java. Contudo, o uso de uma ferramenta externa ao ambiente de desenvolvimento do programador poderia impactar negativamente na agilidade do processo como um todo. Por isso, foi decidido transformar a ferramenta em um plug-in para a IDE Eclipse [Eclipse Foundation, 2008], ambiente para desenvolvimento Java amplamente utilizado.

A Figura 1 apresenta o suporte fornecido pelo CodeKeySearch para o cadastro das metáforas em cartões virtuais.

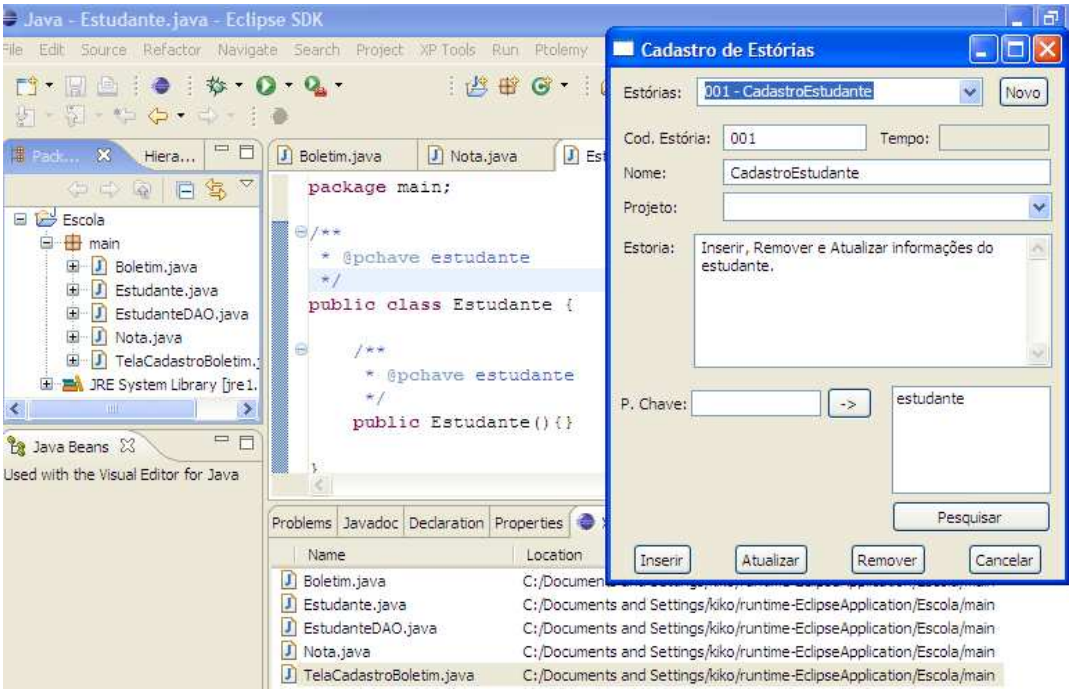

Figura 1 – Cadastro de estórias e utilização de palavras-chave em código

No exemplo da Figura 1, a estória cadastrada é CadastroEstudante, que contém sua descrição e a palavra-chave "estudante" associada. Para associar as classes e métodos às palavras-chave, é utilizado o recurso da documentação em JavaDoc [JavaDoc, 2000], com a utilização das etiquetas @pchave, conforme exibido na Figura 1.

Este cadastro de estórias e palavras-chave serve como base para que uma busca simples nas estórias encontre as palavras-chave relacionadas. Essas palavras-chave encontradas são usadas pelo CodeKeySearch para detectar os métodos e classes relevantes para a busca inicial.

Quando, numa classe, seus métodos e atributos possuírem uma ou mais palavras-chave diferentes entre si, é possível criar um relacionamento entre essas palavras de forma a aumentar o escopo das buscas.

Para melhor ilustrar, considere a Tabela 1 como um relacionamento obtido através do uso desta ferramenta em um projeto. Caso a palavra-chave pesquisada seja "estudante", o resultado será conforme a Figura 2. Vale observar que o método setNota(Estudante, Nota) da classe Boletim possui a palavra-chave "estudante" e "nota". Como a classe Boletim também possui a palavra-chave "boletim", foi possível associar "boletim" ,"estudante" e "nota", por isso, a palavra "boletim" e "nota" foram indicadas.

Tabela 1 - Relacionamento Classes/Métodos e Palavras-chave

Tipo	Nome	Palavra-chave
Classe	Boletim	boletim
Método	setNota(Estudante, Nota)	estudante, nota

Classe	Estudante, EstudanteDAO	estudante
Classe	Materia	materia
Classe	Turma	matéria, estudante

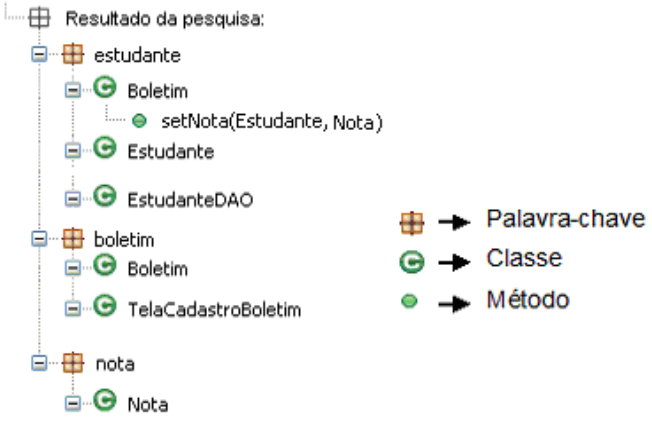

Figura 2 – Resultado da pesquisa pela palavra-chave "estudante"

A busca no CodeKeySearch é feita através das palavras-chave retiradas da busca efetuada nas metáforas previamente cadastradas.

5. Trabalhos relacionados

O *CodeBroker* [Ye, 2003] é um sistema de agentes inteligentes que recomenda o uso de componentes. As atividades do desenvolvedor, em particular comentários e assinaturas de métodos, são monitoradas de forma que seja possível fazer inferências e recomendações de componentes relevantes de bibliotecas Java que não são de conhecimento do desenvolvedor, além de apresentar um trecho de código como exemplo do componente sugerido. O *CodeBroker* não é atrativo para o uso no desenvolvimento com métodos ágeis, uma vez que é preciso que os componentes do repositório para a reutilização estejam devidamente documentados.

O RASCAL [McCarey et al, 2005] teve a atenção voltada para a busca e recuperação de componentes sem a necessidade de nenhum esforço adicional (nem de busca e nem de documentação) por parte do desenvolvedor, pois a recomendação é feita automaticamente. Foi desenvolvido como um plug-in para a IDE Eclipse e à medida que o código está sendo escrito, RASCAL monitora os métodos invocados e usa essa informação para recomendar um conjunto de métodos candidatos para o desenvolvedor. Essas recomendações são feitas com base nos algoritmos de filtragem colaborativa e filtragem baseada em conteúdo. Embora seja para aplicação em desenvolvimento XP, por não ser necessário nenhuma documentação, as recomendações, embora precisas, podem ser de difícil compreensão. Além disso, por não ter um controle sobre os termos utilizados no domínio, boas recomendações para reutilização podem ser deixadas de lado.

O *framework* Rise [Decker et al, 2005] baseia-se no uso de ferramentas Wiki como meio de comunicação assíncrona e para a gestão do conhecimento, servindo como repositório de conhecimentos. A tecnologia Wiki é uma solução leve para capturar, organizar e distribuir o conhecimento que é necessário ser levantado rapidamente para atender aos métodos ágeis. No entanto, esse *framework* provê a reutilização de experiências em projetos ágeis e não a reutilização dos sistemas desenvolvidos.

6. Conclusão

Este artigo apresentou uma abordagem apropriada para promover a reutilização de software no desenvolvimento com XP. Embora necessite a adoção de novas tarefas no processo de desenvolvimento, estas são complementares às práticas adotadas no XP. Além disso, o uso de uma ferramenta para cadastro de metáforas contribui com a automação do processo de levantamento de requisitos e permite uma maior propagação do conhecimento entre diferentes projetos de desenvolvimento. Como próximos passos, estudaremos formas de avaliar a ferramenta e compará-la às outras citadas na seção 5. Possivelmente, um projeto piloto pode ser utilizado para popular o repositório e, a partir desta base de dados, possibilitar a avaliação do uso destas ferramentas, viabilizando a identificação das suas reais contribuições.

Atualmente é necessária a existência de código fonte para a recuperação de classes e métodos. Contudo, um trabalho futuro consiste na utilização de anotações [Sun, 2008] no lugar de JavaDoc para descrever as palavras-chave. Essa estratégia possibilitaria a recuperação de classes e métodos binários (jar), mesmo sem a existência de código fonte, pois as anotações são mantidas mesmo depois do processo de compilação. Outro trabalho futuro consiste na extração automática das palavras-chave a partir das metáforas e das classes e métodos. A adoção de uma taxonomia , que permitisse a ligação entre as palavras-chave extraídas das metáforas com as palavras-chave extraídas das classes e métodos, possibilitaria a classificação e o entendimento desta base, podendo melhorar o sistema de recomendação.

Agradecimentos

Os autores gostariam de agradecer ao CNPq pelo apoio financeiro.

Referências

Bosch, J., 2000, Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach, Addison Wesley, 1st edition.

Decker, B., Ras, E., Rech, J., Klein, B. and Hoecht C. (2005) “Self-Organized Reuse Software Engineering Knowledge Supported by Semantic Wikis”. Workshop on Semantic Web Enabled Software Engineering, in the 4th International Semantic Web Conference, Galway, Ireland, November.

Eclipse Foundation. 2008. Eclipse. http://www.eclipse.org/ (visitado em Jan de 2008)

Frakes, W.B., Kang, K., 2005, "Software Reuse Research: Status and Future", *IEEE Transactions of Software Engineering*, v. 31, n. 7 (July), pp. 529-536.

Highsmith, J. and Cockburn, A. (2001) "Agile Software Development: The Business of Innovation". In: Computer, V. 34, N. 9, pp. 120-127, September.

Javadoc (2000-2004) http://java.sun.com/j2se/javadoc/writingdoccomments/index.html (vistando em 2008).

Krueger, C. W. (1992), "Software Reuse", ACM Computing Surveys, V. 24, N. 2, pp. 131-183, June.

Manifesto ágil (2001) http://www.agilemanifesto.org/ (visitado em Jan de 2008).

McCarey, F., Cinnéide, M. O. and Kushmerick, N. (2005) "An Eclipse Plugin to Support Agile Reuse". In Extreme Programming and Agile Processes in Software Engineering, V. 3556, pp. 162-170, May.

Soares, M. (2004) "Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software". In: INFOCOMP – Journal of Computer Science, V.3, pp. 8-13, November.

SUN (2008), "Java Tutorial, Annotations" http://java.sun.com/docs/books/tutorial/java/javaOO/annotations.html (visitado em Jan de 2008).

Teles, V. M. (2004), "Extreme Programming", NovaTec, 1a edição.

Werner, C.M.L., Mangan, M.A.S., Murta, L.G.P., Souza, R.P., Mattoso, M., Braga, R.M.M., Borges, M.R.S., 2003, "OdysseyShare: an Environment for Collaborative Component-Based Development". In: *IEEE Conference on Information Reuse and Integration (IRI)*, pp. 61-68, Las Vegas, USA, October.

Ye, Y. (2003) "Programming with an Intelligent Agent". In: IEEE Intelligent Systems, V.18, pp. 43-47, May-June.