FERCHEM: Uma nova abordagem para inspeção de documentos arquiteturais baseados em checklist

César A. V. Mariano[1], **Fernanda B. Faria**[1], **Humberto L. Antonelli**[1], **Mayara P. da Costa**[1], **Liliane N. Vale**[1]

[1]Universidade Federal de Goiás (UFG) – Campus Catalão – Departamento de Ciência da Computação – Avenida Dr. Lamartine Pinto de Avelar, 1120 – Setor Universitário – 75.704-020 – Catalão – GO – Brazil

{cesarmineirofla, ferrbontempo, lordantonelli, mmayarapires}@gmail.com

Abstract. *This paper shows a new approach about the checklist to inspection of software architectural document. Initially, a literature review is presented. In sequence, the overall architecture of the new methodology is illustrated with a case study to evaluate the proposal. Thus, this approach seeks to discuss the reduction of time and development costs is to prevent errors, ensure software quality and give to the developer and the customer an idea of what will be the software, performing the inspection prior to the implementation.*

Resumo. *Esse artigo mostra uma nova abordagem do checklist para inspeção de documento arquitetural de software. Inicialmente, uma revisão bibliográfica é apresentada. Em sequência, a arquitetura geral da nova metodologia é ilustrada, juntamente com o estudo de caso para avaliação da proposta. Dessa forma, nesta abordagem, procura-se discutir a diminuição de tempo e custos no desenvolvimento ao prevenir-se de erros, garantir a qualidade do software e dar ao desenvolvedor e ao cliente uma ideia do que será o software, realizando a inspeção anterior à implementação.*

1. Introdução

Quando tentamos solucionar um problema, é possível identificar diversas soluções que poderiam ser utilizadas visando resolvê-lo. Contudo, outros fatores como custo e eficiência influenciam na escolha da solução a ser adotada. No contexto do desenvolvimento de software, o mesmo pode ser observado ao se analisar os requisitos visando à construção de um software: várias soluções computacionais podem ser definidas para atender a esses requisitos, mas uma análise deve ser feita para definir a mais adequada ao contexto de desenvolvimento [Barcelos 2006].

Pela definição do *Institute of Electrical and Electronics Engineering* (IEEE), arquitetura de um sistema de software é a estrutura ou estruturas do sistema, que contempla elementos de software, as propriedades visíveis externamente desses elementos e o relacionamento entre eles [Papo 2008].

A importância da inspeção de software se dá pelo fato de analisar e verificar representações de sistemas como documento de requisitos, diagramas de projeto e código-fonte do programa. Pode-se usar inspeções em todos os estágios do processo. Podem ser suplementadas por alguma análise automática de texto-fonte de um sistema ou de um documento associado [Sommerville 2007].

Esse artigo propõe uma nova metodologia de inspeção de software baseada em *checklist*. Neste contexto, é adiantado a etapa de inspeção de teste, iniciando-a pela arquitetura, visto que neste ponto não temos o código disponível. É feita, então, uma análise dos componentes do *software*, afim de minimizar os erros em código, tornando o trabalho dos desenvolvedores mais ágil e menos dispendioso.

2. Trabalhos Relacionados

Em [Barcelos and Travassos 2006] foi considerado a avaliação sobre requisitos funcionais e de requisitos de qualidade. A análise dos resultados foi feita mostrando que a abordagem proposta melhora a qualidade do software e torna a revisão uma atividade relevante para o sucesso do projeto. Neste trabalho, além da verificação dos requisitos funcionais, foi feita uma avaliação dos requisitos não-funcionais, visando a melhoria do sistema, em menor tempo de desenvolvimento e com menor custo.

Em [Sayão et al. 2003] é abordado técnicas para melhoria da qualidade de um documento de requisitos gerados a partir de metodologias clássicas de desenvolvimento de software. Também é mencionado que através de inspeções podem ser encontradas falhas e defeitos num sistema antes que se passe para a próxima etapa de desenvolvimento. Com ajuda destes dados foram definidas as técnicas de inspeção e, posteriormente, uma comparação entre as técnicas a fim de obter melhores conclusões. O FERCHEM, propõe a utilização da inspeção na fase de projeto do software, o que difere da abordagem supracitada.

Em [Rodrigues 2010] encontra-se uma definição de *checklist* e qual sua finalidade, mostrando que esta abordagem é (e deve ser) muito utilizada. Esta fonte auxilia na definição de *checklist* e demonstra ser uma abordagem vantajosa para o desenvolvimento de sistemas.

Em [Braga 2009] argumenta-se sobre requisitos não funcionais, que descrevem as qualidades do sistema (como o sistema é) e destaca alguns requisitos de qualidade, como usabilidade e performance. Deste foram aproveitados algumas idéias para serem usadas no estudo de caso deste trabalho.

Em [Kalinowski and Spínola 2007], há uma introdução sobre inspeção. São mencionadas algumas variáveis que afetam positiva ou negativamente a atividade de inspeção e que devem ser consideradas. Através de gráficos, são mostrados os benefícios da inspeção de software. Assim observa-se novamente que a inspeção melhora a qualidade do sistema e detecta erros antes que passe para a próxima etapa do desenvolvimento de *software*.

A metodologia deste trabalho consiste em revisão bibliográfica, para posterior estudo de caso, visando a validação do método proposto, portanto, a partir das pesquisas feitas desenvolveu-se um estudo de caso para um sistema projetado pelos próprios autores deste artigo. Neste estudo é explicitado como funciona o sistema e foi utilizado um *checklist* para validar os requisitos do mesmo.

3. Técnicas de inspeção

Inspeção é uma técnica de análise desenvolvida por Fagan em 1972, com o objetivo de aumentar a qualidade de software e a produtividade dos programadores. A inspeção de

software é um tipo particular de revisão que pode ser aplicada a todos os artefatos de software, possuindo um processo de detecção de defeitos rigoroso, através de estruturas e regras bem definidas [Silva et al. 2004, Sayão et al. 2003].

Para a detecção de defeitos, a inspeção é uma das técnicas que apresenta mais eficiência, devido ao número de defeitos/tempo gasto encontrados e com um custo de 10 a 15% do orçamento de desenvolvimento [Pagliuso et al. 2002]. É importante ressaltar que a inspeção pode diminuir o tempo de desenvolvimento, pois remove defeitos em fases anteriores, reduzindo de maneira significativa os custos com correção nas fases seguintes. Visando auxiliar o aprimoramento da Inspeção de Documentos Arquiteturais são utilizadas algumas técnicas conhecidas de leitura, como a *Ad-hoc*, *checklist* e leitura baseada em perspectivas [Sayão et al. 2003, Silva et al. 2004].

Checklists são listas de verificações com itens a serem observados, tarefas a serem cumpridas, materiais a serem comprados, ou seja, é uma lista em que são colocados itens que podem fazer falta em alguma tarefa ou em algo que se esteja planejando ou executando, evitando assim futuros esquecimentos, falhas, faltas. Ele pode ser usado não só por empresas, mas por qualquer pessoa que queira organizar algo a ser feito [Rodrigues 2010].

Para o desenvolvimento de software não é diferente. É necessário definir metas e objetivos a cumprir em determinadas partes do desenvolvimento. Tendo esse documento os projetistas podem reusá-lo e adaptá-lo em outros projetos, o que torna o desenvolvimento mais ágil e eficiente. *Checklist* pode ser usado também para a inspeção de documentos arquiteturais.

Entre as técnicas de inspeção existentes, *checklist* é uma das que pode ser utilizada para identificar defeitos em documentos arquiteturais [Barcelos 2006]. O uso dessa abordagem é justificado pelo fato de técnicas ad-hoc não oferecer apoio ou procedimento de execução formal e sistemática de inspeção e, também porque técnicas de leitura são procedimentos que visam guiar individualmente os inspetores no entendimento de um artefato de software [Barcelos 2006].

Várias abordagens baseadas em *checklist* já foram definidas visando a identificação de defeitos em documentos arquiteturais [Barcelos 2006]. Para que o uso de *checklist* seja eficiente, é necessário tomar algumas decisões e registrá-las, para definir os itens de avaliação deste *checklist*.

4. A Abordagem FERCHEM

O método utilizado nesse artigo visa uma inspeção de documentos arquiteturais baseada em *checklist*, em um momento anterior a implementação do software. A inspeção é importante pelo fato de reduzir o retrabalho e garantir a qualidade do software [Kalinowski and Spínola 2007]. Na Figura 1 é mostrado os principais itens que compõe a arquitetura do FERCHEM de acordo com seus propósitos iniciais.

No item assinalado com o número 1, que corresponde à documentos arquiteturais, mostra os itens que avaliam a consistência do documento. Em 2, os itens que avaliam o atendimento aos requisitos e por fim, em 3 tem-se os itens que avaliam a abordagem para atender aos requisitos de qualidade.

O papel das visões arquiteturais é representar a arquitetura sob diferentes perspectivas. Para avaliar se elas estão representando a mesma arquitetura, itens foram definidos

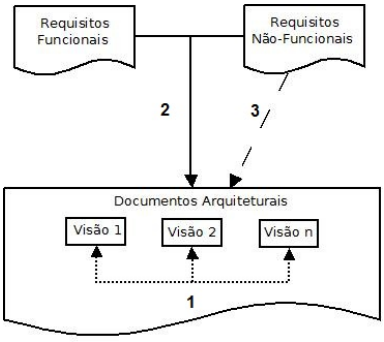

Figura 1. Grupos de itens de avaliação que compõem o *checklist* proposto

para avaliar a consistência das informações de uma visão arquitetural e também a consistência entre diferentes visões [Barcelos and Travassos 2006].

O papel das visões arquiteturais é representar a arquitetura sob diferentes perspectivas, portanto, para avaliar se elas estão representando a mesma arquitetura, itens foram definidos para avaliar a consistência das informações de uma visão arquitetural e também a consistência entre diferentes visões [Barcelos and Travassos 2006].

Tendo como ponto de partida a abordagem de Barcelos e Travassos (2006), foi verificado que a inspeção, na grande maioria das vezes, é feita em nível de código-fonte, o que se torna mais dispendioso caso haja algum erro ou requisitos controversos.

Ao utilizar a abordagem antes da implementação, é possível fazer a previsão de erros, de custos e de tempo que serão gastos no desenvolvimento do sistema, dando ao desenvolvedor e ao cliente uma boa margem do que será o software.

O foco desta nova proposta considera aspectos de requisitos funcionais e requisitos não-funcionais em que estes últimos aspectos descrevem características como qualidade, proteção, segurança e desempenho do sistema (como o sistema é) ao invés de suas funcionalidades (o que ele faz) [Braga 2009], visando uma maior qualidade do software.

Nesse contexto, é necessário avaliar a consistência do documento, sua clareza em relação aos requisitos e a utilização dos mesmos. A intenção dessa checagem é verificar se todas as funcionalidades especificadas foram atendidas pela arquitetura e que todos os elementos arquiteturais foram definidos com base nos requisitos especificados. Ou seja, determina se os itens foram avaliados de acordo com o atendimento aos requisitos [Barcelos and Travassos 2006].

O padrão IEEE 830[1], que recomenda práticas para especificação de requisitos de software, define atributos de qualidade que um documento de requisitos deve possuir [Kalinowski and Spínola 2007], e são esses atributos, mostrados na Tabela 1, que iremos utilizar para extrair o *checklist* de requisitos funcionais e requisitos não-funcionais do documento arquitetural.

Para avaliar a consistência do documento é necessário avaliar as informações nele contidas e as diversas visões, de desenvolvedor e *stakeholders*, para melhor representar a arquitetura em diversas perspectivas.

[1]http://standards.ieee.org/findstds/standard/830-1998.html

Esse *checklist* pode ser incorporada aos diversos tipos de sistemas, inclusive para dispositivos móveis, como apresentado a seguir, em um aspecto abordado no estudo de caso. Pois se trata de um documento genérico, contendo os principais requisitos a serem analisados no desenvolvimento e deve ser integrado à documentação do software, mostrando o que foi analisado e onde são necessárias as alterações no desenvolvimento do *software* do sistema.

Tabela 1. Principais aspectos considerados para elaboração do *checklist*.

Item	Descrição
Omissão	Requisitos necessários na descrição do sistema não são contemplados pelos casos de uso.
Ambiguidade	Dois ou mais requisitos representando os mesmos objetivos.
Inconsistência	Dois ou mais requisitos redundantes.
Fato incorreto	Um requisito descreve um fato que não é verdadeiro, considerando as condições solicitas para o sistema.
Requisitos desnecessário	Requisito que poderia estar incluído na descrição de outro requisito, que já contemplou a idéia.
Usabilidade	A interface é amigável para o usuário.
Performance	Tempo de resposta é satisfatória ao usuário.
Proteção	Capacidade de proteger os dados contra falhas do sistema e a capacidade do sistema de se auto recuperar das falhas.
Segurança	Descrição da segurança, confidencialidade e integridade dos dados.

5. Estudo de Caso

O sistema apresentado para estudo de caso é um conversor de unidades para celular, que está sendo desenvolvido pelos próprios autores do artigo. O aplicativo converte medidas de comprimento, peso, unidades computacional, velocidade e tempo.

Ele tem por objetivo ser amigável aos usuários através de uma interface simples e intuitiva. Os cálculos realizados para conversão devem obter alta precisão. O conversor se limitará a um ambiente gráfico 2D, sendo executável em plataformas móveis. Na Figura 2(a) é mostrado o diagrama de caso de uso do conversor com suas principais funcionalidades.

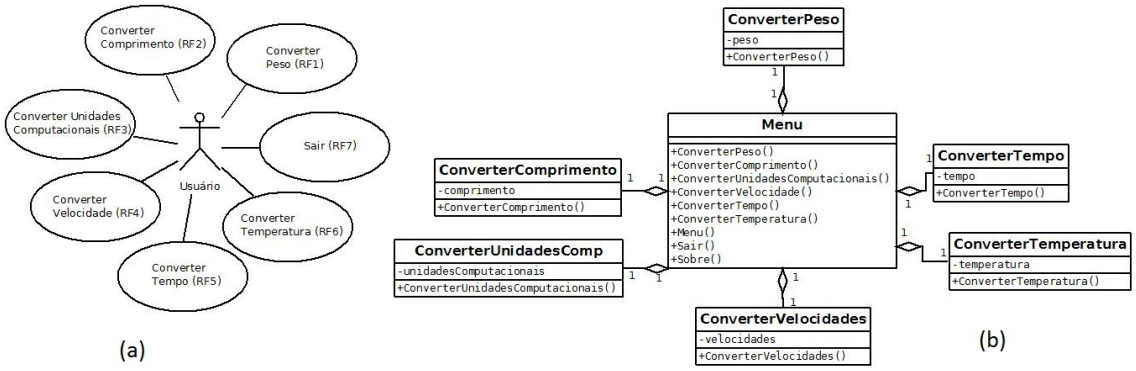

Figura 2. Estrutura Geral do Sistema: (a) Diagrama de Caso de Uso e (b) Diagrama de Classes

O diagrama de classe da Figura 2(b) mostra a arquitetura do sistema, com seus respectivos componentes funcionais e relacionamentos, dando uma visão da estrutura estática do que é o sistema e do que ele se propõe a fazer.

Para estudo de viabilidade da abordagem proposta neste trabalho, foi aplicada a metodologia FERCHEM no sistema que os autores estão desenvolvendo. Os autores inspecionaram o documento arquitetural respondendo um questionário baseado em *checklist*, que teve como objetivo principal a verificação da existência de defeitos que devem ser corrigidos antes do início das próximas fases de desenvolvimento. A Tabela 2 e a Tabela 3, no final deste trabalho, mostram os resultados obtidos.

Os requisitos funcionais cobrem a maioria das características observadas, porém a maioria dos requisitos não-funcionais não são previstos. Isso se dá pelo fato de o sistema ser simples e de pequeno porte. Portanto não foi visado, até então, proteção em relação o acesso aos dados, não houve tratamento de erros, entre outros.

A iniciativa deste estudo tem como finalidade reforçar a importância das atividades de verificação e validação para a qualidade do software, sendo influenciada por inúmeros fatores como redução de esforço, orçamento e a própria utilidade prática. Os resultados experimentais através do estudo realizado indicou a viabilidade do FERCHEM, mostrando ser interessante estender o seu estudo para uma avaliação mais detalhada.

6. Conclusão

Após o estudo das técnicas de inspeção mais utilizadas, optamos pelo uso do *checklist*, pois é a mais flexível para diferentes tipos de softwares, e mostra diversas visões existentes no mesmo.

A principal vantagem de inspeção arquitetural é o fato da detecção de erros ser feita antes da implementação do software, reduzindo o retrabalho ao mesmo tempo em que garante a qualidade do software.

A proposta foi aplicada a um aplicativo móvel de pequeno porte. Entretanto para sistemas desta categoria o método se mostrou eficiente, pois cobriu a grande maioria das informações relevantes do aplicativo.

Para trabalhos futuros é proposta uma avaliação mais detalhada sob outras perspectivas, procurando outras visões e *stakeholders* para verificação e validação dos requisitos, visando tornar a metodologia mais eficiente.

A principio será feita uma avaliação de consistência do modelo, ou seja, se o modelo é aplicável a qualquer natureza e complexidade de software. Será avaliado também a completeza do modelo, analisando se esse cobre e responde pela avaliação justa de todos os requisitos não-funcionais de diversos tipos de sistemas.

Em outro contexto faz-se necessário também a avaliação do impacto, isto é, se um requisito da arquitetura quando inspecionado é avaliado individualmente, ou se é considerado a influência que este requisito causa sobre outros requisitos do esqueleto do software.

Referências

Barcelos, R. F. (2006). *Uma abordagem para inspeção de documentos arquiteturais baseada em checklist*. Mestrado, Universidade Federal do Rio de Janeiro (UFRJ), Rio de

Janeiro, RJ.

Barcelos, R. F. and Travassos, G. H. (2006). Arqcheck: Uma abordagem para inspeção de documentos arquiteturais baseada em checklist. *V Simpósio Brasileiro de Qualidade de Software*, pages 174–188.

Braga, B. (2009). Requisitos não funcionais. Disponível em: http://www.brunobraga.com.br/2009/02/12/requisitos-nao-funcionais/. Acesso em: 17 ago. 2011.

Kalinowski, M. and Spínola, R. O. (2007). Introdução à inspeção de software. *Engenharia de Software Magazine*, pages 68–74.

Pagliuso, P. B. B., Tambascia, C. A., and Villas-boas, A. (2002). Melhoria da inspeção de requisitos segundo a técnica de leitura baseada em perspectiva. Blumenau, SC. XI SEMINCO – Seminário de Computação, Universidade Regional de Blumenau.

Papo, J. (2008). Arquitetura de software: O que é e como documentar. Disponível em: http://josepaulopapo.blogspot.com/2008/10/arquitetura-software.html. Acesso em: 17 ago. 2011.

Rodrigues, M. (2010). Checklist – o que é e qual é a sua importância? Disponível em: http://www.sucessonews.com.br/checklist-o-que-e-e-qual-e-a-sua-importancia. Acesso em: 16 ago. 2011.

Sayão, M., Staa, A. v., and Leite, J. C. S. d. P. (2003). *Qualidade em Requisitos*. Monografia em ciência da computação, DI/PUC-Rio, Rio de Janeiro, RJ.

Silva, L. F. S., Chapetta, W. A., and Travassos, G. H. (2004). Inspeções de requisitos de software utilizando pbr e apoio ferramental. Brasília, DF, Brasil. Simpósio Brasileiro de Qualidade de Software, UCB - Universidade Católica de Brasília.

Sommerville, I. (2007). *Engenharia de Software, 8ª Edição*. Pearson Addison-Wesley, Sã Paulo, SP, 8 edition.

Tabela 2. *Checklist* para Requisitos Não-Funcionais

Requisitos Não-Funcionais	**Sim**	**Não**
Os requisitos de desempenho (tais como tempo de resposta, armazenamento de dados, etc.) foram definidos?		X
As restrições de tempo-real foram especificadas em detalhe suficiente?		X
A precisão e acurácia dos cálculos foram especificadas?	X	
As restrições e dependências foram claramente descritas?	X	
O software foi escrito de modo que possa evoluir para atender às necessidades de mudança de cliente?		X
O nível de confiança do software tem características que incluem confiabilidade, proteção e segurança?		X
O software utiliza recursos do sistema, como memória e ciclos do processador?	X	
O software é usável sem esforço excessivo, pelo tipo de usuário o qual ele foi proposto?	X	

Tabela 3. *Checklist* para Requisitos Funcionais

Requisitos Funcionais	RF1		RF2		RF3		RF4		RF5		RF6		RF7	
	Sim	Não	Sim	Não	Sim	Não	Sim	Não	Sim	Não	Sim	Não	Sim	Não
1. O requisito exibe a distinção clara entre funções e dados?	X		X		X		X		X		X		X	
2. O requisito define todas as informações a ser apresentada aos usuários?	X		X		X		X		X		X		X	
3. O requisito descreve as respostas do sistema ao usuário devido à condições de erro?		X		X		X		X		X		X		X
4. O requisito está descrito claramente, é conciso e está apresentado de forma não ambígua?	X		X		X		X		X		X		X	
5. O requisito é testável?	X		X		X		X		X		X		X	
6. O requisito é implícito ou ambíguo?		X		X		X		X		X		X		X
7. O requisito é conflitantes?		X		X		X		X		X		X		X
8. Existem áreas não tratadas no documento de requisitos que precisam ser consideradas?	X		X		X		X		X		X		X	
9. Se o requisito envolve a descrição de processos de tomada de decisão complexos, ele está descrito de forma a facilitar sua compreensão (i.e., tabelas de decisão, árvores de decisão, etc.)?	X		X		X		X		X		X		X	
10. O requisito para executar as atualizações do software foi especificado?	X		X		X		X		X		X		X	
11. O requisito contêm algum nível desnecessário de detalhe do projeto?		X		X		X		X		X		X		X
12. É possível desenvolver um completo conjunto de casos de teste baseado apenas na informação contida na especificação deste requisito?	X		X		X		X		X		X		X	
13. O documento contém realmente toda a informação prometida em sua introdução?	X		X		X		X		X		X		X	