SAST 2007
I Brazilian Workshop on Systematic and Automated Software Testing

Estratégia de Teste de Software no Desenvolvimento Incremental de um Sistema de Informação

Arilo C. Dias Neto[1], Rodrigo O. Spínola[1], Andrew Bott[2], Guilherme H. Travassos[1]

[1] Programa de Engenharia de Sistemas e Computação – COPPE/UFRJ
Caixa Postal 15.064 – 91.501-970 – Rio de Janeiro – RJ – Brasil.

[2] Fundação Coordenação de Projetos, Pesquisas e Estudos Tecnológicos – COPPETEC

{acdn, ros, ght}@cos.ufrj.br, andrew@coppetec.coppe.ufrj.br

Abstract. *Software testing is the last resource to evaluate a software product before deploying it for the final user. However, in an industrial environment, testing is often conducted in a non-systematic way and thus we do not have an efficient quality assurance regarding the software under test. In this paper we report the experience of implanting a testing strategy in a real project following an incremental life cycle. Among the obtained results we highlight the increase in testing quality without increasing the associated effort and cost, based on the use of three testing levels with different focus and complexity level.*

Resumo. *Teste de software é o último recurso para avaliação do produto antes da entrega ao usuário final. Entretanto, na indústria a aplicação estratégias de teste de software ocorre de forma não sistemática, de forma que não há um controle mais preciso sobre a qualidade do produto avaliado. Neste artigo relatamos a experiência da aplicação de uma estratégia de teste em um projeto real que segue um ciclo de vida incremental. Entre os resultados obtidos destacamos um aumento da qualidade dos testes sem aumentar o esforço e custo desta atividade a partir da implantação de três níveis de teste com diferentes propósitos e diferentes níveis de formalismo.*

1. Introdução

Teste de Software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado (Whittaker 2000).

No processo de desenvolvimento de software, a maioria dos defeitos são humanos e, apesar do uso dos melhores métodos, ferramentas ou profissionais, permanecem presentes nos produtos (Howden 1987), o que torna os testes fundamentais durante o desenvolvimento de um software, pois corresponde ao último recurso para avaliação do produto antes da sua entrega ao usuário final (Pressman 2005).

Teste de Software é considerada uma das atividades mais custosas do processo de desenvolvimento, e desta forma, necessita de um bom planejamento e controle a fim de evitar perdas de recursos e atrasos no cronograma (Juristo *et al.* 2004). Diversas técnicas de teste estão disponíveis na literatura técnica. Escolher qual delas utilizar é uma tarefa complexa, pois vários fatores podem influenciar nessa decisão, tais como categoria de software, esforço e custo associado à sua utilização, e pouco conhecimento científico sobre sua efetividade. Em (Juristo *et al*. 2004) é feita uma análise sobre os estudos experimentais conduzidos com técnicas de teste nos últimos 25 anos (de 1979 a 2004), e revela o pouco conhecimento científico disponível a respeito das técnicas.

Resultados de um survey (Dias Neto *et al*. 2006) conduzido em um cenário de desenvolvimento de software brasileiro ajudam a mapear essa carência observada nas organizações de software a respeito das práticas de teste aplicadas na indústria. Boa parte das práticas de teste questionadas não é aplicada nessas organizações, mas são consideradas importantes pelos profissionais entrevistados. Apesar da limitação dos resultados no contexto do estudo, eles representam um indício sobre o cenário global.

Neste artigo relatamos a experiência da implantação de uma estratégia de teste composta por três níveis em um projeto de software. Este projeto visa o desenvolvimento de um novo sistema de informação baseado na Web para gerenciamento das atividades da Fundação COPPETEC. Trata-se de um projeto de grande porte que envolve diferentes setores da organização, como: recursos humanos, financeiro, contabilidade e protocolo. Ele foi dividido em 11 módulos e está sendo desenvolvido seguindo um ciclo de vida incremental. Entre os fatores que justificaram esta decisão, podemos citar: o cliente está interessado em entregas parciais do produto e na substituição gradual do sistema de informação atual pelo novo. Os níveis de teste que compõem esta estratégia são: teste *estrutural*, *funcional* e *beta*.

A estrutura deste artigo é a seguinte: na seção 2 o projeto e a estratégia e fases de teste aplicadas no projeto são descritos. Na seção 3, a experiência de aplicar diferentes níveis de teste é relatada. A seção 4 apresenta alguns dados quantitativos indicando o ganho obtido com a aplicação de tal estratégia de teste. Por fim, na seção 5, as considerações finais são descritas.

2. Caracterização do Processo de Teste de Software

A partir do ciclo de vida a ser utilizado, definiu-se o processo de desenvolvimento do software fortemente fundamentado em atividades de Verificação e Validação (V&V). As atividades de V&V incluídas no processo são: inspeções (Kalinoswki *et al*. 2007), testes estruturais, testes funcionais e testes beta (de aceitação) (Figura 1). Um gerente de V&V foi definido para viabilizar o planejamento e acompanhamento destas atividades.

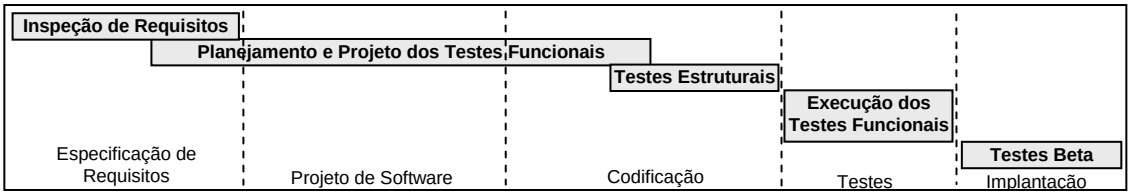

Figura 1. Temporização das atividades de V&V no Processo de Desenvolvimento

O foco deste artigo são as atividades de teste. Nesse contexto, diferentes estratégias foram adotadas para atender os diferentes níveis de teste, que são:

- **Teste Estrutural.** Realizado por qualquer membro do projeto que não exerce o papel de programador. Consiste na execução manual (sem apoio ferramental) e de forma *ad-hoc* (sem uso de qualquer técnica) dos casos de uso do sistema utilizando o documento de requisitos como base para garantir que todos os fluxos de atividades foram implementados e que não existe nenhuma falha na navegação durante a execução dos casos de uso (exemplo: links "quebrados" ou ausências de campos). Essa avaliação é apoiada por um formulário pré-definido que contém questionamentos sobre possíveis falhas que podem ocorrer. Observa-se que o termo ESTRUTURAL está sendo utilizado pois os casos de uso são vistos como grafos, e a preocupação não é com a cobertura dos testes, mas sim em seguir os fluxos e revelar falhas associadas à ausência de passos no fluxo de atividades ou processamento incorreto de determinada operação. Optou-se por essa estratégia para que não houvesse qualquer preocupação com planejamento que pudesse resultar esforço adicional ao processo de testes.
- **Teste Funcional.** Segue uma abordagem de teste mais formal, descrita em Dias Neto e Travassos (2006a), composta por um processo de teste, um conjunto de roteiros de documentos pré-definidos de acordo com o padrão IEEE 829 (1998) e uma infra-estrutura computacional, Maraká (Dias Neto e Travassos, 2006b) para apoiar o planejamento, gerenciamento, controle dos testes e a construção automática dos artefatos do processo. O processo de testes utilizado nessa estratégia é composto por 6 macro-atividades (*Planejar Testes*, *Projetar Testes*, *Especificar*

Casos de Teste, *Definir Procedimentos de Teste*, *Executar Testes* e *Analisar os Resultados dos Testes*) associadas a 3 diferentes papéis (*Gerente de Teste*, *Projetista* e *Testador*). O foco destes testes é a avaliação FUNCIONAL dos casos de uso, fluxos principais e alternativos, suas exceções, regras de negócio e restrições. Inicialmente, o Gerente de Teste participa da inspeção do documento de requisitos a fim de garantir a qualidade deste artefato para planejamento dos testes. Após isso, casos e procedimentos de teste são projetados a partir dos casos de uso, e em seguida ocorre a execução dos testes, que consiste na realização passo-a-passo dos procedimentos de teste especificados. Caso o produto seja reprovado nos testes e haja a necessidade de re-execução dos testes após as correções, uma nova Rodada de Teste com todos os procedimentos de teste planejados é executada. Por fim, o controle dos testes é apoiado por um conjunto de funcionalidades disponibilizadas em Maraká, como cronograma, tabelas de acompanhamento e gráficos de controle.

- **Teste Beta.** Implantação individual de cada módulo após a conclusão e aprovação dos testes funcionais no seu ambiente operacional. O objetivo é a sua disponibilização a um conjunto reduzido de usuários por um período pré-determinado (definido pelos desenvolvedores e cliente) para que esses possam utilizar/avaliar/testar/sugerir alterações no sistema antes de sua entrega definitiva.

Os dados obtidos ao longo de cada etapa de teste são mantidos em uma base histórica. Apenas os dados dos testes funcionais são coletados automaticamente a partir da infra-estrutura Maraká. No cenário de um processo incremental, manter esses dados se mostra interessante, pois eles podem servir como ponto de partida para a melhoria de processo através de análise causal e resolução de defeitos de software, atividade inserida nos níveis de maturidade "A" do MPS (Softex, 2007) e 5 do CMMI (2006).

A seção seguinte descreve como se deu a implantação da estratégia de teste descrita, enfatizando as fases de teste estrutural e funcional.

3. Aplicando a Estratégia de Teste de Software

No momento da escrita deste artigo, dois módulos encontram-se concluídos e estão em fase de *teste beta* e um terceiro módulo está em desenvolvimento. Os testes realizados foram, em ordem cronológica, referentes ao MSL (Módulo de SoLicitações), que teve dois incrementos de desenvolvimento, e MGU (Módulo de Gestão de Usuários).

A implantação da estratégia de teste relatada na seção anterior ocorreu da seguinte forma: inicialmente optamos apenas pela realização de testes funcionais seguindo a abordagem citada na seção anterior, envolvendo um processo formal para planejamento, projeto, execução e controle dos testes. Essa estratégia foi utilizada durante o desenvolvimento do primeiro incremento do MSL, composto por 15 casos de uso. Ao final da execução dos testes, foram relatados 45 incidentes de teste[1], o que resultou em uma densidade de 3 incidentes/caso de uso. O projeto e execução dos testes foram realizados em 90 e 30 horas, respectivamente, o que resultou em um esforço de 4 horas/caso de uso e uma densidade de 0,33 incidentes/hora de teste.

Com essa primeira experiência, foi observado que a maioria dos incidentes relatados (40%) possuía grau de impacto *baixo* ao sistema, e 38% possuíam grau de impacto *médio*. Esses incidentes estavam principalmente relacionados a problemas simples não observados pela equipe de desenvolvimento como, links "quebrados" ou ausência de validação de campos obrigatórios e que muitas vezes impossibilitaram a realização dos testes funcionais por não atenderem as suas condições necessárias. Dessa

[1] Incidentes de Teste: qualquer evento anormal, ou seja, que não era esperado pelo testador, que ocorra durante a execução dos casos e procedimentos de teste e que requeira análise posterior (IEEE-829 1998).

forma, percebeu-se que o esforço de teste estava sendo empregado para detecção de problemas "triviais" que não necessitavam da aplicação de técnicas aprimoradas de teste.

Sendo assim, optou-se pela aplicação de *testes estruturais* de forma *ad-hoc* antes dos *testes funcionais* para verificar se todos os passos estavam implementados conforme especificado no documento de requisitos do sistema. Essa atividade foi realizada a partir do segundo incremento do MSL e também no MGU, e está prevista para todos os módulos subseqüentes que serão desenvolvidos. Além disso, optou-se pela realização de *testes beta* (de aceitação) com o cliente por um período pré-estabelecido por ambas as partes para que os reais usuários ajudem na sua validação e dessa forma se familiarizem com o novo sistema antes da sua implantação em substituição ao sistema atual.

Apenas o nível de *teste funcional* possui um processo definido e utiliza a infra-estrutura Maraká. Os demais (*teste estrutural* e *teste beta*) são realizados de forma *ad-hoc* seguindo diretrizes pré-estabelecidas.

4. Análise dos Resultados dos Testes

Durante a execução do primeiro incremento foram identificadas oportunidades de melhoria nas atividades de V&V. Implantadas as melhorias, passou-se a acompanhar os resultados obtidos nas atividades de garantia da qualidade. Encontram-se a seguir uma análise a respeito dos testes realizados no MSL e MGU seguindo as métricas apresentadas na Tabela 1.

Tabela 1. Resultados da aplicação da estratégia de teste nos novos módulos

Métricas	Apenas teste funcional	Estratégia combinando teste estrutural e funcional			
		MSL - 2º inc		MGU	
	MSL – 1º inc	Estrutural	Funcional	Estrutural	Funcional
# casos de uso	15	4	4	7	7
# horas para projeto dos testes	90	0[2]	10	0[2]	23
# horas para execução dos testes	30	3	14	10	17
# rodadas de teste	2	1	2	1	3
# incidentes de teste	45	14	17	25	28
densidade de incidente/caso de uso	3	2,5	4,25	2,5	4
densidade de incidente/hora	0,4	4,3	0,70	3,6	0,46

- **Módulo de SoLicitações (MSL) – 2º Incremento**

Apesar deste incremento possuir menos casos de uso (apenas 4), conseguimos observar a vantagem da aplicação dos diferentes níveis de teste já neste módulo.

Os testes estruturais foram realizados em apenas 3 horas e revelaram 14 incidentes de teste, o que resulta em uma densidade de 2,5 incidentes/caso de uso e uma densidade de 4,3 incidentes/hora de teste. Esse resultado permitiu uma redução significativa de problemas que poderiam inviabilizar a execução dos testes funcionais.

Para os testes funcionais, foram necessárias **2,5 horas** para definir e projetar os testes **por caso de uso**. Assim, testes para os fluxos de todos os casos de uso puderam ser projetados em paralelo ao desenvolvimento e executados sem exceder o cronograma do projeto. A execução dos testes funcionais foi realizada em 14 horas divididas em 2 rodadas de teste (com 17 e 0 incidentes relatados). Todos os passos da execução dos testes, resultados e incidentes observados foram relatados em Maraká. Para este módulo, tivemos uma densidade de 4,25 incidentes/caso de uso e 0,70 incidentes/hora de teste. Apesar de ser o primeiro momento de aplicação da nova estratégia de teste, observa-se que os resultados apresentam eficiência melhor que os obtidos no primeiro incremento do MSL. A combinação dos níveis possibilitou o aumento da densidade de incidentes por caso de uso e por hora de teste com uma redução do esforço destinado aos testes.

[2] Relembrando: o esforço de projeto dos testes estruturais é 0 (zero) pois essa atividade é realizada de forma *ad-hoc* a partir do documento de requisitos, sem que haja um projeto prévio à execução.

O importante de se destacar é que apesar das densidades obtidas em teste estrutural serem superiores às obtidas em teste funcional, não podemos excluir a segunda atividade, pois seu foco, como já informado, é provocar diferentes tipos de falhas e, como conseqüência, proporciona uma cobertura maior dos testes. Portanto, essas estratégias não são excludentes e devem ser combinadas.

- **Módulo de Gestão de Usuário (MGU)**

Indícios da vantagem de utilizar a estratégia de teste apresentada neste trabalho também podem ser observados para o MGU. Os testes estruturais foram realizados em 10 horas e revelaram 25 incidentes de teste, o que resulta em uma densidade de 2,5 incidentes/caso de uso e uma densidade de 3,6 incidentes/hora de teste.

Os testes funcionais necessitaram, assim como no MSL, em torno de **3 horas** para definir e projetar os testes **por caso de uso**. A execução ocorreu em 17 horas divididas em 3 rodadas de teste (com 12, 16 e 0 incidentes relatos). Para este módulo, tivemos uma densidade de 4 incidentes/caso de uso e 0,46 incidentes/hora de teste. Este já foi o segundo módulo em que a estratégia de teste relatada foi aplicada. Observa-se que, também para este módulo, os resultados apresentam melhor eficiência que os obtidos no primeiro incremento do MSL. Além disso, percebemos uma redução da densidade de incidentes revelados por caso de uso e por hora nos testes funcionais em comparação ao módulo anterior (MSL – 2º inc). A razão para isto é que houve uma análise da causa dos incidentes relatados no módulo anterior, de forma que houve uma preocupação da equipe em evitar a repetição de incidentes similares no módulo seguinte.

- **Lições Aprendidas**

A estratégia adotada possibilitou uma redução significativa no esforço e custo para realização dos testes, pois um número expressivo de falhas tem sido revelado antecipadamente em uma atividade cujo esforço de realização é bem inferior ao da realização de teste funcional. Dessa forma, os testes funcionais puderam ser direcionados para a verificação de outras características do produto não tratadas em teste estrutural, como verificação de regras de negócio, e tiveram um aumento na densidade de incidentes revelados por caso de uso e por hora de teste. Esta estratégia de teste deve ser mantida para os próximos módulos a serem desenvolvidos para que possamos traçar uma curva da sua efetividade ao longo do tempo, comparando com o primeiro módulo onde os testes não seguiram esta estratégia.

A utilização de uma infra-estrutura computacional viabilizou a utilização de um processo mais formal e complexo de testes para teste funcional. A realização dessa tarefa manualmente seria uma atividade custosa para o projeto, e a infra-estrutura reduziu o esforço e simplificou, principalmente, o planejamento e controle dos testes. Um conjunto abrangente de métricas e gráficos de apoio ao controle dos testes é gerado automaticamente pela infra-estrutura, possibilitando o acompanhamento dos testes, rastreamento das falhas reveladas, observação do grau de impacto dos incidentes relatados e itens de teste que resultaram em mais falhas. Todas essas informações podem auxiliar o responsável pela atividade de teste no processo de tomada de decisão a respeito de quando parar os testes, estimar esforço de correção das falhas, indicar nível de cobertura dos testes realizados, dentre outras.

Além disso, esta estratégia simplificou a tarefa de re-execução dos testes funcionais a qualquer instante. Dessa forma, a cada conclusão de um novo módulo, todos os procedimentos de testes dos módulos já desenvolvidos são re-executados para garantir a integridade do software. O esforço associado a essa tarefa consiste simplesmente na execução dos procedimentos, uma vez que todos já estão projetados. O que está sendo feito neste momento é a implantação de um mecanismo para automação dos testes funcionais, utilizando a ferramenta de *capture-replay* Selenium IDE

(http://www.openqa.org/selenium-ide/). Dessa forma, o esforço para execução dos testes só acontecerá uma vez. As demais poderão ser re-executadas automaticamente.

5. Considerações Finais

A realização dos testes estruturais nos casos de uso foi bastante relevante para reduzir o tempo e custo de execução dos testes, pois possibilitou revelar um conjunto de falhas (estruturais) que puderam ser corrigidas antes da liberação do produto para teste funcional, atividade essa que consome mais tempo e recurso. Se observarmos a Tabela 1, o tempo gasto com a atividade de teste estrutural é bastante inferior ao tempo gasto com planejamento, projeto e execução dos testes funcionais em todos os módulos. Em contrapartida, o número de incidentes relatados nessa atividade é sempre superior ao número encontrado em cada rodada de teste funcional. Isso indica a viabilidade da combinação das estratégias, pois elas focam em diferentes categorias de falhas.

Todos os módulos desenvolvidos atingiram seus critérios de aceitação dentro do prazo estabelecido no cronograma do projeto. Especificamente, o módulo MGU necessitou de 3 rodadas de teste para ser aprovado, mas tudo realizado dentro do prazo.

Acreditamos ainda que o apoio ferramental no nível de *teste funcional* tenha facilitado a implantação da estratégia, uma vez que a infra-estrutura Maraká possibilita o planejamento, gerenciamento e controle dos testes funcionais em cada módulo. O uso deste apoio permitiu, ainda, a coleta automática de métricas associadas à atividade de teste. A análise de tais dados, no contexto do processo de desenvolvimento seguindo um ciclo de vida incremental, nos permitiu observar tendências e identificar oportunidades de melhoria tanto nas atividades de especificação do software como no desenvolvimento dos próximos módulos, evitando a repetição de problemas observados durante os testes.

Agradecimentos

Os autores gostariam de agradecer à equipe do projeto, principalmente aos membros da equipe de teste. Agradecemos ainda à CAPES, CNPq e FAPEAM pelo apoio financeiro.

Referências

Capability Maturity Model Integration (CMMI) Version 1.2 Overview, Software Engineering Institute, Carnegie Mellon University, USA (2006).

Dias Neto, A.C., Travassos, G.H. (2006a), "Maraká: Infra-estrutura Computacional para Apoiar o Planejamento e Controle de Testes de Software". In: V SBQS, Vila Velha, ES.

Dias Neto, A.C., Travassos, G.H. (2006b), "Maraká: Apoio ao Planejamento e Controle de Testes de Software". In: XIII Sessão de Ferramentas (20º SBES), Florianópolis, SC.

Dias Neto, A.C., Natali, A.C., Rocha, A.R., Travassos, G.H., (2006) "Caracterização do Estado da Prática das Atividades de Teste em um Cenário de Desenvolvimento de Software Brasileiro". In: V SBQS,Vila Velha,ES.

Howden, W.E., "Functional program testing and analysis". NY, McGrall-Hill, 1987.

IEEE Standard 829-1998: Standard for Software Test Documentation, IEEE Press, 1998.

Juristo, N., Moreno, A. M., Vegas, S. (2004) "Reviewing 25 years of testing technique experiments". Empirical Software Engineering: An International Journal, 9(1), March.

Kalinowski, M., Spínola, R.O., Dias Neto, A.C., Bott, A., Travassos, G. H. (2007) "Inspeções de Requisitos de Software em Desenvolvimento Incremental: Uma Experiência Prática", In: VI SBQS, Porto de Galinhas – PE, Brasil.

Pressman, R. S., "Software Engineering: A Practitioner's Approach", McGraw-Hill, 6th ed, Nova York, NY, 2005.

Softex (2007), "Guia Geral do MPS.BR – Melhoria de Processo do Software Brasileiro", disponível em http://www.softex.br/mpsbr.

Whittaker, J.A. (2000) "What Is Software Testing? And Why Is It So Hard?", IEEE Software, January/February.