Proposta de um método para auditoria de projetos de desenvolvimento de software iterativo e incremental

Francisco Xavier Freire Neto[1]; Aristides Novelli Filho[2]
Centro Estadual de Educação Tecnológica Paula Souza (CEETEPS) – São Paulo – Brasil
(1)franciscoxfn@terra.com.br; (2)aristidesnovelli@uol.com.br

Resumo – A necessidade de se evitarem falhas nos projetos de software exige que profissionais de desenvolvimento aperfeiçoem seus processos e que os gerenciadores mantenham mais controles sobre os projetos. Esse artigo apresenta a proposta de um conjunto de atividades, definidas com base nos principais padrões de referência, que podem ser agregadas aos projetos de software e fazer parte de um método de auditoria destes projetos, auxiliando na condição de que o software seja entregue dentro do prazo e do orçamento, com todas as características e funções previstas.

Palavras-chave: desenvolvimento de software; projeto de software; auditoria de sistemas.

Abstract – In order to avoid failures in software projects, professionals must optimize their development processes, and managers must to keep more control over the projects. This article presents the proposal of a set of activities defined based on the main standards that can be aggregated and be part of an audit methodology software project to software design, helping to condition that the software be met on time and on budget with all the features and functions provided.

Keywords: software development; software design; systems audit.

1. Introdução

O desenvolvimento de software é uma atividade complexa, uma vez que softwares devem atender as necessidades dos usuários e organizações, para que atinjam suas metas. Segundo Sabbag [1], os processos foram informatizados nas organizações a tal ponto que projetos de infraestrutura, desenvolvimento de sistemas e melhoria de operações tornaram-se urgentes e cruciais.

Com base no relatório divulgado pelo Standish Group International [3], apenas 32% dos projetos de software podem ser considerados de sucesso ao final de sua execução e os demais 68% são encerrados antes do planejado ou terminam com resultados considerados insatisfatórios. As principais causas do insucesso são atribuídas à ausência ou à inadequação de métodos, técnicas ou práticas de gerenciamento de projetos. Embora existam boas metodologias de desenvolvimento de software e de gerenciamento de projetos, há espaço para pesquisar a viabilidade da inclusão de atividades de auditoria nos projetos de software como apoio para reduzir as falhas nesse tipo de projeto.

É proposto nesse trabalho um conjunto inicial de atividades para um método de auditoria ao processo de desenvolvimento de software iterativo e

incremental, definidas com base na literatura especializada e nos padrões de referência. Essas atividades foram validadas por meio de uma pesquisa de campo realizada com especialistas em auditoria de sistemas e no processo de desenvolvimento de software.

Esse trabalho relaciona as principais literaturas, padrões e normas utilizados para estruturar as atividades do método de auditoria proposto; apresenta as atividades propostas para o método; analisa os resultados obtidos com a pesquisa de campo realizada para avaliar o método; e conclui apresentando o principal resultado obtido com a pesquisa.

2. Padrões e Normas de Desenvolvimento de Software

O conjunto de atividades do método de auditoria de desenvolvimento de software proposto foi baseado nos autores Pressman [4] e Sommerville [10] e nos padrões de referência: Cobit 4.1 [6], que define o que deve ser controlado nos processos de TI de uma empresa; CMMI for Development [7], que é um modelo de referência que contém práticas necessárias à maturidade das organizações que desenvolvem softwares ou sistemas; NBR ISO/IEC 12207 [8], que define o ciclo de vida do processo de software; e NBR ISO/IEC 27002 [9], que é um código de prática para a gestão da segurança da informação.

3. Atividades do Método de Auditoria para Desenvolvimento de Software

A auditoria de sistemas de informação pode ser considerada como um meio de apoiar as atividades de desenvolvimento de softwares, uma vez que pode abranger desde o exame de dados registrados em sistemas informatizados, até a avaliação do próprio sistema de informação – aplicativos, sistemas operacionais etc. A partir do estudo das referências citadas na seção anterior, foi proposto o conjunto de atividades do método de auditoria ao desenvolvimento de software iterativo e incremental (Tabela 1).

Tabela 1 - Atividades do método de auditoria avaliadas pelos especialistas

Macro-Atividade	Objetivo de Controle	Grupo de Atividades
Concepção	Garantir que sejam estabelecidos o escopo e viabilidade econômica do projeto	Estabelecer o escopo e os limites do projeto
		Realizar o planejamento inicial
Elaboração	Garantir que sejam definidas a funcionalidade do software, seus requisitos e restrições sobre sua operação	Analisar e validar Requisitos
		Definir a especificação formal do software
		Definir especificação para sistemas críticos
		Definir o planejamento do projeto de desenvolvimento de software
		Definir o projeto de arquitetura do software
		Definir o projeto de interface com o usuário
		Definir os requisitos do software
		Definir políticas, processos e procedimentos para as etapas de desenvolvimento do software
		Educar e treinar os usuários envolvidos no desenvolvimento do software
		Gerenciar a qualidade do software
		Gerenciar o projeto de desenvolvimento do software
Construção	Garantir a entrega de um sistema de software em funcionamento, de acordo com sua especificação, e da	Adquirir e Manter a Infraestrutura de Tecnologia
		Atualizar a documentação do usuário
		Definir a modelagem de análise do software
		Definir controles a serem implementados no software

Macro-Atividade	Objetivo de Controle	Grupo de Atividades
	documentação associada pronta para ser liberada para os usuários.	Definir estratégias de testes de software
		Definir o modelo de projeto de software
		Definir o planejamento para a evolução do software
		Definir o planejamento para validação do software
		Definir o planejamento para verificação do software
		Definir o projeto de testes de software
		Definir técnicas para o desenvolvimento de sistemas críticos
		Definir uma estratégia de integração
		Definir uma estratégia de testes adicionais
		Definir uma estratégia de validação
		Desenvolver o plano de testes do software
		Desenvolver o software conforme o projeto
		Gerenciar mudanças no software
		Realizar auditoria técnica do software
		Realizar revisão por pares nos produtos do projeto
Transição	Garantir que seja entregue um sistema de software documentado, funcionando corretamente em seu ambiente operacional e que seja validado pelo cliente para comprovar que atende aos seus requisitos	Gerenciar a implantação do software no ambiente de produção
		Gerenciar configurações do software
		Realizar a validação do software
		Realizar a verificação do software

Fonte: os autores

A partir destas atividades, foi desenvolvida e aplicada uma pesquisa de campo com especialistas em auditoria de sistemas e em desenvolvimento de software, que avaliaram o grau de importância das atividades desse método de auditoria para o processo de desenvolvimento de software, a partir da escala abaixo:

- 0: atividade não é importante;
- 1: atividade pouco importante;
- 2: atividade com média importância;
- 3: atividade importante; e
- 4: atividade muito importante.

Esse questionário foi aplicado a 50 especialistas, localizados nos estados São Paulo, Paraná e Santa Catarina. Foram descartados apenas três desses questionários por estarem preenchidos de forma incompleta. A seguir, serão apresentados os principais resultados desta pesquisa de campo.

4. Avaliação dos Resultados

A Figura 1.1 mostra, em ordem decrescente do grau de importância, as seis atividades melhor avaliadas do método de auditoria proposto e a Figura 1.2 apresenta as três atividades que obtiveram os menores valores, na avaliação de um método de auditoria no processo de desenvolvimento de software, segundo os especialistas desta pesquisa.

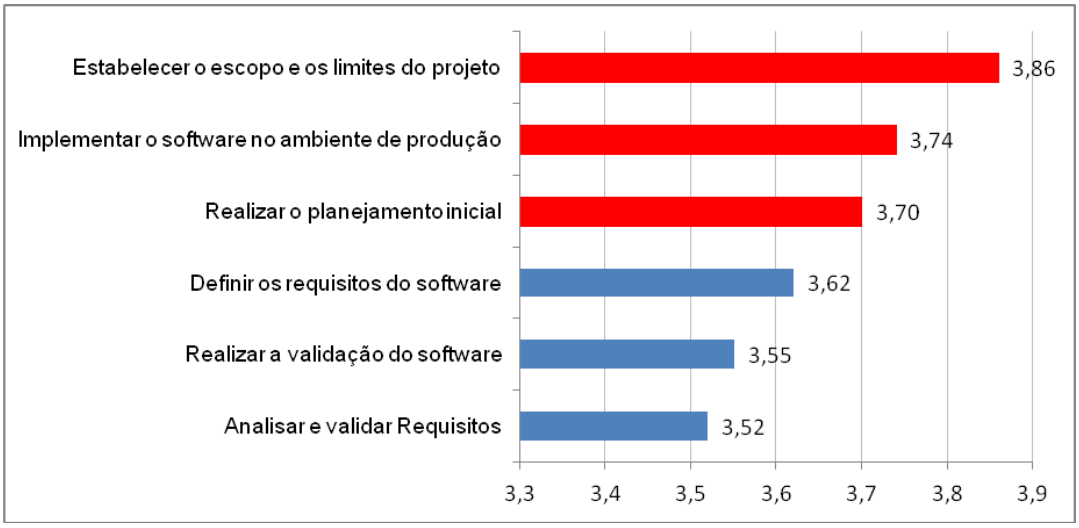

Figura 1.1 – Atividades melhor avaliadas (Fonte: os autores)

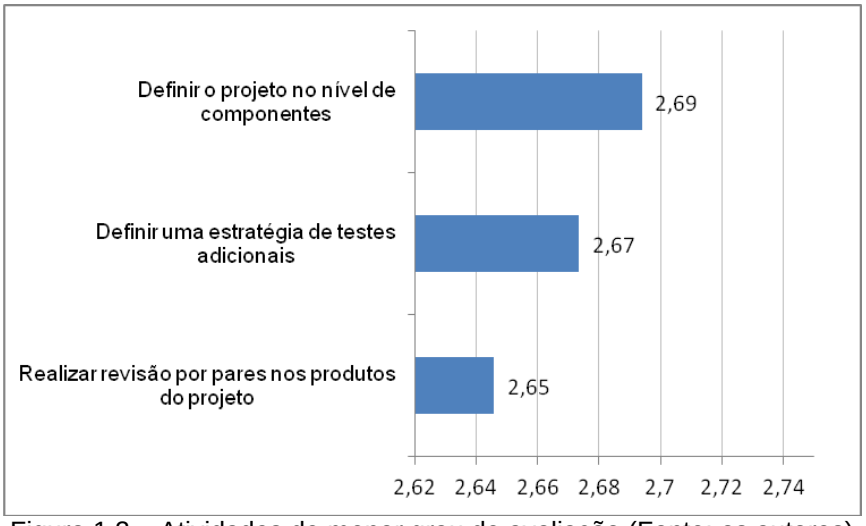

Figura 1.2 – Atividades de menor grau de avaliação (Fonte: os autores)

Entre as principais conclusões da pesquisa podem-se citar:

. A atividade “Estabelecer o escopo e os limites do projeto” foi a melhor avaliada com o valor de 3,86, o que evidencia a relevância de se definir o escopo do software e os limites do projeto, ou seja, as funções e características que deverão ser entregues aos usuários finais, os dados de entrada e saída, as informações que serão apresentadas aos usuários como consequência do uso do software e o desempenho, as restrições, as interfaces e a confiabilidade que limitarão o sistema.

. A segunda atividade melhor avaliada foi “Implementar o software no ambiente de produção”, com o valor de 3,74. Isso mostra que, no contexto de auditoria, devem ser definidas atividades para verificar se o software foi disponibilizado no ambiente de produção de acordo com os requisitos do negócio, no prazo desejado e com um custo razoável.

. A atividade “Realizar o planejamento inicial” foi a terceira melhor avaliada com o valor de 3,70, isto reforça a importância de um planejamento inicial abrangente o suficiente para traduzir os requisitos funcionais de negócio e de controle em um projeto eficiente e eficaz de software.

. Atividades como “Realizar revisão por pares nos produtos do projeto”,

"Definir uma estratégia de testes adicionais" e "Definir o projeto no nível de componentes" foram consideradas de menor importância, obtendo o valor de 2,65, 2,67 e 2,69, respectivamente. Embora tenham sido classificadas como as de menor importância, essas pontuações medianas indicam que os especialistas também as consideram relevantes no método de auditoria proposto.

5. Conclusão

Esse trabalho apresentou um conjunto inicial de atividades que compõem o método de auditoria em projetos de desenvolvimento de software iterativo e incremental, proposto a partir de padrões e normas modelos de grande relevância. Uma pesquisa de campo foi realizada com um número significativo de especialistas em três estados brasileiros para validar estas atividades, e o resultado dessa pesquisa demonstra que as atividades foram bem avaliadas pelos especialistas, o que as qualifica a fazer parte do método de auditoria proposto. Mesmo havendo atividades consideradas menos importantes, todas as atividades definidas para o método foram consideradas significativas pelos especialistas, o que indica que são pertinentes aos projetos de software iterativo e incremental e devem fazer parte do método de auditoria proposto.

6. Referências

[1] Sabbag, P. Y. (2009), *Gerenciamento de Projetos e Empreendedorismo*, São Paulo: Saraiva.

[2] ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE – ABES (2009), *Mercado Brasileiro de Software: Panorama e Tendências - 2009*, São Paulo. Disponível em: http://www.abes.org.br/UserFiles/Image/PDFs/ Mercado_BR2009.pdf. Acesso em: 22 Mai 2010.

[3] STANDISH GROUP INTERNATIONAL (2009), *New Standish Group report shows more project failing and less successful projects*, Massachusetts. Disponível em: http://www1.standishgroup.com/newsroom/ chaos_2009.php. Acesso em: 15 Mai 2010.

[4] Pressman, R.S. (2011), *Engenharia de Software*, São Paulo: McGraw-Hill.

[5] Schmidt, P. et al. (2006), *Fundamentos de auditoria de sistemas*, São Paulo: Atlas.

[6] ITGI - IT Governance Institute. Cobit 4.1. Unites States of America: ITGI, 2007. Disponível em: http://www.isaca.org/Knowledge-Center/cobit/ Documents/cobit41-portuguese.pdf. Acesso em: 22 Mai 2011.

[7] SOFTWARE ENGINEERING INSTITUTE. CMMI for Development, Version 1.3. Disponível em: http://www.sei.cmu.edu/library/abstracts/reports/ 10tr033.cfm. Acesso em: 22 Mai 2011.

[8] ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 12207 – Tecnologia de informação - Processos de ciclo de vida de software. Rio de Janeiro, 2008.

[9] ABNT - ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR ISO/IEC 27002:2005 – Tecnologia da Informação – Técnicas de Segurança – Código de Prática para a Gestão de Segurança da Informação." Rio de Janeiro, 30 de Setembro de 2005.

[10] SOMMERVILLE, I. Engenharia de Software. Vol. 8. São Paulo: Pearson Addison Wesley, 2007.