RICARDO CARDOSO RIBEIRO

# MODELO DE ROI PARA IMPLANTAÇÃO DE INSPEÇÃO DE SOFTWARE

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para conclusão do curso de MBA em Tecnologia de Software.

Área de Concentração: Tecnologia de Software

Orientador: Professor Doutor Kechi Hirama

São Paulo

2007

# DEDICATÓRIA

Dedico este trabalho aos meus pais Milza e Ademir, aos meus colegas de trabalho e à comunidade de desenvolvimento de software

OK

# AGRADECIMENTOS

Ao meu orientador Prof. Dr. Kechi Hirama por sua paciência, simplicidade e objetividade na condução do desenvolvimento do meu trabalho.

À minha esposa Rita e filhas Fernanda e Clara pela paciência e compreensão pelos momentos de ausência.

Aos meus pais Ademir e Milza que sempre acreditaram e investiram na minha formação acadêmica.

Aos meus colegas de trabalho que sempre acreditaram na minha capacidade técnica/profissional em prol do crescimento da empresa.

# RESUMO

Este trabalho tem por objetivo apresentar um modelo de ROI *(Return On Investment)* para implantação de um processo de Inspeção de Software.

Como Inspeção de Software são apresentadas as técnicas de leitura e a descrição das atividades e papéis envolvidos no processo.

Como ROI são apresentados alguns conceitos, descritas algumas das mais diversas fórmulas e a importância do ROI como ferramenta de análise de investimento no desenvolvimento e aquisição de softwares.

Finalmente, é apresentado um modelo de ROI que permite verificar o retorno de um esforço de implantação de um processo de Inspeção de Software.

Palavras-chave: Inspeção. Inspeção de Software. Retorno sobre investimento.

# ABSTRACT

This work presents a ROI (Return On Investment) model to deploy a Software Inspection process.

As Software Inspection reading techniques, activities description and involved roles into the process are presented.

As ROI some concepts, some equations and the the importance of ROI as investiment analysis tool in software acquisition and development are presented.

Finally, a ROI model is presented that allows verify the return on deployment effort of the Software Inspection process.



Keywords: Inspection. Software Inspection. Return on Investment.

# LISTA DE ILUSTRAÇÕES

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| SEI | *Software Engineering Institute* |
| CASE | *Computer Aided Software Engineering* |
| API | *Application Programming Interface.* |
| TAQtIC | *Tailoring Approach Quality-Driven Inspections.* |
| CEO | *Chief Executive Officer* |
| CIO | *Chief Information Officer* |
| ROI | *Return on Investment* |
| CMM | *Capability Maturity Model* |
| V&V | Verificação e Validação |
| CBR | *Check-List Based Reading* |
| DBR | *Defect-Based Reading* |
| UBR | *Usage-Based Reading* |
| PBR | *Perspective-Based Reading* |
| OORT | *Object-Oriented Reading Technique* |
| CRM | *Customer Relationship Management* |
| UML | *Unified Modeling Language* |
| ROIC | *Return On Invested Capital* |
| NOPAT | *Net Operating Profits After Tax* |
| NPV | *Net Present Value* |
| PV | *Present Value* |

# Sumário

# 1. Introdução

Este capítulo descreve as motivações, o objetivo e a estrutura do trabalho.

## *1.1. Motivação*

Nas últimas décadas, alguns métodos de melhorias de processos de software têm sido amplamente adotados para melhoria da qualidade do software. Por outro lado, qualidade de software está intimamente ligada ao retorno sobre investimento, uma vez que o resultado final é um processo de software com menos re-trabalho e um produto com maior índice de satisfação do cliente. Para entender a qualidade de software e sua importância dentro da Engenharia de Software, basta olhar a história dos últimos 50 anos e perceber que o software não melhorou em termos qualitativos na mesma proporção que a evolução da tecnologia (WHITTAKER, 2002).

A qualidade do software começou a cair na década de 1960, na conhecida "crise do software" e com o advento do computador pessoal no fim da década de 1970, a computação passou a ser uma realidade para todas as pessoas e não somente para matemáticos, pesquisadores de universidades e estrategistas militares. As necessidades das empresas também aumentaram vertiginosamente a demanda de softwares cada vez mais complexos e isso gerou euforia na área de desenvolvimento de software (WHITTAKER, 2002).

Na década de 1980 as ferramentas CASE (*Computer Aided Software Engineering*) e as linguagens de 4ª geração vieram para dar a visão de que os programadores criariam softwares melhores se tivessem ferramentas de software que os ajudassem. Estas ferramentas aliadas aos métodos formais produziriam código de melhor qualidade. No entanto, os métodos formais não foram incorporados na cultura da organização devido ao alto custo, incerteza no retorno sobre investimento, à dificuldade de sua utilização à ausência de ferramentas de apoio (WHITTAKER, 2002).

A solução para o problema de falta de qualidade dos softwares surgiria, teoricamente, na década de 1990 com melhorias no processo de software. No centro desse movimento

estava o CMM (*The Capability Maturity Model* do SEI – *Software Engineering Institute* da Universidade de Carnegie Mellon nos Estados Unidos da América). Acreditava-se que controlar a maneira como o software era construído garantiria um software melhor (WHITTAKER, 2002).

Outro método conhecido como SixSigma, bastante difundido inicialmente na área fabril, veio contribuir para a redução de defeitos de software, mas enfrentava um problema, a impossibilidade de medir a introdução de defeitos no mundo do software (WHITTAKER, 2002).

Além disso, a década de 1990 que também foi marcada por diversas evoluções no campo do software: sistemas operacionais mais sofisticados, novas e poderosas linguagens de programação, novas API (*Application Program Interface*) para comunicação, segurança, computação distribuída e a Internet tornou as atividades dos desenvolvedores mais complexas (WHITTAKER, 2002).

Na década atual, as expectativas apontam para melhorias voltadas à qualidade de software na tentativa de vencer alguns desafios como: os empecilhos que, por exemplo, impediram o sucesso da utilização dos métodos formais na década de 1980 e as melhorias de processo de software na década de 1990 e como medir o retorno sobre investimento na adoção de Inspeção de Software (WHITTAKER, 2002).

Alguns estudos mostram que a inspeção de software detecta a maioria dos defeitos existentes nas fases iniciais do desenvolvimento, na faixa entre 60 e 90%, reduzindo significativamente esforços futuros em testes e depuração de código, por exemplo (CROUST; LEXEN, 1999) (DENGER; SHULL, 2007).

Mesmo com esses benefícios, a Inspeção de Software enfrenta empecilhos na sua utilização. No entanto, muitos esforços têm sido feitos desde que a Inspeção de Software foi originalmente proposta por Michael Fagan (FAGAN, 1976). Existem *frameworks* (estruturas de suportes bem definidas que auxiliam na implantação de um processo ou projeto de software) para configuração de processos de inspeção (DENGER; SHULL, 2007), técnicas de leitura dos artefatos gerados durante o ciclo de vida do desenvolvimento e ferramentas que as apóiam (CIOLKOWSKI, 1999).

É importante unir esforços em processo, técnica e ferramentas incentivando o uso de

Inspeção de Software nas organizações para que haja aumento na produtividade, melhoria da qualidade do produto final, redução do custo e conseqüente retorno sobre o investimento.

No que se refere a processo, são destacados alguns pontos que tem que ser considerados durante a adoção da Inspeção de Software, como: o contexto no qual o processo de desenvolvimento está inserido, o envolvimento dos principais interessados no produto final e, principalmente, foco na qualidade do software. Dessa forma, espera-se reduzir significativamente o impacto da adoção de um processo de Inspeção de Software na organização (DENGER; SHULL, 2007).

Além das visíveis melhorias no processo de software vigente na organização, é possível mostrar à Gerência que existe um ROI e ele pode ser estimado com base em cálculos matemáticos.

## *1.2. Objetivo*

O objetivo deste trabalho é apresentar uma proposta de modelo de ROI baseado no estudo feito pelo O'Neil (2002) para um projeto de implantação de um processo de Inspeção de Software.

## *1.3. Estrutura do trabalho*

Capítulo 1 – Introdução

Neste capítulo são apresentados as motivações, o objetivo do trabalho, e a estrutura do trabalho.

Capítulo 2 – Inspeção de Software: conceitos, técnicas e processo

Neste capítulo são apresentadas as principais características, conceitos e algumas técnicas e o processo de Inspeção de Software.

Capítulo 3 – *(Return On Investment)* ROI

Neste capítulo é apresentado o conceito de ROI como instrumento econômico-financeiro e algumas de suas diversas variantes.

Capítulo 4 – Modelo de ROI para implantação de um processo de Inspeção de Software

Neste capítulo é apresentado o modelo do ROI proposto para implantação de um processo de Inspeção de Software com base na redução de custos associada à detecção e correção dos defeitos nas fases iniciais do ciclo de vida do produto.

Capítulo 5 – Conclusão

Neste capítulo são apresentadas conclusões, as contribuições do trabalho e futuros trabalhos que podem ser desenvolvidos.

Referências

Lista as fontes usadas para a elaboração do trabalho.

## 2. Inspeção de Software: conceitos, técnicas e processo

Desde de 1976, o processo de Inspeção de Software proposto por Fagan tem sido usado por diferentes indústrias e em diferentes artefatos de software, porém mais freqüentemente em código. Dez anos depois, Fagan apresentou novos estudos e experimentos que reforçaram o uso do processo de inspeção e aumentaram sua contribuição para o desenvolvimento de software livre de defeitos, no prazo e a custos mais baixos. Excelentes resultados foram obtidos por pequenas e grandes organizações em todos os aspectos tanto no processo de desenvolvimento quanto no de manutenção. Ele cita que existem evidências de que os desenvolvedores que participam da inspeção de seu próprio produto produzem menos defeitos em trabalhos futuros e que o fato de a inspeção formalizar o processo de desenvolvimento, melhorar a produtividade e a qualidade, facilita também a adoção mais fácil e rapidamente de ferramentas (FAGAN, 1986).

Em 1989, vieram à tona os conceitos e processo de Inspeção de Software e, na época, classificaram a Inspeção de Software como um programa fundamental de verificação e validação em que os passos de verificação devem ser executados ao longo do desenvolvimento do software (ACKERMAN *et al*, 1989).

Em 1999, Croust e Lexen relataram através de experiência em uma grande (IBM Laboratory – Vienna) e uma pequena (Pollex Ltd – Austria) organização de software, que a Inspeção de Software também oferece benefícios financeiros como um alto retorno sobre o investimento, mas que ainda não era amplamente usada devido a problemas inerentes ao próprio processo de inspeção clássico proposto por Fagan tais como: redundância na detecção de defeitos, tempo gasto com o agendamento das reuniões, custos com deslocamento para reuniões de inspeção, tempo não produtivo gasto com reuniões em excesso (CROUST; LEXEN, 1999).

Muitos esforços têm sido feitos para facilitar a implantação do processo de Inspeção de Software e mantê-lo na organização. Recentemente, em 2007, Denger e Shull propuseram uma abordagem prática para implantação do processo de Inspeção de Software através da criação de um *framework* chamado TAQtIC *(Tailoring Approach Quality-Driven Inspections)* (DENGER; SHULL, 2007). A inovação do TAQtIC baseia-se

na integração de boas práticas de inspeção em abordagens simples através de alguns conceitos principais como:

- Consideração de regras do contexto: verificar o processo de desenvolvimento utilizado na organização, disponibilidade de recurso pessoal, qualidade específica do projeto requerida no produto final e experiência da equipe de desenvolvimento.
- Envolvimento explícito dos *stakeholders* (principais interessados no produto de software): envolvimento dos *stakeholders* no produto de trabalho, que está sob revisão, reduz a percepção de que a inspeção é uma atividade adicional não produtiva.
- Foco na qualidade do produto final: o foco por parte dos inspetores na qualidade do produto final, tais como: manutenibilidade, reusabilidade e segurança, minimiza as dificuldades na execução de inspeções e cria uma ligação direta entre o esforço de inspeção investido nos produtos de trabalho intermediários e a qualidade final do software.

## *2.1. Conceitos*

Para garantir que um software tenha alta qualidade, atividades de validação e verificação de software podem ser utilizadas. V&V é o nome dado aos processos de Verificação e Validação que asseguram que o software cumpra com suas especificações e atenda às necessidades dos clientes. As atividades de V&V acontecem durante todo o ciclo de desenvolvimento do software, começando por inspeções nos requisitos, no projeto, no código até o teste do produto (ENDO; PEREIRA, 2006).

Nesse contexto, a Inspeção de Software, que busca encontrar defeitos em artefatos de software em geral utilizando técnicas para fazer essa análise, mostrou ser muito eficiente (ENDO; PEREIRA, 2006).

Um motivo que comprova que as inspeções são mais eficazes que testes, é que muitos defeitos diferentes podem ser detectados em uma única sessão de inspeção, enquanto

que nos testes apenas um defeito por teste é detectado. Outro motivo é que as inspeções reutilizam conhecimento de domínio e linguagem de programação, devido à experiência adquirida pelos responsáveis pela inspeção (ENDO; PEREIRA, 2006).

As inspeções se diferenciam de outros métodos de inspeção por seguir um processo bem definido e testado e são incluídas já nos estágios iniciais no processo de desenvolvimento dos produtos de software.

## 2.2. *Técnicas de Inspeção*

Para inspecionar um artefato, é necessário lê-lo portanto, as técnicas de leitura surgiram como forma de melhorar o desempenho da Inspeção de Software, no que se refere à atividade de detecção de defeitos. O termo leitura foi escolhido de forma a enfatizar as similaridades com o processo mental que as pessoas utilizam quando tentam entender o significado de algum texto (MAFRA; TRAVASSOS,2005).

Uma técnica de leitura pode ser caracterizada como uma série de passos para a análise individual de um artefato de software de forma a permitir a extração do entendimento necessário para a execução de uma determinada tarefa. Observaram-se três importantes critérios nessa definição (MAFRA; TRAVASSOS,2005):

- Uma série de passos: a técnica deve prover ao desenvolvedor uma orientação na condução da leitura. Essa orientação pode variar desde um procedimento passo a passo a um conjunto de questões formuladas com o objetivo de manter o foco da leitura;
- Análise individual: técnicas de leitura devem se preocupar com o processo de compreensão individual. Ainda que alguns métodos de garantia da qualidade, nos quais a técnica esteja inserida, devam requerer o trabalho em equipe (como numa reunião de inspeção), a compreensão de certos aspectos do artefato ainda assim é uma tarefa individual;
- Entendimento necessário para a execução de uma tarefa: a técnica deve apoiar a obtenção de entendimento de todos os aspectos de um artefato.

As técnicas de leitura necessitam ser bem definidas, orientadas a objetivos e dependentes de contexto, ou seja, estarem direcionados dependendo do objetivo do documento que está sendo analisado (MAFRA; TRAVASSOS,2005). Baseados nisso, estabeleceram-se requisitos para uma técnica de leitura:

- Estar associada a um tipo de artefato (ex.: documento de requisitos) e a notação pela qual o artefato é descrito (ex.: língua portuguesa);
- Ser adaptável de acordo com as características intrínsecas da organização e do desenvolvimento;
- Ser detalhada, provendo ao desenvolvedor um processo bem definido. A existência de um processo bem definido permite que a técnica seja executada em outros projetos, atuando como um meio de disseminação de conhecimento;
- Ser avaliada experimentalmente para determinar sua viabilidade e seu grau de efetividade na detecção de defeitos.

As técnicas de leitura são técnicas procedurais, desenvolvidas para executar uma tarefa específica relacionada ao software (detecção de defeitos) usando uma parte específica da documentação de software, sendo esta considerada fundamental para alcançar um alto nível de qualidade de software (PAGLIUSO *et al,* 2002). Dentre essas técnicas destacam-se as mais utilizadas: CBR, DBR, UBR, PBR e OORT.

a) **CBR (*Checklist-Based Reading*)**

É uma técnica da leitura baseada no uso de *checklists* (listas de verificação) com um formulário de questões ou de indicações. Enquanto a equipe lê o documento a ser inspecionado, os integrantes devem responder a questões que constam do *checklist* (PAGLIUSO *et al*, 2002).

Estas questões se referem a diferentes requisitos do documento. A idéia por trás dos *checklists* é a captura de experiências, isto é, conhecimentos sobre problemas e defeitos típicos são transformados em questões.

Porém, os *checklists* apresentam alguns pontos negativos:

- As questões são muito genéricas deixando o documento muito longo.
- Os *checklists* são geralmente longos e contém muitas questões. Com isso, a equipe tende a ignorar algumas questões, para não deixar o processo muito tedioso e longo.
- Os *checklists* impõem a equipe que verifique todas as informações dos documentos inspecionados, o que pode fazer com que os integrantes da equipe se dediquem a detalhes desnecessários do documento.

Todos os integrantes da equipe de inspeção usam a mesma lista de verificação. Em uma equipe de inspeção, todos verificam os mesmos aspectos do documento e usam as mesmas técnicas de leitura.

Sendo assim, a técnica CBR se caracteriza por ser não-sistemática, genérica e idêntica. Não-sistemática para responder as questões, genérica, pois os integrantes da equipe devem verificar todos os aspectos do documento, e idêntica, pois todos os membros da equipe de inspeção verificam os mesmos aspectos (CIOLKOWSKI, 1999).

**b) DBR (*Defect-Based Reading*)**

É uma técnica de leitura cujo ponto de vista é orientado pelos defeitos que podem ser encontrados. Fornece cenários para os inspetores detectarem defeitos. Na versão atual do DBR, há três técnicas de leituras ou cenários, cada um deles foca em uma classe específica de defeito: consistência do tipo de dado, funcionalidade incorreta e funcionalidades inexistentes ou ambíguas. Cada cenário consiste de duas partes: instruções de como e onde olharem os defeitos, e no conjunto de questões a serem respondidas em relação aos defeitos (CIOLKOWSKI, 1999).

Embora as técnicas de DBR existentes sejam designadas a documentos formais, DBR também pode ser usado em documentos informais (CIOLKOWSKI, 1999).

Conseqüentemente, pode ser necessário criar uma lista de classes de defeitos para

documentos informais, e criar técnicas de leitura que foquem uma classe específica de defeito (CIOLKOWSKI, 1999).

### c) UBR (*Usage-Based Reading*)

É uma técnica de leitura que auxilia a detecção de defeitos severos do ponto de vista do usuário. A UBR foca na leitura orientada por um modelo de casos de uso priorizados em ordem de importância para o usuário do sistema, durante a fase de preparação de um processo de Inspeção de Software (MAFRA; TRAVASSOS,2005).

A premissa básica é permitir que as expectativas do usuário, no se refere ao atendimento das funcionalidades, governem a inspeção. Os modelos de caso de uso poderiam ser usados para inspeções em todas as fases de desenvolvimento (requisitos, projeto, código, etc) de um projeto específico. Durante a inspeção, os inspetores lêem o documento executando manualmente os casos de uso e tentando detectar defeitos que são mais importantes de acordo com a prioridade estabelecida e, por conseguinte, para o usuário (MAFRA; TRAVASSOS,2005).

Na técnica UBR, os Casos de Uso da linguagem UML (Unified Modeling Language) são usados como guias para os artefatos inspecionados; o objetivo é melhorar a eficiência e a efetividade ao direcionar os esforços de inspeção para os Casos de Uso mais importantes sob o ponto de vista dos usuários; os cenários são específicos para cada projeto, o que significa que os Casos de Uso poderiam ser utilizados apenas dentro do projeto nos quais foram desenvolvidos. Entretanto, poderiam ser utilizados para inspeções de requisitos, projeto e código e apoiar a especificação de teste (MAFRA; TRAVASSOS,2005).

### d) PBR (Perspective-Based Reading)

É uma técnica de leitura baseada em perspectivas que auxiliam os inspetores a certificarem-se de que e como a informação deverá ser verificada. As perspectivas são definidas, por exemplo, do ponto de vista do usuário, projetista, testador, etc.

(CIOLKOWSKI, 1999).

Portanto, usar PBR significa (CIOLKOWSKI, 1999):

- Selecionar um conjunto de perspectivas para revisar requisitos de documentos;
- Criar ou adaptar procedimentos para cada perspectiva que poderá ser usada para a construção de modelos de requisitos de informações relevantes;
- Ampliar cada procedimento com questões para encontrar defeitos e;
- Aplicar o procedimento para revisar o documento.

Durante a aplicação da técnica PBR, cada participante do grupo de inspeção assume uma perspectiva específica de usuário (CIOLKOWSKI, 1999).

A técnica também verifica, através de cenários, a qualidade das especificações de requisitos ao solicitar que cada inspetor assuma a perspectiva de um usuário específico do documento (projetistas, responsáveis por testes e usuários finais). Cada inspetor PBR recebe um cenário para guiar seu trabalho, uma descrição procedural das atividades e questões e os inspetores são guiados nos documentos a serem inspecionados. O cenário consiste em construir um modelo do documento a ser inspecionado para aumentar o entendimento e responder questões sobre o modelo, focado na resolução de problemas de interesse da organização (CIOLKOWSKI, 1999).

A técnica PBR oferece vários benefícios como focar as responsabilidades de cada participante diminuindo a sobreposição das atividades realizadas, possibilitando que o documento seja completamente verificado, sem haver análises duplicadas de informações e/ou falta de análise, garantindo assim uma varredura total do documento uma vez que cada participante assumiu uma perspectiva diferente durante a inspeção (PAGLIUSO *et al*, 2002).

**e) OORT (Object-Oriented Reading Technique)**

A técnica OORT representa uma família de técnicas de leitura que fornecem um

procedimento para revisões individuais dos diferentes diagramas e documentos de projeto OO (*Object Oriented*) (CONRADI *et al*, 2003).

O processo de leitura que usa a OORT deve ser realizado em duas dimensões: leitura horizontal e vertical (CONRADI *et al*, 2003):

- Na leitura horizontal, diferentes diagramas de projeto são verificados para assegurar que estejam consistentes entre si;
- Na leitura vertical, é necessária a validação entre a especificação dos requisitos e os diagramas de projeto, para assegurar que o projeto esteja correto em relação aos requisitos, e o código fonte em relação ao projeto.

Como exemplo de aplicação da leitura horizontal OORT, cita-se uma situação onde Diagramas de Interação da UML necessitem ser comparados com os Diagramas de Estado da UML, identificando-se para um determinado objeto, os eventos, as restrições ou os dados que poderia mudar a forma como as mensagens são enviadas para ele (CONRADI *et al*, 2003).

Uma vantagem da família de técnicas OORT é que se pode selecionar apenas o subconjunto de técnicas que correspondem aos artefatos que devem ser inspecionados ou que são especialmente importantes em um determinado momento (MAFRA; TRAVASSOS,2005).

## *2.3. Papéis*

Com pequenas variações nas nomenclaturas, a equipe de inspeção normalmente é composta de um grupo pequeno de pessoas, tipicamente de 3 a 7, com interesse no produto (FAGAN, 1976) e (MERTZ, 1993).

Equipes maiores são usadas para documentos de alto nível, ou seja, que possuem menos detalhes, enquanto equipes menores são utilizadas para inspeções de documentos com maiores detalhes técnicos. Membros são adicionados às equipes quando seus pontos de vista são necessários (MERTZ, 1993).

Ter participantes que representem várias áreas de especialidade é importante para o

processo, pois cada um vai analisar o produto com base em sua própria perspectiva e experiência, facilitando a identificação dos defeitos. A criação de sinergia entre os participantes é um indicador de um processo de inspeção saudável (MERTZ, 1993).

Os papéis individuais de cada membro podem ser assim descritos (MERTZ, 1993):

### Moderador

O Moderador é a pessoa chave, pois ele lidera a equipe e é responsável por garantir o sucesso da inspeção. O moderador não precisa ser um técnico especialista nos produtos sendo analisados, mas deve usar sua sensibilidade pessoal e habilidade para obter sinergia entre os participantes, e normalmente seu treinamento é mais importante e extenso que o dos demais.

As responsabilidades do moderador incluem (MERTZ, 1993):

- Selecionar a equipe de inspeção e liderá-la durante o processo;
- Distribuir o material a ser inspecionado;
- Agendar as reuniões;
- Atuar como moderador das reuniões de inspeção;
- Coletar dados;
- Supervisionar o acompanhamento (re-trabalho);
- Emitir o relatório de inspeção.

### Inspetor

Todos os membros do grupo de inspeção são considerados inspetores, além dos seus papéis assinalados para uma reunião específica. Os inspetores são responsáveis por identificar defeitos durante a preparação e durante as reuniões de inspeção. Os principais candidatos para inspetores são os *stakeholders*, na etapa anterior, corrente e posterior do ciclo de vida. Por exemplo, para avaliar um documento de projeto de sistema, bons inspetores estão entre os que escreveram os requisitos e os que irão codificar os

programas.

### Leitor

O Leitor é responsável por guiar a equipe de inspeção no artefato que está sendo inspecionado, lendo ou parafraseando seções do artefato e realçando as partes importantes. O indivíduo que irá utilizar o produto durante a próxima fase do ciclo de vida é um excelente candidato para leitor, pois isto o tornará mais familiar com o produto antes de sua entrega.

### Autor

O Autor é o criador do produto que será inspecionado e tem como principais responsabilidades:

- Garantir que os objetos satisfaçam os critérios mínimos para aceite no processo de inspeção;
- Fazer a visão geral do produto;
- Prover esclarecimentos durante as reuniões;
- Executar as adequações (re-trabalho) solicitadas.

### Documentador

O Documentador é responsável por registrar defeitos, bem como as decisões e recomendações feitas durante a reunião. Os dados devem ser classificados e detalhados, não só para serem usados na avaliação do produto inspecionado, mas também para permitir a análise e melhoria do próprio processo de desenvolvimento.

## 2.4. *Processo*

Um processo de uma inspeção é um conjunto de atividades executadas por determinados papéis e realizado em etapas segundo Fagan (1976) e Mertz (1993). Segundo Mertz (1993), a primeira etapa é o Planejamento, seguido pela Elaboração da Visão Geral, Preparação Individual, Reunião de Inspeção, Re-trabalho e Acompanhamento como mostra a Figura 1.

Com pequenas variações nas nomenclaturas, a equipe de inspeção normalmente é composta de um grupo pequeno de pessoas, tipicamente de 3 a 7, com interesse no produto. (FAGAN, 1976) (MERTZ, 1993).

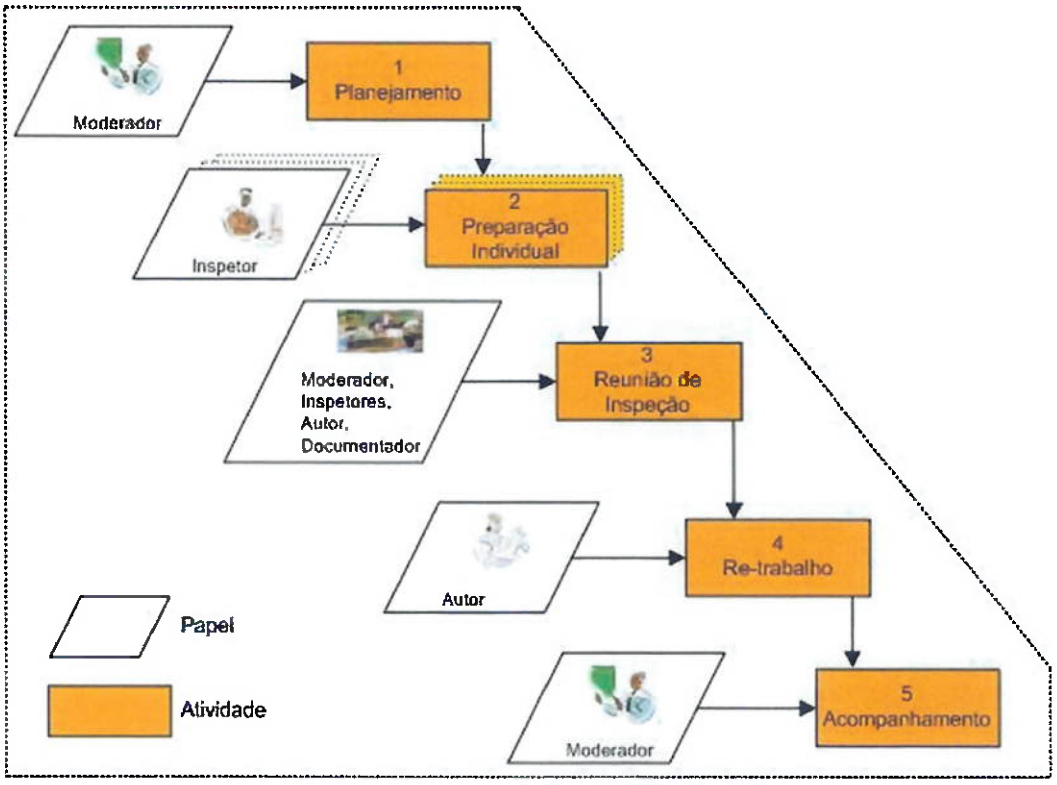


*Figura 1: Figura 1: Processo de Inspeção (MERTZ, 1993)*

### a) Planejamento

As atividades na etapa de Planejamento são executadas pelo Moderador, que assegura

que o produto está pronto para inspeção, seleciona a equipe, assinala os papéis (Inspetor, Leitor, Autor e Documentador), agenda a data e local das reuniões e distribui o material para inspeção (o produto, informações auxiliares, *checklists* entre outros). Também avalia se a etapa Elaboração da Visão Geral é necessária.

Para a avaliação dos critérios de entrada, tem-se uma boa oportunidade para o uso de ferramentas como: corretores ortográficos, analisadores estáticos de código ou compiladores.

### b) Elaboração da Visão Geral

A Elaboração da Visão Geral é uma etapa opcional, que envolve toda a equipe, e é agendada se a equipe de inspeção não é familiar ao material a ser inspecionado, ou se novas técnicas estão sendo introduzidas com o produto. Todos os membros da equipe de inspeção são considerados Inspetores, que identificam defeitos durante a Preparação Individual e durante as Reuniões de Inspeção. Nesta etapa, o Autor apresenta o produto, seu relacionamento com o resto dos produtos sendo desenvolvidos, suas funções e uso pretendido, e a abordagem usada em seu desenvolvimento.

### c) Preparação Individual

A Preparação Individual é elemento chave do processo de inspeção. Durante esta etapa, os membros da equipe de inspeção individualmente se preparam para seus papéis na reunião.

Cada Inspetor revisa o produto. Os Inspetores procuram por problemas gerais, bem como aqueles relacionados a sua área de especialidade.

*Checklists* podem ser utilizados durante esta etapa, como guia dos defeitos típicos no tipo de produto que está sendo inspecionado, bem como violações de normas e padrões. Os defeitos são registrados, juntamente com o tempo gasto durante a preparação. Os

registros então são encaminhados ao Moderador, antes da Reunião de Inspeção, de forma que este possa avaliar se a equipe está adequadamente preparada, podendo inclusive adiar a reunião, se for necessário.

**d) Reunião de Inspeção**

Durante a Reunião de Inspeção, os Inspetores revisam o produto como uma equipe.

Basicamente, as atividades são: o Leitor faz uma leitura e interpretação do produto, o Autor provê os esclarecimentos conforme necessário, e a equipe de inspeção identifica os defeitos, que são classificados e registrados. O foco do trabalho é identificar defeitos, não como corrigí-los ou gerar novas alternativas de solução para o projeto, mas freqüentemente, a solução é simples, podendo assim ser anotada.

Se a equipe é nova, ou possui novos membros, o Moderador pode começar a reunião introduzindo as pessoas da equipe, descrevendo seus papéis e assim reforçando o propósito da inspeção e do produto.

O Leitor então começa uma interpretação lógica e ordenada do produto, que deve conter a descrição das funções dos itens e seus relacionamentos. Os Inspetores podem interromper o Leitor a qualquer tempo, se um possível defeito for identificado. Se uma breve discussão do possível defeito for necessária, a leitura é interrompida temporariamente. A leitura é retomada quando o defeito for registrado e categorizado pelo Documentador.

O Moderador deve limitar as discussões que demorarem demais, talvez impondo um limite de tempo, após o qual o item é declarado em aberto e a reunião prossegue. A equipe deve alcançar consenso se o defeito é real, ou se é um engano por parte do Inspetor ou apenas um mau entendimento a ser esclarecido pelo Autor. Se não houver consenso, o item também é marcado como em aberto. O Documentador anota cada defeito, sua descrição, classificação (tipo e severidade), localização e o nome do Inspetor que a identificou.

Ao fim da reunião, o número de defeitos é sumarizado, e o Moderador determina se será

necessária uma re-inspeção. A re-inspeção pode ser requerida quando o número de defeitos for muito grande, ou o defeito obrigar correções extensas ou complicadas. Nestes casos, é melhor que o produto seja revisto por toda a equipe, ao invés de apenas pelo Moderador.

Para evitar fadiga, a reunião deve se limitar a duas horas, agendando-se novas reuniões se necessário. Após a reunião, o Moderador e o Autor estimam o tempo para o re-trabalho e a agenda para o Acompanhamento.

Para os itens em aberto, tarefas (pesquisas, reuniões, relatórios) são assinaladas para inspetores mais adequados para analisar o assunto, podendo envolver o Autor e mesmo novos participantes.

**e) Re-trabalho**

O propósito do re-trabalho é corrigir os defeitos encontrados durante a inspeção. O autor é responsável pela correção de todos os defeitos graves e pelos não graves (se o custo e tempo permitirem). O Moderador deve se assegurar que qualquer informação gerada pelos itens em aberto seja comunicada e endereçada pelo Autor.

**f) Acompanhamento**

O Acompanhamento normalmente é realizado na forma de uma breve reunião entre o Moderador e o Autor, para assegurar que todos os defeitos graves encontrados durante a inspeção foram corrigidos e que nenhum novo defeito foi introduzido.

São revisadas as ações para correção dos erros e se todos os itens em aberto foram tratados. O Moderador se assegura que todos os critérios de finalização da inspeção foram atingidos e então gera o relatório de conclusão. Se as condições não forem alcançadas, o Autor retorna para a etapa de re-trabalho.

## 2.5. *Considerações do capítulo*

O modelo proposto trata dos ganhos obtidos com a implantação do processo de Inspeção de Software sem a adoção de uma técnica específica, conforme as discutidas neste capítulo. No entanto, estas técnicas são importantes para obter maior produtividade nas atividades de Inspeção de Software.

# 3. Return on Investment – ROI

Segundo Friedlob (1996), os proprietários de empresas e investidores usam o ROI para avaliar a capacidade da empresa em obter uma taxa adequada de retorno, prover informação sobre a eficiência do gerenciamento e projetar ganhos futuros. A gerência mede o desempenho individual de cada segmento de mercado quando este é tratado como um centro de investimento, avalia propostas de gastos com investimentos e auxilia na configuração dos objetivos gerenciais.

O ROI é uma ferramenta financeira que mede o retorno financeiro de um projeto ou investimento. ROI mede a eficácia do investimento através do cálculo do número de vezes que os benefícios líquidos (benefícios menos custos) superam o investimento original. ROI tem se tornado uma das mais populares métricas usadas para entender, avaliar e comparar o valor de diferentes opções de investimentos (FRIEDLOB *et al,* 1996).

Diversas variações de equações de ROI têm surgido nos últimos anos e modificadas para as necessidades individuais das empresas e indústrias. A equação do ROI padrão que será utilizada como base teórica para proposta deste trabalho é apresentada com mais detalhes neste capítulo junto com as definições dos termos da equação.

## *3.1. Modelo do ROI*

O objetivo do modelo do ROI é calcular o retorno sobre o investimento e para isso é necessário ter o valor do projeto. O modelo de ROI pode ser desenvolvido considerando todas as entradas e hipóteses que devem fazer parte da equação do ROI. Para começar é necessário conhecer os dois principais fatores no cálculo do ROI: benefícios e custos associados.

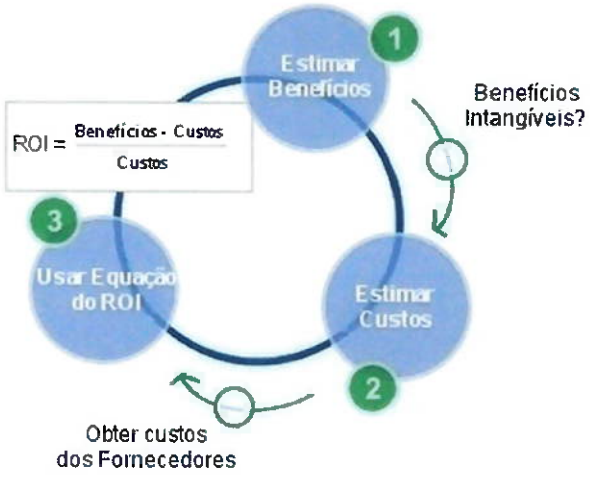


*Figura 2: Modelo do ROI (FRIEDLOB et al, 1996)*

Os benefícios de um projeto são classificados como tangíveis e intangíveis. Os benefícios tangíveis podem ser facilmente medidos e observáveis. Benefícios intangíveis são mais difíceis de observar e não podem ser facilmente determinados como saída por serem dependentes de muitas variáveis. Por exemplo, a satisfação do cliente é dependente de algumas variáveis como: serviço ao cliente, qualidade dos produtos, preços e reputação da empresa. Fazer uma discreta correlação entre a satisfação do cliente e uma melhoria em uma destas áreas não é tarefa fácil (FRIEDLOB *et al,* 1996).

A idéia é completar o modelo do ROI com benefícios tangíveis primeiro para poupar tempo e esforço. No entanto, se o ROI não é como esperado, os benefícios intangíveis terão que ser incluídos no cálculo para justificar o projeto. Dependendo do projeto, esses benefícios intangíveis são geralmente calculados através de modelagem estatística das variáveis de entrada (FRIEDLOB *et al,* 1996).

Uma vez calculados os benefícios do projeto, o próximo passo é identificar e calcular os custos. A idéia é captar plenamente, não só os custos de início do projeto, mas também os custos de manutenção e apoio ao projeto durante o seu ciclo de vida. Os custos são muito mais simples de calcular que os benefícios, pois são mais facilmente observáveis e podem ser obtidos com os fornecedores (FRIEDLOB *et al,* 1996).

O modelo do ROI é completado colocando os benefícios e custos na equação do ROI

(ROI = (benefícios – custos) / custos). Em suma, o desafio de construir o modelo não é o cálculo por si só mas obter as entradas da equação. As hipóteses para o estabelecimento dos benefícios e custos no modelo do ROI é um passo crítico na obtenção da representação exata do valor do projeto (FRIEDLOB *et al,* 1996).

## 3.2. *Equações*

Apresenta-se a seguir a equação padrão para o cálculo do ROI.

**ROI = benefícios líquidos / custos * 100% (1)**

benefícios líquidos = diferença entre os benefícios do projeto e os custos associados usados para gerar esses benefícios (PETERSEN , 2003).

benefícios líquidos = [benefícios – custos]

Na equação do ROI, os benefícios líquidos podem ser calculados com ou sem impostos ou depreciações.

Benefício líquido também pode ser definido como lucro para o primeiro ano ou um lucro médio durante o tempo de vida do projeto.

Algumas empresas usam algum tipo de taxa para ajustar lucros futuros para valor presente. Basicamente, essa taxa ajusta para um determinado valor monetário no tempo, o custo de capital da companhia.

custos = são todos os custos incorridos para obter os benefícios líquidos do projeto ou investimento (PETERSEN , 2003).

O termo custo na equação do ROI pode estar sujeito às mesmas considerações dos benefícios líquidos. Algumas empresas definem custos como gastos de capital do primeiro ano, enquanto outras companhias incluem custos periódicos ou recorrentes como atualização e manutenção de software. Os custos recorrentes são calculados como média do capital investido durante o tempo de vida do projeto (PETERSEN , 2003).

O período de tempo para estimar os benefícios e custos variam criando alguma

complexidade na interpretação dos resultados. Algumas empresas usam um ano, ou seja, aprovam os projetos que estão aptos a recuperar o valor investido no primeiro ano de operação. Outras empresas determinam o ROI final do investimento de acordo com o fluxo de caixa (PETERSEN , 2003).

Existem diversas variações da equação do ROI dadas as múltiplas interpretações e aplicações em diferentes indústrias. Essa falta de consistência pode causar confusão quando são comparados valores de ROI de diversos projetos.

Abaixo seguem algumas variações comuns da equação do ROI (FRIEDLOB *et al*, 1996):

1. **Retorno do capital investido**

**ROIC (Return On Invested Capital) = NOPAT (Net Operating Profits After Tax) / capital investido * 100% (2)**

NOPAT: lucro líquido tirando os impostos

capital investido: custos iniciais e recorrentes

2. **Retorno sobre investimento usando valor presente líquido ou fluxo de caixa descontado**

**ROI = NPV(benefícios líquidos) / PV (custos) * 100% (3)**

Essa equação aponta para o valor do dinheiro no tempo ou o interesse derivado de um investimento com risco similar. O valor presente é descontado de acordo com o custo do capital para a empresa ou a taxa através da qual a empresa emprestaria dinheiro do mercado dado o nível do risco.

NPV(benefícios líquidos) = valor presente dos benefícios menos o valor presente dos custos

PV(custos) = valor presente dos custos

3. **Retorno sobre investimento – Definição financeira**

**ROI = Lucros / Ativos da empresa * 100% (4)**

Lucros = renda líquida. Renda líquida é definida como lucros brutos menos deduções

referentes à depreciação, pagamentos e impostos

Ativos da empresa = recursos da empresa incluindo dinheiro, contas a receber, inventário e propriedades

4. **Retorno sobre investimento – Melhor visão**

**ROI = total de benefícios líquidos/ custos * 100% (5)**

Essa equação assume o melhor cenário possível resultando em um valor maior de ROI. Todos os benefícios estão inclusos no numerador e somente os custos iniciais são calculados no denominador.

total de benefícios líquidos = benefícios menos custos iniciais

custos = custos iniciais

### *3.3. Exemplo de elaboração de um ROI*

O exemplo de ROI a seguir mostra passo a passo como calcular o ROI para um investimento de software (PETERSEN , 2003).

Considere a implementação de um software de CRM (*Customer Relationship Management*) para uma companhia de tamanho médio nos Estados Unidos. O CEO (*Chief Executive Officer*) da empresa solicitou ao CIO (*Chief Information Officer )* que colocasse junto com as finanças as justificativas para o grande investimento de capital. O CIO enfrenta com o desafio, não só da execução do projeto, mas também de desenvolver uma análise financeira sólida. O CIO assume, conforme modelo proposto na Figura 2, os passos a seguir para determinar o retorno do projeto (PETERSEN , 2003):

**Passo 1 – Estimar os benefícios e custos do projeto.**

O primeiro passo neste exemplo de ROI para aplicação de CRM é estabelecer os benefícios e custos do projeto. O software de CRM tem sido muito popular nos últimos

dez anos para ajudar as empresas interagirem com os clientes via Internet, e-mail, telefone, correio, fax, etc. O objetivo da aplicação CRM é melhorar a experiência do cliente global integrando todos os pontos de contato do cliente com a empresa. A aplicação CRM também fornece à empresa uma oportunidade para aumentar as vendas através de uma combinação de análises inteligentes do cliente e oferecer produtos adaptados a cada um deles. Os benefícios e custos descritos a seguir são típicas implementações de um CRM:

**A. Benefícios**

Aumento das receitas

- Aumentar a retenção de clientes: reduzir problemas nos clientes mensalmente, perder menos clientes por causa de um serviço pobre prestado ao cliente.
- Aumentar a velocidade ao mercado (*time-to-market*): aumento de receita com a melhoria da velocidade dos produtos e serviços ao mercado, desenvolvimento do produto.

Redução de custos

- Reduzir custos da transação: reduzir média de tempo para chamadas *handoffs* (processo de transferência de uma chamada em andamento de uma canal conectado a uma rede para outra), número de chamadas, custo de análise e geração de relatórios e aumento na solução de casos na primeira chamada.
- Reduzir custos de operação: reduzir complexidade das operações existentes na gerência de requisições de clientes e de entregas.
- Reduzir custos de TI: reduzir custos de manutenção através de plataforma consolidada.

**B. Custos**

Licença: Licença do software da empresa que fornece o software de CRM

- Fase de levantamento de requisitos de software e customizações: Desenvolvimento de requisitos específicos de software devido à operação da empresa e seus sistemas legados
- Fase de Desenvolvimento: Customização do software para atender aos requisitos da empresa
- Garantia de Qualidade: Testes das mudanças do software e integração com sistemas legados
- Treinamento: Treinamento do pessoal que irá utilizar o software de CRM
- Conversão: Conversão de sistemas legados e integração ao novo software de CRM
- Implantação: Implantação do software, o que inclui o tempo que este ficará fora e o impacto na satisfação do cliente durante a implantação
- Manutenção e Suporte: Manutenção do software e o suporte provido pelo fornecedor do software
- Atualizações do software: Atualizações do software durante o ciclo de vida do projeto de CRM.

Os benefícios e custos do CRM exemplo são tabulados para os primeiros 4 anos do projeto (valores em milhões) (PETERSEN , 2003):

| Benefícios | Ano 1 | Ano 2 | Ano 3 | Ano 4 |
|---|---|---|---|---|
| Reduzir custos da transação | $2.5 | $3.0 | $3.2 | $3.5 |
| Reduzir custos da operação | $0.8 | $1.0 | $1.1 | $1.2 |
| Reduzir custos com TI | $0.0 | $1.5 | $1.5 | $1.5 |
| Aumentar a retenção de clientes | $3.5 | $4.0 | $5.0 | $6.5 |
| Aumentar a velocidade ao mercado | $0.0 | $2.5 | $2.5 | $2.5 |
| **Total de benefícios** | $6.8 | $12.0 | $13.3 | $15.2 |

| Custos | Ano 1 | Ano 2 | Ano 3 | Ano 4 |
|---|---|---|---|---|
| Licença | $1.0 | $1.0 | $1.0 | $1.0 |
| Fase de levantamento de requisitos | $0.5 | $0.0 | $0.0 | $0.0 |
| Fase de Desenvolvimento | $0.5 | $0.0 | $0.0 | $0.0 |
| Garantia de Qualidade | $0.8 | $0.2 | $0.2 | $0.2 |
| Treinamento | $0.7 | $0.1 | $0.1 | $0.1 |
| Conversão | $0.3 | $0.0 | $0.0 | $0.0 |
| Implantação | $0.4 | $0.0 | $0.0 | $0.0 |
| Manutenção e Suporte | $0.0 | $0.1 | $0.1 | $0.1 |
| Atualizações de software | $0.0 | $0.0 | $0.0 | $0.4 |
| **Total de custos** | $4.2 | $1.4 | $1.4 | $1.8 |

**Passo 2 – Calcular o valor do projeto usando o ROI**

O segundo passo neste exemplo é calcular os termos da equação do ROI. Os termos do CRM exemplo são calculados abaixo (PETERSEN , 2003):

| **Resultados** | **Total** |
|---|---|
| Benefícios no primeiro ano | $6.8 |
| Custos no primeiro ano | $4.2 |

O último passo é usar os números na mais comumente usada equação do ROI:

**1- ROI padrão**

ROI = (benefícios – custos) / custos * 100%

ROI = ($6.8-$4.2)/$4.2 = 62%

## *3.4. Benefícios*

O principal benefício do uso do ROI e a razão da sua popularidade é a simplicidade de cálculo. O ROI estima o retorno de um investimento olhando os benefícios e custos associados ao investimento. O ROI é um primeiro e apropriado passo para estimar o retorno econômico de um investimento (PETERSEN , 2003).

ROI também é um veículo útil para comunicar o retorno de um investimento para aqueles que tem graus variados de conhecimento financeiro. O conceito de ROI possibilita aos gerentes falar a mesma língua quando comunicam objetivos do projeto em termos financeiros nos diversos departamentos da empresa. Dessa forma, fornecedores de TI usam ROI como ferramenta de venda para transmitir facilmente o valor econômico dos seus produtos (PETERSEN , 2003).

Além de ser útil como veículo de comunicação, ROI pode ser usado como filtro inicial na avaliação de projetos. Por exemplo, o ROI pode ser calculado para todos os projetos na empresa e somente aqueles que excederem o ROI mínimo serão analisados. Assim, o ROI pode identificar projetos válidos sem utilizar recursos consideráveis (PETERSEN , 2003).

Outros benefícios do ROI incluem o uso da ferramenta para estabelecer um primeiro conjunto de hipóteses a fim de determinar o valor do projeto. O cálculo do ROI força a uma profunda reflexão de variáveis tais como investimentos intangíveis, risco e tempo. Estas hipóteses são tão importantes como o cálculo do ROI em si, porque eles são os fatores que, em última análise, impactam o resultado global do ROI (FRIEDLOB *et al,* 1996).

Devido aos benefícios anteriormente descritos, o conceito do ROI tornou-se particularmente popular na avaliação de projetos de TI. Às vezes, o ROI tem quase se tornado sinônimo de análise financeira para investimentos em TI. Para evitar qualquer confusão nas nossas discussões, consideramos o conceito do ROI como apenas um dos instrumentos financeiros disponíveis.

## 3.5. *Limitações*

Para entender os benefícios e o potencial do uso do ROI, é necessário também entender suas limitações. Gerentes que tem um claro entendimento das limitações podem tirar vantagens da metodologia sem distorcer seu valor como ferramenta de avaliação financeira (FRIEDLOB *et al*, 1996).

O cálculo do ROI não leva em conta o valor temporal do dinheiro e o risco associado com um projeto ou investimento. O conceito de valor temporal do dinheiro afirma que um dólar ganho hoje vale mais que um dólar ganho amanhã. Isto é particularmente verdade quando se considera outras alternativas de investimento, bem como os efeitos da inflação a partir de uma perspectiva macroeconômica (PETERSEN , 2003).

Além disso, o risco do projeto precisa de ser contabilizado pela incorporação de todos os possíveis resultados financeiros associados ao projeto. Estes possíveis resultados incluem as possibilidades de o projeto não produzir os resultados esperados por causa dos riscos inerentes ao projeto (FRIEDLOB *et al*, 1996).

O fato de risco e valor temporal do dinheiro serem omitidos no cálculo do ROI pode levar os gestores a rejeitar, por engano, projetos que outrora deveriam ter sido aprovados. Este é particularmente o caso de projetos com perfis de risco diferentes quando comparados usando o modelo de ROI. O valor do ROI do projeto com o maior risco deve ser reduzido para ter em conta o vasto leque de resultados, quando comparados com um projeto com menor risco (FREIDLOB *et al*, 1996).

O fato de o ROI poder ser calculado de diferentes maneiras cria um problema de consistência. Poucas empresas têm desenvolvido um único modelo de ROI, tornando assim difícil de comparar e avaliar com precisão o rendimento econômico de vários projetos.

Este problema é acentuado quando se compara o valor do software de múltiplos fornecedores, cada um deles pode ter um diferente modelo utilizado para chegar ao ROI para seu software específico. Gerentes que selecionam softwares de fornecedores têm a tarefa de identificar o verdadeiro ROI de cada fornecedor e criar uma visão única e padronizada para o propósito de comparação (PETERSEN , 2003).

## 3.6. *ROI em Software*

O ROI, fora do Brasil, é amplamente adotado pelas empresas para avaliação do valor do investimento em software em cada fase do desenvolvimento desde o início da sua implementação. De fato, nos últimos anos, as análises de ROI tornaram-se sinônimos de avaliação financeira de software (PETERSEN , 2003).

Esta ampla adoção do ROI dentro a indústria de software é principalmente impulsionado pelos seguintes fatores (FRIEDLOB *et al*, 1996):

- Simplicidade. ROI é um conceito relativamente simples e intuitivo. O cálculo do ROI é simples, em comparação com outros mais complexos, tais como ferramentas avaliação financeira: *Net Present Value* (NPV) e *Real Options*.
- Fácil de Comunicar. ROI pode ser usado como um veículo para comunicar o valor de um projeto para equipes funcionais e executivas com diferentes graus de conhecimentos em Finanças.
- Alinhamento com Requisitos da Empresa. As empresas estão exigindo mais do que nunca a recuperação de investimentos em software dentro do primeiro ano de operações. Esta regra é principalmente impulsionada pelas expectativas do mercado, bem como o ritmo da tecnologia, o que torna novos desenvolvimentos obsolescentes dentro de alguns anos. O conceito de ROI é bem adaptado para esta aplicação uma vez que o cálculo ROI envolve os custos e benefícios do projeto, tendo o primeiro ano como base, para estimar o valor do investimento.
- Padrão da Indústria. Fornecedores de software têm adotado amplamente o modelo de ROI como uma ferramenta de vendas para fornecer aos seus clientes o valor esperado de seu software.

Todos esses fatores têm contribuído para a popularidade e a continuidade do uso de ROI para avaliar investimentos em softwares.

## 3.7. *Considerações do capítulo*

A equação padrão do ROI é uma ferramenta fácil e prática para mostrar os benefícios obtidos com investimentos na área de Software, por isso será utilizada no âmbito da Qualidade, no capítulo 4 como base para a conceituação, definição e exemplificação do modelo de ROI para Inspeção de Software.

# 4. Modelo de ROI para Implantação de um Processo de Inspeção de Software

## *4.1. Introdução*

Para entender a proposta de um modelo de ROI para a Implantação do Processo de Inspeção de Software, é necessário destacar alguns pontos discutidos nos 2 últimos capítulos e assim traçar um relacionamento entre ambos.

O capítulo 2 conceituou a Inspeção de Software, descreveu as técnicas mais utilizadas e descreveu o processo. Foi visto que tratam-se de atividades de Verificação e Validação que podem ocorrer durante todo o ciclo do processo de desenvolvimento de software e que a Inspeção de Software é mais eficaz que os testes na detecção dos defeitos do software. Destacou-se também que há vantagens no uso desse método pois as atividades são incluídas já nas fases iniciais possibilitando assim a detecção dos defeitos previamente.

As técnicas descritas têm vantagens umas em relação às outras mas, não é escopo deste trabalho destacar tais diferenças no modelo de ROI proposto para a Inspeção de Software.

No capítulo 3 o ROI foi citado como uma das principais ferramentas que as empresas têm para avaliar o retorno financeiro e assim medir a eficácia do investimento. Destacou-se que existem diversas variações de ROI, mas é usada no modelo de ROI proposto para a implantação do Processo de Inspeção de Software a equação padrão do ROI para facilitar a compreensão da proposta deste trabalho. Apesar de a equação padrão do ROI ser simples, uma vez que os fatores são os benefícios e os custos associados, o principal desafio é identificá-los conforme a Figura 2.

## 4.2. *Relação do ROI com a Inspeção de Software*

Para traçar o relacionamento do ROI com a Inspeção de Software, os benefícios podem ser descritos como a Prevenção de Custos que está associada à detecção e correção dos defeitos nas fases iniciais do ciclo de evolução do produto. Os custos podem ser descritos como o Custo para Detecção do defeito em função da utilização da Inspeção de Software (O'NEIL, 2002).

Portanto, o modelo de ROI, segundo Friedlob (1996), adequado à Inspeção de Software segundo O'NEIL (2002) pode ser assim descrito:

ROI = (benefícios – custos) / custos (FRIEDLOB *et al*, 1996)

ROI = Economia Líquida / Custo para Detecção (O'NEIL, 2002)

sendo:

Economia Líquida = Prevenção de Custo – Custo para Reparar Agora

Custo para Detecção = Custo do Esforço da Preparação + Custo do Esforço da Condução

Portanto, para a chegar ao propósito deste trabalho que é obter ganho financeiro com a implantação do processo de Inspeção de Software, o mapeamento das entradas da equação padrão do ROI pode ser feito da seguinte forma:

- benefícios = Prevenção de Custo
- custos (numerador) = Custo para Reparar Agora
- custos (denominador) = Custo do Esforço da Preparação + Custo do Esforço da Condução

A Economia Líquida é função da Prevenção de Custo que resulta da detecção e correção precoce evitando o aumento do multiplicador do custo associado com a detecção e correção tardia de defeitos no ciclo de vida (O'NEIL, 2002).

Portanto, quanto mais cedo o defeito for detectado maior será a Economia Líquida, uma vez que os defeitos maiores que não são detectados da fase de Desenvolvimento para o

Teste podem custar de 2 a 10 vezes mais para detectar e corrigir. Alguns desses defeitos que não são detectados da fase de Teste para o Cliente podem custar adicionalmente de 2 a 10 vezes para detectar e corrigir. Os defeitos menores podem custar adicionalmente de 2 a 4 vezes para corrigir depois (O'NEIL, 2002).

O Custo para Detecção aumenta naturalmente uma vez que está se implantando um processo de Inspeção de Software cujas atividades demandam capacitação dos desenvolvedores nos seus mais diversos papéis e dedicação de tempo para executar o processo descrito no capítulo 2.

## *4.3. Pontos a observar no modelo proposto*

Para que haja um ROI com Inspeção de Software satisfatório, a Economia Líquida obtida com a Implantação de um processo de Inspeção de Software deve superar o Custo para Detecção de tal forma que justifique; por isso, alguns pontos devem ser observados na hora de implantar este processo:

- Maturidade da equipe – mitigar risco de insucesso na implantação;
- Abordagem da Engenharia de Software adotada pela organização no Processo de Desenvolvimento.

Como o Processo de Inspeção de Software contém atividades que apontam defeitos em artefatos, tanto os Autores como os Inspetores devem ser imparciais e tratar essas atividades como ferramenta para melhoria do produto e do processo. Com a ajuda do Moderador, muitas vezes, é possível conduzir as Reuniões de Inspeção para que elas não se desviem do seu principal objetivo, mas de nada adiantará se todos os envolvidos não tiverem plena consciência dos seus papéis dentro de um contexto estritamente profissional.

Na Engenharia de Software a abordagem Ad-hoc é marcada pela ausência de um processo de desenvolvimento bem definido, na abordagem Estruturada o processo acontece em fases seqüenciais, uma só inicia quando a outra termina e na abordagem Disciplinada o processo é iterativo e incremental, ou seja, as atividades das disciplinas (Modelagem de Negócios, Requisitos, Análise e Projeto, Implementação, Teste, etc)

ocorrem em pequenas iterações dentro das fases (Concepção, Elaboração, Construção e Transição).

Além disso, as abordagens da Engenharia de Software (Ad-hoc, Estruturada e Disciplinada) adotadas pela organização antes de implantar o processo de Inspeção de Software podem contribuir para dar pesos diferentes para os parâmetros que compões os custos e os benefícios. Tais parâmetros são: Multiplicador de custo, Taxa de detecção de defeitos e Custo para corrigir (O'NEIL, 2002).

- Multiplicador de custo: as economias resultam da detecção e correção precoce do defeito evitando o aumento do multiplicador do custo associado à detecção e a correção tardia dos defeitos no ciclo de vida (O'NEIL, 2002);
- Taxa de detecção de defeitos: número de defeitos detectados / número de defeitos presentes (O'NEIL, 2002);
- Custo para corrigir: custo para corrigir um defeito com base na experiência da equipe (O'NEIL, 2002).

As abordagens da Engenharia de Software interferem nos parâmetros citados da seguinte maneira:

- Na abordagem Ad-hoc, o Multiplicador de Custo é de 8 a 10 vezes para detectar e corrigir defeitos maiores e de 4 vezes para defeitos menores, a Taxa de detecção de defeitos está entre 0,50 e 0,65 (O'NEIL, 2002);
- Na abordagem Estruturada, o Multiplicador de Custo é de 5 a 7 vezes para detectar e corrigir defeitos maiores e de 3 vezes para defeitos menores, a Taxa de detecção de defeitos está entre 0,70 e 0,80 (O'Neil, 2002);
- Na abordagem Disciplinada, o Multiplicador de Custo é de 2 a 4 vezes para detectar e corrigir defeitos maiores e de 2 vezes para defeitos menores, a Taxa de detecção de defeitos está entre 0,85 e 0,95 (O'NEIL, 2002).

Essa interferência nos parâmetros pela abordagem adotada pode ser melhor explicada pela tabela a seguir:

| Abordagem da Engenharia de Software | Multiplicador de Custo | Taxa de detecção de defeitos | Custo para corrigir |
|---|---|---|---|
| **Ad-hoc** | Defeitos Maiores: 8-10<br>Defeitos Menores: 4 | 0,50 – 0,65 | Experiência Equipe |
| **Estruturada** | Defeitos Maiores: 5-7<br>Defeitos Menores: 3 | 0,70 – 0,80 | Experiência Equipe |
| **Disciplinada** | Defeitos Maiores: 2-4<br>Defeitos Menores: 2 | 0,85 – 0,95 | Experiência Equipe |

## *4.4. Considerações do capítulo*

Pode-se destacar resumidamente que quanto mais maduro for o Processo de Desenvolvimento de Software, menor será o Multiplicador de Custo e maior será a Taxa de detecção de defeitos, ou seja, maior será a Prevenção de Custo (benefícios) e portanto, maior será a Economia Líquida resultando assim em um ROI maior comparando-se a um Processo de Desenvolvimento de Software menos maduro (O'NEIL, 2002).

# 5. Considerações Finais

Este trabalho apresentou uma visão geral e conceitual sobre o Processo de Inspeção de Software e ROI e propôs um modelo de ROI adequado ao Processo de Inspeção de Software relacionando todas as entradas do modelo de ROI padrão aos fatores da equação diretamente relacionados à Inspeção de Software mapeando os benefícios e os custos apropriadamente.

Descreveu de forma detalhada as técnicas e o Processo de Inspeção de Software, assim como o ROI conceituando-o e descrevendo algumas das diversas equações existentes. Mostrou também a importância cada vez maior na utilização do ROI como ferramenta para tomada de decisão sobre investimentos por parte das empresas e fornecedores como instrumento de venda de seus produtos.

No entanto, ainda há que se aprofundar na discussão com relação às vantagens no uso de algumas técnicas de Inspeção de Software em detrimento de outras levando-se em consideração a abordagem da Engenharia de Software adotada pela organização, uma vez que dependendo do nível de maturidade do processo e da equipe, talvez haja a necessidade de se detalhar melhor como seria calculado o ROI, segundo o modelo proposto, para que se possa obter o melhor resultado e assim fornecer subsídios mais fundamentados e concretos de acordo com o contexto de cada organização.

Propõe-se ainda uma discussão mais detalhada do capítulo 2 levando-se em consideração as três abordagens da Engenharia de Software (Ad-hoc, Estruturada e Disciplinada) discutidas no capítulo 4 de maneira a possibilitar uma análise de como as técnicas podem ser escolhidas e como o processo de Inspeção de Software pode ser implantado.

## Referências

Ackerman, A. F.; Buchwald, L. S.; Lewski, F. H. **Software Inspections: An Effective Verification Process.** Software IEEE May 1989, p.31-36, vol.6, n.3.

CIOLKOWSKI, M. **Evaluating the effectiveness of different inspection techniques on informal requirements documents.** Tese(Doutorado) – University of Kaiserlautern,1999, 165p.

CONRADI, R.; MOHAGHEGHI, P.; ARIF, T.; HEDGE, L.C.; BUNDE, G.A.; PEDERSEN, A. **Object-Oriented Reading Techniques for Inspection of UML Models An Industrial Experiment.** European Conference on Object-Oriented Programming, Darmstadt, 2003. Proceedings of the European Conference on Object-Oriented Programming. Darmstadt: Springer-Verlag, 2003. p.483-501.

CROUST, G.; LEXEN H. **Software Inspections - Theorie, New approaches and an Experiment, Proceedings.** 25th EUROMICRO Conference, 1999, p.286 - 293 vol.2.

DENGER, C.; SHULL, F. **A Practical Approach for Quality-Driven Inspections**, March/ April 2007, p.79-86 vol.24, n.2.

ENDO, A. T., PEREIRA Jr, C.A.F. **Atividades de Verificação e Validação: Técnicas de Leitura de Software.** Artigo - IME-USP 2006. 7p.

FAGAN, M. **Design and code inspection to reduce errors in program development.** IBM System Journal 1976, p.182-211 v.5, n.3.

FAGAN, M. **Advances in Software Inspection.** IEEE Transactions on Software Egineering July 1986, p.744-751, vol.12, n.7.

FRIEDLOB, G. T.; PLEWA Jr., F. J. **Understanding Return on Investment.** Wiley Publisher (April 1996), 237p.

MAFRA, S.N.; TRAVASSOS, G.H. **Técnicas de Leitura de Software: Uma Revisão Sistemática.** XIX Simpósio Brasileiro de Engenharia de Software, Uberlândia, 2005. Anais Eletrônicos da BDBComp (Biblioteca Digital Brasileira de Computação). Minas Gerais: UFMG, 2005. 16p.

MERTZ , CHARLES W. **Software Formal Inspection Guidebook.** Office of Safety and Mission Assurance. Washington DC: National Aeronautics Space and Administration, 1993. (Technical Report NASA-GB-A302). 68p.

O'NEILL, D. **Return on Investment using Software Inspections.** Available WWW <URL:http://members.aol.com/ONeillDon/roi-essay.html> (2002).

PAGLIUSO, P. B. B.; TAMBASCIA, C. A.; VILLAS-BOAS, A. **Melhoria da Inspeção de Requisitos segundo a técnica de Leitura Baseada em Perspectiva.** XI Seminário de Computação, Blumenau, 2002. Anais Eletrônicos da Universidade Regional de Blumenau. Blumenau: FURB, 2002. p.105-116.

PETERSEN, G. S. **ROI: Building The CRM Business Case**, Xlibris Publisher (September 16, 2003). 380p.

WHITTAKER, J.; VOAS, J. **50 Years of Software: Key Principles for Quality.** IT Professional Nov/Dec 2002, p.28-35, v.4, n.6,