Aplicabilidade da Automação de Testes Funcionais – A Experiência no Instituto Atlântico

Paula M. Donegan, Liane R. P. Bandeira, Ana Cristina Matos, Paula Luciana Cunha, Camila Maia, Carlo Giovano S. Pires
Instituto Atlântico
R. Chico Lemos, 946 – Cep 60822-780 – Fortaleza – CE – Brasil
{donegan,liane,cristina,paula,camila,cgiovano}@atlantico.com.br

Abstract. *In the area of software development, organizations have been aiming to build quality products in order to satisfy clients and to stay competitive in the marketplace. Great efforts have been devoted to improvement of test activities, since these evaluate the quality of the final product. Test automation is a widely used way to improve the test process, although it does not always benefit every project. This article describes experiences in automation of functional systemic tests in an organization evaluated as SW-CMM level 2, and analyses applicability of functional tests automation in software development projects in terms of adopted strategies.*

Resumo. *No âmbito de desenvolvimento de software, as organizações têm buscado qualidade dos produtos que constroem, a fim de satisfazer clientes e se manterem competitivas no mercado. Grandes esforços têm sido empenhados em melhorar as atividades de teste, uma vez que estas avaliam a qualidade do produto final. A automação de testes é uma forma amplamente difundida de melhorar o processo de testes, porém, não favorece qualquer projeto. Este artigo apresenta experiências em automação de testes sistêmicos funcionais em uma organização avaliada SW-CMM nível 2 e uma análise da automação de testes funcionais em projetos de desenvolvimento de software de acordo com estratégias adotadas.*

1. Introdução

O mercado de software tem almejado a excelência em qualidade devido à forte exigência dos clientes e à alta competitividade. As organizações da área têm buscado aprimorar o seu processo de teste uma vez que "o teste enfatiza principalmente a avaliação da qualidade do produto" [RUP, 2002].

Um software é construído a partir de um conceito abstrato (requisitos), para um tangível (o produto). No entanto, o mesmo pode se distanciar dos objetivos definidos inicialmente ao longo de seu processo de desenvolvimento, levando-o a se comportar indevidamente. O foco dos testes é inspecionar aquilo construído para identificar tais anomalias, executando um programa com a intenção de encontrar erros [Myers, 2004].

Testar é uma atividade complexa, pois existem várias combinações de dados de entrada e diversos caminhos de execução, limitações de tempo e recursos, regras de negócio com crescente complexidade, interpretações subjetivas das especificações de

requisitos e de testes, dentre outros fatores [Molinari, 2003]. Enquanto esses aspectos têm contribuído para que essa atividade não seja tão disseminada nas organizações, uma metodologia bem concebida, um planejamento adequado e o uso de ferramentas podem melhorar a eficácia dos testes de software.

A automação de testes, considerada o principal recurso para melhorar a eficiência de um processo de testes [Rios, 2003], consiste em utilizar-se de uma ferramenta para o acompanhamento do processo desde o planejamento, passando pela execução e chegando ao controle dos testes. Entretanto, por ser um projeto custoso, é preciso que sejam avaliadas as reais necessidades de teste de uma organização para verificar o quanto a automação traz benefícios e/ou identificar o nível de aplicação em cada projeto, levando a execução dos testes a ser realizada de forma manual, automatizada ou híbrida. A primeira baseia-se no teste de uma aplicação por uma ou mais pessoas que seguem um procedimento de teste, executando ações e inserindo dados de testes. No segundo caso, a execução é realizada com o apoio de uma ferramenta, onde as ações de teste executadas sobre uma aplicação são transformadas em instruções de máquina. Por último, a execução híbrida é aquela que utiliza testes manuais e automatizados, aproveitando as vantagens de cada um.

O artigo analisa a aplicação de testes funcionais automatizados no contexto de projetos de desenvolvimento de software e não considera atividades de manutenção, estando organizado da seguinte forma: a seção 2 contextualiza o trabalho desenvolvido; em seguida a seção 3 apresenta a estratégia adotada para a automação de testes; na seção 4 é abordada uma análise da aplicabilidade; por fim, são explanados considerações finais e trabalhos futuros almejados.

2. Contextualização

Esse trabalho foi elaborado no Instituto Atlântico, situado em Fortaleza-Ceará, uma organização que desenvolve e difunde tecnologias de alto valor agregado para os seus clientes. Seus valores são baseados em inovação e melhoria, esta última refletida principalmente no aperfeiçoamento da qualidade de seu processo de software, visando aumento de produtividade, cumprimento de prazos estabelecidos, redução de custos e, sobretudo, atendimento satisfatório às necessidades de seus clientes. A organização foi avaliada oficialmente como SW-CMM nível 2 ainda no seu segundo ano de existência, em 2003. Em 2004, realizou-se um trabalho de melhoria do processo de testes em conformidade com o SW-CMM nível 3 [Matos, 2005]. Com o amadurecimento das metodologias de desenvolvimento adotadas pelos projetos, as atividades de teste passaram a ter grande relevância para o Instituto, sendo importante o grupo independente de teste, criado em 2003.

Não obstante, a estratégia de teste de regressão[1] adotada pela maioria dos projetos aumentou o esforço requerido para a execução dos testes, intensificando a necessidade de aquisição de ferramentas. Assim, a instituição adquiriu ferramentas da *IBM Rational Corporation*® [Rational, 2005] (*TestManager*®, para planejamento e execução testes e geração de relatórios; *Rational Robot*®, para gravação e manutenção de scripts de testes) e

[1] Teste de Regressão: Após a correção de um erro, é realizada a repetição de todos testes, para garantir que a mesma não impactou negativamente na aplicação [Rakitin].

utiliza-se de uma ferramenta *free* para registro e acompanhamento de problemas. Além de tais investimentos, cursos foram ministrados a fim de capacitar os colaboradores nas mesmas [Bandeira, 2005].

A automação de testes sistêmicos funcionais foi primeiramente implantada em um projeto piloto, a fim de verificar a sua aplicabilidade, servindo como base para outros projetos da organização, como foi feito por [Dustin, 1999], [Kaner, 2002] e [Mosley, 2003]. Atualmente, testes automatizados são utilizados em projetos da organização, tendo em vista que sua aplicação no projeto piloto foi realizada com sucesso, oferecendo ganhos em termos de esforço de testes. Esses projetos são comumente de médio porte, implementam a metodologia iterativo-incremental e possuem um analista de teste dedicado ao projeto.

3. Estratégia da Automação de Teste

Os principais objetivos de estabelecer uma estratégia são otimizar a automação, controlar seus riscos e aumentar reuso. No início é necessário um planejamento para um projeto específico, feito através de reuniões com equipe de teste, coordenador e demais integrantes do projeto. Várias estratégias são apresentadas, sendo levantados os pontos positivos e negativos de cada uma, para que sejam formadas diretrizes de atuação.

A automação de testes é recomendada quando há um processo de testes bem definido e consolidado, em virtude de sua ineficácia comprovada em processos desorganizados e sem metodologia sólida [Kaner, 2002], [Molinari, 2003]. Assim, a formalização do processo de testes do Instituto Atlântico [Matos, 2005] propiciou a implantação da automação.

Como aconselhado em [Dustin, 2002] e [Kaner, 2002], a automação deve ser tratada como um projeto de desenvolvimento de software, o qual requer disciplina em seus processos e gerenciamento de suas variáveis críticas – custo, tempo, risco e escopo [Pmbok, 2000]. São estruturadas e estabelecidas algumas premissas para o contexto da organização.

3.1. Planejamento da Automação

Antes de iniciar a automação, elaboram-se as especificações de teste com seus respectivos Casos de Teste [Myers, 2004], que correspondem ao projeto dos testes funcionais necessários. As atividades de automação correspondem à implementação dos casos de teste especificados anteriormente.

Finalizando as especificações de testes, realiza-se a estruturação dos casos de teste na ferramenta, agrupando-os por funcionalidades, cenários e dados válidos e inválidos, respectivamente, padronizando nomenclaturas. Em seguida, analisam-se quais devem ser executados automaticamente.

Devem-se desenvolver dados de teste para validar as regras da aplicação, antes da geração dos scripts. A organização dos dados de teste é baseada na técnica *Data-Driven* [Mosley, 2003], na qual os mesmos são separados dos scripts, como forma de diminuir o esforço necessário para manutenção.

Não obstante, gera-se um script para cada caso de teste a ser automatizado. A independência de scripts garante que a falha de um deles não interfira na execução de outros. Além disso, devem ter o ponto de partida em comum e serem autocontidos.

Com o intuito de diminuir o retrabalho, estabiliza-se a nomeação de campos e demais elementos da interface ao longo do desenvolvimento, uma vez que eles posteriormente direcionam o preenchimento dos campos de entrada na execução dos scripts.

3.2. Geração de Scripts

Os scripts desenvolvidos apresentam as seguintes tarefas: navegação da interface, funções específicas de negócio e pontos de verificação.

A geração de scripts pode ser realizada de duas formas: utilizando-se do protótipo da aplicação e deixando a gravação de pontos de verificação para o final da fase de implementação; ou após a execução manual do primeiro ciclo de testes.

Ao longo dessa etapa, premissas vão sendo incorporadas aos scripts, os quais são alterados para atender às práticas citadas na seção 3.1. Em seguida, são executados novamente, a fim de assegurar a corretude dos mesmos. Caso seja detectada alguma falha, são efetuadas as devidas correções até que cumpram adequadamente o propósito do teste.

3.3. Execução de Scripts

Agrupam-se os scripts de teste por caso de uso, após sua geração, visando organizar e agilizar o processo de execução. Esse agrupamento, denominado *suíte*, facilita o controle do acompanhamento dos testes, uma vez que fornece uma melhor visibilidade do *status* de cada caso de uso.

Uma falha detectada durante a execução dos testes é cadastrada na ferramenta de registro de problemas, sendo vinculada ao seu caso de teste, a uma *baseline* de teste e classificada mediante sua severidade. Além disso, os passos utilizados para reprodução da falha são descritos detalhadamente com o intuito de facilitar sua correção pela equipe de desenvolvimento.

3.4. Acompanhamento

Após a execução da *suíte*, a ferramenta fornece o resultado de cada script que a compõe. Esse resultado pode ser salvo para posterior geração de relatórios.

A ferramenta permite gerar vários relatórios, quer sejam gráficos ou textuais, podendo também ser customizados pela versão do *build*, plano de teste e status dos casos de teste (implementados, falhos ou com sucesso). No caso de execução híbrida, pode ser gerado um relatório único, com resultados de ambos os testes.

Caso existam mudanças na aplicação, deve-se realizar manutenção nas *suítes* de testes afetadas, para prosseguirem válidas em testes posteriores.

4. Análise da Aplicabilidade da Automação de Teste

A análise apresentada nessa sessão é baseada na experiência de testes manuais e da automação de testes do Instituto Atlântico, demonstrando os possíveis ganhos da abordagem automatizada sobre a manual, delineando a sua aplicabilidade.

Toma-se como insumo para a comparação realizada entre essas abordagens as categorias de casos de uso [Karner, 1993] adotadas pela instituição. Essa classificação é

realizada conforme a complexidade dos mesmos (simples, intermediário e complexo) para fins de estimativa. Considera-se um caso de uso simples quando são necessárias poucas transações de interação com o usuário e/ou quando se abordam regras de negócio simplórias. Os casos de uso intermediários caracterizam-se por apresentar poucas regras de negócio e a forte interação do usuário com a aplicação, por meio de operações do tipo CRUD (*Create, Read, Update e Delete*). Quanto aos complexos, eles normalmente incorporam regras de negócio específicas do domínio da aplicação. [Bandeira, 2005]

Não obstante, utiliza-se a base histórica da organização, Delphi e PERT [Pmbok, 2000], como premissa para a estimativa de esforço de testes manuais, a qual foi estruturada de dados coletados a partir da experiência adquirida nas atividades desenvolvidas por analistas de testes. Atualmente é usada pelos projetos para avaliar o esforço em suas atividades de testes sistêmicos.

A estimativa obtida para testes automatizados foi alcançada por meio do cálculo da média aritmética dos tempos despendidos, em horas, nas atividades de codificação (geração e manutenção) e execução de scripts de teste por complexidade de caso de uso, considerando uma margem de erro comumente utilizada pela organização de 5%, como pode ser visto na tabela 1.

Tabela 1 – Tempo Médio de Atividades de Testes por Complexidade de Caso de Uso

Complexidade	Teste Manual	Teste Automatizado	
	Execução (h)	Codificação (h)	Execução (h)
Simples	2,0	9,10	0,15
Intermediário	4,0	17,72	0,20
Complexo	6,0	26,67	0,34

Desenvolveu-se uma simulação de testes de uma unidade de caso de uso em cada nível de complexidade, considerando as estimativas da tabela 1. Para isso, utilizou-se como base o conceito de ciclo de teste, sendo fixado na simulação como uma bateria de teste e outra de re-teste.

Vários ciclos de teste são executados ao longo do desenvolvimento de um projeto, acumulando, assim, o tempo despendido em cada teste. No caso da automação, o esforço do primeiro ciclo de um caso de uso é a junção da codificação com a execução de um teste e um re-teste dos seus scripts. Nos ciclos seguintes, são adicionados apenas os esforços referentes à execução de scripts (teste e re-teste), como mostrado na tabela 2.

Tabela 2 – Comparativo de Execução de Testes por Casos de Uso

Ciclo de Teste	Esforço despendido (horas) por Caso de Uso					
	Simples		Intermediário		Complexo	
	Manual	Automatizado	Manual	Automatizado	Manual	Automatizado
1	4,0	9,40	8,0	18,12	12,0	27,36
2	8,0	9,69	16,0	18,52	24,0	28,04
3	12,0	9,99	24,0	18,92	36,0	28,73
4	16,0	10,28	32,0	19,31	48,0	29,41
5	20,0	10,58	40,0	19,71	60,0	30,10

Ao analisar o terceiro ciclo de teste, percebe-se que, em todos os níveis de complexidade, o esforço despendido nos testes manuais ultrapassa os automatizados, chegando a 26,84% no caso de uso intermediário. A tabela 2 exibe até o quinto ciclo, pois

nele os testes automatizados chegam a utilizar metade do esforço que seria necessário nos testes manuais. Esses resultados podem ser claramente visualizados através dos gráficos da figura 1, concluindo que projetos utilizando o modelo cascata, sob o âmbito do esforço despendido, não apresentam benefícios com testes funcionais automatizados.

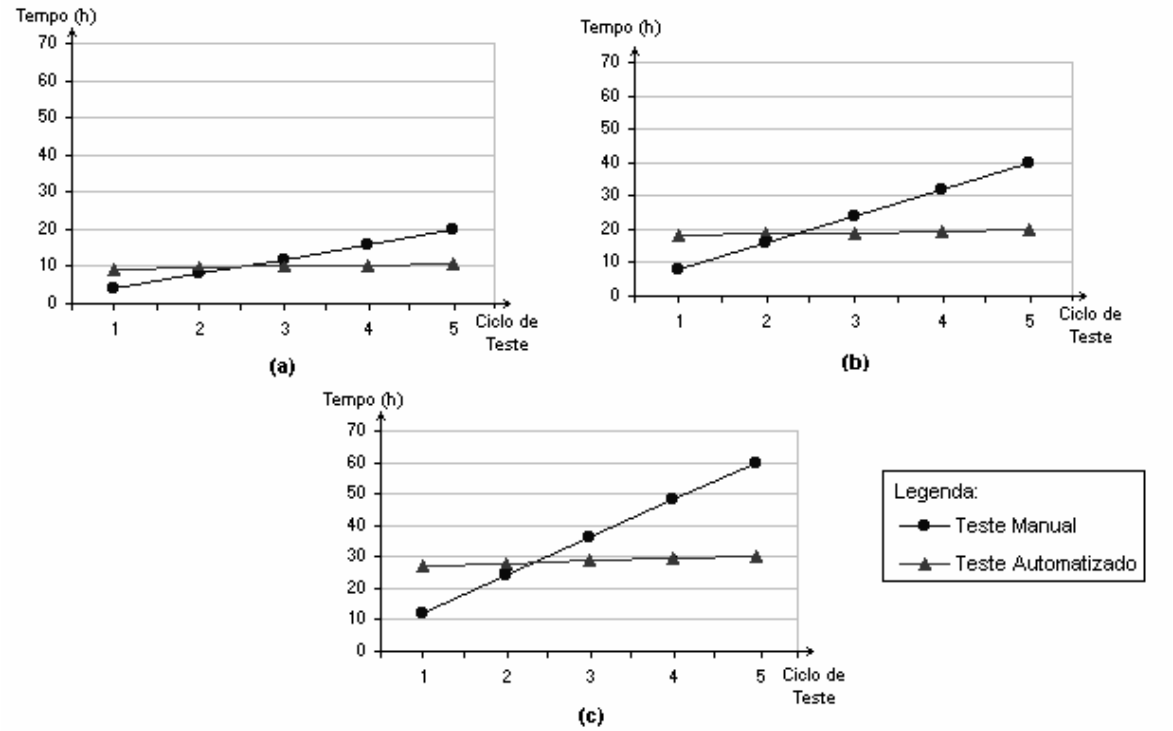

Figura 1 - Tempos de Execução de Testes por Casos de Uso
(a) Simples; (b) Intermediário; (c) Complexo

Outra simulação foi realizada analisando as diferentes formas de execução de testes (manual, híbrido e automatizado) em projetos com ciclo de vida iterativo e incremental. Para desenvolver a simulação, considerou-se a proporcionalidade padrão organizacional de casos de uso de 20%, 50% e 30% para os tipos simples, intermediário e complexo, respectivamente, usada para estimativas na instituição.

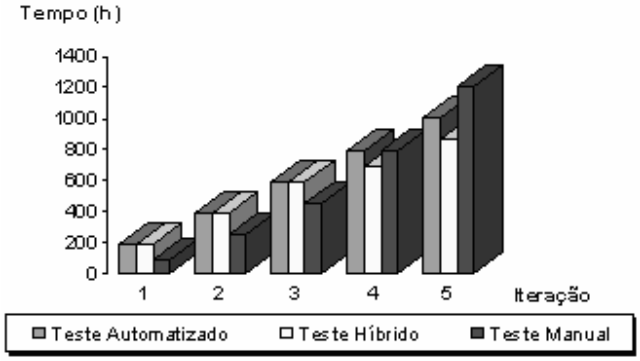

Gráfico 1 – Abordagens de Teste em um Projeto Fictício

Como forma de representar a percentagem acima citada, o gráfico 1 apresenta o resultado do desenvolvimento de dois casos de uso simples, cinco intermediários e três complexos por iteração, em um total de cinqüenta casos de uso para um projeto fictício.

Em virtude do acréscimo de novos casos de uso a cada iteração e o ganho efetivo ser obtido apenas após a terceira iteração (tabela 2), simulou-se a execução híbrida, codificando scripts para os casos de uso apenas nas primeiras iterações e executando testes manuais para aqueles implementados nas últimas. O melhor resultado obtido foi realizando testes híbridos com a seguinte distribuição: três iterações automatizadas e duas manuais; tendo como resultado um esforço de 868,11 horas. No caso de testes totalmente automatizados, seria necessária a alocação de 1.003,65 horas (15,61% superior ao híbrido). O pior resultado foi obtido nos testes exclusivamente manuais, 1.206 horas (38,92% superior ao híbrido).

5. Considerações Finais

A automação de testes sistêmicos funcionais gerou bons resultados para a organização. Verificou-se a garantia da qualidade dos testes, pois eram repetidos com os mesmos dados de teste e procedimentos de execução durante os ciclos de teste, sendo esse aspecto muito relevante para testes de regressão.

Por meio da obtenção do esforço para geração e execução de scripts em projetos, tornou-se conhecido o momento em que os testes automatizados geram ganhos. Essas análises serão consideradas no planejamento das atividades de automação em projetos posteriores.

Os padrões estabelecidos para a geração e execução dos scripts diminuíram o impacto das mudanças, sendo considerados parte das lições aprendidas do trabalho realizado. Assim o processo organizacional de automação incorporou-os.

A comparação das abordagens de testes manuais e automatizados se mostrou limitada em alguns casos. As análises comprovaram a ineficiência da automação de teste em projetos que seguem ciclo de vida cascata e que não apresentem mais de três de ciclos de teste. Não obstante, a execução híbrida de testes se mostrou válida quando não se realiza codificação de scripts nas últimas iterações de um projeto. Obteve-se também um maior ganho em casos de uso de complexidade intermediária, em virtude do alto esforço despendido para realizar as operações de interação com o usuário. Alguns resultados que indicam vantagens na abordagem de testes manuais podem ser revisados se considerarmos ciclos de manutenção do produto. Essa análise será explorada em trabalhos futuros.

Esse artigo apresentou uma abordagem para escolha da forma de teste mais eficiente em termos de esforço despendido e, assim, obter um nível maior de qualidade no produto entregue ao cliente.

6. Referências Bibliográficas

[Bandeira, 2005] BANDEIRA, Liane R. P.. “Qualidade do Produto a partir de Automação de Testes: Um Estudo de Caso”. Monografia, Universidade Estadual do Ceará, 2005.

[Dustin, 1999] DUSTIN, Elfriede. "Lessons in Test Automation," In: Software Testing & Quality Engineering Magazine. – Software Quality Engineering, September/October 1999.

[Dustin, 2002] DUSTIN, Elfriede. "Effective Software Testing: 50 Specific Ways to Improve Your Testing". – Addison-Wesley, 2002.

[Fewster, 1999] FEWSTER, Mark; GRAHAM, Dorothy. "Software Test Automation". – ACM Press, 1999.

[Kaner, 2002] KANER, Cem; BACH, James; PETTICHORD, Bret. "Lessons Learned in Software Testing: A Context-Driven Approach". – John Wiley & Sons, 2002.

[Karner, 1993] Karner, G.. "Metrics for Objectory". Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[Matos, 2005] MATOS; Ana C.; PIRES, Giovano; BANDEIRA, Liane; FERREIRA, Paula; DONEGAN, Paula. "Uma Definição de Processo de Teste para SW-CMM Nível 3". In: Anais SUCESU – Congresso Nacional de Tecnologia de Informação e Comunicação, 2005.

[Molinari, 2003] MOLINARI, Leonardo. "Testes de Softwares – Produzindo Sistemas Melhores e Mais Confiáveis". – Ed. Érica, São Paulo, 2003.

[Mosley, 2003] MOSLEY, Daniel J; POSEY, Bruce A.. "Just Enough Software Test Automation". – Just Enough Series/Yourdon Press, 2003.

[Myers, 2004] MYERS, Glenford J.. "The Art of Software Testing". – John Wiley & Sons, 2nd. Edition, 2004.

[Pmbok, 2000] PMBOK Guide. "A Guide to the Project Management Body of Knowledge". – ANSI/PMI - Project Management Institute, 2000.

[Pressman, 1992] PRESSMAN, R.. "Engenharia de Software". – Ed. Makron Books do Brasil, São Paulo, 1992.

[Rakitin, 2001] RAKITIN, Steven R.. "Software Verification and Validation: A Practitioner's Guide". – Artech Computer Science Library, 2001.

[Rational, 2005] IBM Rational Corporation. Acessado em: 08/03/2005. Disponível em: http://www-306.ibm.com/software/rational/.

[Rios, 2003] RIOS, Emerson; MOREIRA, Trayahú R. F.. "Projeto & Engenharia de Software - Testes de Software". – Ed. Atlas Book, Rio de Janeiro, 2003.

[RUP, 2002] Rational Unified Process®, RUP® – Versão 2002.05.00.