



Universidade Federal de Pernambuco
Centro de Informática

Pós-Graduação em Ciência da Computação

Uma meta estratégia para melhoria de produtividade no desenvolvimento
de software, baseada em melhores práticas identificadas na literatura.

Dissertação de Mestrado

por

Suzana Cândido de Barros Sampaio

Recife

24 de Maio de 2010

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA

SUZANA CÂNDIDO DE BARROS SAMPAIO

Uma meta estratégia para melhoria de produtividade no desenvolvimento de software, baseado em melhores práticas identificadas na literatura.

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: Prof. Dr. Silvio Lemos Meira

Recife, 24 de maio de 2010

Sampaio, Suzana Cândido de Barros

Uma meta estratégia para melhoria de produtividade no desenvolvimento de software, baseada em melhores práticas identificadas na literatura / Suzana Cândido de Barros Sampaio. - Recife: O Autor, 2010.

viii, 126 folhas : il., fig., tab.

Dissertação (mestrado) Universidade Federal de Pernambuco. CIn. Ciência da Computação, 2010.

Inclui bibliografia.

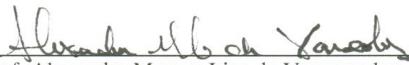
1. Engenharia de software. 2. Gestão de produtividade. I. Título.

005.1

CDD (22. ed.)

MEI2010 – 0145


Dissertação de Mestrado apresentada por **Suzana Cândido de Barros Sampaio** à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **“Uma Meta Estratégica para Melhoria de Produtividade no Desenvolvimento de Software, Baseado em Melhores Práticas Identificadas na Literatura”**, orientada pelo **Prof. Silvio Romero de Lemos Meira** e aprovada pela Banca Examinadora formada pelos professores:



Prof. Alexandre Marcos Lins de Vasconcelos
Centro de Informática / UFPE

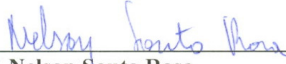


Prof. Jones Oliveira de Albuquerque
Departamento de Estatística e Informática / UFRPE



Prof. Silvio Romero de Lemos Meira
Centro de Informática / UFPE

Visto e permitida a impressão.
Recife, 24 de maio de 2010.



Prof. Nelson Souto Rosa
Coordenador da Pós-Graduação em Ciência da Computação do
Centro de Informática da Universidade Federal de Pernambuco.

AGRADECIMENTOS

Aos professores do CIn - Centro de Informática (CIn) da Universidade Federal de Pernambuco (UFPE), pelas aulas, pelo aprendizado e por todo o conhecimento construído e compartilhado. Em particular, ao Professor Silvio Meira pela oportunidade de fazer parte deste centro de excelência.

Aos amigos do PROSE, CIn e Pitang pela motivação e idéias. Em especial a Gibeon Aquino por abrir as portas para um mundo novo, onde processo e qualidade não eram os únicos objetivos.

Aos gestores e amigos do NEXT pela oportunidade de desaceleração e aprendizado. Meu agradecimento em especial para Ivaldir Júnior que me deu forças em momentos de desespero, a Vitor Costa (o meu Wilson) por simplesmente me ouvir, a Ermeson pela eterna disponibilidade, a Marcos Gomes e Marcelo Clemente pela confiança.

À minha amiga e principal parceira de pesquisa Emanuella Aleixo - Manu - por compartilhar os finais de semana, feriados, carnaval e tantas horas de CIn comigo. Sem você Manu, os últimos três meses de trabalho teriam se transformado em nove. Muito obrigada!

Aos meus irmãos, sobrinhos, sogros, cunhadas e amigos, agradeço o amor, carinho e compreensão pela CONTÍNUA ausência durante esse período.

Aos meus pais pelo amor incondicional, pelos exemplos e por mostraram que a única forma de vencer é através de muita luta. Pela força quando a família multiplicou e pela confiança que sempre me foi depositada. Pai, filho de guerreiro, guerreira é!

A meu marido e filhos pela inspiração de todos os dias. Inspiração esta que me renova as forças e imensa vontade de vencer a cada dia. E por me ajudarem na transformação de oportunidades em conquistas.

Ao pai celestial pela oportunidade que me foi dada de nesta vida vir em um lar equilibrado, mas de guerreiros. E ainda, de permitir que eu encontrasse, ou voltasse a encontrar meu eterno amigo, parceiro e namorado.

RESUMO

Em resposta ao mercado globalizado, diversos esforços têm sido realizados para o desenvolvimento de soluções buscando a melhoria da produtividade no desenvolvimento de software. Atualmente ser competitivo é um grande desafio. Entre os ingredientes para alcançar esse objetivo temos a maximização do uso de recursos, eliminação das ineficiências e melhoria de produtividade.

Atualmente, existem várias estratégias para melhoria de produtividade que são objeto de pesquisa no campo da engenharia de software. Alguns estudos endereçam apenas um fator e estratégias para minimizar seu efeito ou maximizar seus resultados. Ou ainda há os que apresentam as armadilhas a serem evitadas na adoção e maximização dos resultados de determinado fator. Mesmo com inúmeros estudos sobre o assunto, organizações ainda não sabem quais os fatores mais relevantes para sua produtividade e o que fazer com eles.

Para encurtar o *time-to-market* e para melhorar a produtividade, cada organização de software deve olhar e focar nos seus próprios fatores e fraquezas, e decidir quais estratégias podem fornecer o maior ganho. Alguns estudos defendem que uma ação de melhoria isolada não trará grandes ganhos de produtividade. Por isso é preciso executar várias ações para ter sucesso, ou seja, adotar uma iniciativa integrada em diversas áreas, tais como: ferramentas, métodos, ambiente de trabalho, gerenciamento, incentivos pessoais, reuso de software, entre outros fatores.

Para que seja possível escolher o conjunto adequado de estratégias para alcançar melhoria na produtividade de software, este trabalho levantou através de uma revisão na literatura uma amostra de estratégias para melhoria de produtividade que foram publicadas ao longo dos anos, e as apresenta de forma consolidada. Além da revisão da literatura em estratégias de melhoria da produtividade, este trabalho apresenta uma proposta de meta estratégia que permite para cada organização a definição e manutenção de seus programas de melhoria contínua de produtividade. A meta estratégia leva em consideração os principais fatores que afetam na produtividade da organização e apóia na definição e na escolha das estratégias para maximizar os fatores de influência positiva e minimizar ou evitar os fatores de influência negativa. Um experimento exploratório foi realizado em uma organização de software do Porto Digital em Pernambuco.

Palavras-chave: Produtividade, melhoria de produtividade, estratégias de produtividade e fatores de produtividade.

ABSTRACT

In response to markets globalization, diverse efforts have been made for developing solutions to improve software productivity. Nowadays being competitive is a great challenge. Among the ingredients that combined can help us to achieve this goal, we have concerns like maximizing the resources, eliminating inefficiencies and productivity improvement.

There are several strategies for improving productivity that are subject of research in the software development field. Some studies addresses just one factor and its strategies, or there are those which presents the pitfalls to be avoided in the adoption of the strategies and there are others that helps maximizing the results of a particular factor. Despite numerous studies on this subject, organizations do not yet know which factors are most relevant to their productivity and what to do with them.

To be able to shorter the time-to-market and to increase productivity, each software organization has to look to their own factors and their own weakness to decide which strategies are the ones that can get them the most gain.

Some researches defend that just one isolated improvement action do not bring great productivity gains. Therefore we must execute several actions to succeed, as an integrated initiative in several areas, for example, improved tools, methods, work environment, management, personal incentives, software reuse, among others. In other to be able to choose the right set of strategies to reach software development productivity improvement, this work puts together strategies that have been used and published along the years.

Through a literature review, this work presents a consolidated view of the strategies to address the main factors that have affected productivity over the years. The premise used here to classify the factors as the most relevant were the number of citation. This research aims to support software development industry on the definition of their strategies to improve productivity. Besides the literature review on software development productivity strategies, we propose a Meta Strategy to help software development organizations to establish and maintain their own productivity program and specific strategies to reach productivity improvement.

An experiment done at an software organization at Porto Digital, in Recife is presented, along with the motivation for each strategy chosen.

Keywords: Productivity, productivity improvement, productivity strategies and productivity factors.

SUMÁRIO

Capítulo 1—Introdução	1
1.1 Motivação	1
1.2 Contexto	4
1.3 Trabalhos Relacionados	6
1.3.1 PDCA	7
1.3.2 DMAIC	9
1.3.3 IDEAL	10
1.4 Problema	11
1.5 Objetivos	12
1.5.1 Objetivo Geral	12
1.5.2 Objetivo Específicos	12
1.6 Metodologia	13
1.6.1 Termos Chave da Pesquisa	13
1.6.2 Engenhos de Busca	14
1.6.3 Definição de critérios de seleção de fontes	14
1.6.4 Definição de critérios para escolha dos fatores mais relevantes	15
1.7 Organização da Dissertação	15
Capítulo 2—Conceitos Fundamentais	17
2.1 Produtividade	17
2.2 Tendência de melhoria de produtividade	19
2.3 Estratégia	20
2.4 Sistemas de Medição	21
2.5 Disfunção em Sistemas de Medição	23
2.6 Considerações Finais	24
Capítulo 3—Estado da Arte	25
3.1 Introdução	25
3.2 Contribuições dos anos setenta (1970 - 1979)	25
3.3 Contribuições dos anos oitenta (1980 - 1989)	27
3.4 Contribuições dos anos noventa (1990 - 1999)	34
3.5 Últimas Contribuições (2000-2009)	39
3.5.1 A coletânea de estratégias oriundas das metodologias ágeis	41
3.5.2 Concluindo as últimas contribuições	45
3.6 Classificação dos Fatores e Estratégias	52

3.6.1	Classificação Adotada	53
3.6.2	Seleção e Qualificação dos Fatores	57
3.7	Considerações Finais	57
Capítulo 4—Catalogação de Fatores e Estratégias		60
4.1	Introdução	60
4.2	Abordagem da Documentação Adotada	62
4.3	Catálogo de Estratégias	63
4.3.1	Fatores e Estratégias Relacionados ao Produto	63
4.3.1.1	Fator: Complexidade de Software	63
4.3.1.2	Fator: Reuso	66
4.3.1.3	Fator: Tamanho do Produto	67
4.3.2	Fatores e Estratégias Relacionados ao Projeto	68
4.3.2.1	Fator: Sistemas de Incentivos e Recompensas	68
4.3.2.2	Fator: Comunicação	70
4.3.2.3	Fator: Restrição do Cronograma	71
4.3.2.4	Fator: Instabilidade dos Requisitos	72
4.3.2.5	Fator: Linguagem de Programação	73
4.3.2.6	Fator: Processo	74
4.3.2.7	Fator: Tamanho da Equipe	75
4.3.2.8	Fator: Ferramentas	76
4.3.3	Fatores e Estratégias Relacionados a Pessoas	77
4.3.3.1	Fator: Experiência e Habilidade da Equipe	77
4.3.3.2	Fator: Motivação	79
4.3.3.3	Fator: Participação e Experiência do Cliente	80
4.3.3.4	Fator: Qualidade no Gerenciamento	81
4.4	Considerações finais	83
Capítulo 5—Meta-Estratégia		84
5.1	Introdução	84
5.2	Estabelecer Patrocínio	87
5.3	Definir responsáveis	87
5.4	Iniciar ou Renovar Programa de Melhoria da Produtividade	88
5.5	Conhecer o Momento da Organização	89
5.6	Identificar os vilões da produtividade mais relevantes	92
5.7	Analisar Catálogo de Estratégias	95
5.8	Planejar Ciclo de Melhoria	96
5.9	Estabelecer política de produtividade da organização	96
5.10	Executar e Acompanhar o programa de melhoria da produtividade	98
5.11	Considerações finais	99

Capítulo 6—Estudo de Caso	100
6.1 Introdução	100
6.2 Aplicando a Meta-Estratégia proposta	100
6.2.1 ETAPA 1: Estabelecer patrocínio	100
6.2.2 ETAPA 2: Definir responsáveis	101
6.2.3 ETAPA 3: Levantar os principais vilões da produtividade	101
6.2.4 ETAPA 4: Definição e priorização das ações que farão parte do plano de ação do projeto	102
6.2.5 ETAPA 5: Planejamento e execução o plano de ação	104
6.2.6 ETAPA 6: Acompanhar o programa de melhoria	106
6.2.7 ETAPA 7: Fim do estudo de caso	107
6.3 Restrições da validação	108
6.4 Outras validações	108
6.4.1 Observações	108
6.4.1.1 Empresa A	108
6.4.1.2 Empresa B	109
6.4.1.3 Depoimentos de pessoas das Empresas B	110
6.5 Considerações Finais	111
Capítulo 7—Conclusão e Trabalhos Futuros	112
7.1 Conclusão	112
7.2 Dificuldades Encontradas e Limitações	113
7.3 Contribuições	114
7.4 Trabalhos Futuros	114

LISTA DE FIGURAS

1.1	Tendência do custo do software [Boehm 87a]	2
1.2	Contexto do estudo [Soares 09a]	5
1.3	Ciclo do PDCA	8
1.4	Método de solução de problemas [Campos 92]	9
1.5	Modelo IDEAL [McFeeley 96]	11
2.1	Limites de produção para uma eficiência técnica [Coelli 05]	18
2.2	Produtividade e eficiência [Coelli 05]	18
2.3	Tendência da produtividade 1987 - 2003(Adaptado de [Premraj 05])	20
2.4	O crescimento anual da produtividade das indústrias de 1988 à 2003. [Groth 04]	21
2.5	Efeito da disfunção [Austin 96]	24
3.1	Timeline das referências mais relevantes	26
3.2	Árvore de oportunidades de melhoria na produtividade de software	30
3.3	Relacionamento linear entre produtividade e taxa de reuso [Basili 96]	36
3.4	Fatores técnicos analisados	54
3.5	Fatores soft analisados (Parte 1)	55
3.6	Fatores soft analisados (Parte 2)	56
4.1	Classificação dos Fatores	64
4.2	Relacionamento de tamanho do projeto com esforço [McConnell 04]	68
4.3	Caminhos de comunicação - adaptado de [McConnell 04]	70
5.1	Meta-Estratégia proposta	86
5.2	Boas Práticas de Medição	90
5.3	Subconjunto de vilões endereçados por estratégias.	95

LISTA DE TABELAS

1.1	Resultado do <i>Chaos Report</i> 2009	3
3.1	Fatores observados na literatura.	58
3.2	Fatores considerados mais relevantes	59
5.1	Etapas da Meta-Estratégia X Passos sugeridos por Boehm [Boehm 81]	85
5.2	Estratégias por vilão: Má Gestão do Projeto	97
5.3	Sugestão de Política de Produtividade	98
6.1	Ranking dos classificados como no. 1	102

LISTA DE ABREVIATURAS E SIGLAS

A seguir são listadas as siglas e acrônimos adotados durante todo o trabalho:

BSC Balanced Scorecard

CMM Capability Maturity Model

CMMI Capability Maturity Model Integration

COTS Componet Of The Shelf

DEMAIC DEfine-Measure-Analyze-Improve-Control

GQM Goal Question Metric

ISO International Organization for Standardization

ISBSG International Software Benchmarking Standards Group

KBSA Knowledge-Based Software Assistants

MPS.BR Melhoria de Processo de Software Brasileiro

PBQP Programa Brasileiro de Qualidade e Produtividade em Software

PDCA Plan - Do - Check - Act

PMBOK Project Management Body Of Knowledge

PROSE Productivity in Software Engineering

P&D Pesquisa e Desenvolvimento

CAPÍTULO 1

INTRODUÇÃO

Neste capítulo é apresentada a motivação deste trabalho, juntamente com o contexto de pesquisa em que ele está inserido. Em seguida, uma descrição da metodologia utilizada na pesquisa é apresentada. Finalmente, é feita uma breve apresentação da proposta do trabalho, abordando o escopo e sua contribuição.

....

1.1 MOTIVAÇÃO

A competitividade entre as organizações, motivada pela velocidade da globalização econômica, se tornou a principal preocupação do mercado mundial. Atualmente ser competitivo é um grande desafio, exigindo das empresas melhoria contínua da qualidade de seus produtos e serviços, bem como a redução dos seus custos de produção e realização dos serviços. Entre os ingredientes para vencer esse desafio temos maximização do uso de recursos, eliminação das ineficiências e melhoria de produtividade. Nesse contexto, a produtividade é considerada uma das mais importantes armas para competitividade e lucratividade da indústria [Porter 80].

No mercado globalizado, a crescente demanda por software cada vez mais complexos, e a redução no preço e aumento do desempenho do hardware [Boehm 82] e tem apoiado o crescimento da indústria de software. Segundo Boehm [Boehm 00b], a diminuição do custo do hardware e a evolução da capacidade de processamento, fizeram com que o setor de software crescesse anualmente entre \$300 a \$400 bilhões nos Estados Unidos e entre \$600 e \$800 bilhões no mundo. Entretanto, o custo do software ainda é alto e segue crescendo, como podemos observar na Figura 1.1 que nos mostra a tendência do custo do software apresentado por Boehm [Boehm 87a].

Além dos custos outras pesquisas mostram que os resultados do desenvolvimento de software como um todo são passíveis de melhoria. Segundo os resultados do relatório *The Chaos Report* [GROUP 09] do ano 2000 para os dias atuais tivemos um modesto crescimento no número de projetos de sucesso e um leve aumento no número de projetos

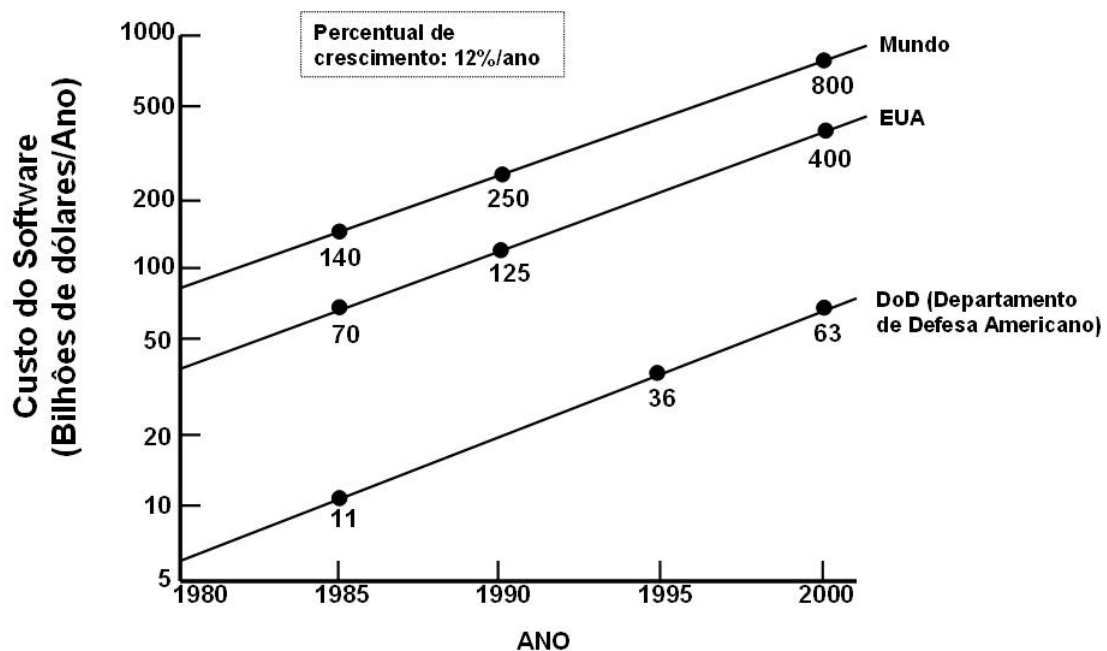


Figura 1.1. Tendência do custo do software [Boehm 87a]

cancelados, como podemos observar nos dados na Tabela 1.1. Se os dados atuais forem comparados com um passado mais recente (2006), os números se mostram ainda mais negativos.

Centenas de pesquisas, *benchmarking*, artigos e livros foram publicados sobre produtividade. Mesmo com todo esse esforço, ainda há uma enorme quantidade de problemas latentes não resolvidos e, tanto a indústria quanto a academia continuam na busca de novas estratégias e novas formas de obter melhorias de produtividade no desenvolvimento de software. Como afirmado pelos autores Wagner e Ruhe, “produtividade ainda é um problema para o desenvolvimento de software atual, e nem todos os fatores e seus relacionamentos são conhecidos”[Wagner 08].

Em busca de um diferencial competitivo, tentando acompanhar a demanda do mercado, e ainda fazer parte do percentual de projetos de sucesso, as empresas de desenvolvimento de software têm investido muito em melhoria de qualidade e produtividade.

A verdade é que não existe uma solução completa ou ideal que possa ser aplicada para resolver todos os problemas relacionados com o desenvolvimento de software [Boehm 87a]. Se durante os anos 90, a melhoria da qualidade constituiu um dos principais desafios para a produção de software, atualmente este desafio se chama produtividade. A vasta literatura sobre produtividade confirma o interesse da sociedade científica mundial neste desafio.

Tabela 1.1. Resultado do *Chaos Report* 2009

	2000	2002	2006	2009
Projetos de Sucesso: entregue no prazo, dentro do orçamento e com os requisitos acordados.	28%	34%	35 %	32%
Projetos Desafiados: entregue atrasado, além do orçamento planejado e/ou com menos requisitos do que o planejado.	49%	51%	46%	44%
Projetos Falhos: cancelados ou entregues e nunca utilizados.	24%	15%	19%	23%

Para uma empresa que busca qualidade, existe uma série de modelos como referência e guia, como é o caso do modelo CMMI (Capability Maturity Model Integration) [Institute 06] e o MPS-BR (Melhoria de Processo de Software Brasileiro) [SOFTEX 07]. O CMMI é um dos modelos de qualidade mais aceito e reconhecido mundialmente, e tem sido uma grande referência para várias organizações rumo à melhoria de processo e melhoria organizacional como um todo. No Brasil a busca por certificações em modelos de qualidade tem crescido muito. Segundo dados do PBQP Software 2006 (Programa Brasileiro de Qualidade e Produtividade em Software) [MEC 06] em 2001 existia apenas 6(seis) empresas certificadas no modelo CMMI e em 2005 já eram 49 (quarenta e nove). No modelo de referência MPS-BR [SOFTEX 07], que foi criado em Dezembro de 2003, existiam um total de 72 empresas que adotaram o modelo até 2007, e 205 até Fevereiro de 2010¹, com previsão de abranger 300 empresas certificadas até 2011.

Mesmo com toda essa crescente busca por qualidade e produtividade, não se sabe qual é o percentual exato de sucesso e falha dos projetos com foco em melhoria de processo de software. Debou e Kuntzmann [Debou 00] relatam que apenas um terço desses projetos obtêm sucesso em melhorar o desempenho da organização de forma significativa. Pesquisa realizada por Santana [Santana 07] observou organizações do pólo tecnológico de Pernambuco, Brasil, e os seus resultados sugerem que na indústria local, o percentual de insucesso é ainda maior que o observado por Debou e Kuntzman.

O estudo de Jiang, Naudé e Comstock [Jiang 07b] mostra que devido à crescente complexidade e magnitude dos projetos, a produtividade no desenvolvimento de software não tem crescido de forma consistente. Segundo os autores, a produtividade tem passado por uma variação irregular - de declínio e ascensão - e não existe nenhum sinal eminente de melhoria.

¹<http://www.softex.br/portal/softexweb/> acessado em 9 de Abril de 2010

Nas revistas e periódicos da administração nos deparamos com colocações contundentes como “*O objetivo de toda e qualquer empresa é alcançar a lucratividade máxima*” [Porter 04]. Ainda é mencionado que entre os ingredientes que compõem a receita para se atingir essa meta estão questões como maximização de recursos, eliminação de ineficiências e melhoria de produtividade. Segundo Porter, o Brasil chegou ao estágio em que todos precisam de uma estratégia [Porter 04]. Ele se refere a um momento de pensar em longo prazo. Com isso, algumas questões são inevitáveis e latentes: *Qual é a estratégia? E quais são os fatores que devem ser levados em consideração em sua elaboração e implementação?* Esses são questionamentos muito amplos que precisariam ser estudados com foco em cada indústria. Para empresas de desenvolvimento de software, uma estratégia para alcançar seu diferencial competitivo é através de uma melhoria de produtividade.

É inegável que muitos avanços já foram feitos quanto à melhoria da produtividade no desenvolvimento de software com a pesquisa, definição, padronização e divulgação de boas práticas. O problema é que esses estudos exploram as práticas ou técnicas de uma maneira muito pontual e ainda muito longe da realidade das pequenas e médias empresas brasileiras. Como citado por MacCormack *et al.* [MacCormack 03], o problema nunca foi a falta de “balas de prata”, mas sim o “como escolher” dentre o arsenal extenso de coloridas balas de prata.

1.2 CONTEXTO

As pesquisas que deram origem a este trabalho foram realizadas em conjunto com o grupo de pesquisa ProSE (*Productivity in Software Engineering*), que tem como objetivo investigar e contribuir com o estado da arte no âmbito da produtividade e métricas em organizações que desenvolvem software. Além disso, propor soluções para tratar produtividade como algo planejado, controlado e executado.

Dentro do grupo existem três linhas de pesquisa: análise de produtividade (métricas de produtividade e fatores que afetam a produtividade); técnicas, processos, ferramentas e ambientes para a melhoria de produtividade; e estimativa e medição de software.

Com base nas pesquisas e estudos analisados pelo ProSE², a Figura 1.2 apresenta uma visão geral de como o problema da produtividade foi mapeado através de um *framework* que contém um conjunto de soluções para que as organizações desempenhem um programa efetivo de produtividade. Ele é composto pelas seguintes partes:

- Infra-estrutura para programas de medição de produtividade: definição de recursos

²<http://prose.cesar.org.br> e <http://www.mct.gov.br/index.php/content/view/2867.html>

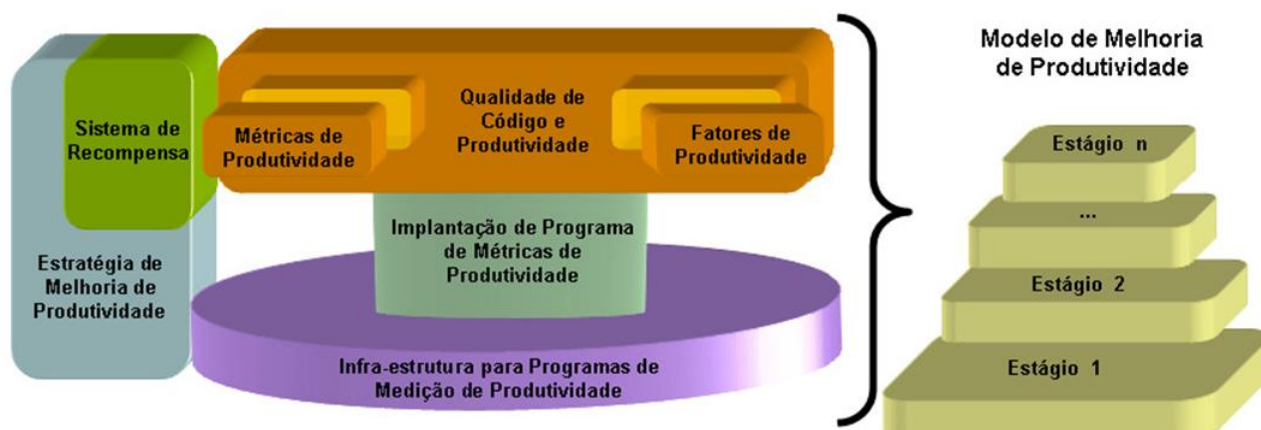


Figura 1.2. Contexto do estudo [Soares 09a]

(ferramentas, papéis e responsabilidades, hardware etc.) que possibilite suporte à implantação de um programa de métricas de produtividade;

- Implantação de programa de métricas de produtividade: definição de um modelo baseado em métricas que possibilite a avaliação de produtividade em diferentes projetos;
- Métricas de produtividade: definição de métricas que avaliem a produtividade dos projetos de desenvolvimento de software;
- Qualidade de código e produtividade: análise da influência da qualidade do código de software na produtividade do time, através de um exame das métricas de código que influenciam na qualidade da arquitetura e métricas de código que influenciam na produtividade;
- Fatores de produtividade: elaboração de um catálogo completo e consistente contendo informações sobre os principais fatores que influenciam na produtividade. Espera-se que este catálogo possa servir como guia para as organizações que desejam iniciar programas de melhoria de produtividade em projetos de desenvolvimento de software. Para as empresas que já medem a produtividade, mas desconhecem a importância desses fatores, o catálogo poderá ajudar na decisão de quais outras dimensões devem ser medidas;
- Sistemas de Recompensa: sistemas de recompensa como estratégia de melhoria de produtividade em organizações de software [Soares 09a];

- Estratégias de melhoria de produtividade: propiciar uma visão sistêmica sobre as práticas relacionadas à produtividade no desenvolvimento de software, permitindo uma documentação da correlação entre as soluções.

Baseado nesse contexto, este trabalho procura contribuir com estratégias para melhoria de produtividade. Propondo uma infra-estrutura para definição de estratégias de melhoria de produtividade em organizações de software, com apoio de estratégias identificadas e catalogadas durante a revisão bibliográfica.

1.3 TRABALHOS RELACIONADOS

Alguns trabalhos se assemelham pela pesquisa em profundidade realizada nos fatores que influenciam na produtividade como o de Maxwell *et al.* [Maxwell 96] e o de Wagner e Ruhe [Wagner 08]. Ambos coontemplam apenas uma visão dos fatores, sem abordar como lidar com eles, quais as estratégias a serem adotadas para maximizar um fator de influência positiva e como evitar a ocorrência de fatores de influência negativa. Embora foquem apenas nos fatores, os dois trabalhos inspiraram uma etapa da metodologia adotada nesta pesquisa, o levantamento dos fatores de influência na produtividade.

Em 1981, Boehm [Boehm 81] em seu trabalho sobre a economia do Software, agrupou um conjunto de estratégias para alguns dos fatores catalogados por ele. No entanto, esse trabalho já possui quase 30 anos e um grande número de artigos surgiu com novas estratégias desde então, além de que alguns fatores de três décadas atrás já não se apresentam como relevantes na atualidade. Esse trabalho foi inspirador tanto para o levantamento das estratégias na melhoria de produtividade quanto para a formatação da Meta-Estratégia proposta.

Clinicy [Clinicy 03] apresentou um relevante trabalho propondo quatro áreas principais que impactam na habilidade da organização em aumentar a produtividade de software. Para cada uma dessas áreas sugeriu atributos e melhores abordagens (estratégias). Esta dissertação se inspirou um pouco nessa idéia, preservando o agrupamento por fator, como realizado por Boehm [Boehm 81].

Para a identificação das estratégias de melhoria de produtividade, foi levantada e analisada uma quantidade maior de fatores que influenciam na produtividade do que a abordada por Clinicy. O mesmo foi feito com as estratégias que, em seguida, foram agrupadas por fator de influência no intuito de apoiar as empresas a maximizar o impacto positivo do fator ou ainda evitar o impacto negativo, com apoio das estratégias. Vale ressaltar que apenas estratégias para os fatores com maior número de citações na literatura

foram levantadas, analisadas e mapeadas.

Wagner e Ruhe [Ruhe 08] ainda possuem um segundo trabalho que compartilham, em parte, o mesmo objetivo deste trabalho. Os autores apresentam um modelo que permite focar apenas nas alavancas específicas que aperfeiçoarão o trabalho de cada organização. Algumas etapas são semelhantes às da Meta-Estratégia desenvolvida neste trabalho, já que a preocupação de “Adaptar a realidade de cada organização” e ainda a priorização das alavancas (estratégias) mais relevantes são preocupações comuns. A Meta-Estratégia proposta nesta dissertação se diferencia, principalmente, pelo uso das estratégias catalogadas durante revisão bibliográfica e ainda pela visão de programa de melhoria contínua.

Entre os trabalhos relacionados temos também os que inspiraram o conceito de melhoria contínua. A Meta-Estratégia proposta tomou como base alguns outros modelos e métodos já conceituados de melhoria contínua de qualidade e desempenho organizacional, tais como PDCA [Deming 86], DMAIC [McCarty 05] e IDEAL [McFeeley 96].

Esses modelos são descritos a seguir.

1.3.1 PDCA

O PDCA (*plan-do-check-action*: planejar, executar, verificar, agir) que foi originalmente desenvolvido na década de trinta, pelo estatístico Shewhart e amplamente divulgado por Deming [Deming 86], servindo como base para muitos modelos de melhoria. Segundo Campos [Campos 97] ele é aplicado para se atingir resultados dentro de um sistema de gestão e pode ser utilizado em qualquer empresa de forma a garantir o sucesso nos negócios, independentemente da área de atuação da empresa. “É um caminho para se atingir a meta” [Campos 97]. O PDCA é uma ferramenta de gestão, um modelo cíclico de melhoria contínua de processos, produtos e serviços, e resolução de problemas. A Figura 1.3 mostra o PDCA, e seus respectivos passos:

- **Planejar:** estabelecer uma meta ou identificar o problema [Campos 97]; analisar as informações relacionadas ao problema e suas causas raízes e elaborar um plano de ação para entregar resultados de acordo com os requisitos e objetivos estabelecidos;
- **Executar:** implementar o que foi planejado e definido no planejamento;
- **Check** (verificar): monitorar e avaliar periodicamente os resultados, confrontando-os com o planejado [Campos 97] e identificando a eficácia das ações implementadas e problemas raízes;

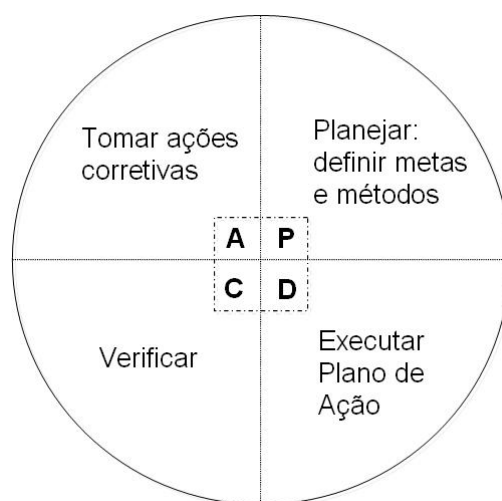


Figura 1.3. Ciclo do PDCA

- **Agir:** tomar ações para endereçar as causas raízes dos problemas identificados no passo anterior.

Os quatro passos do PDCA, e seu princípio cíclico em busca de melhoria contínua, são utilizados pela Meta-Estratégia proposta neste trabalho, complementada ainda com especificidades para área de melhoria de produtividade. O planejamento é feito baseado na etapa de identificação dos vilões³ organizacionais e na decisão de quais estratégias melhor se adequam a realidade da organização. Em seguida, as estratégias são executadas com o objetivo de endereçar os principais vilões. Nesta sequência é realizado também um monitoramento periódico, assim como ocorre no PDCA, onde algumas medidas e análises são realizadas para verificar a efetividade das estratégias de forma cíclica. Desta maneira garante-se melhoria contínua da produtividade.

Esta visão progressiva foi comentada por Falconi [Campos 92] como método de solução de problema, ou ainda como conceito de melhoramento contínuo baseado na conjugação dos ciclos PDCA de manutenção e melhorias, conforme mostrado na Figura 1.4. Para garantir a melhoria contínua, os conceitos de patrocínio e responsáveis foram acrescentados à Meta-Estratégia.

³Fatores de influência negativa na produtividade da organização.

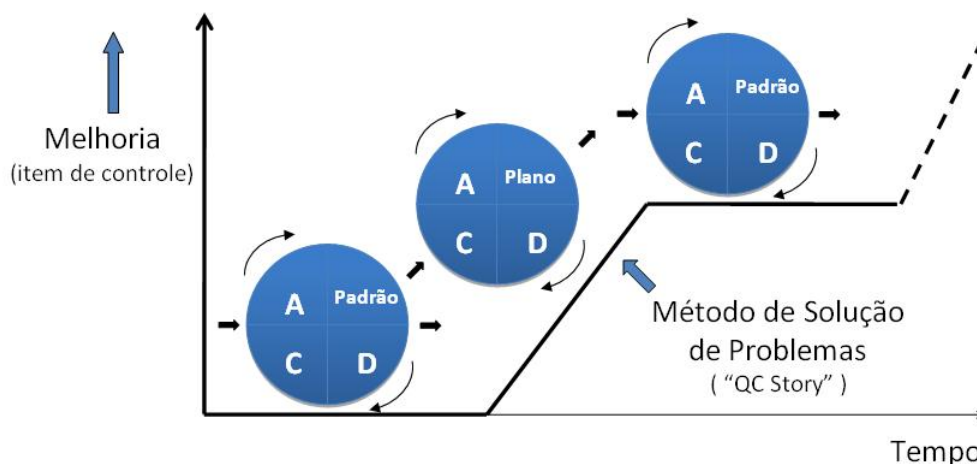


Figura 1.4. Método de solução de problemas [Campos 92]

1.3.2 DMAIC

Six Sigma é uma metodologia que visa ajudar às empresas a reduzirem seus custos com falhas de processo, e busca elevar a satisfação dos clientes e o lucro operacional, desenvolvida pela Motorola na década de oitenta [McCarty 05]. O programa foi elaborado com o severo desafio do “desempenho livre de defeitos”.

O DMAIC (*DEfine-Measure-Analyze-Improve-Control*: Definir, Medir, Analisar, Melhorar e Controlar) é o processo de melhoria contínua usado na metodologia Seis Sigma, que visa a excelência operacional [McCarty 05]. Ele se refere a uma estratégia de qualidade baseada em dados para melhorar processos. As fases do DMAIC são [Siviy 07]:

- **Definir:** Momento de identificação e escolha de quais processos devem ser melhorados para atender a uma característica crítica para o cliente, aumentando a sua satisfação;
- **Medir:** Realizar um levantamento geral de todas as entradas do processo e como se relacionam com as características críticas do cliente. A habilidade do processo em produzir *outputs* não defeituosos deve ser medida;
- **Analisar:** Procurar pelas fontes de variação que aumentam a variabilidade do processo e que são responsáveis pela geração de defeitos;
- **Melhorar:** Melhorar o processo através de soluções criativas para corrigir ou prevenir os problemas. Nesta etapa, um plano de implementação é desenvolvido e

realizado;

- **Controlar:** Etapa para controlar as melhorias e manter o processo em seu novo curso. Responsabilidades com o monitoramento do processo são atribuídas, além da confirmação dos benefícios alcançados.

As etapas do modelo são contempladas pela Meta-Estratégia, e seus conceitos poderiam facilmente ser estendidos para de melhoria de produtividade. Embora, a Meta-Estratégia agregou além dessas etapas pontos do Modelo IDEAL [McFeeley 96], do programa de melhoria de [Boehm 81] e o catálogo de estratégias.

1.3.3 IDEAL

O Modelo IDEAL (Initiating-Diagnosing-Establishing-Acting-Learning: Iniciar, Diagnosticar, Estabelecer planejamento, Agir e Aprender) é uma abordagem de melhoria contínua que identifica as tarefas e passos necessários para estabelecer uma iniciativa melhoria de processo.

O modelo fornece uma abordagem de engenharia disciplinada para a melhoria, que foca no gerenciamento do programa de melhoria e fornece a fundação para uma estratégia de melhoria de longo prazo [McFeeley 96]. O modelo é composto por cinco fases, cada uma delas composta por atividades, como podem ser observadas na Figura 1.5.

As fases do modelo são:

- **Iniciar:** Estabelecer a infra-estrutura inicial - um alicerce - para a melhoria a ser estabelecida, e definir os papéis e as responsabilidades iniciais;
- **Diagnosticar:** Entender o trabalho de melhoria a ser realizado, definindo o estado atual da organização e ainda o estado futuro desejado;
- **Estabelecer planejamento:** Desenvolver um plano de ação com detalhes de como a organização alcançará o objetivo pretendido. Priorizar quais práticas organizacionais podem ser melhoradas e ajustadas baseando-se no resultado da etapa de Diagnóstico;
- **Agir:** Implementar o plano de trabalho de acordo com o planejado, criando melhores soluções que atendam às necessidades organizacionais identificadas. As soluções são testadas com projetos-piloto e são modificadas quando necessário para refletir o conhecimento, experiências e lições obtidas no projeto piloto;

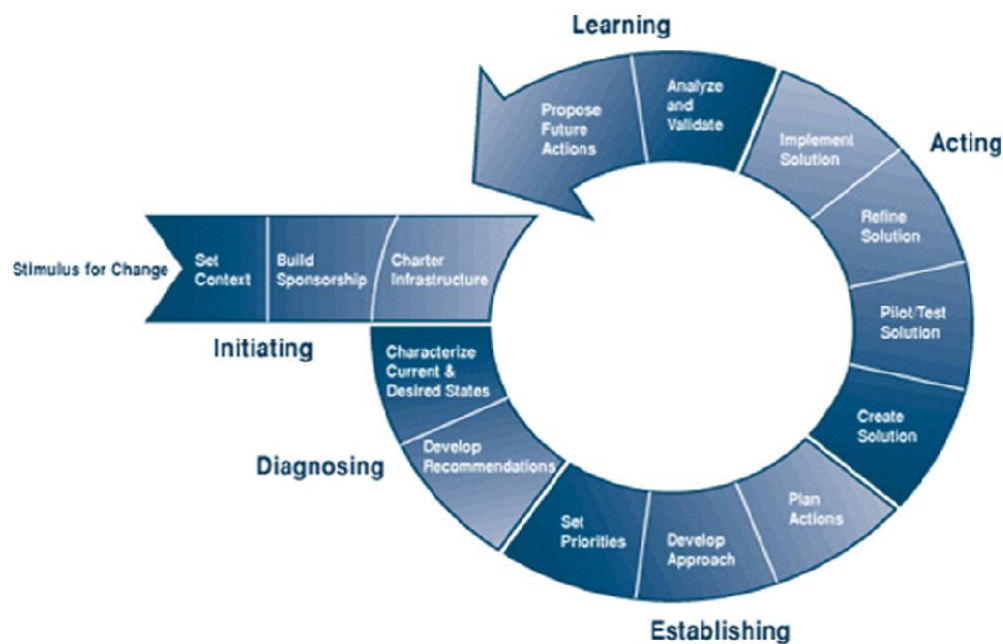


Figura 1.5. Modelo IDEAL [McFeeley 96]

- **Aprender:** Aprender da experiência e melhorar a habilidade da organização de adotar novas tecnologias no futuro. A experiência obtida com execução do modelo IDEAL é revista para determinar se os objetivos foram atingidos. A coleta, análise e documentação das lições aprendidas são realizadas.

Uma das mais importantes características do modelo IDEAL é a possibilidade de implementar ciclos contínuos de melhoria. Este ciclo inicia-se com um escopo reduzido e aumenta à medida que a organização obtém maturidade.

O modelo IDEAL foi a principal referência para a Meta-Estratégia juntamente com o programa de melhoria sugerido por Boehm [Boehm 81]. A idéia de criar a Meta-estratégia é trazer um diferencial de focar na produtividade através da análise dos vilões, da política estabelecida e do uso do catálogo de estratégias.

1.4 PROBLEMA

A partir do cenário descrito na seção 1.1 e do contexto onde esta inserida esta pesquisa, descrito na seção 1.2, define-se o problema deste trabalho da seguinte forma:

- Como alinhar e garantir o apoio da alta direção para um programa de melhoria continua da produtividade de forma a garantir um aumento de produtividade dos times de desenvolvimento de software?

- Uma Meta-Estratégia para definição e manutenção de programas de melhoria contínua de produtividade, específicos para cada organização, é eficaz no aumento da produtividade de uma empresa de software?
- Quais são as estratégias sugeridas pela literatura que devem ser consideradas para maximizar os fatores de influência positiva na produtividade e evitar a influência negativa?

1.5 OBJETIVOS

Em geral, este estudo compartilha dos objetivos do grupo em que se encontra, com o intuito de investigar e contribuir com o Estado da Arte no âmbito da produtividade em organizações que desenvolvem software, além de propor soluções para tratar a produtividade como algo planejado, controlado e executado.

1.5.1 Objetivo Geral

Propor uma Meta-Estratégia para melhoria de produtividade como um diferencial competitivo para a organização que a adota. A Meta-Estratégia permite estabelecer uma infra-estrutura para programas de melhoria contínua de produtividade, apoiando na escolha e priorização de estratégias que agreguem maior valor a organização.

1.5.2 Objetivo Específicos

Para responder as perguntas apontadas na seção 1.4, este trabalho decompôs o objetivo geral em objetivos específicos. São eles:

- Identificar, dentro da literatura, os principais fatores que influenciam na produtividade de uma organização de software;
- Identificar, para os fatores mais citados, tidos como mais relevantes, estratégias para minimizar efeitos negativos ou maximizar seus efeitos positivos;
- Propor uma Meta-Estratégia que permita o planejamento e execução de programas de melhoria, onde cada programa será específico para a realidade e particularidade da organização que a executa;
- Aplicar a Meta-Estratégia em um estudo de caso, em uma empresa de desenvolvimento de software, para identificar pontos positivos e pontos de melhoria.

1.6 METODOLOGIA

O método de pesquisa adotado para execução deste trabalho foi baseado no método descritivo [Gil 02], onde a observação, registro e análise da área de produtividade no desenvolvimento de software foram realizados num conjunto significativo de artigos e livros da área. Esse método também foi adotado para a validação do trabalho, abrangendo um estudo de caso para investigar aspectos relacionados à Meta-Estratégia proposta, em uma organização em específico.

Neste processo de obtenção de resultados, engenhos de busca foram usados com apoio de *queries* de busca e critérios para seleção e exclusão de fontes foram definidos. Esse processo de pesquisa foi dividido em quatro etapas descritas a seguir. Sendo possível levantar e selecionar em sua execução várias publicações realizadas em congressos, periódicos, relatórios técnicos, livros, entre outras fontes.

- **Etapa 1:** A primeira etapa da pesquisa ocorreu a partir da definição do tema de pesquisa. Foi realizada uma revisão na literatura dos conceitos chave associados ao tema (produtividade, fatores de influência na produtividade e estratégias de melhoria). Cada uma das fontes selecionadas foi mapeada quanto aos fatores de influência na produtividade citados.
- **Etapa 2:** A segunda etapa consistiu em uma busca em profundidade nas fontes tidas como mais relevantes na etapa anterior. Foram analisadas as referências mais recentes e as mais citadas. As novas fontes foram também mapeadas quanto aos fatores de influência na produtividade.
- **Etapa 3:** A terceira etapa de pesquisa, consistiu na releitura das fontes selecionadas para identificação de novas estratégias de melhoria de produtividade.
- **Etapa 4:** Por fim, pesquisas para complementar conceitos básicos e associados foram realizadas em menor intensidade.

1.6.1 Termos Chave da Pesquisa

As pesquisas foram baseadas em palavras chave, e utilizaram *queries* de busca apenas em inglês. As palavras chave que ajudaram a compor as *queries* foram as seguintes:

- Produtividade de Software: Software Productivity, Software Development Productivity, Software Engineering Productivity, Productivity Improvement.

- Estratégias: Productivity Strategies,
- Fatores de Influência na Produtividade: Productivity factors, productivity related factors, factors influencing productivity, Factors Influencing Software.

Além dos resultados retornados pelas *queries*, outros trabalhos foram também considerados. Por exemplo, trabalhos realizados por outros membros do mesmo grupo de pesquisa relacionados na seção de contexto.

1.6.2 Engenhos de Busca

Este material foi levantado a partir de pesquisas realizadas nos seguintes engenhos de busca:

- ACM Digital Library: <http://portal.acm.org/>
- IEEE Xplore: <http://ieeexplore.ieee.org/Xplore/dynhome.jsp>
- Science Direct: www.sciencedirect.com
- Google: www.google.com
- Google acadêmico: <http://scholar.google.com.br>

1.6.3 Definição de critérios de seleção de fontes

A grande maioria dos artigos selecionados foi no idioma inglês, mas alguns poucos livros traduzidos e artigos em português também foram considerados.

Apenas os estudos que apresentavam um dos termos pesquisados nas palavras chave, ou ainda citados em seu *abstract*, foram considerados para uma primeira análise. As fontes pré-selecionadas passaram por uma segunda análise apenas pela leitura do *abstract*. Esta pré-seleção permitiu a exclusão de pelo menos 90% das fontes inicialmente encontradas.

As fontes resultantes dessas duas primeiras análises - por palavras chave e *abstract* - foram lidas e analisadas quanto à sua relevância, para só então serem selecionadas.

Alguns critérios auxiliaram nesse processo de avaliação e seleção ou exclusão. Os critérios utilizados foram os seguintes:

1. Atender aos termos chave de pesquisa. Inicialmente apenas fontes disponíveis na Web, com acesso através de mecanismos de busca e termos chave de pesquisa.

2. Fonte referenciadas pelos demais autores. Em um segundo momento, alguns artigos e livros referenciados por autores conceituados ou muito referenciados pelos demais

autores, que não necessariamente atenderam ao critério de busca, foram analisados quanto à sua relevância para o estudo.

3. Fontes sugeridas pela comunidade. Algumas fontes foram sugestões do grupo de pesquisa para o bom entendimento de fatores que influenciam na produtividade e ainda pela simplicidade da forma de descrição dos fatores e seus exemplos. Surpreendentemente, muitas estratégias surgiram desta forma. Um melhor exemplo é o livro Code Complete [McConnell 04]. Estas fontes foram caracterizadas como “Outras”.

4. Fontes dos conceitos relacionados. Uma pesquisa, em um segundo momento, foi direcionada para os conceitos relacionados citados no Capítulo 2. Embora não tenha se aprofundado tanto nestes quanto nos conceitos fundamentais, o resultado desta fase da pesquisa foi utilizado para formar e consolidar o referencial teórico associado a este trabalho.

1.6.4 Definição de critérios para escolha dos fatores mais relevantes

Dentre as fontes analisadas, foram encontrados aproximadamente trinta fatores de influência na produtividade. Como documentar estratégias para todos os fatores tornaria o trabalho muito extenso, apenas os fatores mais relevantes foram considerados. Os fatores tidos como mais relevantes foram os com mais de quinze citações ou ainda com pelo menos seis citações da última década. O critério de manter como relevante os fatores com mais de seis citações na última década se deu por acreditar que estes podem ser uma tendência. Os fatores mais relevantes e um detalhamento das etapas desta seleção pode ser visto na seção 3.6.2.

1.7 ORGANIZAÇÃO DA DISSERTAÇÃO

Além deste capítulo introdutório, o trabalho é formado pelos seguintes capítulos:

- Capítulo 2 introduz os conceitos fundamentais a serem utilizados na dissertação, tais como: produtividade, estratégia, sistema de medição, disfunção, modelos de melhoria contínua, juntamente com parte da revisão bibliográfica.
- Capítulo 3 apresenta a revisão bibliográfica das estratégias e fatores que influenciam na produtividade, organizados por década.
- Capítulo 4 apresenta o catalogo de estratégias agrupado por fatores que influenciam na produtividade.

- Capítulo 5 apresenta a Meta-Estratégia para definição de estratégias específicas de melhoria de produtividade para empresas de desenvolvimento de software.
- Capítulo 6 apresenta um estudo de caso no qual a Meta-Estratégia original foi utilizada para colocar em prática as estratégias específicas levantadas no estudo bibliográfico. Este capítulo também apresenta a definição de uma estratégia específica de melhoria de produtividade adotada por uma empresa do Porto Digital em Recife, Pernambuco. Como nem todos os resultados foram conclusivos, dois experimentos exploratórios foram realizadas em duas diferentes empresas de Natal, Rio Grande do Norte.
- Finalmente, o Capítulo 7 conclui o trabalho e sugere possíveis caminhos a serem abordados para trabalhos futuros.

CAPÍTULO 2

CONCEITOS FUNDAMENTAIS

Este capítulo descreve os conceitos fundamentais a serem utilizados na dissertação, para o seu bom entendimento, tais como: produtividade, estratégia, sistema de médio e longo prazo.

....

2.1 PRODUTIVIDADE

Jones [Jones 91] afirma que na economia, temos como produtividade a relação entre a quantidade de bens ou serviços produzidos e a despesa ou trabalho necessário para produzi-los. Em um sistema de manufatura, produtividade é a quantidade de unidades produzidas (*output*) dividida pelo número de horas utilizadas na produção (*input*) [SOMMERVILLE 07], ou seja,

$$\text{Produtividade} = \text{Output} / \text{Input} \quad (2.1)$$

Reifer [Reifer 02] aponta produtividade como a habilidade das organizações de gerarem saídas (sistemas, documentação e etc.) utilizando as entradas ou recursos à sua disposição (pessoas, dinheiro, ferramenta, entre outros). De modo geral, na engenharia de software, produtividade tem sido comumente definida pela razão dos *outputs* produzidos pelos *inputs* consumidos [Walston 77, Yu 91, Boehm 87a, Kitchenham 04]. O que nos leva a entender que a produtividade de software é a razão entre a quantidade de software produzido - o que inclui a documentação - pelas despesas necessárias para produzi-lo, onde o tamanho do produto gerado pode ser medido de diversas maneiras como linhas de código, pontos de função [Albrecht 79], pontos por caso de uso [Karner 93], entre outras.

Coelli et al. [Coelli 05] definem produtividade como uma razão de saídas produzidas pelas entradas utilizadas para a produção. Os autores mostram que quando o processo de produção envolve um único *input* e apenas um único *output*, este cálculo é trivial. Entretanto quando existe mais de um *input*, que é o caso da engenharia de software, um método para agregar as entradas deve ser adotado para obter medidas de produtividade. O mesmo problema se dá para *outputs*.

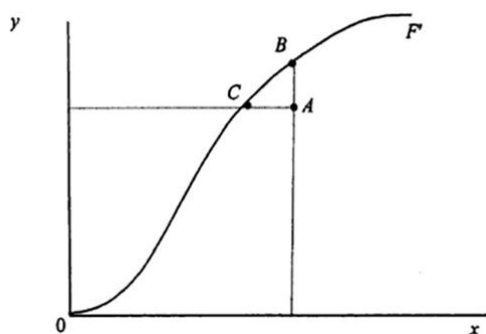


Figura 2.1. Limites de produção para uma eficiência técnica [Coelli 05]

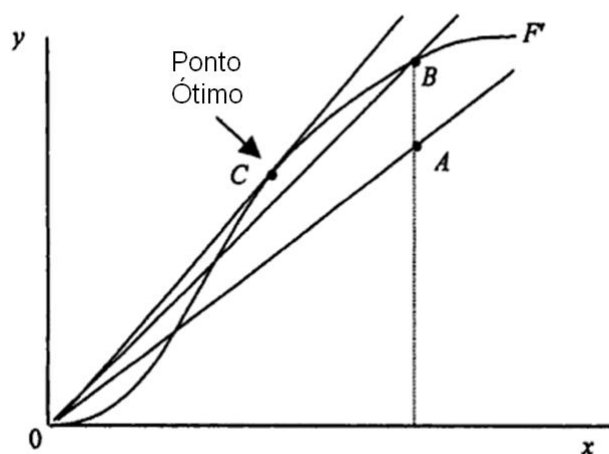


Figura 2.2. Produtividade e eficiência [Coelli 05]

Os autores mostram na Figura 2.1 um sistema de produção simples onde um único *input* (x) é utilizado para produzir um único *output* (y). A linha OF' representa o limite de produção, que representa o máximo de *outputs* possível por cada nível de *input*.

Segundo os autores, as organizações operam no limite se forem eficientes¹, ou abaixo do limite se não forem eficientes. O **ponto A** representa um ponto ineficiente, os **pontos B e C** são pontos eficientes. Uma organização operando no ponto A é ineficiente, porque tecnicamente poderia aumentar o número de *outputs* para o nível associado ao ponto B sem requerer mais *inputs*.

¹Eficiência ou rendimento refere-se à relação entre os resultados obtidos e os recursos empregados

A Figura 2.2 mostra a relação entre produtividade e eficiência. Uma organização pode ser mais produtiva e mais eficiente passando do **ponto A** para o **ponto B**. O **ponto C** é o ponto de maior produtividade possível. Qualquer outro ponto do limite de produção seria de menor produtividade. Os autores concluem assim que podemos ser eficientes e ainda sermos capazes de ser mais produtivos.

Este trabalho se identifica com uma visão mais atual de produtividade, onde o valor produzido (*output*) pode ser analisado em múltiplas dimensões, ou ainda focando no valor agregado resultante. Essas dimensões podem significar resultado financeiro, satisfação do cliente, satisfação da equipe, grau de inovação do produto, experiência adquirida, entre outras, enquanto que o valor produzido (*input*) normalmente é medido pelo esforço ou custo utilizado [Aquino 09b].

A abordagem de ver o *output* por múltiplas dimensões ainda é compartilhada por outros autores. Ruhe e Wagner [Ruhe 08] apresentam produtividade em P&D (Pesquisa e Desenvolvimento) por: $\text{Produtividade} = \text{Valor agregado ao Cliente} / \text{Orçamento efetivo da P\&D}$.

Maxwell [Maxwell 03] nos revela um novo contexto para a produtividade no desenvolvimento de software, onde a produtividade é definida como “Receita gerada por empregado por um produto de software desenvolvido e vendido”.

“O processo chave de uma atividade de desenvolvimento é transformar idéias em produtos. Isso significa que a real produtividade no desenvolvimento de software deveria ser medida por o quão eficientemente e efetivamente nós transformamos idéias em software” [Maxwell 03].

Para complementar a visão de produtividade, na próxima seção analisamos o comportamento da produtividade ao longo dos anos.

2.2 TENDÊNCIA DE MELHORIA DE PRODUTIVIDADE

Uma das evidências mais recentes que temos quanto à modesta melhoria de produtividade nos últimos anos é apresentada por Premraj e Shepperd [Premraj 05], a partir de uma perspectiva recomendada por Kitchenham e Mendes [Kitchenham 04]. O número de projetos analisados para um dado ano é exibido na Figura 2.3.

A Figura 2.3 mostra forte evidência na melhoria de produtividade em projetos de software nos anos 80 e no início dos anos 90, seguido por taxas de melhorias mais modestas, de pouco crescimento. O que nos leva a concluir que mesmo com todo o crescimento do mercado e ainda com todo o esforço dedicado a melhoria de produtividade nas últimas

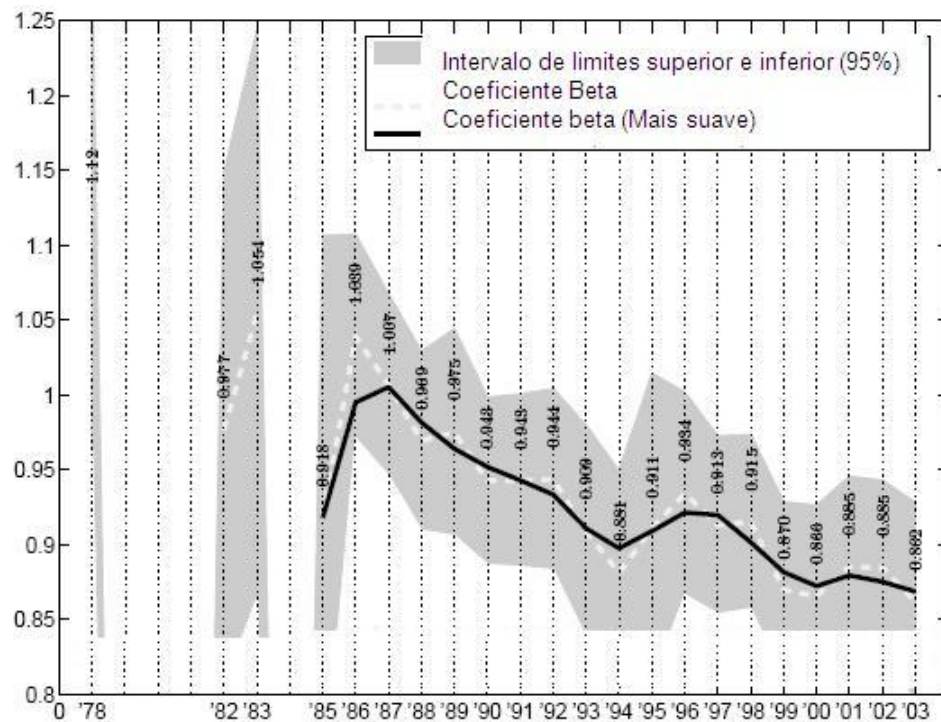


Figura 2.3. Tendência da produtividade 1987 - 2003(Adaptado de [Premraj 05])

décadas, ainda precisamos melhorar os resultados da produtividade na indústria de software. É claro que alguns resultados precisam ser questionados para cada indústria e, no nosso caso, para cada empresa.

Para corroborar com essa visão, Groth [Groth 04] afirma que comparado com outras indústrias, a produtividade na indústria de software tem caído mais do que as demais, conforme exibido na Figura 2.4.

2.3 ESTRATÉGIA

Há diversas abordagens quanto ao conceito de estratégia. Vamos analisar a visão segundo Mintzberg et al [Mintzberg 98], que defende que a estratégia possui cinco definições diferentes:

- Plano: plano de ação para atingir objetivos determinados ou simplesmente caminho para o futuro;
- Padrão: um comportamento permanente ou de longo prazo de uma empresa;
- Posição: localização de determinado serviço/produto em determinada área do mer-

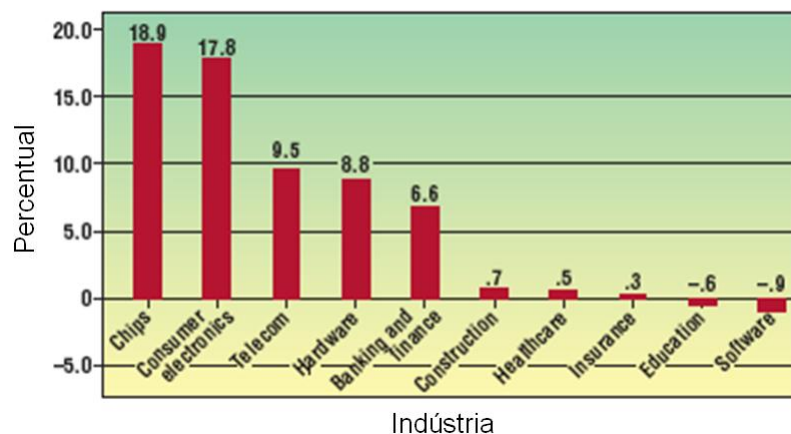


Figura 2.4. O crescimento anual da produtividade das indústrias de 1988 à 2003. [Groth 04]

cado;

- Perspectiva: maneira própria de fazer as coisas dentro de uma organização, forma de conduzir o seu negócio;
- Truque: uma ação específica para burlar a concorrência.

A estratégia também é a **Posição** detida pelo órgão em executar um conjunto de atividades inovadoras que o mantém na liderança, obtendo resultados satisfatórios na comunidade [Mintzberg 98]. Ela também é **Perspectiva** quando desenvolve uma maneira própria de fazer as coisas, dando-lhe reconhecimento no ambiente da comunidade. E, por último, temos estratégia como **Truque**, quando é usada como manobra para ludibriar um oponente ou concorrente, fazendo com que ele pense equivocadamente sobre as intenções reais da administração.

Estratégia na visão deste trabalho é um **Plano** e uma **Perspectiva**. Este trabalho propõe uma estratégia para melhoria de produtividade como um diferencial competitivo para a organização que a adota. Ela também se baseia em um plano de ação formal que precisará ser medido e controlado, onde cada empresa terá sua própria forma de execução, sua própria perspectiva.

2.4 SISTEMAS DE MEDIÇÃO

Para melhor entender o conceito de Sistemas de Medição, é preciso compreender alguns conceitos relacionados, tais como: medição e métrica.

Medição: Medição é o processo pelo qual números ou símbolos são atribuídos a entidades² do mundo real, de forma a tornar possível caracterizar cada entidade por meio de regras claramente definidas [Fenton 97]. A medição tem como principal foco apoiar a tomada de decisão em relação aos projetos, processos e atendimento aos objetivos organizacionais [SOFTEX 07].

Métrica: Uma métrica é o valor da medição de um atributo³ de uma determinada entidade. Segundo Fenton [Fenton 97] são informações sobre os atributos de entidades. Um exemplo de métrica para o atributo Tamanho, da entidade Produto, é “números de linhas de código” obtido através da contagem do número de linhas de código do produto.

Segundo Humphrey [Humphrey 90], são quatro os principais papéis de Medições de Software:

- **Entender:** Métricas ajudam a entender o comportamento e funcionamento de processos, produtos e serviços de software;
- **Avaliar:** Métricas podem ser utilizadas para tomar decisões e determinar o estabelecimento de padrões, metas e critérios de aceitação;
- **Controlar:** Métricas podem ser utilizadas para controlar processos, produtos e serviços de software;
- **Prever:** Métricas podem ser utilizadas para prever valores de atributos.

Essa necessidade de obter números que quantifiquem o desempenho da organização faz com que os Sistemas de Medição sejam a ferramenta pela qual gestores podem entender que rumo ela está tomando [Aquino 09a]. Drucker [Drucker 01] afirma que para uma organização estar apta a controlar seu próprio desempenho, as medições devem ser simples, claras e racionais. Elas também devem ser dados confiáveis e, mesmo que haja margem de erros, devem ser conhecidas.

Para definir um sistema de medição apropriado é preciso responder às seguintes perguntas [Laird 06]:

- Quem são os clientes das medições?
- Quais são seus objetivos em relação ao produto, processo ou recursos a serem atendidos?

²Entidade são processos, produtos, projetos e recursos.

³Atributos são propriedades ou características de determinadas entidades.

- Quais são as métricas e quando coletar? Onde os resultados das medições deve responder se o objetivo foi ou não atingido.

Estabelecido o sistema de medição, para adotar medições relacionadas à produtividade de software, é preciso responder às seguintes questões [Scacchi 95]:

- Por que medir a produtividade de software?
- Quem deveria medir e coletar os dados de produtividade de software?
- O que deve ser medido?
- Como medir a produtividade de software?
- Como melhorar a produtividade de software?

Scacchi [Scacchi 95] afirma que uma justificativa para medir a produtividade é identificar como reduzir os custos do desenvolvimento de software, melhorar a qualidade do software e melhorar a taxa (*rate*) em que ele é produzido. Dependendo de quem é o cliente da métrica e quais são os objetivos da medição, é definido o que vai ser medido e quem deve coletar os dados, ou ainda o que deve ser automatizado. Segundo o autor as opções do que deve ser medido são muitas e complexas. Uma medição integrada de produtividade ou estratégias de melhoria deve considerar características do produto, processo e seus relacionamentos.

Medir a produtividade pressupõe uma habilidade de construir um programa de medição. É preciso garantir que as medições aplicadas são confiáveis, válidas, precisas e repetíveis [Scacchi 95]. A complexidade e diversidade de tipos de dados sugerem que coletar, organizar e gerenciar os dados da produtividade de software se torna uma atividade que pede um cuidadoso projeto, escolha e desenvolvimento, bem como análise de resultados oferecidos por ferramentas automáticas de análise e suporte a medição [Scacchi 89].

2.5 DISFUNÇÃO EM SISTEMAS DE MEDIÇÃO

Disfunção ocorre quando a forma com que os indivíduos ou equipe trabalha para alcançar um objetivo definido pela organização causa falha no desempenho e esta queda não é refletida nas medições que estão sendo realizadas [Austin 96]. O efeito da disfunção pode ser observado na Figura 2.5.

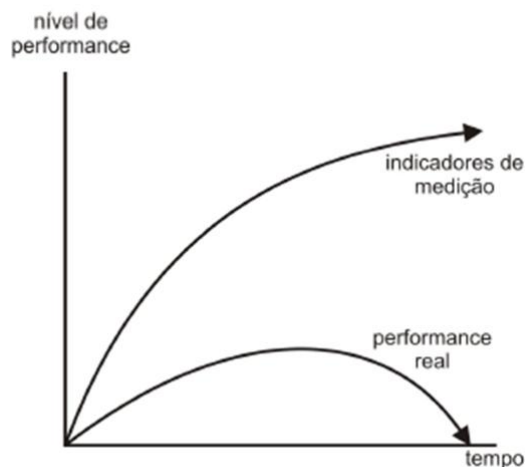


Figura 2.5. Efeito da disfunção [Austin 96]

2.6 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais conceitos que envolvem esta dissertação. Primeiramente, a diversidade dos conceitos diretamente ligados à produtividade na engenharia de software foram citados, juntamente com uma análise de tendência até o ano de 2003. O conceito de estratégia também foi apresentado. Subsequentemente foram explicados os modelos e métodos de melhoria contínua que inspiraram a Meta-Estratégia. Por fim, os conceitos associados à medição foram abordados, sendo um passo muito relevante da Meta-Estratégia proposta.

CAPÍTULO 3

ESTADO DA ARTE

Este capítulo o principal resultado da revisão bibliográfica, mostrando a evolução dos estudos sobre produtividade, bem como os principais trabalhos dentro do contexto de fatores que influenciam na produtividade e as estratégias disponíveis para minimizar seus efeitos ou maximizar os resultados destes fatores, complementando o embasamento de conteúdos futuros que este trabalho irá abordar.

....

3.1 INTRODUÇÃO

Analisar e melhorar a produtividade têm sido uns dos principais objetivos das pesquisas em engenharia de software desde o seu princípio. Ainda assim, muitos estudos sugerem que há um grande potencial para a melhoria. Stensrud e Myrtveit [Stensrud 03] afirmam que este potencial é de 50% em projetos de software. Anselmo e Ledgard [Anselmo 03] afirmam que a produtividade vem caindo mais rapidamente do que em qualquer outra indústria, o que reforça a ideia de que ainda temos muito a melhorar. Os autores Wagner e Ruhe afirmam que a produtividade ainda é um problema para o desenvolvimento de software atual e que nem todos os fatores e seus relacionamentos são conhecidos [Wagner 08].

Para entender parte deste esforço, a Figura 3.1 aponta algumas das referências mais relevantes para este trabalho, associadas ao ano de seu surgimento.

3.2 CONTRIBUIÇÕES DOS ANOS SETENTA (1970 - 1979)

Inicialmente, o objetivo era analisar e catalogar apenas contribuições acadêmicas das últimas três décadas. Entretanto alguns trabalhos foram tantas vezes citados que foi impossível não mencioná-los aqui. Esses estudos, mesmo sendo menos relevantes quanto a estratégias, mapearam alguns fatores que ainda hoje são relevantes quanto à melhoria de produtividade.

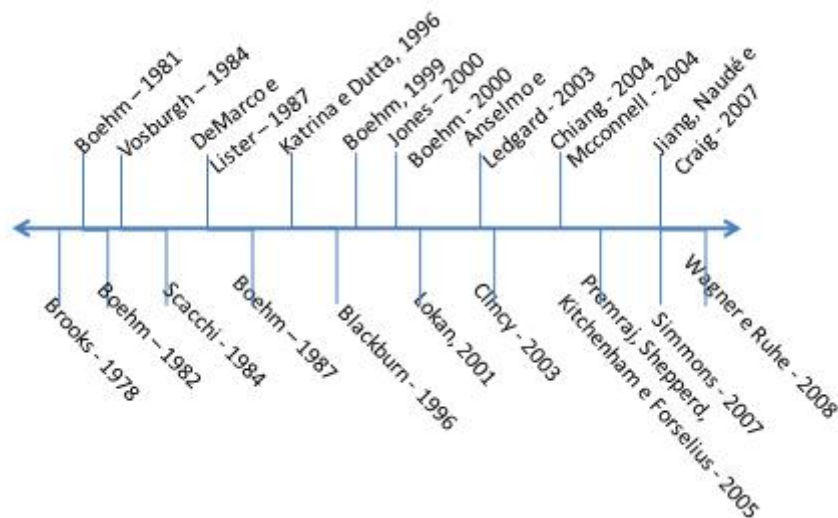


Figura 3.1. Timeline das referências mais relevantes

Um clássico do período é o *Mythical Man-Month* onde Brooks afirma o que até o presente momento não foi refutado: “Adicionar mão de obra a um projeto de software atrasado, o deixa ainda mais atrasado” [Brooks 78].

As premissas citadas no livro de Brooks ainda são aceitas nos dias de hoje, e continuam sendo uma visão importante no momento de traçar estratégias para compor e aumentar os times de desenvolvimento de software. Um pouco mais deste trabalho é novamente descrito na revisão da década de noventa, a edição de aniversário de 20 anos desta obra [Brooks 95].

Os outros três trabalhos mapeados foram [Basili 79], [Albrecht 79] e [Walston 77]. Eles constatarem o impacto de fatores na produtividade do desenvolvimento de software ou ainda medições desta produtividade.

O mais citado foi o trabalho de Walston e Felix, onde utilizando dados de 51 projetos, os autores identificaram vinte e nove fatores com significativa correlação com a produtividade no desenvolvimento de software, em um estudo realizado na IBM. Muitos fatores citados ainda estão presentes em muitos estudos de produtividade, entre eles: complexidade do sistema; participação do usuário e experiência na linguagem de programação. Outros fatores não são considerados como relevantes, nesta pesquisa, devido ao baixo número de citações nos dias atuais, como tamanho e complexidade da base de dados ou desenvolvimento software concorrente com hardware.

3.3 CONTRIBUIÇÕES DOS ANOS OITENTA (1980 - 1989)

Boehm, sem dúvida, foi o autor com maiores contribuições para este trabalho, bem como para a década em questão, sua lista é extensa quando se fala de produtividade e medições.

No livro *Software Economics* [Boehm 81], além de apresentar um modelo de estimativa de custos, o COCOMO (COConstructive COst Model), Boehm propôs um programa de melhoria com a execução de várias ações - ou podemos dizer estratégias - em paralelo para ter um ambiente produtivo e obter ganhos significativos de produtividade. A forma de garantir que o problema produtividade seja atacado por mais de uma perspectiva, o que aumenta as chances de sucesso, é a execução de ações em paralelo. Nesse estudo, Boehm ainda mostra os fatores que impactam no custo do software, favorecendo uma visão de áreas que podem trazer maior retorno em atividades de melhoria de produtividade.

Ainda segundo o autor, o ciclo de vida do produto precisa ser balanceado com o ciclo de vida das pessoas envolvidas em sua produção. Sobre esse ponto, o autor alerta que trabalhos rotineiros como manter um programador eternamente em um projeto de manutenção ou em pedaços insignificantes de código leva a uma produção sem inspiração, faz com que as pessoas percam o interesse profissional na empresa, entre outros pontos.

Além dessa perspectiva, o autor sugere ciclos de desenvolvimento mais curtos, adoção de reuso, geradores de programas, diminuição da complexidade dos programas através de adoção de funções de processamento simples e eliminação de *gold-plating* desnecessários. O trabalho de Boehm, nesse livro, é rico de considerações e idéias quanto a estratégias a traçar em projetos de software. Estratégias estas que mesmo com quase três décadas ainda se mantêm atuais e ainda são citadas por todos os estudiosos dessa área.

O autor [Boehm 81] afirma ainda que embora um programa de melhoria de produtividade para diferentes organizações seja geralmente similar, um programa efetivo requer adaptações para as características únicas de cada organização. Essa é uma premissa também neste trabalho, um ponto muito importante na formatação e adoção da Meta-Estratégia, onde a definição de estratégias específicas, de um programa específico para cada organização é necessária para se atingir de fato os resultados esperados. Por fim o livro apresenta os passos mais significativos para estabelecer e adaptar um programa de melhoria de produtividade.

Ainda segundo o autor [Boehm 81], atributos pessoais e atividades humanas provêm de longe a maior fonte de oportunidade de melhoria de produtividade. Quanto à formação de equipes, Boehm afirma que:

- Equipe melhores, mais capacitadas e experientes, e com menos pessoas tendem a trazer melhores resultados. Augustine (apud Boehm [Boehm 81]) afirma que estudos mostraram que 20% da equipe produziu 50% das saídas.
- Distribua as tarefas pelas habilidades e motivação das pessoas disponíveis.
- A longo prazo as organizações se saem bem com investimento em manter sua equipe atualizada. Só cuidado para não esquecer do que é primordial, atender as necessidades básicas. Como exemplo, não nos preocuparemos em melhorar nossas habilidades em computação se estivermos famintos.
- Selecione pessoas que se complementam e irão se harmonizar entre si, não apenas quanto as habilidades técnicas, mas também quanto aos componentes psicológicos. 11 atacantes não formam um bom time de futebol.
- Não mantenha os que não se enquadram. Mesmo sendo muito cuidadosos, sempre teremos membros do time que inevitavelmente não contribuem nem perto de quanto deveriam, quanto aos objetivos do time. Postergar o tratamento deste problema é catastrófico a longo prazo. O melhor é se livrar dos que não se enquadram o mais rápido possível.

As demais estratégias apontadas por Boehm são documentadas neste trabalho, no capítulo de estratégias.

No primeiro capítulo do livro *Controlling Software Projects* [DeMarco 82], ele fala da “anatomia do fracasso”. DeMarco defende que os projetos fracassam não porque projetaram mal ou programaram lentamente ou por falta de compreensão do problema, nem por falta de motivação da equipe, nem muito menos por falta de conhecimento adequado das tecnologias. O autor culpa as expectativas presunçosas e irracionais. Este trabalho ficou consagrado pela frase “Não pode se controlar o que não se pode medir”. Esta frase foi recentemente comentada pelo autor, onde mais uma vez ele pareceu nos dar sua opinião quanto a expectativas presunçosas.

Hoje todos já compreendemos que métricas de software custam dinheiro e tempo, e devem ser usadas com moderação e cuidado... mas como você administra um projeto sem controlá-lo? Bem, você gerencia as pessoas e controla o tempo e dinheiro. A questão muito mais importante do que “como controlar um projeto de software” é por que diabos estamos fazendo tantos projetos que entregam um valor tão marginal? [Demarco 09]

Em [Boehm 82], de acordo com os resultados do estudo do TRW, Boehm *et al.* reafirmam que ganhos significativos em produtividade requerem um programa integrado de iniciativas em diversas áreas, incluindo melhorias em ferramentas, metodologias, ambiente de trabalho, educação, gestão, incentivos pessoais e reuso de software. Os autores, baseados no experimento realizado apontam que em longo prazo, os maiores ganhos de produtividade virão com o aumento do uso de software já existente.

Behrens afirmou, após estudos em 24 projetos, que o tamanho do projeto, o ambiente de desenvolvimento e a linguagem de programação são fatores determinantes da produtividade, mas não definiu nenhuma estratégia de como lidar com eles [Behrens 83].

Scacchi afirmou que a baixa produtividade é diretamente relacionada com a qualidade do gerenciamento [Scacchi 84]. Essa afirmação é compartilhada por Boehm [Boehm 81].

Vosburgh *et al.* [Vosburgh 84], também seguiu a abordagem da adoção de um programa de melhoria para obter produtividade. Os autores afirmam que para se obter melhoria de produtividade é preciso muito mais do que ações isoladas ou novas tecnologias, para ter sucesso é preciso estabelecer um programa de melhoria da produtividade que deve endereçar os aspectos de produtividade como um todo, onde uma única melhoria pontual não garante grandes ganhos de produtividade. Para Vosburgh e demais autores, um programa de sucesso requer que sejam feitas várias coisas ao mesmo tempo. Os autores ainda separam os fatores que influenciam na produtividade pela habilidade de serem controlados pelo gerente de projeto. Mais uma vez temos a afirmação de que precisamos de muitas ações em paralelo, tentando abraçar o maior número de fragilidades possíveis para que tenhamos bons resultados.

Os quatorze fatores identificados por Vosburgh *et al.* [Vosburgh 84] foram classificados de acordo com a capacidade do gerente de projeto/ organização de controlá-los. Os autores apontam complexidade do software, envolvimento do cliente e tamanho do produto como fatores fora do controle gerencial, enquanto a experiência da equipe e o uso de práticas modernas de programação são fatores citados como dentro do controle do gerente. No ponto de vista dos autores, os fatores relacionados à produto geralmente não estão sob o controle dos gestores, porque os fatores são parte do produto e impõe limitações na produtividade. Por outro lado, fatores relacionados ao projeto são controlados pelos gestores e oferecer oportunidades reais para melhorar a produtividade.

Em [Boehm 87a] Boehm cita mais uma classificação dos fatores que direcionam o custo do desenvolvimento de software: quanto à habilidade de controle do gerente de projeto. O ponto mais relevante para este trabalho é a árvore de oportunidades de melhoria de produtividade. Nela temos um bom conjunto de estratégias a serem adotadas

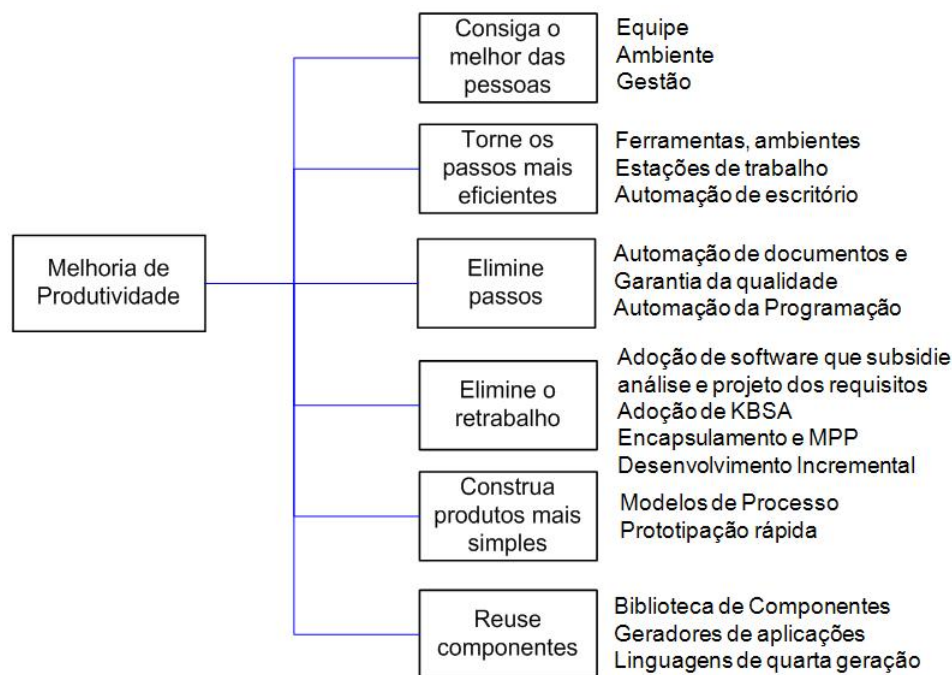


Figura 3.2. Árvore de oportunidades de melhoria na produtividade de software

para melhoria de produtividade, conforme exibido na Figura 3.2.

Em resumo Boehm [Boehm 87a] especifica estratégias de melhoria de produtividade, são elas:

- **Consiga o melhor das pessoas:** através de uma melhor gestão de projetos, melhores incentivos, melhores ambientes de trabalho, e pessoas capacitadas, com boa experiência na linguagem de programação e com a aplicação. “Consiga as melhores pessoas para trabalhar com você e deixe que as pessoas medíocres trabalhem para outro.”
- **Elimine passos e os deixe mais eficientes:** através de automação do processo de programação, trabalhando com ferramentas que transformam especificação de software em código, automação de documentação e etapas do processo.
- **Elimine o retrabalho:** adoção de ferramentas que forneçam subsídio para análise e projeto dos requisitos com melhor visualização de ambiguidades, rastreabilidade, completude e corretude dos requisitos. Uso de KBSA (Knowledge-Based-Software-Assistance: Assistentes de software baseados em conhecimento) e de inteligência artificial para automação de funções de conhecimento apóiam na realização de tarefas mais complexas. Adote MPP (Modernas Práticas de Programação) como inspeções,

verificações e validações iniciais, bem como desenvolvimento incremental. E por fim o que Boehm chamou de “ocultação de informação”, uma espécie de encapsulamento de informações em módulos, de forma a deixá-los auto-contidos.

- **Construa produtos mais simples:** Torne os passos do processo mais eficientes, utilize ambientes integrados de desenvolvimento, trabalhe com prototipação, e automatize o que for possível dentro do processo de desenvolvimento de software - atividades de projeto, testes garantia da qualidade, deixando-o mais eficiente e evitando o *goldplating*.
- **Reuse Componentes:** Mantenha as coisas simples, elimine o retrabalho, desenvolvendo e utilizando bibliotecas de componentes.

Um ano depois, o autor revisitou esse conteúdo, explorando a importância de compreender custos de software e seus impactos, para entender e controlar produtividade. Olhando por este prisma, Boehm e Papaccio [Boehm 88] apontaram como principais influências no custo do software e suas respectivas estratégias para redução, em ordem de relevância:

- O número de instruções de código escolhidas para programar.
Estratégia proposta: Uso de linguagens de quarta geração, uso de componentes para reduzir a quantidade de código fonte gerado, uso de prototipação e outras técnicas de análise de requisitos para garantir que funções desnecessárias não são desenvolvidas.
- A seleção, motivação e gestão de pessoas envolvidas no processo de software.
Estratégia proposta: Manter o melhor grupo de pessoas em sua equipe e mantê-las motivadas.
- Complexidade do produto e tamanho da base de dados.
Estratégia proposta: reduzir complexidade desnecessária (complexidade e tamanho do banco)
- Volatilidade dos requisitos é uma fonte importante e negligenciada de influência nos custos.
Estratégia proposta: Utilizar desenvolvimento incremental.

Em resumo, mais uma vez os autores apontam as seguintes estratégias para redução do custo: **escrever menos código** com o desenvolvimento de produtos mais simples e trabalhando com reuso de componentes de software; **tornar as pessoas mais eficientes** evitando restrições de hardware, melhorando o ambiente de trabalho e promovendo o uso de reuso como, por exemplo, provendo um bônus para pessoas que adotarem reuso ao em vez de recriarem software, eliminando passos manuais e automatizando etapas do processo que for possível; **eliminar retrabalho** através de automação, construção de produtos mais simples com prototipação, e utilização de técnicas de análise de requisitos para garantir que funções desnecessárias não são desenvolvidas, uso de modelos de software melhores e eliminação do *gold-plating* [Boehm 88].

De Marco e Lister [DeMarco 87] mencionam que os principais problemas na engenharia de software não são mais tecnológicos do que sociológicos. Interações humanas são complicadas e nunca claras em seus efeitos, mas são mais relevantes do que qualquer outro aspecto de trabalho. Segundo os autores, a unicidade de cada trabalhador é um incômodo constate para o gerente que cegamente adota um estilo de gestão do mundo da produção. As pessoas que são “gerentes naturais”, por outro lado, percebem que esta unicidade é o que faz a química do projeto vital e efetiva, algo a ser cultivado.

Os autores alertam quanto a falha de falar de produtividade sem falar de *turnover*, já que tipicamente o que é feito para melhoria de produtividade - pressionar pessoas, mais horas de trabalho, padronizar procedimentos, comprometer qualidade - potencializa ou faz com que o trabalho seja menos satisfatório e agradável.

“Produtividade deveria significar conquistar mais dentro de uma hora de trabalho, mas frequentemente tem significado extrair mais de uma hora paga.”[DeMarco 87]

Esta afirmação nos leva à importância de traçar estratégias para motivação do time em contrapartida às demais ações, sempre levando o *turnover* em consideração.

Além do *turnover*, os autores apontam muitos fatores como relevantes para a produtividade, entre eles: a quietude no ambiente de trabalho e a iluminação natural. Eles embasam esta afirmação mostrando que um ambiente de trabalho barulhento leva a um maior número de defeitos. “Trinta por cento do tempo as pessoas são sensíveis ao barulho, os outros setenta por cento são geradoras de barulho”[DeMarco 87].

Outro ponto relevante é a visão da interrupção no trabalho como um problema importante, onde é introduzido o E-Factor como uma razão entre horas de trabalho ininterrupto por horas presentes no trabalho. Os autores ainda afirmam que:

Algumas organizações estão se saindo melhor do que outras porque estão lidando com pessoas de forma mais efetiva, modificando seus ambiente de trabalho e cultura organizacional, além de implementar algumas medições citadas posteriormente.

Por fim, os autores chegam a uma lista de seis fatores que eles chamam de “assassinato de times”[DeMarco 87]: gestão defensiva; burocracia; separação física dos integrantes da equipe; time em tempo parcial; redução da qualidade do produto; e *deadlines* irreais.

Este trabalho concorda com os autores quando eles afirmam que “estamos todos sobre pressão para melhorar a produtividade, mas o problema não é aberto a soluções fáceis, porque todas elas foram pensadas e aplicadas há muito tempo atrás”. É preciso levantar o que já foi feito, somar uma série de ações e colocá-las em paralelo.

O único ponto em que este trabalho não concorda com DeMarco é quanto a não influência do salário na produtividade. Esta discussão é vasta dentre os administradores e neste trabalho trataremos um pouco mais na estratégia relacionado a recompensas. Onde mesmo não sendo o ponto mais importante, o reconhecimento financeiro é um tipo de recompensa, fator com impacto positivo na produtividade.

Banker et al. aponta que conhecendo os fatores que influenciam na produtividade de suas equipes, os gerentes podem influenciá-los positivamente [Banker 87]. Os autores estão entre os primeiros a afirmar que o conhecimento dos fatores que influenciam na produtividade permite que a gerência encoraje as influências positivas e elimine as negativas. O mais interessante é que os autores mencionam em seu artigo que pesquisadores e desenvolvedores “a muito” são interessados em fatores que influenciam na produtividade. Esta afirmação continua verdadeira, mesmo mais de vinte anos depois.

Scacchi [Scacchi 89] aponta como fatores para alcançar alta produtividade de um projeto de software: processo, estabilidade de requisitos, organização da equipe, qualidade na gestão do projeto, experiência e capacidade da equipe, e reuso.

Gilb [Gilb 88](apud [McConnell 04]) introduziu a distribuição evolutiva, de partes do produto, e estabeleceu a base para grande parte da estratégia de programação ágil hoje.

Banker e Kemerer [Banker 89] afirmam que o uso de ferramentas, até então mencionadas como fatores positivos na produtividade, podem ser muito custosas para projetos pequenos, seja pelo custo de aquisição da ferramenta ou pelo custo de treinamento da equipe. Os autores ainda afirmam que todo projeto possui um investimento fixo - *overhead* - de gerenciamento de projeto, portanto a produtividade média aumenta inicialmente com o crescimento do projeto. Embora, na maioria das vezes, grandes projetos se tornam mais complexos de gerenciar e a produtividade do time tende a cair.

Outros estudos foram importantes para mapeamento dos fatores que influenciam na produtividade, bem como na definição das estratégias para sua melhoria [Bailey 81], mas de forma mais pontual do que os citados acima.

3.4 CONTRIBUIÇÕES DOS ANOS NOVENTA (1990 - 1999)

Segundo Yu, Smith e Huang [Yu 91] para ter melhoria na qualidade e na produtividade no processo de desenvolvimento de software é preciso medições precisas de entradas e saídas desse processo, e para isso os fatores de produtividade devem ser identificados e compreendidos. Os autores mapeiam três estágios no processo de melhoria da produtividade de software: a medição é o primeiro deles e é a base fundamental para os próximos. Com um conjunto confiável e operacional de medições de produtividade, o produto e o processo podem ser medidos e analisados para identificar quais os fatores possuem maior impacto na produtividade e qualidade. Só então, as alterações no processo de desenvolvimento e no ambiente podem ser definidas e realizadas.

Ainda motivados pelos resultados do trabalho de DeMarco e Lister, os anos 90 mostraram um maior interesse pelos fatores tidos como não técnicos ou abstratos que influenciam na produtividade.

Rasch nos trouxe estratégias para montagem do time e definição clara de papéis, obrigações tanto no contexto de processo, quanto no contexto de projeto. Rasch apresenta os resultados de estudos sobre os efeitos dos fatores na satisfação do time como: rotatividade dos integrantes da equipe; ambigüidade; e conflitos entre papéis [Rasch 91].

Já Lakhanpal se concentrou nas características dos grupos e sua influencia na produtividade [Lakhanpal 93]. A coesão e a capacidade técnica tinham a mais forte influência em 31 grupos de desenvolvedores, onde experiência teve a influência mais fraca. Mais uma vez, temos boas estratégias quanto à construção de times, desta vez relacionada à experiência e coesão.

Segundo Blackburn e Scudder [Blackburn 96], existem muitos fatores influenciando na produtividade de software que estão além do controle do gerente ou da organização, como: complexidade, tamanho do projeto, envolvimento ou não de projeto de hardware. O estudo relata quais práticas gerenciais resultam em maior produtividade e velocidade. Dentro dos pontos observados, os fatores humanos parecem superar ferramentas e tecnologia. Em uma das amostras, os fatores mais relevantes foram especificações do cliente (mudança de requisitos), comunicação e programadores melhores (mais experientes e habilidosos). Em uma segunda amostra, reuso foi o fator mais apontado. Os autores concluem que quanto a escolher entre investir em fatores humanos ou tecnológicos, as

pesquisas por eles realizadas sugerem que pessoas talentosas são mais importantes do que fatores tecnológicos.

Maxwell e demais autores realizaram uma pesquisa onde reafirmaram um relevante grupo de fatores que influenciam na produtividade no desenvolvimento de software, impactando nas medições, estimativas e comparações de produtividade [Maxwell 96].

Os autores afirmam que algumas das diferenças quanto à produtividade nas organizações podem ser atribuídas à linguagem de programação de baixa produtividade - como Coral e Assembly, entretanto algumas diferenças vêm da forma que os projetos de desenvolvimento de software são gerenciados. Além disso, os autores reafirmaram que a produtividade tende a cair com limitações de armazenamento, tempo, tamanho do time, confiabilidade dos requisitos e duração do projeto. O que os autores entram em divergência com a abordagem adotada neste trabalho e ainda com a maioria das citações [Albrecht 79, Behrens 83, Vosburgh 84, Jiang 07b] é que eles afirmam que a produtividade cresce com o tamanho do software.

Esse é um trabalho inspirador quanto ao número de referências, qualidade das informações e abrangência dos fatores. Um dos objetivos deste trabalho inclui catalogação dos fatores que influenciam na produtividade e estratégias de melhoria da produtividade, de forma similar ao realizado pelos autores.

Brooks [Brooks 95] enriquece este trabalho trazendo uma série de estratégias na segunda edição de seu livro *The Mythical Man-Month*, em “uma visão ampliada com pen-samentos mais atuais sobre o tema”. Em seu trabalho o autor afirma que:

A maioria das grandes conquistas que já ocorreram em produtividade de software vem da remoção das barreiras artificiais que já ocorreram em produtividade de software que tornaram as tarefas acidentais dificílimas, como limite de hardware, linguagens de programação, falta de tempo da máquina. Não só não existem balas de prata a vista, como a própria natureza do software torna improvável que venha a existir alguma - não há invenção que faça pela produtividade, confiabilidade e simplicidade do software o que os componentes eletrônicos, transistores e integração em larga escala fizeram pelo hardware de computadores.

Dos autores estudados até o momento muitos mencionavam reuso como fator impactante na produtividade [Walston 77, Bailey 81, Boehm 82, Boehm 87a, Boehm 88, Scacchi 89]. Devido à relevância deste fator e sua constatada contribuição para a produtividade, trabalhos específicos sobre reuso foram surgindo, dentre eles destacamos

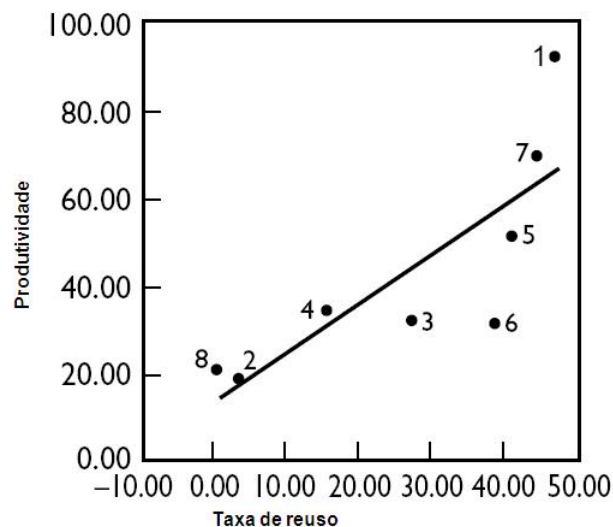


Figura 3.3. Relacionamento linear entre produtividade e taxa de reuso [Basili 96]

[Lim 94, Basili 96, Butler 97]. A escolha dos trabalhos se baseou na relevância quanto a reuso como uma estratégia de melhoria de produtividade. Esses autores analisam o efeito de reuso na produtividade e qualidade.

Lim [Lim 94] apresenta os resultados de dois programas de reuso que documentaram a melhoria de produtividade e qualidade entre outros ganhos. Esse trabalho permitiu o levantamento de justificativas para adoção da estratégia reuso, dentre elas: a produtividade aumenta porque usuários de “produtos de trabalho reusáveis” precisam fazer menos trabalho; reuso permite o uso mais efetivo dos colaboradores e suas *expertises*, já que permite que os mais experientes possam criar o trabalho e os menos apenas reutilizá-los; como os produtos de trabalho são utilizados por múltiplos times, as correções de cada reuso aumenta sua qualidade; produtividade aumenta porque menos produtos de trabalho são feitos desde seu primeiro esboço; o ciclo de vida requer menos *inputs* para obter o mesmo *output*; e por fim, porque reuso trás melhoria da produtividade por reduzir o tempo e esforço necessário para desenvolver e manter produtos.

Basili, Briand e Melo [Basili 96] mostraram em seu estudo um forte impacto do reuso na produtividade no desenvolvimento de sistemas orientados a objetos. A Figura 3.3 mostra a relação linear entre a produtividade e a taxa de reuso, segundo resultados do estudo dos autores.

Butler oferece informações sobre o custo benefício do reuso em [Butler 97]. O autor afirma que reuso é uma abordagem que trás retorno em longo prazo. É preciso ter isso em mente, algumas estratégias apenas serão de fato confirmadas como positivas se

trabalhadas com foco em ganho em longo prazo.

Ainda sobre reuso, Basili e demais autores analisaram como o reuso influencia na produtividade [Basili 96]. O artigo apresenta resultados significativos quanto ao forte impacto de reuso na melhoria de produtividade e na qualidade do produto gerado.

Já Bruckhaus e demais autores apresentam uma série de estratégias relacionadas a ferramentas [Bruckhaus 96]. Os autores afirmam que na escolha de ferramentas para apoiar no desenvolvimento de software é preciso ter em mente o tamanho do projeto e o processo de desenvolvimento a ser adotado. A complexidade do processo e o tamanho da equipe são fatores importantes por definirem que funções serão utilizadas e sua frequência de uso.

Segundo Boehm [Boehm 99], reuso pode elevar a produtividade em até 50%, se as armadilhas comuns que descarrilam muitos programas de reuso forem evitadas. O autor apresenta algumas armadilhas encontradas na utilização do reuso que podem ajudar na definição de estratégias para adoção do reuso, entre elas: esperar a criação de um repositório de componentes para iniciar o reuso; construir componentes sem uma arquitetura, especificações de interfaces bem definidas; e generalizar demais para suportar um grande número de sistemas futuros.

Ainda segundo o autor, os fatores e recursos críticos para o sucesso do reuso são:

- Adote uma abordagem de linha de produto - definir um domínio ideal de atuação para a organização com arquiteturas e soluções específicas;
- Realize uma análise de negócio para determinar o escopo correto e nível de expectativa das linhas de produto;
- Foque em conseguir reuso caixa preta - evitando abrir e ajustar o componente incluindo novos custos;
- Estabeleça um gerente de linhas de produto com poderes, garantindo investimentos para as linhas de produtos e ainda se certificando que os componentes sejam utilizados.
- Estabeleça um processo orientado a reuso;
- Adote a abordagem incremental - empregando cuidadosamente pilotos e feedback;
- Utilize gestão de reuso baseada em medição;
- Estabeleça uma linha de produto pró-ativa e em evolução - isso é sua proteção quanto a ficar obsoleto.

Scacchi em 1995 [Scacchi 95] apresenta um trabalho semelhante ao de 1989 [Scacchi 84], onde ele cita diversos fatores que influenciam na produtividade, dispostos na seguinte classificação: Ambiente; produto; e equipe. O autor afirma que nem sempre é possível ou desejável atingir melhoria de produtividade endereçando todas as estratégias.

Este é mais um reforço da relevância da Meta-Estratégia proposta neste trabalho, já que cada organização ou até mesmo cada time vai gerar sua própria estratégia para melhoria de produtividade, podendo escolher dentre as estratégias levantadas na literatura, o grupo que mais vai agregar valor às suas organizações.

Algumas discrepâncias surgiram ao longo das pesquisas como foi o caso de do trabalho de Gowda e Chand [Chand 93]. O artigo relata o resultado de um estudo empírico que explora o impacto de fatores de indivíduos e de grupo na produtividade. A hipótese base do artigo é que dado um conjunto de ferramentas e um ambiente de trabalho adequado, a produtividade do grupo é dada em uma função das dinâmicas de grupo. Até este ponto, o pensamento dos autores convergia com o deste trabalho, entretanto não concordamos com a afirmação *o impacto dos fatores humanos na produtividade do software não é um problema no desenvolvimento do software*. Segundo os autores o foco dos profissionais de engenharia de software é a melhoria de ferramentas e técnicas para melhorar o ambiente de desenvolvimento. Desconsiderando os estudos que abordam fatores humanos tais como os trabalhos de Boehm já citados anteriormente, ou ainda referências como [Walston 77, Bailey 81, Vosburgh 84, Thadhani 84, Banker 87] entre outros.

O artigo ainda chega a conclusões sobre outros fatores que influenciam ou podem influenciar na produtividade como: maturidade da equipes, experiência na linguagem, complexidade das tarefas, ênfase do líder e outros.

Alguns trabalhos de fora da área da engenharia de software também chamaram atenção durante a pesquisa bibliográfica sobre produtividade. O trabalho de Dumaine em [Dumaine 94], por exemplo, enriqueceu o mapeamento de estratégias para construção de times motivados. O seu trabalho relata uma afirmação do presidente da Boeing, Philip Condit, “A sua competitividade é a sua habilidade de usar efetivamente as habilidades e conhecimentos das pessoas, e times são a melhor maneira de fazer isso.”

O autor nesse trabalho menciona vários pontos no contexto de motivação, recompensa e montagem de times. O papel do gerente como um facilitador é fundamental nesta visão. Dentre eles a importância de se premiar não apenas o indivíduo, mas toda uma equipe, de forma a encorajar o trabalho em equipe. Segundo Dumaine [Dumaine 94] as empresas que premiam como times também premiam como indivíduos só que com uma diferença: eles fazem do trabalho em equipe - habilidade de lidar com outros, e atitude de apoio e

ajuda a outros - um ponto chave para a revisão da performance individual.

Para encerrar os anos 90, temos mais uma contribuição de Boehm, onde o autor afirma que para uma organização conseguir produtividade ela pode escolher entre três estratégias para melhoria de produtividade, ou usá-las de forma complementar [Boehm 99]:

- **Trabalhar mais rápido** – Aumentar a velocidade de produção, através de ferramentas e acelerar tarefas tipicamente manuais;
- **Trabalhar de forma mais inteligente** – Primeiramente através de melhoria de processo que evita ou reduz tarefas que não agregam valor;
- **Evitar trabalhos desnecessários** – Reusar software e documentação sempre que possível, ao invés de desenvolvê-los para cada projeto.

Segundo ele, as duas primeiras têm um impacto de 8% e 17%, respectivamente. Enquanto que a prática de reuso apresenta um impacto de 47% na produtividade.

3.5 ÚLTIMAS CONTRIBUIÇÕES (2000-2009)

Desde DeMarco e Lister [DeMarco 87] os fatores tidos como não técnicos e as estratégias par lidar com eles continuaram em evidência.

Em [Boehm 00a], Boehm e demais autores apontam as principais variáveis que afetam a produtividade. Nele são apontadas 22 fatores, divididos em quatro grupos similares à divisão realizada no COCOMO [Boehm 81]. Os autores ainda analisam os fatores tidos como técnicos - exemplo complexidade de produtos - e também uma variedade de fatores tidos como *soft*. Segundo os autores, esses fatores - como capacidade do programador e baixo *turnover* - combinados são mais importantes que os fatores técnicos. Entre os trabalhos com participação de Boehm, este foi o menos explorado por este trabalho.

Para entendermos melhor o que Bohem quis dizer com fatores *soft*, vamos analisar o trabalho de Lokan [Lokan 01]. O autor afirma que o impacto de fatores quantificáveis como tamanho, linguagem, plataforma, tamanho do time foram estudados exaustivamente. Quanto a esses fatores, o autor ainda afirma que são controláveis no início do projeto, mas não representam nem 50% das variações de esforço em um projeto. O resto da variação é explicado por outros fatores mais relacionados com pessoas - sejam elas gerentes, clientes ou desenvolvedores, fatores chamados de *soft*. Esses fatores são difíceis de quantificar e analisar, e são frequentemente subjetivos. Entre os fatores *soft* citados temos: experiência e habilidade da equipe; volatilidade dos requisitos; e envolvimento, experiência e comprometimento do cliente.

Um fator muito apontado, desde o final da década de noventa, como relevante para a produtividade no desenvolvimento de software é a experiência da equipe. O recurso mais crítico para times que trabalham com conhecimento é o próprio conhecimento, experiência ou habilidades específicas [Faraj 00]. Entretanto, possuir apenas experiência não é suficiente, é preciso gerenciá-la para chegar ao seu potencial. O autor ainda afirma que continuidade do time é algo raro, o que se aproveita é a parte de infra-estrutura e tecnologia.

No mesmo ano, Maxwell e Forselius [Maxwell 00] fazem um *benchmarking* com empresas de diferentes setores de negócio e identificam 15 fatores que influenciam na produtividade. Estabilidade dos requisitos é, por exemplo, um fator que afeta negativamente na produtividade dos setores de banco, seguro e manufatura. Eles também destacam que a alta variação de produtividade entre as companhias deixa evidente a necessidade destas estabelecerem suas próprias bases de dados de métricas de software, além de comparar seus dados com os de outras companhias.

Este trabalho não mapeia setores de negócio ou a abordagem de linhas de produto como estratégia, apesar de concordar que é relevante.

Em um estudo de caso para a indústria civil, Hantos e Gisbert [Hantos 00] citam que projetos de sucesso são baseados em análise e gestão por todo ciclo de desenvolvimento de software e a compreensão total e priorização das limitações dos recursos. Seu estudo de caso mostra a importância de times colaborativos e do conhecimento de como uma tarefa afeta outros colegas de trabalho e o projeto como um todo. Segundo os autores, este único ponto tem um grande impacto na capacidade de entrega da organização e na sua produtividade.

Esse artigo, apesar de não ser citado dentre as grandes referências, foi muito relevante para a este trabalho pelas estratégias sugeridas. Os autores citam técnicas de gestão mencionadas hoje pelos métodos ágeis. Este engajamento de todos que fazem parte do time, permite uma maior visibilidade do todo e, por conseguinte, do impacto que o atraso de uma atividade tem na próxima dentro da linha de produção. Quando sabemos o que o nosso trabalho impacta no próximo e no todo, buscamos ser mais produtivos e assertivos.

Os autores ainda citam a importância de vibrações e ambientes onde se comemora vitórias e novos inícios. Eles chamam de “atmosfera como carnaval” a abordagem deste ambiente no estudo de caso observado [Hantos 00]. Sobre este artigo ainda foram extraídas estratégias para reduzir *turnover*, aumentar a satisfação do time e ainda algumas dicas referentes à condução de projetos grandes.

Reifer afirma que a produtividade pode ser melhorada de duas formas [Reifer 02]:

- Com uma estratégia baseada nas entradas, melhorando a produtividade da força de trabalho através de métodos, ferramentas, ambiente de trabalho e equipamentos.
- Com estratégia baseada nas saídas, onde a ênfase seria dada na redução da quantidade de saída necessária com o uso de componentes, linhas de produto, arquitetura centrada no reuso, eliminando assim parte do trabalho envolvido no desenvolvimento de produtos.

Apesar de concordar com essa perspectiva, este trabalho não divide as estratégias desta forma. As estratégias são agrupadas por fatores que influenciam na produtividade.

Clincy propõe que existem quatro áreas que mais impactam na habilidade de uma organização em aumentar sua produtividade e diminuir o tempo do ciclo de desenvolvimento. As quatro áreas são: Estrutura e clima organizacional, Sistemas de recompensa (definição de como recompensar efetivamente os times com alta produtividade), Processos de desenvolvimento de software e o uso de ferramentas [Clincy 03]. Esse trabalho foi uma fonte rica, de onde saíram diversas estratégias para compor, manter e recompensar times e ainda para definição e melhoria de processo. Pela sua simplicidade e clareza, este trabalho também orientou nos experimentos não exploratórios.

Maxwell afirma que para a indústria melhorar os salários é preciso uma melhoria de produtividade ano a ano [Maxwell 03]. Segundo a autora, este é um momento de esforço sério neste sentido, através de um uso mais eficiente da mão-de-obra e proposta mais efetiva de valores para o cliente. É desta forma que setores econômicos mais maduros têm crescido. A autora sugere boas estratégias para melhorar a produtividade tais como: não trabalhar em requisitos que o cliente não ache que agregue valor, reduza o custo de processo eliminando deficiências; e aumento do valor agregado do produto.

3.5.1 A coletânea de estratégias oriundas das metodologias ágeis

Dentro desta visão de focar nos requisitos que agregam valor ao cliente, alguns artigos e trabalhos apontam para uma tendência de métodos ágeis para endereçar este fator e alcançar a produtividade. Muitas práticas mapeadas nos métodos ágeis são boas estratégias para melhoria de produtividade, entretanto suas origens são de uma coletânea de trabalhos científicos e estudos de casos publicados e comentadas ao longo dos anos. As estratégias - mapeadas como práticas nas metodologias ágeis - foram documentadas neste trabalho, mas dando o mérito da estratégia para os autores da primeira citação. De qualquer maneira, a popularização desta coletânea justifica uma boa descrição.

As metodologias ágeis e suas estratégias tiveram início com o manifesto ágil

[Highsmith 01b], onde representantes de várias metodologias ágeis se uniram para, segundo os assinantes do manifesto, descobrir formas melhores para desenvolver software. O manifesto defende os seguintes valores:

- Indivíduos e interação sobre processos e ferramentas;
- Software executável sobre documentação extensa;
- Colaboração com o cliente sobre negociação de contratos;
- Responder a mudanças sobre seguir o plano.

A idéia dominante do desenvolvimento ágil é que o time pode ser mais efetivo na resposta a mudanças, por reduzir o custo de mover a informação entre as pessoas e ainda por reduzir o tempo entre a tomada de decisão e realizar o que foi decidido e ver suas conseqüências [Cockburn 01]. Cockburn e Highsmith afirmam que o desenvolvimento ágil foca no talento e habilidade dos indivíduos, moldando o processo para pessoas específicas e times específicos. Os autores ainda apontam que os times ágeis ficam fisicamente próximos, substituem documentação por conversas e são auto-organizados. Em outro artigo no mesmo ano os autores afirmam que a estratégia dos métodos ágeis é reduzir o custo das mudanças o que pede que os times ágeis realizem a primeira entrega em semanas, criem soluções simples e melhorem sua qualidade continuamente fazendo a próxima implementação mais barata, e testando constantemente. Tudo isso com proximidade e iteração contínua [Highsmith 01a].

Mais uma vez temos a afirmação de que cada organização e ainda cada projeto deve ser tratado separadamente pelas suas características específicas. Não existe um modelo ideal ou uma estratégia ideal para todas as organizações, é preciso analisar as características de cada projeto e ainda a cultura da organização para definição de seu conjunto de estratégias.

Autores já citados neste trabalho apontam para a adoção de um processo iterativo e adaptativo como a melhor maneira de lidar com estas mudanças. Os modelos de processo incrementais ou evolutivos tentam lidar com esses problemas, assim como as metodologias ágeis e os processos ágeis também.

Uma das justificativas para a classe ágil de processo de software é que requisitos dos clientes mudam com frequência [Reed 04]. Os autores comentam que segundo a sabedoria convencional, mudança de requisitos continua inevitável e até pode ter um efeito positivo no ambiente, negócios ou entendimento das operações de desenvolvimento ao longo do tempo. Por outro lado, eles afirma que a coleta informal das solicitações de mudanças

de requisitos pode aumentar a probabilidade de características mais importantes não serem implementadas inicialmente, o que normalmente quer dizer que elas nunca serão implementadas. Os autores ainda apontam a prototipação como uma estratégia conhecida para lidar com problemas de definição dos requisitos do cliente quando o sistema finalizado pode ter um grande impacto nas expectativas do cliente.

Ainda em [Reed 04], os autores definem o termo *time-box* como ciclos de desenvolvimento bem definidos. Segundo eles estas janelas de tempo definem momento de início e fim para ciclos de desenvolvimento, prevenindo grandes mudanças de esforço durante o projeto. A idéia é que qualquer que seja o tamanho desta janela, o cliente seleciona os requisitos mais importantes para serem desenvolvidos e os desenvolvedores fazem o máximo possível para ficarem de fato pronto dentro daquela janela de tempo. Essas janelas, quando definidas para atividades, normalmente são de quatro a dezesseis horas.

Beck aponta que XP (eXtreme Programming ou Programação Extrema) é uma metodologia para times pequenos e médios que desenvolvem software em face a requisitos vagos, que se modificam rapidamente [Beck 04]. A estratégia de gerenciamento no XP está na tomada de decisão descentralizada. Numa visão geral da metodologia, o autor descreve as práticas de XP, dentre elas algumas podem ser entendidas como estratégias para endereçar alguns fatores, e serão mapeadas posteriormente em conjunto com estratégias semelhantes. Entre elas:

- Jogo do planejamento - envolvimento de todos no planejamento, onde a área de negócios precisa da área de desenvolvimento para tomar decisões técnicas, antecipar riscos técnicos.
- Entregas freqüentes - uma entrega com o menor tamanho possível, com o maior valor para o negócio.
- Projeto simples - a coisa mais simples que pode funcionar. A equipe de um projeto simples se comunica melhor; a simplicidade garante a fácil execução; um rápido feedback já que o projeto simples termina rapidamente; por fim, a coragem de fazer apenas uma pequena parte do projeto, sabendo que se for necessário acrescenta-se depois.
- Propriedade coletiva - todos conhecem pelo menos um pouco de todo o sistema, a qualquer momento ao observar uma oportunidade de melhoria no código ela será implementada.

- Cliente presente - um cliente real - que vá de fato usar o sistema - deve estar disponível para responder as questões.
- Padrões de codificação - o padrão adotado deve exigir a menor quantidade de trabalho possível, consistentes com a regra “uma e somente uma vez”(sem código duplicado).

Da visão apresentada foi mapeada a participação da equipe técnica no planejamento, as entregas freqüentes em conjunto com modelos iterativos, simplicidade no código, adoção de padrões e ainda a participação do cliente quando possível. Quanto à simplicidade do código, levantamos que ações para efetivar esta premissa são extremamente relevantes, já que a complexidade do software foi um dos fatores mais citados nesta pesquisa bibliográfica. Portanto este foi um ponto endereçado neste trabalho. A participação do cliente também é uma prática relevante, citada por muitos trabalhos analisados como sendo positiva para a produtividade, também é uma estratégia a ser adotada quando possível.

Há autores que defendem que a adoção de apenas algumas práticas sugeridas pela metodologia ágil não é eficaz. Shore e Warden [Shore 08] afirmam que criar um método ágil totalmente novo não é uma idéia muito boa, já que existe muito mais no desenvolvimento ágil do que suas práticas. De qualquer maneira, algumas práticas citadas pelos autores, são vistas como estratégias para melhoria de produtividade. Algumas citações dos autores são corroboradas por outros autores e serão apontadas no mapeamento das estratégias. Dentre elas destacamos: *“Prefira melhor à maior.”*, onde os autores se referem a pessoas mais experientes do que a um número maior de pessoas; e *“sua produtividade pode melhorar instantaneamente remanejando pessoas para apenas um projeto de cada vez... A atribuição fracionada é extremamente contra-produtiva.”*

O livro é uma rica referência para quem deseja seguir o conselho dos autores e adotar toda a XP ou em nossa visão parte dela, pela simplicidade adotada e abrangência na descrição de cada prática. Dentre as práticas citadas, as mais relevantes para este trabalho são: trabalho energizado, espaço de trabalho informativo, confiança, sentar junto, reuniões rápidas, adoção de padrões e equipes inter-funcionais e auto-organizadas.

Já o Scrum é um processo simples para lidar com projetos complexos [Schwaber 04]. A metodologia adota algumas das estratégias já citadas - uso do *time-box*, times pequenos, auto-organizados entre outros - e afirma conseguir: uma gestão efetiva de requisitos desconhecidos ou em mudança; simplificação da cadeia de comando com a auto-gestão; construção de produtos e entrega em 30 dias; e evitar perdas através de inspeções regulares, maximizando o retorno do investimento.

Para encerrar esta breve revisão da contribuição dos métodos ágeis para estratégias de melhoria de produtividade, temos uma visão comparativa entre a abordagem tradicional baseada em processo e baseada na agilidade por Boehm e Turner [Boehm 03]. Suas observações mostraram que as duas abordagens, tanto a ágil quanto a tradicional com foco em processo, são dois meios para o mesmo fim: satisfazer clientes com software que atendem suas necessidades, dentro dos custos e prazos. O artigo é apenas um resumo de um trabalho mais aprofundado realizado também em conjunto com Turner. Os autores defendem que ambas as abordagens podem ser caracterizadas como *“lead bullets”*. Por fim eles afirmam que estratégias vêm surgindo para integrar as duas abordagens de forma a tirar vantagem dos pontos fortes e evitar os pontos fracos. Mesmo analisando apenas o resumo do trabalho é evidente que o ideal é somar as boas práticas, os sucessos e não ter que escolher de forma excludente. Para gerir um projeto é preciso todas as armas disponíveis, sejam elas oriundas de uma visão mais ágil ou de uma visão mais focada em processo.

3.5.2 Concluindo as últimas contribuições

McConnell [McConnell 04] escreveu um livro que contém boas práticas de programação. Entre outras coisas o autor apresenta uma análise de alguns fatores e ainda uma rica fonte de estratégias, tais como:

“A medida que o tamanho do software aumenta, o tamanho e a organização da equipe se tornam influências maiores sobre a produtividade. Em um projeto pequeno a construção é de longe a atividade mais proeminente, ocupando 65% do tempo total do desenvolvimento. Em um projeto de tamanho médio, a construção ainda é dominante, mas sua parcela no trabalho total cai cerca de 50%. Em um projeto grande a arquitetura, a integração e os testes de sistemas ocupam mais tempo, e a construção se torna menos dominante. Em resumo, à medida que o tamanho do projeto aumenta, a construção se torna uma parte menor do esforço total.”

“Em um projeto pequeno o talento do programador é a maior influência sobre a qualidade do software (...) Em um projeto maior as características organizacionais fazem uma diferença maior do que as capacidades dos indivíduos envolvidos. Mesmo que tenhamos uma excelente equipe, a capacidade coletiva que dela resulta não é simplesmente a soma das capacidades individuais. A

maneira com que as pessoas trabalham juntas determina-se as suas capacidades são somadas ou subtraídas. O processo define como uma pessoa apóia o trabalho do restante da equipe.”

McConnell [McConnell 04] afirma que independente do tamanho do projeto, algumas técnicas são sempre valiosas, como: práticas de codificação disciplinadas, inspeções de código realizadas por outros desenvolvedores, bom suporte de ferramenta e uso de linguagens de alto nível. Essas técnicas são valiosas para projetos pequenos e imprescindíveis em projetos grandes. O autor ainda aponta algumas estratégias, tais como:

- Revisão, inspeções e testes como uma maneira de compensar as falibilidades humanas.
- Manter as rotinas curtas para reduzir a carga em seu cérebro.
- Escrever programas em termos de domínio de problema, ao em vez de escrever em termos dos detalhes da implementação de baixo nível, para reduzir a carga mental de trabalho.
- A divisão de um sistema em subsistemas em nível de arquitetura.
- A definição criteriosa de interfaces de classe para que você possa ignorar o funcionamento interno da classe.
- Evitar aninhamentos profundos de loop e condicionais, já que eles podem ser substituídos por estruturas de controle simples.
- Definir de forma criteriosa o tratamento de erros, invés de usar uma proliferação arbitrária de diferentes técnicas de tratamentos de erro.
- Manter os nomes das variáveis claros e auto-explicativos.

McConnell [McConnell 04] ainda aponta que devido a um mercado competitivo, metade do que se precisa saber agora ficará obsoleto em três anos, e ainda assim frequentemente os programadores não têm tempo para serem curiosos, não sobra tempo para desenvolvimento das habilidades. Para evitar se tornar um dinossauro, o autor sugere:

- Experimente, faça novas experiências com programação e o processo de desenvolvimento.
- Leia sobre as soluções de problemas.

- Analise e planeje antes de agir.
- Aprenda examinando projetos bem sucedidos.
- Leia outros livros e periódicos.
- Comprometa-se com o seu desenvolvimento profissional.

McConnell neste seu trabalho foi responsável por muitas das estratégias mapeadas no Capítulo 4.

Nesta década novos trabalhos voltaram a nos chamar a atenção quanto à importância do reuso na produtividade do software. Kitchenham e Mendes [Kitchenham 04] encontraram que reuso tem um grande impacto em produtividade. Da mesma forma, Mohagheghi e Conradi [Mohagheghi 07] analisaram a influência positiva do reuso na produtividade. Simmons também aponta reuso como a forma mais rápida de desenvolver software, podendo realizar o reuso através do uso de COTs (Components of the shelf), produto de outro fornecedor, de bibliotecas ou de um amigo [Simmons 07]. O autor ainda destaca complexidade do software, qualidade na gestão e tamanho do time, como fatores que influenciam na produtividade. O tamanho do time ainda apresenta algumas consequências como comunicação e alto *turnover*. O que chama a atenção no trabalho é a afirmação: *Apenas 62,5 % de nosso tempo passamos de fato desenvolvendo, o resto do tempo passamos com reuniões, treinamentos, socialização com colegas entre outros.*

Groth [Groth 04] mostra em seu artigo vários depoimentos de executivos quanto a uma afirmação realizada sobre queda na produtividade. Um dos executivos foi Eric Schurr, vice presidente de marketing da IBM-Rational. Schurr afirma que:

“Dez anos atrás ou até mesmo cinco, as ferramentas disponíveis não eram nem perto tão sofisticadas quanto as de hoje. Um exemplo é que qualquer desenvolvedor pode conseguir um servidor de aplicação completo no domínio opensource (...) as três áreas de melhoria de produtividade são: automação do processo de desenvolvimento de software; introdução de melhores práticas - incluindo a adoção de componentes já desenvolvidos; e utilização de soluções de automação de teste.”

Anselmo e Ledgard [Anselmo 03] apontam o “entendimento” como um ponto importante para lidar com grandes produtos de software. Um bom entendimento do código - bem documentado - é apontado como um fator de grande impacto na produtividade. O acoplamento em projetos é apontado como uma propriedade chave da produtividade

do projeto (design) de software, onde os autores afirmam que uma boa visualização e compreensão da arquitetura proposta contribuem para módulos independentes - menos acoplados. E ainda que quanto mais facilidades possuírem o ambiente para o desenvolvimento das funções, maior a produtividade.

“A abordagem incremental de software, onde funcionalidades são adicionadas em pequenos pedaços, normalmente gera um build diário. Com o apoio de ferramentas é possível adicionar pequenas funcionalidades, fazer um pouco de projeto, construir um pouco, testar um pouco e crescer o sistema de forma incremental, garantindo que os componentes estão atendendo às especificações e mostrando resultados de curto prazo” [Anselmo 03].

Nesta década também surgiu novamente uma das expressões que atuaram como motivação para este trabalho. A expressão *bala de prata* é polêmica, já que na verdade se refere à ausência de uma solução, mas em [MacCormack 03] os autores relembram que para esta necessidade de melhoria organizacional o problema nunca foi à falta de “balas de prata”, e sim o “como escolher” dentre o arsenal extenso de coloridas balas de prata, que ficou impossível. A Meta-Estratégia quando conduzida com apoio do mapeamento das estratégias encontradas na revisão bibliográfica ajuda na escolha do que parece mais adequado para cada organização.

Chiang e demais autores [Chiang 04] mostram em seu trabalho que mesmo que a maioria dos estudos de produtividade foquem em reduzir diretamente o esforço de codificação, a produtividade de um time também é profundamente influenciada por como o esforço de gestão é distribuído. Para os autores, aprimorar a produtividade do desenvolvimento de software requer uma abordagem balanceada dos três pilares do gerenciamento de software: tecnologia, pessoas e processos. Esses três pilares englobam grande parte dos fatores apontados por outros autores em seus estudos.

Dentro deste contexto, é preciso refletir que nem todas as variáveis que impactam na produtividade são conhecidas. Produtividade é impactada por uma larga quantidade de fatores, muitos deles de difícil acesso, como: dificuldade da tarefa, habilidade da equipe do projeto, nível de interação com o cliente, nível dos requisitos não funcionais impostos pelo cliente, tais como performance [Premraj 05].

Estudos realizados pelos autores mostram: um relacionamento claro da produtividade com o tamanho do projeto, onde erros maiores são associados a projetos maiores; impacto do setor de negócio na produtividade, onde o setor de negócio mais improdutivo nos últimos anos foi o setor de seguros; e os modelos de processo como um fator de alta influência na produtividade.

Simmons [Simmons 07], além de fornecer a visão que acrescentar pessoas a equipe de um projeto aumenta o tempo nominal necessário para produzir o produto de software, afirma que equipes de desenvolvimento gastam um tempo considerável se comunicando entre si e com pessoas de fora da equipe. O autor aponta que interrupções causam perda de produtividade, onde a duração média de uma interrupção de trabalho é cinco minutos para um programador típico, e o tempo necessário para retornar à trilha do pensamento após a interrupção é em torno de dois minutos.

Mesmo com tantos estudos comprovados durante as últimas décadas, o estudo de Jiang, Naudé and Comstock [Jiang 07b], [jia] mostra que devido à crescente complexidade e magnitude dos projetos, a produtividade no desenvolvimento de software não tem crescido de forma consistente. O trabalho considerou uma análise no ISBSG (International Software Benchmarking Standards Group), quanto à produtividade e os fatores que a influenciam, revelando que a produtividade tem experimentado variações irregulares de crescimento e queda, entretanto não existe nenhum sinal eminente de sua melhoria.

Empresas que partem para desenvolvimento distribuído, para reduzir o custo de mão de obra, devem estar preparadas para uma queda na produtividade de forma direta [Ramasubbu 07]. Para mitigar este efeito - times dispersos - os autores sugerem como estratégia a adoção de um processo estruturado - com atividades e responsabilidades definidas. Os autores também apontam tamanho do time e reuso como variáveis que afetam a produtividade.

Alguns autores preferem focar que hoje precisamos diminuir o ciclo de desenvolvimento de software para agilizar o *time to market*, outros focam na redução do esforço ou precisão nas estimativas. Zhizhong Jiang e Peter Naudé [Jiang 07c], baseados no conceito de produtividade - razão de unidades produzidas por unidades de entrada, ou seja, tamanho do projeto/ tempo levado para completar o projeto de software - afirmam que a menos que possamos controlar o tamanho do projeto, o aumento da produtividade não necessariamente significa reduzir o tempo de desenvolvimento. Os autores apontam como o mais crítico e mais significativo dos fatores o tamanho do projeto, e citam outros fatores como tamanho da equipe, ambiente de desenvolvimento entre outros. Os dois principais objetivos do trabalho dos autores é identificar os fatores que influenciam na produtividade do desenvolvimento de software e o desenvolvimento de um modelo para prever o esforço com precisão aceitável.

Ruhe e Wagner [Ruhe 08] descrevem um procedimento sistemático para melhoria de produtividade em organizações de pesquisa e desenvolvimento. Assim como neste trabalho, onde os principais vilões da produtividade organizacional são mapeados para que

um conjunto específico de estratégias seja traçado, o modelo dos autores foca em alavancas específicas - a cada organização - para melhoria de produtividade. A proposta dos autores envolve quatro passos que podem ser iterativos, onde os resultados são sempre específicos para a organização observada.

No mesmo ano, os autores tiveram outro trabalho onde concluem - após uma revisão sistemática sobre produtividade - alguns pontos relevantes para ter em mente ao estabelecer estratégias, como: o tamanho do time não aumenta linearmente o custo do projeto; aumentar o tamanho do time em projetos que estão enfrentando atrasos no cronograma pode não ser uma boa idéia; e por fim, selecionar os membros do time baseados em sua experiência e tipo de personalidade pode ser outra forma para os gerentes de projeto controlar o custo no desenvolvimento de software [Wagner 08].

Pendharkar e Rodger [Pendharkar 09] é mais um trabalho onde o fator tamanho de time é explorado. Os autores afirmam que times pequenos, pela homogeneidade de seu background, podem não atender as habilidades necessárias para resolução dos problemas de projetos grandes e complexos. Os autores ao afirmarem que selecionar o tamanho apropriado para projetos de desenvolvimento de software continua sendo um grande desafio, também citam que Faraj [Faraj 00] afirma que grandes times representam uma melhor distribuição de habilidades, mas levam a um maior custo de comunicação e coordenação. Outro ponto relevante para este trabalho são estratégias relacionadas a ferramentas, onde os autores afirmam que ferramentas de desenvolvimento de projeto provêm capacidade avançadas de coordenação e comunicação.

Alguns trabalhos analisados agregam apenas pelo ponto de vista de reafirmar a importância de um determinado fator. Em alguns casos, nenhuma nova estratégia é sugerida para garantir ou diminuir o impacto desses fatores, talvez pela sua perspectiva apenas quantitativa ou ainda pelo foco em outros aspectos como medição de produtividade. Um exemplo é o estudo empírico de Morasca e Russo, que objetiva apoiar gerentes de projeto na avaliação e melhoria do processo de software [Morasca 01]. Os autores confirmaram que o tamanho do software está relacionado com a produtividade, bem como o tamanho do time e sua experiência, mas não sugerem como conseguir softwares pequenos ou times experientes.

Ao longo desta revisão, alguns fatores e estratégias foram citados relacionados com o bom gerenciamento, tamanho de times, ferramentas, reuso, motivação e sistemas de incentivo e recompensa. Modelos de motivação de modo geral têm se mostrado um fator de sucesso para projetos de software. Desde a década de 80 temos a falta de motivação como uma das causas mais citadas de falha no desenvolvimento de software [DeMarco 87].

Para este trabalho catalogamos Motivação e Recompensa de forma diferente. Em alguns casos, o segundo utilizado como estratégia para o primeiro. As estratégias relacionadas à motivação envolvem desde o formato da definição de uma tarefa - tarefas com objetivos claros, interessantes para os indivíduos, claramente definidas e seu envolvimento com atividades maiores claramente expostos [Sharp 09] - até sistemas de recompensa e incentivos. Os autores apontam entre os indicativos de que os engenheiros de software estão motivados: o baixo *turnover*; a melhoria na produtividade e no tempo de entrega do projeto; e a aderência do projeto ao seu orçamento.

Em um trabalho que discute a relação e os fatores que levam à satisfação e fidelidade do empregado é a performance organizacional, Silva e demais autores apontam algumas estratégias implementadas como críticas para a realização destes objetivos [Silva 03]. Entre elas: a descentralização de tomada de decisão; e a utilização de trabalho em equipe.

Para os autores o trabalho desafiador, recompensas, gerência participativa, entre outros como fatores que elevam o comprometimento, diminuem o *turnover* e impactam positivamente no desempenho organizacional.

Como gerenciamento de pessoas não é um problema apenas da indústria de software, o trabalho anteriormente citado, mesmo não sendo da indústria de software, se mostra consistente quanto às estratégias propostas nos demais trabalhos aqui citados.

Clinicy afirma que um sistema de recompensa deve encorajar comportamentos desejáveis de indivíduos e de grupos para obter melhoria de produtividade [Clinicy 03]. Incentivo tem sido apontado como fator que influencia na produtividade de software já a algum tempo [Boehm 82], [Boehm 87a]. Em alguns casos são apresentados o que os gerentes precisam ter em mente para projetar um sistema de incentivos. Segundo Zhuge, dependendo da estratégia organizacional pode ser melhor reconhecimento individual ou em grupo. Para organizações que querem incentivar a gestão e compartilhamento do conhecimento, a recompensa deve ser baseada na produtividade e no conhecimento compartilhado. Se medir for complicado é melhor adotar recompensa para o grupo, entretanto a gestão de incentivo em grupo é menos eficiente e pode levar a problemas na produtividade [Zhuge 08].

Para Welch [Welch 05], vê a recompensa como uma forma de diferenciação. Como as empresas dispõem de quantidade limitada de dinheiro e tempo, os líderes vencedores investem onde o retorno é maior. Para que isso seja possível é trabalhado o Modelo 20-70-10 (20% superiores, 70% intermediários e 10% inferiores). A diferenciação recompensa os membros da equipe que têm mais mérito, e demite um percentual saudável e previsto para renovação da organização.

Soares *et al.* afirmam que sistemas de recompensa e incentivo impactam na produtividade dos times. Os autores oferecem uma proposta de conjunto de *guidelines* para apoiar gerentes na definição de um sistema de recompensa como parte da estratégia organizacional de melhoria de produtividade [Soares 09b]. Segundo os autores, sistemas de recompensa têm sido menos explorados do que outras estratégias de melhoria de produtividade em projetos de software, apesar de terem sido amplamente discutidos e implementados em outras disciplinas.

3.6 CLASSIFICAÇÃO DOS FATORES E ESTRATÉGIAS

Buscando contribuir para uma visão mais abrangente das práticas e técnicas que podem trazer melhoria de produtividade, catalogamos neste trabalho os fatores mais abordados e estratégias a serem adotadas para diminuir ou elevar os efeitos desses fatores. Várias são as abordagens de classificação dos fatores. Ao longo dos anos, muitos autores propuseram a sua própria classificação de fatores de produtividade [Boehm 81, Vosburgh 84, Banker 87, Scacchi 89, Briand 98, Jones 00, Boehm 00a, Lokan 01, Clincy 03, Wagner 08]. Os autores analisam o impacto e / ou a capacidade do fator para aumentar a produtividade em cada classificação. A maioria deles usa algo em comum em sua classificação como projeto, produto e pessoas.

Boehm propôs uma das primeiras e mais importantes classificações dos fatores que influenciam na produtividade [Boehm 81]. Ele divide-os como:

- Atributos do produto: tais como tamanho da base de dados, complexidade do produto e tamanho do produto;
- Atributos do computador: tais como tempo de execução, restrições de armazenamento e controle de recursos;
- Atributos do projeto: tais como modernas práticas de programação, uso de ferramentas e restrições de cronograma
- Atributos de pessoal: tais como equipe, motivação e gestão.

Banker [Banker 87] classifica os fatores como pessoal, gerência de projeto, usuário e ambiente técnico. Scacchi separa os atributos que facilitam a alta produtividade, como: o ambiente de desenvolvimento, produto e equipe [Scacchi 89]. Briand [Briand 98] em seu método de estimar custos categoriza os *drivers* de “custo” em produto, processo, projeto e pessoa.

Já Lokan [Lokan 01] classificou os fatores em “*hard*” e “*soft*”, o que já havia sido apresentado por Boehm *et al*[Boehm 00a]. Semelhante a esta classificação Wagner e Ruhe [Wagner 08] dividiram os fatores como técnicos e não técnicos. Os fatores técnicos foram agrupados em três categorias: produto; processo; e ambiente de desenvolvimento. Os fatores não técnicos, tidos pelos autores como *Soft* são categorizados em: cultura corporativa; cultura do time; capacidade e experiência; ambiente e projeto.

3.6.1 Classificação Adotada

Inicialmente adotamos a classificação de fatores em técnicos e *soft*, entretanto, para facilitar na visualização e análise das estratégias foi necessária uma nova classificação¹. As Figuras 3.4, 3.5 e 3.6 mostram o primeiro agrupamento realizado neste trabalho.

O levantamento inicial contava com vinte e oito fatores, já adotando algumas generalizações devido a diversidade de seu conteúdo, como por exemplo, o fator Processo ora é citado como boas práticas, ora como métodos² ou ainda como metodologia³ de desenvolvimento. Devido a relação entre alguns fatores, uma segunda generalização foi realizada após o término da revisão bibliográfica.

É sabido que para muitos dos fatores existem uma intersecção entre as categorias. Como, por exemplo, participação do cliente que poderia ser classificado como algo referente ao projeto ou ainda como um fator pessoal, porém para fins desta pesquisa, esse fator foi classificado como pessoa; e reuso que poderia ser visto como práticas organizacionais, mas foi mapeado como característica de um produto. Assim, as seguintes categorias foram adotadas:

- **Produto** - Esta classificação está relacionada com as características inerentes ao produto, como os requisitos funcionais e não funcionais, o tamanho do software, sua complexidade e reuso.
- **Pessoas** - Esta classificação endereça fatores relacionados ao capital humano envolvido no ciclo de vida de desenvolvimento de software, abrangendo também a gerência e o cliente. Exemplos de tais fatores são: motivação da equipe, experiência, o conhecimento da equipe com o negócio, qualidade no gerenciamento e experiência e participação do cliente. Esta categoria tem uma grande influência da gerência e organização.

¹Baseada na classificação de Boehm [Boehm 81]

²Caminho ou processo racional para atingir um dado fim(Houaiss

³Estudo científico dos métodos (Michaelis). A explicação detalhada de toda ação desenvolvida no método.

	T e a m u a n p h e o d a	A d m e s i e n v e l v e m e n t o	U s e r a t e m e n t a s	Q u o n c a u t m i e n d a t a e ç ã o e	L i r n o g u r a a m ç ã o d e	T a o m f a t n w h a r e d o	C o o f m p t e s c o n o r e d o	D e o f s e s c o n o r e d o	C o a m p l e x i d e a d e
walston-felix:1977	x					x	x	x	
albrecht:1979					x	x			
bailey-basili-1981	x			x	x		x		
boehm-1981-software-economics		x	x	x	x	x	x		
boehm-1982-trw		x	x						
behrens:1983					x	x			
vosburgh-et-al-1984-productivity-factors		x				x	x	x	
boehm-1987-improving		x	x		x	x	x		
banker-kemerer:1989	x		x		x				
duka-1988			x	x	x	x	x		x
banker-kemerer:1989			x			x			
scacchi-1989	x	x	x		x		x		
yu-smith-huang:1991		x					x		
banker-et-al-model:1991		x							
banker-reuse:1991	x	x	x		x		x		
gaffney:1992						x			
lim-reuse:1994	x			x	x		x		
dumaine-1994			x			x			
brooks-mythical	x		x		x	x			
scacchi-understanding-software:1994	x		x		x	x	x	x	
bruckhaus-et-al-tools-1996			x						
briand:1998				x			x		x
boehm:2000			x	x		x	x		x
maxwell-2001-benchmarking		x	x		x		x		
jones:2000		x	x		x	x	x		
hantos-gisbert-2000-software-productivity	x								
cockburn-highsmith-people-factor:2001	x								
morasca-russo:2001	x								
lokan:2001					x				
reifer:2002			x						
kappeler:2002	x								
anselmo-ledgard-2003-measuring-productivity		x					x		
clincy:2003	x		x	x					
groth:2004			x				x		
chiang-mookerjee-2004-team-productivity	x		x						
mcconnell-code:200	x		x		x	x	x		
premaraj-et-al:2005						x			
Simmons:2007	x			x	x	x	x		
jiang-naude-craig-variation:2007	x				x				
jiang-comstock-the-factors:2007	x		x		x				
jiang-examination:2007	x		x		x	x			
jiang-investigation:2008	x		x		x				
pendharkar-rodger:2009	x		x						

Figura 3.4. Fatores técnicos analisados

Figura 3.5. Fatores soft analisados (Parte 1)

Ano	Fator\Estudo	D u r a ç ã o	p r o j e t o	C r o n o g r a m a	P r o p o s t u l a d o	M e t o d o l o g i c o	A b o r e s t a b i l i z a d o	/E s t a b i l i z a d o	E l e m e n t o s	C o n t e n t o	L e t e r a s	O r t o g r a f i a	P u n t u a r i a d o	R e f e r e n c i a	/C o n t e n t o	Q u e s t i c i o n á r i o	E x p l i c i t a ç ã o	/E x p l i c i t a ç ã o	E x p l i c i t a ç ã o	I n t e r p r e t a ç ã o	A p l i c a ç ã o	T e m p o	R e s u l t a d o	Q u a l i d a d e			
1999	demarco-lister-1999				x	x				x							x										
1999	Boehm-1999																									x	
2000	maxwell-forselius-2000			x				x					x					x				x					
2000	jones-2000			x	x	x			x	x	x	x			x			x								x	
2000	boehm-et-al-2000											x						x		x	x	x	x	x			
2000	faraj-sproull-2000																x		x		x	x					
2000	hantos-gisbert-2000				x										x												
2001	morasca-russo-2001																		x								
2001	lokan-2001							x											x		x	x					
2001	cockburn-highsmith-2001			x											x				x		x	x					
2002	refeir-2002			x		x																				x	
2002	kappeler-2002				x												x								x		
2003	boehm-turner-2003																										
2003	silva-ramos-campos-2003				x																				x		
2003	anselmo-ledgard-2003			x																						x	
2003	clincy-2003			x					x	x	x	x	x	x	x	x	x	x								x	
2004	groth-2004			x																							
2004	chiang-mookerjee-2004			x														x		x							
2004	kitchenham-2004																									x	
2005	mcconnell-2005			x																							
2005	premraj-2005																		x								
2007	simmons-2007			x				x						x		x	x	x							x	x	x
2007	ramasubbu-balan-2007			x						x																x	
2007	jiang-naude-2007			x																							
2007	jiang-comstock-2007			x																							
2007	jiang-naude-comstock-2007			x																							
2008	sharp-et-al-2008				x										x											x	
2008	zhuge-2008				x												x										
2008	wagner-ruhe-2008																										
2009	furtado-aquino-2009				x												x			x							
2009	pendharkar-rodger-2009																	x									

Figura 3.6. Fatores soft analisados (Parte 2)

- **Projeto** - Esta classificação envolve fatores relacionados com o projeto ou regras e políticas organizacionais. Alguns exemplos desta categoria são ambientes de trabalho, cultura organizacional, apoio da alta gestão, sistemas de incentivo e recompensa, tamanho da equipe e instabilidade dos requisitos.

3.6.2 Seleção e Qualificação dos Fatores

Para definição de quais fatores seriam trabalhados, quanto às estratégias de melhoria de produtividade, a quantidade de citações gerais foi considerada, bem como a quantidade de citações específicas da década mais recente. Fatores com mais de quinze citações, ou ainda com pelo menos seis citações na última década, foram estudados e estratégias para maximizar os resultados desses fatores foram traçadas. A Tabela 3.1 mostra o agrupamento por década do conjunto inicial de fatores.

Por fim, baseado nos critérios já definidos, chegamos aos quatorze fatores (e generalização de fatores) que serão tratados no capítulo das estratégias. Os fatores mais relevantes por número de citações são apresentados na Tabela 3.2.

3.7 CONSIDERAÇÕES FINAIS

Os primeiros estudos realizados sobre produtividade foram publicados relacionados à economia do software, onde as estratégias para redução de custos, são também estratégias para melhoria de produtividade. Ao passar dos anos, o conceito produtividade foi sendo explorado e tratado separadamente.

Ao longo deste capítulo foram apresentados os principais trabalhos relacionados a fatores de influência, estratégias de melhoria na produtividade no desenvolvimento de software. Foi possível observar diferentes análises da evolução da produtividade, e a ênfase que se tem dado a estratégias para sua melhoria. O próximo capítulo descreve as estratégias encontradas para cada fator de produtividade tido como mais relevante. O critério de relevância dos fatores foi dado pelo número de citações encontradas na revisão bibliográfica das últimas três décadas.

Tabela 3.1. Fatores observados na literatura.

	1970-1979	1980-1989	1990-1999	2000-2009	Total de Citações
Duração do Projeto	1	3	2	0	6
Restrição de Cronograma	0	7	3	5	15
Processo	2	9	6	16	33
Motivação	0	5	0	7	12
Amb. Trabalho/Espaço Físico	0	3	0	4	7
Estabilidade dos Requisitos	1	6	5	3	15
Estrutura e Clima Organizacional	0	0	0	2	2
Localização da Equipe	0	0	1	4	5
Organização da Equipe	0	3	1	3	7
Participação e Experiência do Cliente	1	2	1	6	10
Recompensa/Incentivos Pessoais	0	4	2	6	12
Comunicação	0	2	1	6	9
Qualidade no Gerenciamento	0	6	1	8	15
Habilidades e Experiência da Equipe	1	9	6	12	28
Experiência na Linguagem	1	3	1	4	9
Experiência na Aplicação	1	5	2	5	13
Continuidade da Equipe/Turnover	0	1	0	5	6
Reuso	1	6	9	8	24
Ambiente de Desenvolvimento	0	7	3	3	13
Ferramentas	0	8	5	12	25
Linguagem de Programação	1	7	5	9	22
Tamanho do Produto	2	6	4	6	18
Tamanho da Equipe	1	3	5	13	22
Complexidade do Software	1	7	6	7	21
Quantidade de Documentação	0	3	2	3	8
Tamanho e Complexidade da base de dados	0	1	1	1	3
Desenvolvimento de Software Concorrente com Hardware	1	1	1	0	3
Qualidade	0	2	2	2	6

Tabela 3.2. Fatores considerados mais relevantes

	1970- 1979	1980- 1989	1990- 1999	2000- 2009	Total de Citações
Restrição de Cronograma	0	7	3	5	15
Processo	2	8	6	16	32
Motivação	0	5	0	7	12
Estabilidade dos Requisitos	1	6	5	3	15
Participação e Experiência do Cliente	1	2	1	6	10
Recompensa/Incentivos Pessoais	0	4	2	6	12
Comunicação	0	2	1	6	9
Qualidade no Gerenciamento	0	6	1	8	15
Habilidades e Experiência da Equipe	1	8	6	12	27
Reuso	1	6	9	8	24
Ferramentas	0	8	5	12	25
Linguagem de Programação	1	7	5	9	22
Tamanho do Produto	2	6	4	6	18
Tamanho da Equipe	1	3	5	13	22
Complexidade do Software	1	7	6	7	21

CATALOGAÇÃO DE FATORES E ESTRATÉGIAS

Este capítulo descreve de forma consolidada as estratégias para endereçar os fatores mais citados na literatura ao longo dos anos, com o objetivo de apoiar as empresas de desenvolvimento de software em maximizar os fatores positivos e minimizar ou evitar os de influência negativa com o “correto” conjunto de estratégias.

....

4.1 INTRODUÇÃO

Existem várias estratégias para melhorar a produtividade que são objeto de estudo tanto da academia quanto da indústria de desenvolvimento de software. Alguns estudos abordam apenas um fator e suas estratégias, tais como: Boehm [Boehm 99] que cita armadilhas a evitar na adoção de reuso de software e estratégias para utilizá-lo com sucesso; Bruckhaus *et al.* [Bruckhaus 96] descreve como ferramentas podem melhorar a produtividade no processo de desenvolvimento de software e cita critérios de quando adotá-las; Chiang e Mookerjee [Chiang 04] contribuem com estratégias de qualidade no gerenciamento através de coordenação de times, juntamente com [Boehm 81, Hantos 00].

Alguns autores tratam de estratégias para manter e ainda aumentar a motivação de equipes de desenvolvimento de software. Sharp *et al.* [Sharp 09] apresenta um modelo de motivação; [Soares 09b, Zhuge 08] defendem estratégias de recompensas e incentivo, e apontam como fazê-las funcionar.

Para ser capaz de reduzir o *time-to-market* e melhorar a produtividade, medições precisas do processo de desenvolvimento de software devem ser realizadas, e fatores que influenciam na produtividade devem ser identificados e compreendidos [Yu 91]. Corroborando e estendendo a visão apresentada pelo autor, cada organização de software deve olhar para seus próprios fatores de produtividade e suas próprias fraquezas para decidir quais estratégias trarão o maior ganho. As características organizacionais e de cada projeto, grau de maturidade e foco de negócio representam realidades que definem uma maior influência de determinados fatores pedindo diferentes focos de melhoria, diferentes estratégias.

Não existe nenhuma solução em definitivo que resolva todos os problemas de produtividade no desenvolvimento de software [Boehm 87a], embora existam muitos métodos, melhores práticas e estratégias disponíveis que sugerem que podem trazer melhoria de produtividade.

Boehm [Boehm 82] relata que obter ganhos significativos de produtividade requerem iniciativas integradas em diversas áreas, por exemplo, melhoria em ferramentas, metodologias, ambiente de trabalho, educação, gerenciamento, incentivos pessoais, reuso em software, dentre diversos outros fatores. Isso quer dizer que para avaliar o quanto os projetos produzem de valor agregado por unidade de valor consumido é necessário entender quais são as várias dimensões que precisam ser consideradas na análise final do desempenho organizacional.

Mesmo que exista alguma verdade na afirmativa “O problema não é mais suscetível a soluções fáceis, porque todas as soluções fáceis foram pensadas e aplicadas há muito tempo.” [DeMarco 87], objetivamos fornecer mais uma ferramenta para que gerentes continuem buscando melhoria de produtividade de forma integrada. Apoiá-los na decisão de qual grupo de estratégias trará o maior ganho e executá-las em paralelo, dentro de um programa de melhoria, como sugerido por alguns estudos [Vosburgh 84, Boehm 82].

Por esta razão, as estratégias e referências foram agrupadas por fatores para facilitar na visualização e escolha de quais ações serão escolhidas para os fatores que os gerentes desejam endereçar. Embora seja importante ter em mente que endereçar todos os fatores ao mesmo tempo nem sempre é possível ou desejável [Scacchi 95], sugerimos a abordagem de Pareto onde 20% das estratégias trarão 80% dos benefícios.

Apenas os fatores mais citados foram levados em consideração. Para chegar aos fatores mais citados, cerca de 90 artigos sobre produtividade foram lidos e analisados. Todos os trabalhos que citavam fatores, desta amostragem, foram contabilizados. Apenas citações diretas foram contabilizadas. Algumas generalizações ou uniformizações foram necessárias devido à nomenclatura utilizada por pesquisadores e ainda pela similaridade dos fatores, tais como: experiência, conhecimentos e habilidades da equipe para cobrir citações referentes à experiência do time com a linguagem de programação ou a experiência com a aplicação e suas habilidades em geral; processo, combina metodologia, que alguns autores chamam de MPP(modernas práticas de programação), técnicas de desenvolvimento, ciclo de vida entre outros; e participação do cliente envolve a participação do cliente e usuário, sua experiência e ainda a interface com o cliente - que se refere à dinâmica de troca de informações com o cliente.

4.2 ABORDAGEM DA DOCUMENTAÇÃO ADOTADA

A abordagem adotada foi documentar as estratégias em um formato conhecido pela comunidade do desenvolvimento de software, seguindo ao máximo a proposta de oferecer uma solução para um problema num determinado contexto. Segundo Gamma, Helm, Johnson e Vlissides [Gamma 95] se documentados em um formato estruturado, descrevendo o problema e a solução, e recebendo um nome, podem ser chamados de padrões. Padrão de projeto pode então ser definido como:

“Padrões usualmente não contêm novas idéias. A justificativa para padrões não é descobrir e expressar novos princípios de engenharia de software. Na verdade, trata-se exatamente do oposto - padrões tentam codificar o conhecimento, os idiomas e os princípios existentes, quanto mais apurado e amplamente usado melhor”[Larman 00].

Para Craig Larman [Larman 00], padrão é uma descrição nomeada de um problema/solução, que pode ser utilizado em novos contextos, com conselhos sobre como utilizá-lo em novas situações. Em termos de idéias, ele fornece aconselhamento sobre como utilizá-lo em circunstâncias variáveis.

Em geral um padrão tem quatro elementos essenciais [Gamma 95]:

- **O nome do padrão:** é uma referência que podemos usar para descrever o problema de projeto, suas soluções e conseqüências, em uma ou duas palavras;
- **O problema:** descreve em que situação aplicar o padrão, ele explica o problema e seu contexto. Pode descrever problemas de projetos específicos, algumas vezes o problema incluirá uma lista de condições que devem ser satisfeitas para que faça sentido aplicar o padrão;
- **A solução:** descreve os elementos que compõem o padrão de projeto, seus relacionamentos, suas responsabilidades e colaborações. A solução não descreve um projeto concreto ou uma implementação em particular porque um padrão é um gabarito que pode ser aplicado em muitas situações diferentes;
- **As conseqüências:** são os resultados e análises das vantagens e desvantagens (*trade-offs*) da aplicação do padrão. Elas são críticas para a avaliação de alternativas de projeto, e a compreensão de custos e benefícios da aplicação do padrão.

Mesmo não estando formalmente em um formato de padrões, a estrutura adotada para documentação das estratégias se assemelha a um padrão. Um das diferenças é

o fato de que além das estratégias coletadas - como resultados da revisão bibliográfica sobre como endereçar os fatores que influenciam na produtividade - estratégias oriundas do experimento exploratório e das entrevistas também foram documentadas. A segunda diferença é que não foi dado nome para as estratégias. Como são muitas as estratégias identificadas por fator de influência na produtividade, elas foram agrupadas e identificadas apenas pelo fator. Os elementos adotados para descrição das estratégias foram:

- Fator de produtividade associado;
- Descrição do fator: descrição conceitual do fator associado às estratégias;
- Problema associado: o impacto na produtividade, nos custos do projeto e em outros fatores.
- Motivação: motivação para adoção das estratégias e um pouco de contexto em relação ao estudo bibliográfico realizado;
- Estratégias: soluções encontradas para tratar o problema citado.

4.3 CATÁLOGO DE ESTRATÉGIAS

Conforme explicado no Capítulo 3, para facilitar na visualização, os fatores de influência na produtividade mais relevantes e suas estratégias foram divididos em três categorias, conforme mostra a Figura 4.1. As estratégias identificadas na literatura foram documentadas por categoria e fator, e são apresentadas a seguir.

4.3.1 Fatores e Estratégias Relacionados ao Produto

4.3.1.1 Fator: Complexidade de Software

Descrição: Para o glossário do IEEE, complexidade de software é o grau em que o sistema ou componente tem um projeto ou implementação que é difícil de entender e verificar.

Problema: Complexidade gera uma queda de eficiência, o que resulta em uma baixa produtividade e aumento nos custos [Boehm 81, Vosburgh 84, Scacchi 95, Jones 00, Blackburn 96, Chiang 04, Maxwell 00, Simmons 07]. Da complexidade vem a dificuldade de comunicação, que leva a deficiências no produto, aumento dos custos, atrasos no cronograma [Brooks 95]. Por fim, a complexidade demanda um maior esforço de coordenação [Chiang 04].



Figura 4.1. Classificação dos Fatores

Motivação: Nem toda complexidade é inevitável [Brooks 95]. Controlar a complexidade de software é o assunto técnico mais importante do desenvolvimento de software [McConnell 04]. Mesmo que a complexidade do código esteja fora de controle do projeto [Boehm 81, Boehm 87a], existem várias tentações a serem evitadas. Projetos falham devido a requisitos pobremente especificados, planejamento mal feito ou gerenciamento deficiente, entretanto quando um projeto falha por questões técnicas, quase sempre é resultado de uma complexidade descontrolada [McConnell 04]. Projetos de software demasiadamente dispendiosos e ineficazes surgem de três fontes [McConnell 04]:

- Uma solução complexa para um problema simples;
- Uma solução simples e incorreta para um problema complexo;
- Uma solução inadequada e complexa para um problema complexo.

“O software moderno é inerentemente complexo e não importa o quanto você tente, finalmente acabará encontrando algum nível de complexidade inerente ao problema do mundo real em si” [McConnell 04].

“Atualmente a complexidade do software é tanta que novos desenvolvedores têm uma curva de aprendizado mais íngreme, quando contratados”(Djenana Campara, Gerente de Tecnologia da Klocwork, [Groth 04])

Estratégias:

- Minimize a quantidade de complexidade essencial¹ com que o time deve lidar [Brooks 95, McConnell 04];
- Impeça que a complexidade acidental² prolifere desnecessariamente [Brooks 95, McConnell 04];
- Divida o sistema em subsistemas [McConnell 04]. Os seres humanos têm mais facilidade em compreender várias informações simples do que uma única informação complexa [McConnell 04];
- Projete o software com os programadores de manutenção em mente [McConnell 04];
- Trabalhe com projetos simples ([Highsmith 01a, Cockburn 01]), evite fazer projetos muito engenhosos, prefira os simples e fáceis de entender [McConnell 04];
- Quando possível trabalhe com código simples, claro, bem estruturado e documentado;
- Adote reuso, reuso ajuda a reduzir a complexidade do software [Butler 97, Simmons 07, Boehm 81, Anselmo 03];
- Busque software de baixo acoplamento, isso significa projetar software de modo a minimizar as conexões entre as diferentes partes de um programa [McConnell 04];
- Trabalhe com componentes de alta coesão - componentes logicamente relacionados;
- Trabalhe com interfaces o mais simples e claras possível para reduzir a complexidade dos componentes e módulos [Brooks 95];
- Adote ferramentas de medição da complexidade, como ferramentas de análise de complexidade³, para apoiar na visibilidade da qualidade do código que está sendo

¹Propriedades essenciais são aquelas que alguma coisa deve ter para ser o que é (Brooks *Apud* [McConnell 04]. McConnell apresenta como exemplo, que o carro deve ter um motor, rodas e portas para ser um carro. Se ele não tiver nenhuma dessas propriedades essenciais, não será realmente um carro. Todo produto de software terá requisitos funcionais e não funcionais, mas é preciso focar apenas no mínimo necessário para atender aos objetivos do projeto.

²Propriedades acidentais são aquelas que se pode ter, mas não se relacionam com o fato de ser o que é. Um carro poderia ter motor V8, 4 cilindros turbo ou algum outro tipo de motor e ainda assim ser um carro, independentemente desse detalhe [McConnell 04]. Requisitos apenas desejáveis, que não fazem parte dos 20% mais relevantes, ou que não estão em alinhamento com os objetivos do projeto só aumentam a complexidade e tamanho do projeto.

³Calcula a quantidade de caminhos linearmente independentes em um módulo de um programa [McCabe 76]

gerado, tais como Metrics Eclipse Plugin⁴ e FxCop⁵;

- Minimize o número de parâmetros passados, ou ainda mais importante passe apenas os parâmetros necessários [McConnell 04].

4.3.1.2 Fator: Reuso

Descrição: Reuso é conhecido pelo uso de produto de software (código, componente, especificação, entre outros) já existente para um novo desenvolvimento. Reuso se aplica não apenas a fragmentos de código, mas também a produtos intermediários gerados no desenvolvimento de software, incluindo documentação de requisitos, especificações do sistema e qualquer informação que desenvolvedores precisem para criar o software [Butler 97].

Problema: Fator que tem uma influência positiva na produtividade, por isso o problema está associado a não adoção desse fator. Ou seja, recriação de produtos de software a cada novo desenvolvimento, mesmo com a afirmação de tantos autores quanto a influência positiva do reuso na produtividade [Boehm 81, Boehm 82, Boehm 87a, Banker 91, Gaffney Jr 92, Scacchi 95, Basili 96, Blackburn 96, Jones 00, Reifer 02, Clincy 03, Kitchenham 04].

Motivação: A forma mais rápida de desenvolver software é não desenvolver e sim reusar software que já funciona [Simmons 07]. A produtividade aumenta porque consumidores de produtos de trabalho reusáveis precisam realizar menos trabalho e o ciclo de vida requer menos *input* para gerar os mesmos *outputs*. Fora isso permite a redução do esforço de manutenção [Boehm 00b, Lim 94]. Em um longo prazo os maiores ganhos de produtividade virão do crescente uso de software já existente [Boehm 82, Anselmo 03, Simmons 07, Butler 97].

Estratégias:

- Adote reuso para todos os produtos intermediários gerados durante o processo de desenvolvimento de software, incluindo requisitos, especificação e documentação de sistemas e qualquer outra informação que o desenvolvedor precise para desenvolver software [Boehm 87a, Anselmo 03];
- Adote reuso em linhas de produto, essa estratégia pode trazer alto percentual de reuso. Identifique em um domínio específico o que pode ser reutilizado e realize uma análise de negócio para determinar o escopo correto. Uma arquitetura genérica é

⁴<http://metrics.sourceforge.net/>

⁵<http://msdn.microsoft.com/en-us/library/bb429476.aspx>

construída para esta família de produtos para estabelecer um papel bem definido para cada componente [Boehm 99, Anselmo 03];

- Adote reuso para balancear seu time, reuso permite que os mais experientes criem componentes para os com menos experiência usarem [Boehm 00b];
- Não espere até que tenha um repositório de componentes para adotar o reuso [Boehm 99];
- Defina uma arquitetura de domínio com especificações de interface apropriadas para seus componentes [Boehm 99].

4.3.1.3 Fator: Tamanho do Produto

Descrição: Tamanho do produto de software pode ser conhecido como número de linhas de código, pontos de função [Albrecht 79] ou pontos de caso de uso [Karner 93] que compõem um sistema.

Problema: Grandes produtos de software possuem mais erros, maior complexidade, demandam maiores times e maior esforço de gerência. A Figura 4.2 mostra que o esforço do projeto cresce exponencialmente com o crescimento do produto [McConnell 04]. A produtividade do desenvolvimento de software tende a cair com o aumento do tamanho do produto [Albrecht 79, Boehm 81, Behrens 83, Vosburgh 84, Premraj 05, Jiang 07c].

Motivação: Tanto a quantidade quanto o tipo dos erros são afetados pelo tamanho do produto, em diferentes proporções. É esperado que um projeto duas vezes maior que outro tenha duas vezes mais erros, entretanto a métrica de densidade dos defeitos - número de defeitos por mil linhas de código - mostra que o número de defeito cresce ainda mais [McConnell 04]. Além disso, a complexidade do produto, o tamanho e organização da equipe, normalmente crescem com o tamanho do produto e se tornam uma maior influência na produtividade [McConnell 04]. Entretanto como todos os projetos possuem um custo fixo de gerenciamento, a produtividade do projeto aumenta inicialmente com o aumento do tamanho do produto pela distribuição do custo de *overhead*, ou ainda devido ao uso de pessoas, ferramentas mais especializadas e possivelmente por maior atenção gerencial [Banker 89].

Estratégias:

- Tente fazer pequenos projetos a partir dos grandes projetos. “A melhor maneira de evitar a “não-economia de escala” é reduzir a escala” [Boehm 81];

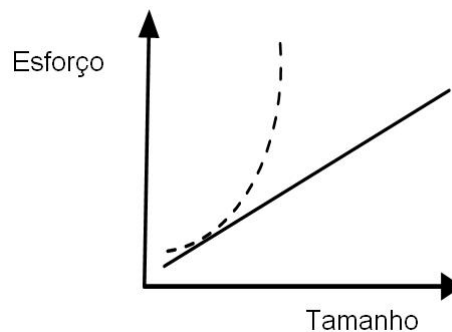


Figura 4.2. Relacionamento de tamanho do projeto com esforço [McConnell 04]

- Se o projeto for grande, identifique módulos, componentes, subsistemas e tente deixá-lo o mais simples possível. Mantenha em mente a necessidade de baixo acoplamento [McConnell 04]. Alto acoplamento aumenta a complexidade do software e a necessidade de desenvolvedores mais experientes;
- Procure projetar o software para que suporte tarefas separadas que permitam paralelismo [Simmons 07];
- Adote o “peso certo” de processo ou metodologia [McConnell 04]. Inicie pequeno e expanda o processo para projetos grandes, através de inclusões de métodos e reduza para projetos menores [Boehm 03];
- Evite codificação desnecessária, elas resultam em redução do esforço e menos frustração de ver o fruto de seu trabalho não ser utilizado [Boehm 87a, Boehm 81];
- Foque nos requisitos essenciais [McConnell 04] e evite o *gold-plating* [Boehm 81, Boehm 87a];
- Adote práticas disciplinadas de codificação, como inspeções, padrões de codificação e um bom suporte de ferramentas. Essas técnicas são muito valiosas para projetos pequenos e essenciais para projetos grandes [McConnell 04].

4.3.2 Fatores e Estratégias Relacionados ao Projeto

4.3.2.1 Fator: Sistemas de Incentivos e Recompensas

Descrição: Recompensas podem ser classificadas como tangíveis ou intangíveis. No primeiro caso, são definidas como sendo prêmios concedidos aos empregados em função de

tarefas realizadas, que vão ao encontro ou superam expectativas inicialmente estabelecidas [Soares 09a]. Para Stajkovic e Luthans(apud [Soares 09a]), no segundo caso, são definidas como elogios concedidos em público, em virtude de realizações amplamente aprovadas no contexto da cultura organizacional.

Problema: Equipes sem incentivos e pouco motivadas. Frustrações e queda na produtividade devido à falta de incentivos ou reconhecimento semelhante para pessoas com produtividades diferentes. Este fator tem uma influência positiva na produtividade, assim o problema está em não adotá-lo ou adotá-lo de forma errada.

Motivação: Um sistema de recompensa é um fator chave para melhorar a produtividade de uma equipe [Clincy 03]. Estruturas organizacionais pobres de recompensa, como recompensar um dos melhores com 6% e os medíocres com 5%, levando a frustrações dos bons executores [Boehm 87a]. Reconhecimento trás motivação [Hantos 00, Clincy 03, Soares 09b, Zhuge 08, Dumaine 94].

Estratégias:

- Defina sistemas de recompensa para aumentar a produtividade organizacional, recompensando as pessoas que atinjam um nível esperado de desempenho [Soares 09a];
- Recompensas podem ser financeiras ou não financeiras [Clincy 03]. Adote distribuição de bônus, promoções, viagens, cursos especiais em programas de incentivos dos melhores executores [Boehm 87a];
- Recompense por alta qualidade e um bom trabalho baseado em critérios objetivos [Sharp 09];
- Recompensas devem ser distribuídas não apenas para o integrante que se destacou, mas para o grupo que alcançou os objetivos de forma a encorajar o trabalho em equipe [Dumaine 94]. Assim como em estratégia de vendas, onde vendedores têm sua bonificação pelo seu resultado à parte e também quando todo o grupo atinge a seu objetivo;
- Dentre os fatores a serem analisados pelos sistemas de incentivo, considere também as circunstâncias do projeto [Clincy 03];
- Recompensas também devem ser consideradas para projetos de pesquisa e novas tecnologias [Clincy 03].

4.3.2.2 Fator: Comunicação

Descrição: O grau e eficiência com que a informação flui dentro do time [Wagner 08].

Problema: Comunicação formal, excessiva, com muitas interrupções e pouco eficiente. Quando a comunicação aumenta, a produtividade máxima do projeto diminui [Brooks 95]. As interrupções, oriundas do aumento da comunicação, causam queda na produtividade [Simmons 07].

Motivação: “Quando a comunicação falha, o trabalho pára” [Brooks 95]. Comunicação entre a equipe do projeto é vital [Simmons 07]. Segundo Brooks [Brooks 78] à medida que o número de pessoas envolvidas no projeto aumenta, o número de possíveis caminhos de comunicação também aumenta de maneira multiplicativa. O número de caminhos de comunicação aumenta proporcionalmente ao quadrado do número de pessoas pertencentes à equipe [McConnell 04], conforme pode ser observado na Figura 4.3.

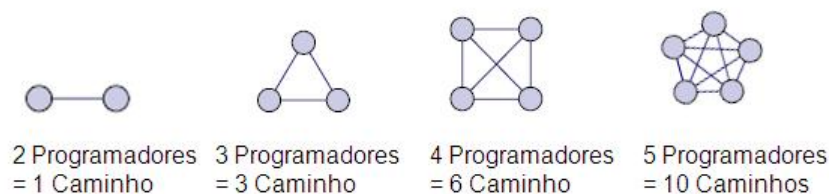


Figura 4.3. Caminhos de comunicação - adaptado de [McConnell 04]

Estratégias:

- Faça todo o possível para garantir uma comunicação eficiente dentro da equipe, adote ferramentas como: mensagem de voz e-mail, conferência eletrônica e outras ferramentas de comunicação [Simmons 07], [Pendharkar 09];
- Prime pela comunicação mais frequente, regular e mais informal [Brooks 95]. Comunicação formal requer tempo de elaboração de documentação, e-mail e repasse de informações [Clincy 03];
- Tente estabelecer um horário comum para as equipes para facilitar que todos estejam produzindo em um mesmo momento. Várias entradas de integrantes da equipe ao longo do dia geram seguidas interrupções, que gera queda de produtividade;
- Comunicação em grandes equipes consome muito tempo, já que cada indivíduo é consumido na comunicação com outros membros da equipe para atualização de informações comuns, repasse de erros ou resolvendo problemas de recursos;

- Monte uma equipe de trabalho mais auto-suficiente, com analistas e desenvolvedores, garantindo uma comunicação informal e eficiente [Clincy 03]. Se os grupos não podem efetivamente formar uma única equipe, aproximá-los fisicamente. Sentar na mesma mesa se possível, facilitando a comunicação informal;
- Documente um diário de bordo com registros diários de desenvolvimento do software desde o seu princípio [Brooks 95];
- Mantenha o time pequeno para evitar os vários canais de comunicação [Brooks 78].

4.3.2.3 Fator: Restrição do Cronograma

Descrição: Cronogramas⁶ inadequados ao projeto ou com restrições.

Problema: Produtividade degradada devido a cronograma não realista. Total de silusão e falta de entusiasmo dos membros da equipe [Boehm 81].

Motivação: Simmons [Simmons 07] afirma que uma tentativa de alcançar um cronograma não real resulta em projetos ineficientes que não têm nenhuma esperança de serem concluídos dentro do cronograma.

“Em sua maioria, projetos de software falharam mais por falta de tempo no calendário do que em função da combinação de todas as outras causas...elas refletem uma premissa não verbalizada e completamente falsa que é a de que tudo correrá bem.” [Brooks 95]

Estratégias:

- Se um cronograma subestimado é requerido certifique que ele vale a pena. Uma redução de 25% no cronograma nominal equivale a um custo adicional de 23% [Boehm 81];
- Se o tamanho e a natureza do trabalho aumentar, altere também o cronograma [Boehm 81];
- Acompanhe seu caminho crítico bem de perto [Boehm 81].

⁶Instrumento de planejamento e controle de compromissos e atividades do projeto a serem executadas durante um período estimado.

4.3.2.4 Fator: Instabilidade dos Requisitos

Descrição: Número de mudanças dos requisitos ao longo do projeto.

Problema: Volatilidade dos requisitos pode resultar em uma grande quantidade de esforço gasto para manter as atualizações dos requisitos [Simmons 07]. A priorização de requisitos desnecessários, pouco importantes ou que não agregam grande valor ao cliente levam à eterna mudança [Boehm 81, Boehm 87a, Maxwell 03]. Requisitos definidos em um tempo distante não são mais reais e tendem a mudar, bem como os que são levantados precariamente, sem clareza e riqueza de detalhes. Por fim, equipes que não conhecem ou que não compreendem os objetivos do projeto tendem a inserir requisitos desnecessários.

Motivação: Estabilidade dos requisitos é uma importante e negligenciada fonte de economia e controle de custos [Boehm 88]. Para minimizar este problema é fundamental focar nos requisitos que trazem maior valor agregado ao cliente, já que “Em torno de dois terços das características de um software típico são raramente ou nunca utilizados. Apenas 20% das características são geralmente utilizadas” [Maxwell 03]. Ainda precisamos considerar a afirmação dos autores Highsmith e Cockburn [Highsmith 01a] que colocam que como não podemos eliminar as mudanças, diminuir os custos de responder a estas mudanças é a única estratégia viável.

Estratégias:

- Foque nos 20% dos requisitos que de fato agregam valor ao cliente e que serão utilizados. Eliminar características extras que ninguém realmente precisa, representa a única e maior oportunidade de melhorar a produtividade no desenvolvimento de software na maioria das organizações [Maxwell 03];
- Use o desenvolvimento incremental [Anselmo 03, Boehm 88] para controlar melhor as mudanças nos requisitos [Boehm 81, Highsmith 01a, Cockburn 01, McConnell 04, Boehm 03];
- Garanta que o time obteve um bom entendimento dos requisitos [Boehm 87a, Clincy 03] e mantenha o time focado nos objetivos do projeto [Clincy 03];
- Promova workshops entre analistas e desenvolvedores para apoiar na distribuição do conhecimento e no bom entendimento dos objetivos do projeto;
- Promova a primeira entrega em semanas, para garantir ganho e *feedback* rápidos [Highsmith 01a];

- Evite o entregar ao cliente o que ele não pediu. Ciclos de vida iterativos e prototipação ajudam neste sentido. Prototipação melhora a produtividade pela construção de produtos mais simples [Boehm 81, Boehm 87a];
- Adote processos que permitam o trabalho em cima apenas do escopo de maior valor agregado ao cliente, ao em vez do processo de definição prematura do escopo [Maxwell 03];
- Se o projeto for pequeno e a estabilidade dos requisitos é um problema adote práticas ágeis [Highsmith 01a, Cockburn 01, Boehm 03].

4.3.2.5 Fator: Linguagem de Programação

Descrição: Linguagem usada para o desenvolvimento do software.

Problema: Os programadores estão adotando linguagens pouco familiares ou desconhecidas ou ainda adotando linguagens pouco produtivas, sem nenhuma justificativa.

Motivação: Pesquisas mostram que a escolha da linguagem de programação afeta a produtividade no desenvolvimento e a qualidade do código de várias formas [McConnell 04]. Desenvolvedores de software são mais produtivos utilizando linguagens de programação que lhes são familiares. Boehm [Boehm 87a] chega a afirmar que assumindo todos os demais fatores como constantes, a familiaridade com a linguagem de programação tipicamente requer 20% homens-hora a menos.

Estratégias:

- Adote uma linguagem de programação familiar a suas equipes [McConnell 04], considerando o que é mais adequado para o momento do projeto, mas sem desconsiderar o conhecimento da equipe e sua experiência;
- Adote uma linguagem que possua uma comunidade ativa e capaz de resolver/postar problemas comuns, com fonte de referências e consultas;
- Adote linguagens de programação de alto nível [Brooks 95, Jones 00, Boehm 00b, Jiang 07a, Anselmo 03, Simmons 07, McConnell 04];
- Se sua organização e equipes já possuem experiência em uma ou algumas linguagens específicas mantenha-os trabalhando com elas. Novas tecnologias, IDEs e linguagens devem ser primeiro testadas e utilizadas em projetos piloto. Depois destas etapas, treine a equipe de forma adequada e implemente as mudanças em alguns projetos por vez.

4.3.2.6 Fator: Processo

Descrição: Processo de desenvolvimento adotado para desenvolver ou manter produtos de software. Segundo Humphrey [Humphrey 90] é o conjunto total de atividades da engenharia de software necessárias para transformar os requisitos dos usuários em software. Este fator agrupa processo, boas práticas, métodos e metodologia de desenvolvimento.

Problema: Equipes se adaptando aos processos que não agregam a qualidade do produto ou a produtividade do time, processos sendo utilizados apenas para que os projetos estejam aderentes ao processo, sem benefício algum e documentação sendo gerada apenas pela documentação em si.

Motivação: Processo deve ser utilizado como uma trilha e não como um trilho, devendo orientar a equipe, servir como norte, ao invés de remover a habilidade de pensar ou de decidir qual é o melhor caminho a ser seguido. Os profissionais de software são “pensantes”, criativos e devem ser capazes de criticar e questionar o processo. O processo deve ser moldado a projetos específicos e a equipes, não o inverso [Clincy 03, Cockburn 01].

Estratégias:

- Mantenha tudo o mais simples possível, concentrando-se nos 20% que realmente agregam valor [Maxwell 03], escolha o ciclo de vida apropriadamente;
- Evite as ineficiências no desenvolvimento, entrega, garantia da qualidade e atividades de suporte, automatizando etapas manuais e repetitivas [Boehm 87a];
- Colete medidas do processo. Antes que possamos esperar que a produtividade melhore, é necessário medir [Anselmo 03]. Melhorar a produtividade de um processo de desenvolvimento de software requer uma habilidade de medir com precisão os atributos dos produtos de software gerados pelo processo e os fatores de produtividade que afetam a elaboração desses produtos [Yu 91];
- Realize reuniões de lições aprendidas para distribuir conhecimento no uso do processo entre os integrantes da equipes e entre equipes, e utilize ferramentas de gestão do conhecimento [Institute 06];
- Defina um processo com múltiplos caminhos [Clincy 03] para facilitar a adaptação do processo ao projeto;
- Inclua práticas de reuso no processo. Componentes que tenham passado por múltiplas equipes acumulam correções e resultam em uma maior qualidade [Boehm 87a];

- Adote ciclos de vida incrementais [Brooks 95, Highsmith 01a, Cockburn 01, Boehm 03, Chiang 04], projete um pouco, construa um pouco, teste um pouco, e cresça o sistema de forma incremental [Anselmo 03].
- Adote técnicas de gerenciamento ágil de projetos [Highsmith 01a, Cockburn 01, Boehm 03], tais como:
 - Foque nos entregáveis que adicionam valor ao cliente;
 - Adote pequenos ciclos de entregas de códigos;
 - Envolve o time de desenvolvimento no planejamento do projeto;
 - Sugira tarefas de tamanhos pequenos;
 - Encoraje as reuniões diárias para monitoramento do progresso do projeto;
- Capacite os times no processo adotado [Boehm 81, Institute 06];
- Busque processos simples que sirvam como uma trilha e não como um trilho;
- Elimine o retrabalho via detecção precoce dos erros, automação da programação, inspeções, testes e/ou adoção de prototipação se necessário [Boehm 88].
- Se sua organização ou projeto tiver que utilizar processos complexos, adote ferramentas de gerenciamento, comunicação e configuração [Bruckhaus 96];
- Evite colocar membros da equipe técnica para realizar tarefas administrativas;
- Defina indicadores adequados para assegurar a produtividade das equipes e estimular a motivação sem causar disfunção no sistema de medição e pouca efetividade na ação [Austin 96].

4.3.2.7 Fator: Tamanho da Equipe

Descrição: Número médio de pessoas que trabalharam ao longo do ciclo de vida de desenvolvimento do projeto na equipe.

Problema: A produtividade decresce quando o tamanho da equipe aumenta [Banker 87, Brooks 95, Clincy 03, Jiang 07b, Pendharkar 09].

Motivação: Mesmo que equipes maiores representem melhor distribuição de habilidades [Pendharkar 09], isso leva a um maior custo de comunicação e coordenação [Faraaj 00]. Maiores equipes têm mais problemas com: comunicação, compartilhamento de recursos, gerenciamento de conflitos, entregas, entre outros. Quanto mais pessoas são

colocadas em uma equipe, mais tempo de cada indivíduo é consumido com outros integrantes do time, na atualização de informações comuns, na resolução de problemas, além do aumento das oportunidades de conflitos [Boehm 81].

Estratégias:

- Mantenha as equipes pequenas, equipes menores demandam custos menores de comunicação e coordenação [Pendharkar 09]. Se necessário, reorganize os times em times menores. Busque o tamanho entre 3 e 9 pessoas [Scacchi 95]. *A equipe ideal, pequena e afiada, pelo senso comum, não deveria exceder dez pessoas* [Brooks 95];
- Divida cada segmento de um grande trabalho de forma que ele seja atacado por uma equipe, onde apenas um faz “os cortes” e os demais apenas dão todo o suporte que aumente a eficácia e produtividade (Mills H. 1971 *apud* [Brooks 95];
- Adote ferramentas de gerenciamento e comunicação. Ferramentas de apoio ao desenvolvimento de software provêm uma coordenação avançada e são frequentemente utilizadas para controlar a comunicação e a coordenação de grandes equipes [Pendharkar 09].

4.3.2.8 Fator: Ferramentas

Descrição: Uso de ferramentas que apóiam as etapas de desenvolvimento ou manutenção de software, tais como ferramentas de gestão de configuração e mudança, ferramentas de gestão e comunicação, ferramentas de modelagem e projeto e ferramentas de apoio ao desenvolvimento.

Problema: Projetos com uso excessivo ou escasso de ferramentas, independente de uma análise de tamanho do projeto, tamanho da equipe e processo a ser adotado.

Motivação: “A inovação tecnológica oferecida pelas ferramentas são frequentemente chamadas de balas de prata para velocidade da programação” [Chiang 04]. Ferramentas a muito foram reconhecidas como uma forma de melhorar a produtividade e a qualidade no desenvolvimento de software, mas é importante ter em mente o tipo de projeto, processo e tamanho da equipe para escolher as ferramentas melhores e mais adequadas.

Estratégias:

- Quando você planeja adotar um processo complexo e sofisticado, o uso de ferramentas parece ser a melhor justificativa [Bruckhaus 96];
- Utilize ferramentas para automação de parte do projeto, testes e atividades de garantia da qualidade [Boehm 87a];

- Adote ferramentas para análise e projeto. Ferramentas de análise e projeto podem eliminar uma grande quantidade do retrabalho através de visualização dos requisitos e especificações de projeto [Boehm 87a];
- Analise a equipe, o tamanho do projeto, bem como o processo a ser adotado para escolher as ferramentas do projeto [Bruckhaus 96];
- Encoraje as equipes a utilizarem ferramentas de modelagem e testes [Clincy 03];
- Analise, principalmente para projetos pequenos, se o ganho de produtividade na adoção de ferramentas é maior que o custo de adquiri-las e de treinar a equipe para o seu uso [Banker 89];
- Adote ferramentas modernas de programação para reduzir o tempo de construção do projeto [Jones 00, Boehm 00b, McConnell 04], ferramentas reduzem o volume de trabalho de programação [McConnell 04];
- Foque nos 20% das ferramentas que realmente serão utilizadas [Boehm 87b]. Se você não tem uma ferramenta realmente útil, você deve estar provavelmente perdendo algo [McConnell 04].

4.3.3 Fatores e Estratégias Relacionados a Pessoas

“As interações sociais entre as pessoas envolvidas em um projeto e as suas características individuais têm um papel fundamental no sucesso da empreitada”[Brooks 95].

4.3.3.1 Fator: Experiência e Habilidade da Equipe

Descrição: Abrange a capacidade e habilidade da equipe, sua experiência com a aplicação e linguagem de programação, habilidades e capacidades em geral.

Problema: Equipes com baixo desempenho.

Motivação: *“Sua competitividade é a sua habilidade de utilizar as habilidades e o conhecimento das pessoas de forma mais efetiva e times são a melhor forma de realizar isso.”* [Dumaine 94]. Uma das qualidades de um desenvolvedor experiente é a sua habilidade em adquirir conhecimento específico do projeto e evitar o retrabalho [Chiang 04].

“(...) em um estágio inicial do projeto, fui nomeado gerente de projetos e dado três trainees para ajudar no projeto. Meu maior erro foi torrar metade do meu

tempo e o tempo do outro projetista sênior tentando manter os trainees ocupados. Como resultado deixamos um grande buraco no projeto o que nos matou no final.” (Depoimento de um gerente de um projeto pequeno [Boehm 87a]).

Estratégias:

- Consiga as melhores pessoas para trabalhar em sua equipe, a melhor combinação de habilidades, conhecimentos e perfil [Boehm 81];
- Gerencie as habilidades, interação, talentos e a comunicação de suas equipes [Cockburn 01];
- Distribua as tarefas pelas habilidades e motivação das pessoas disponíveis [Boehm 81];
- Não se apresse em montar suas equipes muito cedo [Dumaine 94, Boehm 87a]. Tenha certeza de escolher as pessoas corretas para sua equipe;
- Construa equipes auto-contidas e auto-suficientes. Isso reduz o ciclo de vida, reduzindo a perda de tempo na transferência de informações, bem como o formalismo dessa transferência [Clincy 03, Highsmith 01a, Cockburn 01];
- Construa equipes auto-organizadas [Dumaine 94, Highsmith 01a, Cockburn 01];
- Mantenha a equipe próxima. A proximidade ajuda na comunicação e também permite a troca de experiências e troca do conhecimento na aplicação [Clincy 03, Highsmith 01a, Cockburn 01];
- Pessoas que não se enquadram ou não estão seguindo o ritmo da equipe devem ser demitidas ou disponibilizadas para outros projetos o mais cedo possível [Boehm 81, Boehm 87a];
- Selecione pessoas que se complementam e irão se harmonizar entre si [Boehm 81];
- Trabalhe com o senso de pertencer, com a satisfação dos colaboradores em fazer parte desta equipe e deste projeto [Sharp 09];
- Trabalhe com reuso. Reuso permite que pessoas mais experientes criem componentes e outros apenas utilizem os componentes [Boehm 00b, Lim 94];

- Trabalhe com o senso de propriedade coletiva para distribuir o conhecimento da aplicação e aumentar o comprometimento da equipe [Highsmith 01a, Cockburn 01, Faraj 00];
- Aloque as pessoas no time de acordo com a demanda. Para evitar ociosidade a quantidade de pessoas no projeto não deve ser uma constante durante o ciclo de vida do projeto [Boehm 81, Blackburn 96, Anselmo 03];
- Não promova bons engenheiros a gerentes de projeto, mesmo que as vezes isso funcione, na maioria das vezes trás frustrações, mais erros e danos na carreira [Boehm 87a];

“Se um projeto com 200 homens tem 25 gerentes que são programadores experientes e competentes, demita outros 175 e coloque os gerentes de volta na programação” [Brooks 95].

- Integração do time fora do ambiente de trabalho adiciona comunicação informal e trás maior integração também como equipe de desenvolvimento de software [Clincy 03];
- Forneça *feedback* à sua equipe, *feedback* motiva e motivação trás produtividade [Sharp 09].

4.3.3.2 Fator: Motivação

Descrição: Motivação é a forma pela qual a boa vontade, a energia e criatividade de uma pessoa é direcionada para alcançar os objetivos desejados [Boehm 81]. Motivação abrange a motivação do time e a habilidade gerencial e organizacional de manter o time motivado e focado nos objetivos do projeto.

Problema: Ambientes de trabalho de pouca inspiração a motivação, equipes sem estímulos e pouco criativas. Falta de comprometimento da equipe de desenvolvimento com os prazos estabelecidos.

Motivação: Motivação é um fator chave na produtividade de software [Boehm 82]. Equipes motivadas são mais produtivas e apresentam um maior compromisso com os objetivos do projeto e da organização. Motivação no desenvolvimento de software é reconhecida como um fator de sucesso para projetos de software [Sharp 09].

Estratégias:

- Crie um clima e um ambiente no qual a maioria das pessoas se motivem [Boehm 81];

- Envolver o maior número de pessoas possível nas ações do programa de produtividade. A equipe tem um papel muito importante na produtividade, por isso seu entusiasmo e motivação neste ponto é muito importante [Boehm 81]. Envolvimento traz motivação [Vosburgh 84];
- Reconheça os méritos de indivíduos e de grupos [Hantos 00, Clincy 03, Silva 03, Soares 09b, Zhuge 08, Dumaine 94];
- Forneça oportunidades de treinamento para ampliar as habilidades e especializar conhecimentos [Sharp 09];
- Motive através de trabalhos técnicos desafiadores [Silva 03, Sharp 09];
- Utilize a abordagem de times auto-gerenciados. A gestão coletiva motiva e traz mais comprometimento aos objetivos do projeto [Highsmith 01a, Cockburn 01];
- Celebre todas as conquistas. O sentimento de vitória motiva [Hantos 00];
- Evite sistematizar trabalhos rotineiros, como trabalho fragmentado e manutenções longas. Além de produzir software sem inspiração, esses métodos produzem pessoas que têm pouco interesse em crescimento profissional, que perdem interesse nos objetivos da organização e perdem a habilidade de cooperar com novas situações [Boehm 81].

4.3.3.3 Fator: Participação e Experiência do Cliente

Descrição: Participação ou contato com o cliente no desenvolvimento do software e sua experiência com a aplicação.

Problema: Comunicação formal com muitos e-mails e entregas, retardando decisões e impactando cronograma. Falta de conhecimento da equipe no negócio devido à pouca interação com o cliente.

Motivação: A produtividade cresce com a experiência do cliente na aplicação e a sua participação no levantamento e especificação dos requisitos [Scacchi 95]. O envolvimento do cliente em projetos de software é uma das mais conhecidas e difíceis estratégias. Mesmo que seja difícil, tente manter os clientes por perto mostrando o benefício deste esforço, especialmente se o cliente possuir um bom conhecimento do negócio da aplicação a ser construída. Este envolvimento reduz a quantidade de entregas (*handoffs*) e a necessidade de uma comunicação mais formal [Clincy 03].

Estratégias:

- Faça com que os usuários experientes estejam disponíveis para a equipe, ou ainda melhor, façam parte da equipe [Cockburn 01];
- Gerencie o cliente para garantir boa comunicação e relacionamento [Boehm 81]. Adote validações dos incrementos, forneça visibilidade do projeto e realize reuniões frequentes;
- Envolver o cliente durante todo o ciclo de vida do projeto [Clincy 03];
- Se o cliente não estiver disponível, planeje *workshops* presenciais com parte da equipe para que o conhecimento seja em parte compartilhado, estabeleça no time um representante do cliente e os mantenham ao máximo em contato. Estimule o contato e a comunicação informal, a abordagem de enviar um e-mail e aguardar o retorno tipicamente leva a um atraso de cronograma;
- A participação e envolvimento do cliente pede um grande comprometimento e maturidade da gerência, mas ajuda a manter o foco nos objetivos do projeto [Boehm 81].

4.3.3.4 Fator: Qualidade no Gerenciamento

Descrição: A qualidade e a habilidade com que a gerência conduz a equipe a atender aos objetivos pretendidos pelo projeto.

Problema: Projetos com custos elevados ou fracassados, clientes insatisfeitos e equipes improdutivas e desmotivadas.

Motivação: Um bom gerenciamento pode promover um processo de software eficiente, bem coordenado e altos níveis de capacidade e motivação na equipe, que leva a uma produtividade elevada [Boehm 81]. Bons gerentes auxiliam os membros da equipe a permanecerem motivados, comprometidos e focados nos objetivos do projeto. Um gerenciamento pobre pode reduzir a produtividade mais rapidamente do que qualquer outro fator [Boehm 87a]. Frequentemente a principal causa para o fracasso de grandes projetos é o gerenciamento pobre [Simmons 07]. Boehm [Boehm 81] relata que a produtividade em projetos de desenvolvimento de software é afetada por aqueles que desenvolvem o sistema e pela forma como eles estão organizados e gerenciados como time.

Segundo Chiang [Chiang 04], a produtividade de um time também é profundamente influenciada por como o esforço de gestão é distribuído.

Estratégias

- Envolver toda a equipe no planejamento de cada ciclo do projeto. Esta estratégia promove visibilidade dos estágios do projeto e do impacto em atraso de cada ponto [Highsmith 01a, Cockburn 01]. Participação no planejamento trás mais comprometimento e motivação;
- Descentralize a tomada de decisão e utilize o trabalho em equipe, estratégias vistas como críticas para competitividade e o desempenho organizacional [Silva 03];
- Trabalhe dando responsabilidades aos integrantes da equipe, “empoderar” trás motivação e comprometimento [Silva 03, Sharp 09];
- Trabalhe com a motivação de seu time, time motivado aumenta a eficiência geral da equipe [Clincy 03, Sharp 09];
- Recompense os membros das equipes, e equipes, utilizando bônus, promoções, cursos, livros ou até mesmo com palavras e celebrações das conquistas alcançadas [Soares 09b, Zhuge 08, Clincy 03, Dumaine 94, Sharp 09];
- Coordene a *expertise* das equipes. O desempenho da equipe não é questão de ter os conhecimentos e habilidades corretas, é preciso coordená-las pela equipe [Faraj 00];
- Gerentes e líderes são essenciais para o sucesso do projeto, mantenha-os bem treinados e motivados. Capacite os gerentes não apenas em tópicos relacionados à gestão como PMBOK(Project Management Body Of Knowledge)[Guide 04] ou SCRUM [Schwaber 04]. Pense em uma visão mais ampliada, capacite-os em liderança, gestão de pessoas e em conceitos de administração;
- Adote o *mentoring* sistemático na sua organização [Clincy 03]. Adote revisão em pares para troca de idéias, e acompanhamento por gerentes mais experientes até que os novos possam ser considerados aptos a caminhar sozinhos;
- Promova a leitura de livros, e reuniões para todos os líderes - novos e experientes - permitindo a troca de idéias tradicionais e inovadoras.
- Monitore o projeto diariamente [Schwaber 04];
- Conduza gestão quantitativa do processo do projeto, para alcançar o melhor uso de seu pessoal e tecnologia [Chiang 04];
- Realize reuniões diárias [Schwaber 04] de acompanhamento de projeto para garantir maior eficácia na gestão de tempos. Somando essa estratégia com a de se exibir

os resultados para toda a equipe, garante-se que os times vão querer melhorar seus resultados para não ter que no dia seguinte dizer que nada mudou ou ainda para que todos os colegas não vejam que seus resultados estão ruins.

4.4 CONSIDERAÇÕES FINAIS

“Desenvolvimento de software é uma arte e demanda muita criatividade. Mas a criatividade só traz bons frutos se for embasada em muita técnica, conhecimento científico, raciocínio lógico e boas práticas de engenharia”[Brooks 95].

A melhoria de produtividade tem sido uma preocupação constante dentre os pesquisadores e a indústria. O catálogo de estratégias para os fatores mais citados tem como objetivo fornecer uma visão unificada desses resultados, um conjunto de práticas da engenharia de software. Dessa forma, apoiar gestores e organizações na definição ou escolha de estratégias de melhoria de produtividade específicas e adequadas a seus projetos ou organização.

Este capítulo apresentou os resultados da revisão bibliográfica em estratégias para melhoria de produtividade no desenvolvimento de software. Para cada fator relevante de influência na produtividade, sua descrição, problema associado, motivação e estratégias como solução, foram descritas. Os fatores mais citados foram descritos e as estratégias apontadas pela literatura, juntamente com as estratégias oriundas dos experimentos foram documentadas por fator.

As estratégias podem não ser indicadas para todos os projetos e organizações, é preciso analisar qual subconjunto é mais adequado para cada organização. A meta-estratégia proposta e apresentada no próximo capítulo apóia nesta análise e escolha.

CAPÍTULO 5

META-ESTRATÉGIA

Este capítulo descreve a Meta-Estratégia proposta, que almeja a escolha de estratégias específicas para organizações de software, contempla a definição de um programa de melhoria contínua de produtividade.

....

5.1 INTRODUÇÃO

Mesmo que algumas pesquisas ainda mostrem uma leve melhoria na produtividade ao longo dos anos [Premraj 05], a produtividade de software tem caído mais rapidamente do que qualquer outra indústria [Anselmo 03]. Para reverter este fato é preciso um programa de melhoria de produtividade, onde uma série de ações em paralelo são traçadas para se obter ganhos significativos de produtividade [Boehm 81, Vosburgh 84]. O programa deve integrar iniciativas em diversas áreas, incluindo melhoria em ferramentas, metodologias, ambiente de trabalho, educação, gestão, incentivos pessoais e reuso de software [Boehm 82].

Banker *et al* [Banker 91] diz que é preciso identificar os fatores sob o controle gerencial, permitindo assim que a gerência possa agir para reter ou ampliar fatores positivos e eliminar ou pelo menos reduzir o impacto dos fatores negativos. Compartilhando deste princípio, a abordagem de identificação dos fatores foi mapeada como passo na Meta-Estratégia, através da pesquisa de vilões da produtividade (fatores de influência negativa) e definição do plano de ação a ser seguido, tomando como base as estratégias definidas no capítulo 4.

A estratégia proposta neste trabalho valida e complementa a visão de programa de melhoria de Boehm [Boehm 81], estabelecendo um foco em melhoria contínua de produtividade. Os passos mais significativos para estabelecer e adaptar um programa de melhoria de produtividade segundo Boehm [Boehm 81] são:

- Obter comprometimento da alta direção;
- Estabelecer agentes, responsáveis pela produtividade de software;

- Conseguir participação do maior número de pessoas possível dentro da organização. Esta estratégia leva ao estímulo do entusiasmo ao invés da resistência;
- Identificar objetivos, alternativas e restrições relacionadas à produtividade;
- Avaliar as alternativas e escolher a melhor combinação. Identificar e concentrar boas práticas de programação e gestão, assim como ferramentas e ambiente de trabalho;
- Preparar um plano de implementação incremental. O plano deve seguir o “*porque, o que, quem, quando, onde, como e por quanto*”;
- Obter autoridade para prosseguir;
- Implementar o plano;
- Acompanhar as iterações dos planos.

Todos os passos acima citados foram contemplados pela Meta-Estratégia proposta, conforme mostra a Tabela 5.1. Este trabalho busca ainda detalhar as etapas, e acrescentar ou melhorar alguma delas como o apoio do catálogo de estratégias, o estabelecimento da política e acrescentar a característica processual e cíclica de modelos de melhoria contínua.

Tabela 5.1. Etapas da Meta-Estratégia X Passos sugeridos por Boehm [Boehm 81]

Etapas da Meta-Estratégia	Passos sugeridos por Boehm
Estabelecer patrocínio	Obter comprometimento da alta direção.
Definir responsáveis	Estabelecer agentes, responsáveis pela produtividade de software; e Obter autoridade para prosseguir.
Iniciar/Renovar Programa de Melhoria	
Conhecer o Momento da Organização	
Identificar Vilões	Conseguir participação; e Identificar objetivos, alternativas e restrições relacionados à produtividade.
Analisar Catálogo de Estratégias	Avaliar as alternativas e escolher a melhor combinação.
Planejar Ciclo de Melhoria	Preparar um plano de implementação incremental.
Estabelecer Política	
Executar e Acompanhar Programa	Implementar o plano; e Acompanhar as iterações dos planos.

Este trabalho propõe uma Meta-Estratégia para definição de estratégias específicas, através de um programa de melhoria contínua de produtividade focando, a cada ciclo, em fatores que se apresentam como vilões da produtividade organizacional. A ideia é que um programa de melhoria de produtividade, mesmo que geralmente similar em diferentes empresas, para ser efetivo requer adaptações para as características únicas de cada organização [Boehm 82]. A Meta-Estratégia faz uso do conjunto de estratégias catalogadas para facilitar a definição e a manutenção de programas de melhoria específicos para cada organização de desenvolvimento de software. Desta forma, garante que cada particularidade seja atendida.

A Meta-Estratégia proposta está representada na Figura 5.1 e detalhada nas próximas seções.

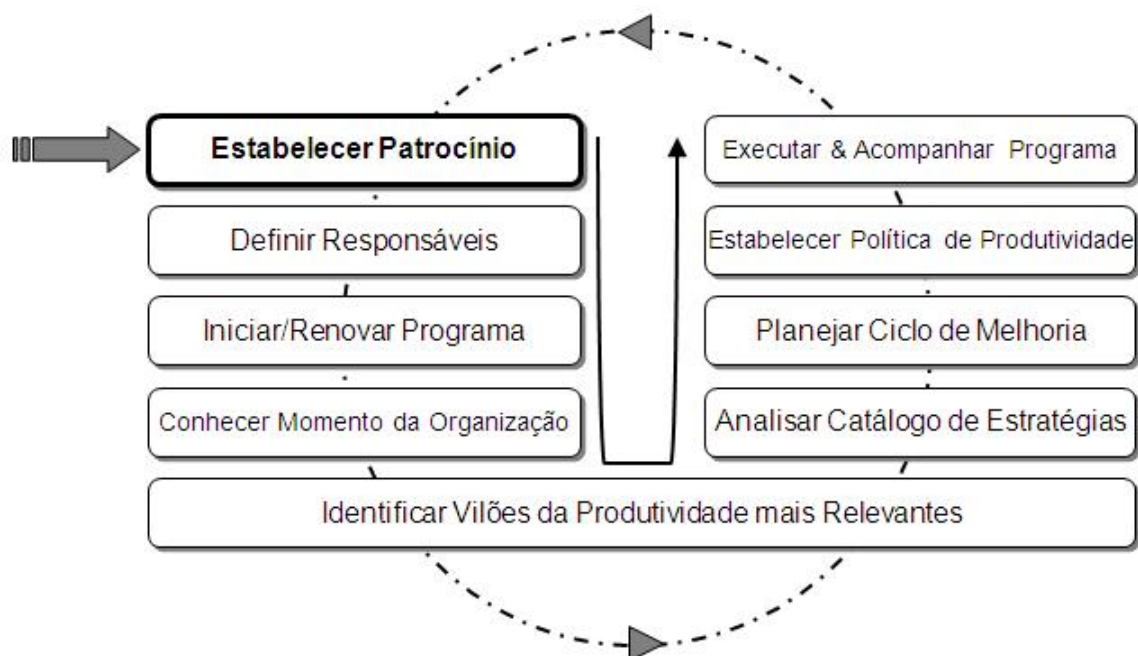


Figura 5.1. Meta-Estratégia proposta

A contribuição deste trabalho não está em repetir ou ainda em derivar mais uma vez o PDCA [Deming 86], até porque adaptações como esta já foram exaustivamente realizadas. A proposta é documentar a visão de Boehm, em conjunto com a visão processual do PDCA, inspirado em modelos de melhoria contínua como o modelo IDEAL [McFeeley 96].

Mesmo não possuindo a maturidade de um programa Six Sigma [McCarty 05] ou abordagens semelhantes, a Meta-Estratégia pode ser aplicada em qualquer tipo de empresa. É

importante ressaltar que os resultados de cada ciclo são específicos para cada organização.

A Meta-Estratégia é dividida em nove etapas ou passos, que devem ser repetidas em busca de melhoria contínua. O principal foco é identificar os principais vilões, trabalhar em cima deles de forma contínua para que sua influência na produtividade possa ser anulada ou minimizada. Essas etapas são descritas ao longo deste capítulo.

5.2 ESTABELECEER PATROCÍNIO

Objetivo: Reconhecer e entender a importância da melhoria de produtividade e estabelecer o comprometimento e os recursos necessários para a execução do plano.

O comprometimento da alta direção é um fator chave para o sucesso do programa de melhoria e de qualquer estratégia a ser tomada. *Sem um forte e inabalado comprometimento e patrocínio, o esforço está fracassado do início* [McFeeley 96].

É nesta etapa inicial que se formaliza, no primeiro ciclo da Meta-Estratégia, ou se reforça, em ciclos consecutivos, oficialmente o interesse da organização pela melhoria contínua de produtividade. Para que isso seja possível pelo menos um representante da alta direção precisa ser envolvido, no início ou a cada nova rodada do programa.

O primeiro passo na Meta-Estratégia de melhoria da produtividade proposta é estabelecer o patrocínio do projeto. Estabelecer o patrocínio envolve não apenas a definição de recursos financeiros, mas também é o momento de estabelecer o comprometimento da alta direção. É importante que todos entendam a importância de melhoria da produtividade, como seus esforços serão realizados, quem será envolvido e por quanto tempo. Assim como os demais projetos de melhoria, o programa de produtividade precisa ser uma prioridade na organização para que de fato possa acontecer. Para que isso seja possível, requisitos e metas devem ser estabelecidos e acordados com a alta direção, tais como:

- Meta de produtividade por natureza do projeto ou por linguagem de programação;
- Meta de satisfação do cliente a ser alcançada;
- Meta de redução do custo gasto na manutenção dos projetos.

O comprometimento da alta direção é importante para permitir que o time que será montado terá condições de realizar o planejamento detalhado e a execução do plano.

5.3 DEFINIR RESPONSÁVEIS

Objetivo: Organizar um time para planejar e elaborar um plano de melhoria de produtividade para a organização.

O segundo passo é definir os responsáveis e um ponto focal que possa traçar possíveis caminhos e liderar a implantação de ações de produtividade. Esta equipe precisa ser formada por pessoas com o perfil de liderança e bem conceituadas dentro da organização. Tais responsáveis devem ter condições e respaldo para executar tarefas e análises mais técnicas de processo ou gerenciais, por isso, é indicada uma equipe multidisciplinar. A equipe pode ser formada por integrantes em tempo parcial e deve comunicar periodicamente às equipes de projeto e à alta direção os resultados alcançados pelo programa.

É de responsabilidade desta equipe, monitorar não apenas as estratégias em execução, mas também os fatores que influenciam na produtividade da organização como um todo. De modo a garantir que estratégias para melhoria de um fator não levem ao detrimento de outros, os fatores devem ser monitorados no programa, como riscos são monitorados em um projeto [Guide 04].

A cada nova rodada do programa essa liderança deve ser redefinida ou renovada. A partir deste momento, o programa de melhoria pode ser dado como iniciado ou renovado.

5.4 INICIAR OU RENOVAR PROGRAMA DE MELHORIA DA PRODUTIVIDADE

Objetivo: Iniciar um novo programa ou novos projetos dentro deste programa renovando os objetivos a serem atingidos.

Um plano de melhoria deve ser elaborado com os requisitos e as metas do programa para conhecimento e o comprometimento com a alta direção, que deverá aprová-lo. O detalhamento de como os objetivos serão alcançados só será realizado na etapa de Planejar Ciclo de Melhoria.

Este é o momento de estabelecer um programa de melhoria da produtividade de software efetivo, onde um conjunto de estratégias para melhoria de produtividade são planejadas e executadas em paralelo, seguindo o trabalho de Boehm [Boehm 82] e Vosburgh [Vosburgh 84]. As estratégias devem endereçar as deficiências¹ organizacionais no contexto da produtividade no desenvolvimento de software.

Para que o programa de melhoria contínua de produtividade possa de fato ser avaliado quanto ao seu sucesso, é preciso analisar os objetivos e metas traçadas, em relação ao momento organizacional. Para isso uma linha de base para medições deve ser estabelecida, como mostra o próximo passo da Meta-Estratégia.

¹Nesta proposta as deficiências são vistas como fatores de influência negativa na produtividade.

5.5 CONHECER O MOMENTO DA ORGANIZAÇÃO

Objetivo: Estabelecer ou acompanhar um sistema de medição.

Melhorar a produtividade do software envolve um esforço longo e sustentado. O retorno é grande, mas requer um compromisso de longo prazo [Boehm 81]. Para alcançar esse objetivo e ainda alcançar o apoio da alta direção, que permite este tipo de compromisso de longo prazo, é preciso mostrar resultados e números. Por esta razão um programa de medição deve ser mantido em paralelo com o programa de produtividade.

À medida que a engenharia de software amadurece, a medição passa a desempenhar um papel cada vez mais importante no entendimento e controle do desenvolvimento de software [Kitchenham 95]. Druker [Drucker 01] afirma que para que a organização esteja apta a controlar seu próprio desempenho, os gestores precisam conhecer muito mais do que apenas metas. Eles devem estar aptos a medir a seu desempenho e analisá-las junto a esses objetivos. Essas medições não necessariamente precisam ser rígidas, mas principalmente devem ser simples, claras e racionais. O resultado dessa medição pode prover aos gerentes de projeto e à alta gerência, o apoio necessário para avaliar fatores específicos, a fim de que eles possam tomar decisões baseados em informações mais quantitativas. Baseados nessas informações, eles devem ser capazes de selecionar e tomar as ações corretas para alcançar a melhoria de produtividade.

Segundo Donald Anselmo e Henry Ledgard [Anselmo 03] precisamos ser capazes de medir a mudança na produtividade para validar nossas suposições, o que está alinhado com a afirmação de Tom DeMarco: “Não podemos controlar o que não podemos medir” [DeMarco 82]. Dentro do contexto estudado, também assumimos esta premissa. Quando não é possível medir ou interpretar os dados, dificilmente teremos o bom entendimento se melhoramos em qualquer questão, o que inclui a produtividade no desenvolvimento de software.

Nesse caso, para conhecer a organização é fundamental aferir a efetividade dos programas de melhoria da produtividade. É preciso saber *onde estamos* para acompanhar o progresso. Concordamos com os autores Yu *et al* [Yu 91] quanto à afirmativa de que a produtividade de um processo de desenvolvimento de software requer uma habilidade de medir os atributos dos produtos de software gerados pelo processo e os fatores de produtividade que afetam a elaboração desses produtos. Os autores afirmam o já dito por Boehm [Boehm 81] anos atrás, que cada projeto e cada ambiente de desenvolvimento, possuem seu próprio conjunto de fatores que influenciam na produtividade.

Segundo Robert Austin [Austin 96], a medição é algo potencialmente perigoso, porque o simples fato de medir, faz com que pessoas foquem apenas na dimensão que está sendo

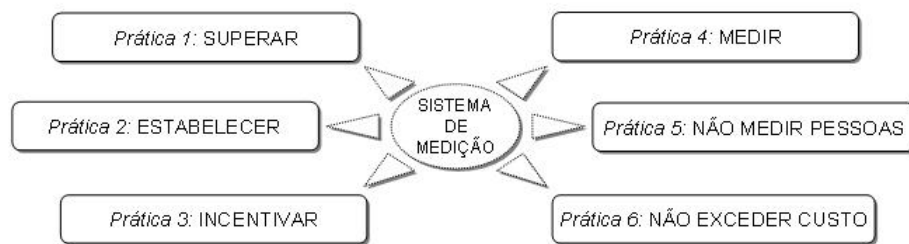


Figura 5.2. Boas Práticas de Medição

medida. Quando se mede qualquer indicador de desempenho, incorre-se no risco de piorá-lo.

Apesar de reconhecer a importância das métricas na gestão de uma organização de software e saber de seu papel no atendimento das necessidades do cliente, os sistemas de medição não têm trazido o retorno esperado. Mas, quais seriam as melhores práticas na adoção de um sistema de medição para que ele traga o retorno esperado?

A Figura 5.2 sugere algumas boas práticas a serem consideradas na adoção de medição em um sistema de melhoria, que são detalhadas abaixo.

- Prática 1: Superar velhos conceitos

Segundo Peter Drucker [Drucker 98], a maioria das nossas suposições sobre negócios, tecnologia e organizações tem pelo menos 50 anos. Elas têm sobrevivido a seu tempo. Como resultado, estamos pregando, ensinando e praticando políticas que estão cada vez mais desalinhadas com a realidade e por isso não são produtivas. Busque mais do que linhas de código produzidas, analise se a satisfação do cliente aumentou, se a equipe está mais motivada, se o valor agregado do produto aumentou e se a margem de lucro do projeto aumentou.

- Prática 2: Estabelecer métricas baseadas nos objetivos da organização;

Todas as métricas do sistema de medição devem ser baseadas nas metas organizacionais. Ao contribuir para o sucesso de uma medição alinhada com os objetivos da organização, toda equipe saberá que está contribuindo para o sucesso da organização.

- Prática 3: Incentivar, preparar e envolver pessoas;

Garantir um sistema de medição sem disfunção através do envolvimento e preparo das pessoas envolvidas. As equipes precisam entender o objetivo de medir e tê-lo em mente, não apenas o objetivo da métrica por si só.

- Prática 4: Medir o que é necessário;

Muitos processos de medição iniciam com o que é mais fácil medir, quando deveriam medir o que realmente é necessário. A aplicação desta prática remete a escolher um conjunto de métricas que agreguem valor, ao invés de gerar apenas dados que distraem os times de desenvolvimento, tirando-os do objetivo e favorecendo a disfunção.

- Prática 5: Não Medir Pessoas;

O processo de medição deve levantar um conjunto de informações necessárias para aferir desempenho dos processos e da organização como um todo. As pessoas quando medidas e observadas não se comportam da mesma maneira ou se concentram apenas nos resultados da medição e não em melhorar seu desempenho.

- Prática 6: Não permita que o custo da medição exceda o custo de produção;

Ao escolher uma medição, por mais alinhada que esteja com as metas organizacionais, é fundamental que se avalie o custo-benefício de sua coleta. Se coletar esses dados é mais caro que o benefício trazido com sua análise, essa métrica deve ser excluída do sistema de medição.

“Sempre existe alguma forma de medir algo, mas faz a diferença quanto você precisa saber sobre este algo comparado com quanto custa para levantar esta informação. DeMarco em depoimento a Austin” [Austin 96]

É importante iniciar as medições antes mesmo de iniciar a execução das estratégias para garantir que elas estão de fato funcionando e trazendo produtividade. Sendo assim, o terceiro passo da estratégia proposta, e a primeira ação do programa de melhoria, é conhecer o momento da organização. As medições iniciais serão utilizadas para estabelecer uma linha de base quanto à produtividade a partir das quais as melhorias poderão ser avaliadas. Mesmo que a organização não esteja pronta para estabelecer um programa de medição, algumas das informações organizacionais devem ser registradas e um conjunto mínimo de medições deve ser escolhido para acompanhamento da efetividade das estratégias traçadas. Sugerem-se medidas simples tais como:

- Satisfação do cliente;

- Variação do orçamento;
- Variação do cronograma.

Quando um sistema já existe ou medições já são coletadas este é o momento de analisar os objetivos e metas das medições e alinhar com os objetivos traçados junto aos patrocinadores.

5.6 IDENTIFICAR OS VILÕES DA PRODUTIVIDADE MAIS RELEVANTES

Objetivo: Conheça quem são os vilões da produtividade em sua organização, quais os pontos mais relevantes naquele instante para a melhoria.

A identificação dos corretos fatores que influenciam na produtividade eleva a efetividade das estratégias de melhoria de produtividade por direcionar atividades de gestão diretamente para o que possui maior impacto [Trendowicz 08].

Trendowicz *et al* [Trendowicz 08] afirma que na prática são duas as abordagens para identificar relevantes fatores que influenciam na produtividade. A primeira é abordagem de especialistas, onde um ou mais especialistas decidem sobre a relevância do fator. A segunda é a abordagem baseada nos dados, onde dados de medições existentes são analisados para selecionar um subconjunto de fatores relevantes baseado em algum critério. O trabalho dos autores apresenta um método integrado para selecionar fatores relevantes para a produtividade, que une as duas abordagens citadas. Onde os autores unificam medições subjetivas e objetivas, bem como incertezas sob o aspecto estatístico, para chegar na seleção dos fatores.

Esta etapa propõe algo menos formal, mas ainda semelhante. Onde, para que a organização possa concentrar seus esforços nas ações que trazem o melhor valor agregado para o programa, um estudo de prioridades deve ser iniciado baseado nos fatores que atuam como vilões de produtividade organizacional. Quando os dados estiverem disponíveis, oriundos de um sistema de medição, também devem ser entrada para análise de deficiências no contexto da produtividade organizacional, embora a seleção final dos fatores deva ficar com os colaboradores.

Para a realização da primeira pesquisa de vilões, os fatores apontados na literatura devem ser listados, para que seja possível identificar quais os mais relevantes para a primeira rodada de ações. De Marco e Lister [DeMarco 99] apontam entre os fatores relevante para a produtividade: o *turnover*, a quietude no ambiente de trabalho, a iluminação natural, gestão defensiva, burocracia, separação física, time fragmentado - part time, deadlines irreais entre outros. Muitos outros fatores foram relacionados ao longo deste trabalho,

mas trabalharemos apenas com os fatores e generalizações, tratadas neste trabalho como os mais relevantes. Novos fatores podem ser acrescentados nessa lista, de acordo com a sua maturidade e realidade das organizações.

É sugerido que os fatores apresentados a seguir façam parte deste primeiro levantamento, contemplando fatores relacionados a pessoas, projetos e produtos. Para facilitar no levantamento, as generalizações realizadas neste trabalho foram listadas abaixo de cada fator, melhor identificando a abrangência de cada fator.

- Fatores relacionados a pessoas
 - Falta de Motivação:
 - * Falta de desafios técnicos;
 - * Falta de um ambiente de fácil motivação.
 - Má Gestão do Projeto:
 - * Falta de desafios técnicos;
 - * Falta de um ambiente de fácil motivação;
 - * Ausência de uma gestão efetiva;
 - * Ausência de um acompanhamento efetivo pelos diversos níveis de gestão;
 - * Falta de experiência do gerente;
 - * Falta de conhecimento do gerente na aplicação.
 - Habilidade e Experiência da Equipe:
 - * Falta de experiência na linguagem;
 - * Falta de experiência na aplicação;
 - * Falta habilidades em geral;
 - * Organização da Equipe
 - Participação do Cliente:
 - * Falta de experiência do cliente na aplicação;
 - * Problemas na interface com o cliente.
- Fatores relacionados a produtos.
 - Adoção de reuso
 - * Falta de reuso;

- * Falha na adoção de reuso;
 - * Qualidade dos componentes reusáveis.
- Tamanho do produto;
- Complexidade da aplicação.
- Fatores relacionados a projetos ou a organização.
 - Tamanho da equipe;
 - Instabilidade dos requisitos;
 - Ambiente de trabalho favorece a dispersão;
 - Falta ou falha no sistema de recompensas;
 - Falta ou falha do processo de desenvolvimento de software:
 - * Ausência ou falta uso de padrões;
 - * Ausência ou falta de definição de papéis.
 - Ferramentas:
 - * Falta de ferramentas;
 - * Ferramentas inadequadas.
 - Linguagem de programação:
 - * Dificuldade ou improdutividade com a linguagem de programação adotada.
 - Cronograma:
 - * Restrição de cronograma;
 - * Cronograma irreal;
 - * Extensão de cronograma.

Esse passo da estratégia propõe realizar uma pesquisa questionando quais são os fatores que influenciam na produtividade de forma negativa, abrangendo o maior número de colaboradores possível. Esta pesquisa permite compreender a cultura organizacional quanto à produtividade, seus elementos principais e suas particularidades. Esta compreensão é necessária para identificação de objetivos, alternativas e restrições. Além disso, permite o envolvimento e comprometimento de um grande número de colaboradores - fator importante em iniciativas como esta.

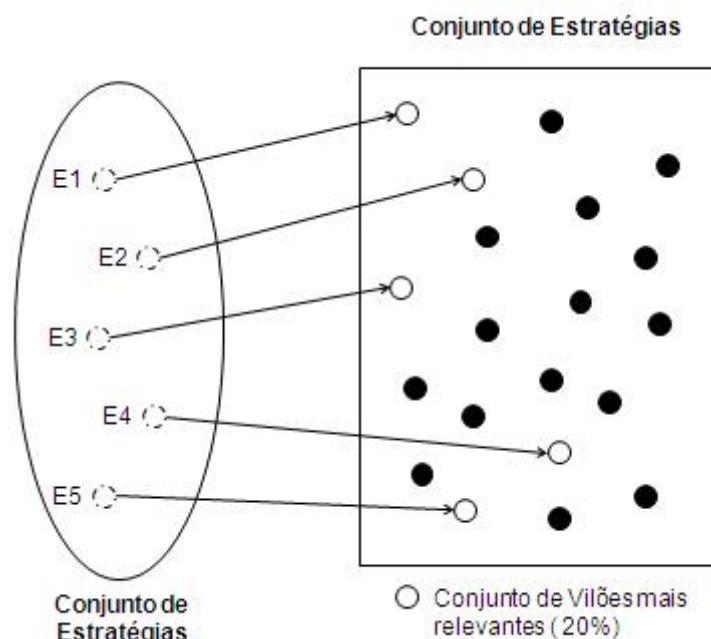


Figura 5.3. Subconjunto de vilões endereçados por estratégias.

Cada questionário deve ser respondido individualmente pelo maior número de colaboradores possível. Quando necessário, entrevistas podem ser realizadas para o melhor e mais eficaz levantamento das informações. É sugerido que o anonimato seja preservado.

O foco principal desta pesquisa é a visão interna e subjetiva e o levantamento dos principais fatores que influenciam na produtividade da organização. Entretanto, uma análise junto aos clientes também pode ser realizada para somar à visão completa da organização.

Os resultados devem ser consolidados para uma análise, onde o princípio de Pareto [Juran 99] deve ser seguido garantindo que a organização vai focar nos 20% dos vilões, os que representam 80% do impacto na produtividade. A Figura 5.3 ilustra esse conceito. Novas rodadas desses estudos devem ser realizadas a cada renovação do programa, ou ainda se o planejamento exigir.

5.7 ANALISAR CATÁLOGO DE ESTRATÉGIAS

Objetivo: Analisar estratégias catalogadas para cada fator selecionado. Selecionar e priorizar as estratégias a serem executadas.

Uma vez levantados os fatores e identificados os 20% mais relevantes, as estratégias

agrupadas por fator, catalogado no capítulo 4, devem ser analisadas para fazerem parte do plano de ação do projeto de melhoria.

Para facilitar a análise e posterior priorização das estratégias, rodadas de reuniões com integrantes do projeto devem ser realizadas apresentando o resultado da pesquisa de vilões juntamente com as estratégias a serem priorizadas. A primeira rodada de análise deve ser feita com Analistas, Desenvolvedores, Engenheiros de Configuração e demais colaboradores operacionais. A segunda deve conter os gerentes de projeto e demais líderes organizacionais. Esta separação visa não inibir os participantes da primeira análise, com a participação de seus superiores, enriquecendo o conteúdo das reuniões.

Para melhor exemplificar o formato de exibição, a Tabela 5.2 mostra as estratégias para um dos vilões citados no experimento exploratório.

Uma vez priorizadas as estratégias para os fatores selecionados, o planejamento do ciclo de melhoria pode ser concluído.

5.8 PLANEJAR CICLO DE MELHORIA

Objetivo: Definir quais estratégias farão parte do plano de ação do ciclo de melhoria, quem é o responsável por sua execução, até que data e com que esforço associado.

O planejamento do ciclo de melhoria deve contemplar as estratégias priorizadas na etapa anterior juntamente com ações sugeridas pelas rodadas de análise e priorização. Essas ações são igualmente importantes, já que a realidade da organização pode exigir necessidades específicas.

As estratégias e demais ações devem minimizar o impacto do fator, sem prejudicar outros fatores².

O planejamento do ciclo de melhoria deve seguir a recomendação de Boehm [Boehm 81] e citar *“porque, o que, quem, quando, onde, como e por quanto”*. O fator apontado como vilão (porque) deve ter uma ou mais estratégias endereçadas (o que), com um responsável associado (quem), um *deadline*(quando) e uma quantidade de horas prevista para sua execução (quanto). Os prazos e custos associados devem estar dentro do orçamento do programa definido nos primeiros passos da Meta-Estratégia.

5.9 ESTABELECEER POLÍTICA DE PRODUTIVIDADE DA ORGANIZAÇÃO

Objetivo: Estabelecer regras e tornar visível para toda a organização.

²Este é um ponto apontado como limitação desta pesquisa que não analisou o relacionamento dos fatores.

Tabela 5.2. Estratégias por vilão: **Má Gestão do Projeto**

Estratégias
Envolva toda a equipe no planejamento de cada ciclo do projeto. Esta estratégia promove visibilidade dos estágios do projeto e impacta no atraso de cada ponto. Participação no planejamento trás mais comprometimento e motivação.
Descentralize a tomada de decisão e utilize o trabalho em equipe, estratégias vistas como críticas para competitividade e o desempenho organizacional.
Trabalhe dando responsabilidades aos integrantes da equipe, “empoderar” trás motivação e comprometimento.
Trabalhe com a motivação de seu time. Time motivado aumenta a eficiência geral da equipe.
Recompense os membros das equipes e equipes utilizando bônus, promoções, cursos, livros ou até mesmo com palavras e celebrações das conquistas alcançadas.
Coordene a <i>expertise</i> das equipes. O desempenho da equipe não é questão de ter as <i>expertises</i> corretas na equipe, é preciso coordená-las pela equipe.
Gerentes e líderes são essenciais para o sucesso do projeto. Mantenha-os bem treinados e motivados. Capacite os gerentes não apenas em tópicos relacionados a gestão como PMBOK ou SCRUM. Pense em uma visão mais ampliada, capacite-os em liderança, gestão de pessoas e em conceitos de administração
Adote o <i>mentoring</i> sistemático na sua organização [Clincy 03]. Adote revisão em pares para troca de idéias, até que os novos gerentes possam ser considerados aptos a caminhar sozinhos.
Promova a leitura de livros e reuniões para todos os líderes - novos e experientes - promovendo a troca de idéias tradicionais e inovadoras.
Monitore o projeto diariamente.
Conduza gestão quantitativa do processo do projeto, para alcançar o melhor uso de seu pessoal e tecnologia.
Realize reuniões diárias de acompanhamento de projeto para garantir maior eficácia na gestão de tempos. Somando essa estratégia com a de se exhibir os resultados para toda a equipe, garante que os times vão querer melhorar seus resultados para não ter que no dia seguinte dizer que nada mudou ou ainda para que todos os colegas não vejam que seus resultados estão ruim.

Dentro do planejamento, uma política de produtividade para toda a organização deve ser estabelecida, como parte de um programa de melhoria da produtividade. A cada nova rodada do programa, quando novos vilões forem encontrados, a política deve ser revisada. A política deve endereçar os vilões encontrados em passos anteriores. A Tabela 5.3 mostra algumas sugestões de política, baseado no experimento realizado.

Tabela 5.3. Sugestão de Política de Produtividade

Vilão	Política
Ambiente e Espaço Físico de Trabalho	Estimular reuniões em salas específicas quando envolver mais de duas pessoas, favorecendo o ambiente silencioso. Manter celular no silencioso.
Experiência e Habilidade da Equipe	Cada novo colaborador deve ter um <i>menthor</i> . Todos os colaboradores devem ter acesso a treinamentos <i>on-line</i> . Estimular o bom uso da internet. Monitorando e bloqueando sites se necessário.

5.10 EXECUTAR E ACOMPANHAR O PROGRAMA DE MELHORIA DA PRODUTIVIDADE

Objetivo: Executar as estratégias previstas no plano de melhoria e acompanhar de seus resultados.

Uma vez priorizadas as estratégias e traçados responsáveis, é o momento de colocá-las em prática, seguindo a abordagem de executar várias iniciativas em paralelo.

As estratégias selecionadas devem ser acompanhadas de forma coordenada pelo programa de medição ou, no mínimo, pelas medições definidas na etapa de conhecer o momento da organização. Uma análise quanto à eficácia das estratégias deve ser realizada, observando não apenas o resultado pontual obtido, mas também sua tendência. Quando possível, uma análise de abrangência deve ser realizada para permitir analisar e entender as dependências entre as estratégias e seu impacto na produtividade. Nenhuma estratégia pode melhorar um ponto observado em detrimento de outro, ou ainda ter um bom resultado de curto prazo em detrimento de um bom resultado a longo prazo.

Ações organizacionais que não partiram do catálogo de estratégias devem ser documentadas.

Durante as rodadas de execução e acompanhamento do plano de ação, pode ser necessário a identificação de novos vilões ou ainda re-priorização dos já levantados e uma nova análise do catálogo de estratégias. As etapas são cíclicas até o atendimento do objetivo acordado com a alta direção ou ainda pelo término do prazo ou orçamento

previsto.

5.11 CONSIDERAÇÕES FINAIS

Este capítulo mostrou os passos da Meta-Estratégia proposta para escolha de estratégias específicas para alcançar melhoria de produtividade. Os passos envolvem desde a definição de uma infra-estrutura (patrocínio, responsáveis e metas) para o programa de melhoria até o levantamento dos principais vilões da produtividade e escolha das estratégias mais adequadas para minimizar o impacto do fator ou maximizar seu potencial quanto à melhoria de produtividade. Para um melhor entendimento da Meta-Estratégia, o próximo capítulo apresenta o estudo de caso realizado.

CAPÍTULO 6

ESTUDO DE CASO

Este capítulo apresenta as tentativas de validação desta pesquisa por meio da apresentação dos resultados obtidos no estudo de caso realizado para validar as etapas definidas na Meta-Estratégia de melhoria de produtividade, bem como o resultado de entrevistas realizadas para validar a relevância desta pesquisa e de algumas estratégias levantada na revisão bibliográfica.

....

6.1 INTRODUÇÃO

A principal validação da proposta deste trabalho se deu através de um estudo de caso realizado em uma empresa de Tecnologia da Informação do Porto Digital em Recife. A empresa, no momento do estudo de caso, possuía menos de 10 (dez) anos de existência e aproximadamente cento e trinta colaboradores em Recife.

A Meta-Estratégia proposta neste trabalho foi rodada pelo grupo de produtividade dentro da organização observada. O estudo de caso será relatado como foi conduzido. Alguns pontos diferem da estratégia proposta, já que esta foi melhorada após a execução do estudo de caso.

Algumas outras observações e entrevistas foram realizadas e são tratadas na seção 6.4.1 deste capítulo.

6.2 APLICANDO A META-ESTRATÉGIA PROPOSTA

6.2.1 ETAPA 1: Estabelecer patrocínio

O estudo de caso foi executado em conjunto com um projeto de estruturação de organização. O projeto surgiu a partir de um objetivo organizacional apontado dentro do planejamento estratégico. A organização estabeleceu como objetivo aumentar em 20% a produtividade no desenvolvimento de software. Vinculado a um objetivo organizacional, nasceu o programa de produtividade que tinha como patrocinador o CEO da organização.

6.2.2 ETAPA 2: Definir responsáveis

A equipe, do primeiro projeto deste programa de melhoria, foi formada por três pessoas em tempo parcial, que receberam como objetivo traçar ações e melhorar a produtividade organizacional. A equipe foi formalmente alocada - mesmo que em tempo parcial - para idealizar, planejar e executar o projeto, sendo um integrante da área de processo e qualidade e os outros dois diretamente de operações. O Diretor de Operações, e patrocinador, atuou como cliente no projeto, apoiando a equipe na priorização das ações executadas.

6.2.3 ETAPA 3: Levantar os principais vilões da produtividade

O ponto inicial do projeto de produtividade foi o levantamento dos vilões da produtividade daquela organização. A lista dos principais fatores citados na literatura foram relacionados em formato de questionário, conforme apresentado na Meta-Estratégia. Entretanto, alguns fatores foram adaptados para a realidade específica da organização e outros foram acrescentados pela diretoria e responsáveis. A lista final de fatores final ainda foi acrescida das sugestões citadas nos questionários. Todos os colaboradores foram visitados em suas posições de trabalho, e foram solicitados a responder, sem identificação, a enquete. O enquete usou como base algo semelhante ao modelo apresentado na seção 5.6.

Um total de 99 pessoas contribuíram com as pesquisas, desse total, 13(treze) questionários foram excluídos por falta de entendimento da tarefa realizada. A pesquisa atingiu desenvolvedores, analistas, gerentes, engenheiros de qualidade, engenheiros de configuração e engenheiros de teste. O resultado foi tabulado de duas formas: a primeira exibe o ranking de todos os itens que foram apontados como principal vilão (primeiro lugar entre os fatores que influenciam negativamente na produtividade do desenvolvimento de software), e a segunda a lista dos 10 fatores mais relevantes.

A Tabela 6.1 mostra o resultado dos 10(dez) fatores classificados como número 1 na pesquisa.

Uma vez consolidado o resultado das pesquisas, foram realizadas rodadas de análises críticas para levantar quais as ações a serem tomadas, para atacar os principais vilões e quais trariam maior ganho de produtividade para a organização.

O resultado mostrou um conjunto de fatores com maior impacto, ajudando a empresa a priorizar suas ações.

Fator	%
Mau uso do Instant Messenger	22,89
Ambiente físico favorece a dispersão	12,05
Capacitação insuficiente	10,84
Baixo uso de frameworks e padrões	9,64
Falta de planejamento / acompanhamento gerencial mais próximo	9,63
Falta de desafios técnicos gerando desmotivação	6,02
Internet	6,02
Ausência não registrada no ponto	4,82
Métricas de produtividade pouco confiáveis	3,61
Reuniões Improdutivas	3,61
Outros	10,96

Tabela 6.1. Ranking dos classificados como no. 1

6.2.4 ETAPA 4: Definição e priorização das ações que farão parte do plano de ação do projeto

Até este momento o estado da arte para as estratégias de melhoria ainda não tinha sido concluído. Isso significa que não tínhamos em mãos as principais estratégias segundo a literatura. As ações foram oriundas apenas das sugestões das equipes de desenvolvimento e da equipe responsável pelo programa de melhoria.

Uma vez concluída a compilação dos dados, quatro seções de apresentação dos resultados e brainstorm de ações de melhoria da produtividade foram realizadas, abrangendo a participação de mais de vinte colaboradores. Uma seção foi realizada com a alta direção da organização, outra com gerentes e outras duas tiveram a participação de desenvolvedores e analistas.

No final das reuniões, o responsável pelo programa de produtividade já tinha em mãos um bom conteúdo de problema e sugestões de ações oriundas de toda colaboração. As rodadas de análise e definição de ações se restringiram em cinco tópicos:

- Mau uso do *Instant Messaging*
- Capacitação insuficiente
- Internet
- Ambiente físico favorece a dispersão
- Baixo uso de frameworks e padrões

Quanto ao mau uso do *Instant Messaging* foram levantadas como ações:

- Criar Política /campanha educativa do bom uso da ferramenta, publicada na intranet e divulgada para novos colaboradores;
- Todos os colaboradores criarão uma nova conta apenas com contatos profissionais;
- Desabilitar alertas e manter o status ocupado;
- Uso de um único *Instant Messaging*;

Quanto a Capacitação insuficiente, foram levantadas como ações:

- Melhorar planejamento dos projetos para identificar as necessidades de capacitação/treinamentos o quanto antes possível;
- Estabelecer parcerias entre a organização do estudo de caso e outras empresas que gerem menor custo em cursos, treinamentos, etc;
- Equipes mais equilibradas em termos de experiência, capacidade, etc;
- Criar um programa de *mentoring*;
- Pairprogramming, checklists, revisões, etc;
- Cobrança compatível com a capacidade do profissional;
- Melhorar a gestão do conhecimento da empresa (registro das boas práticas, experiências e resultados dos projetos, etc);
- Programas de treinamentos internos, com treinamentos on-line;
- Estimulo estudo/capacitação autodidata;
- Incentivo a grupos de estudo.

Quanto a Internet foram levantadas como ações:

- Política/campanha educativa;
- Atualizar política da empresa com relação ao uso da internet;
- Verificar ranking dos domínios mais acessados periodicamente, para bloqueio de sites nocivos e divulgar o rank por e-mail, quadro de aviso, etc.

Durante a pesquisa foi verificado um alto nível de acesso de sites não relacionados com a produção e desenvolvimento de software. O que levou a um bloqueio de alguns sites durante o horário comercial.

Quanto ao “Baixo uso de frameworks e padrões” foram levantadas como ações:

- Definição de padrões de frameworks e ferramentas;
- Definir pessoa/equipe/área responsável por manter o padrão no projeto/empresa.

Quanto a “Ambiente físico favorece a dispersão” foram levantadas como ações:

- Campanha de silêncio;
- Celular no silencioso dentro do ambiente de trabalho e telefone no volume mínimo;
- Ginástica mais silenciosa! Ginástica laboral no hall e não no corredor;
- Evitar reuniões de corredor;
- Quadro em cada rua onde os vizinhos colocam deméritos por excesso de barulho;
- Regra: pessoas que trabalham juntas sentam juntas - analistas e engenheiros trabalhando lado a lado;
- Estudar formas de viabilizar subgrupos - separação física de equipes, diminuir ruído, ex.: uso de biombo, divisórias, paredes móveis, aumento da altura de baias, etc;
- Resolver problema de climatização: muito quente ou muito frio.

6.2.5 ETAPA 5: Planejamento e execução o plano de ação

O projeto teve as ações citadas na etapa anterior vinculada a um *Product Backlog*¹ juntamente com outras ações que também foram identificadas pela equipe do projeto. Os itens mais prioritários foram selecionados para serem executados dentro do primeiro ciclo de execução do projeto.

Várias ações foram executadas, mas as duas mais relevantes para este trabalho são àquelas já amplamente analisada pela comunidade científica e que posteriormente foram convertidas em passos da Meta-Estratégia proposta.

Ação 1 - Conhecer o momento da organização O primeiro desafio dentro do projeto de produtividade era conhecer o momento que a empresa estava vivendo, seus

¹Documento de requisitos de projetos executados com o *scrum* [Schwaber 04]

números, para conseguir de fato acompanhar a melhoria da produtividade. Como primeira ação do projeto - com esforços e recursos alocados - tivemos “Estabelecer uma linha de base da produtividade no desenvolvimento de software da organização”. Esta ação tornaria possível um acompanhamento do progresso desta produtividade.

A organização já possuía um conjunto de medições base como: tamanho do projeto em pontos de função ou pontos de casos de uso, esforço gasto por projeto para realização destes pontos de função, satisfação do cliente e atendimento ao orçamento do projeto. Muitas informações já existiam pulverizadas em ferramentas, entretanto não possuíam padrão organizacional. Foi preciso um esforço extra para coletar das ferramentas - de gestão de mudança, números de defeitos e reportagem de horas - as informações necessárias.

Alguns ajustes foram necessários nas ferramentas de controle de mudança e reportagem de horas para padronização e melhoria na recuperação das informações a partir desta linha de base, favorecendo o acompanhamento da melhoria da produtividade. Um padrão para reportagem de horas por atividade para toda a organização foi estabelecido.

Assumimos também duas medições relevantes para a organização como referencial para nosso acompanhamento de progresso: Variação do Orçamento do projeto e Satisfação do Cliente. Estas medições mostrariam segundo as perspectivas dos acionistas e dos clientes como estava a melhoria da produtividade dos projetos.

Esta ação foi transformada em passo da Meta-Estratégia proposta.

Ação 2 - Estabelecer uma política de produtividade para toda a organização, como parte do programa de melhoria da produtividade. A política foi uma coletânea de boas práticas baseada na experiência de colaboradores da organização. A política criada pela equipe inicialmente foi:

- Reuniões devem ocorrer em salas específicas quando envolver mais de duas pessoas, favorecendo o ambiente silencioso. A intenção sem dúvida é evitar conversas de corredor, entretanto sem tolher a boa comunicação dos times de desenvolvimento;
- Utilizar a internet apenas para fins de trabalho. Para este ponto uma nova pesquisa foi realizada, foi monitorado quais os principais sites de acesso e seus horários de pico de uso. De posse dos sites mais acessados, restrições para acesso a sites que não agregam valor a organização ou ainda que são nocivos para produtividade, foram estabelecidas. Alguns sites foram bloqueados em 100%, outros liberados apenas em alguns horários. Foi ressaltada para todas as equipes a importância de usar o horário de almoço para navegação da internet para fins pessoais;

- Evitar o mau uso das ferramentas de instant messenger. Uma única ferramenta deve ser utilizada, com alertas desabilitados e com email organizacional, apenas com contatos da empresa, sejam eles colegas de trabalho ou clientes. Deve ser utilizada apenas uma única ferramenta com email organizacional, desativar os alertas, colocar status para ocupado, caso necessário. Um prazo limite para esta migração foi estabelecido. O *feedback* dos mais experientes foi muito positivo neste ponto;
- Promover o bom uso dos emails. Este tópico se mistura com a gestão pessoal de tempo. No estudo de caso estimulamos que a checagem de email seja realizada apenas no início e final de cada turno e que os avisos de chegada de email sejam desabilitados;
- Dialogar mais e escrever menos. Quando o problema é de falha na comunicação, vale à pena reforçar que vale mais uma conversa de 15 minutos do que um grupo de emails;
- Estabelecer um horário núcleo² por projeto.

Os quatro times observados após convenção de 5 a 6 horas comuns tiveram diminuição das listas de pendências do projeto.

Esta ação foi transformada em passo da Meta-Estratégia proposta.

6.2.6 ETAPA 6: Acompanhar o programa de melhoria

Assim como qualquer outro projeto, o projeto de produtividade era acompanhado pela diretoria de operações quanto a seus objetivos específicos e quanto aos objetivos do programa. A primeira iteração gerou como resultado a política de produtividade, políticas relacionadas ao ambiente adequado para o trabalho (políticas de silêncio), políticas e metas para a equipe de recursos humanos e projetos internos de incentivo ao silêncio, bom uso da internet e o incentivo a produtividade, e o início do levantamento da *baseline* de medições. Este último ponto foi o mais trabalhoso.

A primeira iteração também produziu parte da infra-estrutura de medição, para que as futuras coletas fossem possíveis.

A segunda iteração focou no fechamento da infra-estrutura de medição e ainda em ações relacionadas a outros pontos tratados em políticas específicas como: estudos em

²Como a organização possuía um horário flexível (entre 7am e 10 pm) de trabalho, nem sempre as equipes estavam juntas para resolver os problemas técnicos. Por isso foi necessário estabelecer um horário mínimo comum para todos os integrantes da equipe.

grupos, aplicação de provas técnicas na seleção de novos colaboradores, programa de mentoring³, coleta sistemática quinzenal dos sites mais acessados, máquinas backup disponível por perfil de cada projeto, capacitação dos líderes, incentivo a certificações, entre outros pontos.

Mesmo não tendo acesso aos dados dos projetos, foi possível realizar uma coleta de *feedbacks* dos desenvolvedores, um ponto importante para este momento do projeto.

6.2.7 ETAPA 7: Fim do estudo de caso

O projeto produtividade continuou dentro do programa de melhoria da organização. O estudo de caso, a observação em si foi interrompida após pouco mais de sete semanas de execução.

A melhoria na produtividade organizacional não pode ser medida diretamente pelo conceito de produtividade mais conhecido ($\text{Produtividade} = \text{Output}/\text{Input}$) [SOMMERVILLE 07]. Entretanto, levando em consideração a visão moderna de produtividade, onde o valor produzido (*output*) pode ser analisado em múltiplas dimensões [Aquino 09b, Kitchenham 04], tais como resultado financeiro, satisfação do cliente, satisfação da equipe, podemos considerar que a validação através de um estudo de caso obteve sucesso.

Um levantamento informal com alguns colaboradores, no término do estudo de caso, constatou que oito em cada dez colaboradores acreditavam que as ações trouxeram melhoria de produtividade.

Pouco mais de seis meses após o término do estudo de caso, a organização disponibilizou apenas alguns dados solicitados. A satisfação do cliente⁴ foi analisada e não apresentou diferenças no resultado geral, segundo dados disponibilizados. Entretanto o fator “prazo”, uns dos itens observado no questionário do cliente, teve uma melhoria significativa, a média para os projetos, neste fator, passou de **6,8** para **8,12**. O que representa uma melhoria de 19,4%. Além disso, a organização tem como medição organizacional, a variação da margem planejada dos projetos. O indicador organizacional teve uma melhoria de aproximadamente 28%, segundo a equipe de qualidade.

³Todo novo colaborador terá um mentor técnico de mesmo papel, que deve acompanhar os primeiros passos de cada nova atividade prevista pelo processo de desenvolvimento de software da organização.

⁴A satisfação do cliente é monitorada através de questionários com notas de zero à dez para alguns fatores como prazo, qualidade, experiência da equipe, visibilidade, qualidade do produto, atuação da gestão entre outros

6.3 RESTRIÇÕES DA VALIDAÇÃO

As informações de desempenho da organização, referentes à produtividade e outros pontos foram tidas como sigilosas e não poderão ser reportadas em sua totalidade. Por isso, ficamos limitados a analisar a melhoria na satisfação do cliente (quanto ao fator prazo) e variação do orçamento do projeto.

Apesar de acreditar no sucesso do estudo de caso, não se pode afirmar que toda a melhoria obtida se deu apenas pela execução das estratégias utilizadas. Isolar as ações do projeto, do momento organizacional, da experiência dos times, do conhecimento do negócio não foi possível.

6.4 OUTRAS VALIDAÇÕES

Com os ajustes realizados na estratégia proposta e o conhecimento ampliado com uma maior revisão bibliográfica, novos experimentos foram necessários quanto a estratégias específicas para melhoria de produtividade. Entretanto apenas observações e pesquisas qualitativas foram possíveis.

6.4.1 Observações

Outras observações foram realizadas em duas empresas de Natal - RN. As empresas eram de pequeno porte (tamanho: 30 funcionários - EMPRESA A, 14 funcionários - EMPRESA B) e ambas orientadas a produtos de software. A primeira observação teve o objetivo de analisar a aceitação da Meta-Estratégia proposta e apoiar a execução desta. A segunda apenas apresentar estratégias para os problemas levantados.

6.4.1.1 Empresa A

Um *brainstorm* foi realizado para levantamento de problemas organizacionais. Um mapeamento dos problemas com fatores de influência na produtividade foi realizado. Uma matriz com os problemas, suas gravidades, tendências e prioridade foram levantados. Para os dois problemas mais urgentes, uma derivação de todas as possíveis causas utilizando o diagrama de causa e efeito foi realizada. A análise não envolveu representantes da alta direção, apenas gerentes, analistas e programadores.

A Meta-Estratégia foi mostrada, em conjunto com algumas estratégias específicas já catalogadas, para que a empresa a adotasse após uma pesquisa mais abrangente sobre os vilões. O resultado da pesquisa de vilões do estudo de caso foi apresentado para incentivar

a realização do projeto.

Os três participantes da reunião concordaram que a Meta-Estratégia teria um impacto positivo no trabalho que seria feito para melhoria de produtividade. As estratégias catalogadas foram repassadas para os gestores da organização pelos participantes.

Segundo o contato na empresa, os gestores iriam designar responsáveis para o programa de melhoria em tempo parcial, e aprovaram a idéia da Meta-Estratégia. Quanto ao catálogo de estratégias, que independente da pesquisa que deveria ser realizada com todos os colaboradores, parte das estratégias de Motivação, Sistemas de Recompensa e Qualidade no Gerenciamento, seriam adotadas.

Não foi possível acompanhar ou certificar a aplicação da Meta-Estratégia. O gerente de tecnologia repassou que as seguintes ações foram oriundas do trabalho realizado:

- a separação dos requisitos do produto X requisitos da iteração. Inicialmente a empresa tinha em um único quadro as ações de tudo que precisava ser feito para o produto o que desestimulava a equipe de desenvolvimento que no final do dia seguia com a visualização de que muito faltava para o término das pendências;
- a revisão do sistema de recompensa. A mudança no tratamento das recompensas iniciaria por uma vibração para o término das iterações. Como o trabalho era realizado dentro do ciclo de vida de desenvolvimento de produtos, nunca se comemorava nem mesmo o lançamento de versões do produto. Premiações e benefícios também foram adotadas;
- participação da equipe no planejamento e na tomada de decisão;

Nenhum *feedback* quanto ao seu sucesso em relação à produtividade foi recebido, apenas em relação à motivação e comprometimento dos times.

6.4.1.2 Empresa B

Na empresa B, durante o levantamento dos principais vilões de produtividade nos deparamos com a falta de descrição de papéis, falta de um processo descrito e comunicação deficiente.

A Meta-Estratégia foi apresentada para um gerente de projeto e dois analistas, e foi bem aceita, entretanto os primeiros ciclos deveriam ser com foco apenas no **Fator Processo**, para só depois adotar outras estratégias. As estratégias mapeadas para o fator observado também foram bem aceitas. As seguintes estratégias foram escolhidas pelo grupo, para o primeiro ciclo do programa de melhoria:

- Mantenha tudo o mais simples possível, concentrando-se nos 20% que realmente agregam valor [Maxwell 03], escolha o ciclo de vida apropriadamente;
- Realize reuniões de lições aprendidas, distribua o conhecimento entre os integrantes da equipes e entre equipes [Institute 06];
- Inclua práticas de reuso no processo;
- Adote técnicas de gerenciamento ágil de projetos;
- Capacite os times no processo adotado;
- Busque por processos simples que sirvam como uma trilha e não como um trilho;
- Evite colocar membros da equipe técnica para realizar tarefas administrativas;

A organização informou que as práticas adotadas tiveram 100% de aceitação. Solicitou uma segunda visita para apoio nos próximos passos, o que não foi possível.

6.4.1.3 Depoimentos de pessoas das Empresas B

Um depoimento importante surgiu de um desenvolvedor que iniciava a prática de métodos ágeis em seus times.

“Desde que nós desenvolvedores iniciamos a participação no planejamento somos obrigados a conhecer mais do produto. Além de ficar mais fácil de comprometer-se com o que estamos planejando temos mais visibilidade do projeto como um todo.”

Com este depoimento a analista do mesmo projeto foi abordada para que fosse possível uma visão mais completa do ganho na adoção da primeira estratégia sugerida, do catálogo de estratégias, para melhorar a Qualidade do Gerenciamento.

Envolve toda a equipe no planejamento de cada ciclo do projeto. Esta estratégia promove visibilidade dos estágios do projeto e do impacto em atraso de cada ponto.

“De fato temos que reconhecer o ganho que existiu quanto ao conhecimento do negócio. Os desenvolvedores ficam mais independentes ao longo do desenvolvimento, já que precisam conhecer mais do que suas atividades para participar do planejamento.”

6.5 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados obtidos na realização das validações. Primeiramente o estudo de caso realizado para validar a Meta-Estratégia foi detalhado mostrando cada uma das etapas, concluindo com resultados e restrições. Em seguida outros resultados de observações, para validação da relevância desta pesquisa, foram apresentados.

Conforme descrito na seção de restrição (6.3), apesar de acreditar no sucesso do estudo de caso, não se pode afirmar que toda a melhoria obtida se deu apenas pela execução da Meta-Estratégia. Ainda assim, o estudo de caso foi fundamental para este trabalho, fortalecendo a necessidade de um apoio das estratégias por fator, validando algumas etapas, e refinando outras etapas da Meta-Estratégia proposta. Além disso, os depoimentos colhidos após a realização do estudo de caso confirmaram a aceitação da proposta da Meta-Estratégia.

As observações e depoimentos também corroboraram com a idéia de uma proposta mais simples para melhoria de produtividade, a Meta-Estratégia, que se mostrou bem aceita nas empresas da região. As observações permitiram ainda validar a relevância de algumas estratégias do catálogo de estratégias, e confirmar a relevância deste para a proposta deste estudo.

Concluímos então que embora executado com restrições, os esforços de validação engrandeceram este trabalho e validaram tanto etapas da Meta-Estratégia quanto sua relevância, além de ter confirmado e reforçado a importância de conhecer estratégias específicas para cada fator. Novos experimentos ou estudos de caso apoiariam a melhoria do trabalho realizado, embora mantivesse as restrições quanto ao ambiente que não favorece o controle das variáveis para análise do real do sucesso da proposta.

CAPÍTULO 7

CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo traz um resumo do trabalho apresentado, mostrando como os objetivos propostos foram atingidos, exibindo também quais os problemas encontrados durante o desenvolvimento do mesmo. Em seguida, so apontadas as contribuições desta pesquisa para a área de conhecimento produtividade. Finalmente, so apontadas possibilidades de melhorias no trabalho e novas linhas de investigação para trabalhos futuros dentro do mesmo tema.

....

7.1 CONCLUSÃO

A preocupação das organizações em aumentarem a produtividade de suas equipes, tem dado às estratégias de melhoria uma grande atenção. Para cada fator estudado e citado na literatura, há sempre vários estudos sugerindo ou comprovando o que pode ser feito para potencializar seu efeito positivo ou minimizar, ou até mesmo anular, os efeitos negativos.

Nesse contexto, várias estratégias podem ser definidas para o aumento de produtividade, sejam elas no âmbito mais técnico e concreto como: definição de ambientes de desenvolvimentos mais modernos e adequados à realidade de cada projeto; adoção de ferramentas que apoiem no gerenciamento, comunicação e controle das etapas de desenvolvimento; escolha de linguagens de programação adequadas ao contexto do projeto, mas ainda familiares à equipe; ações para diminuir a complexidade do software; e ambientes de trabalhos silenciosos, claros e propícios à produtividade. Ou ainda ações relacionadas a fatores menos tangíveis como ações para que as equipes possuam um ambiente de fácil motivação; ações para melhoria da gestão do projeto, relacionamento com o cliente; ações para minimizar a volatilidade dos requisitos; entre tantos outros. O importante é que primeiramente seja feita uma análise sobre quais os fatores que influenciam na produtividade da organização para só então escolher dentre as estratégias a serem executadas.

Este trabalho propôs uma Meta-Estratégia para melhoria de produtividade como um diferencial competitivo para a organização que a adota, uma forma simples de definir estratégias de melhoria de produtividade e melhorá-la de forma contínua. Apesar de apenas parte das etapas da Meta Estratégia ter sido validada, o estudo de caso e observações foram suficiente para confirmar sua relevância.

Levantar o estado da arte sobre produtividade e fatores de influência foi fundamental para a consolidação do conhecimento adquirido e a base para a proposta neste trabalho. Entretanto a experiência de pouco mais de treze anos na indústria permitiu complementar este aprendizado e definir algo em sintonia com as necessidades de empresas de desenvolvimento de software.

Por fim, para apoiar a execução da Meta Estratégia e a escolha de estratégias específicas, foram apresentadas estratégias para endereçar fatores de influência na produtividade. O capítulo mais cansativo deste trabalho para o leitor e também o mais trabalhoso, permite suprir a necessidade percebida durante o estudo de caso, abrangendo estratégias para cada um dos fatores tidos como relevantes. Mesmo sendo extenso, como a proposta é focar apenas nos principais vilões, apenas alguns fatores devem ser analisados por rodada do programa de melhoria.

Para uma melhor visualização das estratégias a partir da literatura, estas foram agrupadas por categoria (produtos, projetos e pessoas) e fator. Além disso, para facilitar no entendimento, o formato de padrão foi parcialmente adotado, onde cada fator foi documentado associado a um problema, motivação e estratégias catalogadas.

Concluimos então que este trabalho foi relevante não apenas para a academia, mas principalmente para a indústria, apoiando os gestores na visualização dos fatores de influência na produtividade, permitindo conhecê-los para melhor escolher quais estratégias podem minimizar o impacto de um fator ou maximizar os resultados de outro. Neste contexto, para ser possível maximizar seus resultados de forma contínua, e ainda manter a sintonia com as boas práticas citadas pela literatura, a Meta-Estratégia permite estabelecimento e manutenção de programas de melhoria contínua da produtividade.

7.2 DIFICULDADES ENCONTRADAS E LIMITAÇÕES

Uma das dificuldades encontradas durante o desenvolvimento deste trabalho foi a pluralidade dos termos que referenciam produtividade no desenvolvimento de software. Muitos deles não faziam parte das *strings* originais de pesquisa e foram surgindo, e sendo exploradas apenas superficialmente. Uma revisão sistemática seria mais adequada para este tipo de pesquisa.

Além disso, o estudo de caso ocorreu na primeira versão da Meta-Estratégia, onde nem todos os passos estavam maduros e um segundo estudo de caso não foi possível.

Outro ponto importante é que como a melhoria de produtividade é um esforço de longo prazo, alguns dos resultados necessários para comprovar a efetividade da Meta-Estratégia ainda não foram coletados e outros não foram abertos pela organização do estudo de caso.

Por fim, outro aspecto que deve ser considerado é o envolvimento pessoal do autor no processo de condução dos experimentos. Mesmo com os devidos cuidados a imparcialidade nunca é plena.

7.3 CONTRIBUIÇÕES

Este trabalho trouxe três principais contribuições:

- Análise e mapeamento de estudos sobre fatores de influência na produtividade (Estudos X Fatores);
- Documentação de forma consolidada de estratégias para melhoria da produtividade agrupada por fator de influência;
- Uma Meta-Estratégia para apoio a gestores e organizações na definição de suas estratégias de melhoria de produtividade, e a criação e manutenção de programas de melhoria contínua de produtividade.

7.4 TRABALHOS FUTUROS

Como trabalho futuro temos:

- Uma revisão sistemática formal e completa sobre produtividade, fatores de influência e estratégias de melhoria, para complementação do trabalho realizado, incluindo estratégias mais técnicas para melhoria de produtividade;
- Documentação de estratégias para um maior número de fatores;
- Um modelo de melhoria contínua de produtividade semelhante ao CMMI [Institute 06] e ao MPS-BR [SOFTEX 07] que apresentasse diferentes níveis de maturidade, com um conjunto de práticas específicas e genéricas, onde as estratégias mais relevantes fossem definidas para cada nível.

REFERÊNCIAS BIBLIOGRÁFICAS

- [Albrecht 79] A. Albrecht. *Measuring application development productivity*. in IBM Application Development Symp, pages 83–92, 1979.
- [Anselmo 03] Donald Anselmo & Henry Ledgard. *Measuring productivity in the software industry*. Communications of the ACM, vol. Vol.46 No.11, no. 11, pages 121–125, 2003.
- [Aquino 09a] G Aquino, F. Furtado, R. Alchorne, S. Sampaio & S. Meira. *Disfunção dos Sistemas de Medição em Organizações de Software*. XII Conferência de Engenharia de Requisitos e Ambientes de Software Medellín, 2009.
- [Aquino 09b] S. R. L. Aquino G. S. and Meira. *An Approach to Measure Value-Based Productivity in Software Projects*. 9th International Conference on Quality Software - QSIC, aug 2009.
- [Austin 96] R Austin. *Measuring and managing performance in organizations*. Dorset House Publishing Company., 1996.
- [Bailey 81] John W. Bailey & Victor R. Basili. *A meta-model for software development resource expenditures*. In ICSE '81: Proceedings of the 5th international conference on Software engineering, pages 107–116, Piscataway, NJ, USA, 1981. IEEE Press.
- [Banker 87] Rajiv D. Banker, Srikant M. Datar & Chris F. Kemerer. *Factors Affecting Software Maintenance Productivity: An Exploratory Study*. In Proceedings of the Eighth International Conference on Information Systems, pages 160–175, 1987.
- [Banker 89] Rajiv D. Banker & Chris F. Kemerer. *Scale Economies in New Software Development*. IEEE Trans. Softw. Eng., vol. 15, no. 10, pages 1199–1205, 1989.

- [Banker 91] Rajiv D. Banker, Srikant M. Datar & Chris F. Kemerer. *A model to evaluate variables impacting the productivity of software maintenance projects*. Manage. Sci., vol. 37, no. 1, pages 1–18, 1991.
- [Basili 79] V. R. Basili & Jr. Reiter R. W. *An Investigation of Human Factors in Software Development*. Computer, vol. 12, no. 12, pages 21–38, 1979.
- [Basili 96] Victor R. Basili, Lionel C. Briand & Walcélio L. Melo. *How reuse influences productivity in object-oriented systems*. Commun. ACM, vol. 39, no. 10, pages 104–116, 1996.
- [Beck 04] Kent Beck. *programação extrema aplicada*. bookman, 2004.
- [Behrens 83] C. A. Behrens. *Measuring the Productivity of Computer Systems Development Activities with Function Points*. IEEE Trans. Softw. Eng., vol. 9, no. 6, pages 648–652, 1983.
- [Blackburn 96] J.D. Blackburn, G.D. Scudder & L.N. Van Wassenhove. *Improving speed and productivity of software development: a global survey of software developers*. Software Engineering, IEEE Transactions on, vol. 22, no. 12, pages 875–885, Dec 1996.
- [Boehm 81] Barry W. Boehm. *Software engineering economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
- [Boehm 82] Barry W. Boehm, James F. Elwell, Arthur B. Pyster, E. Donald Stuckle & Robert D. Williams. *The TRW Software Productivity System*. In ICSE '82: Proceedings of the 6th international conference on Software engineering, pages 148–156, Los Alamitos, CA, USA, 1982. IEEE Press.
- [Boehm 87a] B.W. Boehm. *Improving Software Productivity*. Computer, vol. 20, no. 9, pages 43–57, September 1987.
- [Boehm 87b] B.W. Boehm. *Industrial software metrics top ten list*. IEEE Software, vol. 4, pages 84–85, 1987.
- [Boehm 88] B.W. Boehm & P.N. Papaccio. *Understanding and Controlling Software Costs*. IEEE Transactions on Software Engineering, vol. 14, pages 1462–1477, 1988.

- [Boehm 99] B. Boehm. *Managing Software Productivity and Reuse*. Computer, vol. 32, no. 9, pages 111–113, 1999.
- [Boehm 00a] Barry W. Boehm, Chris Abts, A. Winsor Brown, Sunita Chulani, Bradford K. Clark, Ellis Horowitz, Ray Madachy, Donald J. Reifer & Bert Steece. *Software cost estimation with cocomo ii*. Prentice Hall PTR, Upper Saddle River, NJ, 2000.
- [Boehm 00b] Barry W. Boehm & Kevin J. Sullivan. *Software economics: a roadmap*. In ICSE '00: Proceedings of the Conference on The Future of Software Engineering, pages 319–343, New York, NY, USA, 2000. ACM.
- [Boehm 03] Barry Boehm & Richard Turner. *Observations on Balancing Discipline and Agility*. In ADC '03: Proceedings of the Conference on Agile Development, page 32, Washington, DC, USA, 2003. IEEE Computer Society.
- [Briand 98] Lionel C. Briand, Khaled El Emam & Frank Bomarius. *COBRA: a hybrid method for software cost estimation, benchmarking, and risk assessment*. In ICSE '98: Proceedings of the 20th international conference on Software engineering, pages 390–399, Washington, DC, USA, 1998. IEEE Computer Society.
- [Brooks 78] Frederick P. Brooks. *The mythical man-month: Essays on softw.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1978.
- [Brooks 95] Frederick P. Brooks. *The mythical man month: Essays on software engineering*, 2e. Pearson Education India, 1995.
- [Bruckhaus 96] Tilmann Bruckhaus, Nazim H. Madhavji, Ingrid Janssen & John Henshaw. *The Impact of Tools on Software Productivity*. IEEE Softw., vol. 13, no. 5, pages 29–38, 1996.
- [Butler 97] Greg Butler. *Quality and Reuse in Industrial Software Engineering*. In In Proceedings of Asia-Pacific Software Engineering Conference and International Computer Science Conference, pages 3–12. IEEE Computer Society Press, 1997.

- [Campos 92] V. Falconi. Campos. Tqc: Controle da qualidade total (no estilo japonês). Bloch, 1992.
- [Campos 97] V.F. Campos. Qualidade: gerenciamento da rotina do trabalho do dia-a-dia. Universidade Federal de Minas Gerais, Escola de Engenharia, Fundação Christiano Ottoni, 1997.
- [Chand 93] Donald R. Chand & Raghava G. Gowda. *An exploration of the impact of individual and group factors on programmer productivity*. In CSC '93: Proceedings of the 1993 ACM conference on Computer science, pages 338–345, New York, NY, USA, 1993. ACM.
- [Chiang 04] I. Robert Chiang & Vijay S. Mookerjee. *Improving software team productivity*. Commun. ACM, vol. 47, no. 5, pages 89–93, 2004.
- [Clincy 03] Victor A. Clincy. *Software development productivity and cycle time reduction*. J. Comput. Small Coll., vol. 19, no. 2, pages 278–287, 2003.
- [Cockburn 01] Alistair Cockburn & Jim Highsmith. *Agile Software Development: The People Factor*. Computer, vol. 34, no. 11, pages 131–133, 2001.
- [Coelli 05] T. Coelli, D. Rao, C. O'Donnell & G. Battese. An introduction to efficiency and productivity analysis. Springer Verlag, 2005.
- [Debou 00] Christophe Debou & Annie Kuntzmann-Combelles. Linking software process improvement to business strategies: experiences from industry, volume 5. John Wiley and Sons, Ltd., 2000.
- [DeMarco 82] Tom DeMarco. Controlling software projects: Management, measurement, and estimation. Prentice Hall Yourdon Press, 1982.
- [DeMarco 87] Tom DeMarco & Timothy Lister. Peopleware: Productive projects and teams. Dorset House Publishing Company, Incorporated, 1 edition, 1987.
- [DeMarco 99] Tom DeMarco & Timothy Lister. Peopleware: Productive projects and teams. Dorset House Publishing Company, Incorporated, 2 edition, 1999.

- [Demarco 09] Tom Demarco. *Responses to “Software Engineering: An Idea Whose Time Has Come and Gone?”*. IEEE Software, vol. 26, page 5, 2009.
- [Deming 86] W.E. Deming. *Out of the crisis*. MIT Center for Advanced Engineering Study. Cambridge, Mass, 1986.
- [Drucker 98] Peter Drucker. *Management’s New Paradigms*. Forber Magazines, page 1, 1998.
- [Drucker 01] P. Drucker. *The essential drucker*. Harper Business, 2001.
- [Dumaine 94] Brian Dumaine. *The Trouble with Teams*. Rapport technique 5, September 1994. ISSN-0015-8259.
- [Faraj 00] Samer Faraj & Lee Sproull. *Coordinating Expertise in Software Development Teams*. Manage. Sci., vol. 46, no. 12, pages 1554–1568, 2000.
- [Fenton 97] Pfleeger S. L. Fenton N. E. *Software metrics: A rigorous and practical approach*. PWS Publishing, 1997.
- [Gaffney Jr 92] JE Gaffney Jr & RD Cruickshank. *A general economics model of software reuse*. In Proceedings of the 14th international conference on Software engineering, pages 327–337. ACM, 1992.
- [Gamma 95] E. Gamma, R. Helm, R. Johnson & J. Vlissides. *Design patterns*. Addison-Wesley Reading, MA, 1995.
- [Gil 02] A.C. Gil. *Como elaborar projetos de pesquisa*. 2002.
- [Gilb 88] T. Gilb & S. Finzi. *Principles of software engineering management*. Addison-Wesley Wokingham, UK, 1988.
- [Groth 04] Robert Groth. *Is the Software Industry’s Productivity Declining?* IEEE Softw., vol. 21, no. 6, pages 92–94, 2004.
- [GROUP 09] STANDISH GROUP. *The Chaos Report*, <http://www.standishgroup.com>. Acessado em: 16/09/2009., 2009.

- [Guide 04] P. Guide. *A guide to the project management body of knowledge*. Project Management Institute, Pennsylvania, USA, 2004.
- [Hantos 00] Gisbert Mario Hantos Peter. *Identifying Software Productivity Improvement Approches and risks:Construction Industry case Study*. fev 2000.
- [Highsmith 01a] Jim Highsmith & Alistair Cockburn. *Agile Software Development: The Business of Innovation*. Computer, vol. 34, no. 9, pages 120–122, 2001.
- [Highsmith 01b] Jim Highsmith & Martin Fowler. *The Agile Manifesto*. Software Development Magazine, vol. 9, no. 8, pages 29–30, 2001.
- [Humphrey 90] W.S. Humphrey. *Managing the software process*. Pearson Education India, 1990.
- [Institute 06] SEI Software Engineering Institute. *CMMI for Development, version 1.2, staged representation*. Rapport technique, SEI, <http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf>, CMU/SEI-2006-TR-008., 2006.
- [jia]
- [Jiang 07a] Z. Jiang & C. Comstock. *The factors significant to software development productivity*. Proceedings of World Academy of Science, Engineering and Technology., vol. 21, 2007.
- [Jiang 07b] Z. Jiang, P. Naudé & C. Comstock. *The variation of software development productivity 1995- 2005*. World Academy of Science, Engineering and Technology, 2007.
- [Jiang 07c] Z. Jiang & P. Naude. *An examination of the factors influencing software development effort*. International Journal of Computer Information and Systems Science and Engineering., vol. 1, no. 3, pages 182–191, 2007.
- [Jones 91] Capers Jones. *Applied software measurement: assuring productivity and quality*. McGraw-Hill, Inc., New York, NY, USA, 1991.

- [Jones 00] C. Jones. Software assessments, benchmarks, and best practices. 2000.
- [Juran 99] J. M. Juran & A. Blanton Godfrey. Juran's quality handbook. McGraw Hill, 5 edition, 1999.
- [Karner 93] G. Karner. *Resource estimation for objectory projects*. Objective Systems SF AB, vol. 17, 1993.
- [Kitchenham 95] B. Kitchenham, S. Pfleeger & N. Fenton. *Towards a Framework for Software Measurement Validation*. In IEEE Transactions on Software Engineering. December 1995, 21 (12), pages 929–943. IEEE Press, 1995.
- [Kitchenham 04] E. Kitchenham B.and Mendes. *Software Productivity Measurement Using Multiple Size Measures*. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, vol. 30, no. 12, 2004.
- [Laird 06] L.M. Laird & M.C. Brennan. Software measurement and estimation: a practical approach. Wiley-IEEE Computer Society Pr, 2006.
- [Lakhanpal 93] B. Lakhanpal. *Understanding the factors influencing the performance of software development groups: Na exploratory group level analysis*. Information and Software Technology, vol. 35, pages 468–473, 1993.
- [Larman 00] C. Larman. Utilizando uml e padrões: uma introdução à análise e ao projeto orientados a objetos. 2000.
- [Lim 94] Wayne C. Lim. *Effects of Reuse on Quality, Productivity, and Economics*. IEEE Software, vol. 11, pages 23–30, 1994.
- [Lokan 01] C.J Lokan. *Impact of subjective factors on software productivity*. Australian Conference on Software Metrics, pages 29–30, November 2001.
- [MacCormack 03] A. MacCormack, C. F. Kemerer & B. Cusumano M.and Crandall. *Trade-offs between Productivity and Quality in Selecting Software Development Practices*. IEEE Software, vol. 20, no. 5, pages 78–85, 2003.

- [Maxwell 96] Katrina D. Maxwell, L. V. Wassenhove & S. Dutta. *Software development productivity of European space, military, and industrial applications, software Engineering*. IEEE Transactions, vol. 22, no. 10, pages 706–718, 1996.
- [Maxwell 00] KD Maxwell & P. Forselius. *Benchmarking software development productivity*. IEEE Software, vol. 17, no. 1, pages 80–88, 2000.
- [Maxwell 03] Katrina Maxwell. *Software development productivity*. Advances in Computers, vol. 58, pages 2–48, 2003.
- [McCabe 76] TJ McCabe. *A complexity measure*. IEEE Transactions on software Engineering, pages 308–320, 1976.
- [McCarty 05] T. McCarty, L. Daniels, M. Bremer & P. Gupta. *The six sigma black belt handbook*. McGraw-Hill New York, 2005.
- [McConnell 04] S. McConnell. *Code Complete*, 2004.
- [McFeeley 96] B. McFeeley. *IDEAL: A user s guide for software process improvement*. Handbook, CMU/SEI-96-HB-001, Pittsburgh, PA, 1996.
- [MEC 06] MEC, Secretaria de Política de Informática. *Programa brasileiro da qualidade e produtividade*, número 4, 2006.
- [Mintzberg 98] Henry Mintzberg, Bruce Ahlstrand & Joseph Lampel. *Strategy safari: a guided tour through the wilds of strategic management*(financial times series). The Free Press, New York, NY, 1998.
- [Mohagheghi 07] Parastoo Mohagheghi & Reidar Conradi. *Quality, productivity and economic benefits of software reuse: a review of industrial studies*. Empirical Softw. Eng., vol. 12, no. 5, pages 471–516, 2007.
- [Morasca 01] Sandro Morasca & Giuliano Russo. *An Empirical Study of Software Productivity*. In COMPSAC '01: Proceedings of the 25th International Computer Software and Applications Conference on Invigorating Software Development, pages 317–322, Washington, DC, USA, 2001. IEEE Computer Society.

- [Pendharkar 09] Parag C. Pendharkar & James A. Rodger. *The relationship between software development team size and software development cost*. Commun. ACM, vol. 52, no. 1, pages 141–144, 2009.
- [Porter 80] Michael Porter. *Competitive Strategy*. New York: the free Press, 1980.
- [Porter 04] Michael Porter. *Estratégia para o Brasil*. Revista Exame, vol. 15.01.04, 2004.
- [Premraj 05] Rahul Premraj, Martin Shepperd, Barbara Kitchenham & Pekka Forselius. *An Empirical Analysis of Software Productivity over Time*. In METRICS '05: Proceedings of the 11th IEEE International Software Metrics Symposium, page 37, Washington, DC, USA, 2005. IEEE Computer Society.
- [Ramasubbu 07] Narayan Ramasubbu & Rajesh Krishna Balan. *Globally distributed software development project performance: an empirical analysis*. In ESEC-FSE '07: Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, pages 125–134, New York, NY, USA, 2007. ACM.
- [Rasch 91] Ronald H. Rasch. *An investigation of factors that impact behavioral outcomes of software engineers*. In SIGCPR '91: Proceedings of the 1991 conference on SIGCPR, pages 38–53, New York, NY, USA, 1991. ACM.
- [Reed 04] Karl Reed, Ernesto Damiani, Gabriele Gianini & Alberto Colombo. *Agile management of uncertain requirements via generalizations: a case study*. In QUTE-SWAP '04: Proceedings of the 2004 workshop on Quantitative techniques for software agile process, pages 40–45, New York, NY, USA, 2004. ACM.
- [Reifer 02] Donald J. Reifer. *Let the Numbers Do the Talking*. Crosstalk: The Journal of Defense Software Engineering, pages 4–8, March 2002.
- [Ruhe 08] Melanie Ruhe & Stefan Wagner. *Using the ProdFlow(TM) approach to address the myth of productivity in real organizations*. In ESEM '08: Proceedings of the Second ACM-IEEE

- international symposium on Empirical software engineering and measurement, pages 339–341, New York, NY, USA, 2008. ACM.
- [Santana 07] André Felipe L. Santana. Problemas em iniciativas de melhoria de processo de software sob a ótica de uma teoria de intervenção. Master’s thesis, UFPE, 2007.
- [Scacchi 84] Walt Scacchi. *Managing Software Engineering Projects: A Social Analysis*. IEEE Trans. Software Eng., vol. 10, no. 1, pages 49–59, 1984.
- [Scacchi 89] W. Scacchi. *Understanding Software Productivity: A Comparative Empirical Review*. In In Proceedings of the 22nd Hawaii International Conference on system Sciences, volume 2, pages 969–977, Hawaii, 1989. IEEE Press.
- [Scacchi 95] Walt Scacchi. *Understanding Software Productivity*, 1995.
- [Schwaber 04] Ken Schwaber. Agile project management with scrum. Microsoft Press, Redmond, WA, USA, 2004.
- [Sharp 09] Helen Sharp, Nathan Baddoo, Sarah Beecham, Tracy Hall & Hugh Robinson. *Models of motivation in software engineering*. Inf. Softw. Technol., vol. 51, no. 1, pages 219–233, 2009.
- [Shore 08] J. Shore & D. Warden. The art of agile development. Alta books ltda, 2008.
- [Silva 03] Francisca Silva, Rubens Ramos & Ana Célia Campos. *Gestão de pessoas e performance organizacional: Uma investigação sobre os fatores direcionadores de satisfação e fidelidade de empregados*. XXIII Encontro Nac. de Eng. de Produção, vol. 10, no. 1, 2003.
- [Simmons 07] D. Simmons. *Software Organization Productivity*, 2007.
- [Siviy 07] J. Siviy, M. Penn & R. Stoddard. *Cmmi and six sigma: partners in process improvement*. 2007.
- [Soares 09a] F. Felipe Soares. Sistemas de recompensa como estratégia de melhoria de produtividade em organizações de software. Dissertação

- Centro de Informática, Universidade Federal de Pernambuco, Recife, 2009.
- [Soares 09b] Felipe Santana Furtado Soares, Gibeon Soares de Aquino Junior & Silvio Romero de Lemos Meira. *Incentive Systems in Software Organizations*. Software Engineering Advances, International Conference on, pages 93–99, 2009.
- [SOFTEX 07] SOFTEX. *MPS.BR - Melhoria de Processo do Software Brasileiro, Guia Geral (v. 1.2)*. <http://www.softex.br/mpsbr/>, 2007.
- [SOMMERVILLE 07] I. . 8.ed. 2007. 552 p. ISBN 9788588639287. SOMMERVILLE. Engenharia de software. Pearson Education, 8 edition, 2007.
- [Stensrud 03] Erik Stensrud & Ingunn Myrtveit. *Identifying High Performance ERP Projects*. IEEE Trans. Softw. Eng., vol. 29, no. 5, pages 398–416, 2003.
- [Thadhani 84] A. J. Thadhani. *Factors affecting programmer productivity during application development*. IBM Syst. J., vol. 23, no. 1, pages 19–35, 1984.
- [Trendowicz 08] Adam Trendowicz, Michael Ochs, Axel Wickenkamp, Jurgen Munch, Yasushi Ishigai & Takashi Kawaguchi. *An Integrated Approach for Identifying Relevant Factors Influencing Software Development Productivity*. pages 223–237, 2008.
- [Vosburgh 84] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben & Y. Liu. *Productivity factors and programming environments*. In 7th ICSE '84, pages 143–152, Piscataway, NJ, USA, 1984. IEEE Press.
- [Wagner 08] Stefan Wagner & Melanie Ruhe. *A systematic Review of Productivity factors in Software Development*. ESEM, ACM press, 2008.
- [Walston 77] C. E. Walston & C. P. Felix. *A method of programming measurement and estimation*. IBM Syst. J., vol. 16, no. 1, pages 54–73, 1977.
- [Welch 05] Jack Welch & Suzy Welch. *Paixão por vencer*. Elsevier, 2005. ISBN: 8535215050.

- [Yu 91] W.D. Yu, D.P. Smith & S.T. Huang. *Software productivity measurements*. In Computer Software and Applications Conference, 1991. COMPSAC '91., Proceedings of the Fifteenth Annual International, pages 558–564, Sep 1991.
- [Zhuge 08] Jianping Zhuge. *Reward Systems for Implementing Knowledge Sharing in Knowledge - Intensive Corporation*. In CCCM '08: Proceedings of the 2008 ISECS International Colloquium on Computing, Communication, Control, and Management, pages 514–518, Washington, DC, USA, 2008. IEEE Computer Society.