Padrões de Projeto, Qualidade de *Software* e Reúso de *Software*: Técnicas para um bom *Software*

Douglas Augusto de Sousa Silva[1]

Fabrício José de Sousa[2]

João Ramyllo Guedes Araújo Feitosa[3]

Renan de Sousa Rêgo[4]

Sebastião Sales Rodrigues Macedo[5]

Universidade Federal do Piauí (UFPI - CSHNB)

64607-675 – Picos – PI – Brazil

douglas-augusto@hotmail.com, fabriciosousa16@hotmail.com, joaoramyllo@hotmail.com, renansccp077@gmail.com,sebastiao_sales@hotmail.com

Abstract. *This article makes a brief approach to the terms Design, Software Quality and Reuse of Software that are terms present in the computation, more specifically in the areas of programming, seeking to explain concepts, characteristics and relation between them.*

Resumo. *Este artigo faz uma breve abordagem em cima dos termos Padrões de Projeto, Qualidade de Software e Reúso de Software que são termos presentes na computação, mais especificamente nas áreas de programação, buscando explanar conceitos, características e relação entre os mesmos.*

1. Introdução

O planejamento e o desenvolvimento de um *software* não são tarefas tão simples. O tempo de desenvolvimento de um *software* está diretamente relacionado aos custos e aos gastos dos vários outros recursos, tendo ocasiões que esses gatos chegam a ocasionar até a não entrega do produto. Alguns aspectos podem ajudar e facilitar na conclusão rápida e eficiente dessas tarefas.

Neste artigo serão abrangidos alguns desses aspectos, mais especificamente Padrões de Projeto, Qualidade de *Software* e Reúso de *Software*, que dentro deste contexto, são técnicas para melhor aprimoramento, organização, alcance da qualidade e produtividade nos processos do produto de *software* até sua entrega final.

2. Padrões de Projeto

Padrões de projeto podem ser vistos como uma solução que já foi testada para um problema. Desta forma, um padrão de projeto geralmente descreve uma solução ou uma instância da solução que foi utilizada para resolver um problema específico. Padrões de projeto são soluções para problemas que alguém um dia teve e resolveu aplicando um modelo que foi

documentado e que você pode adaptar integralmente ou de acordo com necessidade de sua solução.

> Padrões para arquitetura de *software* são soluções de eficiência já comprovadas e amplamente utilizadas para a resolução de problemas comuns em projeto de *software*. Estas soluções são desenvolvidas e conhecidas por especialistas e tornam-se padrões por serem reutilizadas várias vezes em vários projetos e por terem eficácia comprovada (ALEXANDER, 1978).

De acordo com Alexander (1978), um padrão descreve um problema que ocorre inúmeras vezes em determinado contexto, e descreve ainda a solução para esse problema, de modo que essa solução possa ser utilizada sistematicamente em distintas situações.

De acordo com o livro: "Padrões de Projeto: soluções reutilizáveis de *software* orientado a objetos", os padrões "GoF" são divididos em 23 tipos. Em função dessa grande quantidade de padrões, foi necessário classificá-los de acordo com as suas finalidades.

> Um padrão de projeto nomeia, abstrai e identifica os aspectos-chave de uma estrutura de projeto comum para torná-la útil para a criação de um projeto orientado a objetos reutilizável. O padrão de projeto identifica as classes e instâncias participantes, seus papéis, colaborações e a distribuição de responsabilidades. Cada padrão de projeto focaliza um problema ou tópico particular de projeto orientado a objetos (GAMMA et al., 2000, p. 20).

Os padrões GoF foram divididos e categorizados de acordo com a natureza do problema que eles resolvem, essas categorias sendo:

- Padrões de Criação: Tem como objetivo abstrair a instanciação de objetos.
- Padrões estruturais: Os padrões dessa categoria se preocupam em melhor organizar a estrutura das classes e os relacionamentos entre classes e objetos.
- Padrões comportamentais: Os padrões dessa categoria atuam diretamente na delegação de responsabilidades, definindo como os objetos devem se comportar e se comunicar.

Os principais padrões são:

1. Padrões de Criação (*Creational*):

   Abstract Factory, *Builder*, *Factory Method*, *Prototype*, *Singleton*.

2. Padrões de Estrutura (*Structural*):

   Adapter, *Bridge*, *Composite*, *Decorator*, *Facade*, *Flyweight*, *Proxy*.

3. Padrões de Comportamento (*Behavioral*):

Chain of Responsability, *Command*, *Interpreter*, *Iterator*, *Mediator*, *Memento*, *Observer*, *State*, *Strategy*, *Template Method*, *Visitor*.

3. Qualidade de *Software*

A qualidade de *software* é uma área de conhecimento da engenharia de *software* que objetiva garantir a qualidade do *software* através da definição e normatização de processos de desenvolvimento. Apesar dos modelos aplicados na garantia da qualidade de *software* atuarem principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente.

Para Bourque e Fairley (2014), qualidade de *software* são as características desejadas de produtos de *software*, a extensão em que um produto de *software* em particular possui

essas características e aos processos, ferramentas e técnicas que são usadas para garantir essas características.

Apesar dos modelos aplicados na garantia da qualidade de *software* atuarem principalmente no processo, o principal objetivo é garantir um produto final que satisfaça às expectativas do cliente, dentro daquilo que foi acordado inicialmente. Um *software* de qualidade é fácil de usar, funciona corretamente, é de fácil manutenção e mantém a integridade dos dados para evitar possíveis falhas, fora ou não, do seu controle.

De acordo com a ISO/IEC 9126-1, qualidade de *software* é a totalidade de características de um produto de *software* que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas.

Os principais atributos que um *software* deve possuir para que possa ser considerado um *software* de qualidade são os seguintes:

- Funcionalidade
- Confiabilidade
- Usabilidade
- Eficiência
- Manutenibilidade
- Portabilidade

É importante destacar que essa lista tem como objetivo ser exaustiva. Portanto, de acordo com a norma, todas as qualidades que venham a ser requisitadas ao *software* estão presentes nessa lista.

4. Reúso de *Software*

Reúso de *software* é uma estratégia em que o desenvolvimento de *software* é baseado na reutilização de *software* existente.

Para Meena et al. (2011), reutilização de *software* tem sido considerada como uma das áreas mais importantes para melhorar a produtividade de desenvolvimento de *software* e da qualidade do *software*.

Reúso de *software* é considerado uma maneira promissora para desenvolver sistemas. Ele ajuda uma organização a aumentar sua produtividade e a qualidade. Reúso de *software* pode ser aplicado a qualquer ciclo de vida de produtos, não apenas ao código fonte (JONES, 1993).

A reutilização de *software* se baseia no uso de ideias, produtos ou soluções anteriormente elaboradas ou alcançadas para criação de um novo *software*, com objetivo de melhorar significativamente a qualidade e a produtividade.

Existem várias técnicas e abordagens que norteiam e ajudam na tarefa de reúso de *software*. Essas técnicas evidenciam muitas vezes quais ações e estratégias a se seguir para que o desenvolvimento de um projeto seja feito em acordo com os padrões de reúso, também como utilizar os conteúdos já existentes evitando o retrabalho. Duas técnicas muito utilizadas para aplicar o reúso de *software* são:

1. Linhas de Produto de *Software*

O desenvolvimento por meio de linhas de produto de *software* é uma abordagem utilizada para alcançar o reúso de forma bastante abrangente (*reuse-in-the-large*). O uso

dessa abordagem não possui a premissa de reusar apenas código, mas sim um conjunto de *assets* (características) do sistema (FERREIRA e NAVES, 2011).

2. Engenharia Reversa e Reengenharia

O processo de engenharia reversa pode gerar códigos, artefatos e componentes que muitas vezes podem ser diferentes do sistema original, mas proporcionam ao desenvolvedor um conhecimento adicional sobre o problema gerando melhorias que não foram feitas no processo progressivo e que dão mais ênfase e qualidade ao reúso (FERREIRA e NAVES, 2011).

Dos benefícios da reutilização, os principais considerados pela maioria dos autores, são:

- Aumento da produção com a redução no esforço do desenvolvimento.
- Redução dos custos e do prazo de entrega, pois se o esforço de desenvolvimento diminui logo o tempo de entrega também diminui e a quantidade de homens hora necessária a ser paga também.
- Como as soluções aplicadas foram anteriormente testadas e validadas, a probabilidade de que esteja correta é ainda maior, portanto temos um aumento da qualidade do produto final.
- Padronização dos produtos desenvolvidos pela empresa, pois como as soluções reusáveis foram desenvolvidas segundo uma padronização pré-definida, o reúso em um sistema provoca consequente padronização e agilidade na manutenção das aplicações devido a esta padronização da arquitetura.

5. Trabalhos Relacionados

Um *software* bem projetado e desenvolvido ao final de todas suas etapas de desenvolvimento terá sido composto por uma série de técnicas que auxiliam na confecção e estruturação do seu código fonte, para um melhor entendimento, manutenibilidade e funcionamento.

A Qualidade de *Software* sendo uma dessas técnicas, abrange o projeto em si como um todo, ou seja, esta engloba o funcionamento de todas as outras técnicas implantadas.

Um estudo feito em 2013, pela estudante do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba (IFPB), Nadja N. Rodrigues, de título: Praticando Qualidade de *Software*: Ensinando e Aprendendo seus Valores através de Ambiente Real, propôs a ideia de apresentar os principais elementos que relacionam Engenharia de *Software* e Qualidade de *Software*, no contexto de uma metodologia desenvolvida para ensinar aos alunos algumas boas práticas para a construção de *softwares* de qualidade. O lado prático do estudo baseou-se na instanciação de projetos reais de uma Empresa Júnior, executados através de ambiente específico, em sala de aula. O ambiente permitiu que o aluno entendesse a relação entre qualidade e processos de *software*, através dos papéis e atividades, *templates*, procedimentos e ferramentas utilizados no desenvolvimento de projetos.

6. Discussão

Tais técnicas citadas mostram inúmeras opções aos desenvolvedores, opções essas que tornam o desenvolvimento de qualquer produto de *software* um procedimento bem mais descomplicado, produtivo e com qualidade. Ressalta-se que, uma vez que essas técnicas são aplicadas a um projeto de *software*, as chances de possíveis situações indesejadas no desenvolver do produto ou na sua utilização, são consideravelmente menores.

7. Conclusões

Podemos concluir do seguinte estudo apresentado, que inúmeras são as alternativas para um bom desenvolvimento de um projeto de *software*. Essas alternativas variam de acordo com o contexto de cada problema para as quais foram projetadas para ser uma solução. Tais contextos podem se referir aos diversos aspectos do *software*, podendo ser, padronização do projeto a ser seguido, medidas de qualidade, produtividade, dentre inúmeros outros.

Em projetos menores pode-se até achar que a integração dessas técnicas pode demandar mais trabalho e tempo, dizer isso é um equívoco, pois no decorrer da confecção do projeto, os benefícios trazidos por tais técnicas são consideravelmente perceptíveis.

Projetos maiores, em si já demandam até um certo conhecimento mais avançado, tendo em vista essa situação, os desenvolvedores costumam já ter uma percepção maior sobre todo o contexto, sabendo todas as medidas necessárias para um bom desenvolvimento.

Seja em projetos grandes ou de menor estrutura, as técnicas apresentadas neste trabalho se fazem de grande importância e utilidade, já que, ninguém quer fazer esforços desnecessários ou até mesmo trabalhar perdido. Padrões de Projeto, Qualidade de *Software* e Reutilização de *Software* são de forma sucinta, um atalho para a qualidade e rapidez na entrega do produto final.

8. Referências

ABNT. Associação Brasileira de Normas Técnicas. NBR ISO/IEC 9126-1: Engenharia de software - Qualidade de produto. Rio de Janeiro: ABNT, 2003. 21 p.

ALEXANDER, Christopher: A Pattern Language, Oxford Press, Oxford, R. Unido, 1978.

BOURQUE, Pierre; Fairley, Dick (2014). SWEBOK 3.0 Guide to the Software Engineering Body of Knowledge. [S.l.]: IEEE Computer.

DEVMEDIA, Devmedia. Reutilização de Software: Revista Engenharia de Software Magazine 39. 2011. Disponível em: <https://www.devmedia.com.br/reutilizacao-de-software-revista-engenharia-de-software-magazine-39/21956>. Acesso em: 13 out. 2018.

FERREIRA, Hiran N. M.; NAVES, Thiago F. Reuso de Software: Suas vantagens, técnicas e práticas. 2011. 8 p. Artigo (Computação) - Faculdade de Computação, Universidade Federal de Uberlândia, Uberlândia, 2011.

GAMMA, Erich et al. Padrões de Projetos: Soluções Reutilizáveis. 1. ed. São Paulo: Bookman, 2000. 368 p.

JONES, C. Software return on investment preliminary analysis. 1993.

MACORATTI, José Carlos. Padrões de Projeto: Design Patterns. Disponível em: <http://www.macoratti.net/vb_pd1.htm>. Acesso em: 11 out. 2018.

MEENA Jha and L. O'Brien. A comparison of software reuse in software development communities. In Software Engineering (MySEC), 2011 5th Malaysian Conference in, pages 313–318, 2011.

RODRIGUES, Nadja N. Praticando Qualidade de Software: Ensinando e Aprendendo seus Valores através de Ambiente Real. 2013. 12 p. Artigo (Computação)- Instituto Federal de

Educação, Ciência e Tecnologia da Paraíba, Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, João Pessoa, 2013.

SOUZA, Givanaldo Rocha de. Qualidade de Software. Disponível em: <https://docente.ifrn.edu.br/givanaldorocha/disciplinas/engenharia-de-software-licenciatura-em-informatica/qualidade-de-software>. Acesso em: 12 out. 2018.

SAMETINGER, Johannes. Software Engineering with Reusable Components. Springer-Verlag, 1997.

WIKIPEDIA, Enciclopédia Livre. Qualidade de software. Disponível em: <http://pt.wikipedia.org/wiki/Qualidade_de_Software>. Acesso em: 12 out. 2018.