Paulino Wagner Palheta Viana

SPIREM-OBK – SOFTWARE PROCESS IMPROVEMENT ROI ESTIMATE MODEL ORIENTED BY KNOWLEDGE

Universidade Fernando Pessoa
Porto 2023

Paulino Wagner Palheta Viana

SPIREM-OBK – SOFTWARE PROCESS IMPROVEMENT ROI ESTIMATE MODEL ORIENTED BY KNOWLEDGE

Universidade Fernando Pessoa
Porto 2023

Paulino Wagner Palheta Viana

SPIREM-OBK – SOFTWARE PROCESS IMPROVEMENT ROI ESTIMATE MODEL ORIENTED BY KNOWLEDGE

Tese apresentada á Universidade Fernando Pessoa como parte dos requisitos para a obtenção do grau de Doutor em Ciência da Informação, especialização em Sistemas, Tecnologias e Gestão da Informação, sob a orientação do Prof Doutor Luis Borges Gouveia.

Resumo

Em geral empresas de software não conseguem justificar os seus investimentos em projetos de melhoria em processo de software por não possuírem um modelo de estimativa de software que demonstre, de forma objetiva, o ROI (*return on investment*). O facto é que, modelos de estimativas de software, não informam o ROI para melhoria de processo de software. Muitos dos argumentos encontrados na literatura e através de artigos, relatam experiências de vários benefícios (ROI) em implantações de melhoria de processo de software no *post mo*rten. Mas o principal desafio é como apresentar aos CEOs das empresas de software, que investir em melhoria de processo de software, utilizando uma técnica de estimativa paramétrica, e juntando um arcabouço de conhecimento sobre gestão de projetos e gestão de pessoas da própria empresa, trará um ROI satisfatório.

Esta tese propõe um Modelo de Estimativa de ROI para Melhoria de Processo de Software orientado por conhecimento adquirido nos projetos. O estudo baseou-se em um resultado da pesquisa realizada através de uma revisão sistemática da literatura sobre o tema. Para testificar a relevância e autenticidade do tema em questão. Após a concepção da técnica e o modelo matemático, foi aplicado um estudo de caso de forma estruturada em uma organização de software.

Como o foco principal era atender a questão da pesquisa sobre o nível de satisfação da organização, ao executar o modelo para estimar o ROI em melhoria de processo de software, constatou-se que, o programa de melhoria da organização conclui todos os trabalhos em 11 meses e a estimativa do ROI apresentada pelo *SPIREM-OBK* foi de 10 meses, portanto, o nível de satisfação da estimativa foi alto, pois pontou 90,90% do tempo real. Mas o sucesso da execução do *SPIREM-OBK* fica a cargo exclusivamente do nível de conhecimento contido na organização, que é rico em valores e a habilidade de discernir esse conhecimento faz a diferença.

O modelo proposto possui adaptabilidade para qualquer outra natureza de processos organizacionais, não exclusivamente na área de software. Para tal feito, será necessária a

denominação de uma nomenclatura mais genérica contendo um Guia Instrucional (uma instância), que possa ser utilizado pelas organizações de outros segmentos de negócios.

Palavras-chave: Processo de Desenvolvimento de Software, Precificação de Software, Capitalização de Custos de Desenvolvimento de Software, Melhoria de Processo de Software, *Return on Investment* (ROI).

Abstract

In general, software companies cannot justify their investments in software process improvement projects because they do not have an expected software model that objectively demonstrates the ROI (return on investment). The fact is, software estimation models do not report the ROI for software process improvement. Many of the arguments found in the literature and through articles, report experiences of various benefits (ROI) in software process improvement implementations in the postmortem. But the main challenge is how to present to the CEOs of software companies, who invest in software process improvement, using a parametric estimation technique, and joining a framework of knowledge about project management and people management from the company itself, designed a Prolonged ROI.

This thesis proposes an ROI Estimation Model for Software Process Improvement guided by knowledge acquired in projects. The study was based on a result of research carried out through a systematic review of the literature on the subject. To testify to the relevance and authenticity of the topic in question. After designing the technique and the mathematical model, a case study was applied in a structured way in a software organization.

As the main focus was to answer the research question about the organization's level of satisfaction, when running the model to estimate the ROI in software process improvement, it was found that the organization's improvement program completes all work in 11 months and the ROI estimate presented by SPIREM-OBK was 10 months, therefore, the satisfaction level of the estimate was high, as it scored 90.90% of the real time. But the successful implementation of SPIREM-OBK is exclusively in charge of the level of knowledge contained in the organization, which is rich in values and the ability to discern this knowledge makes the difference.

The proposed model has adaptability to any other nature of organizational processes, not exclusively in the software area. To do so, it will be necessary to name a more generic nomenclature containing an Instructional Guide (one instance), which can be used by organizations in other business segments.

Keywords: Software Development Process, Software Pricing, Capitalization of Software Development Costs, Software Process Improvement, Return on Investment (ROI).

Résumé

En général, les éditeurs de logiciels ne peuvent pas justifier leurs investissements dans des projets d'amélioration des processus logiciels car ils ne disposent pas d'un modèle d'estimation de logiciel qui démontre objectivement le ROI (retour sur investissement). Le fait est que les modèles d'estimation de logiciels ne rapportent pas le retour sur investissement de l'amélioration des processus logiciels. De nombreux arguments trouvés dans la littérature et à travers des articles rapportent des expériences de divers avantages (ROI) dans les implémentations d'amélioration des processus logiciels dans le postmortem. Mais le principal défi est de savoir comment présenter aux PDG des éditeurs de logiciels qu'investir dans l'amélioration des processus logiciels, en utilisant une technique d'estimation paramétrique et en rejoignant un cadre de connaissances sur la gestion de projet et la gestion des personnes de l'entreprise elle-même, apportera un retour sur investissement satisfaisant.

Cette thèse propose un modèle d'estimation du ROI pour l'amélioration des processus logiciels guidé par les connaissances acquises dans les projets. L'étude était basée sur le résultat d'une recherche menée à travers une revue systématique de la littérature sur le sujet. Pour témoigner de la pertinence et de l'authenticité du sujet en question. Après avoir conçu la technique et le modèle mathématique, une étude de cas a été appliquée de manière structurée dans une organisation logicielle.

Comme l'objectif principal était de répondre à la question de recherche sur le niveau de satisfaction de l'organisation, lors de l'exécution du modèle pour estimer le retour sur investissement dans l'amélioration des processus logiciels, il a été constaté que le programme d'amélioration de l'organisation achève tous les travaux en 11 mois et l'estimation du retour sur investissement présentée par SPIREM-OBK était de 10 mois, par conséquent, le niveau de satisfaction du devis était élevé, car il a obtenu un score de 90,90 % du temps réel. Mais la réussite de la mise en place de SPIREM-OBK est exclusivement en charge du niveau de connaissances contenues dans l'organisation, qui est riche en valeurs et la capacité à discerner ces connaissances fait la différence.

Le modèle proposé est adaptable à toute autre nature de processus organisationnels, pas exclusivement dans le domaine logiciel. Pour ce faire, il sera nécessaire de nommer une nomenclature plus générique contenant un guide pédagogique (un cas), qui pourra être utilisé par des organisations d'autres secteurs d'activité.

Mots-clés: processus de développement logiciel, tarification des logiciels, capitalisation des coûts de développement logiciel, amélioration du processus logiciel, retour sur investissement (ROI).

Dedicatória

Dedico esse trabalho às minhas amadas: minha falecida mãe que sempre esteve presente me apoiando em todos os meus desafios; minha filha Carla Beatriz que faleceu de Covid-19; minha esposa Milka Palheta que me apoia desde o início; e minha filha Kátia Rozannah Palheta que sempre demonstrou amor e carinho.

Agradecimentos

Agradeço a Deus, por ter sido abençoado na seleção do Doutorado;

Agradeço a Deus, por ter me protegido e ter colocado pessoas do bem durante minha estada em Porto, Portugal;

Agradeço a Deus, por mesmo ter recebido bênçãos e graças do Senhor, ter tido a oportunidade de ajudar outras pessoas que cruzaram meu caminho;

Agradeço o Amor incondicional da minha família, cito minha mãe falecida Bia Palheta, minha esposa Milka Palheta, e minhas filhas Carla Beatriz e Kátia Rozannah.

Agradeço ao meu Orientador Professor Doutor Luis Borges Gouveia, por ter tido paciência, compreensão e me orientado nesse desafio;

Agradeço aos colegas de doutoramento que sempre se dispuseram em ajudar;

Agradeço aos membros da banca de avaliação do meu trabalho;

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Abreviaturas

CMMI	*Capability Maturity Model Integration*
COCOMO	*COnstructive COst Model*
COTS	*Commercial-Off-The-Shelf*
DMAIC	*Define-Measure-Analyse-Improve-Control*
GQM	*Goal-Question-Metric*
IDEAL	*Initiating, Diagnosing, Establishing, Acting, Leveraging*
MPS	Melhoria de processo de Software
MPS.BR	Melhoria do Processo de Software Brasileiro
PDCA	*Plan-Do-Check-Action*
QIP	*Quality Improvement Paradigm*
ROI	*Return on Investment*
TIR	*Taxa Interna de Retorno*
VPL	*Valor Presente Líquido*

"Confia no Senhor de todo o teu coração, e não te estribes no teu próprio entendimento.
Reconhece-o em todos os teus caminhos, e ele endireitará as tuas veredas.
Não sejas sábio a teus próprios olhos; teme ao Senhor e aparta-te do mal."
Provérbios 3:5-7

Capítulo 1

Introdução

Neste capítulo, são apresentados de forma introdutória, os principais motivos e características que contextualizam o problema relacionado à estimativa de retorno de investimento em iniciativas de melhoria de processo de software. Em seguida, a questão de pesquisa para construir a hipótese é apresentada. As seções seguintes apresentam os principais objetivos dessa tese, descrevem o contexto da pesquisa, as principais contribuições e os resultados esperados e por último a estrutura desse documento.

1.1 MOTIVAÇÃO

Nos últimos anos, as organizações de software têm se preocupado muito com Melhoria de Processo de Software (MPS) para diminuir o tempo e custo, aumentar a produtividade e principalmente a qualidade dos seus produtos (CURIEL *et al.*, 2011). Com o objetivo de apoiar a implementação de MPS, várias iniciativas têm sido conduzidas para desenvolver e aprimorar *frameworks* de melhores práticas de desenvolvimento de software, tais como o CMMI (SEI, 2006) e o MPS.BR (SOFTEX, 2021) (MONTONI E ROCHA, 2011). Apesar de, ter crescido nos últimos anos, a adoção de normas e modelos de referência para MPS, a quantidade de organizações que adotam esses modelos ainda é pequena em relação ao total de organizações de software (STAPLES *et al.*, 2007).

Estudos mostram as razões para entender essa diferença e apontam para o alto custo e à burocracia de recursos demandados para execução dos processos (STAPLES et al., 2007; COLEMAN e O'CONNOR, 2008). Outros fatores mostram a atitude dos indivíduos, por exemplo, a falta de motivação e resistência a mudança pelos colaboradores das organizações e pela falta de apoio e comprometimento da alta direção da organização (BADDOO, 2001; NIAZI et al., 2006). Portanto, a implementação de MPS é uma atividade complexa e repleta de conhecimento que depende de aspectos de caráter sócio-cultural, tecnológico e organizacional (MONTONI E ROCHA, 2011).

David Rico (2004) apresenta em seus estudos uma série de seis métricas para demonstrar os benefícios ao implementar MPS: (i) *Benefit* – benefício é a quantidade de dinheiro que resulta de um método; (ii) *Cost* – custo é a quantidade de dinheiro que é gasto em um método; (iii) *Benefit/Cost Ratio* – benefício / custo é simplesmente a relação de benefícios a custo; (iv) *ROI – ROI* é a quantidade de dinheiro que se ganha depois de passar uma quantidade de dinheiro; (v) *Net Present Value (NPV)* – valor presente líquido é o que o dinheiro vale no futuro; (vi) *Breakeven Point* – o ponto de equilíbrio é o valor numérico

no qual os benefícios ultrapassar ou exceder os custos. Os modelos utilizados nos estudos incluem *Inspection*, *PSP*[sm], *TSP*[sm], *SW-CMM*®, *ISO 9001, e CMMI*® (RICO, 2004).

Para se compreender mais sobre as abordagens de melhoria de processo de software, torna-se necessário observar as características dos processos de software mais utilizados no mercado. Não sendo excludente, buscou-se apresentar os processos de software mais tradicionais, até aos processos de software que trazem abordagens da metodologia ágil. O que fica bem claro e óbvio, é que todos os processos possuem atividades e, de senso comum, as atividades são exercidas por pessoas ou ferramentas, e que por sua vez, representam um esforço ao executar tal atividade. Essa representatividade de esforço, uma vez valorado, direciona para a composição de um orçamento monetário, uma proposta comercial.

A proposta comercial, uma vez aceita pelo cliente, contextualiza um projeto de software. Uma vez concebido o projeto de software, precisa de ser executado com maestria e competência. Pois agora, o projeto sendo executado, todas as despesas se transformam em custos que precisam de ser registrados contabilmente.

Com o entendimento do verdadeiro valor dos processos organizacionais. Pensar em melhoria de processo de software fica mais fácil, pois além de conhecer os benefícios que podem trazer para a organização de forma tangível, não se pode esquecer os benefícios intangíveis que refletem o impacto na imagem da organização e principalmente no principal ativo da organização, que são as pessoas envolvidas nos processos. Para justificar todo esse investimento, os parágrafos a seguir demonstram as principais iniciativas.

A Metodologia ROI está baseada em cinco componentes que constituem os fundamentos e os blocos constituintes de sua estrutura de avaliação: *Framework* de Avaliação; Modelo de Processo; Prática e Casos de Aplicação; Filosofia e Padrões de Operação; e Implementação. Apresenta grandes resultados quando aplicados à Gestão de

[sm] *Personal Software Process*, PSP, *Team Software Process*, TSP, CMMI são marcas da Carnegie Mellon University.

pessoas e Treinamentos e Desenvolvimentos organizacionais, é flexível, é adaptativa e pode ser aplicada em diversos cenários, inclusive para projetos de TI (PHILLIPS, 2007).

Solingren (2004) apresenta alguns fatores relevantes para analisar a importância do ROI de MPS:

- Convencer gerentes a investir dinheiro e esforço, e convencê-los de que MPS pode ajudar a resolver problemas estruturais;
- Estimar o quanto de esforço se deve investir para resolver um determinado problema ou estimar se determinados benefícios pretendidos valem o seu custo;
- Decidir qual a melhoria de processos para implementar em primeiro lugar. Muitas organizações devem priorizar os processos, devido ao calendário e restrições de recursos;
- Programas de melhoria contínua, orçamentos de MPS são atribuídos e discutidos anualmente, os benefícios devem ser explícitos e as organizações devem mostrar um ROI suficiente, ou a continuação está em risco;
- Sobreviver, porque qualquer investimento em uma organização deve ser questionado sob o seu retorno. Caso contrário, o dinheiro provavelmente será desperdiçado e corre o risco de falência, em longo prazo.

Calcular o custo e benefícios é um pré-requisito para a tomada de decisão em investimento. Isso é tão verdadeiro para MPS como para qualquer outro investimento (SOLINGEN, 2004).

No Brasil, iniciativas em MPS já demonstravam uma preocupação com a eficácia e os possíveis benefícios encontrados em MPS para as organizações brasileiras (SPINOLA, 2004). Implementação de MPS no processo de Engenharia de Requisitos, realizados em um grupo de empresas do Porto Digital[1], as organizações identificaram os seguintes

[1] O Porto Digital é resultado do ambiente de inovação que se consolidou em Pernambuco nas últimas décadas: *http://www.portodigital.org/*

aspectos de expectativas de ROI: aumento da produtividade; satisfação dos clientes; redução de *bugs* e o aumento da qualidade do processo e do produto (ALVES, 2007).

Os resultados extraordinários apresentados nos últimos de 10 anos de implementações de MPS.BR e CMMI durante o período de 2004-2013 no Brasil, totalizam 804 avaliações onde 538 MPS com 67% e 266 CMMI com 33%. Dessa forma, as avaliações de MPS e CMMI posiciona o Brasil em 4º lugar no ranking mundial de países com melhor qualidade nos seus processos de software, ficando somente atrás da China, Estados Unidos e Índia (WEBER, 2014).

Perante estes factos, coloca-se a questão de como auxiliar a tomada de decisão dos executivos das organizações de software, o que justifica investir em MPS, pois trará benefícios tangíveis e intangíveis. É que uma das principais preocupações desses executivos é procurar saber quando haverá o retorno sobre o investimento (*Return on Investment – ROI*). A ausência de um método formal de estimativa de ROI para MPS direcionou o estudo para definir uma metodologia para identificar os aspectos que podem influenciar o ROI em iniciativas de melhoria de processos de software. Os métodos tradicionais de avaliação econômica de investimentos são insuficientes, pois não contemplam uma das principais características da área de Tecnologias de Informação (TI) que são os benefícios intangíveis.

1.1.1 DEFINIÇÃO DO PROBLEMA

Uma organização desenvolvedora de software, só se mantem no mercado se conseguir controlar um fator muito importante que é o custo. Pois utiliza como matéria prima para esse desenvolvimento, tecnologias de alto nível e que, no início se apresentam como um investimento, mas ao longo do tempo se transformam em custo para manter. Outra matéria prima extremamente importante, é o intelecto dos colaboradores, pessoas capacitadas que exercerão funções importante na organização. Entre estes custos de alto valor para a organização, pode-se pensar que não são custos e sim investimentos. Pois sem

esses recursos, a organização não será competitiva no mercado, não produzirá bons produtos de software, que atendam as necessidades demandadas pelo mercado.

Além disso, para garantir uma melhor atratividade de seus produtos/serviços pelos clientes, uma boa gestão dos processos organizacionais, poderá oferecer um preço desses produtos/serviços, de forma mais equilibrada ao mercado. Evitando assim, prejuízo nos investimentos.

Por características de operações de mercado, as organizações de software direcionam os seus esforços de desenvolvimento de produtos, para determinados nichos de mercado, ou no pior caso, apontam para várias direções. Isso gera um grande esforço para garantir a qualidade dos seus produtos. Diante desse cenário, as organizações precisam identificar problemas durante a execução de projetos de software, e através de gestão de métricas, possam auxiliar na melhor condução dos projetos e ainda, garantir a qualidade necessária para satisfazer seus clientes. Caso sejam identificados resultados não satisfatórios nas medições, será necessário aplicar melhorias nos processos que apresentarem problemas para alcançar melhores níveis de resultados.

Evidentemente que existem vários aspectos que influenciam a implementação de MPS, já mencionados na seção 1.1. Mas para o proprietário da organização ou investidor, precisa ser informado quanto vai custar todo o investimento em melhorar os processos organizacionais. Uma forma de convencimento é apresentar os benefícios, através de um *benchmarking*, com resultados de melhores práticas e quais foram os principais benefícios alcançados. Apresentado tais resultados, junto com o cenário contextualizado da organização, pode-se planejar um investimento mais seguro, com expectativas mais alcançáveis.

1.1.2 QUESTÕES DE PESQUISA

Sabe-se que o ROI em MPS tem sido pesquisado e difundido pela academia e indústria em diversas publicações após a implementação de melhoria (EMAN, 2003),

(SOLINGEN, 2004), (RICO, 2004), (RICO, 2006), (ALVES, 2007), (FERREIRA *et al*., 2007), (PHILIPPS, 2007) (TRAVASSOS, G. H. e KALINOWSKI, M., 2012). Mas surge uma questão: Quão satisfeito ficará a organização de software do Brasil ao estimar o retorno de investimento em iniciativas de melhoria de processos de software, mesmo conhecendo os fatores críticos de sucesso, por meio da utilização de método de estimativa.

1.1.3 HIPÓTESE

A definição da hipótese para essa tese baseia-se que ao utilizar o *SPIREM-OBK*. A estimativa do retorno de investimento para iniciativas de melhorias de processos de software será satisfatória para as organizações de software?

1.1.4 OBJETIVOS

1.1.4.1 OBJETIVO GERAL

O Para validar a hipótese definida acima, o objetivo geral deste trabalho de pesquisa é criar um modelo teórico de estimativa preditivo considerando os fatores críticos de sucesso em iniciativas de melhoria de processos de software visando responder quando ocorrerá o retorno de investimento no contexto do setor de software do Brasil, denominado *SPIREM-OBK – Software Process Improvement ROI Estimate Model Oriented by Knowledge*. Com o intuito de:

Analisar resultados que demonstrem o ROI de iniciativas de melhoria de processos de software.

Com o propósito de comparar os resultados alcançados dessas iniciativas de melhoria de processos de software.

Com relação às estimativas de ROI dessas iniciativas realizadas pelo SPIREM.

Do ponto de vista do pesquisador.

No contexto acadêmico e industrial com foco em iniciativas de melhoria de processos de software.

1.1.4.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos desse trabalho da pesquisa, muito alinhados com o percurso a ser realizadado, são:

- Objetivo 1: Realizar uma revisão da literatura sobre o tema para definir o estado da arte.
- Objetivo 2: Realizar uma pesquisa secundária através de uma revisão sistemática para verificar a existência de modelos de estimativas de ROI em melhoria de processo de software.
- Objetivo 3: Definir o *SPIREM-OBK – Software Process Improvement ROI Estimate Model Oriented by Knowledge*.
- Objetivo 4: Aplicar o *SPIREM-OBK* em um estudo de caso experimental em projetos de melhoria de processo de software.
- Objetivo 5: Analisar e comparar os resultados coletados.

1.1.5 CONTEXTO

A pesquisa será iniciada com uma revisão da literatura para definir o estado da arte sobre o tema. Em seguida será realizada uma pesquisa secundária através da revisão sistemática para identificar a unicidade da pesquisa sobre o tema. Feito isso, será realizada um estudo de caso experimental em iniciativas de melhoria de processo de software em organizações.

1.2 CONTRIBUIÇÕES/RESULTADOS ESPERADOS

Uma das principais contribuições desse trabalho, será o conhecimento gerado sobre os processos organizacionais, um entendimento maior da importância de utilização de métricas factíveis para direcionar o crescimento da organização. Incentivar e amadurecer a filosofia de melhoria contínua na organização. Com a utilização do *SPIREM-OBK*, a organização aprimorará, com ajustes finos nos parâmetros de fatores de impactos, proporcionando uma análise crítica mais adequada para o direcionar o investimento, mediante apresentação dos resultados simulados. Além disso, possibilitará a comparação entre projetos similares e a própria disseminação dos resultados dos experimentos realizados durante a pesquisa.

1.3 ESTRUTURA DO TRABALHO

A organização desta tese possui a seguinte estrutura: Este Capítulo Introdutório. O Capítulo 2 – Processo de Desenvolvimento de Software e Custos Associados: Descreve a fundamentação teórica relacionada a Processos de Desenvolvimento de Software, os Ciclo de Vida de Desenvolvimento de Software, metodologias ágeis, principais abordagens sobre Precificação, Custos relacionados ao desenvolvimento de Software e Aspectos sobre Capitalização dos custos na Contabilidade.

O Capítulo 3 – Estimativas, Melhoria de Processo de Software e ROI: Apresenta a fundamentação teórica sobre Medições e Estimativas de Software e principalmente sobre Melhoria de Processo de Software, os modelos de Maturidade e Capacidade CMMI e MPS.BR, explica-se sobre o principal tema em questão que é o ROI, além disso, as iniciativas como Soluções Alternativas e Trabalhos Relacionados para medir o ROI em MPS.

Capítulo 4 – Metodologia: Descreve a fundamentação teórica relacionada a Revisão Sistemática que apresenta estudos relacionados com o tema e a estrutura definida para o Estudo de Caso.

O Capítulo 5 – *SPIREM-OBK*: Descreve o modelo conceitual do *SPIREM-OBK*, a descrição dos Fatores e as suas características, e como calcular o ROI. O Capítulo 6 – Análise Crítica dos Resultados: apresenta as evidências providas da execução do Estudo de Caso em uma organização de software e análise dos dados. Por último, o Capítulo 7 – Apresenta a conclusão do trabalho e novos desafios. O trabalho é encerrado com a listagem das referências bibliográficas utilizadas ao longo do texto e dois apêndices que mostram, respetivamente os dados associados com o uso e cálculos do caso de estudo e a lista de artigos associados com a revisão da literatura.

1.4. CONSIDERAÇÕES FINAIS

Este capítulo introdutório apresentou a motivação, a questão de pesquisa para construir a hipótese, os principais objetivos deste trabalho e descreve o contexto da pesquisa. Apresentou-se a importância de entender sobre os processos de desenvolvimento de software, as suas atividades e os seus custos e características que contextualizam o problema relacionado com a estimativa de retorno de investimento em iniciativas de melhoria de processo de software. Em seguida, as principais contribuições e os resultados esperados e por último a estrutura deste documento.

Capítulo 2

Processo de Desenvolvimento de Software e Custos Associados

Neste capítulo são apresentadas as principais características do arcabouço do processo de software, observar o comportamento e a flexibilidade dos ciclos de vida mais tradicionais e metodologias ágeis de desenvolvimento de software. Entender como é complexo estruturar um bom preço para o software pois garante a subsistência da organização desenvolvedora de software. Identificar os fatores que impactam nos custos durante o planejamento e execução de projetos de software. Bem como, registrar todas essas despesas na Contabilidade da organização.

2.1 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

Segundo o *IEEE – Institute of Electrical and Electronics Enginners*, define processo de desenvolvimento de software, como sendo uma sequência de passos executados para um dado propósito (IEEE, 1990). Segundo Roger Pressman (2016), define processo de desenvolvimento de software, como um arcabouço para as tarefas que são necessárias para construir softwares de alta qualidade (PRESSMAN, 2016). Para Ian Sommerville, o processo de desenvolvimento de software é um conjunto de atividades e resultados associados que produzem um produto de software (SOMMERVILLE, 2011).

Para Humphrey, processo de desenvolvimento de software é definido como "*Um conjunto de ferramentas, métodos e práticas usadas para produzir um produto de software*" (HUMPHREY, 1989). Pode-se concluir que um Processo de Desenvolvimento de Software precisa de insumos como entrada que, para serem processados por um método ou ferramenta que terá como saída um produto ou parte dele, como componentes de software.

Normalmente é exemplificado por um arcabouço (*framework*) de processo, que estabelece a base para o processo de desenvolvimento de software completo, pela identificação de atividades aplicáveis aos projetos de software. Além disso, o arcabouço de processo possui um conjunto de subatividades, chamadas de atividades guarda-chuva, que são aplicáveis durante todo o processo de desenvolvimento de software (PRESSMAN, 2016). Para cada atividade do arcabouço de processo de software é relacionada por um conjunto de ações de engenharia de software – uma coleção de tarefas relacionadas que produz um produto importante de trabalho de engenharia de software. Cada ação é preenchida por tarefas de trabalho individuais que realizam alguma parte do trabalho determinado pela ação.

Os modelos de processo de desenvolvimento de software podem ser caracterizados dentro do Arcabouço de Processo, sendo uma prática inteligente de aplicação, adaptações são essenciais para o sucesso de uma nova definição de processo. O processo deve ser

avaliado para garantir que satisfaça a um conjunto de critérios básicos de processo, que se demonstraram essenciais para engenharia de software.

A família de normas ISO/IEC 330xx substitui e amplia algumas partes da ISO/IEC 15504 e estabelece um *framework* para melhorar a qualidade dos processos da organização alinhado aos objetivos de negócio, define um perfil de processo como um conjunto de atributos de processos e um processo de avaliação disciplinado para avaliar o processo da unidade organizacional em relação a um modelo de avaliação de processos.

A Figura 2.1 apresenta como os processoes estão organizados. No contexto de melhoria de processos, a avaliação de processos segundo a ISO/IEC 33002:2015[2] fornece meios para caracterizar a prática corrente, dentro de uma unidade organizacional, em termos de capacidade de um determinado processo para responder às necessidades. As análises dos resultados na perspectiva das necessidades de negócio da organização identificam forças, fraquezas e riscos inerentes aos processos. Estas, por sua vez, conduzem à possibilidade de determinar se os processos são efetivos no alcance dos seus objetivos, e na identificação das principais causas de qualidade fraca, ou desvios no tempo ou nos custos. Estas evidências fornecem os indicadores para o estabelecimento de prioridades de melhorias nos processos (ISO 33001, 2015).

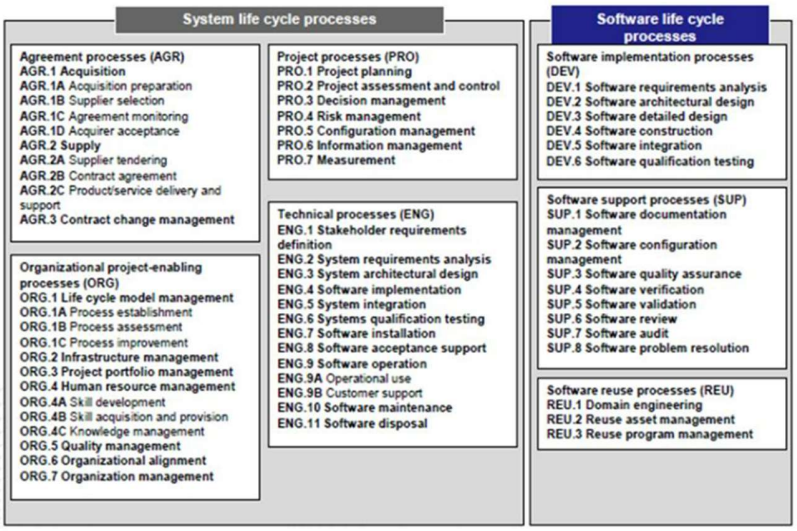


Figura 2.1. ISO IEC 330x. (ISO 33001, 2015).

[2] Versão revista e confirmada em 2020.

A determinação de capacidade dos processos está relacionada com a análise dos perfis de capacidade propostos para os processos selecionados versus o perfil das capacidades a atingir. Esta relação permite identificar os riscos envolvidos na gestão de um projeto usando os processos. A capacidade proposta pode ser baseada nos resultados de avaliações anteriores, ou pode ser baseada nos resultados de avaliações executadas com o objetivo de medir a capacidade proposta que servirá como um parâmetro atual.

2.2 Modelos de Ciclo de Vida de Desenvolvimento de Software

O ciclo de vida é a estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, contemplando o ciclo de vida do sistema, desde a concepção dos seus requisitos até a descontinuidade de utilização. O modelo de ciclo de vida é a primeira escolha a ser feita no processo de desenvolvimento de software. A partir desta escolha definir-se-á desde a maneira mais adequada de obter as necessidades do cliente (levantamento de requisitos), até quando e como o cliente receberá sua primeira versão operacional do sistema.

Processo de software é o conjunto de atividades que constituem o desenvolvimento de um sistema computacional. Estas atividades são agrupadas em fases, como: definição de requisitos, análise, projeto, desenvolvimento, teste e implantação. Em cada fase são definidas, além das suas atividades, as funções e responsabilidades de cada membro da equipe, os produtos de trabalhos que são produzidos, os artefatos (PRESSMAN, 2016).

O que diferencia um processo de software de outro é a ordem em que as fases vão ocorrer, o tempo e a ênfase dado a cada fase, as atividades presentes, e os produtos de trabalhos entregues. Com o crescimento do mercado de software, houve uma tendência a repetirem-se os passos e as práticas que deram certo, denominados como melhores práticas. A etapa seguinte foi a formalização em modelos de ciclo de vida. Em outras palavras, os modelos de ciclo de vida são o esqueleto, ou as estruturas pré-definidas nas quais encaixamos as fases do processo.

Não existe uma “bala de prata” e muito menos uma solução universal, a característica e complexidade do negócio do cliente, o tempo disponível, o custo, a equipe,

o ambiente operacional são fatores que influenciarão diretamente na escolha e definição do ciclo de vida de software a ser adotado. Considerando esses fatores, também é difícil uma empresa adotar um único ciclo de vida. Na maioria das organizações, utilizam mais de um ciclo de vida.

2.2.1 MÉTODOS TRADICIONAIS

Os ciclos de vida se comportam de maneira sequencial (fases seguem determinada ordem) e/ou incremental (divisão de escopo) e/ou iterativa (retroalimentação de fases) e/ou evolutiva (software é aprimorado). Segue alguns mais conhecidos:

- Cascata;
- Modelo em V;
- Incremental;
- Evolutivo;
- RAD;
- Prototipagem;
- Espiral;
- Modelo de Ciclo de Vida Associado ao RUP.

2.2.1.1 MODELO EM CASCATA

Formalizado por Royce em 1970, é o modelo mais antigo. Suas atividades fundamentais são:

- análise e definição de requisitos;
- projeto;
- implementação;
- teste;
- integração.

O modelo em cascata tem o grande mérito de ser o primeiro a impor o planejamento e o gerenciamento ao processo de software, que antes era casual. O nome “cascata” foi

atribuído em razão da sequência das fases, onde cada fase só começa quando a anterior termina; e da transmissão do resultado da fase anterior como entrada para a fase atual (o fim de cada fase resulta em um documento aprovado). Nesse modelo, portanto, é dada muita ênfase às fases de análise e projeto antes de partir para a programação, a fim de que o objetivo do software esteja bem definido e que sejam evitados retrabalhos, conforme podemos observar na Figura 2.2 (PRESSMAN, 2016).

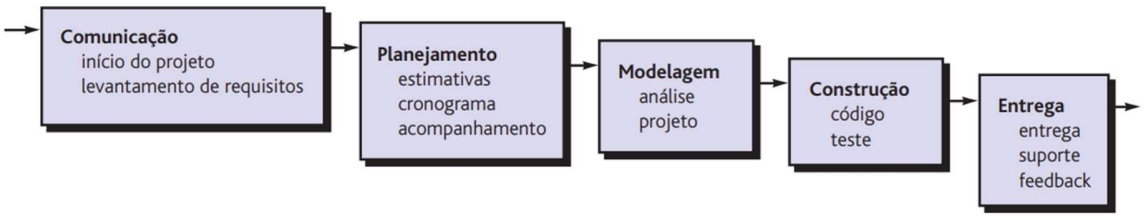


Figura 2.2. Ciclo de Vida Cascata. Fonte (PRESSMAN, 2016).

2.2.1.2 MODELO EM V

Neste modelo, do Ministério de Defesa da Alemanha, 1992, o modelo em cascata é colocado em forma de "V". Do lado esquerdo do V ficam da análise de requisitos até o projeto, a codificação fica no vértice e os testes, desenvolvimento, implantação e manutenção, à direita, conforme Figura 2.3.

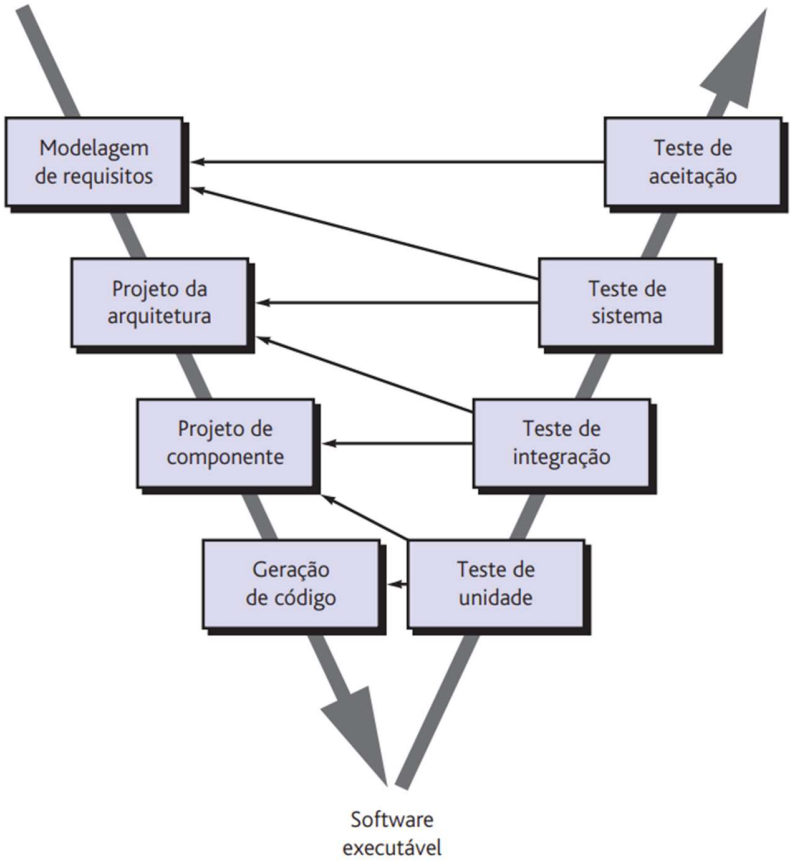


Figura 2.3. Modelo V. Fonte (PRESSMAN, 2016).

A característica principal desse modelo, que o diferencia do modelo em cascata, é a ênfase dada à verificação e validação: cada fase do lado esquerdo gera um plano de teste a ser executado no lado direito. Mais tarde, o código fonte será testado, do mais baixo nível ao nível sistêmico para confirmar os resultados, seguindo os respectivos planos de teste: o teste de unidade valida o projeto do programa, o teste de sistema valida o projeto de sistema e o teste de aceitação do cliente valida a análise de requisitos. Da mesma forma que o modelo em cascata, o cliente só recebe a primeira versão do software no final do ciclo, mas apresenta menos risco, devido ao planejamento prévio dos testes nas fases de análise e projeto.

2.2.1.3 MODELO INCREMENTAL

Neste modelo, de Mills em 1980, os requisitos do cliente são obtidos, e, de acordo com a funcionalidade, são agrupados em módulos. Após este agrupamento, a equipe, junto ao cliente, define a prioridade em que cada módulo será desenvolvido, escolha baseada na importância daquela funcionalidade ao negócio do cliente. Cada módulo passará por todas as fases "cascata" de projeto, conforme se observa na Figura 2.4, e será entregue ao cliente um software operacional. Assim, o cliente receberá parte do produto em menos tempo (PRESSMAN, 2016).

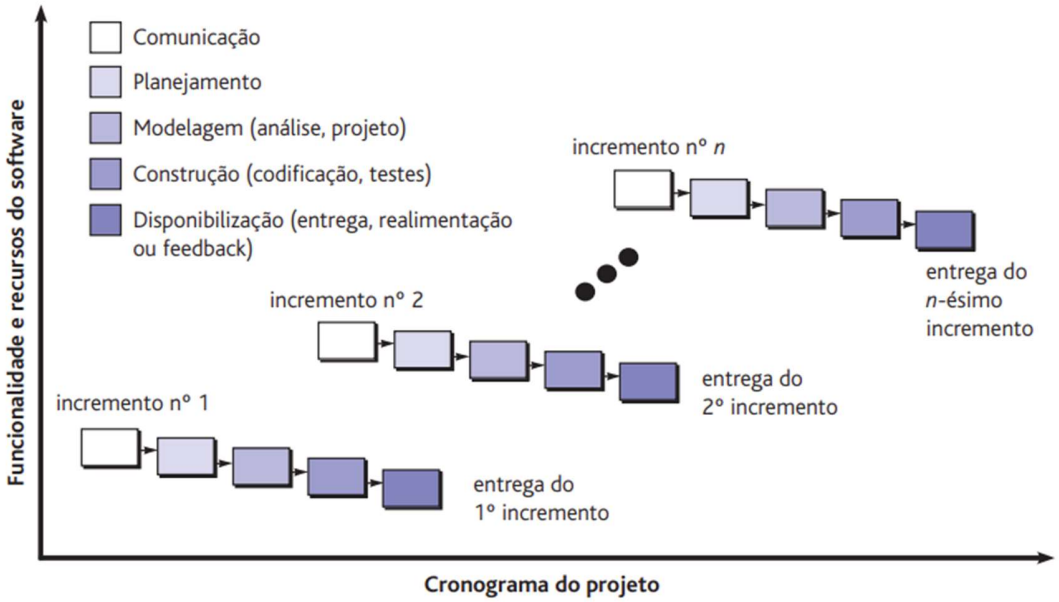


Figura 2.4. Modelo Incremental. Fonte (PRESSMAN, 2016).

Como o cliente já trabalhará no primeiro incremento ou módulo, é muito importante que haja uma especial atenção na integração dos incrementos, o que exige muito planejamento, afinal não é aceitável que o cliente se depare com muitos erros de software a cada incremento, tampouco, que a cada incremento ele precise de se readaptar a grandes mudanças. Uma atenção especial deve ser dada ao agrupamento dos requisitos e à qualidade no desenvolvimento das funções comuns a todo o sistema, que inevitavelmente deverão ser entregues no primeiro incremento.

Desta forma, além de atender as necessidades mais críticas do cliente mais cedo, as partes mais importantes serão, também, as partes mais testadas no ambiente real. Será mais difícil gastar recursos em conceitos errados, ou que um mau entendimento dos requisitos alcance uma escala difícil de ser ajustada, visto que durante todo o projeto haverá o *feedback* do cliente (a opinião do cliente realimenta o sistema).

Esse ciclo de vida não exige uma equipe muito grande, pois a modularização diminui o escopo de cada incremento, e não há um paralelismo nas atividades. Haverá, por outro lado, uma dificuldade em manter a documentação de cada fase atualizada devido às melhorias no sistema e aos ajustes de requisitos solicitados pelos clientes.

2.2.1.4 MODELO EVOLUTIVO

Neste modelo, os requisitos são adquiridos em paralelo à evolução do sistema. O modelo evolutivo parte do princípio de que o cliente não expõe todos os requisitos, ou os requisitos não são tão bem conhecidos, ou os requisitos ainda estão sofrendo mudanças. Desta forma, a análise é feita em cima dos requisitos conseguidos até então, e a primeira versão é entregue ao cliente. O cliente usa o software no seu ambiente operacional, e como *feedback*, esclarece o que não foi bem entendido e dá mais informações sobre o que precisa e sobre o que deseja (ou seja, mais requisitos) (PRESSMAN, 2016).

A partir deste *feedback*, nova análise, projeto e desenvolvimento são realizados, e uma segunda versão do software é entregue ao cliente que, novamente, retorna com mais *feedback*. Assim, o software vai evoluindo, se tornando mais completo, até atender todas as necessidades do cliente dentro do escopo estabelecido. Tem-se assim a versão final, pelo menos até novos requisitos aparecerem, apresentado na Figura 2.5.

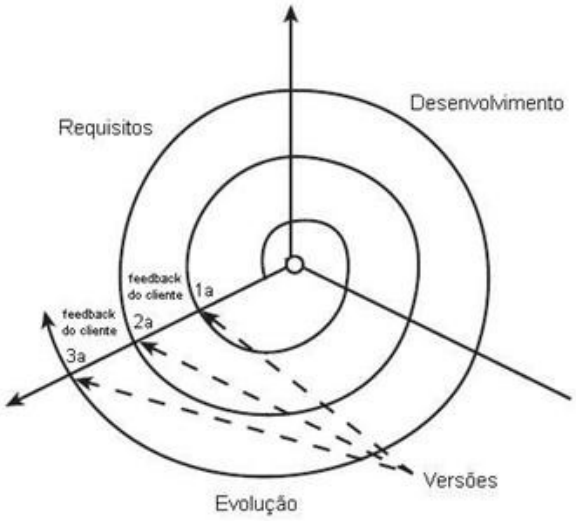


Figura 2.5. Ciclo de vida Evolutivo. Fonte (*https://www.devmedia.com.br/introducao-aos-processos-de-software-e-o-modelo-incremental-e-evolucionario/29839*)

A participação constante do cliente é uma grande vantagem desse modelo, o que diminui o risco de má interpretação de requisitos dos modelos que só oferecem a primeira versão do software no final do processo. Da mesma forma, o software já atende algumas necessidades do cliente muito mais cedo no processo. Não é dada muita ênfase à documentação, pois a geração de versões torna este trabalho muito árduo. Além disso, como a análise de requisitos e desenvolvimento estão sempre acontecendo, a preocupação em documentar todo o processo pode fazer com que haja atrasos na entrega.

Há uma alta necessidade de gerenciamento nesse tipo de modelo, pois a falta de documentação adequada, o escopo de requisitos não determinado, o software crescendo e estando ao mesmo tempo em produção podem ter consequências negativas. Seguem alguns exemplos: o sistema nunca terminar, pois o cliente sempre pede uma alteração; o sistema não ter uma estrutura robusta a falhas nem propícia a uma fácil manutenção, pelas constantes alterações; o cliente mudar de ideia radicalmente entre uma versão e outra ou revelar um requisito que exija uma versão bem diferente da anterior, fazendo com que toda a base (de dados ou de programação) precise de ser revista.

Os citados problemas podem implicar em um grande ônus financeiro e de tempo. É muito importante que o cliente esteja ciente do que se trata este ciclo de vida e que sejam esclarecidos os limites de escopo e de tempo, para que não haja frustrações de expectativas.

2.2.1.5 RAD – "*RAPID APPLICATION DEVELOPMENT*"

Este modelo formalizado por James Martin em 1991, como uma evolução da "prototipagem rápida", destaca-se pelo desenvolvimento rápido da aplicação. O ciclo de vida é extremamente comprimido, de forma a encontrarem-se exemplos, na literatura, de duração de 60 e 90 dias. É ideal para clientes buscando lançar soluções pioneiras no mercado. É um ciclo de vida incremental, iterativo, onde é preferível que os requisitos tenham escopo restrito. A diferença principal do ciclo anterior é o forte paralelismo das atividades, requerendo, assim, módulos bastante independentes. Aqui os incrementos são desenvolvidos ao mesmo tempo, por equipes diferentes (PRESSMAN, 2016).

Além do paralelismo, a conquista do baixo tempo se dá graças à compressão da fase de requisitos e da fase de implantação. Isso significa que, na obtenção dos requisitos, costumam-se optar por metodologias mais dinâmicas e rápidas, como *workshops* ao invés de entrevistas. Permite-se também um desenvolvimento inicial no nível mais alto de abstração dos requisitos visto o envolvimento maior do usuário e visibilidade mais cedo dos protótipos, apresentado na Figura 2.6.

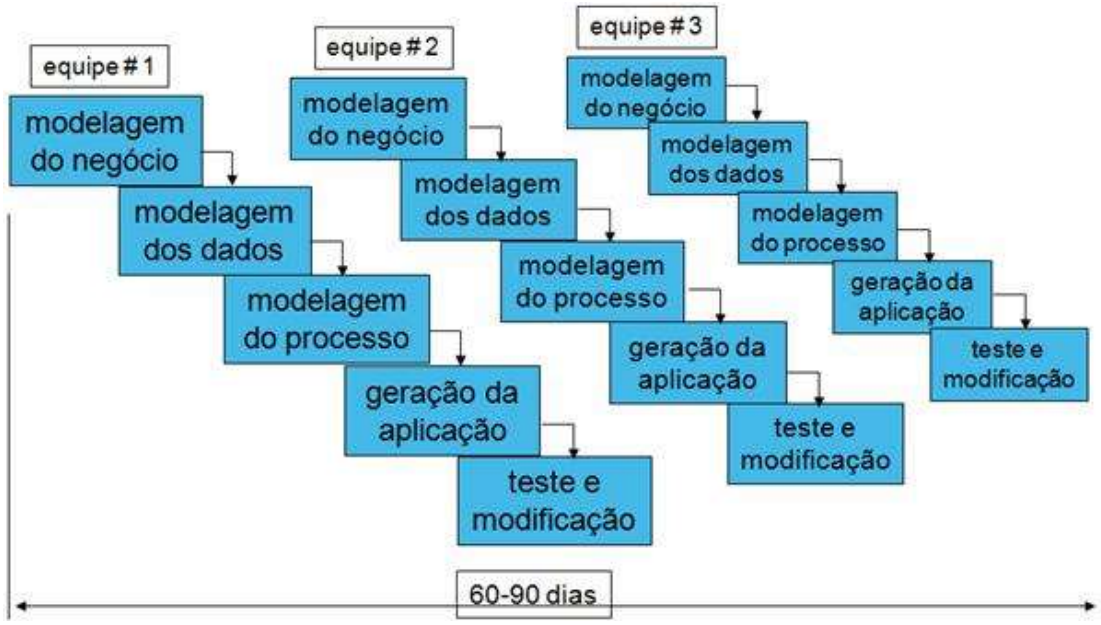


Figura 2.6. Ciclo de vida RAD. Fonte (*https://metodologiasclassicas.blogspot.com/p/incremental.html*)

As fábricas de software que resolvem por adotar este modelo devem ter uma estrutura prévia diferencial de pessoas e ferramentas, tais como:

- Pessoas:
  - Profissionais experientes (funcional e gerência);
  - Profissionais de rápida adaptação;
  - Equipes de colaboração mútua;
  - Maior quantidade de pessoas;
- Gerenciamento:
  - Empresas pouco burocráticas que encorajem a eliminação de obstáculos;
  - Alto controle do tempo;
- Uso de Ferramentas:
  - CASE;
  - Muita diagramação;

- Uma biblioteca prévia de componentes reutilizáveis (APIs, *wizards*, *templates*,...);
- Fácil manutenção (por exemplo: linguagens de programação que suportem Orientação a Objetos, tratamento de exceção, ponteiros);
- Adoção de ferramentas maduras, pois não há tempo de atualizar versões e tratar erros inesperados;

Os sistemas desenvolvidos no ciclo RAD tendem a ter uma padronização de telas muito forte, devido a bibliotecas reutilizáveis e *templates*, porém tendem a perder em desempenho do sistema e na análise de risco (atividades estas que demandam tempo em qualquer projeto). Assim, é preferível o seu uso para software de distribuição pequena.

2.2.1.6 PROTOTIPAGEM

Prototipagem é a construção de um exemplar do que foi entendido dos requisitos capturados do cliente. Pode ser considerado um ciclo de vida ou pode ser usado como ferramenta em outros ciclos de vida. Um protótipo em engenharia de software pode ser o desenho de uma tela, um software contendo algumas funcionalidades do sistema. São considerados operacionais (quando já podem ser utilizados pelo cliente no ambiente real, ou seja, em produção), ou não operacionais (não estão aptos para serem utilizados em produção). Os protótipos podem ser descartados, ou reaproveitados para evoluírem até a versão final (PRESSMAN, 2016).

No ciclo de vida de prototipagem, não é exigido um conhecimento aprofundado dos requisitos num primeiro momento. Isso é bastante útil quando os requisitos não são totalmente conhecidos, são muitos complexos ou confusos. Desta forma, se o cliente não sabe expressar o que deseja (o que ocorre bastante quando não é um sistema legado), a melhor maneira de evitar que se perca tempo e recursos com uma má interpretação é a construção de modelos, ou seja, de protótipos do que o software faria (PRESSMAN, 2016).

Assim, o cliente experimentará, na prática, como o sistema ou parte dele funcionará. A partir desse primeiro contato, o cliente esclarece o que não foi bem interpretado, aprofunda alguns conceitos e até descobre um pouco mais sobre o que realmente precisa. A partir deste *feedback*, novos requisitos são colhidos e o projeto ganha maior profundidade. Outro protótipo é gerado e apresentado ao cliente, que retorna com mais *feedback*. Ou seja, o cliente participa ativamente do início ao fim do processo, apresentado na Figura 2.7.

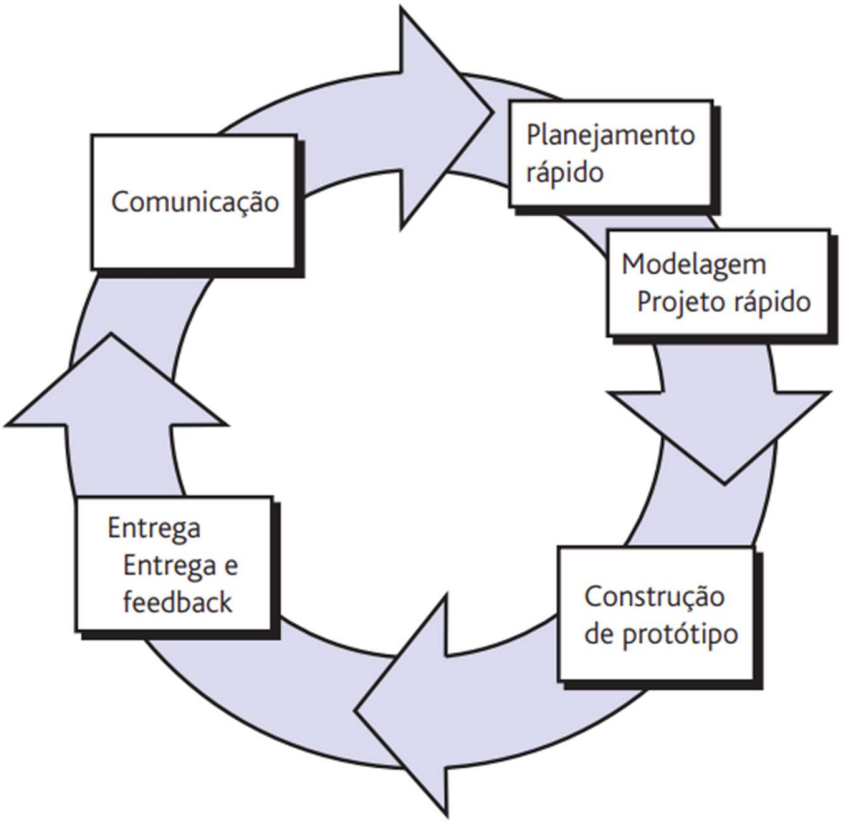


Figura 2.7. O modelo e prototipagem (PRESSMAN, 2016).

A geração de protótipos pode ser facilitada por ferramentas geradoras de telas, de relatórios, poupando esforço de programação e diminuindo o tempo de entrega. Cada protótipo tem uma finalidade diferente. Um protótipo pode servir para esclarecer dúvidas sobre uma rotina, demonstrar a aparência das telas, conteúdo de tabelas, formato de relatórios. Os protótipos podem também ser utilizados para apresentar opções ao cliente

para que ele escolha a que mais lhe agrade, como opções de navegação, de fluxo de telas, entre outras.

Por isso, é muito importante explicar previamente ao cliente que protótipos são apenas modelos para melhorar a comunicação. Caso contrário, pode causar uma frustração por não funcionar corretamente, ter funções limitadas, ter resposta lenta, ou a aparência ruim. Certamente um protótipo construído para esclarecer uma rotina provavelmente terá uma "cara feia"; para demonstrar a aparência das telas, não terá funcionalidade; para apresentar o formato dos relatórios, os dados não serão coerentes.

O cliente fará comparações entre o sistema final e o que foi "prometido" através do protótipo e pode ficar insatisfeito. Por exemplo, geralmente o protótipo não acessa rede ou banco de dados, pois as informações são "desenhadas" com a tela, fazendo com que tudo fique muito rápido. Já no ambiente operacional haverá uma degradação de desempenho e o cliente pode se decepcionar.

Faz parte de um bom gerenciamento no modelo de prototipagem planejar se, quais e que funções dos protótipos não operacionais serão reaproveitadas na versão operacional, para que a sua confecção siga as boas práticas de engenharia de software. Os protótipos não operacionais são construídos com pouca qualidade em prol da velocidade de entrega. Ou seja, não há preocupação na programação, em refinar o código, em usar comentários, em aproveitar eficientemente os recursos de hardware e software, na manutenção, no reuso de componentes e na integração com outras funções ou sistemas. Com certeza será um problema se a equipe sucumbir à pressão do cliente, cada vez mais ansioso para ver a versão final daquele trabalho, e transformar à revelia, protótipos não operacionais em operacionais.

O gerente também deve se preocupar com o escopo do projeto versus a quantidade de protótipos, para que não se perca muito tempo nesse processo, tampouco se transforme num processo de "tentativa e erro". Não é uma tarefa fácil documentar o modelo de ciclo

de vida baseado na prototipagem devido aos requisitos não serem totalmente conhecidos no primeiro momento e a consequente quantidade de mudanças ocorridas.

2.2.1.7 MODELO ESPIRAL

O modelo proposto por Boehm em 1988 trata de uma abordagem cíclica das fases do processo, em que a cada "volta" ou iteração temos versões evolucionárias do sistema. Este é um modelo guiado por risco, suporta sistemas complexos e/ou de grande porte, onde falhas não são toleráveis. Para isso, a cada iteração há uma atividade dedicada à análise de riscos e apoiada através de geração de protótipos, não necessariamente operacionais (desenhos de tela, por exemplo) para que haja um envolvimento constante do cliente nas decisões (PRESSMAN, 2016).

Cada iteração ou volta é dedicada a uma fase do processo de vida de um software (viabilidade do projeto, definição de requisitos, desenvolvimento e teste, ...). Ao mesmo tempo, cada volta é seccionada em 4 setores, da seguinte forma:

- Iteração: Viabilidade do projeto:
    - Definição de objetivos;
    - Avaliação e redução de riscos;
    - Desenvolvimento e validação;
    - Planejamento da próxima fase;
- Iteração: Definição de requisitos do sistema:
    - Definição dos objetivos;
    - Avaliação e redução de riscos;
    - Desenvolvimento e validação;
    - Planejamento da próxima fase;
- Iteração: Projeto do sistema:
    - ...
    - ...
    - ...

- ...
- Iteração: Desenvolvimento e teste de unidade
  - ...
  - ...
  - ...
- Iteração: Implantação
  - ...

Ou, na representação gráfica deste modelo conforme Figura 2.8.

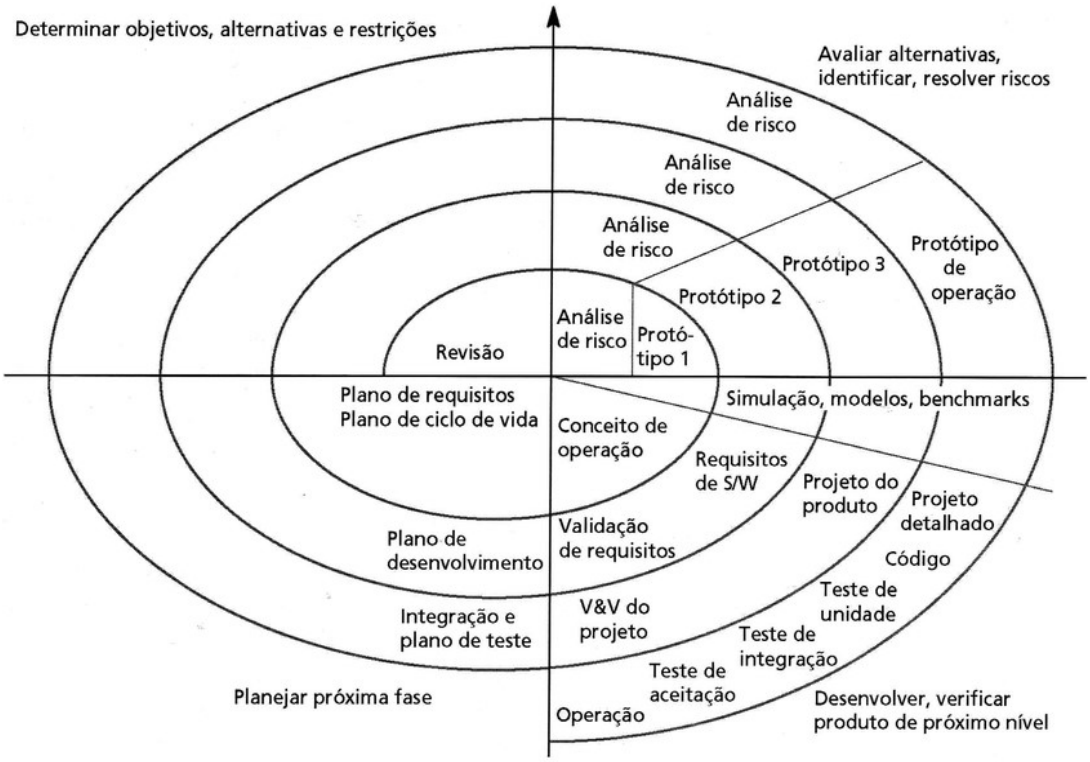


Figura 2.8. Modelo Espiral. Fonte (ÁVILA, 2015).

Os quatro setores são explicados da seguinte forma:

- Na Definição de Objetivos, desempenhos, funcionalidade, entre outros objetivos, são levantados. Visando alcançar esses objetivos são listadas alternativas e restrições, e cria-se um plano gerencial detalhado.
- Na Análise de Riscos, as alternativas, restrições e riscos anteriormente levantados são avaliados. Neste setor (porém não apenas neste) protótipos são utilizados para ajudar na análise de riscos.
- No Desenvolvimento e Validação um modelo apropriado para o desenvolvimento do sistema é escolhido, de acordo com o risco analisado no setor anterior (cascata, interativo, ...).
- No Planejamento da Próxima fase ocorre a revisão do projeto e a decisão de partir para a próxima fase.

Ou seja, cada volta ou iteração do processo é vista por quatro ângulos. No final da Viabilidade do Projeto teremos como resultado a Concepção das Operações; da Definição de Requisitos o produto serão os requisitos; no final do Desenvolvimento e Testes o projeto é criado e os testes habilitados. Pode-se parar por aí, pode-se incluir mais fases, pode a espiral ficar adormecida até uma nova alteração do sistema se requisitada, e desta forma estender até o fim de vida do sistema.

Neste modelo, apenas o início é definido. A evolução e amadurecimento dos requisitos demandam tempo ajustável (assim como custo). Isto torna o sistema difícil de ser vender ao cliente e exige um alto nível de gerenciamento em todo o processo.

2.2.1.8 MODELO RUP

Derivado da UML e do Processo Unificado de Desenvolvimento de Software, o RUP, *Rational Unified Process*, é um modelo de processo iterativo e incremental, dividido em fases, orientado a casos de uso. Possui *framework* (entendido aqui como uma estrutura)

de processo e manuais que guiam na utilização das melhores práticas de especificação de projeto. O objetivo do RUP é produzir software com qualidade (melhores práticas de engenharia de software) que satisfaça as necessidades dos clientes dentro de um prazo e orçamento estabelecidos. Este modelo foi desenvolvido pela *Rational Software Corporation* e adquirido pela IBM, que o define da seguinte maneira: "*IBM Rational Unified Process*®", ou RUP, é uma plataforma de processo de desenvolvimento de software configurável que oferece melhores práticas comprovadas e uma arquitetura configurável, apresentado na Figura 2.9 (PRESSMAN, 2016).

Figura 2.9. Conceitos chaves do RUP.Fonte (QUATRANI, 2001)

O RUP possui quatro fases de negócio. O nome de cada fase revela o que será entregue por ela:

- Concepção: define o escopo do projeto, ou "*business case*"; onde é julgado se o projeto deve ir adiante ou ser cancelado.
- Elaboração: elabora modelo de requisitos, arquitetura do sistema, plano de desenvolvimento para o software e identificar os riscos.

- Construção: constrói o software e a documentação associada.
- Transição: finaliza produto, define-se plano de entrega e entrega a versão operacional documentada para o cliente.

2.2.2. MÉTODOS ÁGEIS

O conceito de desenvolvimento ágil foi proposto em 2001 pela *Agile Team* e, em seguida, vários desenvolvedores de software, equipas e empresas reconheceram e aceitaram o conceito. Então, o seu uso aumentou gradativamente em projetos de software. Os métodos para desenvolvimento ágil de software são um conjunto de práticas que foram criadas por profissionais experientes.

Ficam mais evidentes os problemas relacionados ao processo de desenvolvimento de software, aspectos que estamos discutindo nesse capítulo, como: alto custo, alta complexidade, dificuldade de manutenção, e uma distância grande entre as reais necessidades dos clientes e o produto desenvolvido (SOMMERVILLE, 2011). Estes métodos podem ser vistos como uma reação alternativa, aos métodos tradicionais de desenvolvimento de software, que prioriza uma abordagem racionalizada baseada na engenharia de software, afirmando que os problemas são totalmente especificáveis e que existem soluções ótimas e previsíveis para qualquer tipo de problema.

Acreditando que o processo utilizado é um dos motivos para a ocorrência desses problemas, um segmento crescente da Engenharia de Software vem defendendo a adoção de processos mais simplificados conhecidos como métodos ágeis, que visam a desburocratização das atividades associadas ao desenvolvimento (FOWLER, 2001). Os métodos ágeis têm atraído um grande interesse entre a comunidade de desenvolvimento de software. E, consequentemente, uma considerável quantidade de métodos apresentando características ágeis têm aparecido nos últimos anos.

Para apoiar os profissionais, ligados ao desenvolvimento de software, que possuem interesse na utilização de metodologias ágeis, na escolha de um método ágil que melhor se

adeque às características dos projetos, equipes e/ou organizações, segue as principais utilizadas no mercado:

2.2.2.1. *EXTREME PROGRAMMING* (XP)

Programação extrema (do inglês *eXtreme Programming*), ou simplesmente XP, é considerada uma metodologia ágil e se ajusta bem a projetos de software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software (BECK, 2000). O XP possui algumas características marcantes que são:

- *Feedback* constante. Abordagem incremental. Encoraja a comunicação entre as pessoas envolvidas.

- Os cinco valores fundamentais são: comunicação, simplicidade, *feedback*, coragem e respeito. A partir desses valores, possui como princípios básicos: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade.

- Dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo. Para isso, recomenda-se a priorização de funcionalidades que representem maior valor possível para o negócio. Desta forma, caso seja necessário a diminuição de escopo, as funcionalidades menos valiosas serão adiadas ou canceladas.

- A XP incentiva o controle da qualidade como variável do projeto, pois o pequeno ganho de curto prazo na produtividade, ao diminuir qualidade, não é compensado por perdas (ou até impedimentos) a médio e longo prazo.

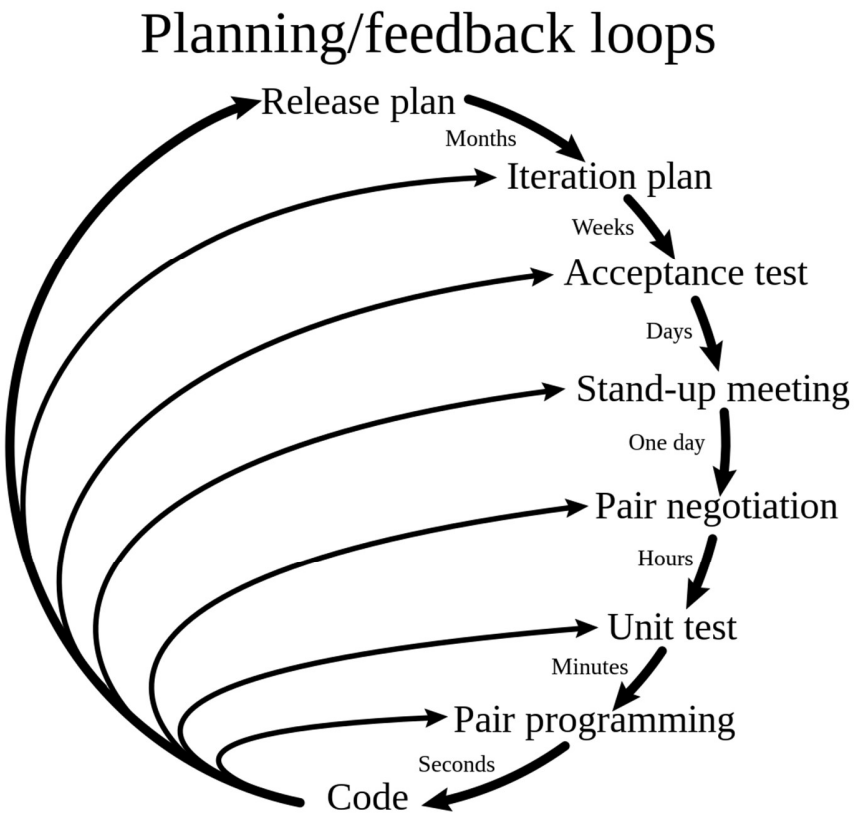


Figura 2.10. *Extreme Programming*. Fonte (*https://oness.sourceforge.net/proyecto/html/ch05s02.html*)

2.2.2.2. SCRUM

Scrum é uma estrutura leve, iterativa e incremental para desenvolver, entregar e sustentar produtos complexos. Ao contrário da abordagem sequencial para o desenvolvimento de produtos, a estrutura Scrum permite que as equipes se auto-organizem, incentivando a co-localização física ou a estreita colaboração *online* de todos os membros da equipe, bem como a comunicação face a face diária entre todos os membros da equipe e as disciplinas envolvidas (BEEDLE et al, 1998).

Segundo Ken Schwaber (2004), o Scrum é um *framework* dentro do qual pessoas podem tratar problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível. Além disso, é leve, simples de entender

e extremamente difícil de dominar. Está sendo usado para gerenciar o desenvolvimento de produtos complexos desde o início de 1990. Scrum não é um processo, mas um *framework* dentro do qual se pode empregar vários processos ou técnicas. O *framework* Scrum consiste em equipes de projeto (*times*) do Scrum associadas a papéis, eventos, artefatos e regras. Cada componente dentro do *framework* serve a um propósito específico e é essencial para o uso e sucesso do Scrum. As regras do Scrum integram os eventos, papéis e artefatos, administrando as relações e interações entre eles (SCHWABER, 2004).

Scrum é fundamentado nas teorias empíricas de controle de processo, ou empirismo. O empirismo afirma que o conhecimento vem da experiência e de tomada de decisão baseada no que é conhecido. O Scrum emprega uma abordagem iterativa e incremental para aperfeiçoar a previsibilidade e o controle de riscos. Três pilares apoiam a implementação de controle de processo empírico:

- Transparência – Aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados. Esta transparência requer aspectos definidos por um padrão comum para que os observadores compartilharem um mesmo entendimento do que está sendo visto.

- Inspeção – Os usuários (utilizadores) Scrum devem, frequentemente, inspecionar os artefatos Scrum e o progresso em direção a detectar variações. Esta inspeção não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas. As inspeções são mais benéficas quando realizadas de forma diligente por inspetores especializados no trabalho a se verificar.

- Adaptação – Se um inspetor determina que um ou mais aspectos de um processo desviou para fora dos limites aceitáveis, e que o produto resultado será inaceitável, o processo ou o material sendo produzido deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

O coração do Scrum é a *Sprint*, um *time-boxed* (perído) de um mês ou menos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado. Os *Sprints* têm durações coerentes em todo o esforço de desenvolvimento. Uma nova *Sprint* inicia imediatamente após a conclusão da *Sprint* anterior. As *Sprints* são compostas por uma reunião de planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento, uma revisão da *Sprint* e a retrospectiva da *Sprint*. O *Framework* Scrum é apresentado na Figura 2.11.

O Scrum prescreve quatro Eventos formais, contidos dentro dos limites da *Sprint*, para inspeção e adaptação, como descrito na seção Eventos do Scrum deste documento:

- Reunião de planejamento da *Sprint*
- Reunião diária
- Reunião de revisão da *Sprint*
- Retrospectiva da *Sprint*

Os principais Valores do Scrum são os valores de comprometimento, coragem, foco, transparência e respeito são assumidos e vividos pela equipe Scrum, os pilares do Scrum de transparência, inspeção e adaptação tornam-se vivos e constroem a confiança para todos. Os membros da equipe Scrum aprendem e exploram estes valores à medida que trabalham com os eventos, papéis e artefatos do Scrum.

A equipe (*time*) Scrum é composto pelo *Product Owner*, a equipa de Desenvolvimento e o *Scrum Master*. As equipes Scrum são auto-organizáveis e multifuncionais. Equpas auto-organizáveis escolhem qual a melhor forma para completarem seu trabalho, em vez de serem dirigidos por outros de fora da equipe. As equipes multifuncionais possuem todas as competências necessárias para completar o trabalho sem depender de outros que não fazem parte da equipe. O modelo de equipe no Scrum é projetado para aperfeiçoar a flexibilidade, criatividade e produtividade (SCHWABER, 2004).

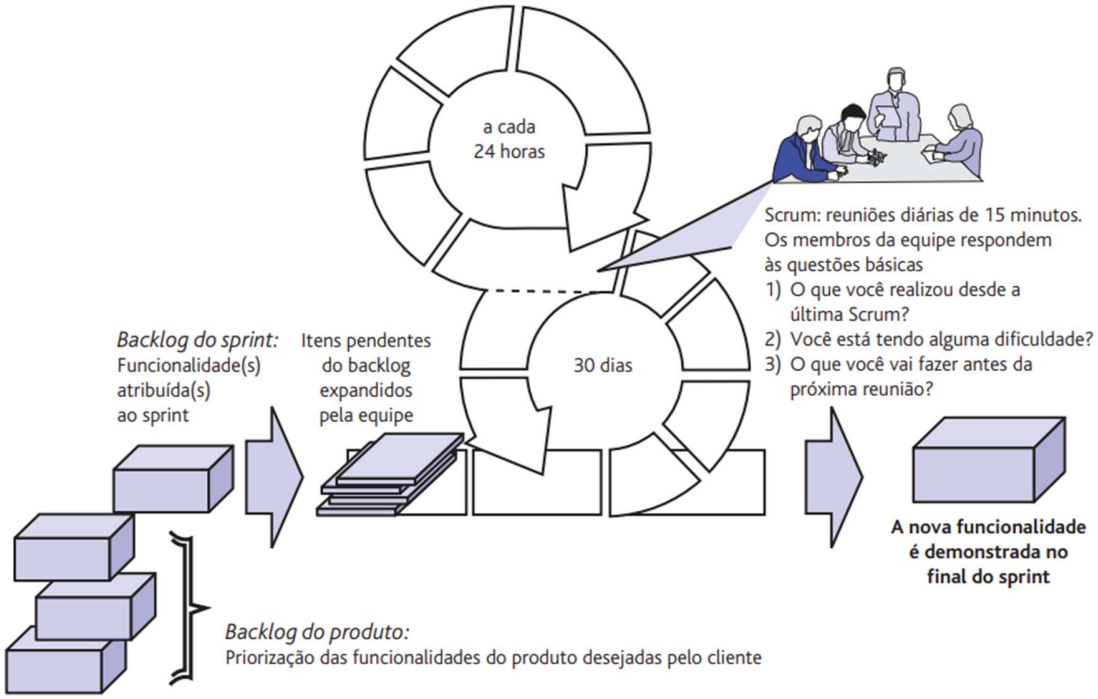


Figura 2.11. SCRUM. Fonte (PRESSMAN, 2016).

2.2.2.3. *FEATURE DRIVEN DEVELOPMENT* (FDD)

Feature-driven development (FDD), ou Desenvolvimento Dirigido por Funcionalidades, é um método leve e iterativo para desenvolvimento de software. Criado por Jeff de Luca e Peter Coad, combina gestão de projetos com boas práticas de engenharia de software (DE LUCA, 2002).

FDD (*Feature-Driven Development*) que significa desenvolvimento guiado a funcionalidades, assim como o XP, ASD, Scrum e AUP, faz parte das metodologias ágeis originais, sendo este um modelo incremental e interativo do processo de desenvolvimento de software que tem como lema: Resultados frequentes, tangíveis e funcionais.

Desenvolvido por Peter Coad e Jeff De Luca, foi inicialmente publicado em 1999 no livro "*Java Modeling in Color with UML*", sendo que uma das principais obras, que possui uma versão completa da metodologia foi publicada em 2002 após a sua utilização numa agência do *United Overseas Bank* em Singapura.

A FDD incorpora muitas das boas práticas de desenvolvimento já reconhecidas pela indústria em um conjunto coeso. Estas práticas todas são orientadas a funcionalidades, que é um conceito de valor do ponto de vista do cliente. O principal objetivo do FDD é entregar uma peça de software tangível e funcional para o cliente em espaços de tempo regulares.

Uma funcionalidade (ou *feature*), segundo Coad, "*é uma função com valor para o cliente que pode ser desenvolvida em duas ou menos semanas*". Pois um dos objetivos do FDD é apresentar para o cliente tais funcionalidades em um período fixo, em geral, duas semanas ou menos. Uma funcionalidade pode ser descrita como seguinte *template*.

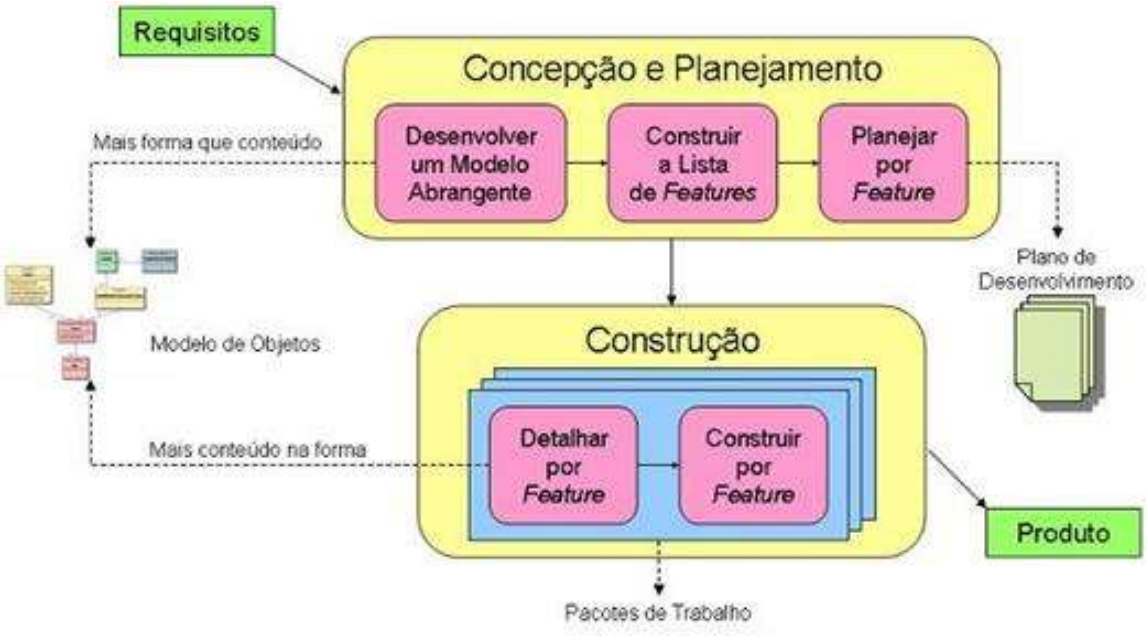


Figura 2.12. Feature Driven Development. Fonte (*https://jorgeaudy.com/2012/07/24/fdd-feature-driven-development/*)

2.2.2.4. *ADAPTIVE SOFTWARE DEVELOPMENT* (ASD).

Adaptive Software Development (ASD), em português: Desenvolvimento de Software Adaptativo ou Desenvolvimento adaptável de software é uma técnica para o desenvolvimento de software complexo, proposta por Jim Highsmith.

O apoio filosófico do ASD concentra-se na colaboração humana e na auto-organização. A auto-organização aparece quando agentes individuais independentes cooperam para criar resultados emergentes. Um resultado emergente é uma propriedade além da capacidade de qualquer agente individual (HIGHSMITH, 2002).

Características do ASD:

- Iterativo e incremental
- Sistemas grandes e complexos
- Arcabouço para evitar o caos
- Cliente sempre presente
- Desenvolvimento de aplicações em conjunto (*Joint Application Development* – JAD)

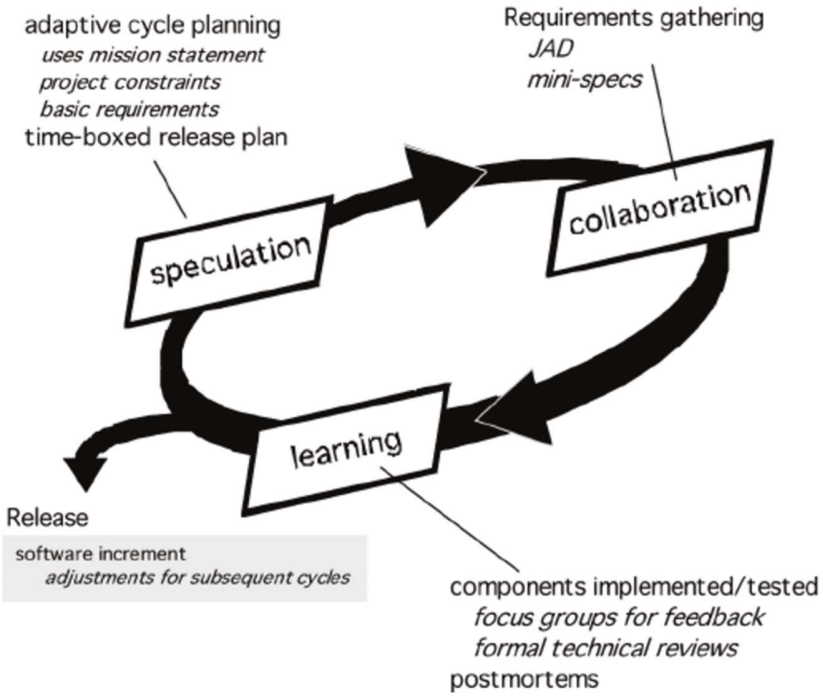


Figura 2.13. Adaptive Software Development. Fonte (ALNOUKARI, 2016).

2.2.2.5 *TEST DRIVEN DEVELOPMENT*

Test Driven Development (TDD) ou em português Desenvolvimento Dirigido a Testes é uma abordagem ágil de desenvolvimento de software que se intercalam testes e desenvolvimento de código, e que se relaciona com o conceito de verificação e validação, baseado em ciclos curtos de repetições. Inicia com o desenvolvedor escrevendo um código de forma incremental, em paralelo com um caso de teste para esse incremento. Não se passa para o próximo incremento até que o mesmo passe no teste (SOMMERVILLE, 2011).

O código uma vez produzido é validado pelo teste para posteriormente o código ser ré-fatorado para um código sob padrões aceitáveis, apresentado na Figura 2.14. Kent Beck, considerado o criador ou o "descobridor" da técnica, declarou em 2003 que TDD encoraja designs de código simples e inspira confiança. Desenvolvimento dirigido a testes é relacionado a conceitos de programação de *Extreme Programming*, iniciado em 1999. Através de TDD, programadores podem aplicar o conceito de melhorar e depurar código legado desenvolvido a partir de técnicas antigas.

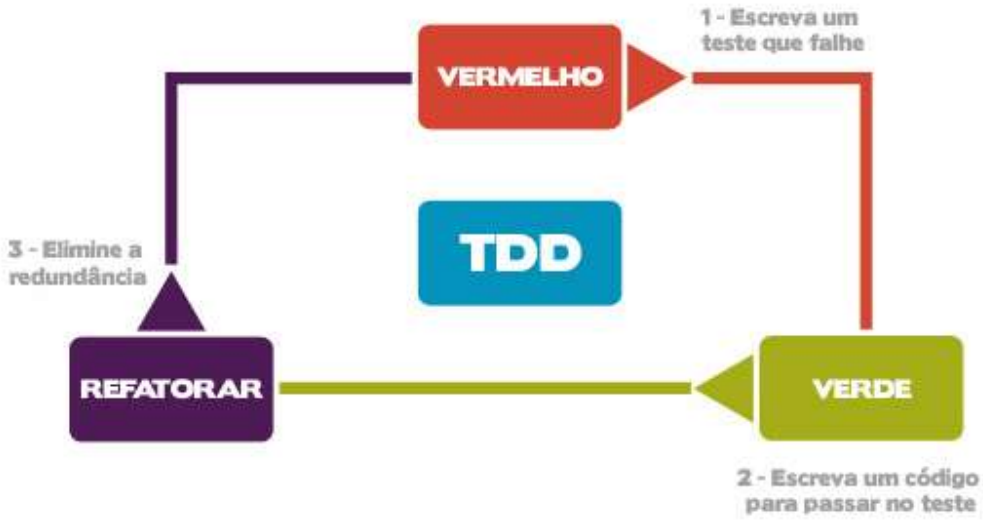


Figura 2.14. Test Driven Development. Fonte (*https://micreiros.com/tdd-test-driven-development-2/*)

As etapas do processo são:

- Identifica-se o incremento de funcionalidade necessária e deve ser pequeno e implementável em poucas linhas de código;
- Escreve-se um teste para essa funcionalidade e o executa-o de forma automatizada. Isso significa que o teste pode ser executado e relatado se passou ou falhou;
- Executa-se o teste, junto com todos os outros testes implementados. Inicialmente, a funcionalidade não é implementada, de forma proposital, o teste irá falhar e o resultado acrescentará ao conjunto de testes.
- Então, implementa-se a funcionalidade e executa-se novamente o teste. Isso pode envolver a ré-fatoração do código existente para melhorá-lo e adicionar um novo código sobre o que já existe;
- Após todos os testes serem executados com sucesso, será implementada a próxima parte da funcionalidade de forma incremental.

Um argumento forte a favor de utilização do TDD, é que facilita aos programadores, clarearem as suas ideias sobre o que um segmento de código supostamente deve fazer. Para escrever um teste, precisa-se entender a que se destina, e como esse entendimento faz que seja mais fácil escrever o código necessário. Certamente, se não tem conhecimento ou compreensão incompleta, o TDD não ajudará. Se não sabe o suficiente para escrever os testes, não vai desenvolver o código necessário. Por exemplo, se o cálculo envolve divisão (operação matemática), deve-se verificar se não está dividindo o número por zero. Se esse facto, for esquecido de escrever um teste para isso, então o código para essa verificação nunca será incluído no programa.

Além de um melhor entendimento do problema, outros benefícios do desenvolvimento dirigido a testes são:

- Cobertura de código. Em princípio, todo segmento de código que se escreve deve ter pelo menos um teste associado. Portanto, pode-se ter certeza de que todo o código no sistema foi realmente executado. Cada código é testado enquanto está sendo escrito; assim, os defeitos são descobertos no início do processo de desenvolvimento.

- Teste de regressão. Um conjunto de testes é desenvolvido de forma incremental enquanto um programa é desenvolvido. Pode executar testes de regressão para verificar se as mudanças no programa não introduziram novos *bugs* (erros ou falhas de *software*).

- Depuração simplificada. Quando um teste falha, a localização do problema deve ser óbvia. O código recém escrito precisa ser verificado e modificado. Não precisa usar as ferramentas de depuração para localizar o problema. Alguns relatos de uso de desenvolvimento dirigido a testes sugerem que, em TDD, quase nunca é necessário usar um sistema automatizado de depuração.

- Documentação de sistema. Os testes em si mesmos agem como uma forma de documentação que descreve o que o código deve estar fazendo. Ler os testes pode tornar mais fácil a compreensão do código.

Um dos benefícios mais importantes do TDD é que a redução dos custos dos testes de regressão. O teste de regressão envolve a execução de conjuntos de testes que tenham sido executados com sucesso, após as alterações serem feitas em um sistema. O teste de regressão verifica se essas mudanças não introduziram novos bugs no sistema e se o novo código interage com o código existente conforme o esperado. O teste de regressão é muito caro e geralmente impraticável quando um sistema é testado manualmente, pois os custos com tempo e esforço são muito altos. Em tais situações, precisa tentar escolher os testes mais relevantes para executar novamente, e é fácil perder testes importantes.

2.2.2.6. KANBAN

Em administração da produção, Kanban é um quadro de sinalização que controla os fluxos de produção ou transportes em uma indústria. O cartão pode ser trocado por outro sistema de sinalização, como luzes, caixas vazias e até locais vazios demarcados. Já na Engenharia de Software, Kanban é uma estratégia para otimizar o fluxo de valor para partes interessadas através de um processo que utiliza um sistema visual que limita a quantidade de trabalho em andamento através de um sistema puxado (ANDERSON, 2016).

Na década de 1960 a empresa Toyota criou o sistema Kanban para abastecimento e controle de estoques (*stocks*). O sistema movimenta e fornece itens de acordo com o consumo, fazendo com que não haja abastecimento de materiais sem solicitação. O Kanban foi baseado num sistema visual de abastecimento de um supermercado: conforme os produtos vão sendo vendidos (consumidos), os espaços vazios vão sendo reabastecidos. São utilizados cartões para o controle e funcionamento do Kanban, cartão utilizado indica que um material foi utilizado e precisa ser reposto, os cartões também são divididos por prioridade de reabastecimento, sendo separados pelas cores verde, amarela e vermelha, em sequência para itens de menor prioridade para maior prioridade. Esse procedimento dá uma visão mais ampla para o gestor responsável pelo processo de produção, que fica por dentro do andamento do sector. Para se realizar com sucesso o Kanban são feitos cálculos do número de cartões para cada item de material, levando em consideração o *Lead time*, o pedido médio, o estoque de segurança e a quantidade de peças no contentor. O sistema Kanban não tem como função reduzir estoques, apenas limita seu nível a um valor máximo, não podendo assim ser confundido com o sistema *just-in-time*, o sistema Kanban é considerado apenas uma parte do sistema *Just-in-Time,* metodologia desenvolvida e aperfeiçoada em 1940 por Taiichi Ohno e Sakichi Toyoda conhecida como Sistema Toyota de Produção (ANDERSON, 2016).

O Kanban foi adaptado por David J. Anderson (fundador da Kanban University) como técnica para ser utilizada no desenvolvimento de software. Neste contexto ele é

definido pelo seu criador como "*Um método para definir, gerenciar e melhorar serviços que entregam trabalho de conhecimento, tais como serviços profissionais, atividades criativas e o design de produtos físicos e de software. Pode ser caracterizado como um método de 'começar com o que você faz agora' – um catalisador para mudanças rápidas e focadas nas organizações – que reduz a resistência a mudanças benéficas de acordo com os objetivos da organização"* (ANDERSON, 2016).

São 6 as práticas do Kanban utilizados para o desenvolvimento de software:

- Visualização do Fluxo de Trabalho: visualização através do uso do quadro Kanban para fomentar as conversas certas no momento certo e criar oportunidades de melhoria;
- Limitando o Trabalho em Progresso (WIP): trabalho em Andamento (do inglês *Work in Progress*, ou WIP) se refere aos itens de trabalho que a equipe iniciou e que ainda não terminou, uma prática para diminuir excessos de trabalho paralelo;
- Gerenciamento do Fluxo; maximizar o valor entregue, garantindo que os Itens de Trabalho não são deixados para envelhecer desnecessariamente e que sejam completados;
- Políticas Explícitas sua definição de "Fluxo de Trabalho": incluindo a criação de políticas documentadas e que toda a equipe compreenda, úteis para melhorar as suas performances;
- Ciclos de *feedback,* cadências para inspecionar e controlar o fluxo de trabalho, observar melhorias e aplicar no sistema, sempre com foco no trabalho;
- Melhoria Colaborativa e Evolução Experimental utilizando pelos como o método científico e observações empíricas, executar experimentos baseados em dados para melhorar e evoluir o sistema Kanban.

O Kanban pode ser utilizado sozinho, sem a necessidade de outros *Frameworks* ou metodologias. David J. Anderson descreve o Kanban como um caminho alternativo para a agilidade. Porém, o método também pode ser usado para apoiar outros métodos. A perspectiva baseada em fluxo do Kanban pode melhorar e complementar o *framework* Scrum e sua implementação. As equipes podem aplicar o Kanban tanto se estiverem começando a usar o Scrum quanto se já o utilizam há muito tempo, apresentado na Figura 2.15.

Utilizando o Kanban para visualizar o trabalho de novas formas, uma equipe Scrum pode aplicar o conjunto de práticas do Kanban para otimizar o valor da entrega de maneira mais efetiva. Essas práticas se apoiam e expandem os princípios do pensamento Lean, fluxo de desenvolvimento de produtos e teoria das filas (ANDERSON, 2016).

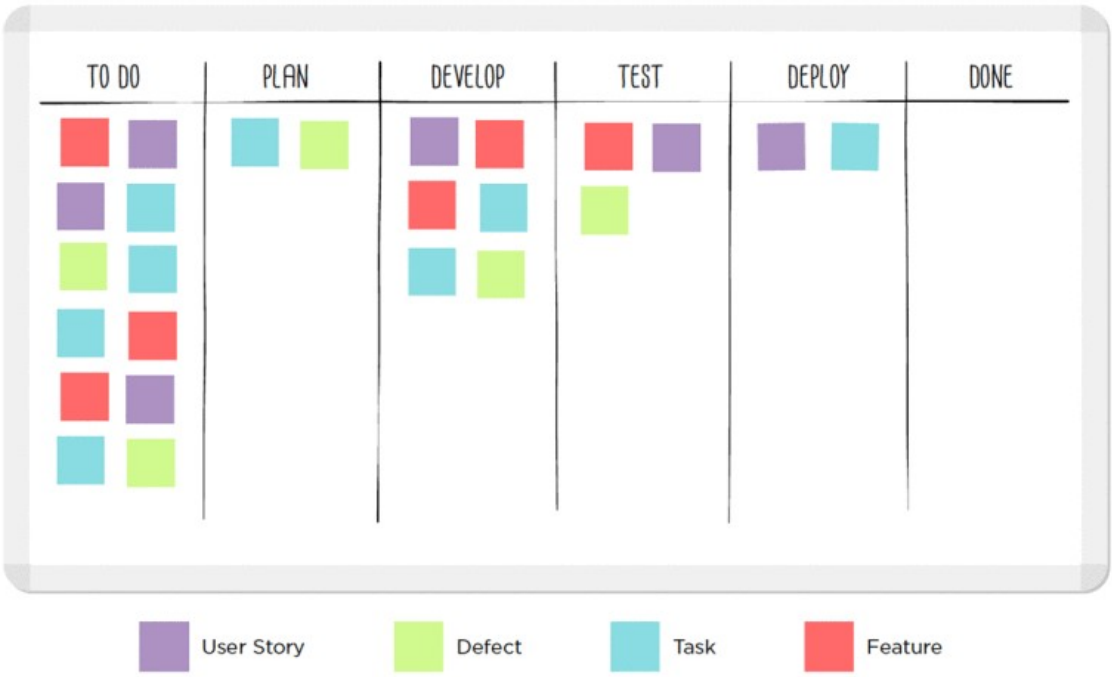


Figura 2.15. Kanban no Desenvolvimento de Software. Fonte (*https://www.heflo.com/pt-br/agil/metodo-kanban/*)

2.3. PRECIFICAÇÃO DE SOFTWARE

Ação de precificar corresponde de colocar preço em algum produto ou serviço. A composição do preço não pode basear-se em palpites. Trata-se de processo estratégico que envolve um conjunto de análises: financeira, de mercado e marketing do negócio. Quando conduzida corretamente, a precificação se torna um ponto de equilíbrio entre preço, valor e margem de lucro. Mas é importante afirmar que essa composição não é estática. Os preços do seu produto ou serviço devem sempre priorizar os objetivos comerciais da empresa dentro de um monitoramento constante das mudanças mercadológicas (CORA, 2023).

Especificar o preço do produto é um desafio para qualquer empresa. Mas para alguns segmentos esse processo é bem mais desafiador. Em uma empresa de modelo fabril, esse cálculo poderia ser simples: considera-se o valor da matéria prima, o custo de maquinário, custo de operação, custo administrativos e tributários, acrescentado a margem de lucro da empresa, e obtém-se o preço do produto final. Mas para a área de Software, precisa-se levar em consideração outras variáveis, bem mais subjetivas e complicadas de calcular. É necessário encontrar o equilíbrio entre o valor de investimento realizado pela empresa e o valor percebido pelo mercado.

O mercado não quer saber quanto custou produzir o produto de software, há não ser quando o desenvolvimento for por demanda. Mas nesses casos o processo de aquisição é gerido por contrato de compra e venda entre as partes. Tudo que importa para o mercado, é o quanto o software vai ajudar a empresa a aumentar a produtividade, economizar tempo, cumprir obrigações, pois no final, o mercado visa o lucro. Se o produto de Software estiver com um preço muito caro, e que o mercado não acredite que terá bom ROI. Esse produto será bem difícil de vender. Por outro lado, se o preço do produto de Software estiver com um preço muito baixo, poderá até atrair muitos clientes, mas pode implicar nos custos operacionais.

Portanto, fazer os cálculos corretos é indispensável e tem como foco a sustentabilidade da empresa. Podemos observar algumas vantagens ao precificar assertivamente:

- Maximização dos lucros da sua empresa:
    - Sem uma composição de preços bem elaborada, os riscos de prejuízo são bem elevados;

- Aumento das vendas:
    - Com o preço correto, a empresa se torna muito mais competitiva no mercado;

- Otimização do planejamento financeiro:
    - Dessa forma tem-se uma perspectiva mais certa sobre o futuro da empresa;

- Melhora da projeção bancária:
    - Obter o controle sobre os custos reais da empresa, consegue investir mais seguramente;

- Compreensão das flutuações do mercado:
    - Em caso de crise, conseguirá lançar novos valores sem ter prejuízo;

- Entendimento mais claro dos gastos da empresa:
    - Saberá o que está pesando demais no orçamento e pode buscar saídas para isso;

Diante disso, para calcular o preço do produto de Software, terá como base dois critérios: O Valor Investido e o Valor percebido pelo mercado (TECHNO, 2023).

O Valor Investido compreende a análise dos custos de produção e manutenção do produto (no caso, as atividades inerentes de um ciclo de vida de desenvolvimento de

software). Bem como as demais despesas fixas e variáveis da empresa, aluguel do imóvel, energia elétrica, Internet, custos administrativos, custos trabalhistas (mão de obra direta e indireta, junto com encargos), etc. O objetivo é encontrar um valor mínimo para a empresa não operar no vermelho.

Custos com a Implementação e Manutenção baseiam-se na quantidade de horas trabalhadas multiplicado pelo valor da hora técnica da equipe. Nos dias atuais, o Treinamento e Suporte ao software é mais difícil determinar um valor para cliente, pois muito têm-se praticado a produção de materiais de treinamento e isso seria disponibilizado aos clientes. Em determinados mercados, a adoção de atendimento *online* ou até robotizados resolve uma grande parte das necessidades dos clientes. Há não ser em casos em que a formalização de contratos se faz necessário.

O Valor Percebido é o resultado do quanto o mercado está disposto a pagar pelo produto de Software. O objetivo é encontrar o valor máximo de oferta sem que o mercado ache o preço absurdo ou incompatível com os benefícios que o produto de Software pode oferecer. É procurar entender o quanto o mercado vai ganhar ao utilizar o produto de Software. Além dessa consideração, qual impacto o produto de Software ofereceria ao mercado como, comodidade, performance, facilidade de implantação e uso, integração com outros produtos, controle com obrigações fiscais, auxílio às tomadas de decisão utilizando relatórios gerenciais?

Além disso, o modelo comercial como compra de licença ou aluguel (*Software as a Service*) são claros e simples? E principalmente, as ameaças dos concorrentes. Consolidar essas variáveis é compor um Valor ao produto de Software que o mercado considere atrativo.

2.4. CUSTOS ASSOCIADOS AO DESENVOLVIMENTO DE SOFTWARE

Tendo como premissa, o entendimento da relevância do arcabouço de processo de software, junto com o conhecimento sobre o comportamento de um ciclo de vida de desenvolvimento de software, e principalmente da adoção de uma boa estratégia de

precificação, como forma de subsistência da organização desenvolvedora de software. Pode-se questionar o seguinte. Todo esse conhecimento torna mais fácil entender os custos de desenvolvimento de software? A resposta é não. Pois com todo esse conhecimento, se o projeto de desenvolvimento de software não tiver uma boa gestão, a falta de controle dos requisitos essenciais de uma boa gestão (controle de custos, qualidade e tempo), levará a um prejuízo financeiro e principalmente a insatisfação do cliente.

Diversos fatores influenciam os custos de desenvolvimento de software. Além dos problemas de ingerência nos projetos de software, já citados. Existem os inerentes ao produto de software a ser desenvolvido. Destaca-se os seguintes:

- Tamanho do Software;
    - Pode-se usar métricas como Ponto de Função para determinar o tamanho de Software, entre outras. O importante é entender a necessidade de métrica, pois sempre estará atrelado a um custo de esforço para mensurar. E principalmente, entender que para a melhor definição do tamanho do software, baseia-se no refinamento repetitivo da mensuração, e isso aumenta os custos.

- Complexidade do Software;
    - Além do aspecto de inovação, desbravar o desconhecido até estabelecer um requisito de software. Criar as funcionalidades do software e atender a critérios não funcionais, como segurança, usabilidade, etc. (requisitos não funcionais). O produto de software deverá atender às necessidades do negócio organizacional do cliente, pois isso tem impacto na produtividade e o ROI investido pelo cliente. Além disso, os aspectos tecnológicos como portabilidade e interoperabilidade entre dispositivos necessários colaboram para complexidade.

- Complexidade do Projeto;
    - As características organizacionais para trabalhar com projetos, sejam funcionais ou orientado a projetos, já possuem seus próprios desafios. Além disso, se intensificou o desenvolvimento distribuído de software, chamado

de (remoto ou híbrido) atendendo questões sanitárias pós pandemia COVID19. Apesar de um aparato tecnológico para colaborar no comprometimento dos membros da equipe, para atender as atividades dos projetos. Surgiu a necessidade do fortalecimento da habilidade do gestor do projeto, ao tratar todos esses desafios, e que não são simples.

- Quantidade de recursos necessários;
    - As organizações têm a responsabilidade de prover todos os recursos necessários para execução do projeto de desenvolvimento de software. Sendo que recentemente, agravou-se mais, pois a atividade de gerenciar os recursos enviados para os membros executarem as suas atividades remotamente, aumentou os custos pois envolve (computador, telefonia móvel, até mesmo contratar sinal de Internet), tudo isso para atender as demandas do projeto.

- Qualidade de Software;
    - De facto, o mito de que quanto mais se garante a qualidade de software, pois representa o custo de um esforço, mais encarece o produto de software. Não se espera que o software embarcado em um avião apresente falhas em pleno voo, pois espera-se que tenha sido testado ao extremo. Entende-se que Qualidade de Software é o caminho mais fácil para garantir a subsistência de uma organização desenvolvedora de software. Pois sem qualidade nos seus produtos, não haverá mais contratações de novos projetos.

- Migração de dados existentes;
    - Em casos específicos, os projetos de software precisam garantir que os dados de sistemas legados da organização cliente sejam utilizados no novo projeto. Dessa forma, a migração das diversidades de plataformas de banco de dados pode aumentar os custos e principalmente os riscos do projeto, por fazem parte dos ativos (dados refletem o conhecimento) da organização.

- Suporte em operação e manutenção;

- Conforme a ISO/IEC 12207, a Operação do produto de software, baseado em homologação do software (aceite por parte do cliente) e treinamento aos utilizadores precisam de ser planejados e executados nos projetos. Alguns modelos de negócios cobrem contratos de manutenções (corretivas ou evolutivas) e isso está inserido nos custos do produto de software.

2.5. CONTABILIZAÇÃO EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

Atualmente, investidores consideraram o investimento em intangíveis um ativo, se o investimento é capitalizado. Por exemplo, ágio (juros), alguns custos de desenvolvimento de software e custos de exploração de petróleo ou gastos, este caso, por exemplo, P&D (I&D), tecnologias de informação, recursos humanos, alguns custos de desenvolvimento de software e custos de exploração (GIVOLY & SHI, 2007).

A constatação de que os investimentos em intangíveis são valorizados pelos investidores é frequentemente usada como argumento em favor da capitalização desses gastos. Observa-se, no entanto, que por ser empreendidos pelas empresas, os investimentos em intangíveis devem ter um valor presente líquido esperado positivo, pelo menos aos olhos da administração das empresas investidoras e muito provavelmente aos olhos dos outros investidores. Nesse sentido, a constatação empírica de que o mercado vê os gastos com P&D como um ativo, apenas confirma o óbvio – que os investidores não põem em dúvida ao investir em intangível.

Tampouco os contadores levantam essa questão, embora aceita pelos princípios gerais da contabilidade (PGC), determinam a contabilização de despesas de P&D. Na verdade, a razão para isso é a exigência de despesas porque o nível de receita líquida esperada de P&D seja benefícios, mas sim por causa da incerteza que os cerca. Descontando o custo desses intangíveis, fica de acordo com a definição contábil de um ativo. Ou seja, para um investimento ser qualificado como um ativo no balanço, não deve ser associado apenas ao lucro líquido esperado como benefícios econômicos, mas esses benefícios também devem ser “prováveis”.

Enquanto a condição de ter "prováveis benefícios econômicos" não é definido com precisão, sugere que um grau relativamente alto de certeza sobre a recuperabilidade do investimento é necessário antes que ele se qualifique como um ativo.

O debate perene sobre capitalizar ou gastar o custo dos intangíveis, portanto, não pode ser resolvido em favor da capitalização pela constatação de que o mercado valoriza positivamente o investimento intangível. No entanto, o suporte empírico para a capitalização pode ser fornecido por dois outros tipos de evidência. Um tipo de evidência é aquele que mostra que a incerteza em relação ao futuro benefícios de intangíveis não é "suficientemente" maior do que, o de ativos tangíveis (como imobilizado) que são capitalizados de modo a justificar um tratamento contábil diferenciado. Surpreendentemente, as evidências sobre o grau diferencial de risco do intangível, em comparação com os ativos tangíveis, é escasso (GIVOLY & SHI, 2007).

Outro suporte para a capitalização poderia estar na forma de evidência mostrando que a capitalização, quando permitida, é informada aos investidores. Nos EUA, os únicos custos de intangíveis desenvolvidos internamente que podem ser capitalizados (sob certas condições) são aqueles associados ao desenvolvimento de software. Pois exige que os custos envolvidos na criação de software sejam tratados de acordo com a viabilidade tecnológica do projeto. A viabilidade tecnológica é normalmente estabelecida após a conclusão de um projeto detalhado do programa ou modelo de trabalho. Todos os custos antes de atingir o estágio de viabilidade tecnológica devem ser lançados como despesas conforme incorridos em P&D. Custos de desenvolvimento de software incorridos após a obtenção da viabilidade tecnológica, rotulados como "custos de software em produção", devem ser capitalizados. A avaliação da elegibilidade do produto para capitalização é feita produto a produto. Na prática, a avaliação da viabilidade tecnológica é influenciada pelo critério.

Especificamente, a capitalização indica que é provável, que o custo de desenvolvimento de software (doravante, CDS) (capitalizado) seja recuperado, ou seja, a distribuição de probabilidade da administração dos fluxos de caixa futuros é truncada de

baixo para cima. Por outro lado, a eleição da administração para contabilizar todo o seu CDS não fornece informações sobre a distribuição dos fluxos de caixa futuros decorrentes do investimento no desenvolvimento de software. Portanto, a capitalização do CDS, por ser mais informada sobre a distribuição dos fluxos de caixa futuros, reduz a assimetria de informação e, portanto, diminui a extensão da subavaliação de empresas que capitalizam esses custos, em relação às empresas que os gastam esses custos.

Existem casos de empresas que usaram a capitalização de despesas operacionais regulares contra procedimentos contábeis comuns, provavelmente para aumentar artificialmente o seu fluxo de caixa operacional. Apesar disso, pode influenciar os lucros a curto prazo da empresa, essa prática ilegal é geralmente exposta a longo prazo. É essencial separar os conceitos de capitalização e capitalização. Empresas podem normalmente capitalizar custos somente quando o recurso adquirido fornecer valor futuro, significando que os recursos são vantajosos para os negócios por mais de um ciclo operacional.

Dessa forma, as despesas com a aquisição desses recursos são registradas por contadores como ativos no balanço patrimonial da empresa. Assim, os custos aparecerão no balanço patrimonial nos próximos anos financeiros por meio de amortização ou depreciação. As empresas devem também considerar capitalizar custos quando eles aumentam significativamente o valor de um recurso existente. Se a empresa atualiza parte das ferramentas, propriedades ou equipamentos utilizados de forma a aumentar diretamente o valor do ativo, ele pode ser capitalizado.

2.6. CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentadas as principais características do arcabouço do processo de software, que servem de sustentação para definir um processo de software. O comportamento e a flexibilidade dos ciclos de vida mais tradicionais e as principais metodologias ágeis de desenvolvimento de software também foram apresentados. Da mesma forma, foi apresentado a complexidade de se estruturar um bom preço para o software pois, a organização precisa garantir a subsistência da organização desenvolvedora de software, através de uma boa observação do mercado no qual atua. Foi apresentado os

fatores que impactam nos custos durante o planejamento e execução de projetos de software. Bem como, as peculiaridades para registrar as despesas na contabilidade da organização.

Capítulo 3

Estimativas, Melhoria de Processo de Software e ROI

Neste capítulo são apresentadas as principais disciplinas que compõem o Referencial Teórico sobre Estimativas de Software, Melhoria de Processo de Software e ROI. Em seguida, serão apresentadas Soluções Alternativas, Trabalhos Relacionados com o tema.

3.1 ESTIMATIVAS DE SOFTWARE

Boas estimativas é a chave para o sucesso de projetos e produtos. As estimativas fornecem dados para avaliar a adesão a especificações de desempenho e planos, tomar decisões, revisar (rever) projetos e planos, e melhorar as estimativas futuras e processos. Os engenheiros usam as estimativas e medições para avaliar a viabilidade e acessibilidade dos produtos propostos, escolha entre projetos alternativos, avaliar os riscos, e apoiar decisões de negócios. Por sua vez, os gestores usam os recursos de estimativas para calcular o custo do projeto e o seu cronograma, e para preparar os orçamentos e planos. Um bom método produz uma boa estimativa para todas as quantidades necessárias, sem exceder os recursos alocados para a estimativa. O requisito primário para o método é o de proporcionar um valor para uma quantidade com um nível conhecido e adequado de precisão. Todas as estimativas, por sua própria natureza, possuem erros (STUTZKE, 2005).

As questões-chave são: Quão certo o valor estimado tem alguma precisão particular? Por quanta precisão realmente precisa? Por quanta precisão está disposto a pagar? A precisão de um valor de estimativa depende de duas coisas: Inerente à precisão da técnica de estimativa ou modelo que você escolher, e quaisquer erros nos valores dos parâmetros de entrada. Para reduzir os erros nos valores de entrada, pode-se definir e estruturar o processo de estimativa, usar dados históricos, e treinar as pessoas que preparam as estimativas. Para criar um método de estimativa, devem-se seguir alguns critérios mandatórios, desejáveis e práticas de bons métodos de estimativas (STUTZKE, 2005).

- Mandatório: o método deve ajudar os participantes a identificar os principais fatores que afetam os valores estimados e a incerteza nesses valores. Deve também identificar todas as tarefas do projeto usado para produzir esses itens e os recursos necessários para a equipe do projeto. A validação verifica a exatidão, integridade e consistência dos valores estimados. Os interessados devem aprovar a estimativa documentada e aceitá-la como uma *baseline* para o planejamento seguinte do projeto;

- Capacidades desejáveis: o método de estimativa deve permitir refinar a estimativa como recurso de projeto. Também deve promover uma melhor compreensão dos elementos e fatores que podem contribuir para a estimativa do valor de uma quantidade específica. Um bom método deve organizar e preservar a sua experiência através da captura de técnicas comprovadas, relações de estimativas e os valores de produtividade;

- Aspecto prático: O método de estimativa deve ser relevante para o seu ambiente particular, confiável e acessível. Ambiente inclui o domínio da aplicação, linha de produtos, organização e cliente. O método deve fornecer uma precisão aceitável dentro das restrições de tempo e dinheiro. Deve usar dados de entrada que está disponível no momento da estimativa, ou dados que podem ser facilmente estimados.

3.1.1. TÉCNICAS DE ESTIMATIVAS

Conforme apresentado no Quadro 3.1, existem várias técnicas de estimativas de software que são classificadas como paramétricas, análogas ou por julgamento de especialistas. Veremos a seguir a relação dessas técnicas de estimativas.

Quadro 3.1 Técnicas de Estimativas (VALENÇA, 2007).

Nome da Técnica	Organização/Pesquisador	Ano
Delphi	Rand Corp.	1959
Nelson´s SDC	SDC	1966
Wolverton	TRW	1974
RCA Price-S System	RCA	1976
Halstead	M.H.Halstead	1977

Walston and Felix	IBM	1977
SLIM – *Software Lifecycle Management*	QSM	1978
Function-point Method	IBM	1979
Parr Model	F.N.Parr	1980
COCOMO-Model	TRW	1981
Wideband Delphi	Barry Boehm	1981
SOFTCOST	JPL	1981
Bailey and Basili	NASA	1981
Bang Metrics	Tom DeMarco	1982
SEER-SEM – *System Evaluation and Estimation of Resources – Software Engineering Model*	Galorath Incorporated	1988
MARK II *Function Points*	*United Kingdom Software Metrics Association*	1988
NESMA	*Netherlands Software Metrics Association*	1989
3D Function Points	Boeing Company	1992
Use Case Points	Objectory	1993
Object Points	Banker D.R.	1994
COCOMO II	University of Southern California	1995
Proxy based Estimation (PROBE)	SEI-*Software Engineering Institute*	1996
COBRA	Fraunhofer Institute for Experimental Software Engineering	1997

Full Function Points	COSMIC – *Common Software Measurement International Consortium*	1997
Class-Method Points	Galorath Incorporated	1999
Web Objects	Reifer Consultants, Inc	2000
Web Points	CHARISMATEK Software Metrics	2000
Planning Game – Extreme Programming	Beck and Fowler	2001
Internet Points	Cost Xpert Group	2001
Planning Poker – Extreme Programming	Object Mentor	2002
Data Web Points	Computer Sciences Department. University of Chile	2003
Agile COCOMO II	USC Center for Software Engineering	2003
Story Points	Mike Cohn	2003
OO-Method Function Points (OOmFP)	Department of Information System, Valencia University of Technology	2004
TUCP – Technical Use Case Points	UNIFOR – Universidade de Fortaleza – Tatiana Monteiro	2005

Pôde-se observar que, ao longo dos tempos, várias técnicas de estimativas foram criadas, atendendo ao seu determinado contexto e necessidade, mas algumas técnica já ficaram obsoletas.

3.1.1.1. TÉCNICAS PARAMÉTRICAS OU ALGORÍTMICAS

Na Técnica Paramétrica ou Algorítmica, a estimativa é calculada usando informação histórica da organização, que relacionam algumas das métricas de software sobre o tamanho ao esforço do projeto. Uma estimativa é feita da métrica, e a técnica prevê o esforço requerido. As técnicas paramétricas assumem que existe um relacionamento

matemático entre o tamanho, esforço, prazo e qualidade e que o relacionamento normalmente é afetado por fatores mensuráveis de desempenho também chamados parâmetros.

A técnica é projetada para criar equações matemáticas para realizar a estimativa de software. Estas equações matemáticas são baseadas em pesquisa e dados históricos da organização e usam como entradas, dados como linhas de código fonte (SLOC), número de funções a serem realizadas, e outros direcionadores de custo tais como linguagem, metodologia de projeto, níveis de conhecimento, exposição a riscos, tamanho da equipe, etc.

As técnicas algorítmicas têm sido largamente estudadas, sendo desenvolvidas diversos, tais como as técnicas COCOMO, e as baseadas em unidades funcionais. Para obter melhores resultados, as técnicas paramétricas devem ser calibradas com dados da organização no local de desenvolvimento (McGARRY, 2001).

3.1.1.2. TÉCNICAS ANÁLOGAS

A estimativa por analogia é a técnica de estimativa mais usual na indústria de software (JORGENSEN, 2002). Estimar por analogia significa, comparar o objeto proposto aos projetos similares que a organização já finalizou e onde as informações do desenvolvimento dos mesmos são conhecidas. Dados dos projetos finalizados são utilizados para estimar o projeto proposto. Esta técnica pode ser usada tanto em nível de sistema quanto em nível de componente. A analogia é geralmente realizada por especialista em estimativas (utilizando como base aa experiência de projetos anteriores) ou por algoritmos de busca de projetos similares. Esta abordagem requer um conhecimento detalhado do projeto, para identificar as diferentes características específicas entre o projeto proposto e projetos anteriores, que foram usados como base para realizar a estimativa.

3.1.1.3. TÉCNICAS POR JULGAMENTO DE ESPECIALISTAS

Técnicas de julgamento de especialistas, são realizadas através da consulta entre especialistas em estimativas de software, usando a sua experiência e conhecimento em

projetos similares ao do projeto proposto para obter uma indicação de estimativa de seu esforço de desenvolvimento. As técnicas baseadas na experiência de especialistas são úteis na ausência de dados históricos e quantitativos. Estudos em diferentes áreas de conhecimento demonstram que a formação de uma equipe com 3 a 5 especialistas com formação multidisciplinares parecem ser suficientes. Recomenda-se a utilização de especialistas com diferentes formações, diferentes papeis, ou que usam técnicas diferentes de estimativas (JORGENSEN, 2002b).

3.1.2 TÉCNICA DE ESTIMATIVA ADOTADA

Como vimos, algumas técnicas de estimativas de software possuem características paramétricas, que prevê os valores médios para contextos semelhantes. Modelos paramétricos utilizam algoritmos para calcular os valores das variáveis dependentes com base nos valores de diversas variáveis independentes (parâmetros). Construtores de modelo definem as relações quantitativas entre as variáveis dependentes e independentes com base em dados empíricos, teorias, técnicas estatísticas e heurísticas. Modelos paramétricos provem a integridade e consistência, pois o modelo deve fornecer valores para um conjunto de parâmetros, que a estimativa considera cuidadosamente todos os fatores que afetam a quantidade estimada (STUTZKE, 2005).

Segundo PARTHASARATHY (2007), a menos que todos os ingredientes chave para estimar sejam identificados e avaliados cuidadosamente, o processo de estimativa em si será incompleto e, em muitas situações, o resultado não será de muita utilidade (PARTHASARATHY, 2007). Os principais elementos a serem discutidos são:

Escopo ativo: O elemento chave importante que constitui a base de cálculo é o escopo da atividade que está sendo estimado. Escopo é um termo muito solto e vai assumir diferentes formas em diferentes situações. Por exemplo, se na construção de uma aplicação de software, o escopo pode ser o tamanho do software em termos de funcionalidade que ele oferece ou em termos de linhas de códigos entregues.

Ambiente de trabalho: O ambiente em que a atividade está sendo executada faz um enorme impacto sobre a estimativa global. Tópicos de ambientes podem variar de

condições meteorológicas para a interferência política. Todos eles desempenham papel crucial na estimativa final.

Consistência: Dados históricos, a competência da equipe que executa a atividade, e melhores práticas desenvolvidas ao longo dos anos, derivar a estimativa de tempo e, portanto, a programação para a atividade a ser entregues. Apesar da repetição de conjuntos semelhantes de atividades envolvidas na execução de uma determinada tarefa, cada nova tentativa é diferente de alguma forma de uma anterior. Equipes encontram novos impedimentos e problemas, que podem ou não podem ter ocorrido na ocasião anterior. São essas iterações que fazem uma pessoa ou uma equipe "*experiente*". E com cuidado registrar e analisar as experiências, você pode adicionar grande valor para a estimativa final. Em linguagem de software, a taxa de entrega de uma unidade de trabalho é mais conhecida como produtividade.

Utilização de ferramentas: Ferramentas que são usadas para executar uma atividade pode desempenhar um papel significativo na definição do esforço e do tempo necessário para completar a atividade. Ferramentas podem assumir diferentes formas e tamanhos, dependendo do tipo de atividade. Uso eficaz de ferramentas tem um grande impacto na produtividade.

Segundo BOEHM (2000), o modelo COCOMO (*COnstructive COst Model*) também conhecido como *COCOMO 81* é um método que busca medir esforço, prazo, tamanho de equipe e custo necessário para o desenvolvimento do software. O modelo não apresentava capacidade de lidar com ciclos iterativos e com a utilização de componentes COTS (*Commercial-Off-The-Shelf*). Hoje em dia é considerado obsoleto, e foi substituído pelo COCOMO II (STUTZKE, 2005).

O COCOMO II estima o custo e tempo baseado em pessoa-mês e meses, respectivamente, para a determinação do *baseline* de exigências de um produto para a conclusão de uma atividade. O modelo apresenta três sub-modelos que abordam as diferentes fases em que o projeto ou atividade que está sendo realizada, onde a utilização de cada sub-modelo aumenta a precisão e fidelidade de estimativa ao longo do processo de planejamento e execução de um projeto. O sub-modelo *Application Composition* é o mais

apropriado para o estágio de prototipação no ciclo de vida espiral. O sub-modelo *Early Design* é apropriado quando as exigências são conhecidas e as alternativas de arquitetura foram exploradas. O sub-modelo *Post-Architecture* é mais detalhado e envolve as etapas de construção real do software e de manutenção (STUTZKE, 2005).

A estimativa *Function Points- FP* (Ponto de Função), originalmente concebido por ALBRECH em 1979, ganhou crescente popularidade a partir da criação do *International Function Point Users Group* (IFPUG), em 1986. Em 2002 o FP passou à condição de padrão internacional, através da norma ISO/IEC 20926.

A Análise por Pontos de Função (APF) mede o tamanho do software pela quantificação de suas funcionalidades, baseadas no projeto lógico ou a partir do modelo de dados segundo a visão e os requisitos do usuário final. As suas principais características são: ser independente da tecnologia, ser aplicável desde o início do sistema, apoiar a análise de produtividade e qualidade e estimar o tamanho do software com uma unidade de medida padrão (STUTZKE, 2005).

Para mensurar o tamanho do software utilizando esta abordagem, devem-se seguir os seguintes passos: (i) Estabelecer o objeto de contagem (projetos de desenvolvimento, projetos de manutenção ou contagem de uma aplicação); (ii) Determinar a fronteira de medição (a fronteira de medição deve ser sempre determinada sob o ponto de vista do usuário); (iii) Contar as funções de dados, divididos em Arquivos Lógicos Internos (ALIs – que são grupos lógicos de dados mantidos dentro da fronteira da aplicação) e Arquivos de Interface Externa (AIEs – arquivos somente referenciados pelo aplicação); (iv) Contar as funções transacionais, divididos em Entradas Externas (EEs), Saídas Externas (SEs) e Consultas Externas (CEs); (v) Determinar o Fator de Ajuste, conjunto de 14 características que influenciarão a complexidade do software; e, (vi) Determinar o tamanho do projeto (considera as funções de dados, transacionais, fatores de ajuste e tipo de projeto).

A estimativa *Use Case Points-UCP* (Pontos por Caso de Uso) foi criado por Gustav Karner em 1993, como uma adaptação específica de *FP*. É uma técnica de estimativa de tamanho de projeto de software orientado a objetos. A técnica é simples, rápida e fácil de

usar. O processo de contagem dessa métrica é contar os atores e casos de uso. Com base nessa contagem são calculados os UCP não ajustados. Posteriormente, são determinadas as complexidades dos fatores técnicos e dos fatores ambientais. Extensões do modelo como a TUPC – *Technical Use Case Points* (Pontos de Caso de Uso Técnicos) tem como objetivo um cálculo mais acurado para as estimativas a partir de casos de uso. O grau de precisão da TUCP depende se a organização possui um modelo para especificação de casos de uso, e o entendimento claro sobre transação. Esses fatores influenciam no cálculo do tamanho do projeto (MONTEIRO e BELCHIOR, 2006).

3.2. MELHORIA DE PROCESSO DE SOFTWARE

A preocupação com a qualidade sempre teve como intuito, colaborar na eliminação de problemas encontrados nos produtos de software. Da mesma forma, observou-se que poderia diminuir a propagação de erros durante o desenvolvimento do produto, através da monitoração de indicadores de qualidade que poderiam indicar o status do processo. Os métodos para análise e solução de problemas são alguns dos recursos existentes para melhoria da qualidade.

O pensamento científico serve de base para os métodos de análise de solução de problemas. Para auxiliar visualizar e solucionar problemas utiliza-se o pensamento sistêmico ou pensamento cartesiano, pois são algumas das linhas de pensamento científico. No pensamento sistêmico as propriedades das partes não são propriedades intrínsecas, mas só podem ser compreendidas a partir da consideração da inter-relação de todas as partes de um processo.

O pensamento cartesiano ou analítico consiste em diminuir a granularidade dos problemas a fim de alcançar a compreensão do todo a partir das suas partes. Resumidamente, o pensamento sistêmico significa realizar uma análise no contexto do processo como um todo e o pensamento analítico significa tratar o problema de forma isolada para melhor compreensão (CAPRA, 1998).

A estruturação do método PDCA (*Plan-Do-Check-Action*), surgiu do pensamento cartesiano ou analítico (AGUIAR, 2001). O PDCA proposto por Walter Shewart surgiu na

década de 20 inicialmente como PDSA, onde o "*S*" significava *Study*. Após a segunda guerra foi amplamente difundido na indústria japonesa, já no formato conhecido por William Edwards Deming (1990). Naquela época, a proposta deveria ser iniciada pela definição dos objetivos de melhoria e o planejamento das atividades (*P-Plan*). Em seguida o plano era colocado em prática (*D-Do*), os resultados coletados eram verificados (*C-Check*), e por fim, possíveis ações de refinamento e ajustes (*A-Act*) eram realizadas no processo e o ciclo começava novamente. Ainda nos dias de hoje, tem sido muito utilizado em abordagens tradicionais de implantação do gerenciamento da qualidade total (CAMPOS, 1992). A Figura 3.1 apresenta o modelo de melhoria

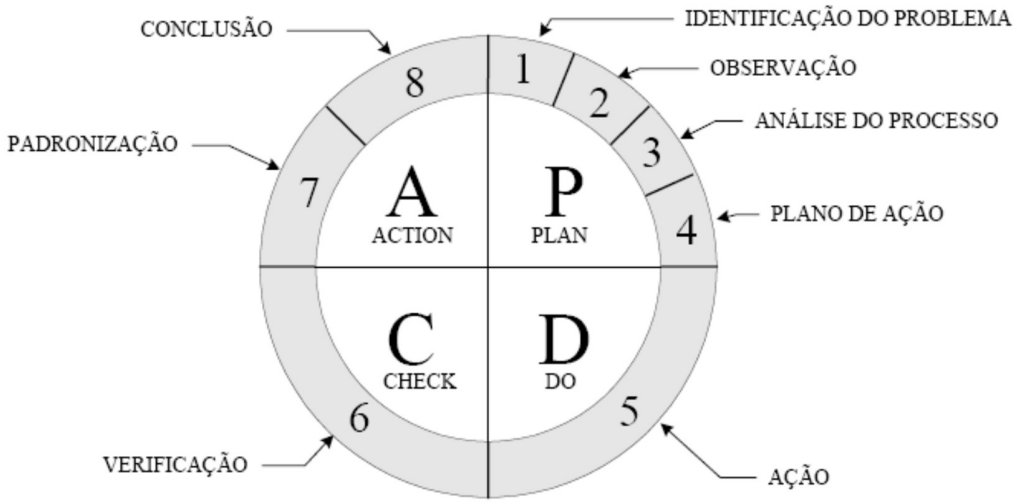


Figura 3.1 – PDCA adaptado (VIANA, 2008)

O QIP – *Quality Improvement Paradigm*, teve como base o PDCA, evoluiu do esforço cooperado entre Centro de Vôo Espacial Goddard da NASA e o Departamento de Ciência da Computação da Universidade de Maryland. Essa cooperação foi realizada dentro do *Software Engineering Laboratory*, denominado SEL em 1976. O principal objetivo era reduzir a taxa de defeito, o custo e o tempo do ciclo de desenvolvimento do software (BASILI *et al.*, 1994).

A proposta do modelo *QIP* é dar suporte à melhoria de processo contínuo e a engenharia dos processos de desenvolvimento (ROMBACH, 1994). A sua aplicação consiste na aprendizagem da organização, onde a organização estabelece uma forma de desenvolver práticas através de suas experiências, e então o conhecimento é capturado e empacotado em um formulário para ser reusado em qualquer parte, dentro de determinados limites (BASILI, 1994) (BASILI e McGARRY, 1998). O modelo *QIP* original é baseado nos princípios de que a disciplina do software é, por sua natureza, evolucionária e experimental e o desenvolvimento de software é baseado na criatividade humana (BASILI, 1994).

O QIP teve uma variação desenvolvida no instituto de *Fraunhofer* na Alemanha, denominado como PIA – *Perfect Improvement Approach*. O modelo tem um ciclo modificado do projeto e as etapas foram refinadas para um nível maior de detalhamento (FRAUNHOFER, 1998). A Figura 3.2 apresenta o modelo de melhoria.

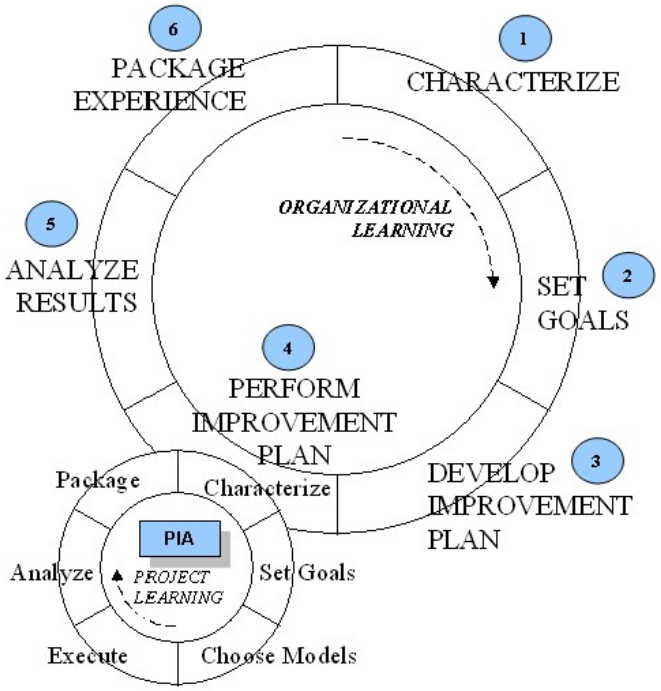


Figura 3.2 – QIP adaptado (VIANA, 2008)

O modelo IDEAL (*Initiating, Diagnosing, Establishing, Acting, Leveraging*) originado pelo PDCA, foi criado pelo SEI – *Software Engineering Institute* com o intuito de orientar iniciativas de melhorias de processo baseadas no CMM – *Capability Maturity Model* (GREMBA e MYERS, 1997). Posteriormente foi revisado e atualizado para ser genérico o suficiente, podendo ser utilizado em outros tipos de iniciativas de melhoria. Como característica forte é que o modelo IDEAL incorporou explicitamente na fase "*learning*" a ideia de *aprendizagem* a partir das ações realizadas. Trata-se de um guia genérico para implementação de melhorias, a quantidade de iterações a serem realizadas e o tempo de cada uma pode variar de acordo com a necessidade e a estratégia adotada pelas organizações em cada esforço de melhoria. A Figura 3.3 apresenta o modelo de melhoria.

Figura 3.3 – IDEAL adaptado (VIANA, 2008)

Outro método que surgiu do pensamento cartesiano ou analítico é o DMAIC *(Define-Measure-Analyse-Improve-Control)*. O método DMAIC foi criado pela empresa Motorola em sua busca por uma estratégia para aumentar o nível de desempenho visando torná-la mais competitiva no mercado. Esse esforço culminou num programa de qualidade

chamado Seis Sigma, método utilizado nos projetos de melhoria dentro da empresa Motorola (HARRY, 1998) (WIGGENBORN, 2000) (PANDE *et al*, 2001). A Figura 3.4 apresenta o modelo de melhoria.

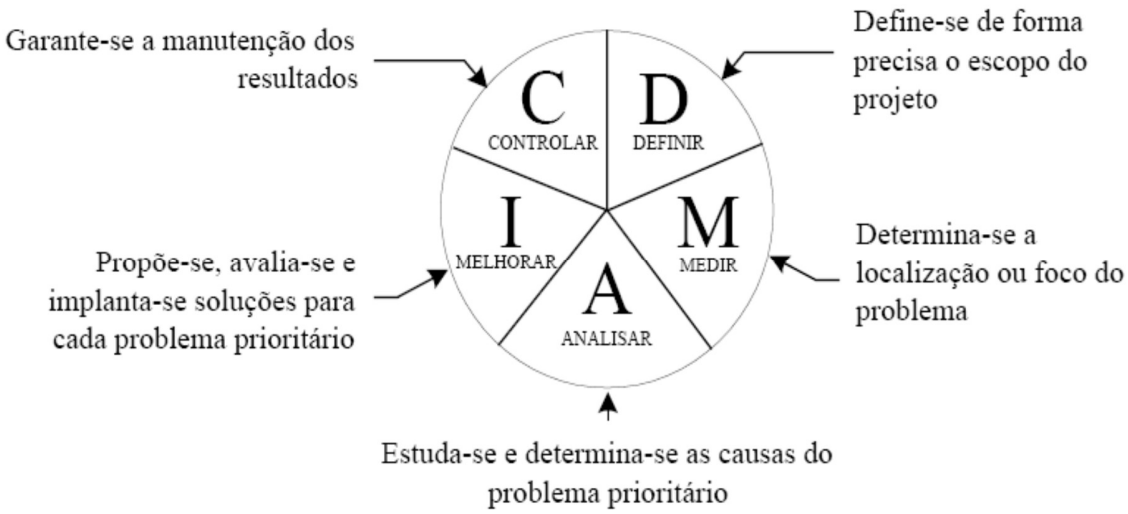


Figura 3.4 – DMAIC adaptado (VIANA, 2008)

O GQM *Goal-Question-Metric* é fortemente recomendado para definição de métricas, pois é uma abordagem orientada a objetivos para medição de produtos e processos de Engenharia de Software. Baseado na definição de medições de forma objetiva serve para identificar, explicitar e especificar de forma precisa os objetivos de medição da organização e também de forma especifica as medições de cada projeto; deve relacionar esses objetivos aos dados necessários, para os definir de forma operacional; e também deve fornecer um *framework* para análise e interpretação dos dados com respeito aos objetivos definidos (BASILI *et al.*, 1994).

O GQM apoia objetivos de medição relativos a qualquer tipo de produto ou processo de software, orientados a qualquer propósito, desde a caracterização até o controle e a melhoria de processo, com foco em todos os aspectos de qualidade, definido a partir de todas as perspectivas e em todos os contextos. Segundo BASILI, o GQM é um mecanismo que serve para definir e avaliar um conjunto de objetivos operacionais, e trata-se de uma abordagem sistemática para customização e integração dos objetivos com modelos de

produtos, processos de software e perspectives de qualidade, com base nas necessidades especificas do projeto e da organização (BASILI, 1999).

Para qualquer programa de medição baseado em GQM, a atividade de análise é definida de forma precisa e explícita através de um objetivo de medição. Os objetivos são definidos de tal forma que possam ser tratáveis de modo operacional, filtrados para um conjunto de questões quantificáveis que são utilizadas para coletar a informação apropriada dos modelos, no sentido de representar as dimensões dos objetivos. A técnica *top-down* é utilizada para derivar as métricas, com base em questões, definidas a partir dos objetivos, formalizando o processo e levando à definição de métricas relevantes (BRIAND *et al.*, 1996).

Esse refinamento é documentado de forma bem detalhada em um plano GQM, registrando todo o critério utilizado na escolha das métricas. A técnica *bottom-up* serve para interpretar os dados coletados, no contexto dos objetivos e questões definidas, considerando os limites e suposições relativas a cada métrica. A estrutura hierárquica de três níveis, denominada como plano GQM é apresentada na Figura 3.5 (BASILI *et al.*, 1994).

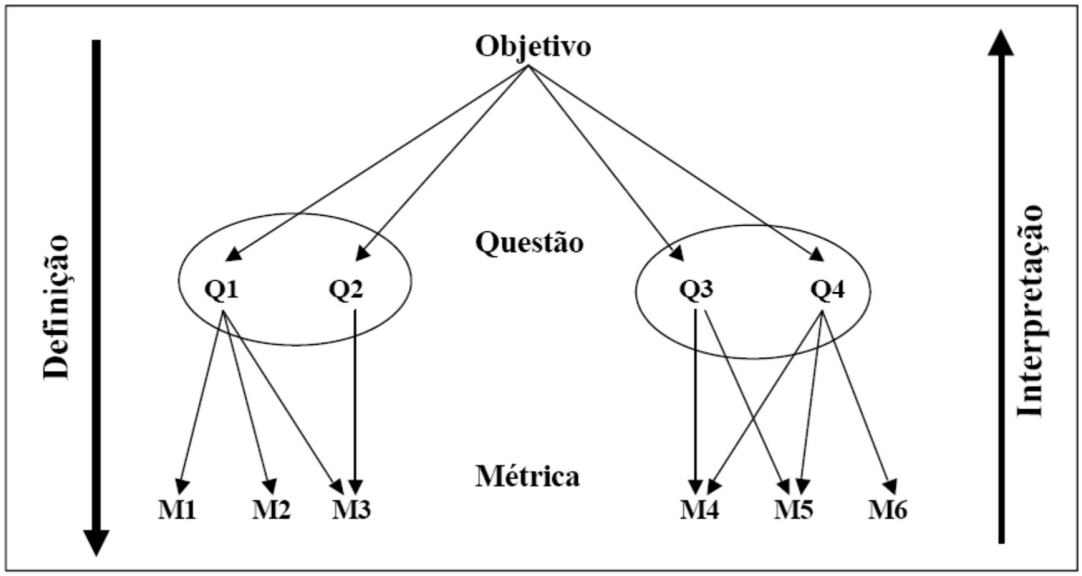


Figura 3.5 – Estrutura hierárquica de três Níveis do GQM (BASILI *et al.*, 1994).

Segundo SOLINGEN, durante a definição dos objetivos de medição, algumas vezes os objetivos se justificam por estarem relacionados a necessidades da organização. Porém, outras são difíceis de serem selecionados e priorizados. Nestas situações, SOLINGEN orienta o pensamento às seguintes questões que podem apoiar a seleção dos objetivos (SOLINGEN e BERGHOUT, 1999): i) Quais são os objetivos estratégicos da organização?; ii) Que forças têm impacto nestes objetivos estratégicos?; iii) Como pode ser melhorado o seu desempenho?; iv) Quais são os principais problemas?; v) Quais são os objetivos de melhoria?; vi) Como atingir seus objetivos de melhoria?; vii) Como atingir seus objetivos de melhoria?; viii) Quais são os possíveis objetivos de melhoria e quais são as prioridades?

Os modelos CMMI são coleções de melhores práticas que ajudam as organizações a melhorar os seus processos. Estes modelos são desenvolvidos por equipes de produto com membros da indústria, do governo e do SEI *(Software Engineering Institute)*. O modelo CMMI-DEV fornece orientação para a aplicação das melhores práticas do CMMI em uma organização de desenvolvimento. O modelo possui 22 áreas de processo, sendo que cada área possui práticas que, uma vez implementadas, satisfazem um conjunto de objetivos importantes para fazer melhorias nesta área (SEI, 2010).

O CMMI é composto por um conjunto de requisitos e guias que ajudam a organização a estruturar seus processos. Possui dois tipos de representação: a representação por estágios corresponde a um conjunto de áreas de processos para definir uma estratégia de melhoria para uma organização ou unidade organizacional, caracterizados por níveis de maturidade. Os níveis de maturidade na representação por estágios correspondem à: 1 – *Initial*, 2 – *Managed*, 3 – *Defined*, 4 – *Quantitatively Managed* e 5 – *Optimizing*. A representação contínua permite que a organização possa selecionar uma determinada área de processo e faça melhorias necessárias. Os níveis de capacidade de uma área de processo correspondem à: 0 – *Incomplete*, 1 – *Performed*, 2 – *Managed*, 3 – *Defined*, O Quadro 3.2 apresenta o CMMI-DEV e os níveis de maturidade (SEI, 2010).

Quadro 3.2 – Modelo CMMI-DEV (SEI, 2010).

Nível de Maturidade	Área de Processo
2	*Project Monitoring and Control (PMC)*
	Project Planning (PP)
	Requirements Management (REQM)
	Measurement and Analysis (MA)
	Process and Product Quality Assurance (PPQA)
	Configuration Management (CM)
	Supplier Agreement Management (SAM)
3	*Integrated Project Management (IPM)*
	Risk Management (RSKM)
	Organizational Process Definition (OPD)
	Organizational Process Focus (OPF)
	Organizational Training (OT)
	Requirements Development (RD)
	Product Integration (PI)
	Technical Solution (TS)
	Validation (VAL)
	Verification (VER)
	Decision Analysis and Resolution (DAR)
4	*Quantitative Project Management (QPM)*
	Organizational Process Performance (OPP)
5	*Organization Performance Management (OPM)*
	Causal Analysis and Resolution (CAR)

O MPS.BR criado pela Associação para Promoção da Excelência do Software Brasileiro (SOFTEX) em 2003 tem como principal objetivo definir um modelo de melhoria e avaliação de processo de software, direcionada para as micro, pequenas e médias empresas de software brasileiras (SOFTEX, 2021). Segundo Weber (2005), o programa MPS foi definido de modo a prover um modelo de referência evolutivo, a exemplo dos modelos CMMI e ISO/IEC 15504, dessa forma possibilita que as organizações que adotem o MPS.BR possam desenvolver um plano de desenvolvimento de longo prazo (WEBER *et al.,* 2005).

O modelo apresenta uma maior granularidade no escalonamento de níveis de maturidade iniciais, permitindo uma estratégia de implementação e evolução mais adequada às condições do mercado brasileiro. Além disso, o MPS.BR é plenamente aderente e compatível com os modelos/normas: ISO/IEC 330xx, NBR ISO/IEC 12207, CMMI-SVC e CMMI-DEV (SOFTEX, 2021).

O grande diferencial do MPS.BR é o Modelo de Negócio (MN-MPS) que incorpora tanto os princípios utilizados pelo SEI para a avaliação de organizações conforme o modelo CMMI, quanto os princípios recomendados pela ISO *(International Organization for Standardization)* para as avaliações de conformidade de terceira parte. Existem duas estratégias de implantação, uma para empresas específicas, outra para grupos de empresas cooperadas – essa estratégia facilita sua utilização por empresas de pequeno porte, por dividir os custos e dispor de linhas de financiamento (WEBER *et al.,* 2005).

O MPS.BR possui quatro componentes: Modelo de Referência (MR-MPS-SW), Modelo de Referência (MR-MPS-SV), Método de Avaliação (MA-MPS) e Modelo de Negócio (MN-MPS). Cada um deles possui documentos e guias específicos. Ainda possui um Guia de Aquisição, orientado às organizações que adquirem software e serviços correlatos.

No MR-MPS-SW a definição dos processos segue os requisitos para um modelo de referência de processo contida na norma ISO/IEC 330xx, declarando o propósito e seus resultados esperados na sua execução (SOFTEX, 2021). Os níveis de maturidade estabelecem uma forma de prever o desempenho futuro dos processos da organização. Um nível de maturidade é um patamar definido de evolução de processos. A dimensão de capacidade é um conjunto de atributos de um processo que estabelece o grau de refinamento e institucionalização com que o processo é executado na organização. À medida que evolui nos níveis, um maior ganho de capacidade para desempenhar o processo é atingido pela organização.

O MPS.BR possui sete níveis de maturidade: G – (Parcialmente Gerenciado), F – (Gerenciado), E – (Parcialmente Definido), D – (Largamente Definido), C – (Definido), B – (Gerenciado Quantitativamente) e A – (Em Otimização). Para cada nível de maturidade

existe um perfil de processos e de capacidade de processos (baseados na ISO/IEC 330xx). A capacidade do processo é representada por um conjunto de atributos de processo descrito em termos de resultados esperados. O atendimento aos Atributos do Processo (AP), pelo atendimento aos Resultados esperados dos Atributos do Processo (RAP) é requerido para todos os processos no nível correspondente ao nível de maturidade. Os Níveis de Maturidade do Modelo MPS.BR é apresentado na Figura 3.6.

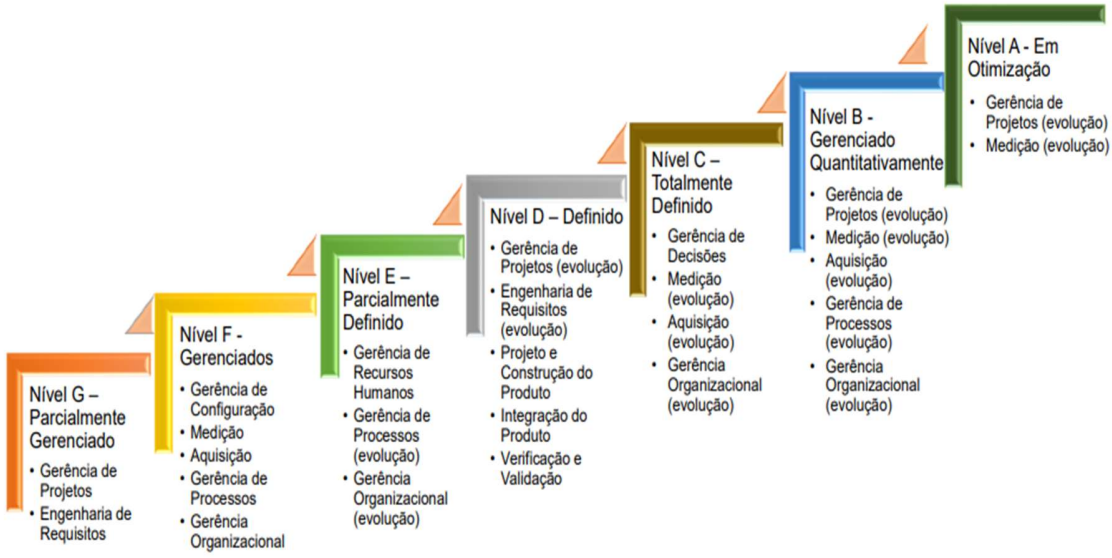


Figura 3.6 – Evolução dos Processos nos Níveis de Maturidade do MPS.BR (SOFTEX, 2021).

Além das normas e modelos que colaboram para MPS, existem alguns fatores que exercem grande impacto no sucesso de programa de melhoria. Segundo DYBA (2003) um desses fatores é o tamanho das organizações. Nessas organizações os colaboradores tiveram uma grande participação e uma maior aprendizagem sobre a abordagem do programa de melhoria.

Os fatores motivacionais também são importantes para o sucesso, os colaboradores precisam ter o sentimento de donos do processo isso os torna mais responsáveis, os benefícios que o MPS traz para a organização precisam ser mais claros através das experiências de sucesso de MPS, é necessário garantir recursos para execução do programa de melhoria (BADDOO e HALL, 2003).

Segundo MONTONI e ROCHA (2010) existem três contextos diferentes que influenciam em programas de melhoria: O Contexto Individual que envolve questões relacionadas com a alta gerência, aos membros da organização e da consultoria em MPS; O Contexto Organizacional que envolve questões relacionadas ao ambiente organizacional e estratégias e políticas organizacionais; e Contexto Tecnológico que envolve questões de processo de software, hardware e recursos de software que apoiam o MPS.

Mas segundo BADDOO e HALL (2003) existem os fatores desmotivadores, a exemplo a pressão exercida pela execução do programa de melhoria, as próprias experiências negativas que os colaboradores possuem e a falta de habilidade dos gerentes de projeto. A nossa proposta, designada por SPIREM-OBK, como posteriormente apresentado, possui uma relação de fatores coletados desses estudos, pois foram considerados relevantes para o sucesso de programas de melhoria.

3.3 ROI (FINANCEIRO)

Quando se propõe uma análise de investimento, as primeiras perguntas que surgem são: Quanto custa o investimento? Quanto tempo será necessário para recuperar o investimento realizado? Os analistas contábeis e/ou financeiros têm sido de fundamental importância nas decisões que consideram os investimentos em TI (Tecnologias de Informação) na busca de respostas em termos financeiros, isto porque eles têm a cultura puramente focada na análise de custo-benefício, medidos em valores monetários (ANANDARAJAN e WEN, 1999).

Segundo Hirschfeld (1998), a priorização de investimento pode ser compreendida quando se realiza um investimento em um bem, em uma aplicação financeira ou em um empreendimento, e geralmente é realizada quando se analisa o recebimento, ou devolução futura, de uma quantia de dinheiro, em relação ao investimento, que corresponde a Taxa Mínima de Atratividade (TMA), também chamada de expectativa ou equivalência. Ou seja, a TMA é representada pelo índice de percentual que o investimento oferece como retorno (HIRSCHFELD, 1998). Segundo Clemente (1998), a TMA representa o **custo de oportunidade** do capital para a empresa que é a opção de investir em uma alternativa de

investimento em detrimento às demais opções existentes, utilizando critério de opção pela alternativa que gerará o maior retorno (CLEMENTE, 1998).

Por se tratar de um indicador econômico, o ROI (*Return on Investment*) é uma ferramenta de administração que mede o desempenho passado e decisões de investimento do futuro; resumindo, mede os resultados históricos e antecipados. A definição do ROI depende da base de investimentos utilizada. Se o patrimônio líquido for usado como base do denominador, a definição é ROE (*Return on equity*), se os ativos forem usados como base, à definição é ROA (*Return on assets*), sendo o numerador o lucro esperado do investimento. O ROI é uma medida que quantifica o retorno produzido pelas decisões de investimento e avalia a atratividade econômica do investimento. Serve de parâmetro para avaliação de desempenho da empresa ou de um determinado projeto em um período de tempo pré-estabelecido. A Tabela 3.1 fornece as possíveis variáveis de cálculo do ROI (CLEMENTE, 1998).

Tabela 3.1 – Equação do ROI adaptada (SOLINGEN, 2004).

Equação do ROI	
Equação 1 $ROI = \frac{VPL}{VPi}$ onde: ***ROI*** = Retorno sobre Investimento ***VPL*** = Valor Presente Líquido ***VPi*** = Valor Presente do investimento	**Equação 2** ROI(%)= (Benefícios – Custos) x 100 Custos Exemplo: ***ROI(%)*** = (R$ 286.000,00 – R$ 20.000,00) x 100 R$ 20.000,00 ***ROI(%)*** = 1330% = R$ 14,30 ou (14 3:1)

Segundo Schaicoski (2002) existem cinco razões para usar o ROI: Força o planejamento; provê uma base para tomada de decisão; avalia oportunidades de desenvolvimento; ajuda na avaliação do desempenho da administração e mede as respostas

do mercado. Também se identifica uma série de usos e aplicações para o uso do ROI, destacando: Melhora na utilização dos recursos; Avaliação dos gastos de capital; Análise da linha de produção e Avaliação dos recursos humanos etc.

As vantagens da utilização do ROI são: o cálculo simplificado; a fácil comparação de projetos diferentes; pode ser aplicado como ponto de equilíbrio, ou seja, ponto que o investimento começa a retornar; demonstra a liquidez do projeto a ser analisado. A desvantagem é que não pode ser utilizado isoladamente na tomada de decisão, mas, sim, como complementar a outros indicadores financeiros (SCHAICOSKI, 2002).

Segundo Suwardy (2003), os investimentos em TI são avaliados a partir de bases puramente financeiras usando técnicas contábeis, como *Pay-back*, ROI, VPL (Valor Liquido Presente), TIR (Taxa Interna de Retorno), como são normalmente avaliados outros projetos dentro das organizações (SUWARDY, 2003). Uma das características do setor de TI é que frequentemente são oferecidos benefícios intangíveis, que são muito difíceis de serem mensurados que, entretanto, se podem tornar fatores decisivos na tomada de decisão.

Para melhor compressão da necessidade de mensurar os benefícios intangíveis, é necessário observar o que mais impacta em qualquer melhoria de processo é o recurso humano, pois a absorção de conhecimentos e a eficiência com que se produz são justificadas pela sua eficácia, portanto a utilização de metodologia de avaliação de treinamento passou a ser fator crítico de sucesso.

3.4 SOLUÇÕES ALTERNATIVAS

Algumas iniciativas são bastantes relevantes dentro do contexto de estimativas de ROI em MPS. Aqui, apresenta-se dois trabalhos de extrema importância, mesmo analisando dados *post-mortem* para o ROI. Evidenciando ao mercado de software, que realmente vale a pena investir em melhoria de processo de software, pois traz valores tangíveis e intangíveis.

3.4.1 ROI EM MELHORIA DE PROCESSO DE SOFTWARE

Segundo RICO (2004), o ROI é uma abordagem amplamente usada para medir o valor de uma melhoria de processo ou uma nova tecnologia de produto. O ROI é uma ferramenta simples e poderosa para analisar custos e benefícios. Um grande ROI indica dinheiro bem gasto. Existe uma variedade de métodos de MPS para processos como: suporte, programas de treinamento, gerenciamento, e seleção de fornecedor. Os exemplos incluem Inspeções, PSP^{sm}, TSP^{sm}, SW-CMM®, ISO 9001, e CMMI®. Segundo RICO, a aplicação de seis métricas de custos durante a implementação de tais modelos, obtiveram um ROI significante como apresentado na Figura 3.7 (RICO, 2004).

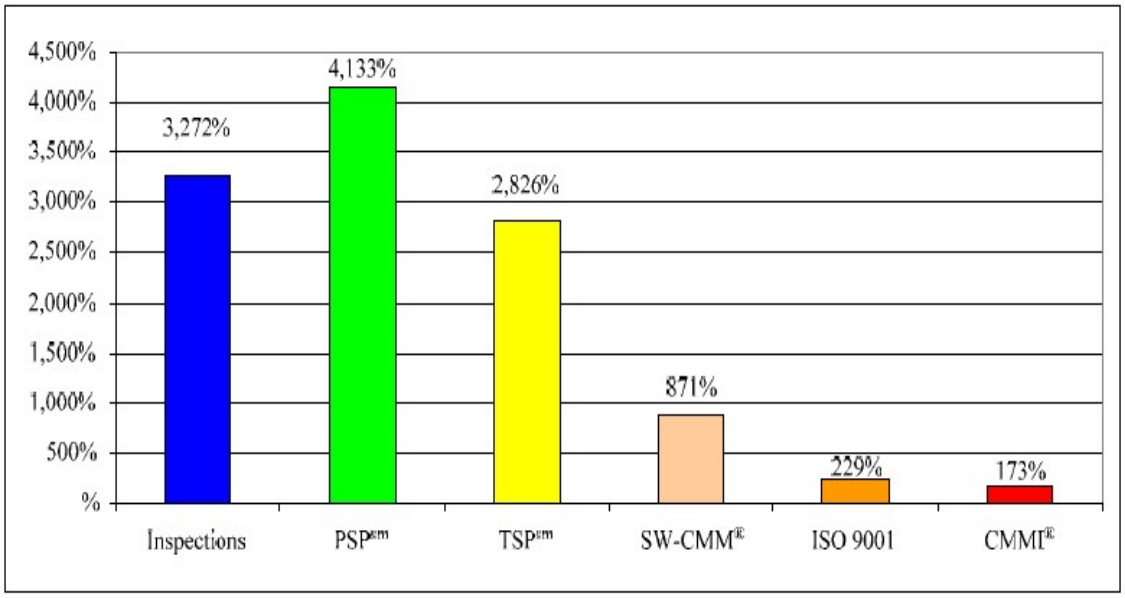


Figura 3.7 – ROI dos vários métodos de MPS mostram o retorno sobre investimento diminuindo da esquerda para a direita (RICO, 2004).

As métricas de RICO para calcular o ROI são projetadas para medir o valor econômico de uma MPS. Cada métrica do ROI é um indicador relevante de quanto uma MPS vale à pena. Existem seis métricas básicas relacionadas ao ROI, sendo elas: custos, benefícios, relação de benefícios/custo ou B/CR, retorno sobre investimento ou ROI, valor líquido atual ou *Net Present Value*, e *Break even Point* ou BEP. Cada métrica do ROI deve

ser analisado de forma individual. Por exemplo, os custos podem ser altos ou os benefícios podem ser insignificantes, distorcendo a relevância de outras métricas.

David Rico tem direcionado seus experimentos para justificar o ROI em metodologias ágeis, realizando comparações entre as metodologias convencionais. A Figura 3.8 apresenta o resultado de uma pesquisa em 69 estudos de caso de metodologia ágil, mas somente 29 desses estudos comentavam sobre ROI. Apesar da porcentagem de 1,788% e 3,103%, a metodologia ágil apresentou um ROI significante, em comparação com o custo de investimento.

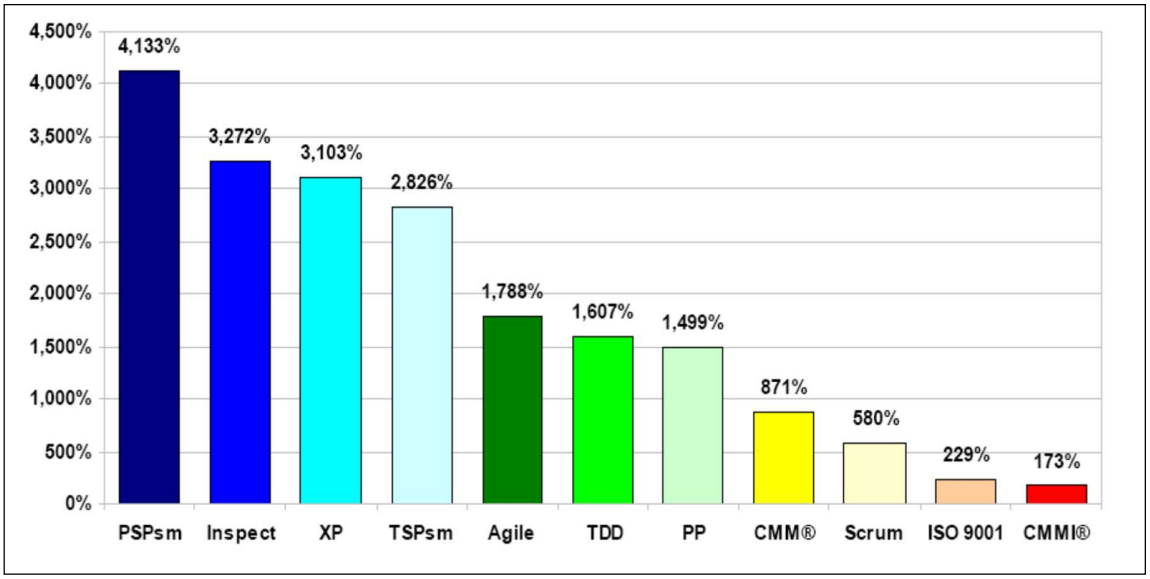


Figura 3.8 – ROI de Metodologias Ágeis versus Metodologias tradicionais.
Fonte: (RICO, 2006)

RICO concluiu que nem todos os métodos ágeis e tradicionais foram criados iguais, existem armadilhas na utilização dos métodos que podem levar para um baixo ROI, e existem lições a serem aprendidas dos melhores métodos do software. Entretanto, não se deve comparar os métodos otimizados para a produtividade e qualidade (onde os custos são mais elevados), com os métodos otimizados para medir a satisfação, o sucesso do projeto. É importante notar que o poder de métodos ágeis não está em minimizar os custos de ciclo de vida, mas sim em maximizar o valor de negócio através das entregas bem-sucedidas do software em face ao risco (RICO, 2006).

Uma das questões bastante pertinentes e desafiadoras para as organizações é determinar o valor dos benefícios em treinamento e desenvolvimento de recursos humanos, melhoria de desempenho, mudanças, qualidade e tecnologia. A Metodologia ROI criada por Jack Phillips (2007) possui uma abordagem viável, acurada e de credibilidade para as questões de responsabilidade financeira para vários tipos de organizações (PHILLIPS, 2007). Por ser muito flexível, diversificada e abrangente a Metodologia ROI passou a ser utilizada em vários países.

A implementação da metodologia gerou, coletou e processou as seguintes categorias de dados: Reação/satisfação em relação ao projeto; Aprendizagem das habilidades e do conhecimento necessário ao sucesso do projeto; Aplicação e implementação do projeto; Impacto no negócio e consequências do projeto; ROI; Intangíveis; Custo do Projeto (PHILLIPS, 2007).

Alguns fatores são fundamentais para que um ROI seja viável, é necessário equilibrar questões como: a praticidade, simplicidade, credibilidade e confiabilidade. A metodologia ROI contempla esses desafios e, além disso, existe uma preocupação com os públicos-alvo que serão afetados, por exemplo (PHILLIPS, 2007):

- **Gerentes de projeto:** precisam de uma abordagem de fácil compreensão para a mensuração. Se algo parecer confuso e complexo, a equipe poderá se sentir frustrada, supondo que o estudo de ROI não pode ser realizado ou que é muito dispendioso.

- **Clientes:** os apoiadores e financiadores do projeto precisam de um processo que forneça resultados quantitativos e qualitativos, bem como um processo que mereça confiança.

- **Pesquisadores:** os pesquisadores em mensuração e avaliação necessitam de um processo que possa ser replicado de um projeto para outro, um processo confiável que resulte nas mesmas medidas ainda que o projeto tenha sido avaliado por dois profissionais diferentes.

3.5 TRABALHOS RELACIONADOS

Solingen (2004) apresenta abordagens de como motivar e envolver gerentes em programa de MPS, já mencionados (seção 1.1). Portanto, MPS é um investimento para que os benefícios excedam os seus custos. Segundo o pesquisador, um argumento frequente na prática de software é que a medição dos benefícios de MPS é impossível, ou pelo menos difícil. As organizações acham fácil medir o custo pela medição do esforço, mas apresentam dificuldades em medir os benefícios. No entanto, isso é devido a um grave equívoco de medição de custos.

Os custos são muito mais amplos do que somente o esforço. Custo envolve também outros recursos, tais como espaço de escritório, consultoria, treinamento, viagens, infraestrutura, etc. Geralmente quando as organizações calculam o custo, usam uma taxa fixa de hora que se aproxima do valor real do custo. Nada mais é do que uma estimativa de custo. O pesquisador afirma que se concordarmos em aproximar o valor real do custo é suficiente, e se concordarmos com o procedimento de estimativa, nós podemos medir benefícios da mesma forma que medimos custo. Medição de benefícios é, portanto, tão fácil como medir o custo.

O envolvimento das partes interessadas na estimativa dos benefícios é muito importante, pois possuem diferentes pontos de vista, não somente a observação de um número qualquer. Isto significa que não basta apenas medir o esforço das atividades de MPS, mas identificar esse valor do MPS e tendo esse valor como próprio benefício. A utilização do *GQM* em um programa de medição orientada à objetivos, abordou as interrupções durantes as atividades dos desenvolvedores.

O programa de medição teve como objetivo descobrir as razões para as interrupções dos desenvolvedores com o objetivo de reduzi-los. Foram detectadas 60 horas de interrupções durante o programa e com isso, a conscientização por parte da equipe aumentou sobre essas interrupções e consequentemente aumentou também a produtividade (SOLINGEN, 2004).

Solingen continuou as suas pesquisas e em 2009 publicou o resultado de uma pesquisa onde ressalta que forneceu uma visão detalhada de publicações com reais resultados da medição de aplicações práticas de MPS e medição de ROI. O estudo incluiu 20 casos, onde apresentou uma média de ROI de 7,0 e uma mediana 6,6. Isso indica que o lucro líquido sobre o investimento em MPS, é de aproximadamente U$ 7,00 para cada U$ 1,00 investido.

O pesquisador chegou à conclusão de que a expressão "*valor*" é crucial, e a engenharia de software e os seus aperfeiçoamentos constituem, muitas vezes, grandes investimentos para as organizações. E considera importante expressar qualquer esforço de engenharia de software e os seus benefícios em termos financeiros (SOLINGEN, 2009).

A abordagem de melhoria mais conhecida na manufatura é provavelmente o Six Sigma. Esta abordagem implanta um conjunto de métodos de gestão da qualidade, fortemente dependendo técnicas estatísticas, e cria uma infraestrutura especial na organizacional de pessoas que são especialistas em aplicar esses métodos. Desencadeia uma série interminável de projetos de melhoria para estabilizar o desempenho do processo, revelando causas dos problemas a fim de resolvê-los.

Todos esses projetos têm como objetivo produzir benefícios financeiros quantificados. Os projetos começam com um conjunto sumarizado de problemas para resolver e o valor para cada solução, e termina com as ações de implementação para assegurar a entrega contínua dos benefícios. A utilização do Six Sigma em engenharia de software significa:

- Incorporar a mensuração do valor diretamente em tudo o que tentar melhorar;
- Fornecer conhecimento sobre como nossas atividades realmente contribuem;
- Ter comprometimento na gestão, porque vamos ser capazes de indicar e medir valores importantes;
- Desafios tecnológicos e métodos para provar os benefícios da vida real antes de serem implantados;

- Conduzir sucessos repetidos porque as melhorias bem-sucedidas em uma área vão ser colocadas no topo da lista para repetir em outras áreas;

- Ajudar a estabilizar o desempenho do processo, deslocando o foco do aumento de um indicador significa diminuir sua propagação;

Estabelecer habilidades estatísticos e de gestão da qualidade para as pessoas que conduzem o desenvolvimento de software e melhoria de processos. Porque a engenharia de software é tão fortemente relacionada com o sucesso e fracasso dos negócios, temos de encontrar uma maneira de medir esses valores (SOLINGEN, 2009).

Partindo da premissa que, organizações bem gerenciadas e com processos bem definidos possam garantir a execução de projetos de desenvolvimento de software, que atendam às necessidades dos clientes, dentro do prazo estabelecido, dentro do custo planejado e da qualidade desejada para o produto. Portanto investir em MPS e essencial para chegar nesse nível de organização. Ferreira (2007) afirma que obter certificados ou alcançar altos níveis de maturidade é um resultado de sucesso que justifica o investimento feito em MPS.

Além disso, aspectos relacionados ao negócio, redução de custos de projetos, o aumento da produtividade, a melhoria da qualidade dos produtos, e principalmente a satisfação do cliente, são outras formas de medir o retorno de investimento.

A organização em questão, motivada pelos benefícios de um programa da qualidade e pela observação da necessidade dos seus clientes, iniciou o programa de melhoria de processo em 2003 com o foco na certificação ISO 9001:2000. Após sua certificação em dezembro de 2004, iniciou a implementação de práticas requeridas no nível F do MPS.BR e foi avaliada com sucesso em setembro de 2005. Em seguida buscaram o nível 3 do CMMI na qual foi avaliada com sucesso em julho de 2006. Ferreira relata que algumas medidas básicas já faziam parte da cultura organização antes do projeto de MPS, e que dessa forma, ficou fácil avaliar os ganhos obtidos com as iniciativas.

A organização utilizou técnica de análise do valor agregado (PMI, 2004), para analisar o benefício relacionado às estimativas de custo e tempo. Dois índices de desempenho foram utilizados:

- O índice de desempenho de custo (*Cost Performance Index* – CPI), que indica quando um produto de trabalho é entregue, o quanto do orçamento aprovado foi economizado ou excedido, aumentando a confiança do método de estimativa de custo;

- O índice de desempenho de tempo (*Schedule Performance Index* – SPI), que indica quando um produto de trabalho é entregue, o quanto do cronograma aprovado foi adiantado ou excedido, aumentando a confiança sobre o método de estimativa de tempo.

Para os dois índices de desempenho, a regra funciona quando o valor for 1 o projeto está dentro do orçamento/cronograma, se o valor for <1 (menor que um), está acima do orçamento/atrasado e, se o valor for >1 (maior que um), o projeto está abaixo do orçamento/adiantado.

O retorno de investimento foi bastante significativo, pois o índice de SPI teve um aumento de 11%, ou seja, os projetos terminaram 11% mais cedo que o previsto. O índice CPI teve uma redução de 54%, ou seja, os projetos passaram a custar 54% menos. A produtividade aumentou 57% nas atividades de desenvolvimento e testes. O cálculo do ROI considerava como custo: o treinamento externo, consultoria e avaliações, e como ganho: o lucro dos projetos desenvolvidos durante o programa de MPS. Houve resultado negativo no início, mas os outros benefícios reverteram em lucros financeiros até o patamar de 54% de lucro com a qualidade. A organização cresceu em 287% em colaboradores, 200% em números de projetos e 2695% em faturamento (FERREIRA *et al*, 2007).

3.6 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o "*Estado da Arte*" sobre Estimativas de Software, as suas principais características, técnicas paramétricas, análogas e julgamento de especialistas. A técnica adotada para a nossa proposta do SPIREM-OBK foi a paramétrica, pois oferece calibragem ao longo que a organização adquire mais conhecimento sobre as tecnologias e habilidades das equipes.

Foi apresentado o conceito sobre Melhoria de Processo de Software, foram abordadas as primeiras iniciativas para melhoria de processo de software, como PDCA, QIP, IDEAL, DMAIC, o Paradigma GQM e os modelos de maturidade e capacidade em qualidade de software CMMI e MPS.BR, ambas iniciativas para melhoria de processo de software. E podemos constatar, através dos trabalhos realizados, a grande busca pela evidenciar o ROI em MPS.

Também foi apresentada a utilização do Six Sigma como alternativa de melhoria de processo, muito utilizando em Controle Estatístico de Processo no modelo fabril. Foi apresentado o conceito do ROI financeiro e suas características.

Capítulo 4

Metodologia

Neste capítulo são apresentadas as abordagens metodológicas utilizadas nesse trabalho e suas principais características, a Revisão Sistemática da Literatura e o Estudo de Caso.

4.1 REVISÃO SISTEMÁTICA

Por se tratar de um tema pouco explorado, decidiu-se que, não seria realizada uma revisão convencional da literatura, mas uma Revisão Sistemática para alcançar um maior rigor científico (KITCHENHAM, 2004). O principal objetivo desta Revisão Sistemática é identificar a existência de métodos de estimativas de software que demonstre quando iniciativas de melhoria de processo de software terão retorno de investimento para as organizações.

4.1.1 INTRODUÇÃO

Segundo BIOLCHINI (2005), a Revisão Sistemática é usada para se referir a uma metodologia específica de pesquisa, desenvolvida a fim de coletar e avaliar as evidências disponíveis referentes a um tema focado (BIOLCHINI *et al.*, 2005). Diferente das revisões de literatura convencionais, a revisão sistemática possui um processo que deve ser executado obedecendo a um protocolo pré-estabelecido, exigindo assim um maior rigor na sua execução. Desta forma, o resultado é mais confiável, pode ser auditado e permite repetição (CONTE *et al.*, 2005). Pois segundo Mafra e Travassos (2006), as revisões da literatura, não sendo conduzidos de forma confiável e abrangente, os seus resultados vão possuir pouco valor científico.

As revisões da literatura são conhecidas por não possuírem formalidade, não possuírem planejamento, terem critérios de seleção fracos, não ser repetível, oferecerem pouca confiança e dependerem fortemente dos revisores.

Sendo assim, uma abordagem sistemática de revisão visa a estabelecer um processo formal para conduzir este tipo de investigação, evitando a introdução de eventuais problemas da revisão de literatura informal (SANTOS, 2008).

A calibragem das palavras-chave faz parte de uma revisão sistemática. Dieste e Padua (2007) analisam o uso de termos como o objetivo de encontrar estudos experimentais

na literatura e analisa o custo-benefício do uso de diferentes combinações de palavras-chave. Segundo Dieste e Padua (2007), é aceitável uma taxa de 72% a 80% de sensitividade (refere-se ao total de artigos identificados dentro do universo de busca) e 15% a 25% de precisão (refere-se ao total de artigos realmente relevantes dentro dos artigos encontrados pela busca). Mas existem alguns problemas nos mecanismos de busca que devem ser considerados: recursos bibliográficos limitados, problemas com o algoritmo de busca, falha em reconhecimento de plural e textos completos ou *abstracts* (resumos) incompletos.

Dieste e Padua (2007) fazem considerações sobre os termos utilizados em buscas: o uso combinado da palavra-chave principal com sinônimos mais proximamente relacionados, melhora as propriedades da busca (sensitividade e precisão); o uso apenas da palavra-chave principal para procurar nos títulos e *abstracts* não é uma má estratégia; para reduzir o número de artigos relevantes não encontrados através da palavra-chave principal, devem-se utilizar sinônimos próximos e geralmente aceitos utilizados com frequência; se for adicionado termos mais gerais à busca do que aos sinônimos mais frequentes e aceitos, podem-se detectar mais artigos relevantes, mas também pode aumentar o número de artigos irrelevantes; não deve se utilizar somente os sinônimos da palavra-chave principal, pois eles podem omitir um conjunto expressivo de resultados importantes.

4.1.2 PROCESSO DE APOIO

Caracterizado como um tipo de estudo secundário (KITCHENHAM, 2004), a revisão sistemática possui processo de pesquisa com um conjunto de passos metodologicamente bem definidos de acordo como o protocolo estabelecido (BIOLCHINI *et al.*, 2005) apresentado na Figura 4.1.

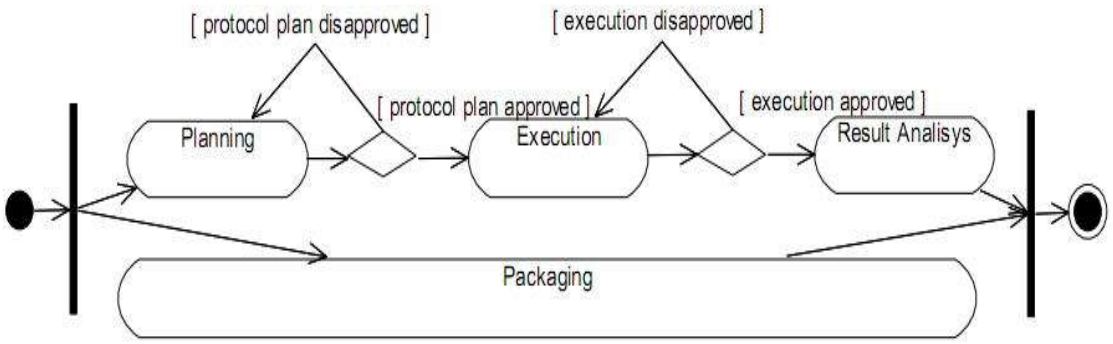


Figura 4.1 – Processo de Revisão Sistemática (BIOLCHINI *et al*., 2005)

Durante a fase do planejamento, os objetivos da pesquisa são identificados e um protocolo de avaliação é definido. Esse protocolo especifica a principal questão de pesquisa e os métodos que serão utilizados para executar a pesquisa. A fase de execução envolve estudos primários de identificação, seleção e avaliação de acordo com o critério de inclusão e exclusão, estabelecidas no protocolo da pesquisa. Uma vez que foram selecionados os estudos, os dados são extraídos e sintetizados durante a fase de análise dos resultados. Sendo assim, quando cada fase for executada, os resultados devem ser armazenados. Portanto o empacotamento da revisão sistemática é realizado durante todo o processo.

4.1.3 DEFINIÇÃO DO PROTOCOLO

A definição do protocolo para revisão sistemática da literatura serve para guiar a execução da pesquisa. O protocolo é composto pelos seguintes itens: contexto do estudo, objetivos, questões de pesquisa, escopo, idiomas, método de busca, procedimentos de seleção, critérios para inclusão das publicações, procedimentos para extração de dados e procedimentos para analisar os resultados (SANTOS, 2008). Nas seções seguintes serão apresentadas as descrições desses itens.

4.1.3.1 CONTEXTO

No cenário internacional, diversas publicações de iniciativas em melhoria de processo de software demonstram certa preocupação ao demonstrar o ROI para justificar a contínua busca por melhoria de processos de software. No cenário nacional, a instituição organizadora do programa MPS.BR busca evidenciar cada vez mais o sucesso dos resultados obtidos das avalições nas organizações de software brasileiras e de iniciativas em parcerias internacionais (SOFTEX, ALVES, 2007). No entanto, tais resultados só são contabilizados após o término das iniciativas em melhoria de processo.

A estimativa do ROI em inciativas de melhoria de processo de software, levando em consideração alguns fatores relevantes que impactam na execução de tais melhorias, fomentaria mais informações aos empresários na tomada de decisão com uma margem maior de segurança.

4.1.3.2 OBJETIVO

Objetivo (não estruturado)
O principal objetivo desta Revisão Sistemática é identificar a existência de métodos de estimativas de software que demonstre quando iniciativas de melhoria de processo de software terão retorno de investimento para organizações de software.

Objetivo (estruturado) baseado no paradigma GQM (BASILI *et al*, 1994)
Analisar relatos de experiência e publicações científicas através de um estudo baseado em revisão sistemática.

Com o propósito de identificar os métodos de estimativas de software que demonstrem quando iniciativas de melhoria de processos de software terão retorno de investimento para organizações de software.

Com relação aos aspectos relevantes que colaboram para a definição, implantação, execução e melhoria dos processos de software em organizações e corporações.

Do ponto de vista dos pesquisadores.

No contexto acadêmico e industrial com foco em programas de melhoria de processos de software em organizações e corporações.

4.1.3.3 QUESTÃO DE PESQUISA

Uma única questão foi definida para atender a pesquisa:

- *Q1* – Quais métodos de estimativa de software que são utilizados para estimar o retorno de investimento em iniciativas de melhoria de processo de software nas organizações de software?

4.1.3.4 ESCOPO DA PESQUISA

Para definir o escopo da pesquisa foram estabelecidos critérios para garantir a viabilidade da execução, acessibilidade aos dados e abrangência do estudo. A pesquisa terá seu início pelas bibliotecas digitais através dos seus engenhos de busca e, quando as publicações não estiverem disponíveis em forma digital, através de consultas manuais em anais.

CRITÉRIOS ADOTADOS PARA SELEÇÃO DAS FONTES

Para as bibliotecas digitais é desejado:

- Possuir engenho de busca que permita o uso de expressões lógicas ou mecanismo equivalente;

- Incluir em sua base publicações da área de exatas ou correlatas que possuam relação direta com o tema a ser pesquisado;

- Os engenhos de busca deverão permitir a busca no texto completo das publicações.

Deve-se garantir que as publicações pertençam a uma das editoras listadas no Portal de Periódicos da CAPES.

Os engenhos de busca utilizados devem garantir unicidade de resultados através da busca de um mesmo conjunto de palavras-chaves. Quando isto não for possível, deve-se estudar e documentar uma forma de minimizar os potenciais efeitos colaterais desta limitação.

Serão considerados, também, simpósios patrocinados pela Sociedade Brasileira de Computação (SBC) na área de Engenharia de Software.

RESTRIÇÕES

A pesquisa está restrita à análise de publicações obtidas, exclusivamente, a partir das fontes selecionadas a partir dos critérios mencionados.

O estudo terá a abrangência dos dados disponíveis nas fontes considerando o período de 01 de janeiro de 2000 até 31 de dezembro de 2020.

4.1.3.5 IDIOMAS

Para a realização desta pesquisa foram selecionados o idioma inglês e o português. O idioma inglês foi escolhido pela sua adoção em grande maioria das conferências e periódicos internacionais relacionados com o tema de pesquisa e por ser o idioma utilizado pela maioria das editoras relacionadas com o tema listadas no Portal de Periódicos da CAPES. O idioma português foi escolhido pela sua adoção nas principais conferências e periódicos nacionais da área de Engenharia de Software.

4.1.3.6 MÉTODOS DE BUSCA DE PUBLICAÇÕES

As fontes digitais serão acessadas via Web, através de expressões de busca pré-estabelecidas. Caso não seja possível obter o artigo completo através dos sites de busca, os autores dos artigos deverão ser contatados via e-mail. As publicações das fontes não-digitais serão analisadas manualmente, quando disponíveis, considerando a expressão de busca definida. Foram utilizados os sinônimos das palavras-chave mais comuns em Engenharia de Software.

EXPRESSÃO DE BUSCA

Para artigos em inglês deve-se utilizar a expressão de busca abaixo:
(enterprise OR organization OR organisation OR corporation OR company OR association OR corporate OR organizations OR organisations OR companies OR enterprises OR corporations) AND (return OR ROI OR "return on investment" OR "return of investments") AND ("software process" OR "software processes" OR "process evolution") AND (improvement OR enactment OR execution OR control)

Para artigos em português deve-se utilizar a expressão de busca abaixo:
(empresa OR organização OR corporação OR companhia OR associação OR corporativo OR organizações OR companhias OR empresas OR corporações OR negócio OR indústria OR indústrias) AND (retorno OR roi OR benefícios OR "retorno de investimento" OR "retorno sobre o investimento") AND ("processo de software" OR "processos de software" OR "evolução de processo") AND (melhoria OR execução OR controle)

BUSCA MANUAL

Para realizar a consulta manual, devem-se procurar as palavras-chave relacionadas na expressão de busca nos títulos e resumos (*abstracts*) dos artigos. Nos artigos em português com *abstracts* em inglês, deve-se pesquisar primeiro no *abstract*, em caso de dúvida sobre a seleção do artigo, devem-se pesquisar as palavras-chave no resumo.

4.1.3.7 PROCEDIMENTOS DE SELEÇÃO E CRITÉRIOS

A pesquisa será aplicada por um pesquisador para identificar as publicações relevantes. As publicações identificadas serão selecionadas por dois pesquisadores (incluindo o que fará a busca) através da verificação dos critérios de inclusão e exclusão e de qualidade estabelecidos. Existirá o consenso entre os pesquisadores nas avaliações das publicações selecionadas que apresentarem conflitos.

Em caso de impasse entre os pesquisadores, a publicação será incluída na lista de selecionadas. Em caso de dúvida a publicação sempre será incluída, para mitigar o risco de uma publicação ser excluída em uma das etapas do estudo.

Serão aceitas publicações que descrevam pelo menos provas de conceito e/ou relatos de experiência na academia ou na indústria.

PROCEDIMENTO DE SELEÇÃO

A seleção dos estudos foi realizada em três etapas:

- *1ª Etapa – Seleção e catalogação preliminar dos dados coletados:* A seleção preliminar das publicações será feita pela utilização da expressão de busca às fontes selecionadas. Cada publicação será catalogada em uma base de dados para ser analisado e auditado posteriormente;

- *2ª Etapa – Seleção dos dados relevantes – [1º filtro]:* Com o uso da expressão de busca, a seleção preliminar não garante que todo o material coletado seja útil no contexto da pesquisa, pois sua aplicação está restrita ao aspecto sintático. Para excluir os estudos irrelevantes contidos no conjunto selecionados preliminarmente, devem-se aplicar os seguintes critérios.
    - *CS1* – Não serão selecionadas publicações em que as palavras-chave não estejam presentes na publicação e não haja variações destas palavras-chave (exceto plural).

- *CS2* – Podem ser selecionadas publicações que descrevam utilização de métodos e/ou modelos de estimativas de software que estejam relacionados à melhoria de processos de software.

- *3ª Etapa – Seleção dos dados relevantes – [2º filtro]:* Apesar da limitação imposta pelo 1º filtro, não garante que todo o material coletado seja útil na pesquisa. Sendo assim, após a leitura dos artigos selecionados no 1º filtro, será verificado se as publicações respeitem os critérios a seguir:
  - *CS3* – Só podem ser selecionadas publicações que descrevam uma iniciativa de melhoria de processo de software, descrevam a utilização de métodos e/ou modelos de estimativas de software utilizados para estimar retorno de investimento e que a iniciativa de melhoria de processo de software é real e que demonstre seus benefícios.

4.1.3.8 PROCEDIMENTOS PARA EXTRAÇÃO DOS DADOS

NA SELEÇÃO E CATALOGAÇÃO PRELIMINAR DOS DADOS COLETADOS

Armazenamento das referências completas selecionadas a partir da fonte consultada no repositório de dados do estudo.

NA SELEÇÃO DOS DADOS RELEVANTES

As referências catalogadas serão examinadas com o objetivo de ser submetida aos critérios de seleção dos filtros identificados. Os dados que atenderem aos critérios de seleção deverão ser marcados como "*verificado no [número do filtro]º filtro, passou*", do contrário, o registro deverá ser marcado como "*verificado no [número do filtro]º filtro, não passou no critério [número do critério]*".

EXTRAÇÃO DE DADOS

Os dados extraídos das publicações selecionadas serão armazenados em uma planilha eletrônica e deve conter a seguinte estrutura:

- Dados da publicação:
    - Título,
    - Autor (es),
    - Data de publicação,
    - Referência completa;

- Resumo da publicação;

- Dados derivados das características de interesse declaradas no objetivo do estudo:

    - Geral – Representa uma categoria de itens gerais sobre o artigo. O preenchimento dos itens dessa seção é obrigatório quando a publicação for considerada válida para o estudo baseado em revisão sistemática.

    - Iniciativa de melhoria de processos – Descrição de características da iniciativa de melhoria de processos descrita no artigo podem ser uma estratégia definida e estabelecida ou apenas a abordagem utilizada no contexto do estudo de caso apresentado. O preenchimento dos itens dessa seção é obrigatório quando o artigo for considerado válido para o estudo baseado em revisão sistemática.

    - Caracterização da utilização de métodos e/ou modelos de estimativas de software;

    - Caracterização do tamanho da organização (se pequena, média ou grande);

4.1.3.9 PROCEDIMENTOS PARA ANÁLISE

ANÁLISE QUANTITATIVA

A análise quantitativa será feita pela extração direta dos dados a partir da base de dados com os registros dos achados.

A análise quantitativa consiste em fornecer:

- Uma lista de publicações selecionadas para fazerem parte do estudo;

ANÁLISE QUALITATIVA

Para análise qualitativa utilizou-se como base os dados quantitativos e realizou-se considerações com o intuito de discutir os dados com relação à questão de pesquisa definida.

4.1.4 PLANEJAMENTO

O protocolo apresentado na seção anterior foi utilizado para executar o estudo baseado em revisão sistemática. Algumas etapas são importantes para se realizar o planejamento:

DEFINIÇÃO DO ESCOPO E ESTUDOS PRELIMINARES

Os estudos preliminares identificados em Viana (2008), colaboraram como prospecção do tema de pesquisa. Tendo o estudo como premissa, passou-se a definir quais palavras-chave poderiam ser utilizadas para a pesquisa. Para definir a expressão de busca, definiu-se "*software process improvement*", "*ROI estimate*" e "*ROI estimating*". Nos primeiros resultados dos testes da expressão, pôde-se notar que, apesar das palavras-chave serem abrangentes, quando se juntava as palavras-chave retornam poucos artigos e assim buscava-se uma melhor utilização das palavras-chave.

IDENTIFICAÇÃO DE PUBLICAÇÕES DE CONTROLE E PALAVRAS-CHAVE

Concluído o tema da pesquisa, foi dado início a calibragem da expressão de busca. Trata-se do teste das palavras-chave diretamente na máquina de busca, experimentando uma melhor adaptação. Como as interfaces das máquinas de buscas são diferentes, se faz necessário uma adaptação até que os resultados estejam satisfatórios.

A calibragem consiste em:

1. Definição da máquina de busca;
2. Identificação de publicações para o grupo de controle;
3. Identificação da expressão de busca inicial;
4. Testes da expressão de busca;
5. Análise dos resultados retornados.

A calibragem é executada de forma iterativa, tendo como principal ciclo os passos 2 a 5 até que os resultados sejam considerados satisfatórios. Após duas rodadas utilizando a expressão de busca nas máquinas de buscas *ACM*, *IEEE* e *Scopus* concluiu-se a calibragem e definição da expressão de busca padrão do protocolo.

("*roi estimate*" OR "*return on investment estimate*" OR "*roi estimating*" OR "*return on investment estimating*") AND ("*software process improvement*" OR "*spi*")

DEFINIÇÃO DAS MÁQUINAS DE BUSCA

Verificou-se que durante a execução dos testes do protocolo e da expressão de busca, a base de dados da *IEEE* (http://ieeexplore.ieee.org/) apresentava um número maior de artigos em relação às maquinas de busca da *ACM* (http://www.acm.org) e *Scopus* (http://www.scopus.com/). Os números reduzidos de resposta das máquinas de busca *ACM* e *Scopus* demonstravam o quão raro eram as publicações sobre o tema em questão.

Foi realizado uma busca manual utilizando os critérios do protocolo nos anais do Simpósio Brasileiro de Engenharia de Software (SBES) desde 2000 e do Simpósio Brasileiro de Qualidade de Software (SBQS) desde 2002. Foi considerado também as publicações dos Workshops do MPS.BR (WAMPS) pois possuem publicações relevantes.

EXPRESSÕES DE BUSCA NA BIBLIOTECA DIGITAL ACM

(“*roi estimate*” OR “*return on investment estimate*” OR “*roi estimating*” OR “*return on investment estimating*”) AND ("*software process improvement*” OR “*spi*”)

EXPRESSÕES DE BUSCA NA BIBLIOTECA DIGITAL IEEE

(((*'roi estimate*' OR '*return on investment estimate*' OR '*roi estimating*' OR '*return on investment estimating*')) AND ‘*software process improvement*' OR '*spi*')

EXPRESSÕES DE BUSCA NA BIBLIOTECA DIGITAL SCOPUS

TITLE-ABS-KEY (("*ROI estimate*" OR "*Return on investment*" OR "*ROI estimating*" OR "*Return on investment estimating*") AND ("*software process improvement*" OR spi)) AND SUBJAREA (mult OR ceng OR CHEM OR comp OR eart OR ener OR engi OR envi OR mate OR math OR phys OR mult OR arts OR busi OR deci OR econ OR psyc OR soci) AND PUBYEAR AFT 1979

IDENTIFICAÇÃO DO PERÍODO DE BUSCA

A pesquisa está restrita à análise de publicações obtidas, exclusivamente, a partir das fontes selecionadas a partir dos critérios mencionados. O estudo teve a abrangência dos dados disponíveis nas fontes considerando o período de 01 de janeiro de 2000 até 31 de dezembro de 2020.

4.1.5. EXECUÇÃO DO PROTOCOLO DE PESQUISA

Dando início a seleção das publicações, a expressão de busca foi executada na máquina de busca IEEE, retornando 221 publicações equivalentes a 87%. A segunda máquina de busca foi Scopus, retornando 13 publicações equivalentes a 5%. A terceira máquina de busca foi ACM e retornou 20 publicações equivalentes a 8%, totalizando 254 publicações conforme apresentados no gráfico da Figura 4.2.

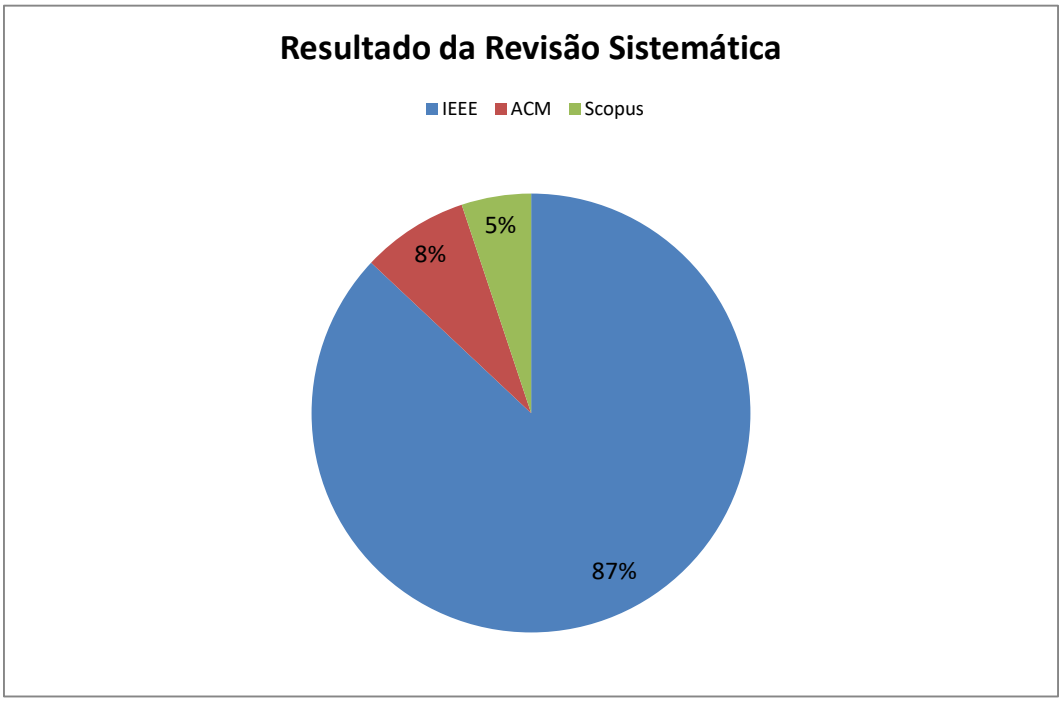


Figura 4.2 – Total de Artigos retornados pela Expressão de Busca. Fonte: Próprio autor.

Na segunda etapa de seleção das publicações foram lidos os títulos e resumos (*abstract*) de cada publicação. Baseado nos critérios, foram selecionadas 37 publicações tendo como distribuição 30 publicações da IEEE equivalentes a 81%, 3 publicações da Scopus equivalentes a 8% e 4 da ACM equivalentes a 11%, conforme o gráfico na Figura 4.3. As leituras foram realizadas por dois avaliadores, as divergências encontradas não foram descartadas e sim inclusas na base de dados dos artigos.

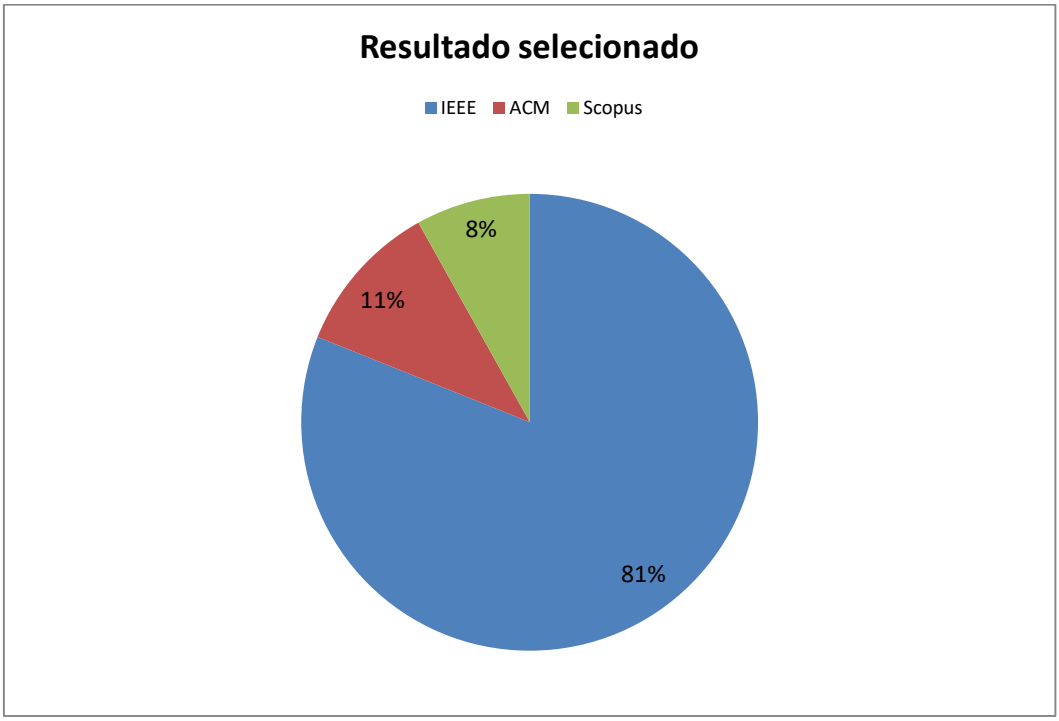


Figura 4.3 – Total de Artigos que passaram no Primeiro Filtro. Fonte: Próprio autor

A terceira etapa é mais rigorosa e crucial para atender os objetivos da pesquisa, pois trata da leitura completa das publicações. Entretanto, nenhuma publicação foi encontra que apresentasse alguma evidência da utilização de métodos de estimativas de ROI para MPS. As publicações foram catalogadas e um resumo das características foi registrado e serão apresentadas na seção 4.1.7.2.

4.1.6. AVALIAÇÃO DO RESULTADO DA PESQUISA

As publicações selecionadas não satisfizeram o principal objetivo da pesquisa, que era identificar nas publicações algum modelo de estimativa de ROI para MPS. Contudo, algumas publicações evidenciam o ROI após a implementação de melhorias. Esse foi o principal motivo para que as publicações fossem apresentadas, foram selecionadas oito publicações mais relevantes para este estudo. Acredita-se que o conhecimento gerado neste trabalho possa ser acrescido utilizando como base as experiências dessas publicações.

4.1.7. RESULTADO DA PESQUISA

Nesta seção são apresentados os seguintes resultados. Em primeiro lugar, todas as publicações selecionadas na execução do protocolo de pesquisa. Em segundo lugar, como não houve publicação que atendesse a pesquisa, os dados extraídos dessas publicações são apresentados para se entender as características das publicações.

4.1.7.1. LISTAGEM DAS PUBLICAÇÕES

O Apêndice B apresenta o resultado geral de todas as publicações encontradas na execução do protocolo e o resultado da seleção de cada publicação.

4.1.7.2. INFORMAÇÕES EXTRAÍDAS DAS PUBLICAÇÕES

Nesta seção os dados extraídos das publicações selecionadas são apresentados a seguir.

1	Dados da publicação
Título:	Measuring the ROI of software process improvement
Autor(es):	van Solingen, R.;
Data de publicação:	2004
Referência completa	van Solingen, R.; "Measuring the ROI of software process improvement," Software, IEEE, vol.21, no.3, pp. 32- 38, May-June 2004
http://dl.acm.org/citation.cfm?id=1437089 *http://www.scopus.com/inward/record.url?eid=2-s2.0-3042675818&partnerID=40&md5=bd23ad0d54d4a967fb81e1cc1f00e4e3*	
Resumo da Publicação	
O autor relata qual a relevância em analisar o ROI em MPS. Apresenta também uma pesquisa que mostra o ROI em MPS de 20 organizações. O autor argumenta sobre a importância de mensurar os benefícios da mesma forma que se mensura custos. E que o envolvimento de *stakeholders* no processo de mensuração dos benefícios é importante, pois eles sabem melhor o valor desses benefícios.	

Dados extraídos da publicação
Argumentos importantes para justificar o investimento em MPS: (i) convencer gerentes a investir dinheiro e esforço, e convencê-los de que MPS pode ajudar a resolver problemas estruturais; (ii) estimar o quanto de esforço deve-se investir para resolver um determinado problema ou estimar se determinados benefícios pretendidos valem o seu custo; (iii) decidir qual a melhoria de processos para implementar em primeiro lugar. Muitas organizações devem priorizar devido ao calendário e restrições de recursos; (iv) programas de melhoria contínua, orçamentos de MPS são atribuídos e discutidos anualmente, os benefícios devem ser explícitos e as organizações devem mostrar ROI suficiente, ou a continuação está em risco; (v) sobreviver, porque qualquer investimento em uma organização deve ser questionado sob o seu retorno.
Iniciativas de melhoria de processos
Utilizou um programa de métrica baseado no GQM para entender o tempo gasto em interrupções durante o projeto de desenvolvimento. No final houve uma conscientização por partes dos colaboradores e os projetos passaram a terminar antes do prazo.
Métodos/Modelos de estimativas de software
Não identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

2	Dados da publicação
Título:	Successful process implementation
Autor(es):	Borjesson, A.; Mathiassen, L.
Data de publicação:	2004
Referência completa	Borjesson, A.; Mathiassen, L.; , "Successful process implementation," Software, IEEE , vol.21, no.4, pp. 36- 44, July-Aug. 2004
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1309645&isnumber=29063	
Resumo da Publicação	
Os autores relatam a utilização de IDEAL para conduzir as melhorias em uma organização. O programa de melhoria seguia a abordagem do IDEAL, logo após o início, as iniciativas passaram por um ou mais ciclos de diagnóstico de problemas, estabelecendo melhorias focadas, atuando para melhorar e aprender a partir dos resultados. Os processos foram mapeados e suas fraquezas foram identificadas. As iniciativas de melhorias foram planejadas para cada processo e lições foram aprendidas.	

Dados extraídos da publicação
As lições aprendidas foram: i) Avaliar criticamente o novo processo a partir de um ponto de vista de fácil adoção; ii) Examinar os papéis que as partes interessadas devem desempenhar durante a implementação; iii) Escolher uma estratégia de implementação que se adapte a iniciativa; iv) Avaliar e resolver os riscos de implementação; v) Delinear um plano inicial para a implementação; vi) Para ter sucesso com MPS, precisa de um compromisso sério dos principais interessados; vii) Iniciativas MPS que executam apenas uma única iteração nunca entrar na fase em que o novo processo é exposto a praticar. Iniciativas MPS que executam várias iterações são mais propensos a entrar e passar através da fase em que os praticantes resistem à mudança; quando o processo definido satisfaz na prática, os engenheiros de processo podem aprender e reagir. viii) Deve-se esperar certo nível de caos em iniciativas SPI. Caos é muitas vezes um sinal de que o processo de implementação está em seu caminho e que está prestes a receber valiosas informações que irão ajudar a ter sucesso. ix) Alcançar o sucesso em MPS é difícil se os engenheiros de processo não estiverem envolvidos na fase Ação do modelo IDEAL x) As iniciativas de MPS que têm como alvo várias unidades e projetos, muitas vezes carecem de recursos e motivação necessária para enfrentar a complexa mudança das questões envolvidas.
Iniciativas de melhoria de processos
Utilização do IDEAL para conduzir a melhoria
Métodos/Modelos de estimativas de software
Não identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

3	Dados da publicação
Título:	An empirical investigation of the key factors for success in software process improvement
Autor(es):	Dyba, T.
Data de publicação:	2005
Referência completa	Dyba, T.; "An empirical investigation of the key factors for success in software process improvement," Software Engineering, IEEE Transactions on , vol.31, no.5, pp. 410-424, May 2005. doi: 10.1109/TSE.2005.53
URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1438376&isnumber=30973*	

Resumo da Publicação
O Autor apresenta que, entender como implementar MPS com sucesso é sem dúvida a questão mais desafiadora nos dias de hoje. Este artigo estende e integra modelos da pesquisa anterior, realizando uma investigação empírica dos fatores-chave para o sucesso em SPI. Um *Survey* quantitativo com 120 organizações de software foi projetado para testar o modelo conceptual e as hipóteses do estudo. Os resultados indicam que o sucesso depende criticamente de seis fatores organizacionais, que explicaram mais de 50 por cento das variações no resultado da variável. A principal contribuição do trabalho é aumentar a compreensão da influência das questões organizacionais, mostrando empiricamente que eles são pelo menos tão importante quanto à tecnologia para ter sucesso com MPS e, assim, fornecer aos pesquisadores e profissionais, novos conhecimentos importantes a respeito dos fatores críticos de sucesso no SPI.
Dados extraídos da publicação
Seis Fatores organizacionais contribuem para o sucesso de MPS, são: Orientação empresarial Liderança envolvida Participação dos colaboradores Preocupação para a medição Exploração do conhecimento existente Exploração de novos conhecimentos
Iniciativas de melhoria de processos
Não identificado
Métodos/Modelos de estimativas de software
Não Identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

4	Dados da publicação
Título:	A Nationwide Program for Software Process Improvement in Brazil
Autor(es):	da Rocha, A.R.C.; Montoni, M.; Weber, K.C.; de Araujo, E.E.R.
Data de publicação:	2007
Referência completa	da Rocha, A.R.C.; Montoni, M.; Weber, K.C.; de Araujo, E.E.R.; , "A Nationwide Program for Software Process Improvement in Brazil," Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the , vol., no., pp.167-176, 12-14 Sept. 2007 doi: 10.1109/QUATIC.2007.15

URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4335244&isnumber=4335220*
Resumo da Publicação
Os autores confirmam que implementação de MPS baseado em modelos de referência de processo de software e padrões é um esforço complexo e em longo prazo e que requer um investimento de grandes somas de dinheiro. Este artigo descreve o MPS.BR com o objetivo de estabelecer um caminho viável para organizações alcançar benefícios da implementação de MPS em custos razoáveis, especialmente as Pequenas e Médias Empresas (PME). Embora o foco principal do Modelo MPS é PME, o modelo demonstrou ser adequado para suportar implementação MPS em avaliação em grandes organizações.
Dados extraídos da publicação
Apresenta a evolução do Modelo MPS.BR e quantidade das avaliações
Iniciativas de melhoria de processos
Não identificado
Métodos/Modelos de estimativas de software
Não identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

5	Dados da publicação
Título:	Implementing Software Process Improvement Initiatives in Small and Medium-Size Enterprises in Brazil
Autor(es):	Santos, G.; Montoni, M.; Vasconcellos, J.; Figueiredo, S.; Cabral, R.; Cerdeiral, C.; Katsurayama, A.E.; Lupo, P.; Zanetti, D.; Rocha, A.R.;
Data de publicação:	2007
Referência completa	Santos, G.; Montoni, M.; Vasconcellos, J.; Figueiredo, S.; Cabral, R.; Cerdeiral, C.; Katsurayama, A.E.; Lupo, P.; Zanetti, D.; Rocha, A.R.; , "Implementing Software Process Improvement Initiatives in Small and Medium-Size Enterprises in Brazil," Quality of Information and Communications Technology, 2007. QUATIC 2007. 6th International Conference on the, vol., no., pp.187-198, 12-14 Sept. 2007. doi: 10.1109/QUATIC.2007.22
URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4335246&isnumber=4335220*	
Resumo da Publicação	
Os autores relatam que abordagens de implementação de processo adequadas são fundamentais para as Pequenas e Médias Empresas (PME) para obter o máximo dos	

benefícios de MPS. A COPPE / UFRJ realizou prestação de serviços em consultoria de MPS para a indústria de software brasileira por mais de duas décadas. Para tratar os fatores que têm influência sobre o sucesso de MPS, desenvolveu uma abordagem para implementar iniciativas de MPS com base no modelo MSP.BR, através da adopção de uma estratégia de MPS (chamado de SPI-KM) e com o apoio de um ambiente centrado em Processo de Engenharia de Software chamado Taba. Este artigo apresenta a trabalho realizado em um grupo de PME brasileiras com o objetivo de implantar as iniciativas modelo MPS a estratégia SPI-KM apoiada pela ferramenta Taba. Apresenta também as lições aprendidas com a implementação de MPS nesse grupo de PME e também todas outras lições recolhidas de experiências anteriores de MPS.

Dados extraídos da publicação

Lições aprendidas com a utilização da ferramenta Taba:

i) Iniciativas de MPS devem melhorar de forma eficaz os processos de software;
ii) Não terá sucesso sem um líder;
iii) Compromisso é crucial;
iv) Sem cérebro, sem ganho;
v) A iniciativa de MPS é facilitada pela infraestrutura do processo de software;

Lições aprendidas

i) A internalização das vantagens e benefícios do programa de melhoria é favorecida pela constante presença de consultores na organização, desde que os consultores realizem suas atividades com o objetivo não apenas do sucesso da avaliação, mas também a efetiva melhoria dos processos.
ii) Organizações que nunca seguiu um processo tem dificuldade em definir um processo sem ajuda. Portanto, a definição de um processo inicial da consultoria é importante. Durante a sua utilização em projetos de software, a organização adquire o conhecimento necessário para definir outra versão adaptada à sua característica específica. Além disso, o compromisso com o programa de MPS e para o processo de adesão é aumentada.
iii) O primeiro projeto utilizando o processo definido normalmente apresenta algumas dificuldades ao ser executado. Os membros da equipe do projeto precisam de tempo para adaptar as suas práticas para as novas atividades e ferramentas de processo. Em alguns casos, o projeto é realizado como um piloto para a implantação do processo.
iv) Um ponto-chave é o compromisso formal durante o projeto. Portanto, às vezes é observada a resistência interna e externa em relação à obtenção de compromisso para produtos de trabalho dos projetos. Alguns *stakeholders*, principalmente clientes, pode não estar interessado no estabelecimento de um compromisso formal.
v) Uma infra-estrutura de processos de software adequada facilita treinamento, implantação e institucionalização de processos de software, uma vez que reduz o processo do tempo de implantação. Se a organização tem utilizado ferramentas de apoio para atividades do processo de software, as mudanças no processo são introduzidas com menos resistência. Além disso, o uso da ferramenta Taba tem

	sido muito importante para o aprendizado de novos conceitos e práticas relacionadas a implantação de processo. Esses conceitos e práticas também ajudam a organização na definição de novas exigências para outras ferramentas de apoio.
vi)	A estratégia baseada em uma implantação gradual de processo de software aderentes ao MR-MPS é viável, desde que a organização perceba o benefício do software de disciplinar o desenvolvimento baseado no processo.
vii)	Pequenas organizações que pretendam estabelecer MR-MPS Nível G de Maturidade, geralmente limitaram recursos financeiros causando problemas para o programa de melhoria. Quanto mais rápido a institucionalização dos processos, menores são os riscos relacionados com o programa de MPS.
viii)	Se o SEPG da organização possui conhecimento sobre engenharia de software, menos esforço de consultoria para a implantação do programa será gasto.
ix)	Durante a implantação de MPS, as organizações têm um melhor controle da gestão e atividades de desenvolvimento (que começa com maturidade MR-MPS Nível G). Assim, eles podem negociar novos termos e custos com clientes após as mudanças de requisitos mostrando o impacto quantitativo dessas mudanças para o projeto.

Iniciativas de melhoria de processos
Utilização da ferramenta Taba para implementa o MPS.BR
Métodos/Modelos de estimativas de software
Não identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

6	Dados da publicação
Título:	Cost Benefit Analysis of Personal Software Process Training Program
Autor(es):	Taek Lee; Dookwon Baik; In, H.P.
Data de publicação:	2008
Referência completa	Taek Lee; Dookwon Baik; In, H.P.; , "Cost Benefit Analysis of Personal Software Process Training Program," Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on , vol., no., pp.631-636, 8-11 July 2008 doi: 10.1109/CIT.2008.Workshops.120
URL: *http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4568574&isnumber=4568462*	
Resumo da Publicação	
A publicação relata que para o desenvolvimento de software rápido e de boa qualidade, muitos pesquisadores estão estudando ativamente a área de melhoria do processo de	

software. O *Personal Software Process* (PSP) é um processo popular individual para engenheiro de software, que foi proposto pelo SEI na Universidade *Carnegie Mellon*. O artigo apresenta as lições aprendidas da participação de um programa de treinamento PSP e aplicar uma abordagem de análise de custo-benefício para investigar a eficácia das fases de treinamento de PSP. Ao quantificar os benefícios e custos que dizem respeito a prática do PSP e estimar o ROI do treinamento de PSP, esperamos ajudar os estudantes e desenvolvedores compreender a relação custo-eficácia do programa de treinamento de PSP e visualizar o que se ganha com PSP.
Dados extraídos da publicação
O Personal Software Process é um treinamento curricular. É projetado para ensinar aos engenheiros de software, princípios básicos de projeto de software e gestão de qualidade. Um objetivo do PSP é mudar o comportamento de engenheiros de software e ensiná-los a medir as suas atividades. Ensinar a analisar as medidas e alcançar metas de desempenho.
Iniciativas de melhoria de processos
Treinamento de PSP para engenheiros de software.
Métodos/Modelos de estimativas de software
Visualização dos valores de B/CR e ROI em cada fase de treinamento de PSP: B/CR (Benefício / Custo) = Benefício / Custo ROI = (custo - benefício) / Custo * 100%.
Tamanho da organização
[x] Pequena [ ] Média [ ] Grande

7	Dados da publicação
Título:	ROI of software process improvement at BL informática: SPIdex is really worth it
Autor(es):	Ferreira A.I.F., Santos G., Cerqueira R., Montoni M., Barreto A., Rocha A.R., Barreto A.O.S., Silva Filho R.C.
Data de publicação:	2008
Referência completa	Ferreira A.I.F., Santos G., Cerqueira R., Montoni M., Barreto A., Rocha A.R., Barreto A.O.S., Silva Filho R.C.; " ROI of software process improvement at BL Informática: SPIdex is really worth it"; Software Process Improvement and Practice 13 (4), pp. 311-318
URL: *http://www.scopus.com/inward/record.url?eid=2-s2.0-55349139330&partnerID=40&md5=233ea4cbc8db67d146265de800e33f2c*	
Resumo da Publicação	
Os autores relatam que devido ao ambiente competitivo, organizações tentam melhorar suas práticas visando atingir níveis mais altos de produtividade, qualidade e competitividade. Apesar de a melhoria de processo de software ser uma das abordagens	

mais usadas para atingir esses objetivos, isto não é fácil e geralmente requer grandes investimentos. Assim, determinar o retorno desses investimentos é crucial para justificá-los e manter a motivação alta. Este trabalho apresenta resultados quantitativos do retorno do investimento relacionados a cronograma, custos, produtividade, qualidade e financeiros na BL Informática, mostrando que esses investimentos realmente podem trazer bons retornos.
Dados extraídos da publicação
A organização motivada pelos benefícios de um programa da qualidade e pela observação da necessidade de seus clientes, iniciou o programa de melhoria de processo em 2003 com o foco na certificação ISO 9001:2000. Em 2005 foi avaliada no nível F do MPS.BR. Em 2006 foi avaliada no nível 3 do CMMI. Algumas medidas básicas já faziam parte da cultura organização antes do projeto de MPS. O ROI de SPI teve um aumento de 11%, CPI teve uma redução de 54%, produtividade aumentou 57% nas atividades de desenvolvimento e testes. A organização cresceu em 287% em colaboradores, 200% em números de projetos e 2695% em faturamento.
Iniciativas de melhoria de processos
Iniciou se certificando na ISO 9001:2000, depois utilizou o MPS.BR e CMMI A organização utilizou técnica de análise do valor agregado para analisar o benefício relacionado às estimativas de custo e tempo. O índice de desempenho de custo (*Cost Performance Index – CPI*) e o índice de desempenho de tempo (*Schedule Performance Index – SPI*).
Métodos/Modelos de estimativas de software
O cálculo do ROI considerava como custo: o treinamento externo, consultoria e avaliações, e como ganho: o lucro dos projetos desenvolvidos durante o programa de MPS.
Tamanho da organização
[ ] Pequena [x] Média [ ] Grande

8	Dados da publicação
Título:	A follow-up reflection on software process improvement ROI
Autor(es):	van Solingen R.
Data de publicação:	2009
Referência completa	van Solingen, R.; , "A Follow-Up Reflection on Software Process Improvement ROI," Software, IEEE , vol.26, no.5, pp.77-79, Sept.-Oct. 2009 doi: 10.1109/MS.2009.120
URL: *http://www.scopus.com/inward/record.url?eid=2-s2.0-70049114749&partnerID=40&md5=74f0cae56936dedcb3f8019caf9447a2*	
Resumo da Publicação	

O Autor relata o resultado de uma pesquisa onde ressalta que forneceu uma visão detalhada de publicações com reais resultados da medição de aplicações práticas de MPS e medição de ROI. Utilizou a abordagem Six Sigma. Esta abordagem implanta um conjunto de métodos de gestão da qualidade, fortemente dependendo técnicas estatísticas, e cria uma infraestrutura especial na organizacional de pessoas que são especialistas em aplicar esses métodos. Porque a engenharia de software é tão fortemente relacionada com o sucesso e fracasso dos negócios, temos de encontrar uma maneira de medir esses valores (SOLINGEN, 2009).
Dados extraídos da publicação
O estudo incluiu 20 casos, onde apresentou um ROI de 7,0 de média e uma mediana 6,6. Isso significa que retorna aproximadamente U$ 7,00 para cada U$ 1,00 investido.
Iniciativas de melhoria de processos
A utilização do Six Sigma em engenharia de software significa: i) Incorporar a mensuração do valor diretamente em tudo o que tentar melhorar; ii) Fornecer conhecimento sobre como nossas atividades realmente contribuem; iii) Ter comprometimento na gestão; iv) Desafios tecnológicos e métodos para provar os benefícios da vida real antes de serem implantados; v) Conduzir sucessos repetidos; vi) Ajudar a estabilizar o desempenho do processo; vii) Estabelecer habilidades estatísticas e de gestão da qualidade.
Métodos/Modelos de estimativas de software
Não identificado
Tamanho da organização
[ ] Pequena [ ] Média [x] Grande

4.1.8. CONSIDERAÇÕES SOBRE O RESULTADO DA PESQUISA

O estudo apresentou um resultado não satisfatório à questão da pesquisa. Os artigos apresentados na seção anterior serviram para constatar que as iniciativas de mensurar o ROI em MPS sempre ocorrem após a implementação de melhoria. Também ficou bastante evidente a preocupação da comunidade em utilizar o valor do ROI em MPS para justificar às organizações novos investimentos em programas de MPS.

4.2. ESTUDO DE CASO

Ao se pensar em realizar um Estudo de Caso, a principal questão é como transformar o estudo de caso em uma estratégia de pesquisa legítima e válida? Poderia a ciência avançar gerando conhecimento relevante observando um caso ou poucos casos? Como construir um protocolo de pesquisa que torne o estudo de caso uma estratégia de pesquisa repetível? Como fazer análise dos dados de uma pesquisa a partir de informações, muitas vezes subjetivas e abstratas, garantindo a validade do estudo de caso? Essas são questões que preocupam pesquisadores que adotam a estratégia, pesquisadores que a contestam e pesquisadores que estão começando a carreira de jovens cientistas na área da computação em geral. A seguir, uma estrutura baseada na literatura, foi definida para orientar a documentação dos estudos de casos realizados na execução do *SPIREM-OBK*.

4.2.1. INTRODUÇÃO

O Estudo de Caso se identifica tradicionalmente com a metodologia qualitativa. Tem como origem uma tradição de sociólogos e tem como características dar especial atenção a questões que podem ser conhecidas por meio de casos. O Estudo de Caso refere-se ao levantamento com mais profundidade de determinado caso ou grupo humano sob todos os seus aspectos. Todavia, é limitado, pois o conhecimento analisado fica restringido ao caso que se estuda, ou seja, não pode ser generalizado (MARCONI E LAKATOS, 2010).

O pesquisador deve iniciar uma pesquisa definindo o objetivo e qual abordagem será utilizada (qualitativa, quantitativa ou uma combinação destas). As pesquisas são classificadas por Selltiz, Jahoda E Destsch (1974) em três grupos:

- Pesquisa exploratória: São todos os estudos que buscam descobrir ideias e soluções, na tentativa de adquirir maior familiaridade com fenômeno de estudo (SELLTIZ; JAHODA; DEUTSCH, 1974);
- Pesquisa descritiva: Expõe características de determinada população ou de determinado fenômeno. Pode também estabelecer correlações entre

variáveis e definir a sua natureza. Não tem compromisso em explicar os fenômenos que descreve, embora sirva de base para tal explicação (VERGARA, 2004);

- Pesquisa explicativa (ou causal): Busca identificar os fatores que contribuem para a ocorrência de determinado fenômeno, deste modo, visa a explicar a razão dos acontecimentos (GIL, 2007; VERGARA, 2004).

4.2.2. ABORDAGEM DA PESQUISA

Após a definição dos objetivos da pesquisa, é necessário adotar o tipo de abordagem mais adequada para atingir os objetivos da pesquisa. Desta forma, utiliza-se a seguinte recomendação:

i) Quando a finalidade da pesquisa é descritiva ou causal, a abordagem é quantitativa;

ii) Quando a finalidade é explicar ou descrever um evento ou uma situação, a abordagem é qualitativa;

iii) Quando o pesquisador inicia a pesquisa usando abordagem qualitativa, e quando necessário, finalize a pesquisa validando as evidências obtidas através da abordagem quantitativa, é denominado como triangulação metodológica ou *mixed-methodology*.

A combinação metodológica é considerada uma forma robusta de se produzir conhecimento, superando as limitações das abordagens tradicionais (qualitativa e quantitativa) (GODOY, 1995B).

A pesquisa qualitativa apresenta as seguintes características: o pesquisador é o instrumento-chave, o ambiente é a fonte direta dos dados, não requer o uso de técnicas e métodos estatísticos, tem caráter descritivo, o resultado não é o foco da abordagem, mas

sim o processo e seu significado, ou seja, o principal objetivo é a interpretação do fenômeno de estudo (GODOY, 1995B).

A pesquisa quantitativa possibilita ao pesquisador mensurar opiniões, hábitos, atitudes e reações por meio de uma amostra estatística que representa o universo pesquisado. A abordagem qualitativa é viável quando o fenômeno em estudo é complexo, de natureza social e de difícil quantificação. Para usar adequadamente a abordagem qualitativa, o pesquisador precisa aprender a observar, analisar e registrar as interações entre as pessoas e entre as pessoas e o sistema.

O interesse do pesquisador não está em quantificar uma ocorrência ou quantas vezes uma variável aparece, mas sim na qualidade em que elas ocorrem (MINAYO, 1994). Para MILES (1979), os pesquisadores, ao realizarem pesquisas de caráter qualitativo, têm grandes dificuldades, por causa: i) da coleta e da análise dos dados qualitativos ser extremamente trabalhosa; ii) da tendência de sobrecarregar o pesquisador em vários pontos, como a exigência de considerável disponibilidade de tempo para transcrever o conhecimento; iii) da falta de clareza dos métodos de análise, diferentemente das análises quantitativas que apresentam convenções claras para o pesquisador utilizar.

A principal vantagem da abordagem qualitativa, em relação à quantitativa, refere-se à profundidade e à abrangência, ou seja, o "*valor*" das evidências que podem ser obtidas e trianguladas por meio de múltiplas fontes, como entrevistas, observações, análise de documentos. Desta forma, permite ao pesquisador detalhes informais e relevantes dificilmente alcançados com o enfoque quantitativo, admitindo também uma relação bem mais próxima e sistêmica do objeto de estudo. Isto difere da abordagem quantitativa que procura interpretar determinado objeto de estudo a partir da definição de variáveis, que às vezes, não podem ser totalmente identificadas e analisadas com a aplicação de ferramentas estatísticas (GODOY, 1995B).

4.2.3. CARACTERÍSTICA DO ESTUDO DE CASO

O Estudo de Caso tem como propósito reunir informações detalhadas e sistemáticas sobre um fenômeno (PATTON, 2002). É um procedimento metodológico que enfatiza entendimentos contextuais e a sua representatividade (LLEWELLYN; NORTHCOTT, 2007), com foco na compreensão da dinâmica do contexto real (EISENHARDT, 1989) e envolvendo-se num estudo profundo e exaustivo de um ou poucos objetos de estudo, de maneira que se permita o seu amplo e detalhado conhecimento (GIL, 2007).

Para Yin (2005, p. 32), "*o estudo de caso é uma investigação empírica que investiga um fenômeno contemporâneo dentro de seu contexto da vida real*". Adequando-se quando "*as circunstâncias são complexas e podem mudar, quando as condições que dizem respeito não foram encontradas antes, quando as situações são altamente politizadas e onde existem muitos interessados*" (LLEWELLYN; NORTHCOTT, 2007, p. 195).

Voss, Tsikriktsis e Frohlich (2002) afirmam que no desenvolvimento da teoria é importante cruzar a teoria emergente com a literatura existente. Rever a teoria emergente envolve refletir sobre questões como: o que as teorias têm em comum? No que são diferentes? E por quê? (EISENHARDT, 1989). Desta forma, o pesquisador pode desenvolver a pesquisa com maior clareza, apresentando coerência lógica com os preceitos teóricos abordados ou buscando quebrar paradigmas conceituais com determinado fenômeno pesquisado, colaborando assim, para o desenvolvimento da Ciência.

A realização de um estudo de caso não é trivial, exige tempo e dedicação e, frequentemente, "*os trabalhos são sujeitos a críticas em função de limitações metodológicas na escolha do(s) caso(s), análise dos dados e geração de conclusões suportadas pelas evidências*" (MIGUEL, 2007, p. 217).

Yin (2005) afirma que, muitos pesquisadores demonstram certo descrédito em relação à utilização de estudo de caso como estratégia, devido:

i) À falta de rigor nas investigações;

ii) Fornecem pouca base para generalizações;

iii) Consomem muito tempo do pesquisador.

Llewellyn e Northcott (2007, p. 196) destacam as principais críticas que a academia impõe à estratégia de estudo de caso, como "*(conclusões) pontuais, infundadas e subjetivas*", além de considerarem que o estudo de caso é uma estratégia de pesquisa "antificientífica". Porém, Sammartino (2002) afirma que vieses não são problemas exclusivos do estudo de caso e distorções são riscos possíveis em qualquer método de pesquisa científica. O estudo de caso, apesar das limitações, é o método mais adequado para conhecer em profundidade todas as características de um determinado fenômeno organizacional. Nesse sentido, mesmo conduzindo-se um único caso, podem-se tentar algumas generalizações, quando o contexto envolve, casos decisivos, raros, típicos, reveladores e longitudinais (YIN, 2005).

Para aumentar a validade externa da pesquisa, pode-se utilizar pelo menos três ou quatro casos, em razão da literatura propor que casos múltiplos são mais convincentes e permitem maiores generalizações (YIN, 2005). Assim, para garantir a qualidade e o sucesso da pesquisa científica, Gummesson (2007) e Yin (2005) destacam que a investigação precisa preencher três critérios para a garantia da excelência em pesquisa científica:

- Validade: Interna – quando se refere a estudos explanatórios que buscam relações causais; e externa – quando as descobertas do estudo de caso são generalizáveis, ou seja, seus resultados são aplicáveis a outros casos (YIN, 2005).

- Generalização: Está intimamente relacionada com a validade e às vezes é chamada validade externa, sendo que os resultados da pesquisa são utilizados em aplicações específicas (GUMMESSON, 2007).

- Confiabilidade: Um estudo com alta confiabilidade pode ser replicado por outros pesquisadores (GUMMESSON, 2007), sendo que o objetivo é

garantir que outro pesquisador possa chegar aos mesmos resultados, para tanto se utiliza um protocolo de estudo de caso (YIN, 2005).

4.2.4. DEFINIÇÃO DO ESTUDO DE CASO

Yin (2005) definiu quatro tipos de estudo de caso:

- Casos únicos: são válidos e decisivos para testar a teoria, quando é raro ou extremo; quando é representativo ou típico, quando é revelador, e longitudinal, em que se estuda o caso único em momentos distintos no tempo;
- Casos múltiplos: são mais consistentes e permitem maiores generalizações, mas demandam maiores recursos e tempo por parte do pesquisador;
- Enfoque incorporado: no estudo de caso pode envolver mais de uma unidade de análise;
- Enfoque holístico: busca examinar apenas a natureza global de um programa ou da organização.

De acordo com Voss, Tsikriktsis e Frohlich (2002), é possível utilizar diferentes casos na mesma organização para pesquisar diferentes questões, ou pode-se utilizar a pesquisa do mesmo assunto em uma variedade de contextos dentro da mesma organização. Segundo Freitas e Jabbour (2011), não há uma definição clara do que seja um estudo de caso único ou uma unidade (objeto) de análise. Por exemplo, no nível organizacional, pode se verificar como a estratégia empresarial afeta a operacionalização e a produtividade na produção, vendas, suprimentos. Por outro lado, se a análise ocorre no nível individual (objeto), verifica-se o nível de absenteísmo, rotatividade, assiduidade, demissões, produtividade e satisfação. Em outras palavras, a unidade de análise (objeto) pode ser o indivíduo, uma prática cultural, um processo de trabalho, um grupo de pessoas ou mesmo

a política e a estratégia organizacional. A definição da unidade de análise vai depender do objetivo que o pesquisador pretende alcançar com o estudo de caso.

4.2.5. PROTOCOLO DE ESTUDO DE CASO

O pesquisador possui um papel que não se pode confundir com outros profissionais, como auditores, que inspecionam, avaliam e supervisionam as ações das organizações, ou seja, deve exercer de forma transparente para as suas fontes ou os seus informantes (MARTINS, 2008). Para Zanelli (2002, p. 83), a "*credibilidade de uma pesquisa consiste na articulação da base conceitual e de adotar critérios rigorosos no uso da metodologia, além de transmitir confiança às pessoas e à organização estudada, de modo que o pesquisador se certifique e garanta que não trará nenhum transtorno na condução do estudo*". Sendo assim, a utilização de um protocolo é essencial para garantir a confiabilidade da pesquisa e serve de orientação na coleta de dados (YIN, 2005).

Um protocolo de pesquisa, em estratégia de estudo de casos deve apresentar os seguintes itens:

i) questão principal da pesquisa;
ii) objetivo principal;
iii) temas da sustentação teórica;
iv) definição da unidade de análise;
v) potenciais entrevistados e múltiplas fontes de evidência;
vi) período de realização;
vii) local da coleta de evidências;
viii) obtenção de validade interna, por meio de múltiplas fontes de evidências;
ix) síntese do roteiro de entrevista.

O protocolo é um *check list* que serve para o pesquisador como um roteiro que deve ser seguido a fim de levantar todos os aspectos propostos na pesquisa (VOSS; TSIKRIKTSIS; FROHLICH, 2002).

4.2.6. PROCEDIMENTO DE COLETA DE DADOS

A coleta de dados normalmente é uma tarefa difícil e complexa, e se não for bem planejada e conduzida, todo trabalho de pesquisa poderá ser prejudicado (YIN, 2005). O planejamento da pesquisa "*assegura a direção, rumo às informações que o problema requer e, ao mesmo tempo, preserva a ética*" (ZANELLI, 2002, p. 82).

Os dados de campo podem ser obtidos de tal forma que permitem caracterizar e explicar detalhadamente os aspectos singulares do caso em estudo, bem como apontar semelhanças e diferenças quando comparados com outros casos estudados Mattar (2001).

Para uma efetiva condução da pesquisa, o pesquisador deve efetuar um planejamento operacional, que pode consistir em seis etapas:

i) Contato formal com a(s) organização(ões) a fim de obter a autorização para realização da pesquisa;

ii) Explanação dos objetivos do estudo para as organizações;

iii) Definição das pessoas a serem entrevistadas;

iv) Definição de critérios para acesso à organização e aos documentos, quais são confidenciais e quais podem ser divulgados;

v) Coleta das evidências, por meio de diversas técnicas;

vi) Devolução aos respondentes/organização para validação ou não das evidências coletadas.

Após o planejamento operacional da pesquisa, a próxima etapa é a definição das técnicas de obtenção de dados e evidências. As principais técnicas apresentadas por

Eisenhardt (1989), Voss, Tsikriktsis e Frohlich (2002), Yin (2005) e Bryman (2008) são as seguintes:

i) Entrevistas – A entrevista é um procedimento de coleta de informações sobre determinado tema científico, realizada por iniciativa do entrevistador, destinada a fornecer informações pertinentes a um objeto de pesquisa (MINAYO, 1994), podendo ser realizada com um único entrevistado ou com um grupo de pessoas (VOSS; TSIKRIKTSIS; FROHLICH, 2002);

ii) Consulta a arquivos e análise de documentos – Documentos de diversos tipos podem ser utilizados, visando a prover dados complementares para a melhor compreensão do problema investigado (GODOY, 1995A, p. 67-68);

iii) Observação – Coloca o pesquisador dentro do contexto de pesquisa, para compreender a complexidade, deve ser informal e dirigida, centrada em observar objetos, comportamentos e factos de interesse para o problema em questão, mesmo que obtidos informalmente (MATTAR, 2001, p. 23);

iv) Conversas informais – Poderá obter evidências não perceptíveis na análise de documentos e na observação. É necessário manter o sigilo das pessoas com as quais obteve informações.

v) Artefatos físicos – Consiste em analisar os artefatos físicos que podem fornecer subsídios relevantes para o estudo de caso;

Um critério para aumentar a credibilidade e a confiabilidade dos resultados é a utilização de múltiplas fontes e a triangulação dos dados e evidências das diversas fontes (YIN, 2005). Para Zanelli (2002, p. 83), "*o rigor na condução de estudos qualitativos é dado pela clareza e sequência lógica das decisões de coleta, pela utilização de métodos e fontes variadas e pelo registro cuidadoso do processo de coleta, organização e interpretação*", em outras palavras, depende da habilidade do pesquisador perceber e captar todas as características do objeto de estudo, sistematizando com perfeição as evidências coletadas das múltiplas fontes.

4.2.7. ESTRATÉGIA PARA ANÁLISE DOS DADOS E EVIDÊNCIAS

Segundo Borges, Hoppen e Luce (2009, p. 886) consistem em "*examinar, categorizar, tabular e recombinar os elementos de prova, mantendo o modelo conceitual e as proposições iniciais do estudo como referências*". Em pesquisas conduzidas por meio de estratégias de estudo de casos não existe um padrão ou formato específico, é a etapa mais difícil e a menos codificada do processo (EISENHARDT, 1989). Um roteiro para o processo de análise das evidências utilizado em estudos de casos tem as seguintes etapas:

i) Transcrição fidedigna das evidências coletadas – que serão organizados e enviados aos entrevistados para confirmação;

ii) Descrição detalhada das evidências coletadas – permite a identificação de dados e informações relevantes para a pesquisa, bem como *insigths* (Miguel, 2007, p. 224);

iii) Análise das evidências coletadas com base nos principais conceitos – utiliza-se a base no referencial teórico, de onde serão identificadas as convergências e divergências da literatura;

iv) Cruzamento das evidências coletadas entre os casos – nas situações em que o estudo é realizado em dois ou mais casos, uma quarta estratégia deve ser adotada, visando à comparação das evidências de cada caso, com o objetivo de obter uma replicação literal ou teórica.

Diante da subjetividade no processo de análise das evidências em estudo caso, o pesquisador precisa ser imparcial e ético, deverá usar a razão em vez da emoção, avaliando os resultados do estudo de forma coesa com os pressupostos teóricos, dentro dos padrões metodológicos e objetivos definidos. Entretanto, antes de realizar a análise comparativa dos vários casos, é fundamental se familiarizar com padrões únicos de cada caso, antes de

buscar a generalização por meio dos casos, o que permitirá ao pesquisador o entendimento detalhado para realizar uma análise com cruzamento dos casos (EISENHARDT, 1989).

4.3. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o "*Estado da Arte*" considerado sobre duas metodologias de pesquisa que serviu de guia para condução desse trabalho, a Revisão Sistemática da Literatura e o Estudo de Caso. Para a Revisão Sistemática da Literatura foi definido o protocolo de busca que foi executado nos sites de buscas da ACM, IEEE e Scopus. Para o Estudo de Caso também foi criado um protocolo de pesquisa.

Capítulo 5

SPIREM-OBK

Neste capítulo são apresentadas as características do *SPIREM-OBK*, o modelo de estimativa de ROI para MPS orientado por conhecimento, os seus objetivos e o fluxo das atividades. Em seguida será apresentado o detalhamento de suas atividades, os dados relevantes e a descrição das fórmulas. E por último as considerações finais do capítulo.

5.1 INTRODUÇÃO

O Modelo de Estimativa de ROI para Melhoria de Processo de Software (*SPIREM-OBK – Software Process Improvement ROI Estimate Model Oriented by Knowledge)* teve como premissa as recomendações de Stutzke (2005) e Parthasarathy (2007) para criar um modelo de estimativa, já mencionadas na seção 2.1.1. Para definir o *SPRIEM-OBK*, foi utilizado o arcabouço da estimativa *Function Points* (STUTZKE, 2005), por se tratar de uma estimativa paramétrica, pois trabalha com 14 características que influenciam na Análise de Ponto de Função e que, por analogia, foi definida uma série de atividades e fatores críticos que irão influenciar a realização da estimativa do ROI no *SPIREM-OBK*. Os fatores possuem uma classificação, características técnicas conforme especificidade e peso de ponderação.

5.2 ATIVIDADES DO SPIREM-OBK

O modelo apresenta sete atividades agrupas em três módulos, que devem ser executadas em uma única iteração. Os resultados esperados após o término da execução de todas as atividades são a estimativa do ROI (em tempo) e a estimativa do ROI (financeiro). Segundo Stutzke (2005) as estimativas possuem erros, portanto, a margem de erro do *SPIREM-OBK* poderá ser minimizada pela melhor adequação dos pesos e atributos das características dos fatores.

Os módulos e as suas atividades:

- Informações do Projeto
    - Escopo da estimativa ROI;
    - Processos de software;
    - Perfil do SEPG;
- Determinar Fatores de Impacto
    - Fatores de Processo;
    - Fatores de Ambiente;
    - Fatores Humano;
- Estimar o ROI.

As três primeiras atividades são caracterizadas como um fator de risco, pois os dados solicitados são de extrema importância e sem essas informações a estimativa do ROI não poderá ser realizada. Após a entrada de dados iniciais, os dados referentes às aos Fatores de Impactos, Fatores de Processo, Fatores Ambientais e Fatores Humanos, deverão ser informados utilizando a experiência técnica e conhecimentos sobre a cultura organizacional.

Para cada um dos três Fatores, foram criados dez fatores de sucesso para MPS baseado em estudos de (DYBA, 2002), (DYBA, 2003), (BADDOO e HALL, 2002), (MONTONI e ROCHA, 2010) e (SANTOS *et al*, 2011).

E a última atividade é realizar a estimativa do ROI, que utiliza todas as informações registradas, gerando assim o ROI em tempo e o ROI financeiro. A Figura 5.1 apresenta a sequência de atividades do *SPIREM-OBK* e em seguida, será apresentado o detalhamento de cada atividade e suas características.

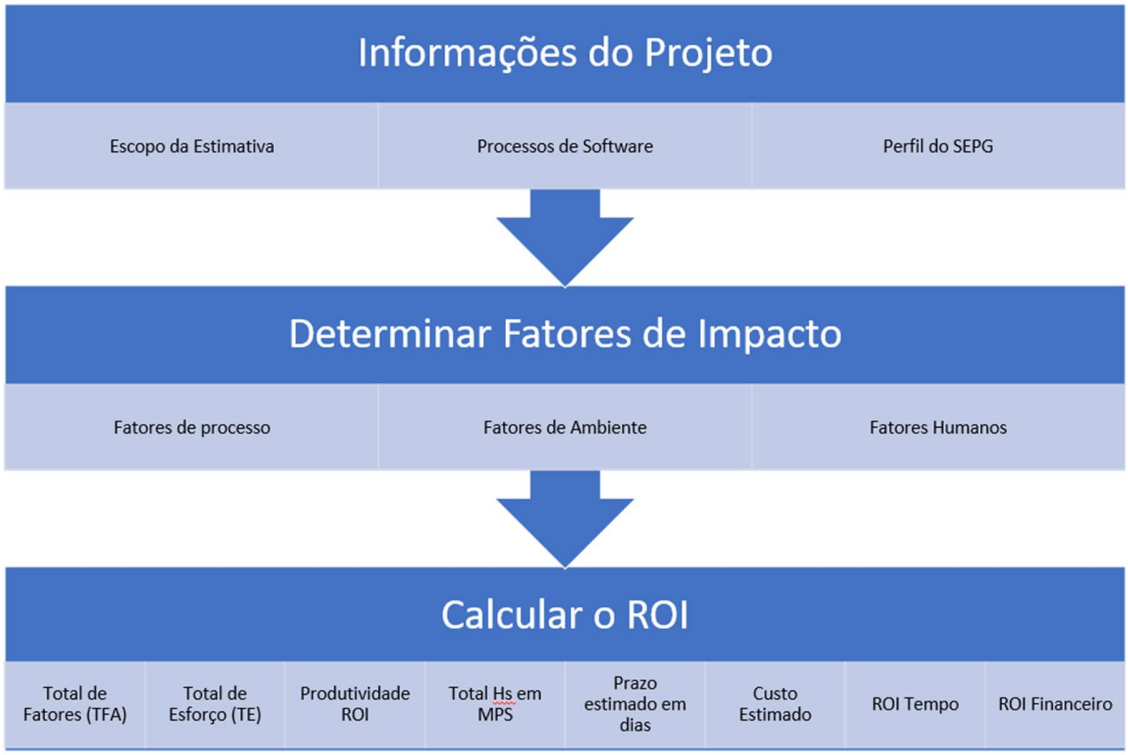


Figura 5.1 – *SPIREM-OBK – Software Process Improvement ROI Estimate Model Oriented by Knowledge*. Fonte: Próprio autor.

5.2.1. IDENTIFICAR ESCOPO DA ESTIMATIVA

A primeira atividade possui um objetivo específico, que se trata em registrar informações básicas para estimar o ROI. Deve-se informar qual a produtividade da equipe técnica de engenheiro de software da organização. A produtividade é o resultado da divisão de total de demandas dessa equipe técnica por horas de trabalho. Normalmente as organizações possuem o controle da produtividade do que é contratado (faturado) *versus* entregas de software. Este percentual determina o índice de produtividade da organização, podendo compor como base de propostas para orçamentos futuros e até estimativas por Ponto de Função. Pode-se adotar a produtividade padrão da organização ou adotar a produtividade especifica da equipe de engenheiros de software do programa de MPS que irão compor o SEPG (*Software Engineering Process Group*).

Os dados referentes à jornada de trabalho da equipe, como Horas de Trabalho determinada para a equipe de engenheiros de software, Valor Médio Homem/Hora e Dias Trabalhado no Mês, Valor Médio da Receita da organização e a Margem de Risco são necessários. Ao informar esses dados é necessário ponderar em relação a esses valores, pois a equipe de engenheiros de software possui valor/hora e função distinta para cada membro. O Valor Médio da Receita da organização será necessário para calcular o quanto à organização poderá dispor de recursos financeiros para o programa de MPS e saber se o ROI financeiro será alcançado. E por último, não menos importante, a Margem de Risco para implementar o programa de MPS. Quanto maior o conhecimento sobre os Fatores de Processo, Ambientais e Humanos da própria organização, menor será a margem de risco para o sucesso do programa de MPS.

Por recomendação, essa atividade deve ser conduzida por um engenheiro de software da organização, pois não encontrará nenhum impedimento ou dificuldade ao realizar o levantamento das informações e preenchimento dos dados. O Quadro 5.1 apresenta o detalhamento dessa atividade.

Quadro 5.1 – Identificar o escopo da estimativa.

Identificar o escopo da estimativa	
Propósito	Registrar dados básicos para estimar o ROI
Papel	Engenheiro de software responsável pelo programa de MPS
Passo:	
1. Informar	Produtividade (%); Horas de Trabalho; Valor médio Homem/Hora; Dias de trabalho no Mês; Valor Médio da Receita da Organização; Margem de Risco.

5.2.2. DETERMINAR QUANTIDADE DE PROCESSO DE SOFTWARE

A segunda atividade possui três objetivos específicos. O primeiro objetivo é identificar o nível de complexidade do processo de software. Levou-se em consideração a quantidade de processos que farão parte do programa de MPS. Quanto maior a quantidade de processos de software envolvidos, maior o nível de complexidade. A complexidade considerada aqui se refere ao detalhamento de documentação dos processos, política de processo, controles de inter-relacionamento entre processos envolvidos, a definição de métricas necessárias etc.

A complexidade do processo possui três atributos sendo o Nível, Descrição e Peso. As complexidades dos processos são caracterizadas como:

i) Baixa – número de processos menor-igual a sete;

ii) Média – número de processos maior-igual a oito e menor-igual a quinze;

iii) Alta – número de processos maior-igual a dezesseis.

O atributo Peso se caracteriza, por analogia, ao esforço a ser realizado pela equipe de engenheiros de software. Para o nível de complexidade Baixa o valor do peso é 1, para o nível de complexidade Média o valor do peso é 1,5 e para o nível de complexidade Alta o valor do peso é 3. O peso é definido por média de 1 processos por engenheiro de software. A complexidade dos processos é apresentada no Quadro 5.2.

Quadro 5.2 – Complexidade de Processos

Nível de Complexidade	Descrição	Peso
Baixa	<= 7	1,0
Média	>= 8 e <=15	1,5
Alta	>= 16	3,0

O segundo objetivo é informar a quantidade de processos que farão parte do programa de MPS. O terceiro objetivo é calcular o CPS – Complexidade de Processo de Software trata-se da multiplicação do Peso pela Quantidade de Processos. O Quadro 5.3 apresenta os passos dessa atividade.

Quadro 5.3 – Determinar Quantidade de Processo de Software.

Determinar Quantidade de Processo de Software	
Propósito	Registrar a complexidade dos processos.
Papel	Engenheiro de Software responsável pelo programa de MPS
Passos:	
1. Informar complexidade do processo	Peso pelo nível de Complexidade
2. Informar quantidade de processos	Número de processos
3. Calcular o CPS	Calcular Peso X Número de Processo

5.2.3. DETERMINAR NÚMERO DE PESSOAS NO SEPG

A terceira atividade possui quatro objetivos específicos. O primeiro objetivo é identificar a classificação do peso da quantidade de engenheiros de software que farão parte da equipe do SEPG.

O Número possui três atributos sendo Classificação, Descrição e Peso. As classificações são as seguintes:

i) Baixa – número de pessoas menor-igual a três;

ii) Média – número de pessoas maior-igual a quatro e menor-igual a oito;

iii) Alta – número de pessoas maior-igual a nove.

O atributo Peso se caracteriza pelo esforço a ser realizado pela equipe. Para a classificação Baixa o valor do peso é 1, para a classificação Média o valor é 1,5 e para a classificação Alta o valor é 3. A classificação das pessoas é apresentada no Quadro 5.4.

Quadro 5.4 – Classificação das Pessoas da Equipe do SEPG

Classificação	Descrição	Peso
Baixa	<= 3	1,0
Média	>= 4 e <=8	1,5
Alta	>= 9	3,0

O segundo objetivo é informar a quantidade de engenheiros de software que farão parte da equipe do SEPG. O terceiro objetivo é calcular o QPE – Quantidade de Engenheiro de Software que significa multiplicar o Peso pela Quantidade de Pessoas. O quarto objetivo é calcular o TPE – Total de Processo por Engenheiro que significa calcular a razão do CPS por QPE. Portando o total de processos será distribuído pelo número de membros da equipe, dessa forma, monta-se um perfil do SEPG e cada um dos membros ficará

responsável por uma determinada quantidade de processos. O Quadro 5.5 apresenta os passos dessa atividade.

Quadro 5.5 – Determinar Número de pessoas no SEPG.

Determinar Número de pessoas no SEPG	
Propósito	Registrar o número de pessoas no SEPG.
Papel	Engenheiro de Software responsável pelo programa de MPS
Passos:	
1. Informar peso pela classificação	Peso pela classificação dos engenheiros de software
2. Informar quantidade de pessoas	Número de engenheiros de software
3. Calcular o QPE	Calcular Peso X Número de Engenheiros de software
4. Calcular o TPE	Calcular **CPS / QPE**

5.2.4. DETERMINAR IMPACTO FATOR DE PROCESSO

A quarta atividade possui três objetivos específicos. O primeiro objetivo é identificar o Peso do Fator de Processo. O Peso é determinado pelo esforço de realização do Fator de Processo pela organização. Por exemplo, caso o Fator de Processo não seja executado pela organização, isso significa que para executar o Fator de Processo será o Esforço Máximo por causa da ausência de experiência dos engenheiros de software. A determinação dos pesos dos fatores é muito importante e caberá a um engenheiro de software que conheça muito bem os processos da organização.

O Peso é definido por uma escala que inicia de 0,75 a 3,0 onde o valor 0,75 indica um Esforço Mínimo, o valor 1,50 indica um Esforço Moderado, o valor 2,25 indica um

Esforço Forte e o valor 3,0 indica um Esforço Máximo apresentados no Quadro 5.6. A denominação Processo de Software poderá ser qualquer área de processo como, por exemplo, Gerência de Configuração, Gerência de Requisitos etc. ou o próprio ciclo de vida do processo de desenvolvimento de software da organização.

Quadro 5.6 – Classificação de Peso por Fator de Processo

Classificação	Peso
Esforço Mínimo	0,75
Esforço Moderado	1,50
Esforço Forte	2,25
Esforço Máximo	3,00

O segundo objetivo é informar o Atributo do Fator de Processo. Partiu-se do pressuposto que a execução do Fator de Processo pela organização, influencia no sucesso do programa de MPS. Portanto deverá ser registrado o risco no nível de influência do atributo do Fator de Processo sendo executado ou não pela organização. No exemplo anterior, onde o Fator de Processo não era executado pela organização e o Peso seria Esforço Máximo para executá-lo, o Fator de Processo seria essencial, portanto, teria uma Influência Máxima de sucesso para o programa de MPS.

O Atributo de Fator de Processo é definido por uma escala que inicia de 0 a 5 onde o valor 0 indica Influência Nula, o valor 1 indica uma Influência Mínima, o valor 2 indica uma Influência Baixa, o valor 3 indica uma Influência Média, o valor 4 que indica uma Influência Alta e o valor 5 indica uma Influência Máxima, conforme apresentado no Quadro 5.7.

Quadro 5.7 – Atributo de Peso por Fator de Processo

Classificação	Peso
Influência Nula	0
Influência Mínima	1

Influência Baixa	2
Influência Média	3
Influência Alta	4
Influência Máxima	5

A relação dos 10 fatores de processo é apresentada no Quadro 5.8.

Quadro 5.8 – Fator de Processo.

Fator	Descrição	Peso	Atributo
FP1	Existe Processo de Software		
FP2	Existe Métrica no Processo de Software		
FP3	Existe Automação no Processo de Software		
FP4	Existe Processo institucionalizado		
FP5	Existem dados históricos de Projetos de Software (*Post Mortem*)		
FP6	Existem Inspeções no Processo de Software		
FP7	Revisões são realizadas gerando oportunidade de melhoria		
FP8	Existem Normas e Procedimentos do Processo de Software		
FP9	Ferramentas de Estimativas são utilizadas		
FP10	Existe um objetivo estratégico para melhoria de processo		

O terceiro objetivo é somar todos os fatores de processo e acrescentar o percentual da Margem de Risco registrada na primeira atividade. O Quadro 5.9 apresenta os passos dessa atividade.

Quadro 5.9 – Determinar Impacto Fator de Processo.

Determinar Impacto Fator de Processo	
Propósito	Registrar o peso e atributo dos fatores de processo.
Papel	Engenheiro de software responsável pelo programa de MPS
Passos:	

1. Informar peso do fator	Registrar Peso do fator de processo
2. Informar atributo do processo	Registrar Atributo do fator de processo
3. Calcular a somatória	Calcular a somatória do FP = (FP1 + FP2 + FP3 + FP4 + FP5 + FP6 + FP7 + FP8 + FP9 + FP10) + Percentual da Margem de Risco

As atividades para determinar os fatores de processos, ambientais e humanos possuem a mesma metodologia, havendo somente pequenas adaptações.

5.2.5. DETERMINAR IMPACTO FATOR DE AMBIENTE

A quinta atividade possui três objetivos específicos. O primeiro objetivo é identificar o Peso do Fator de Ambiente. O Peso é determinado pelo esforço de realização do Fator de Ambiente pela organização. Por exemplo, caso o Fator de Ambiente não exista, isso significa que para executar o Fator de Ambiente será o Esforço Máximo por causa da ausência de práticas pela organização. A determinação dos pesos dos fatores é muito importante e caberá a um engenheiro de software que conheça muito bem o ambiente da organização.

O Peso é definido por uma escala que inicia de 0,75 a 3,0 onde o valor 0,75 indica um Esforço Mínimo, o valor 1,50 indica um Esforço Moderado, o valor 2,25 que indica um Esforço Forte e o valor 3,0 indica um Esforço Máximo apresentados no Quadro 5.10.

Quadro 5.10 – Classificação de Peso por Fator de Ambiente

Classificação	Peso
Esforço Mínimo	0,75
Esforço Moderado	1,50

Esforço Forte	2,25
Esforço Máximo	3,00

O segundo objetivo é informar o Atributo do Fator de Ambiente. Seguindo o mesmo conceito do Fator de Processo, a existência do Fator de Ambiente na organização, influencia no sucesso do programa de MPS. Portanto deverá ser registrado o nível de influência do Fator de Ambiente sendo evidenciado ou não na organização. No exemplo anterior, onde o Fator de Ambiente não era evidenciado na organização e o Peso seria Esforço Máximo para evidenciá-lo, o Fator de Ambiente seria essencial, portanto, teria uma Influência Máxima de sucesso para o programa de MPS.

O Atributo de Fator de Ambiente é definido por uma escala que inicia de 0 a 5 onde o valor 0 indica Influência Nula, o valor 1 indica uma Influência Mínima, o valor 2 indica uma Influência Baixa, o valor 3 indica uma Influência Média, o valor 4 que indica uma Influência Alta e o valor 5 indica uma Influência Máxima, conforme apresentado no Quadro 5.11.

Quadro 5.11 – Atributo de Peso por Fator de Ambiente

Classificação	**Peso**
Influência Nula	0
Influência Mínima	1
Influência Baixa	2
Influência Média	3
Influência Alta	4
Influência Máxima	5

A relação dos 10 fatores de ambiente é apresentada no Quadro 5.12.

Quadro 5.12– Fator de Ambiente.

Fator	Descrição	Peso	Atributo
FA1	A organização provê Treinamento e Orientação		
FA2	Existe Apoio da alta Direção		
FA3	Existe Consultoria Externa contratada		
FA4	A organização possui Sistema de recompensa		
FA5	Existe Controle rigoroso das atividades		
FA6	Existe Liderança interna		
FA7	Existe o Comprometimento da organização		
FA8	Existe colaboração entre as equipes		
FA9	Existe uma política de banco de horas		
FA10	Existem recursos para o programa de MPS		

O terceiro objetivo é somar todos os fatores ambientais e acrescentar o percentual da Margem de Risco registrada na primeira atividade. O Quadro 5.13 apresenta os passos dessa atividade.

Quadro 5.13 – Determinar Impacto Fator de Ambiente.

Determinar Impacto Fator de Ambiente	
Propósito	Registrar o peso e atributo dos Fatores de Ambiente.
Papel	Engenheiro de software responsável pelo programa de MPS
Passos:	
1. Informar peso do fator	Registrar Peso do fator de ambiente
2. Informar atributo do ambiente	Registrar Atributo do fator de ambiente
3. Calcular a somatória	Calcular a somatória do FA = (FA1 + FA2 + FA3 + FA4 + FA5 + FA6 + FA7 + FA8 + FA9 + FA10) + Percentual da Margem de Risco

5.2.6 DETERMINAR IMPACTO FATOR HUMANO

A sexta atividade possui três objetivos específicos. O primeiro objetivo é identificar o Peso do Fator Humano. O Peso é determinado pela existência da especificidade do Fator Humano nos próprios membros da equipe do SEPG. Por exemplo, caso o Fator Humano não exista, ou não seja evidenciado na equipe, isso significa que para existir o Fator Humano na equipe será o Esforço Máximo por causa da ausência dessas características na equipe. A determinação dos pesos dos fatores é muito importante e caberá a um engenheiro de software que conheça muito bem a equipe do SEPG.

O Peso é definido por uma escala que inicia de 0,75 a 3,0 onde o valor 0,75 indica um Esforço Mínimo, o valor 1,50 indica um Esforço Moderado, o valor 2,25 que indica um Esforço Forte e o valor 3,0 indica um Esforço Máximo apresentados no Quadro 5.14.

Quadro 5.14 – Classificação de Peso por Fator Humano

Classificação	Peso
Esforço Mínimo	0,75
Esforço Moderado	1,50
Esforço Forte	2,25
Esforço Máximo	3,00

O segundo objetivo é informar o Atributo do Fator Humano. Seguindo o mesmo conceito do Fator de Processo, a existência do Fator Humano nos membros da equipe do SEPG, influencia no sucesso do programa de MPS. Portanto deverá ser registrado o nível de influência do Fator Humano evidenciado ou não nos membros da equipe do SEPG. No exemplo anterior, onde o Fator Humano não existia nos membros da equipe e o Peso seria de Esforço Máximo para evidenciá-lo, o Fator Humano seria essencial, portanto, teria uma Influência Máxima de sucesso para o programa de MPS.

O Atributo de Fator Humano é definido por uma escala que inicia de 0 a 5 onde o valor 0 indica Influência Nula, o valor 1 indica uma Influência Mínima, o valor 2 indica uma Influência Baixa, o valor 3 indica uma Influência Média, o valor 4 que indica uma Influência Alta e o valor 5 indica uma Influência Máxima, conforme apresentado no Quadro 5.15.

Quadro 5.15 – Atributo de Peso por Fator Humano

Classificação	Peso
Influência Nula	0
Influência Mínima	1
Influência Baixa	2
Influência Média	3
Influência Alta	4
Influência Máxima	5

A relação dos 10 fatores humanos é apresentada no Quadro 5.16.

Quadro 5.16 – Fator Humano.

Fator	Descrição	Peso	Atributo
FH1	Existe conhecimento técnico		
FH2	Equipe com dedicação apropriada		
FH3	Equipe está motivada		
FH4	Equipe está comprometida		
FH5	Equipe tem experiência em MPS		
FH6	Equipe conhece suas responsabilidades		
FH7	Existe pró-atividade na equipe		
FH8	Equipe é autogerenciada		
FH9	Equipe é resistente a mudança		
FH10	Existe rotatividade de pessoas na organização		

O terceiro objetivo é somar de todos os fatores humanos e acrescentar o percentual da Margem de Risco registrada na primeira atividade. O Quadro 5.17 apresenta os passos dessa atividade.

Quadro 5.17 – Determinar Impacto Fator Humano.

Determinar Impacto Fator Humano	
Propósito	Registrar o peso e atributo dos Fatores Humanos.
Papel	Engenheiro de Software responsável pelo programa de MPS
Passos:	
1. Informar peso do fator	Registrar Peso do fator humano
2. Informar atributo do processo	Registrar Atributo do fator humano
3. Calcular a somatória	Calcular a somatória do FH = (FH1 + FH2 + FH3 + FH4 + FH5 + FH6 + FH7 + FH8 + FH9 + FH10) + Percentual da Margem de Risco

5.2.7. ESTIMAR O ROI

Nesta última atividade, todos os dados registrados servirão como premissa para estimar o ROI em tempo e o ROI financeiro. Para estimar o ROI foram definidos oitos subatividades que serão apresentadas a seguir:

a) Calcular o **Total de Fatores (TFA):** O **TFA** é o resultado da somatória de todos os Fatores Processo, Ambientes e Humanos, e que representa o total de fatores que influenciam com seus respectivos pesos e atributos no programa de MPS.

$$\mathbf{TFA = (FP + FA + FH)}$$

b) Calcular o **Total de Esforço (TE):** O **TE** é resultado do **Total de Fatores (TFA)** multiplicado por **Total de Processo por Engenheiro (TPE),** vide seção 3.2.3, Quadro 5.5**.** Isto significa que cada membro da equipe será impactado por todos os fatores.

TE = TFA * TPE

c) Calcular a **Produtividade ROI (PRDR):** O **PRDR** é o inteiro da razão do **Total de Esforço (TE)** por **Produtividade** registrada pela equipe, vide seção 3.2.1, Quadro 5.1. Significa que o **PRDR** é o total de horas para cada ponto do **TE**.

PRDR = int(TE / Produtividade)

d) Calcular o **Total de Horas de MPS (THMPS):** O **THMPS** é o resultado da multiplicação da **Produtividade ROI (PRDR)** por 100 pontos percentuais. O que significa o total de horas para o projeto de MPS.

THMPS = PRDR * 100

e) Calcular o **Prazo Estimado em dias (PZD):** O **PZD** é a razão do **Total de Horas de MPS (THMPS)** pelo resultado da multiplicação do **Total de Processo por Engenheiro (TPE)** por **Horas de trabalho por dia (HT)**.

PZD = THMPS / (TPE * HT)

f) Calcular o **Custo Estimado (CTOR):** O **CTOR** é o resultado em valor monetário do **Total de Horas de MPS (THMPS)** multiplicado por **Valor Médio Homem-Hora (VMH)**.

CTOR = THMPS * VMH

g) Estimar o **ROI em tempo (ROIT):** O **ROIT** é o inteiro da razão do **Prazo Estimado em dias (PZD)** por **Dias trabalhados no mês (DT)**. Isto resultará em um número de meses.

ROIT = int(PZD / DT)

h) Estimar o **ROI financeiro (ROIF):** O **ROIF**$_1$ é a estimativa do ROI, onde a diferença do **Custo Estimado (CTOR)** subtraído do **Valor Médio da Receita (VMR)** é dividida pelo próprio **Custo Estimado (CTOR)** e, a razão é multiplicada por 100. O **ROIF**$_2$ é a notação do ROI onde para cada 1 (valor monetário) retornará o **ROIF**$_2$.

ROIF$_1$ = ((VMR – CTOR) / CTOR) * 100
ROIF$_2$ = (ROIF$_1$ / 100) + 1

O resumo com as formulas para estimar o ROI é apresentado no Quadro 5.18.

Quadro 5.18 – Estimar o ROI.

7. Cálculo do ROI		
7.1	**Calcular o Total de Fatores (TFA):**	TFA = (FP + FA + FH)
7.2	**Calcular o Total de Esforço (TE):**	TE = TFA * TPE
7.3	**Calcular a Produtividade ROI:**	PRDR = int(TE / Produtividade)
7.4	**Calcular o Total de Horas de MPS:**	THMPS = PRDR * 100

7.5	**Calcular o Prazo Estimado em dias:**	PZD = THMPS / (TPE * HT)
7.6	**Calcular o Custo Estimado:**	CTOR = THMPS * VMH
7.7	**Calcular o ROI em tempo:**	ROIT = PZD / DT
7.8	**Calcular o ROI financeiro:**	$ROIF_1$ = ((VMR – CTOR) / CTOR) * 100 $ROIF_2$ = (ROIF1 / 100) + 1

A Figura 5.2 apresenta o fluxo completo do SPIREM-OBK com suas informações necessárias para calcular o ROI em MPS.

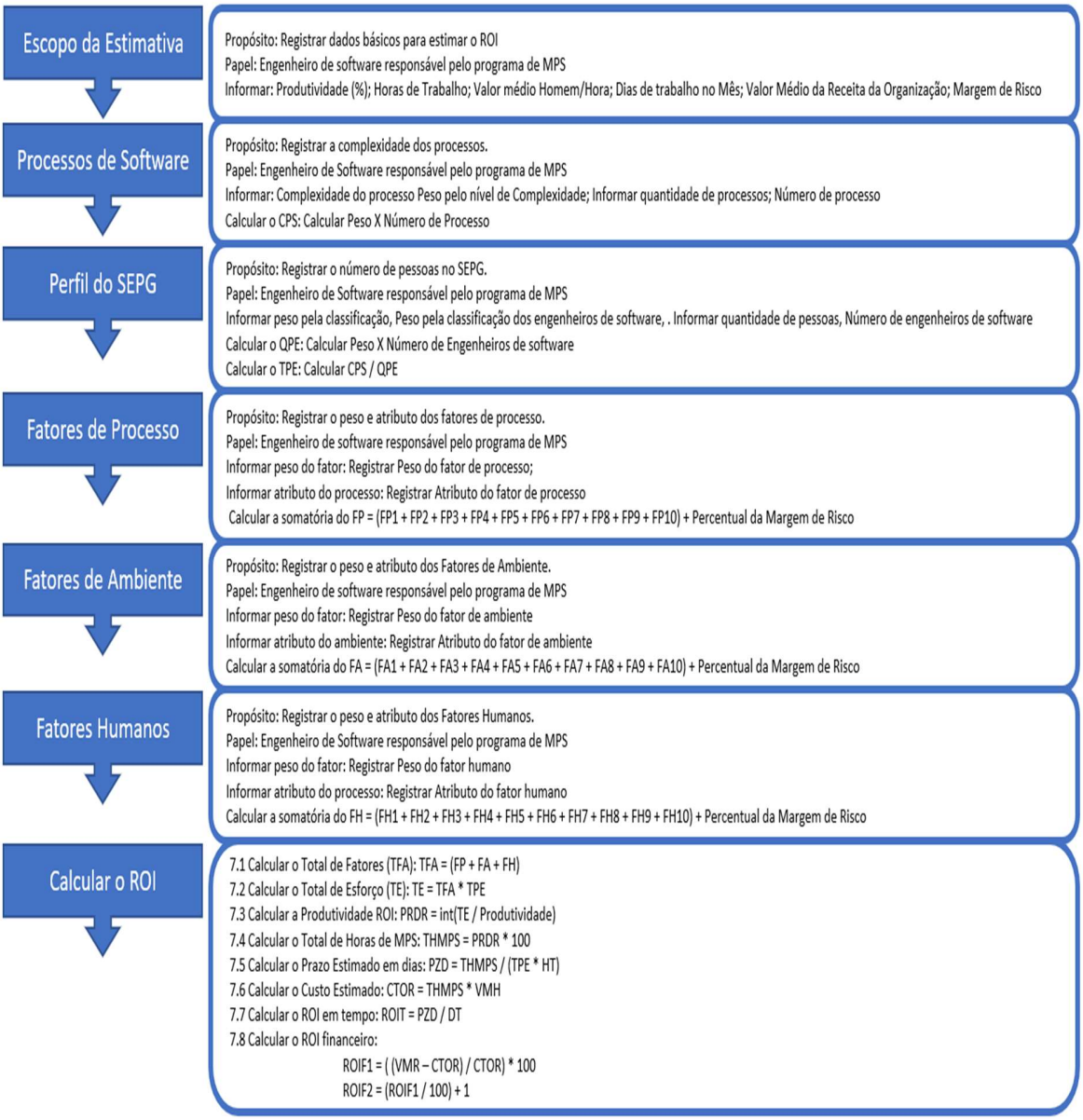


Figura 5.2 – Fluxo completo do *SPIREM-OBK*. Fonte: Próprio autor.

5.3. CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a proposta do *SPIREM-OBK* com objetivo de estimar o ROI em iniciativas de MPS. Foram apresentados os dados básicos necessários para calcular o ROI e as características dos Fatores que influenciam o sucesso em programas de melhoria. Também foram apresentados, o conceito matemático e a sua relevância para a estimativa do ROI. A representação para simular o cálculo do ROI está descrita e apresentada no Apêndice A.

Capítulo 6

Análise crítica dos resultados

Neste capítulo, é apresentada a análise crítica dos resultados do Estudo de Caso que foi aplicado em uma organização de software. A organização iniciou um programa MPS com o objetivo de ser avaliada no nível F do modelo MPS.BR, almejando dessa forma, aumentar os seus ganhos financeiros em novos negócios. Sabe-se que o mercado é extremamente competitivo, e um fator de alta visibilidade no mercado de software é o quanto a organização consegue demonstrar a qualidade dos seus processos, aferido por um selo de qualidade no setor. Em seguida são apresentadas a análise dos dados e as considerações finais.

6.1 INTRODUÇÃO

O estudo de caso foi executado em uma Fundação de Pesquisa e Desenvolvimento em Tecnologia e Comunicação no estado do Amazonas. A organização passou por diversas tentativas sem sucesso de obter uma certificação em modelos de qualidade da sua fábrica de software. A primeira tentativa teve foco no modelo CMMI, buscou-se uma consultoria especializada e por vários meses, entre tentativas e erros, muitos problemas administrativos e técnicos a organização desistiu do projeto por excesso de desgastes entre as partes. O que resultou como legado desse esforço, foi um processo de desenvolvimento de software padrão que sofreu muita rejeição, não foi utilizado pelas equipes de desenvolvimento, muitas áreas de processos definidos e que não integradas, e sucessivas auditorias de qualidade em projetos que só relatavam desvios de processo.

A segunda tentativa foi buscando solução de melhoria utilizando metodologias ágeis, mas por falta de experiência dos colaboradores e muitas resistências em aceitar as mudanças pela parte da equipe técnica, as iniciativas foram caindo em descrédito, gerando muito conflito entre grupos de qualidade e as equipes de projetos.

Após grandes mudanças no quadro técnico, esperava-se que houvesse um profissionalismo maior, mesmo assim a organização não conseguia estabelecer um padrão de qualidade nas entregas dos projetos. A insatisfação dos clientes só aumentava e qualquer iniciativa de melhoria de processo era refutada pelos membros-chave. Até que uma grande ameaça pairou na organização. O maior cliente da organização, orientado pelo Tribunal de Contas da União, iria renovar os editais de contratação e passar a exigir, como critério de seleção de fornecedores, a certificação de qualidade nos processos de desenvolvimento das fábricas de software participantes do edital.

Baseado nas experiências fracassadas anteriores, a diretoria da organização definiu não contratar consultoria e preferiu buscar dentro do quadro de funcionários, profissionais que pudessem contribuir, com suas habilidades técnicas, nesse novo projeto e de extrema

importância para a organização. Como professor e pesquisador da organização, participamos na equipe do SEPG da organização.

6.2 OBJETIVO DA PESQUISA

A pesquisa tem como principal objetivo identificar o ROI em MPS, aplicando o *SPIREM-OBK*, mediante a implementação do programa de melhoria de processo de software, para atender os requisitos de nível F do modelo MPS.BR.

6.3 ABORDAGEM

A definição da abordagem da pesquisa foi originada pela característica da metodologia do *SPIREM-OBK* e o contexto organizacional do estudo de caso. O *SPIREM-OBK* trabalha com um grupo de informações quantitativas e outro grupo de extrema importância que são os Fatores de Processos, Ambientes e Humanos, aplicados na forma de calibragem e que necessitam da habilidade de entendimento qualitativo para serem transformados em uma representação numérica. Portanto a abordagem da pesquisa é denominada triangulação metodológica, por usar as abordagens qualitativa e quantitativa.

6.4 CARACTERÍSTICA DO ESTUDO DE CASO

A organização formou novo grupo multidisciplinar do SEPG composto por dez colaboradores internos, sendo que somente seis pessoas técnicas que iriam trabalhar diretamente no projeto, as outras quatro pessoas faziam parte das atividades administrativas do projeto. Estabeleceu um prazo máximo de doze meses para realizar a implementação e avaliação. Provisionou recurso orçamentário para contratar uma Instituição Avaliadora do modelo MPS.BR e demais despesas. Além disso, havia um risco agravante, pois em uma iniciativa paralela, a fábrica de software estava trocando a plataforma de gestão de projeto, e o projeto de melhoria de processo teve que se adequar em certas demandas.

Por determinação da alta direção, e de certa forma autoritária, foram minimizadas todas as resistências a mudanças e conflitos internos, por pena de advertência. A Direção procurou apresentar todos os riscos do projeto e o grau de importância, inclusive a própria sustentabilidade do negócio. O projeto iniciou com uma reunião geral com toda fábrica de software, informando a todos os colaboradores os objetivos do projeto e sua importância.

6.5 DEFINIÇÃO DO ESTUDO DE CASO

O projeto de melhoria de processo da organização buscava atender os requisitos do nível F do modelo MPS.BR, para tanto, tinha que atender aos requisitos do nível G. Os seguintes processos foram definidos para atender os objetivos:

i) Gerência de Requisitos (GRE);
ii) Gerência de Projeto (GPR);
iii) Medição (MED);
iv) Garantia da Qualidade (GQA);
v) Aquisição (AQU);
vi) Gerência de Configuração (GCO);
vii) Gerência de Portfólio de Projetos (GPP).

Além disso, o processo padrão de desenvolvimento da organização tinha que ser reformulado diante das mudanças necessárias para torná-lo mais controlado e eficiente. Denomina-se como Processo Organizacional. A organização tinha uma grande necessidade de organizar os dados produzidos nos projetos, os dados históricos, as lições aprendidas, as experiências dos projetos, buscou-se estabelecer uma ferramenta para apoio da gestão de conhecimento. A nova plataforma para gestão de projeto estava em evolução em um projeto paralelo, mas demandas do projeto de MPS não podiam sobrepor às demandas da evolução da plataforma. As atividades não podiam ser executadas em paralelo, pois existia dependência entre elas.

6.6 PROTOCOLO

Seguindo as diretrizes para se conduzir o Estudo de Caso. Foi elaborado um protocolo para orientar a execução da pesquisa e apresentado no Quadro 6.1.

Quadro 6.1. Protocolo de pesquisa do Estudo de Caso.

Protocolo
Questão principal da pesquisa
Qual o ROI para o projeto de melhoria da organização?
Objetivo principal
Executar a pesquisa de forma honesta, registrar as evidências e os resultados respeitando o acordo de confiabilidade e segurança da informação.
Temas da sustentação teórica
Processos correspondes aos níveis G e F do modelo MPS.BR; Gestão de Conhecimento e Processo Padrão de desenvolvimento de software da organização e toda documentação do processo padrão existente na organização.
Definição da unidade de análise
Dados da Organização; Contagem de Processos de Software; Quantidade de engenheiros de software na equipe do SEPG; Fatores de Processo; Fatores Ambientes; Fatores Humanos; Estimativa do ROI.
Potenciais entrevistados/múltiplas fontes de evidências
Gerentes de projeto; desenvolvedores; equipe do SEPG; documentos de processo e produtos de trabalho dos projetos.
Período de realização

Agosto de 2019 a maio 2020
Local de coleta de evidências
Fábrica de Software da Organização
Obtenção de validade interna
Todos os dados inseridos na planilha eletrônica foram acompanhados pela equipe SEPG, e ajustes foram realizados quando houve necessidade pelo próprio grupo.
Síntese do roteiro de entrevista
Os dados da organização foram levantados e inseridos na planilha eletrônica; A definição da contagem dos processos era baseada na quantidade de processos dos níveis G e F no MPS.BR; A definição da quantidade de membros do SEPG era baseada no número de pessoas do grupo; A definição do Peso e Atributo dos Fatores de Processo, ambiente e Humano, era baseada pela observação e experiência do SEPG.

6.7 ANÁLISE DE DADOS E EVIDÊNCIAS

Dando início a pesquisa, a planilha eletrônica elaborada para apoiar aos cálculos foi apresentada e discutida em uma reunião, mostrando a importância de cada dado necessário para estimar o ROI. O primeiro conjunto de dados faz parte da seção **Dados da Organização**. Alguns desses dados tiveram que ser criados exclusivamente para o preenchimento na planilha, pois não havia necessidade de tê-lo feito antes.

O dado sobre Produtividade foi exclusivamente para a equipe do SEPG, pois a organização utilizava o dado de produtividade geral da fábrica de software para calcular orçamentos de projetos para clientes. A Quantidade de horas trabalhada por dia foi estabelecidas sendo oito horas. O dado **Valor médio homem-hora** foi calculado a média dos valores dos salários dos membros da equipe do SEPG, resultando em R$ 21,42. A inserção dos **Dias Trabalhados** seguiu a mesma quantidade de dias estabelecida na ferramenta de gestão de cronograma, pois o projeto só seria executado em dias uteis. Tais ajustes seriam necessários devido a variabilidade desses dados, por sofrerem diversas alternâncias mediante ao perfil técnico de capacidade e habilidades individual dos colaboradores.

A questão orçamentaria é um fator crítico e sigiloso nas organizações. E por essa razão foi tratado com extrema cautela. O total de receita da organização era composto por um conjunto de receitas de outras unidades de negócio, portando foi estabelecido um valor monetário máximo para limitar o orçamento que seria aplicado ao projeto de melhoria de processo. Não houve contratação de consultoria para implementar as melhorias no projeto. O custo da avaliação foi tratado ao final das atividades do projeto de forma independente. O risco foi estabelecido 30%, baseado na experiência da equipe do SEPG, comparando ao que era estabelecido nos projetos de software que a organização já desenvolvia. Esses dados foram homologados pela equipe do SEPG.

O segundo conjunto de dados fazem parte da seção **Contagem de Processos de Software**. Foi estabelecido peso para indicar o nível de complexidade dos processos. Neste ponto, vale ressaltar que, quanto maior a quantidade de processos envolvidos, maior será o esforço aplicado pela equipe do SEPG. Foi indicado Peso 1 por se tratar de sete processos. A Complexidade de Processo de Software é calculada pela multiplicação da quantidade de processos pelo peso de referência.

O terceiro conjunto de dados faz parte da seção **Contagem de Engenheiros de Software na equipe do SEPG**. Foi indicado seis engenheiros de software que faziam parte da equipe do SEPG. O peso estabelecido pela quantidade de pessoas equivale a fração de

esforço necessário por pessoa. A Quantidade de Engenheiros de Software é calculada pela multiplicação da quantidade de pessoas pelo peso de referência. O Total de Esforço de Processo por Engenheiro é a fração do esforço necessário para a atividade. Equivale a indicação da dedicação de recursos humanos em atividades no cronograma, podendo ser 25%, 50%, 75% ou 100%.

O quarto conjunto de dados faz parte da seção **Fatores de Processos**. Os pesos e atributos de processos foram definidos pela equipe do SPEG, levando em consideração a experiência da equipe e conhecimento da organização. Da mesma forma, se aplica para os **Fatores Ambientais** e **Fatores Humanos**. A indicação desses valores baseados em uma análise subjetiva só tem valor significativo quando indicado e validado por especialistas.

Por se tratar de uma planilha eletrônica, os cálculos são executados imediatamente e foi estabelecida a estimativa de ROI em Tempo para 10 meses e um ROI financeiro informando que, para cada 1,00 valor monetário investido em MPS, 56,00 de valor monetário seria de retorno. A satisfação saltou aos olhos, mas a equipe sabia que os desafios seriam enormes durante toda a execução do projeto

6.8 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentado o estudo de caso executado em uma organização desenvolvedora de software. Foi utilizado o *SPIREM-OBK* com objetivo de auxiliar a identificação de ROI em programas de MPS. No caso específico desses processos, na época da implementação, eles faziam parte da versão 2012 do MPS.BR e o modelo contempla que avaliações contratadas na versão anterior a vigentes, possuem uma carência de seis meses, pois os impactos são grandes em implementação de processo e validação dos projetos (SOFTEX, 2012).

Foram apresentados os dados básicos identificados na organização utilizados para estimar o ROI e as características dos Fatores que influenciaram o sucesso do programa de

MPS. Os *SPIREM-OBK* utiliza como ferramenta uma planilha eletrônica para estimar o ROI que está no Apêndice A.

Capítulo 7

Conclusão e Trabalho Futuro

Este capítulo apresenta as conclusões e principais contribuições deste trabalho de pesquisa e algumas das perspectivas futuras para a direção de novo trabalho.

7.1 CONCLUSÃO

As organizações de software têm se preocupado muito com MPS para diminuir o tempo, custo, aumentar a produtividade e a qualidade dos seus produtos. Com o objetivo de apoiar a implementação de MPS, várias iniciativas têm sido conduzidas para desenvolver e aprimorar *frameworks* de melhores práticas de desenvolvimento de software, tais como o CMMI e o MPS.BR (MONTONI E ROCHA, 2011). A quantidade de organizações que utilizam esses modelos ainda é pequena em relação ao total de organizações de software. Estudos apontam que o alto custo e a burocracia de recursos para execução dos processos são responsáveis por essa diferença (STAPLES et al., 2007; COLEMAN e O'CONNOR, 2008). Outros estudos mostram que a falta de motivação e resistência a mudança pelos colaboradores das organizações e pela falta de apoio e comprometimento da alta direção da organização também são responsáveis (BADDOO, 2001; NIAZI et al., 2006). Portanto, implementar MPS é uma atividade complexa e repleta de conhecimento que depende de aspectos de caráter sócio-cultural, tecnológico e organizacional (MONTONI E ROCHA, 2011).

Existem várias representações de *frameworks* para calcular o ROI depois de implementar MPS (RICO, 2004) (PHILLIPS, 2007) (SOLINGEN, 2004) (SOLINGEN, 2009).

No Brasil as preocupações são as mesmas (SPINOLA, 2004) (ALVES, 2007) (FERREIRA, 2007). E os resultados apresentados posicionaram o Brasil no 4º lugar no ranque mundial de países com melhor qualidade nos seus processos de software, atrás da China, Estados Unidos e Índia (WEBER, 2014).

Perante estes factos, decidiu-se, nesta tese, pela adoção de métodos qualitativos e quantitativos para guiar a condução de duas pesquisas. A Revisão Sistemática da Literatura para investigar publicação se relatassem o uso de estimativa de ROI para iniciativas de MPS. E o Estudo de Caso que foi executado no contexto real de uma organização de software, com o objetivo estimar o ROI antes de iniciar o projeto de MPS, levando em

consideração fatores relevantes que impactam na melhoria de processo. Os resultados observados empiricamente, pôde-se constatar que a implementação de melhorias em processos de software é de fato um processo social.

Os resultados encontrados pela execução da RSL foram contundentes e apresentaram grandes experiências em iniciativas para demonstrar a importância do ROI em MPS. Entretanto, constatou-se a inexistência de modelo de estimativa de ROI antes de iniciar MPS. A proposta do *SPIREM-OBK* foi executada no Estudo de Caso em uma organização de software. A validação dos dados utilizados na planilha durante a execução do modelo foi realizada pela equipe do SEPG. Atendendo a questão da pesquisa sobre o nível de satisfação da organização ao executar o modelo para estimar o ROI em MPS, constatou-se que, o programa de melhoria da organização conclui todos os trabalhos em 11 meses e a estimativa do ROI apresentada pelo *SPIREM-OBK* foi de 10 meses, portanto, o nível de satisfação da estimativa foi alto, pois representou 90,90% do tempo real. Mas o sucesso da execução do *SPIREM-OBK* fica a cargo exclusivamente da qualidade técnica dos membros da equipe do SPEG, por que o conhecimento contido na organização é rico em valores e a habilidade de discernir esse conhecimento faz a diferença.

O Quadro 7.1, apresenta um comparativo do SPIREM-OBK executado em ambiente simulado, destacando para os possíveis cenários como, Pior Caso, Moderado e Melhor Caso.

Quadro 7.1. Análise comparativa de resultados

Cenário	TFA	TE	PRDR horas	THMPS horas	PZD dias	CTOR	ROIT meses	ROIF %
Pior caso	304,43	456,64	22	2200h	183	$158000	7	3,16
Moderado	325,65	325,65	16	1600h	200	$134000	6	3,73
Melhor Caso	117,98	117,98	5	500h	63	$90000	2	5,56
Estudo de Caso	195,98	152,43	15	1500h	241	$192780	10	1,56

Pôde-se observar que, as simulações sempre são bem diferentes da realidade executata no Estudo de Caso, pôs as variáveis parametrizadas refletem a o conhecimento dos dados da organização, e dessa forma agrega-se um enorme valor para o processo.

7.2 CONTRIBUIÇÕES

As principais contribuições desta tese são:

- O *SPIREM-OBK*, que orienta a inserção de dados relevantes para estimar o ROI em iniciativas de MPS.

Além da proposta do modelo houve outras contribuições:

- Conceitos inter-relacionados sobre a importância do ROI em MPS.
- A possibilidade de replicação do protocolo da RSL.
- A estrutura definida para conduzir o Estudo de Caso.

Por se tratar de uma proposta de modelo de estimativa paramétrico, é importante enfatizar que, quanto mais se conhece sobre os processos organizacionais, e mais se conhece das habilidades e competências técnicas da equipe de SEPG (*Knowledge*), torna-se mais interessante e assertiva, a experiência de realizar as estimativas usando o *SPIREM-OBK*. Pois esses conhecimentos susterão a calibragem adequada nos Fatores que influenciam diretamente os resultados.

7.3 TRABALHO FUTURO

Os resultados obtidos através da revisão da literatura permitiram identificar novos fatores que poderiam ser utilizados, visando evoluir o *SPIREM-OBK*. Entre as limitações do modelo, encontra-se a necessidade de maior sistematização, pois a ferramenta encontra-se no contexto de planilha eletrônica, aumentando assim a confiabilidade e controle sobre os dados inseridos e gerados pelo *SPIREM-OBK*.

Outro ponto interessante é a flexibilização do modelo para ser adaptado e utilizado em qualquer outra natureza de processos organizacionais, não exclusivamente na área de software. A denominação de uma nomenclatura mais genérica contendo uma forma de Guia Instrucional (uma instância), que possa ser utilizado pelas organizações de outros segmentos de negócios.

A utilização de dados integrados, sugerindo aqui uma aplicação de Banco de Dados Relacional. E que no demonstrativo analítico, pudesse através de *Power BI* representá-los em um Dashboard personalizado.

Referências

AGUIAR, S., (2001), “Integração das ferramentas da qualidade ao PDCA e ao programa Seis Sigma”. Belo Horizonte: Editora de desenvolvimento Gerencial, 1 ed., 2001.

ALNOUKARI, M., (2016) “ASD-BI: A Knowledge Discovery Process Modeling Based on Adaptive Software Development Agile Methodology – Scientific Figure on ResearchGate. Available from: *https://www.researchgate.net/figure/Adaptive-software-development-ASD-phases-adapted-from-Pressman-2001_fig1_289672280* [accessed 8 Feb, 2024]

ALVES, C., (2007), “Um Framework de Engenharia de Requisitos para Desenvolvimento de Produto de Software”. PBQP – Programa Brasileiro de Qualidade e Produtividade em Software, 2007.

ANANDARANJAN, A.; WEN, H. J., (1999), “Evaluation of information technology investment”. Management Decision, v.37, n.4, p.329-37, 1999

ÁVILA, T. J. T., (2015) C., (2007), “Uma Proposta de Modelo de Processo para Publicação de Dados Abertos Conectados Governamentais”, Scientific Figure on ResearchGate. Available from: *https://www.researchgate.net/figure/Figura-20-Modelo-de-processo-de-desenvolvimento-em-espiral-proposto-por-Boehm_fig15_317087042* [accessed 8 Feb, 2024]

BADDOO, (2001), “Motivators and de-motivators in software process improvement: na empirical study”, PhD, University of Hertfordshire, UK.

BADDOO, N., HALL, T., (2002), "Software process improvement motivators: An analysis using multidimensional scaling", Empirical Software Engineering, v. 7, n. 2, pp. 93-114.

BADDOO, N., HALL, T., (2003), "De-motivators for software process improvement: Na analysis of practitioners' views", Journal of Systems and Software, v. 66, n. 1, pp. 23-33.

BASILI, V. e McGARRY F., (1998), “The Experience Factory: How to Build and Run One. Tutorial TF01, 20th International Conference on Software Engineering (ICSE'98), Kyoto, Japan, 1998.

BASILI, V., CALDEIRA, G., ROMBACH, H. D., (1994), “Goal Question Metric Paradigm, Encyclopedia of Software Engineering”, 2 Volume Set, John Wiley & Sons, Inc, 1994.

BASILI, V., (1994), “The Maturing of the Quality Improvement Paradigm” in the SEL. Presentation, Nokia Research Centre: Software Engineering Workshops, Helsinki, Finland, 1994.

BECK, K. (2000). Extreme Programming Explained. Massachusetts: Addison-Wesley.

BEEDLE, M; et al. (1998). “SCRUM: An extension pattern language for hyperprodutive software Development”, In: Pattern Languages of Programs'98 Conference, Monticello.

BIOLCHINIM J., MIAN, P., NATALI, A., TRAVASSOS, G. (2005). “Systematic Review in Software Engineering”, Techinical Report, PESC – COPPE/UFRJ.

BOEHM, B. W., (2000), “Software Cost Estimation with COCOMO II”, Prentice Hall PTR.

BORGES, M. HOPPEN, N.; LUCE, F. B. (2009). Information technology impact on market orientation in e-business. Journal of Business Research, v. 62, p. 883-890.

BORSSATTO, Í. (2008). “A implementação do nível F na Synos”, Workshop de Empresas do MPS.BR.

BRIAND, L., DIFFERDING, C. M., ROMBACH, H., D. (1996). "Practical Guidelines for Measurement-Based Process Improvement". Software Process, 2(4), December.

CAMPOS, V. F. (1992), “TQC: Controle da Qualidade Total” (no estilo japonês). Fundação Cristiano Ottoni, 6ª Edição, 1992.

CAPRA, F. (1998). “A teia da vida: uma nova compreensão científica dos sistemas vivos”. São Paulo: Cultrix, 1 ed.

CLEMENTE, A. (1998). “Projetos empresariais e públicos”. São Paulo: Editora Atlas.

CHRISSIS, M. B., KONRAD, M., SHRUM, S. (2006). “CMMI (Second Edition): Guidelines for Process Integration and Product Improvement “. SEI Series in Software Engineering, Addison Wesley Professional.

COLEMAN, G., O’CONNOR, R. (2008). "Investigating software process in practice: A grounded theory perspective", Journal of Systems and Software, v. 81, n. 5, pp.772-784.

CONTE, T. U., MENDES, M. E., TRAVASSOS, G. (2005). Processos de Desenvolvimento para Aplicações Web: Uma Revisão Sistemática. XI Simpósio Brasileiro de Multimidia e Web, volume 1, pag 63-75, Poços de Caldas, SBC.

CORA. (2023), Precificação na prática: o guia completo para acertar os cálculos. E-book Cora. Disponível em: *https://www.cora.com.br/*

CURIEL, I.E.E., JACOBO, J.R., ZEPEDA, J.A.F. (2011). “A Competency Framework for the Stakeholders of a Software Process Improvement Initiative”, in ICSSP '11: Proceedings of the 2011 International Conference on Software and Systems Process, Honolulu, USA.

DE LUCA, J. (2002). Feature-Driven Development (FDD) Overview Presentation.

DEMING, W. E. (1990). “Qualidade: A Revolução da Administração”. Ed. Marques Saraiva, Rio de Janeiro.

DIESTE, O., PADUA, A. G. (2007). “Developing Search Strategies for Detecting Relevant Experiments for Systematic Reviews”. In: Proceeding of First International Symposium on Empirical Software Engineering and Measurement.

DYBA, T. (2000). "An Instrument for Measuring the Key Factors of Success in Software Process Improvement", Journal of Empirical Software Engineering, v. 4, pp. 357-390.

DYBA, T. (2003). "Factors of software process improvement success in small and large organizations: an empirical study in the scandinavian context".

EISENHARDT, K. M. (1989). Building Theories from Case Study Research. The Academy of Management Review, v. 14, n. 4, p. 532-550.

EMAN, K. E. (2003). “Return on Investment Models for Static Analysis Tools”. Klocwork Inc.

FERREIRA, A. I., CERQUEIRA, R., SANTOS, G., MONTONI, M., BARRETO, A., BARRETO, A. O. S., ROCHA, A. R. (2007). “Retorno de Investimento da Melhoria de Processo de Software na BL Informática”, Simpósio Brasileiro de Qualidade de Software.

FOWLER, M. (2001). The New Methodology.

FRAUNHOFER Institute. (1998). “PIA – Perfect Improvement Approach”. Univärsität Kaiserslautern.

FREITAS W. R. S. e JABBOUR, C. J. C. (2011). "ESTUDO & DEBATE", Lajeado, v. 18, n. 2, p. 07-22.

GIVOLY, D. E SHI, C. (2007). “Acconting for Software Development Costs and the Cost of Capital: Evidence from IPO Underpricing in the Software Industry”. Disponível em: *https://ssrn.com/abstract=999263*

GODOY, A. S. (1995B). Introdução a pesquisa qualitativa e suas possibilidades. Revista de Administração de Empresas. São Paulo, v. 35, n. 2, p. 57-63, Mar./Abr.

GREMBA, J. e MYERS, C. (1997). “The IDEAL(SM) Model: A Practical Guide for Improvement”. Carnegie Mellon University – Software Engineering Institute.

GIL, A. C. (2007). Métodos e Técnicas de Pesquisa Social. 5 ed. São Paulo: Atlas.

HARRY, M. (1998). “Six Sigma: a breakthrough strategy for profitability”. Quality Progress. v. 31, n. 5, p. 60-64.

HIGHSMITH, J. (2002). Agile Software Development Ecosystems. Boston: Addison Wesley.

HIRSCHFELD, H. (1998). “Engenharia Econômica e Análise de Custos: Aplicações Práticas para Economistas, Engenheiros, Analistas de Investimentos e Administradores”, 6ª ed. São Paulo: Editora Atlas.

HUMPHREY, W. S. (1989). “Managing the Software Process”. Addilson-Wesley Publishing Co., Reading, Massachusetts.

IEEE, (1990). Institute of Electrical and Electronics Engineers. IEEE-STD-610: Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE.

ISO/IEC 15504. (2003). “ISO/IEC 15504 - Information Technology – Process Assessment, International Standard (IS)”.

ISO/IEC 15504-5. (2006). “ISO/IEC 15504 - Information Technology – Process Assessment – Part 5: An exemplar Process Assessment Model”.

ISO/IEC 33001. (2015). International Organization for Standardization/International Electrotechnical Commission. ISO/IEC 33001:2015 Information Technology – Process Assessment –Concepts and Terminology, Geneve: ISO.

JORGENSEN M., INDAHL U., SJOBERG D. (2002). “Software Effort Estimation by Analogy and “Regression Toward the Mean””; Simula Research Laboratory, Oslo, Noruega.

JORGENSEN M.; “A Review of Studies on Expert Estimation of Software Development Effort”; Simula Research Laboratory, Oslo, Noruega; 2002b

KITCHENHAM, B., 2004, Procedures for Performing Systematic Reviews”, Keele technical. report SE0401 and NICTA technical report 0400011T.1.

LAKATOS, E. M., MARCONI, M. A. (2010). "Fundamentos de metodologia científica". São Paulo: Atlas, 2010. 315 p. ISBN 978-85-224-5758-8.

LLEWELLYN, S.; NORTHCOTT, D. (2007). "The “singular view” in management case studiesqualitative research in organizations and management". An International Journal, v. 2, n.3, p. 194-207.

MAFRA, S., TRAVASSSOS, G. H. (2006). "Estudos Primários e Secundários apoiando a busca por Evidências em Engenharia de Software". Relatório Técnico RT-ES 687/06, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

MARTINS, G. A. (2008). Estudo de caso: uma reflexão sobre a aplicabilidade em pesquisas no Brasil. Revista de Contabilidade e Organizações, v. 2, n. 2, p. 9-18, jan./abr.

MATTAR, F. N. (2001). Pesquisa de marketing. 3 ed. São Paulo: Atlas.

McGARRY J., CARD D., JONES C., LAYMAN B., CLARK E., DEAN J., HALL F.; (2001). "Practical Software Measurement: Objective Information for Decision Makers": Addison-Wesley.

MIGUEL, P. A. C. (2007). Estudo de caso na administração: estruturação e recomendações para sua condução. Produção, v. 17, n. 1, p.216-229, jan./abr.

MINAYO, M. C. S. (1994). Pesquisa Social: teoria, método e criatividade. Petrópolis: Vozes.

MONTEIRO, T. C., PIRES, C. G. S., BELCHIOR, A. D. (2005). "TUCP: Uma Extensão da Técnica UCP", IV Simpósio Brasileiro de Qualidade de Software.

MONTONI, M., ROCHA, A. R. (2010). "Aplicação de Grounded Theory para Investigar Iniciativas de Implementação de Melhorias em Processos de Software". In: IX Simpósio Brasileiro de Qualidade de Software (SBQS 2010), pp. 167-181, Belém, PA.

MONTONI, M., ROCHA, A. R. (2011). "Uma investigação sobre fatores críticos de Sucesso em Iniciativas de Melhoria de Processos de Software". Tese de Doutorado do Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro.

NIAZI, M., WILSON, D., ZOWGHI, D. (2006). "Critical success factors for software process improvement implementation: An empirical study", Software Process Improvement and Practice, v. 11, n. 2, pp. 193-211.

PANDE, P., NEUMAN, R., CAVANAGH, R. (2001). "Estratégia Seis Sigma: como a GE, a Motorola e outras grandes empresas estão aguçando seu desempenho". 1 ed. Rio de Janeiro: Qualitymark.

PARTHASARATHY, M. A. (2007). "Practical Software Estimation", Infosys, Addison-Wesley.

PATTON, M. G. (2002). “R Qualitative Research and Evaluation Methods”, 3 ed. Thousand Oaks, CA: Sage.

PHILIPPS, J. (2007). “ROI – Retorno sobre o investimento em projetos”, Revista MundoPM, Rio de Janeiro, Número 15, p.8-16, junho/julho.

PRESSMAN, R. S. (2016). “Engenharia de Software: uma abordagem profissional”; Tradução: João Eduardo Nóbrega Tortello, Revisão Técnica Reginaldo rakaki, Julio Arakaki, renato Manzan de Andrade. 8. Ed, Porto Alegre: AMGH, 2016.

QUATRANI, T (2001). “Modelagem Visual com Rational Rose 2000 e UML Rio de Janeiro, Ed. Ciência Moderna.

RAINER, A., HALL, T. (2002). "Key success factors for implementing software process improvement: A maturity-based analysis", Journal of Systems and Software, v. 62, n. 2, pp. 71-84.

RICO, D. F. (2004). “ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers”, J, Ross Publishing, Boca Raton, FL.

RICO, D. F. (2006). “What is the ROI of Agile vs. Traditional Methods?”, 2006.: *http://davidfrico.com*

ROMBACH, DIETER (1994). “Systematic Quality Improvement”, DIPOLI, Helsinki, Finland.

SAMMARTINO, W. (2002). “A integração do sistema de gestão de recursos humanos com as estratégias organizacionais”. Tese (Doutorado) – Programa de Pós-Graduação em Administração da Faculdade de Economia, Administração e Contabilidade da Universidade de São Paulo, São Paulo.

SANTOS, G. (2008). Ambientes de Engenharia de Software Orientados a Corporação, Tese de D.Sc., COPPE, UFRJ, Rio de Janeiro, RJ. Brasil.

SANTOS, G. (2010), Mini-curso: Revisão Sistemática. Simpósio Brasileiro de Qualidade de Software (SBQS 2010), Belém, PA, Brasil.

SANTOS, D. V., VILELA, D., SOUZA, C., CONTE, T. (2011). “Programas de Melhoria de Processo de Software – Uma pesquisa sobre a influência dos aspectos humanos”, Simpósio Brasileiro de Qualidade de Software.

SALVIANO, C. F. (2006). “Melhoria e Avaliação de Processo de Software com ISO/IEC 15504-5:2006”. Lavras: UFLA/FAEPE. 2006. (Publicação do Curso de Pós-graduação “Latu Sensu” (Especialização) à Distância em Melhoria de Processo de Software).

SCHAICOSKI, J. C. (2002). “A utilização do ROI na análise de projetos de tecnologia da informação”, Dissertação (Mestrado) – Universidade Federal de Santa Catarina, Florianópolis.

SCHWABER, K. (2004). Agile Project Management with Scrum. Seattle: Microsoft Press.

SEI. (2006). Capability Maturity Model® Integration (CMMI-DEV): Version 1.2: CMMI-DEV for Development. Carnegie Mellon University – Software Engineering Institute.

SEI. (2007). Capability Maturity Model® Integration (CMMI-SM), Version 1.2. CMMI-SM for Software Engineering – Staged Representation. Carnegie Mellon University – Software Engineering Institute.

SELLTIZ, C.; JAHODA, M.; DEUTSCH, M. (1974). "Métodos de Pesquisa nas Relações Sociais". São Paulo: EDUSP.

SOFTEX. (2012). “Melhoria de Processo do Software Brasileiro – Guia Geral, Versão 2012”. Disponível em: *http://www.softex.br*

SOFTEX. (2021). “Melhoria de Processo do Software Brasileiro – Guia Geral, Versão 2021”. Disponível em: *http://www.softex.br*

SOFTEX. (2008). iMPS : resultados de desempenho de organizações que adotaram o modelo MPS / Guilherme Horta Travassos e Marcos Kalinowski. – Campinas, SP: Associação para Promoção da Excelência do Software Brasileiro. Disponível em: *http://www.softex.br/mpsbr*

SOLINGEM, R. V., e BERGHOUT, E. (1999). “The Goal/ Question/ Metric Method: A Pratical Guide for Quality Improvement of software development”. London: McGraw Hill.

SOLINGEM, R. V. (2004). “Measuring the ROI of Software Process Improvement”, IEEE Computer Society, 2004

SOMMERVILLE, I. (2011). Engenharia de Software / Ian Sommerville; tradução Ivan Bosnic e Kalinka G. de O. Gonçalves; revisão técnica Kechi Hirama. — 9. ed. — São Paulo: Pearson Prentice Hall.

SPINOLA, M. (2004). “Eficácia e Benefícios dos Modelos de Gestão da Qualidade de Processo de Software em Empresas Brasileiras”. PBQP – Programa Brasileiro de Qualidade e Produtividade em Software.

STAPLES, M., NIAZI, M., JEFFERY, R., et al. (2007). "An exploratory study of why organizations do not adopt CMMI", Journal of Systems and Software, v. 80, n. 6, pp. 883-895.

STUTZKE, R. D. (2005). “Estimating Software-Intensive Systems: Projects, Products, and Processes, SEI Series in Software Engineering, Addison Wesley.

SUWARDY, T., RATNATUNGA, J., SOHAL, A. S., SPEICHT, G. (2003). “IT projects: evaluation, outcomes and impediments”. Benchmarking: An International Journal, v.10, n.4, p.325-42.

TECHNO. (2023). Precificação de Software: quanto cobrar pelo meu sistema? Disponível em: *https://blog.tecnospeed.com.br/precificacao-de-software/*

TRAVASSOS, G.H., KALINOWSKI, M. (2009). "iMPS 2009: caracterização e variação de desempenho de organizações que adotaram o modelo MPS", SOFTEX, Campinas,SP.

VALENÇA, A. R. (2007). “Implantação de Processo de Estimativa de Esforço de Desenvolvimento de Software – Caso Real”, UFPE.

VERGARA, S. C. (2004). Projetos e relatórios de pesquisa em administração. 5. ed. São Paulo: Atlas.

VIANA, P. W. P.; VASCONCELOS A. M. L. (2008)."FROISPI – Framework Return on Investment of Software Process Improvement". VI Workshop de Teses e Dissertações em Qualidade de Software (WTDQS) no Simpósio Brasileiro de Qualidade de Software (SBQS).

VOSS, C.; TSIKRIKTSIS, N.; FROHLICH, M. (2002). "Case research in operations Management". International Journal Of Operations & Production Management, v. 22, n. 2, p. 195-219.

WEBER, K., MACHADO, C. F., SCALET, D. (2005). “Modelo de Referência e Método de Avaliação Para Melhoria de Processo de Software – versão 1.0 (MR-MPS e MA-MPS)”. Proceedings do IV Simpósio Brasileiro de Qualidade de Software / SBQS, Porto Alegre.

WEBER, K., OLIVEIRA, N.H.F., DUARTE, V.C. (2014). "Estudo de caso: 10 anos de MPS.BR", Campinas, SP: Softex.

WIGGENBORN, W. (2000). “A universidade Motorola: quando o treinamento se transforma em educação”.

YIN. R. K. (2005). “Estudo de caso: planejamento e métodos”. 3 ed. Porto Alegre: Bookman.

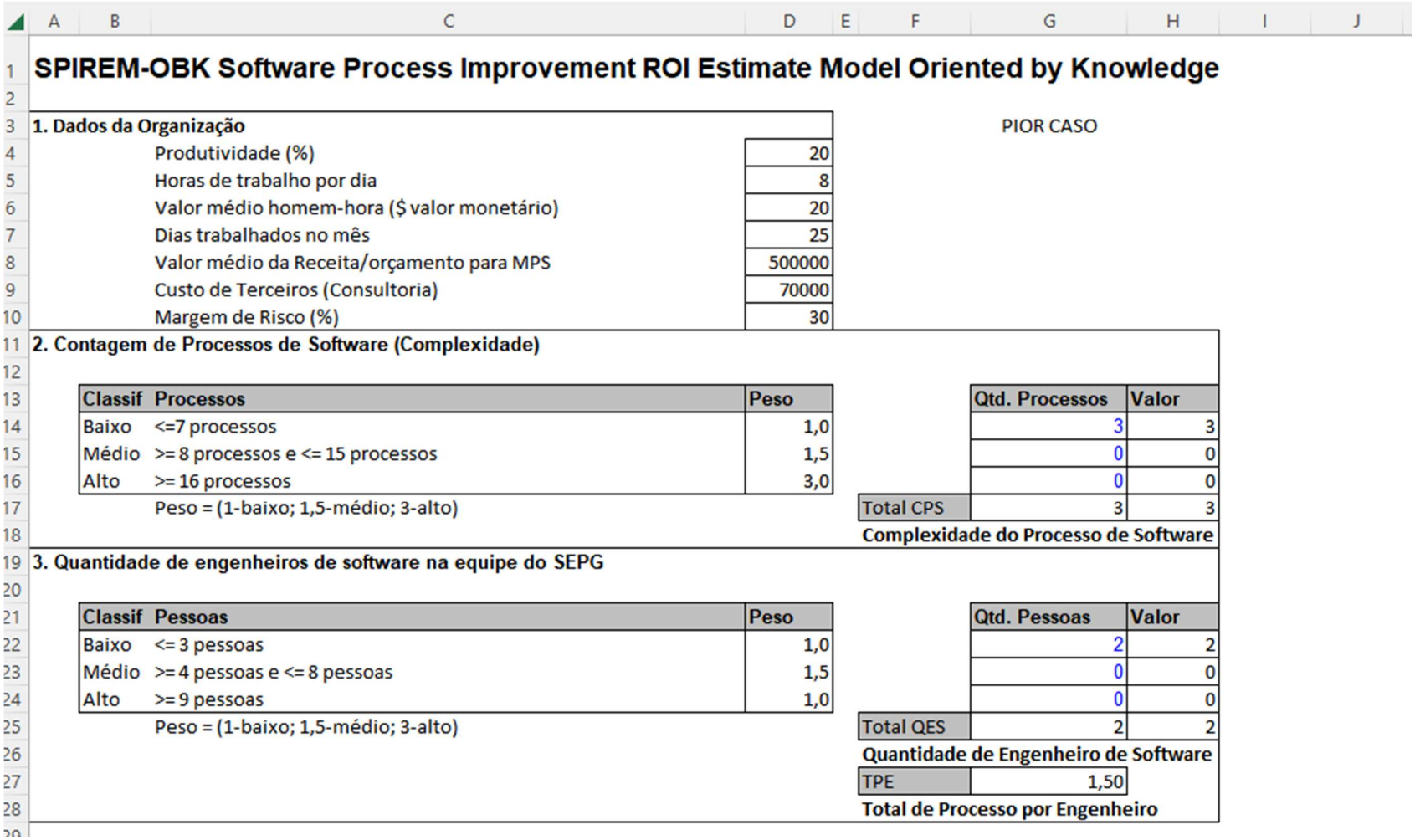
ZANELLI, J. C. (2002). “Pesquisa qualitativa em estudos da gestão de pessoas. Estudos da Psicologia”, n. 7, p. 79-88.

Apêndice A

Planilha eletrônica do SPIREM-OBK

Este documento apresenta a planilha eletrônica para apoiar o *SPIREM-OBK*. Na primeira seção é apresentada a planilha com dados fictícios simulando e validando os cálculos. Na segunda seção são apresentados os dados coletados no estudo de caso executado na organização.

SIMULAÇÃO DO PIOR CASO



SPIREM-OBK Software Process Improvement ROI Estimate Model Oriented by Knowledge

PIOR CASO

1. Dados da Organização

Produtividade (%)	20
Horas de trabalho por dia	8
Valor médio homem-hora ($ valor monetário)	20
Dias trabalhados no mês	25
Valor médio da Receita/orçamento para MPS	500000
Custo de Terceiros (Consultoria)	70000
Margem de Risco (%)	30

2. Contagem de Processos de Software (Complexidade)

Classif	Processos	Peso	Qtd. Processos	Valor
Baixo	<=7 processos	1,0	3	3
Médio	>= 8 processos e <= 15 processos	1,5	0	0
Alto	>= 16 processos	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total CPS	3	3

Complexidade do Processo de Software

3. Quantidade de engenheiros de software na equipe do SEPG

Classif	Pessoas	Peso	Qtd. Pessoas	Valor
Baixo	<= 3 pessoas	1,0	2	2
Médio	>= 4 pessoas e <= 8 pessoas	1,5	0	0
Alto	>= 9 pessoas	1,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total QES	2	2

Quantidade de Engenheiro de Software

TPE	1,50

Total de Processo por Engenheiro

4. Fatores de Processos

Fator	Descrição	Peso
FP1	Existe processo de software *	3,00
FP2	Existe Métricas no processo de software	3,00
FP3	Existe Automação no processo de software	3,00
FP4	Existe Processo institucionalizado	3,00
FP5	Existe dados históricos de Projetos de software (Post Mortem)	3,00
FP6	Existe Inspeções no processo de software	3,00
FP7	Revisões são realizadas gerando oportunidade de melhoria	3,00
FP8	Existe Normas e Procedimentos do processo de software	3,00
FP9	Ferramentas de Estimativas são utilizadas	3,00
FP10	Existe um objetivo estratégico para melhoria de processo	3,00

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
5	15,00
FatorT	150,00
FP	165

FP = Fator de Processo

Atributos
0 - Influência Nnula
1 - Influência Mínima
2 - Influência Baixa
3 - Influência Média
4 - Influência Alta
5 - Influência Máxima
Peso
0,75 - Esforço Mínimo
1,50 - Esforço Moderado
2,25 - Esforço Forte
3,00 - Esforço Máximo

5. Fatores Ambientais

Fator	Descrição	Peso
FA1	A organização provê Treinamento e Orientação	0,75
FA2	Existe Apoio da alta Direção	0,75
FA3	Existe Consultoria externa	0,75
FA4	A organização possui Sistema de recompensa	3,00
FA5	Existe Controle rigoroso das atividades	3,00
FA6	Existe Liderança interna	0,75
FA7	Existe o Comprometimento da organização	3,00
FA8	Existe colaboração entre as equipes	3,00
FA9	Existe uma política de banco de horas	3,00
FA10	Existe recursos para o programa de MPS	3,00

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor
1	0,75
4	3
0	0
3	9
5	15
2	1,5
5	15
5	15
5	15
5	15
FatorA	89,25
FA	98,175

FA = Fator Ambiental

Atributos
0 - Influência Nnula
1 - Influência Mínima
2 - Influência Baixa
3 - Influência Média
4 - Influência Alta
5 - Influência Máxima
Peso
0,75 - Esforço Mínimo
1,50 - Esforço Moderado
2,25 - Esforço Forte
3,00 - Esforço Máximo

6. Fatores Humanos

Fator	Descrição	Peso	Atribuído	Valor
FH1	Existe conhecimento técnico	0,75	5	3,75
FH2	Equipe com dedicação apropriada	0,75	5	3,75
FH3	Equipe está motivada	0,75	5	3,75
FH4	Equipe está comprometida	0,75	5	3,75
FH5	Equipe tem experiência em MPS	0,75	5	3,75
FH6	Equipe conhece suas responsabilidades	0,75	5	3,75
FH7	Existe pró-atividade na equipe	0,75	5	3,75
FH8	Equipe é auto-gerenciada	0,75	5	3,75
FH9	Equipe é resistente a mudança	0,75	5	3,75
FH10	Existe rotatividade de pessoas na organização	0,75	5	3,75
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorA	37,5
			FA	41,25

FH = Fator Humano

Atributos
0 - Influência Nnula
1 - Influência Mínima
2 - Influência Baixa
3 - Influência Média
4 - Influência Alta
5 - Influência Máxima
Peso
0,75 - Esforço Mínimo
1,50 - Esforço Moderado
2,25 - Esforço Forte
3,00 - Esforço Máximo

7. Calculo do ROI

7.1	**TFA**	FP + FA + FH	304,43		
7.2	**Total de Fator * Total Processo pro Engenheiro de Software (TE=TFA*TPE)**		456,64		
7.3	**Calcular a Produtividade ROI (PRDR=TE/Produtividade)**		22		
7.4	**Esforço = Produtividade * 100% do programa de MPS (THMPS=PRDR*100)**		2200	horas	
7.5	**Prazo estimado em dias PZD=(THMPS/(TE * HT))**		183	Dias	
7.6	**Custo estimado (CTOR=THMPS*VMH) + terceiros**		158000,00	$	
7.7	**ROI em tempo (ROIT = PZD / DT)**		7	meses	
7.8	**ROI Financeiro (ROIF1=((VMR-CTOR)/CTOR)*100) e (ROIF2=(ROIF1/100)+1)**	%	216,46	3,16	

SIMULAÇÃO CASO MODERADO

SPIREM-OBK Software Process Improvement ROI Estimate Model Oriented by Knowledge

1. Dados da Organização — MODERADO

Produtividade (%)	20
Horas de trabalho por dia	8
Valor médio homem-hora ($ valor monetário)	20
Dias trabalhados no mês	30
Valor médio da Receita/orçamento para MPS	500000
Custo de Terceiros (Consultoria)	70000
Margem de Risco (%)	30

2.0 Contagem de Processos de Software (Complexidade)

Classif	Processos	Peso	Qtd. Processos	Valor
Baixo	<=7 processos	1,0	2	2
Médio	>= 8 processos e <= 15 processos	1,5	0	0
Alto	>= 16 processos	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		**Total CPS**	2	2

Complexidade do Processo de Software

3. Quantidade de engenheiros de software na equipe do SEPG

Classif	Pessoas	Peso	Qtd. Pessoas	Valor
Baixo	<= 3 pessoas	1,0	2	2
Médio	>= 4 pessoas e <= 8 pessoas	1,5	0	0
Alto	>= 9 pessoas	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		**Total QES**	2	2

Quantidade de Engenheiro de Software

TPE	1,00

Total de Processo por Engenheiro

4. Fatores de Processos

Fator	Descrição	Peso	Atribuído	Valor	Atributos
FP1	Existe processo de software *	2,25	5	11,25	0 - Influência Nnula
FP2	Existe Métricas no processo de software	2,25	4	9,00	1 - Influência Mínima
FP3	Existe Automação no processo de software	3,00	3	9,00	2 - Influência Baixa
FP4	Existe Processo institucionalizado	3,00	5	15,00	3 - Influência Média
FP5	Existe dados históricos de Projetos de software (Post Mortem)	3,00	4	12,00	4 - Influência Alta
FP6	Existe Inspeções no processo de software	3,00	5	15,00	5 - Influência Máxima
FP7	Revisões são realizadas gerando oportunidade de melhoria	2,25	4	9,00	Peso
FP8	Existe Normas e Procedimentos do processo de software	1,50	4	6,00	0,75 - Esforço Mínimo
FP9	Ferramentas de Estimativas são utilizadas	3,00	4	12,00	1,50 - Esforço Moderado
FP10	Existe um objetivo estratégico para melhoria de processo	0,75	4	3,00	2,25 - Esforço Forte
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorT	101,25	3,00 - Esforço Máximo
			FP	131,625	

FP = Fator de Processo

5. Fatores Ambientais

Fator	Descrição	Peso	Atribuído	Valor	Atributos
FA1	A organização provê Treinamento e Orientação	3,00	4	12	0 - Influência Nnula
FA2	Existe Apoio da alta Direção	0,75	5	3,75	1 - Influência Mínima
FA3	Existe Consultoria externa	3,00	0	0	2 - Influência Baixa
FA4	A organização possui Sistema de recompensa	3,00	3	9	3 - Influência Média
FA5	Existe Controle rigoroso das atividades	3,00	4	12	4 - Influência Alta
FA6	Existe Liderança interna	2,25	4	9	5 - Influência Máxima
FA7	Existe o Comprometimento da organização	3,00	5	15	Peso
FA8	Existe colaboração entre as equipes	1,50	4	6	0,75 - Esforço Mínimo
FA9	Existe uma política de banco de horas	0,75	4	3	1,50 - Esforço Moderado
FA10	Existe recursos para o programa de MPS	0,75	4	3	2,25 - Esforço Forte
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorA	72,75	3,00 - Esforço Máximo
			FA	94,575	

FA = Fator Ambiental

6. Fatores Humanos

Fator	Descrição	Peso	Atribuído	Valor
FH1	Existe conhecimento técnico	1,50	5	7,5
FH2	Equipe com dedicação apropriada	0,75	4	3
FH3	Equipe está motivada	0,75	4	3
FH4	Equipe está comprometida	0,75	4	3
FH5	Equipe tem experiência em MPS	3,00	5	15
FH6	Equipe conhece suas responsabilidades	3,00	4	12
FH7	Existe pró-atividade na equipe	1,50	4	6
FH8	Equipe é auto-gerenciada	3,00	4	12
FH9	Equipe é resistente a mudança	3,00	4	12
FH10	Existe rotatividade de pessoas na organização	0,75	4	3
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorH	76,5
			FH	99,45

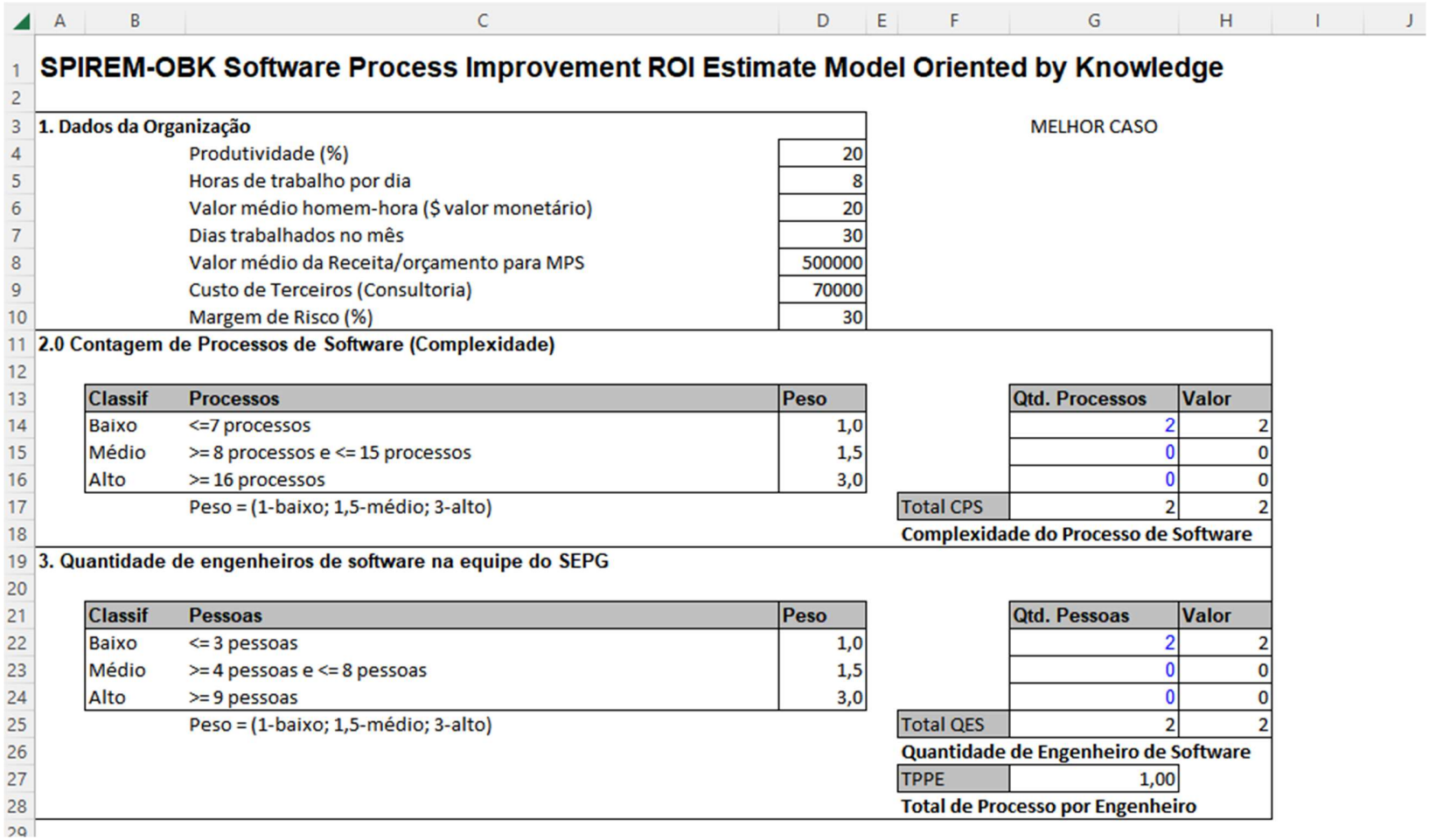
FH = Fator Humano

Atributos
0 - Influência Nnula
1 - Influência Mínima
2 - Influência Baixa
3 - Influência Média
4 - Influência Alta
5 - Influência Máxima
Peso
0,75 - Esforço Mínimo
1,50 - Esforço Moderado
2,25 - Esforço Forte
3,00 - Esforço Máximo

7. Estimar o ROI

7.1	TFA	FP + FA + FH		325,65	
7.2	Total de Fator * Total Processo pro Engenheiro de Software (TE=TFA*TPE)			325,65	
7.3	Calcular a Produtividade ROI (PRDR=TE/Produtividade)			16	
7.4	Esforço = Produtividade * 100% do programa de MPS (THMPS=PRDR*100)			1600	horas
7.5	Prazo estimado em dias PZD=(THMPS/(TE * HT))			200	Dias
7.6	Custo estimado (CTOR=THMPS*VMH) + terceiros			134000,00	$
7.7	ROI em tempo (ROIT = PZD / DT)			6	meses
7.8	ROI Financeiro (ROIF1=((VMR-CTOR)/CTOR)*100) e (ROIF2=(ROIF1/100)+1)		%	273,13	3,73

SIMULAÇÃO DO MELHOR CASO



SPIREM-OBK Software Process Improvement ROI Estimate Model Oriented by Knowledge

1. Dados da Organização — MELHOR CASO

Produtividade (%)	20
Horas de trabalho por dia	8
Valor médio homem-hora ($ valor monetário)	20
Dias trabalhados no mês	30
Valor médio da Receita/orçamento para MPS	500000
Custo de Terceiros (Consultoria)	70000
Margem de Risco (%)	30

2.0 Contagem de Processos de Software (Complexidade)

Classif	Processos	Peso	Qtd. Processos	Valor
Baixo	<=7 processos	1,0	2	2
Médio	>= 8 processos e <= 15 processos	1,5	0	0
Alto	>= 16 processos	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total CPS	2	2

Complexidade do Processo de Software

3. Quantidade de engenheiros de software na equipe do SEPG

Classif	Pessoas	Peso	Qtd. Pessoas	Valor
Baixo	<= 3 pessoas	1,0	2	2
Médio	>= 4 pessoas e <= 8 pessoas	1,5	0	0
Alto	>= 9 pessoas	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total QES	2	2

Quantidade de Engenheiro de Software

TPPE	1,00

Total de Processo por Engenheiro

4. Fatores de Processos

Fator	Descrição	Peso
FP1	Existe processo de software *	0,75
FP2	Existe Métricas no processo de software	0,75
FP3	Existe Automação no processo de software	0,75
FP4	Existe Processo institucionalizado	0,75
FP5	Existe dados históricos de Projetos de software (Post Mortem)	0,75
FP6	Existe Inspeções no processo de software	0,75
FP7	Revisões são realizadas gerando oportunidade de melhoria	0,75
FP8	Existe Normas e Procedimentos do processo de software	0,75
FP9	Ferramentas de Estimativas são utilizadas	0,75
FP10	Existe um objetivo estratégico para melhoria de processo	0,75

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor	Atributos
5	3,75	0 - Influência Nnula
4	3,00	1 - Influência Mínima
4	3,00	2 - Influência Baixa
5	3,75	3 - Influência Média
4	3,00	4 - Influência Alta
5	3,75	5 - Influência Máxima
4	3,00	Peso
4	3,00	0,75 - Esforço Mínimo
4	3,00	1,50 - Esforço Moderado
4	3,00	2,25 - Esforço Forte
FatorT	32,25	3,00 - Esforço Máximo
FP	41,925	

FP = Fator de Processo

5. Fatores Ambientais

Fator	Descrição	Peso
FA1	A organização provê Treinamento e Orientação	0,75
FA2	Existe Apoio da alta Direção	0,75
FA3	Existe Consultoria externa	0,75
FA4	A organização possui Sistema de recompensa	0,75
FA5	Existe Controle rigoroso das atividades	0,75
FA6	Existe Liderança interna	0,75
FA7	Existe o Comprometimento da organização	0,75
FA8	Existe colaboração entre as equipes	0,75
FA9	Existe uma política de banco de horas	0,75
FA10	Existe recursos para o programa de MPS	0,75

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor	Atributos
4	3	0 - Influência Nnula
5	3,75	1 - Influência Mínima
0	0	2 - Influência Baixa
3	2,25	3 - Influência Média
4	3	4 - Influência Alta
4	3	5 - Influência Máxima
5	3,75	Peso
4	3	0,75 - Esforço Mínimo
4	3	1,50 - Esforço Moderado
4	3	2,25 - Esforço Forte
FatorA	27,75	3,00 - Esforço Máximo
FA	36,075	

FA = Fator Ambiental

6. Fatores Humanos

Fator	Descrição	Peso	Atribuído	Valor	Atributos
FH1	Existe conhecimento técnico	0,75	5	3,75	0 - Influência Nnula
FH2	Equipe com dedicação apropriada	0,75	4	3	1 - Influência Mínima
FH3	Equipe está motivada	0,75	4	3	2 - Influência Baixa
FH4	Equipe está comprometida	0,75	4	3	3 - Influência Média
FH5	Equipe tem experiência em MPS	0,75	5	3,75	4 - Influência Alta
FH6	Equipe conhece suas responsabilidades	0,75	4	3	5 - Influência Máxima
FH7	Existe pró-atividade na equipe	0,75	4	3	Peso
FH8	Equipe é auto-gerenciada	0,75	3	2,25	0,75 - Esforço Mínimo
FH9	Equipe é resistente a mudança	0,75	4	3	1,50 - Esforço Moderado
FH10	Existe rotatividade de pessoas na organização	0,75	4	3	2,25 - Esforço Forte
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorA	30,75	3,00 - Esforço Máximo
			FH	39,975	

FH = Fator Humano

7. Calculo do ROI

7.1	TFA	FP + FA + FH	117,98		
7.2	Total de Fator * Total Processo pro Engenheiro de Software (TE=TFA*TPE)		117,98		
7.3	Calcular a Produtividade ROI (PRDR=TE/Produtividade)		5		
7.4	Esforço = Produtividade * 100% do programa de MPS (THMPS=PRDR*100)		500	horas	
7.5	Prazo estimado em dias PZD=(THMPS/(TE * HT))		63	Dias	
7.6	Custo estimado (CTOR=THMPS*VMH) + terceiros		90000,00	$	
7.7	ROI em tempo (ROIT = PZD / DT)		2	meses	
7.8	ROI Financeiro (ROIF1=((VMR-CTOR)/CTOR)*100) e (ROIF2=(ROIF1/100)+1)	%	455,56	R$ 5,56	

SPIREM-OBK Software Process Improvement ROI Estimate Model Oriented by Knowledge

1. Dados da Organização

ESTUDO DE CASO

Produtividade (%)	10
Horas de trabalho por dia	8
Valor médio homem-hora ($ valor monetário)	21,42
Dias trabalhados no mês	22
Valor médio da Receita/orçamento para MPS	300000
Custo de Terceiros (Consultoria)	0
Margem de Risco (%)	30

2.0 Contagem de Processos de Software (Complexidade)

Classif	Processos	Peso	Qtd. Processos	Valor
Baixo	<=7 processos	1,0	7	7
Médio	>= 8 processos e <= 15 processos	1,5	0	0
Alto	>= 16 processos	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total CPS	7	7

Complexidade do Processo de Software

3. Quantidade de engenheiros de software na equipe do SEPG

Classif	Pessoas	Peso	Qtd. Pessoas	Valor
Baixo	<= 3 pessoas	1,0	0	0
Médio	>= 4 pessoas e <= 8 pessoas	1,5	6	9
Alto	>= 9 pessoas	3,0	0	0
	Peso = (1-baixo; 1,5-médio; 3-alto)		Total QES	6	9

Qtd. de Engenheiro de Software

TPPE	0,78

Total Esforço de Processo por Engenheiro

4. Fatores de Processos

Fator	Descrição	Peso
FP1	Existe processo de software *	1,50
FP2	Existe Métricas no processo de software	1,50
FP3	Existe Automação no processo de software	1,50
FP4	Existe Processo institucionalizado	3,00
FP5	Existe dados históricos de Projetos de software (Post Mortem)	3,00
FP6	Existe Inspeções no processo de software	1,50
FP7	Revisões são realizadas gerando oportunidade de melhoria	3,00
FP8	Existe Normas e Procedimentos do processo de software	1,50
FP9	Ferramentas de Estimativas são utilizadas	1,50
FP10	Existe um objetivo estratégico para melhoria de processo	0,75

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor	Atributos
5	7,50	0 - Influência Nnula
4	6,00	1 - Influência Mínima
4	6,00	2 - Influência Baixa
5	15,00	3 - Influência Média
4	12,00	4 - Influência Alta
5	7,50	5 - Influência Máxima
4	12,00	Peso
4	6,00	0,75 - Esforço Mínimo
4	6,00	1,50 - Esforço Moderado
4	3,00	2,25 - Esforço Forte
FatorT	81	3,00 - Esforço Máximo
FP	105,3	

FP = Fator de Processo

5. Fatores de Ambiente

Fator	Descrição	Peso
FA1	A organização provê Treinamento e Orientação	0,75
FA2	Existe Apoio da alta Direção	0,75
FA3	Existe Consultoria externa	0,75
FA4	A organização possui Sistema de recompensa	1,50
FA5	Existe Controle rigoroso das atividades	0,75
FA6	Existe Liderança interna	0,75
FA7	Existe o Comprometimento da organização	0,75
FA8	Existe colaboração entre as equipes	3,00
FA9	Existe uma política de banco de horas	0,75
FA10	Existe recursos para o programa de MPS	0,75

Peso = 0,75; 1,50; 2,25 e 3,0

Atribuído	Valor	Atributos
4	3	0 - Influência Nnula
5	3,75	1 - Influência Mínima
0	0	2 - Influência Baixa
3	4,5	3 - Influência Média
4	3	4 - Influência Alta
4	3	5 - Influência Máxima
5	3,75	Peso
4	12	0,75 - Esforço Mínimo
4	3	1,50 - Esforço Moderado
4	3	2,25 - Esforço Forte
FatorA	39	3,00 - Esforço Máximo
FA	50,7	

FA = Fator de Ambiente

6. Fatores Humanos

Fator	Descrição	Peso	Atribuído	Valor	Atributos
FH1	Existe conhecimento técnico	0,75	5	3,75	0 - Influência Nnula
FH2	Equipe com dedicação apropriada	0,75	4	3	1 - Influência Mínima
FH3	Equipe está motivada	0,75	4	3	2 - Influência Baixa
FH4	Equipe está comprometida	0,75	4	3	3 - Influência Média
FH5	Equipe tem experiência em MPS	0,75	5	3,75	4 - Influência Alta
FH6	Equipe conhece suas responsabilidades	0,75	4	3	5 - Influência Máxima
FH7	Existe pró-atividade na equipe	0,75	4	3	Peso
FH8	Equipe é auto-gerenciada	0,75	3	2,25	0,75 - Esforço Mínimo
FH9	Equipe é resistente a mudança	0,75	4	3	1,50 - Esforço Moderado
FH10	Existe rotatividade de pessoas na organização	0,75	4	3	2,25 - Esforço Forte
	Peso = 0,75; 1,50; 2,25 e 3,0		FatorA	30,75	3,00 - Esforço Máximo
			FH	39,975	

FH = Fator Humano

7. Estimar o ROI

7.1	TFA	FP + FA + FH	195,98		
7.2	Total de Fator * Total Processo p Engenheiro de Software (TE=TFA*TPE)		152,43		
7.3	Calcular a Produtividade ROI (PRDR=int(TE/Produtividade))		15	horas	
7.4	Esforço = Produtividade ROI * 100% do prog de MPS (THMPS=PRDR*100)		1500	Horas	
7.5	Prazo estimado em dias PZD=(THMPS/(TE * HT))		241	Dias	
7.6	Custo estimado (CTOR=THMPS*VMH) + terceiros		192780,00	Valor monetário	
7.7	ROI em tempo (ROIT = PZD / DT)		10	Meses	
7.8	ROI Financeiro (ROIF1=((VMR-CTOR)/CTOR)*100);(ROIF2=(ROIF1/100)+1)	%	55,62	R$ 1,56	

Apêndice B

Lista de arquivos selecionados na RSL

Este documento apresenta a relação dos documentos encontrados ao executar a RSL e a seleção dos documentos para gerar conhecimento para este trabalho.

O Quadro B.1 apresenta o resultado geral de todas as publicações encontradas na execução do protocolo e o resultado do processo de seleção para cada publicação.

Quadro B.1. Lista das publicações

Autor(es)	Título	Site	Ano	1º Filtro	2º Filtro
Biswas, N.N.;	Computer Aided Minimization Procedure for Boolean Functions	IEEE	1984	Não	Não
Hayakawa, A.; Ohno, S.;Oka,K.	Xerographic development simulation on digital halftone images	IEEE	1989	Não	Não
Schupp, J.P.; Cockx, J.; Claesen, L.; De Man, H.	SPI: an open interface integrating highly interactive electronic CAD tools	IEEE	1990	Não	Não
Rombach, M.R.; Solzbach, U.; Tites, U.; Zeiher, A.M.; Wollschlager, H.; Just, H.	PACS for cardiology-perspective or fiction?	IEEE	1991	Não	Não
Leban, M.; Zemva, A.; Zajc, B.	LOM: a logic minimizer for Boolean functions	IEEE	1991	Não	Não
Bhatt, D.; Jha, R.; Steeves, T.; Bhatt, R.; Wills, D.	SPI: an instrumentation development environment for parallel/distributed systems	IEEE	1995	Não	Não
Feitelson, D.G.; Corbett, P.F.; Prost, J.-P.;	Performance of the Vesta parallel file system	IEEE	1995	Não	Não
Boek, C.; Lajbcygier, P.; Palaniswami, M.; Flitman, A.;	A hybrid neural network approach to the pricing of options	IEEE	1995	Não	Não
Debou C., Haux M., Jungmayr S.	A measurement framework for improving verification processes	Scopus	1995	Não	Não
Krasner, H.; Scott, G.	Lessons learned from an initiative for improving software process, quality and reliability in a semiconductor equipment company.	IEEE	1996	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**

Boldyreff, C.; Newman, J.; Taramaa, J.	Managing process improvement in virtual software corporations	IEEE	1996	Não	Não
Pressman, R.;	Software process impediment	IEEE	1996	Não	Não
Herbsleb, J.D.; Goldenson, D.R.;	A systematic survey of CMM experience and results	IEEE	1996	SIM	Não
Rule, P.G.;	Using Jackson methods to implement software process improvement	IEEE	1996	Não	Não
Oh, S.W.; Cheah, K.L.; Li, K.H.; Chia-Lu Ho;	Application of saddlepoint integration on forward-link analysis of asynchronous cellular DS-CDMA over Rayleigh-faded channels	IEEE	1997	Não	Não
Wang, Y.; Court, I.; Ross, M.; Staples, G.; King, G.; Dorling, A.;	Quantitative analysis of compatibility and correlation of the current SPA/SPI models	IEEE	1997	Não	Não
Wang, Y.; Court, I.; Ross, M.; Staples, G.; King, G.; Dorling, A.;	Quantitative evaluation of the SPICE, CMM, ISO 9000 and BOOTSTRAP	IEEE	1997	Não	Não
Sakamoto, K.; Nakakoji, K.; Takagi, Y.; Niihara, N.	Toward computational support for software process improvement activities	IEEE	1998	Não	Não
Oh, S.W.; Li, K.H.	Application of saddlepoint integration on forward-link analysis of asynchronous cellular DS-CDMA over fading channels	IEEE	1998	Não	Não
Jakobsen, A.B.;	Bottom-up process improvement tricks	IEEE	1998	Não	Não
Ernst, R.; Ziegenbein, D.; Richter, K.; Thiele, L.; Teich, J.	Hardware/software codesign of embedded systems the SPI workbench	IEEE	1999	Não	Não
Boria, J.L.; Bianchi, A.J.	Software process improvement under duress: experiences of SPI in a town hall in Argentina	IEEE	1999	Não	Não
Jokikyyny, T.; Lassenius, C.	Using the Internet to communicate software metrics in a large organization	IEEE	1999	Não	Não
Sawyer, P.; Sommerville, I.; Viller, S.	Capturing the benefits of requirements engineering	IEEE	1999	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Bodei, C.; Degano, P.;	Authentication via localized names	IEEE	1999	Não	Não

Focardi, R.; Priami, C.					
Kelly, D.P.; Culleton, B.	Process improvement for small organizations	IEEE	1999	Não	Não
Stokes, J.W.; Ritcey, J.A.;	Evaluation of error probabilities for general signal constellations	IEEE	1999	Não	Não
Sooyong Lee; Jangwook Lee; Woojin Chung; Munsang Kim; Chong-Won Lee; Mignon Park;	A new exoskeleton-type masterarm with force reflection: controller and integration	IEEE	1999	Não	Não
Sooyong Lee; Jangwook Lee; Dae-Sung Choi; Munsang Kim; Chong-Won Lee;	The distributed controller architecture for a master arm and its application to teleoperation with force feedback	IEEE	1999	Não	Não
Warner, E.S.; Proudler, I.K.;	Penalty function method for reducing weight jitter in adaptive broadband array	IEEE	1999	Não	Não
Harrison W., Raffo D., Settle J., Eickelmann N.	Technology Review: Adapting Financial Measures: Making a Business Case for Software Process Improvement	Scopus	1999	Não	Não
CChristof Ebert, Thomas Liedtke, Ekkehard Baisch	Improving reliability of large software systems	ACM	1999	Não	Não
Warren harrison, David Raffo, John Settle, Nancy Eickelmann	Technology Review: Adapting Financial Measures: Making a Business Case for Software Process Improvement	ACM	1999	Não	Não
Abrahamsson, P.	Is management commitment a necessity after all in software process improvement?	IEEE	2000	SIM	Não
Kautz, K.; Nielsen, P.A.	Implementing software process improvement: two cases of technology transfer	IEEE	2000	SIM	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Arent, J.; Norbjerg, J.	Software process improvement as organizational knowledge creation: a multiple case analysis	IEEE	2000	SIM	Não
Jersak, M.; Ying Cai;	A transformational approach to constraint relaxation of a time-driven simulation model	IEEE	2000	Não	Não

Ziegenbein, D.; Ernst, R.;					
O'Hara, F.;	European experiences with software process improvement	IEEE	2000	Não	Não
Arent, J.; Iversen, J.H.; Andersen, C.V.; Bang, S.;	Project assessments: supporting commitment, participation, and learning in software process improvement	IEEE	2000	Não	Não
Sinha, A.; Karmakar, A.; Maiti, K.; Halder, P.	A reconfigurable architecture for a class of digital signal/image processing applications	IEEE	2001	Não	Não
Abrahamsson, P.	Commitment development in software process improvement: critical misconceptions	IEEE	2001	Não	Não
Makinen, T.; Varkoi, T.; Jaakkola, H.	Assessment of a software process assessment process	IEEE	2001	Não	Não
Scott, L.; Jeffery, R.; Carvalho, L.; D'Ambra, J.; Rutherford, P.	Practical software process improvement - the IMPACT project	IEEE	2001	Não	Não
Blanco, M.; Gutierrez, P.; Satriani, G.	SPI patterns: learning from experience	IEEE	2001	SIM	Não
Enzenhofer, W.; Chroust, G.	Best practice approaches in know-how and technology transfer methods for manufacturing SMEs	IEEE	2001	Não	Não
Lepasaar, M.; Kalja, A.; Varkoi, T.; Jaakkola, H.	Key success factors of a regional software process improvement programme	IEEE	2001	Não	Não
Kautz, K.; Ramzan, F.	Software quality management and software process improvement in Denmark	IEEE	2001	Não	Não
Abrahamsson Pekka	Commitment Development in Software Process Improvement: Critical Misconceptions	ACM	2001	SIM	Não
Cater-Steel, A.P.;	Process improvement in four small software companies	IEEE	2001	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Jaakkola, H.; Varkoi, T.; Lepasaar, M.; Makinen, J.;	Diffusion and infusion of software process improvements	IEEE	2001	Não	Não
Lee, R.; Mizuno, T.; Togashi, A.;	Managing evolution of software engineering development environments	IEEE	2001	Não	Não

Christof Ebert, Casimiro Hernandez Parro, Roland Suttels, Harald Kolarczyk	Improving validation activities in a global software development	ACM	2001	Não	Não
Stanbrook, T.	TRIZ for software process improvement	IEEE	2002	Não	Não
Abrahamsson, P.; Kautz, K.	The personal software process: experiences from Denmark	IEEE	2002	Não	Não
Abrahamsson, P.; Iivari, N.	Commitment in software process improvement - in search of the process	IEEE	2002	Não	Não
Reidar Conradi, Alfonso Fuggetta	Improving Software Process Improvement	ACM	2002	SIM	Não
Siakas, K.V.;	What has culture to do with SPI?	IEEE	2002	Não	Não
Pei Zheng; Ni, L.M.	The impact of non-DS domains in a multi-domain DiffServ network	IEEE	2002	Não	Não
Jalote, P.;	Lessons learned in framework-based software process improvement	IEEE	2002	Não	Não
Karlstrom, D.; Runeson, P.; Wohlin, C.;	Aggregating viewpoints for strategic software process improvement-a method and a case study	IEEE	2002	SIM	Não
Richardson I.	SPI Models: What Characteristics are Required for Small Software Development Companies?	Scopus	2002	Não	Não
McGibbon T., Nicholls D.	Making the (Business) case for software reliability	Scopus	2002	Não	Não
Klappholz, D.; Bernstein, L.; Port, D.	Assessing attitude towards, knowledge of, and ability to apply, software development process.	IEEE	2003	Não	Não
Borjesson, A.; Mathiassen, L.	Making SPI happen: the IDEAL distribution of effort.	IEEE	2003	SIM	Não
Niazi, M.; Wilson, D.; Zowghi, D.;	A model for the implementation of software process improvement: a pilot study	IEEE	2003	SIM	Não

Autor(es)	Título	Site	Ano	1º Filtro	2º Filtro
Serrano, M.A.; de Oca, C.M.; Cedillo, K.;	An experience on using the team software process for implementing the Capability Maturity Model for software in a small organization	IEEE	2003	Não	Não
Borjesson, A.; Mathiassen, L.	Successful process implementation	IEEE	2004	SIM	Não
Liu Sheng-mei; Zhao Chun-ming; Li Can-wei;	A novel PAR reduction approach for OFDM system	IEEE	2004	Não	Não

Das, D.	IPSec-based delegation protocol and its application	IEEE	2004	Não	Não
Harjumaa L.; Tervonen I.; Vuorio Pekka;	Improving Software Inspection Process with Patterns	ACM	2004	SIM	Não
Kurniawati, F.; Jeffery, R.;	The long-term effects of an EPG/ER in a small software organization	IEEE	2004	Não	Não
Pozza, D.; Sisto, R.; Durante, L.	Spi2Java: automatic cryptographic protocol Java code generation from spi calculus	IEEE	2004	Não	Não
Baldamus, M.; Parrow, J.; Victor, B.	Spi calculus translated to π-calculus preserving may-tests	IEEE	2004	Não	Não
Serrano, M.A.;	State of the art and future of research in software process improvement	IEEE	2004	Não	Não
Revesz, P.; Wu, S.;	Visualization of recursively defined concepts	IEEE	2004	Não	Não
Cater-Steel, A.P.;	Low-rigour, rapid software process assessments for small software development firms	IEEE	2004	Não	Não
Khirallah, C.; Coulton, P.; Ruan, J.;	A novel interference cancellation technique for MIMO CC-CDMA system	IEEE	2004	Não	Não
Card, D.N.;	Research directions in software process improvement	IEEE	2004	SIM	Não
Harjumaa L., Tervonen I., Vuorio P.	Using software inspection as a catalyst for SPI in a small company	Scopus	2004	Não	Não
Rini Van Solingen	Measuring the ROI of Software Process Improvement	ACM	2004	SIM	Não
Mathiassen, L.; Ngwenyama, O.K.; Aaen, I.	Managing change in software process improvement	IEEE	2005	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
McCaffery, F.; McFall, D.; Donnelly, P.; Wilkie, F.G.; Sterritt, R.	A software process improvement lifecycle framework for the medical device industry	IEEE	2005	Não	Não
Madsen, A.L.	Variations over the message computation algorithm of lazy propagation	IEEE	2005	Não	Não
Nikula, U.; Sajaniemi, J.	Tackling the complexity of requirements engineering process improvement by partitioning the improvement task	IEEE	2005	Não	Não
Lu, Y.H.; Gu, Y.G.; Chen, X.R.; Fu, Y.	Analyzing security protocols by a bisimulation method based on environmental knowledge	IEEE	2005	Não	Não

Jaume, M.; Morisset, C.	Formalisation and implementation of access control models	IEEE	2005	Não	Não
Yonggen Gu; Yuxi Fu	A Simple Process Calculus for the analysis of Security Protocols	IEEE	2005	Não	Não
Zhongcheng Wu; Enliang Song; Fei Shen; Dezhang Xu; Bing Fang	The biological inspired somatic neuron design and its application in robot nervous system	IEEE	2005	Não	Não
Yoonjung Choi; EunSeok Lee; Sujung Ha	The management of software processes with software process improvement tool based on ISO 15504	IEEE	2005	SIM	Não
Dyba, T.	An empirical investigation of the key factors for success in software process improvement	IEEE	2005	SIM	Não
Tao Li; Cheolwoo Jo	Parameter changes across different emotions in human speech	IEEE	2005	Não	Não
Gihan Kim, Minkwang Lee, Jongphil Lee, Kyungwhan Lee	Design of SPICE experience factory model for accumulation and utilization of process assessment experience	IEEE	2005	Não	Não
Yongjun Xu; Lingyi Liu; Peifu Shen; Tao Lv; Xiaowei Li;	Low power processor design for wireless sensor network applications	IEEE	2005	Não	Não
Haverinen, J.; Parpala, M.; Roning, J.;	A Miniature Mobile Robot With a Color Stereo Camera System for Swarm Robotics Research	IEEE	2005	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Ylitalo, J.; Salmela, P.; Tschofenig, H.;	SPINAT: Integrating IPsec into Overlay Routing	IEEE	2005	Não	Não
Qing Wang; Mingshu Li;	Measuring and improving software process in China	IEEE	2005	Não	Não
Salo, O.; Abrahamsson, P.;	Integrating agile software development and software process improvement: a longitudinal case study	IEEE	2005	Não	Não
Lee, J.W.; Jung, S.H.; Park, S.C.; Lee, Y.J.; Jang, Y.C.;	System based SQA and implementation of SPI for successful projects	IEEE	2005	Não	Não
Muzik, O.; Pourabdollah, S.; Juhasz, C.;	Application of an objective method for localizing bilateral cortical FDG PET	IEEE	2005	Não	Não

Chugani, D.C.; Janisse, J.; Draghici, S.;	abnormalities to guide the resection of epileptic foci				
Popa, M.; Popa, A.S.; Cretu, V.; Micea, M.	Monitoring Serial Communications in Microcontroller Based Embedded System	IEEE	2006	Não	Não
Henderson-Sellers, B.	SPI - A Role for Method Engineering	IEEE	2006	SIM	Não
Ming-Fa Tsai; Fu-Jing Ke; Ying-De Lin; Jui-Kum Wang	Design of a Digital Programmable Control IC for Single-Phase Controlled Rectifiers	IEEE	2006	Não	Não
Alexandre, S.; Renault, A.; Habra, N.	OWPL: A Gradual Approach for Software Process Improvement in SMEs.	IEEE	2006	SIM	Não
Subramanian, S.; Johnson, C.A.; Devasahayam, N.; Matsumoto, K.; Hyodo, F.; Cook, J.; Krishna, M.C.	In vivo spectral-spatial imaging for oxygen mapping using single-point, time-domain electron paramagnetic resonance	IEEE	2006	Não	Não
Donoghue, J.	A direct brain interface to restore function in humans with spinal cord injury.	IEEE	2006	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Xiong, N.; Yanxiang He; Yan Yang; Yang, L.T.; Chao Peng;	A Self-Tuning Multicast Flow Control Scheme Based on Autonomic Technology	IEEE	2006	Não	Não
Mascella, R.; Tallini, L.G.;	Efficient m-ary balanced codes which are invariant under symbol permutation	IEEE	2006	Não	Não
Ping-Ping Xiao; Yan-Tao Tian;	A Congestion Control Approach Based on Predictive Model in Delay Network	IEEE	2006	Não	Não
Sankalita Saha; Shuvra S. Bhattacharyya; Wayne Wolf;	A Communication Interface for Multiprocessor Signal Processing Systems	IEEE	2006	Não	Não
Hao Wang; Yizhu Tong; Hong Liu; Taoying Liu;	Application-aware Interface for SOAP Communication in Web Services	IEEE	2006	Não	Não

Seungyong Yoon; Jintae Oh; Jongsoo Jang;	Design of SPI module in large-scale network	IEEE	2006	Não	Não
Yili Fu; Fuxiang Zhang; Shuguo Wang; Qinggang Meng;	Development of an Embedded Control Platform of a Continuous Passive Motion Machine	IEEE	2006	Não	Não
Damm L.-O., Lundberg L.	Results from introducing component-level test automation and Test-Driven Development	Scopus	2006	Não	Não
van Solingen R., Rico D.F.	Calculating Software Process Improvement's Return on Investment	Scopus	2006	SIM	Não
Barbara Kitchenham	Empirical paradigma – The role of experiments	ACM	2006	Não	Não
Alok Mishra, Deepti Mishra	Software quality assurance models in small and médium organizations: a comparison	ACM	2006	Não	Não
Patrick Keil, Marco kuhrmann	An approach to model the return on investment of organization-wide improvement projects using the concept of externa effects	ACM	2006	Não	Não
Han-Wen Tuan; Chia-Yi Liu; Chiou-Mei Chen;	Using ABC Model for Software Process Improvement: A Balanced Perspective	IEEE	2006	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
da Rocha, A.R.C.; Montoni, M.; Weber, K.C.; de Araujo, E.E.R.	A Nationwide Program for Software Process Improvement in Brazil	IEEE	2007	SIM	Não
Min Gan; Guoying Yue; Hongfa Wang	Network Protocols Analyzing by an Environmental Bisimulation Method	IEEE	2007	Não	Não
Sumathi, M.; Krishna, M.C.; Murugesan, R.	Evolutionary Computational Approach for Artifact-Free Image Reconstruction from Reduced Samples: Application to Fourier EMRI	IEEE	2007	Não	Não
Jiangping Wan; Yunfeng Wang; Chuwei Zheng	Research on IT Service Management Knowledge Support Structure	IEEE	2007	Não	Não
Zaabar, Imen; Berregeb, Narjes	Reasoning about Cryptographic Protocols in Observational Theories	IEEE	2007	Não	Não
Lu, D.D.-C.	Synthesis of Single Phase DC/AC Inverters	IEEE	2007	Não	Não

Liang, S.-A.	Design Optimization for LCD TV Power Supply with Resonant Technique	IEEE	2007	Não	Não
Montoni, M.; Santos, G.; Rocha, A.R.; Weber, K.C.; de Araujo, E.E.	MPS Model and TABA Workstation: Implementing Software Process Improvement Initiatives in Small Settings	IEEE	2007	SIM	Não
Martins, P.V.; da Silva, A.R.;	A comparative study of SPI Approaches with ProPAM	IEEE	2007	SIM	Não
Alagarsamy, K.; Justus, S.; Iyakutti, K.	On the Implementation of a Knowledge Management Tool for SPI	IEEE	2007	Não	Não
Szekacs, A.; Szakall, T.; Hegykozi, Z.;	Realising the SPI communication in a multiprocessor system	IEEE	2007	Não	Não
Dong-Hong Xua; Yong Qia; Di Houa; Ying Chen; Liang Liu;	Towards formal basis for security aspects of dynamic web services composition	IEEE	2007	Não	Não
Diez, D.; Fernandez, C.; Dodero, J.M.; Diaz, P.; Aedo, I.;	Instructional Software Analysis: Lessons from Software Development Process Improvement	IEEE	2007	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
San Kim; Dong Hwan Kim;	A design and implementation of moving object tracking system for omni-directional robot	IEEE	2007	Não	Não
Alagarsamy, K.; Justus, S.; Iyakutti, K.	The Knowledge Based Software Process Improvement Program: A Rational Analysis	IEEE	2007	Não	Não
Savage, S.;	Implementing high-speed serial and custom digital protocols thru fpga technology and graphical programming techniques	IEEE	2007	Não	Não
Shyh-Shyuan Sheu; Lieh-Chiu Lin; Wen-Han Wang; Pei-Chia Chiang; Keng-Li Su; Ming-Jer Kao; Ming-Jinn Tsai;	4-Mb SPI Flash Compatible Phase-Change Memory	IEEE	2007	Não	Não
Dong-Hong Xu; Yong Qi; Di Hou; Ymg	A Formal Model for Security-Aware dynamic Web Services Composition	IEEE	2007	Não	Não

Chen; Liang Liu;					
Santos, G.; Montoni, M.; Vasconcellos, J.; Figueiredo, S.; Cabral, R.; Cerdeiral, C.; Katsurayama, A.E.; Lupo, P.; Zanetti, D.; Rocha, A.R.;	Implementing Software Process Improvement Initiatives in Small and Medium-Size Enterprises in Brazil	IEEE	2007	SIM	Não
Jianguo Wang Yun Bai; Xiuyu Zhang;	Design of Wireless Data Communication System Based on PTR8000	IEEE	2007	Não	Não
Dong-Hong Xu; Yong Qi; Di Hou; Ying Chen; Liang Liu;	A Formal Model for dynamic Web Services Composition MAS-Based and Simple Security Analysis Using Spi Calculus	IEEE	2007	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Ferreira A.I.F., Santos G., Cerqueira R., Montoni M., Barreto A., Soares Barreto A.O., Rocha A.R.	Applying ISO 9001:2000, MPS.BR and CMMI to achieve software process maturity: BL informatica's pathway	Scopus	2007	Não	Não
Subramanian G.H., Jiang J.J., Klein G.	Software quality and IS project performance improvements from software development process maturity and IS implementation strategies	Scopus	2007	Não	Não
Holzmann M.T.	Is SPI the yield improvement tool we've waited for?	Scopus	2007	Não	Não
Mark Staples, Mahmood Niazi, Ross Jeffery, Alan Abrahams, Paul Byatt, Russell Murphy	An Exploratory study of why organizations do not adopt CMMI	ACM	2007	Não	Não
I. Allison, Y. Merali	Software process improvement as emergentes change: A structural analysis	ACM	2007	Não	Não
C. Verhoef	Quantifying the effects of IT-governance rules	ACM	2007	Não	Não
Taek Lee; Dookwon Baik; In, H.P.	Cost Benefit Analysis of Personal Software Process Training Program	IEEE	2008	SIM	Não

Habib, M.; Ahmed, S.; Rehmat, A.; Khan, M.J.; Shamail, S.	Blending Six Sigma and CMMI - an approach to accelerate process improvement in SMEs	IEEE	2008	SIM	Não
Micea, M.V.; Certejan, C.; Stangaciu, V.; Cioarga, R.; Cretu, V.; Petriu, E.	Inter-task communication and synchronization in the hard real-time compact kernel HARETICK	IEEE	2008	Não	Não
Vanamali, B.; Bella, F.; Hormann, K.	From CMMI to SPICE - Experiences on How to Survive a SPICE Assessment Having Already Implemented CMMI	IEEE	2008	Não	Não
Iwami, Y.	How to Measure Quality of Software Developed by Subcontractors (Short Paper)	IEEE	2008	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Dong-Hong Xu; Yong Qi; Di Hou; Gong-Zhen Wang; Ying Chen	An Improved Calculus for Secure Dynamic Services Composition	IEEE	2008	Não	Não
Varkoi, T.; Makinen, T.	Proactive elicitation of software process improvements	IEEE	2008	SIM	Não
Mendes, M.L.; Pimenta, L.C.A.; Mesquita, R.C.; Silva, E.J.; Santana, T.C.	Smoothed particle electromagnetics with boundary absorbing condition using perfectly matched layers	IEEE	2008	Não	Não
Babar, M.A.; Niazi, M.	Implementing Software Process Improvement Initiatives: An Analysis of Vietnamese Practitioners' Views	IEEE	2008	Não	Não
Xiaojun Qi; Bialkowski, S.; Brimley, G.	An adaptive QIM- and SVD-based digital image watermarking scheme in the wavelet domain	IEEE	2008	Não	Não
Weiguang Sheng; Liyi Xiao; Zhigang Mao;	Versatile and Efficient Techniques for Speeding-Up Circuit Level Simulated Fault-Injection Campaigns	IEEE	2008	Não	Não
Habib, L.; Jaume, M.; Morisset, C.	A Formal Comparison of the Bell & LaPadula and RBAC Models	IEEE	2008	Não	Não
Akingbehin, K.	Baseline-Based Framework for Continuous Software Process Improvement (CSPI)	IEEE	2008	SIM	Não

Pearson, C.; Giuma, T.; Harris, A.	Transparent Connectivity for Embedded System Design	IEEE	2008	Não	Não
Gaidhane, A.; Khanapurkar, M.M.; Bajaj, P.R.	Design Approach for FPGA Implementation of Flexray Controller Using VHDL for Intra Vehicular Communication Application	IEEE	2008	Não	Não
Xu Dong Hong; Qi Yong; Hou Di; Chen Ying	SpiG4WSC: A Calculus for Secure Services Composition	IEEE	2008	Não	Não
Carrizo, C.; Hatalkar, A.; Memmott, L.; Wood, M.	Design of a context aware computing engine	IEEE	2008	Não	Não
Alagarsamy, K.; Justus, S.; lyakutti, K.	Implementation specification for software process improvement supportive knowledge management tool	IEEE	2008	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Dong-Hong Xu; Yong Qi; Di Hou; Gong-Zhen Wang; Ying Chen	A novel formal framework for secure dynamic services composition	IEEE	2008	Não	Não
Capocchi, L.; Federici, D.; Yazidi, A.; Henao, H.; Capolino, G.A.	Asymmetrical behavior of a double-fed induction generator: Modeling, discrete event simulation and validation	IEEE	2008	Não	Não
Bugliesi, Michele; Focardi, Riccardo	Language Based Secure Communication	IEEE	2008	Não	Não
Rhou, B.; Sawan, M.; Desilets, T.; Bellemare, F.	Real-time filtering technique to remove ECG interference from recorded esophageal EMG	IEEE	2008	Não	Não
Hayashi, A.; Kataoka, N.	A Method to Identify Critical Software Process Improvement Area Using Quality Function Deployment	IEEE	2008	SIM	Não
Saha, S.; Puthenpurayil, S.; Schlessman, J.; Bhattacharyya, S.S.; Wolf, W.;	The Signal Passing Interface and Its Application to Embedded Implementation of Smart Camera Applications	IEEE	2008	Não	Não

Haifeng Wang; Chunjie Li; Lu Wu; Minglei Shu; Jialiang Lv;	Design of portable terminal device in emergency system for turnpike	IEEE	2008	Não	Não
Xuhui Chen; Dengyi Zhang; Hongyun Yang;	Design and Implementation of a Single-Chip ARM-Based USB Interface JTAG Emulator	IEEE	2008	Não	Não
McCaffery, F.; Pikkarainen, M.; Richardson, I.;	Ahaa --agile, hybrid assessment method for automotive, safety critical smes	IEEE	2008	Não	Não
Pironti, A.; Sisto, R.;	Formally Sound Refinement of Spi Calculus Protocol Specifications into Java Code	IEEE	2008	Não	Não
Marton, L.;	Distributed controller architecture for advanced robot control	IEEE	2008	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Micea, M.V.; Carstoiu, G.N.; Ungurean, L.; Chiciudean, D.; Cretu, V.; Groza, V.;	Predictable data communication interface for hard real-time systems	IEEE	2008	Não	Não
Saha, S.; Schlessman, J.; Puthenpurayil, S.; Bhattacharyya, S.S.; Wolf, W.;	An Optimized Message Passing Framework for Parallel Implementation of Signal Processing Applications	IEEE	2008	Não	Não
Porobic, V.B.; Marcetic, D.P.; Katic, V.A.;	Data logging in the electrical drives	IEEE	2008	Não	Não
Makinen, T.; Varkoi, T.;	Assessment driven process modeling for software process improvement	IEEE	2008	Não	Não
Suwanya, S.; Kurutach, W.;	An analysis of software process improvement for sustainable development in Thailand	IEEE	2008	Não	Não
Beijun Shen; Tong Ruan;	A Case Study of Software Process Improvement in a Chinese Small Company	IEEE	2008	Não	Não
Ferreira A.I.F., Santos G., Cerqueira R., Montoni M., Barreto A.,	ROI of software process improvement at BL informática: SPIdex is really worth it	Scopus	2008	SIM	Não

Rocha A.R., Barreto A.O.S., Silva Filho R.C.					
Andre Luiz Becker, Rafale Prikladnicki, Jorge Luis Nicolas Audy	Strategic alignment of software process improvement programs using QFD	ACM	2008	Não	Não
F. Petterson, M. Ivarsson, T. Gorschek, P. Öhman	A practitioner's guide to light weight software process assessment and improvement planning	ACM	2008	Não	Não
R. J. Peters, C. Verhoef	Quantifying the yield of risk-bearing IT-portfolios	ACM	2008	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Da Yang, qing Wang, Mingshu Li, Ye Yang, Kai Ye, Jing Du	A survey on software cost estimation in the chinese software industry	ACM	2008	Não	Não
Pedro E. Colla, Jorge Marcelo Montagna	Framework to evaluate software process improvement in small organizations	ACM	2008	Não	Não
Fruhwirth, C.; Mannisto, T.	Improving CVSS-based vulnerability prioritization and response with context information	IEEE	2009	Não	Não
Blunt, S.D.; Shackelford, A.K.; Gerlach, K.; Smith, K.J.	Doppler Compensation & Single Pulse Imaging using Adaptive Pulse Compression	IEEE	2009	Não	Não
Attarzadeh, I.; Hock, O.S.	Modern Project Management: A New Forecasting Model to Ensure Project Success	IEEE	2009	Não	Não
Iranmanesh, S.H.; Mirseraji, G.H.; Shahmiri, S.;	An Emotional Learning based Fuzzy Inference System (ELFIS) for improvement of the completion time of projects estimation	IEEE	2009	Não	Não
Mongkolnam, P.; Silparcha, U.; Waraporn, N.; Vanijja, V.	A Push for Software Process Improvement in Thailand	IEEE	2009	Não	Não
Chen Run; Huang Shi-zhen; Lin Wei; Li Lei	Design and Implementation of a Reused Interface	IEEE	2009	Não	Não

Flores-Abad, A.; Arpidez, A.	Embedded Control System for a 5-DOF Manipulator by Means of SPI Bus	IEEE	2009	Não	Não
Bengtson, J.; Johansson, M.; Parrow, J.; Victor, B.;	Psi-calculi: Mobile Processes, Nominal Data, and Logic	IEEE	2009	Não	Não
Du Dongmei; He Qing; Tang Bing	ARM-Based Handheld Analyzer for Vibration	IEEE	2009	Não	Não
Shaikh, A.; Ahmed, A.; Memon, N.; Memon, M.	Strengths and Weaknesses of Maturity Driven Process Improvement Effort	IEEE	2009	SIM	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Attarzadeh, I.; Hock, O.S.	Implementation and Evaluation of Earned Value Index to Achieve an Accurate Project Time and Cost Estimation and Improve "Earned Value Management System	IEEE	2009	SIM	Não
Baldassarre, M.T.; Piattini, M.; Pino, F.J.; Visaggio, G.	Comparing ISO/IEC 12207 and CMMI-DEV: Towards a mapping of ISO/IEC 15504-7	IEEE	2009	Não	Não
Harish, K.; Rao, M.V.; Borgohain, R.; Sairam, A.; Abhilash, P.	Tremor quantification and its measurements on parkinsonian patients	IEEE	2009	Não	Não
Guang Xu; Hong An; Gu Liu; Ping Yao; Mu Xu; Wenting Han; Xiaoqiang Li; Xiurui Hao;	Performance and Power Efficiency Analysis of the Symmetric Cryptograph on Two Stream Processor Architectures	IEEE	2009	Não	Não
Yong Qi; Weihua Li; Zhonghua Li	A Classification with Random SPI: Better Models in Uncertain Environment	IEEE	2009	Não	Não
Attarzadeh, I.; Hock, O.S.	A New Forecasting Model to Improve Earned Value Index to Achieve an Accurate Project Time and Cost Estimation	IEEE	2009	SIM	Não
Capocchi, L.; Federici, D.; Yazidi, A.; Henao, H.; Capolino, G.-A.	New trends and solutions for the simulation of electrical circuits using a discrete event approach	IEEE	2009	Não	Não

Ming He; Limin Dong; Wuchen Wu; Lu Kong; Chao Zhu; Zhonghua Zhou	A configurable SoC platform design based on LEON microprocessor and Linux operation system	IEEE	2009	Não	Não
Rahaman, M.S.; Chowdhury, M.H.; Nasir, I.; Lih-Tyng Hwang	VSIB: A Sensor Bus Architecture for Smart-Sensor Network	IEEE	2009	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Hartmann, V.A.; Briggs, K.; Shivarudrappa, S.; Yeager, K.M.; Diaz, R.	The impact of hypoxia on bioturbation rates in the louisiana continental shelf, northern Gulf of Mexico	IEEE	2009	Não	Não
Leens, F.;	An introduction to I^2C and SPI protocols	IEEE	2009	Não	Não
Giannantonio, Roberta; Gravina, Raffaele; Kuryloski, Philip; Seppa, Ville-Pekka; Bellifemine, Fabio; Hyttinen, Jari; Sgroi, Marco;	Performance analysis of an activity monitoring system using the SPINE framework	IEEE	2009	Não	Não
Wang Dafang; Zhu Yueying; Xin Ming; Zhao Guifan;	Research on auto and rapid exchange system of EV battery cases	IEEE	2009	Não	Não
Dingfang Liu; Suiping Qi; Tundong Liu; Shou-zhi Yu; Funchun Sun;	The design and realization of communication technology for streetlamps control system	IEEE	2009	Não	Não
Elmasry, G.F.; Jain, M.; Orlando, R.; Jakubowski, K.; Lo, R.	Joint network management across future force tactical networks	IEEE	2009	Não	Não
Salviano, C.F.;	A Multi-model Process Improvement Methodology Driven by Capability Profiles	IEEE	2009	SIM	Não

Sun Myung Hwang; Hee Gyun Yeom;	Analysis of Relationship among ISO/IEC 15504, CMMI and K-model	IEEE	2009	Não	Não
Du Dongmei; He Qing; Tang Bin; Chen Fei;	Development of window CE driver of high-speed ADC based on SPI	IEEE	2009	Não	Não
Johnson, D.;	Implementing serial bus interfaces with general purpose digital instrumentation	IEEE	2009	Não	Não
Attarzadeh, I.; Hock, O.S.;	Using Enhancement Method to Improve Earned Value Index to Achieve an Accurate Project Time and Cost Estimation	IEEE	2009	SIM	Não
Nikula, U.; Oinonen, P.; Hannola, L.;	Extending Process Improvement into a New Organizational Unit	IEEE	2009	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Gil, A.; Belver, D.; Cabanelas, P.; Castro, E.; Diaz, J.; Garzon, J.A.; Gonzalez-Diaz, D.; Kolb, B.W.; Traxler, M.; Trebacz, R.; Zumbruch, P.;	Control and Monitoring System for the HADES RPC detector	IEEE	2009	Não	Não
Magyari-Saska, Z.; Haidu, I.	Drought and extreme moisture evaluation and prediction with GIS software module	IEEE	2009	Não	Não
Sharma, B.; Sharma, N.;	Software Process Improvement: A Comparative Analysis of SPI models	IEEE	2009	SIM	Não
He Qing; Du Dongmei; Tang Bin;	System design of ARM-based handset vibration analyzer	IEEE	2009	Não	Não
van Solingen R.	A follow-up reflection on software process improvement ROI	Scopus	2009	SIM	Não
Elliott M., Dawson R., Edwards J.	An evolutionary cultural-change approach to successful software process improvement	Scopus	2009	Não	Não
Ljubomir Lazic, Nikos E. Mastorakis	OptimalSQM: integrated and optimized quality management	ACM	2009	Não	Não
Loos, Andreas; Schmidt, Michael; Fey, Dietmar; Gröbel, Jens;	Dynamically Programmable Image Processor for Compact Vision Systems	IEEE	2010	Não	Não

Liu Jun; Yang Ming-xin; Wang Jian-bo	Thick Film Integrated Circuit Design of Multi-measurement Module	IEEE	2010	Não	Não
Sevcik, Bretislav	Modeling and signal integrity testing of digital potentiometers	IEEE	2010	Não	Não
Basri, Shuib Bin; O'Connor, Rory V.	Organizational commitment towards software process improvement an irish software vses case study	IEEE	2010	Não	Não
Tian, Mao; Liu, Wenchao; Zhong, Zhifeng; Pan, Yongcai;	Design and research on the hardware of wireless oxygen saturation detection based on ARM	IEEE	2010	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Hanbai Fan; Shaoxian Wang	Design of expendable conductivity temperature depth survey data processing system	IEEE	2010	Não	Não
Zhao Xiaoqiang; Zhang Zuhou	Development of Remote Waste Gas Monitor System	IEEE	2010	Não	Não
Liang Chen; Zhiqun Li; Qin Li; Li Zhang; Wei Li	A novel spiral inductor model with a new parameter-extraction approach	IEEE	2010	Não	Não
Carstoiu, B.; Carstoiu, D.	High performance eventually consistent distributed database Zatara	IEEE	2010	Não	Não
Xu Donghong; Jiang Shujuan; Qi Yong	Security Properties Analysis of Routing Protocol for MANET	IEEE	2010	Não	Não
Jewell, Ward; Grossardt, Ted; Bailey, Keiron; Gill, Raman;	A new method for public involvement in electric transmission line routing	IEEE	2010	Não	Não
Jianhong Liang; Tianyu Gu;	An optimized realization of resource extension on microcontroller based embedded system for Educational Robotic Kit	IEEE	2010	Não	Não
Shan Chuang; Hou Min-xian;	Wireless landscape lighting control system based on ARM-Linux	IEEE	2010	Não	Não
Bo Qu; Daowei Fan;	Design of remote data monitoring and recording system based on ARM	IEEE	2010	Não	Não
Wu, Jianjun; Zhou, Lei; Zhang, Jie; Liu, Ming; Zhao, Lin; Zhao, Feifei;	Analysis of relationships among vegetation condition indices and multiple-time scale SPI of grassland in growing season	IEEE	2010	Não	Não

Xiao Jinzhuang; Xiong Peng; Liu Kun; Wang Hongrui;	Design and test method of a six-degree of freedom motion collection device	IEEE	2010	Não	Não
Naixue Xiong; Vasilakos, A.V.; Yang, L.T.; Yi Pan; Cheng-Xiang Wang; Vandenberg, A.;	Distributed Explicit Rate Schemes in Multi-Input–Multi-Output Network Systems	IEEE	2010	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
Lapadatu, Daniel; Blixhavn, Bjorn; Holm, Reidar; Kvisteroy, Terje;	SAR500 - A high-precision high-stability butterfly gyroscope with north seeking capability	IEEE	2010	Não	Não
Otón, Salvador; Hilera, José R.; García, Eva; García, Antonio; de-Marcos, Luis; Ortíz, Antonio; Gutiérrez, José A.; Martínez, José J.; Gutiérrez, José M.; Barchino, Roberto;	A Proposal to Improve the Simple Query Interface (SQI) of Learning Objects Repositories	IEEE	2010	Não	Não
Hasan, S.M.S.; Nealy, R.; Brisebois, T.J.; Newman, T.R.; Bose, T.; Reed, J.H.;	Wideband RF front end design considerations for a flexible white space software defined radio	IEEE	2010	Não	Não
Zhao Ruimei; Wang Mei;	Design of ARM-based embedded Ethernet interface	IEEE	2010	Não	Não
Tiu, Alwen; Dawson, Jeremy;	Automating Open Bisimulation Checking for the Spi Calculus	IEEE	2010	Não	Não

Sun-Myung Hwang;	Quality Metrics for Software Process Certification Based on K-model	IEEE	2010	Não	Não
Wong, Kiing-Ing; Barsoum, Nader; Cho Zin Myint;	Teaching the electronic design and embedded system course with body sensor nodes	IEEE	2010	Não	Não
Zhang Yan; Gao Lizhong; He Bingjie;	A General Module for the Detecting and Decoding of Serial Buses	IEEE	2010	Não	Não
Aljanaideh, Omar; Rakheja, Subhash; Su, Chun-Yi;	Compensation of a piezoceramic actuator hysteresis nonlinearities using the stop operator-based Prandtl-Ishlinskii model	IEEE	2010	Não	Não
Autor(es)	**Título**	**Site**	**Ano**	**1º Filtro**	**2º Filtro**
La Mantia, G.; Rossi, D.; Finamore, A.; Mellia, M.; Meo, M.;	Stochastic Packet Inspection for TCP Traffic	IEEE	2010	Não	Não
Sivashankar, M.; Kalpana, A.M.; Jeyakumar, A.E.;	A framework approach using CMMI for SPI to Indian SME'S	IEEE	2010	SIM	Não
Mikeka, C.; Arai, H.;	Development of a batteryless sensor transmitter	IEEE	2010	Não	Não
Yan Sun, Xiaoging (Frank) Liu	Business-oriented software process improvment based on CMMI using QFD	ACM	2010	Não	Não