Revista de Engenharia e Pesquisa Aplicada, Volume 2, Número 1, 2016

Reuso e Seus obstáculos na Engenharia de Software

Soares, L. S. L
Escola Politécnica de Pernambuco
Universidade de Pernambuco
50.720-001 - Recife, Brasil
lsls2@ecomp.poli.br

Resumo *Reusar um trecho de código ou um componente para desenvolver um novo sistema é uma atividade antiga e corriqueira. Afinal, os profissionais da Tecnologia da Informação e Comunicação ambicionam softwares que atendam os limites de custo e tempo e satisfaçam seus clientes. O processo de reusar requisitos é uma das várias alternativas para reutilizar artefatos de software, porém não é comumente empregado de maneira sistemática e segura. Este trabalho apresenta conceitos importantes, dados sobre o status do reuso em 30 empresas locais e uma análise de vantagens e desvantagens do reuso.*

Abstract *Reuse a piece of code or a component to develop a new system is an old and commonplace activity. After all, the professionals of Communication and Information Technology aspire for softwares which comply to the limits of cost and time and win satisfied customers. The process of reuse requirements is one of several alternatives to reuse software artifacts, but it is not commonly used in a systematic and safe way. This paper presents important concepts, a poll opinion about the status of reuse in 30 local companies and an analysis of advantages and disadvantages on reusing.*

1 Introdução

O processo de desenvolvimento de software mudou consideravelmente nas últimas décadas para atender às necessidades do mercado competitivo. Assim como a indústria de *software* aumenta a sua capacidade de produção, também aumenta a demanda por sistemas mais baratos, eficientes, confiáveis e que podem ser atuar sobre fortes restrições (não apenas de custo, mas também de tempo). A maneira tradicional de desenvolver sistemas a partir do zero já não atende a estas demandas, e a indústria de software tem sofrido algumas modificações para tentar resolver estes problemas. E reuso de *software* é uma das alternativas.

Reuso de *software* é uma abordagem que visa criar novos sistemas a partir de sistemas já existentes, de forma a reduzir o esforço necessário para desenvolver e manter estes novos sistemas [3].

Essa parece uma solução clara e simples e que todo programador utiliza diariamente, entre ferramentas e algoritmos já inventados. Porém, eles reutilizam informalmente. A falta de reuso generalizado e sistemático é o problema na engenharia do reuso [4].

Muitas técnicas têm sido desenvolvidas para apoiar a reutilização [1]. Dentre estas técnicas podemos citar: frameworks [6], que são aplicados quando se identifica que existem várias características comuns entre as aplicações existentes, um framework é um esqueleto de um aplicativo que pode ser personalizado para uma determinada aplicação, servindo como uma espécie de guia relacionado com o projeto de classes abstratas envolvidas [7]; Linhas de Produtos (LP) de software, que se concentra no desenvolvimento de uma família de aplicações [8] e que orientam o desenvolvimento de produtos de artefatos existentes, ao invés de produtos separados (um por um); Padrões de projeto, que propõem uma documentação padronizada de uma solução recorrente, já bem testada, proporcionando uma especificação explícita de interações entre classes e objetos [9], entre outros.

Artefatos potencialmente reusáveis vão além de fragmentos de código. Podem ser dados de testes, arquiteturas, documentação, especificações de requisitos, ou quaisquer outros artefatos de desenvolvimento, uma vez que a definição para um bom reuso não é o reuso de *software* por si só, mas o reuso das ideias para solucionar problemas humanos [2].

Recentemente, desenvolvedores e gerentes perceberam que a reutilização de requisitos e arquiteturas de software pode trazer benefícios maiores do que os obtidos apenas com a reutilização de componentes de software individuais. Apoiados nesta tendência, os autores de [5], também da Universidade de Pernambuco, propuseram um método para criar fragmentos requisitos genéricos visando sua reutilização. Seguindo outro viés, este trabalho foca no reuso de especificações de casos de uso.

2 Referencial Teórico

De acordo com [3], existem algumas dimensões relacionadas com as boas formas de reuso, são elas: abstração, seleção, especialização e integração.

- Abstração: esconde níveis de detalhe e implementação. Sem abstrações, o desenvolvedor teria de percorrer a coleção de artefatos reutilizáveis descobrindo o que cada um faz, quando poderia ser reutilizado e como seria reutilizado;

- Seleção: é processo onde o ator, construindo com reuso, compara e escolhe o que reusar a partir dos artefatos disponíveis.

- Especialização: é necessária no caso do elemento de reuso ser personalizado. Logo, a parte variável da abstração será utilizada para especializar de acordo com o que se deseja.

- Integração: consiste em combinar a coleção dos artefatos selecionados e especializados em um sistema completo.

A sequência destas dimensões não é por acaso. Este é o fluxo da construção que leva em consideração o reuso.

2.1 Abstração

Abstrair é generalizar, reduzindo o conteúdo da informação de um conceito ou fenômeno observável, normalmente para conservar apenas a informação que é relevante para um determinado propósito. Em [3] é esclarecido o conceito de abstração no reuso de *software*: uma abstração para um artefato de *software* é uma descrição sucinta que suprime os detalhes que não são importantes para um desenvolvedor de software e enfatiza a informação que é importante. Todas as abordagens para reutilizar software utilizam algum tipo de abstração, que é a característica essencial de qualquer técnica de reutilização.

A prática do reuso de software é dividida em dois eixos:

- Construção para reuso: consiste em identificar o conhecimento reusável, representá-lo de forma abstrata e o armazenar de forma indexada e classificada.

- Construção com reuso: consiste em buscar o conhecimento reusável e integrá-lo adaptando ao domínio específico de um novo sistema.

A partir da abstração diversas técnicas apresentam-se como boas oportunidades para dar suporte ao reuso, como por exemplo: a) Padrões de caso de uso, que são padrões (representados por especificações de caso de uso genéricas) cujo objetivo é ajudar a compreender a funcionalidade de um domínio de aplicação sobre o ponto de vista do usuário; b) Especificações de casos de uso, que são formas conhecidas e fáceis de ser usadas, tendo assim um bom potencial para documentar requisitos para reuso. E também os fragmentos de caso de uso [10], que contemplam além de especificações de caso de uso, regras e estruturas de dados a eles associados originando uma documentação mais completa.

3 Pesquisa local em reuso de software

Apesar de todo o potencial da reutilização de *software*, nem sempre é fácil colocar em prática. Algumas pessoas preferem construir a partir do zero ao invés de adaptar artefatos existentes construídos por outras pessoas. Com o objetivo de entender melhor a prática do reuso em empresas de desenvolvimento de software em Recife-Pernambuco-Brasil, uma pesquisa local foi realizada através de um questionário respondido por 50 pessoas (30 empresas), no período de janeiro a março de 2012. Esta pesquisa foi uma iniciativa do aluno de mestrado da Escola Politécnica de Pernambuco (UPE) Maurício Manuel Coelho Junior. Foram reunidas informações de profissionais de desenvolvimento e teste, analistas, designers e gerentes de TI. A maioria entre 5 a 10 anos de experiência e distribuídos em diversas áreas de atuação (Figura 1).

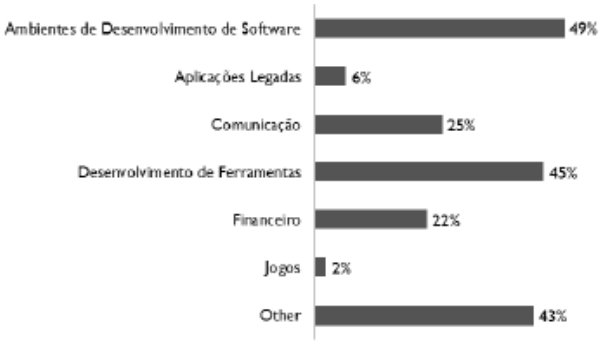

Fig 1. Áreas de atuação dos profissionais.

A partir dos resultados foi possível observar que a maioria das pessoas respondeu que durante o desenvolvimento, a reutilização de artefatos originados de projetos anteriores representa 25% a 50% do escopo. E a Figura 2 mostra a representação do reuso distribuído nas etapas de desenvolvimento de *software* citadas pela amostra pesquisada.

Fig 2. Etapas de desenvolvimento de software onde é aplicado o reuso.

Em uma questão do tipo subjetiva os profissionais expressaram quais eram os principais desafios para praticar reuso. Entre as respostas mais esclarecedoras, estão:

- Falta de um processo definido pela empresa com seus colaboradores;
- Falta de catálogos de busca de artefatos e seleção eficiente;
- Conhecimento/maturidade da equipe;
- Falta de padronização no desenvolvimento;
- Desenvolver pensando em reuso.
- O tempo curto dos projetos não permite analisar bem o que pode ser reusado.

Mesmo com todas as dificuldades, a maioria dos profissionais respondeu que o reuso vale a pena, desde que feito com responsabilidade.

4 Vantagens e Desvantagens

O reuso de software trás muitas vantagens quando utilizado de maneira efetiva e sistemática. A padronização facilita o entendimento de conceitos que se encontram em domínios diferentes. E como o desenvolvimento não ocorre a partir do zero, há um aumento da velocidade de produção de software. Além de crescer a confiança quando o que você reutiliza já foi testado e aprovado e foi integrado o seu sistema de maneira segura.

Porém, o reuso pode gerar um aumento no custo com manutenção e existe um tempo para buscar, compreender e adaptar os artefatos potencialmente reusáveis. Em [11] essas desvantagens são explicadas a partir da visão dos usuários dos recursos de software reutilizáveis. Eles podem ser classificados em três grupos:

- Os desenvolvedores originais.
- Indivíduos na mesma organização.
- Pessoas em diferentes organizações.

Quando os recursos reutilizáveis são bem classificados e facilmente recuperáveis, qualquer um na mesma organização deve ser capaz de usá-los para o desenvolvimento de sistemas de software. Porém, reutilização de *software* por pessoas em diferentes organizações implica em problemas com padronização e direitos legais.

5 Conclusões

Através da literatura e por pesquisas com profissionais da grande área de Tecnologia da Informação e Comunicação foi possível identificar as dificuldades e o potencial intrínseco ao reuso de *software*.

Este artigo considera os problemas enfrentados ao reusar artefatos através de uma abordagem não sistemática; entre os problemas estão ambiguidade, redundância e inconsistências. Todos estes pontos tornam o *software* difícil de ser mantido e entendido, contribuindo para a produção de sistemas de baixa qualidade.

Entretanto, é possível afirma que de maneira metódica e considerando todos os meios legais sobre licença e autorias de software, utilizar artefatos reusáveis torna-se uma atividade comum no dia-a-dia da maioria dos profissionais envolvidos no desenvolvedores de sistemas.

Referências

[1] I. Sommerville, “Engenharia de Software”, Pearson Addison Wesley, 2007.

[2] B. H. Barnes e T. B. Bollinger, “Making Reuse Cost-Effective”, IEEE Software, 1991, 8(1): 13-24.

[3] C. W. Krueger, “Software Reuse”, ACM Computing Surveys, 1992, 24:131-183.

[4] R. Pietro-Diaz, “Status report: software reusability”, IEEE Software, 10(3): 61-66

[5] A. R Araújo, M. Lencastre e D. S. Silveira, “A Method for the Creation of Requirements Fragments for Reuse in Information Systems”, 25º Simpósio Brasileiro de Engenharia de Software, 233-242, 2011.

[6] M. Fayad e R. Johnson, “Building Application Frameworks: Object-Oriented Foundations of Framework Design”, John Wiley & Sons, 1999.

[7] R. Johnson e B. Foote, “Designing Reusable Classes”, Journal of Object- Oriented Programming, 1:22-35, 1988.

[8] P. Clements e L. Northrop, “Software Product Lines: Practices and Patterns”, Boston: Addison Wesley, 2002.

[9] E. Gamma, R. Helm e J. Vlissides, “Design Patterns: Elements of Object-Oriented Software”, Addison Wesley, 1995.

[10] F. G. Dias, E. A. Schmitz, M. L. M. Campos, A. L. Correa e A. J. Alencar, “Elaboration of Use Case Specifications: an approach based on Use Case Fragments”, ACM symposium on Applied computing, 614-618, 2008.

[11] Y. Kim e E. Stohr, “Software reuse: issues and research directions”, Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, 612-623, 1992.