

# Aplicação de um *Checklist* de Pré-Teste

Odair Jacinto da Silva  
Ampla Consultoria em Informação, Faculdades  
Integradas Metropolitanas de Campinas -  
METROCAMP Campinas – SP – Brasil.  
odair@amplaconsultoria.com.br

Adalberto Nobiato Crespo  
Centro de Pesquisas Renato Archer (CenPRA) –  
Campinas – SP – Brasil  
adalberto.crespo@cenpra.gov.br

Mario Jino  
UNICAMP – Campinas – SP – Brasil  
jino@dca.fee.unicamp.br

## Resumo

*O uso de checklists é considerado uma boa prática pela indústria de software. Um experimento foi desenvolvido em uma organização de desenvolvimento de software para avaliar o impacto do uso de checklists para inspecionar e realizar testes básicos de software. Este artigo apresenta a metodologia utilizada e os resultados deste experimento.*

**Palavras Chaves:** teste de software, checklist de inspeção.

## Abstract

*The use of checklists is considered a best practice in software industry. An experiment was developed by a software organization to evaluate the impact of using a checklist to inspect and perform basic testing of software. This article presents the methodology and the results achieved.*

**Keywords:** software testing, inspection checklist.

## 1. Introdução

Nos anos 90 a indústria de software começou a perseguir a qualidade como foi feito por outras indústrias mais antigas, como a automobilística. Diversos *frameworks*, metodologias e normas têm sido produzidos desde então com o objetivo de atender às exigências cada vez maiores do mercado que compra e depende de software para seus negócios, empregos e, por que não, suas vidas.

Todos os modelos, normas e *frameworks* servem como um guia para o estabelecimento de padrões para processos de desenvolvimento de software. De maneira geral deseja-se criar nas empresas uma cultura de utilização de processos e práticas com o objetivo de melhorar a qualidade dos software produzidos.

*Checklists* ou listas de verificação são muito utilizadas em diversos tipos de indústrias. Na indústria metalúrgica, um torneiro mecânico, por exemplo, utiliza diversos *checklists* no seu dia-a-dia. O procedimento de limpeza de um torno no início e término de seu turno é realizado por meio de uma seqüência de passos que devem ser executados, ou seja, um *checklist*.

*Checklists* têm como objetivo garantir que uma tarefa seja executada de uma forma objetiva, eficiente e padronizada. Objetiva porque não se deve desperdiçar tempo com atividades que não contribuem diretamente com o produto ou serviço. Eficiente porque é uma forma de garantir que todos os passos em um procedimento sejam realizados sem que o seu executor tenha que confiar em sua memória. Padronizada porque estabelecemos como padrão as melhores práticas e deseja-se que elas sejam realizadas por todos os envolvidos na criação do produto ou serviço.

Além disso, procedimentos padronizados constituem uma forma de uniformizar a coleta de dados, permitindo que um determinado processo possa ser medido e sua qualidade avaliada [1]. Sem um padrão de comparação é mais difícil obter dados confiáveis e determinar a eficiência de um processo. Vários tipos de *checklist* têm sido propostos para utilização nas diversas fases de desenvolvimento de software [1].

Serão apresentados os resultados de um experimento com a utilização de *checklist* de liberação para teste de software, realizados na Ampla Consultoria em Informação, uma

empresa de desenvolvimento de sistemas de informação para Web.

## 2. A necessidade de se utilizar um *checklist*

Organizações de pequeno porte que desenvolvem projetos de software possuem um quadro de profissionais muito heterogêneo do ponto de vista da experiência. Tais organizações precisam atender as expectativas de seus clientes com relação ao prazo, custo e qualidade. Como resolver o problema da produtividade e qualidade quando se trabalha com profissionais pouco experientes?

Empresas desenvolvem *frameworks* de construção de software como forma de padronizar e acelerar a criação de seus projetos. Ao utilizar um *framework* um desenvolvedor inexperiente toma contato com as boas práticas de desenvolvimento de software da empresa em que atua. Com o contato contínuo ele acaba assimilando tais práticas e as utiliza mesmo em projetos que não são baseados no *framework*.

Esta é uma ótima solução para a construção do software, mas será que o desenvolvedor inexperiente integrou os módulos corretamente? Será que ele tomou o cuidado de verificar se a rotina de alteração do cadastro de clientes realmente altera os dados do cliente selecionado? Será que ele verificou se foram utilizados termos corretos do ponto de vista do idioma utilizado? E a questão das barras de rolagem nas aplicações Web? Será que ele verificou a possibilidade de eliminá-las? Uma coisa simples, mas que aumenta a usabilidade da interface.

Um *checklist* de inspeção para liberação de versões para o teste atende a estas necessidades. Na empresa onde o experimento foi realizado o *checklist* faz parte do processo de desenvolvimento, ou seja, nenhum software é liberado sem que o desenvolvedor tenha antes aplicado o *checklist*.

O *checklist* ajuda a produzir uma unidade com melhor qualidade na medida em que apóia o desenvolvedor na realização de inspeções e testes básicos. Wiegers [2] afirma que *checklists* proporcionam um crivo de avaliação da qualidade pelo qual os artefatos devem passar antes de serem aceitos como concluídos. Defeitos básicos que anteriormente eram detectados pela equipe de teste passaram a ser detectados e eliminados mais cedo no processo. Desta forma a equipe de teste se concentra na verificação da implementação das especificações do projeto de software.

Segundo Fagan [3] o custo de correção de defeitos em programas torna-se maior quanto mais tarde eles forem encontrados, portanto torna-se necessário fazer de tudo para encontrá-los o quanto antes no processo. Para evitar este problema o uso de práticas de inspeções deve ser adotado. Fagan [3] ainda ressalta que inspeções são métodos formais, eficientes e econômicos de encontrar defeitos em projeto e código.

O *checklist* é um artefato utilizado pelo processo. Ele pode atender a práticas genéricas do *Capability Maturity Model Integration for Development* 1.2 [4], *Collect Improvement Information*, na medida em que os dados por ele coletados podem ser utilizados para avaliação de um produto gerado pelo processo.

## 3. O *checklist*

Uma versão do *checklist* foi desenvolvida e inserida no processo de desenvolvimento de software da Ampla em 1999. O novo artefato foi inserido com o objetivo de melhorar o produto liberado para teste. Conforme Hebert [5], isso contribuiu para o aumento da cooperação entre desenvolvedores e testadores. A Figura 1 exhibe os processos da fase de construção de software da empresa.



Figura 1 – Processos da fase de construção

Os processos de codificação e pré-teste (inspeção e testes básicos) são realizados pelo desenvolvedor, ou seja, logo após a codificação de uma unidade sob sua responsabilidade, ele deve realizar o pré-teste da unidade antes de liberá-la para a equipe de teste, que executa os casos de testes criados de acordo com as especificações do projeto do software. Na empresa, o processo de teste está baseado na norma IEEE Std. 829-1998 [6], [7].

Entre 1999 e 2001 a maioria dos projetos desenvolvidos foram sistemas de informação baseados em interface não-Web e o *checklist* espelhava esta característica. A partir de 2002 a empresa passou a desenvolver projetos apenas para a plataforma Web. Assim, o *checklist* precisou refletir esta nova realidade, isto é, percebe-se que existe uma relação entre a estrutura do *checklist* e o tipo de software que será inspecionado.

O *checklist* deve ser construído de forma a auxiliar na inspeção dos elementos tecnológicos da aplicação sob inspeção ou teste. Se, por exemplo, a usabilidade é fator fundamental nas aplicações desenvolvidas pela empresa o *checklist* deve conter uma seção específica para tratar de questões relacionadas à usabilidade, tais como: idioma, utilização de termos técnicos, barra de rolagem, mensagens de exceção etc. A Figura 2 mostra uma visão

parcial da seção de inspeção de usabilidade do *checklist* da Ampla Consultoria em Informação. Outras características de software consideradas importantes para o aumento da qualidade tais como segurança, padronização de termos, utilização correta do idioma, operações de banco de dados etc. também devem ser consideradas.

### **Inspeção de usabilidade**

Verifique a grafia das palavras utilizadas na interface.

Estão grafadas corretamente segundo o idioma utilizado?  
 Sim       Não       Não se aplica

Geram familiaridade para o usuário? Não são técnicas, jargões de informática, são claras e completas (cuidado com uso de siglas).  
 Sim       Não       Não se aplica

As mensagens de erro ou exceção estão grafadas corretamente?  
 Sim       Não       Não se aplica

As mensagens de erro ou exceção estão sintaticamente corretas?  
 Sim       Não       Não se aplica

**Figura 2 – Visão parcial do *checklist* utilizado**

### **Teste de entrada de dados**

O teste executou todas as validações de entrada de dados?  
 Sim       Não       Não se aplica

As mensagens relacionadas a cada validação de entrada de dados são claras?  
 Sim       Não       Não se aplica

O teste foi realizado em campos do tipo texto? Informe caracteres especiais (aspas, arroba etc.).  
 Sim       Não       Não se aplica

O teste foi realizado em campos numéricos? Informe letras ou símbolos.  
 Sim       Não       Não se aplica

**Figura 3 - Visão parcial do *checklist* utilizado**

É importante também que o *checklist* inclua uma sessão de preparação do ambiente para sua execução. Isso aproximará o teste realizado pelo desenvolvedor daquele que será realizado pela equipe de teste. Por exemplo, lembrá-lo de testar uma versão compilada da aplicação, fora do ambiente de desenvolvimento. Lembrá-lo

também de executar o teste em mais de um perfil de usuário.

A Figura 3 mostra uma visão parcial da seção do *checklist* que orienta o desenvolvedor a realizar testes básicos de entrada de dados.

## 4. O experimento

Durante o período de treinamento na empresa, todos os novos contratados devem construir um pequeno sistema de informação para a plataforma Web utilizando o *framework* de desenvolvimento de software existente.

Trata-se de um software para Internet, com uma arquitetura segundo o padrão Model-View-Controller (MVC) [8], contendo 150 pontos de função, acesso a uma base de dados relacional padrão SQL e desenvolvido com tecnologia .NET. O tempo de implementação variava entre uma e três semanas, dependendo das habilidades do treinando. Em relação a este aspecto, apenas um entre os oito desenvolvedores envolvidos no experimento possuía alguma experiência como desenvolvedor, mas nunca havia trabalhado com um *framework* de desenvolvimento de software e nem com desenvolvimento baseado em MVC.

Com o objetivo de validar a eficiência de utilização de *checklist* para pré-teste de software um experimento foi conduzido com oito desenvolvedores que passaram pelo treinamento da empresa.

O experimento foi realizado na seguinte seqüência:

1. Cada desenvolvedor foi orientado por um profissional experiente que atuou como coordenador do projeto.
2. Todos os desenvolvedores receberam a mesma especificação de projeto.
3. O prazo de desenvolvimento foi de três semanas.
4. Após o término do desenvolvimento todos eram orientados a realizar inspeções e testes na aplicação desenvolvida. Os defeitos encontrados foram registrados.
5. O coordenador fazia uma cópia do projeto desenvolvido.
6. O desenvolvedor em treinamento recebia instruções sobre o uso do *checklist*.
7. O desenvolvedor em treinamento realizava novas inspeções e testes básicos no software, desta vez utilizando o *checklist* como roteiro. Este processo consumia, em média, dois dias. A quantidade de defeitos encontrados pelo desenvolvedor foi registrada.
8. Utilizando a cópia do projeto feita anteriormente (passo 5), o coordenador

realizava testes baseados em sua experiência, desenvolvida a partir do uso do *checklist*, e registrava os defeitos encontrados.

## 5. Análise dos resultados

A Tabela 1 exibe os resultados do experimento realizado. A coluna “A” mostra a quantidade de defeitos encontrados pelo coordenador do projeto. A coluna “B” exibe a quantidade de defeitos detectados pelo desenvolvedor, sem uso do *checklist*. A coluna “C” exibe a quantidade de defeitos adicionais detectados pelo desenvolvedor com uso do *checklist*. A coluna “D” mostra quanto o desenvolvedor conseguiu melhorar com o uso do *checklist*. A coluna “E” exibe o desempenho do desenvolvedor em relação ao coordenador.

No projeto 1, o coordenador encontrou 48 defeitos. O desenvolvedor encontrou 13, sem a utilização do *checklist* e 33 com o uso do *checklist*. O uso do *checklist* melhorou em 72% seu desempenho no teste e ele conseguiu 69% de eficiência em relação ao coordenador. A Figura 4 mostra a distribuição dos defeitos encontrados nos oito projetos.

Sem o uso do *checklist*, as funcionalidades implementadas seriam liberadas com muitos defeitos básicos que poderiam impedir a execução plena de todos os casos de testes pela equipe de teste.

Consideram-se como defeitos básicos erros ortográficos, problemas de layout e validações de entrada de dados simples. Tais defeitos são recorrentes por parte de desenvolvedores inexperientes e o uso de um *checklist* aumenta a possibilidade de eliminá-los. Esses tipos de defeitos básicos são independentes de especificações de requisitos, isto é, eles devem ser considerados em qualquer tipo de sistemas de informação desenvolvidos; são eles, portanto, que devem ser o foco de um *checklist*.

A Tabela 2 compara a quantidade de defeitos encontrados na camada de interface dos projetos com o restante do software. Setenta e três por cento (73%) dos defeitos detectados pelo coordenador estavam localizados na camada de interface.

Projeto	Coordenador [A]	Desenvolvedor		C/(B+C) [D]	(C/A) [E]
		Sem Checklist [B]	Com Checklist [C]		
1	48	13	33	72%	69%
2	77	21	51	71%	66%
3	75	20	56	74%	75%
4	62	17	43	72%	69%
5	90	21	68	76%	76%
6	76	19	59	76%	78%
7	72	46	56	55%	78%
8	77	20	57	74%	74%

Tabela 1 – Defeitos encontrados em cada projeto

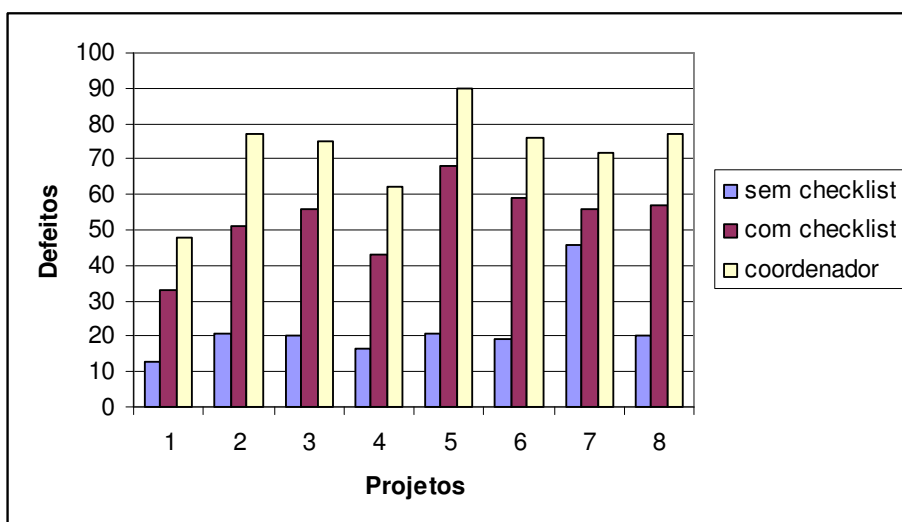


Figura 4 – Distribuição dos defeitos encontrados nos projetos

Tabela 2 – Comparação entre defeitos encontrados na camada de interface e no restante do software

Projeto	Defeitos		Total
	Interface	Restante	
1	40	8	48
2	58	19	77
3	55	20	75
4	48	14	62
5	62	28	90
6	55	21	76
7	52	20	72
8	54	23	77
	424	153	577
	73%	27%	

## 6. Conclusões

Com base nos resultados acima apresentados pode-se concluir que:

- O *checklist* aumenta a capacidade de o desenvolvedor inexperiente detectar defeitos. No experimento realizado, a capacidade de detectar defeitos foi melhorada na ordem de 70% da capacidade do coordenador.
- A relação entre o novo processo, que inclui o *checklist*, e o processo anterior, sem o *checklist*, não foi avaliada no experimento; no entanto, não parece haver um aumento no custo do processo de desenvolvimento como um todo, uma vez que o *checklist* ajuda a melhorar a qualidade do software liberado para a equipe de testes. Os defeitos encontrados na interface impediriam que a

equipe de teste executasse plenamente os casos de testes, o que poderia causar atrasos no projeto, aumentando seu custo..

- A eliminação de grande parte dos defeitos existentes na interface das aplicações pode colaborar com a equipe de teste no sentido de liberar uma versão mais estável de software. Problemas de interface dificultam a execução dos casos de teste.
- Não parece haver um aumento no custo do processo de desenvolvimento uma vez que o *checklist* é um artefato simples de ser utilizado e a quantidade de incidentes de teste diminui, aumentando a produtividade da equipe de teste. No entanto este ponto merece mais investigações.
- Existem evidências de que o *checklist* é um instrumento de aprendizado para aqueles desenvolvedores que ainda não desenvolveram suas habilidades para teste de software. Ao entender como uma aplicação será testada os desenvolvedores passam a fazer uma “programação defensiva”, pois passam a entender como o teste será realizado.

Uma limitação do estudo realizado é que os dados referem-se a um único experimento. Outros experimentos devem ser realizados para confirmar as conclusões apresentadas e para avaliar a efetividade da aplicação do *checklist*; além disso, esses experimentos serviriam para aprimorar as questões do *checklist*.

## 7. Agradecimentos

À equipe de desenvolvimento de software da Ampla Consultoria em Informação pela sua colaboração na realização deste experimento, em especial a Flávio Fontes Nígra e Ana Paula de Souza Dias.

## 8. Referências

- [1] Pressman, R. S. (2005), “Engenharia de Software”, Pearson, 5ª Edição.
- [2] Wiegers, K. E. (2003), “Software Requirements”, Microsoft Press, Second Edition.
- [3] Fagan, M.E., Design and Code inspections to reduce errors in program development, 1976, IBM Systems Journal, Vol. 15, No 3, Page 258-287.
- [4] Software Engineering Institute (2006), “Capability Maturity Model Integration for Development”, Version 1.2.
- [5] Hebert, J. S. (1999), “Aprimorando a Qualidade de Software através do Teste Cooperativo”, X Conferência Internacional de Tecnologia de Software (CITS), Curitiba.
- [6] IEEE Computer Society (1998), “IEEE Std. 829: Standard for Software Test Documentation”.
- [7] Silva, O. J., Crespo, A. N., Salviano, C., Borges, C. A., Jino, M. (2004), “Uma metodologia para teste de software no contexto de melhoria do processo”, III Simpósio Brasileiro de Qualidade de Software (SBQS), Brasília.
- [8] Fowler, M., “GUI Architectures” (2006). Disponível: <http://www.martinfowler.com/eaaDev/uiArc hs.html>. Acesso em 13 de abril de 2007.