

Universidade Federal de Pernambuco  
Centro de Informática

Pós-graduação em Ciência da Computação

Modelo de Melhoria de Produtividade  
para Organizações de Desenvolvimento  
de Software

por  
Gibeon Soares de Aquino Júnior

**Relatório de Doutorado 2008**

Recife, 30 de maio de 2008

# Conteúdo

<b>1 Proposta de Trabalho</b>	<b>3</b>
<b>2 Plano de Trabalho</b>	<b>3</b>
2.1 Status atual . . . . .	3
2.2 Planejamento do Trabalho . . . . .	4
<b>3 Resultados</b>	<b>4</b>
3.1 Grupo de Pesquisa . . . . .	4
3.2 Orientações e Co-Orientações em Andamento . . . . .	5
3.2.1 Trabalhos de Mestrado . . . . .	5
3.2.2 Trabalhos de Pós-Graduação Lato Sensu . . . . .	6
3.3 Aplicação Industrial . . . . .	7
3.4 Publicações . . . . .	8
<b>4 Referências Relevantes</b>	<b>8</b>
4.1 Organização das referências . . . . .	9
<b>5 Apêndice: Survey (<i>Em andamento</i>)</b>	<b>10</b>
<b>1 Introduction</b>	<b>10</b>
<b>2 The Genesis</b>	<b>11</b>
<b>3 Productivity Measurement and Metrics</b>	<b>12</b>
3.1 Output Metrics . . . . .	13
3.1.1 Line of Code Based Metrics . . . . .	13
3.1.2 Others Code Based Metrics . . . . .	13
3.1.3 Value Based Metrics . . . . .	14
3.2 Compilação dos Resultados . . . . .	15
<b>4 Productivity Factors</b>	<b>15</b>
4.1 Compilação dos Resultados . . . . .	17
<b>5 Productivity Improvement Strategies</b>	<b>18</b>

# 1 Proposta de Trabalho

O trabalho proposto tem como direcionamento tratar a melhoria da produtividade como algo que podemos planejar, controlar e executar, independente da melhoria dos modelos de processo de software comuns como o CMMi (Capability Maturity Model Integration), ISO (International Organization for Standardization) e outros. A idéia é definir um modelo ou guia, focado na melhoria da produtividade, que permita a uma organização de software ser mais competitiva, implantando apenas as melhores práticas para aumentar a sua capacidade de produção. Historicamente, o aumento da produtividade foi tratado com uma consequência da melhoria dos programas de qualidade baseado nos modelos de maturidade, controle estatístico e outros. Sobre isso, este trabalho propõe uma nova abordagem na qual a produtividade é o foco central e a melhoria dos processos de software é apenas um dos métodos para alcançá-la. Outra abordagem é agir em áreas como motivação da equipe, ferramentas, ambiente de trabalho, estratégias de gerenciamento, educação, reuso, práticas de programação e outras. Além da definição do modelo, este trabalho tem como objetivo implantar sua proposta em organizações de desenvolvimento de software para provar que esta proposta é realizável na prática.

A fundamentação teórica para esta proposta é descrita com detalhes em um *Survey*, que está sendo escrito no momento, e sua versão inacabada pode ser visto na Seção 5.

## 2 Plano de Trabalho

Esta seção tem como objetivo descrever o plano de trabalho para conclusão das pesquisas e conseqüente defesa da tese de doutorado. Para isto será apresentado o progresso atual do projeto, na Seção 2.1, assim como o planejamento atualizado até o fim do projeto, na Seção 2.2.

### 2.1 Status atual

O aluno ingressou no doutorado no ano de 2005 e deveria estar no quarto ano de trabalho, mas o mesmo necessitou se afastar de suas pesquisas por pouco mais de um ano, tendo inclusive trancado oficialmente dois semestres letivos (2006.2 e 2007.1), justificados junto ao seu orientador e coordenação da pós-graduação. Sendo assim, o aluno encontra-se no terceiro ano do projeto e para finalizar o trabalho no prazo normal tem aproximadamente 1 ano e 9 meses (considerando a defesa em Fevereiro de 2010). Esta seção descreve a situação atual e as principais metas já atingidas em relação ao planejamento inicial de trabalho.

Dentre os objetivos definidos o aluno já conseguiu concluir as seguintes atividades:

- **Conclusão das disciplinas** – As disciplinas já foram cursadas antes mesmo do trancamento. A única pendência é a entrega de um Trabalho Individual, que já foi replanejado e estará sendo entregue até o fim de Junho deste ano;
- **Revisão Bibliográfica** – O aluno já fez uma vasta análise de trabalhos relacionados ao seu assunto, incluindo livros e artigos, conforme descrito na Seção 4;
- **Segmentação do Problema** – Baseado nas leituras já realizadas o aluno conseguiu segmentar seu problema em três partes principais, como pode ser visto na Seção 5, as quais

estão sendo estudadas e serão analisadas em Surveys diferentes, conforme planejamento de trabalho descrito na Seção 2.2. Além disso, o aluno já definiu diversos trabalhos de mestrados oriundos de suas pesquisas, dos quais 8 estão em andamento, como pode ser visto na Seção 3.2;

- **Elaboração do Modelo de Produtividade (Draft)** – Uma versão inicial do modelo de melhoria de produtividade já foi definida e está sendo implantada no primeiro estudo de caso, conforme pode ser visto na Seção 3.3.

## 2.2 Planejamento do Trabalho

A Figura 1 apresenta o cronograma das atividades planejadas desde o início de 2008 até o final da projeto de doutorado, previsto para Fevereiro de 2009.

	2008												2009												2010	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2	3	4	5	6	7	8	9	10	11	12	1	2
Revisão Bibliográfica sobre Produtividade	■	■	■	■																						
Survey – Métricas de Produtividade					■	■	■																			
Survey – Fatores que afetam a Produtividade							■	■																		
Survey – Estratégias de Melhoria de Produtividade									■	■	■															
Elaboração do Modelo de Melhoria de Produtividade																										
Implantação do Modelo																										
Case 1 (C.E.S.A.R)																										
Case 2																										
Submissão de Artigos																										
Qualificação																										
Defesa da Proposta																										
Escrita da Tese de Doutorado																										
Defesa da Tese de Doutorado																										

Figura 1: Cronograma de Trabalho

## 3 Resultados

Nas subseções subsequentes são descritos os resultados alcançados até o momento. Na Seção 3.1 é descrita a iniciativa de condução de um grupo de pesquisa com foco no assunto de produtividade. A Seção 3.2 descreve os trabalhos de pesquisas originadas a partir deste trabalho de doutorado e que estão em andamento no momento. A Seção 3.3 descreve a aplicação prática do estudo na indústria. Por fim, a Seção 3.4 descreve os resultados relacionados à publicação de trabalhos.

### 3.1 Grupo de Pesquisa

No final do ano de 2007 foi formado um grupo de pesquisa em produtividade de software denominado ProSE - *Productivity in Software Engineering*<sup>1</sup>. Este grupo tem como missão compreender todos os aspectos relacionados à melhoria e avaliação da produtividade em projetos de software. Os trabalhos de pesquisas de interesse do grupo têm como objetivo investigar e desenvolver o estado-da-arte relacionado à medição e melhoria da produtividade em engenharia de software.

O grupo atualmente conta com 15 membros, sendo 1 doutor, 1 doutorando, 3 mestre, 8 mestrandos, 1 pós-graduado e 1 graduado. O grupo atua em 3 linhas de pesquisa, Análise

<sup>1</sup><http://prose.cesar.org.br>

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.

O grupo se reúne semanalmente para discussão sobre artigos e livros de acordo com uma agenda previamente definida. Além disso, em alguns encontros são discutidos e apresentados os trabalhos dos próprios membros do grupo. Estes encontros possibilitam um alinhamento e identificação dos pontos de sinergia entre os trabalhos de pesquisa, além de ser um momento de troca de conhecimentos e experiência.

Neste ano a iniciativa do ProSE foi submetida para o PBQP Software (Programa Brasileiro de Qualidade e Produtividade em Software)<sup>2</sup>, sendo aceito para fazer parte dos projetos avaliados no ciclo de 2008.

## 3.2 Orientações e Co-Orientações em Andamento

O escopo do trabalho proposto originalmente era bastante extenso e a medida que as pesquisas foram sendo feitas percebeu-se uma vasta literatura científica sobre o assunto e diversos problemas em aberto. Esta constatação motivou a definição de sub-propostas de trabalhos de pesquisas e o recrutamento de pessoas com potencial de realizar tais trabalhos. As seções subsequentes listam os trabalhos de mestrados e monografias *Lato Sensu* em andamento, diretamente relacionadas ao trabalho de doutorado proposto.

### 3.2.1 Trabalhos de Mestrado

Atualmente existem 8 alunos de mestrados trabalhando em 6 linhas de pesquisas distintas, as quais são listadas e detalhadas a seguir. Deste total de alunos, 3 ingressaram em 2007.1 e 5 em 2008.1.

- **Uso de Medições Estatísticas como ferramenta de apoio à adoção de práticas ágeis em organizações CMMI 4 e 5** – Este projeto tem como objetivo estudar a influência das medições estatísticas implementadas pelo nível 4 do CMMI para definir e analisar dados que ajudem na melhoria e simplificação dos processos maduros, tornando-os cada vez mais compatíveis com os processos ágeis. Esta compatibilidade com métodos ágeis pode ser alcançada pela mensuração da velocidade de produção dos itens valorizados pelo cliente e pela utilização dessa métrica no planejamento do projeto. Ao definir um conjunto de métricas ágeis, construindo-as dentro de uma estrutura de controle estatístico, preparamos o processo para ter uma abordagem ágil e ao mesmo tempo aderente ao CMMI.
- **Estimativa de Esforço no Desenvolvimento Distribuídos** – Este trabalho tem como objetivo identificar quais são os fatores que afetam diretamente o planejamento de esforço em ambiente de Desenvolvimento Distribuído de Software (DDS), além do grau de influência de cada um. Deseja-se também, sugerir adaptações em métodos de estimativa incluindo os fatores detectados durante as análises. Como resultado deste trabalho, pretende-se melhorar a acurácia de estimativa de esforço em ambientes DDS e consequentemente contribuir com sugestões de melhoraria no planejamento deste tipo de projetos.

---

<sup>2</sup><http://www.mct.gov.br/index.php/content/view/2867.html>

- **Análise da influência da qualidade do código na produtividade do time** – Este trabalho analisa a influência da qualidade do código de software na produtividade do time através um exame das métricas de código que influenciam na qualidade da arquitetura e métricas de código que influenciam na produtividade. O resultado principal esperado é a identificação de um conjunto de métricas para controlar a arquitetura de software afim de a aumentar produtividade do time.
- **Linguagem de padrões de produtividade no desenvolvimento de software** – A proposta deste trabalho é registrar e divulgar boas e más práticas relacionadas à produtividade no desenvolvimento de software. Estas práticas são largamente citadas em literatura. Este trabalho seguirá um formato estruturado, baseado em linguagens de padrão, como usado por Christopher Alexander, Eric al de et de Gama e J. Coplien. Uma linguagem de padrão de desenvolvimento aplicado a este contexto legaliza experiências registradas ao longo de anos, de um modo simples, unificado e em um formato suficientemente conhecido. Além disso, este formato permite documentação da correlação entre soluções, uma vez que um dos principais problemas dessa área é a falta de visão sistêmica.
- **Métodos para aferir a eficiência de times de desenvolvimento de software** – Este projeto propõe modos para medir a eficiência em desenvolvimento de software que inclui cada membro do time e o time inteiro. O trabalho envolve o desenvolvimento de novos indicadores ou combinação do que existe na literatura, a definição de procedimentos para coleccionar e analisar estes indicadores e possíveis ferramentas para ajudar o trabalho.
- **Catálogo de fatores que influenciam na produtividade em projetos de software** – O objetivo deste trabalho é compilar os diversos fatores que afetam a produtividade segundo a literatura científica sobre o assunto, classificá-los e organizá-los em um catálogo. A idéia é construir um catálogo similar aos existentes nas áreas de comércio, saúde, biologia, por exemplo. Já em um catálogo de doenças temos informações como nome científico, agente transmissor, formas de contágio, principais sintomas, formas de prevenção e tratamento. Na biologia, temos como exemplo, um catálogo de animais onde se encontram dados como nome, nome científico, distribuição geográfica, habitat, hábitos alimentares, tamanho, peso, etc.

### 3.2.2 Trabalhos de Pós-Graduação Lato Sensu

Os trabalhos de monografia estão sendo conduzidos por alunos da Pós-graduação Lato Sensu em Gestão da Tecnologia da Informação, Centro de Informática, UFPE. Estes trabalhos estão previstos para serem concluídos em Outubro de 2008 e são detalhados a seguir.

- **Lean Software Development: práticas para melhoria da produtividade** – O objetivo da monografia é analisar a metodologia de desenvolvimento de software lean a fim de identificar práticas que implicam diretamente na melhoria da produtividade do processo. A intenção é estudar, para cada um dos princípios lean, práticas de processos de desenvolvimento de software que visem direcionar o processo para uma melhoria de produtividade, alinhadas com o princípio norteador do pensamento lean: enxugar o processo para entregar valor ao cliente rápida e eficientemente.

- **Scrum e Lean Software – um estudo sobre as semelhanças, diferenças e a possibilidade de integração** – Esta monografia tem como objetivo fazer um estudo individual destes dois métodos ágeis, que possuem muitas características focadas em produtividade, em seguida realizar uma comparação identificando as semelhanças e diferenças entre eles e estudar a possibilidade de fazer um uso integrado dos dois métodos.
- **Uso de métricas no acompanhamento de projetos utilizando o método de desenvolvimento ágil Scrum** – Este trabalho tem como objetivo discutir a utilização de métricas no Scrum uma vez que o mesmo tem conquistado bastante popularidade nos últimos tempos. Ambientes de desenvolvimento que utilizam essa metodologia exigem um ciclo constante de inspeção, adaptação e melhoria uma vez que o escopo dos projetos é bastante dinâmico. Em decorrência dessa característica não é recomendável a utilização das métricas ditas tradicionais uma vez que as mesmas partem do princípio que o escopo do projeto é estático ou pouco mutável.
- **Catálogo com Boas Práticas na Engenharia de Testes de Software** – Esta monografia terá como principal objetivo a compilação das boas práticas da Engenharia de Testes, encontradas na literatura junto às experiências práticas, em forma de um catálogo, que permita tornar as atividades do Engenheiro de testes mais eficientes e produtivas.

### 3.3 Aplicação Industrial

Desde de Dezembro de 2007 o aluno vem trabalhando em tempo integral na condução de um projeto de melhoria de produtividade no C.E.S.A.R. Este projeto envolve a aplicação direta da compilação das recomendações e conclusões das pesquisas científicas em projetos de software em andamento na organização. Atualmente já existe um modelo de melhoria de produtividade *draft*, que está sendo construído e refinado à medida que novas referências científicas são analisadas e, principalmente, com os resultados práticos verificados durante a aplicação do mesmo nos projetos da organização.

O escopo de trabalho no C.E.S.A.R tem duas linhas de ação específicas, são estas:

- **Definição de métricas de produtividade** – Esta é uma das linhas pesquisada como escopo deste trabalho de doutorado e um dos problemas clássicos desta área. O objetivo da linha de atuação no C.E.S.A.R é definir um modelo baseado em métricas que possibilite a avaliação de produtividade em diferentes projetos. A versão 1.0 do modelo já foi definida e as informações coletadas para alguns projetos. Diversas melhorias foram sugeridas e um nova versão está em estudo.
- **Ações com foco em melhoria de produtividade** – Baseado na revisão bibliográfica realizada durante os estudos do aluno diversas ações para melhoria de produtividade foram identificadas e compiladas para que pudesse ser feito um planejamento de quais ações seriam, inicialmente, aplicadas nos projetos da organização. Após diversas discussões com os gerentes e dirigentes da instituição um conjunto de ações foram selecionadas como prioritárias e estão sendo implantadas em alguns projetos pilotos (atualmente 3), envolvendo aproximadamente 20 pessoas. Cada umas das ações têm o seu próprio cronograma e após um determinado momento será possível fazer uma avaliação dos benefícios das mesmas.

### 3.4 Publicações

Em relação às publicações, poucos resultados concretos foram obtidos. Este ano foi submetido apenas um artigo ao SBES. Apesar dos resultados serem insuficientes, algumas ações já estão sendo executadas para reverter esta situação e alcançar resultados importantes em relação a este tópico, estas ações são:

- Identificação de conferências e Jornais alvos para submissão de trabalhos relacionados à pesquisa sendo conduzida. Esta identificação já foi realizada este ano e compilada em uma seção<sup>3</sup> no site do ProSE.
- Um survey sobre o assunto está sendo escrito, como pode ser visto na Seção 5, e assim que ele for concluído (previsto para o final de Junho) o aluno terá mais subsídios para tentar publicar nas conferências e jornais identificados.

O aluno reconhece que precisa investir mais tempo em relação a isso, mas por outro lado tem a convicção que as publicações são consequência de um bom trabalho de pesquisa científico, e não o objetivo principal de um doutorado. Isso não significa que o mesmo, ignora ou minimiza a importância de tal veículo. Muito pelo contrário, ele acredita que o mesmo é essencial para a validação da relevância científica do trabalho junto à comunidade. Para alcançar o objetivo de realizar publicações relevantes, ele está investindo muito mais tempo na revisão bibliográfica, visto que a literatura sobre o assunto é bastante ampla, e na aplicação prática de uma versão embrionária do modelo de melhoria de produtividade que é objetivo deste trabalho. Assim que tiver resultados mais concretos o foco do trabalho será na escrita e publicação de artigos.

## 4 Referências Relevantes

Esta seção lista todas as referências relevantes relacionadas ao assunto de produtividade que foram lidas pelo aluno durante o período de estudos. Vale a pena comentar que ainda existem aproximadamente umas 50 referências que não estão listadas aqui por que foram identificadas como potencialmente relevantes, mas não foram lidas ainda.

A estratégia de busca das referências seguiu uma lógica, que pode inclusive ser usada para escrita de um *Systematic Review* e que é descrita nos pontos a seguir:

- Busca de referências em portais tais como o IEEE, ACM, Science Direct e até mesmo o Google Scholar, em alguns casos. As buscas foram feitas considerando-se janelas de tempo de 10 anos, desde de 1960, e a utilização do termo “software productivity” e variações incluindo as palavras “measurement”, “improvement”, “strategies” e “factors” como restrição da busca.
- Uma pré-análise foi realizada em cima dos resultados das buscas afim de eliminar as referências não relacionadas.
- Após a leitura de cada um dos artigos, eram compiladas algumas informações sobre os mesmos e registradas inicialmente em planilhas, posteriormente no próprio bibtex.

---

<sup>3</sup><http://prose.cesar.org.br/dokuwiki/doku.php?id=conferences>

- Para cada artigo lido, eram analisadas as citações que potencialmente fossem relacionadas ao estudo e registradas em um backlog para posterior leitura.

#### 4.1 Organização das referências

A seguir estão listadas as referências relevantes lidas sobre o assunto. Como já descrito anteriormente as referências foram organizadas em janelas de 10 anos desde de 1960.

- **Publicações entre 1960 e 1970:** [3, 34, 31, 33, 32, 36, 52, 53, 63, 64, 84, 20, 62]
- **Publicações entre 1971 e 1980:** [1, 4, 25, 37, 38, 48, 82]
- **Publicações entre 1981 e 1990:** [5, 7, 9, 18, 14, 16, 17, 26, 29, 30, 39, 42, 44, 50, 54, 55, 61, 70, 74, 77, 81, 83]
- **Publicações entre 1991 e 2000:** [8, 11, 10, 15, 19, 24, 6, 27, 47, 46, 45, 56, 58, 59, 60, 68, 76, 80, 85, 86]
- **Publicações entre 2001 e 2010:** [2, 12, 13, 21, 22, 23, 28, 35, 40, 41, 43, 51, 57, 65, 66, 67, 69, 72, 78, 79, 87]
- **Publicações Miscelâneas:** [49, 71, 73, 75]

## 5 Apêndice: Survey (*Em andamento*)

### Lessons learned about Software Productivity

#### 1 Introduction

A competitividade empresarial, motivada pela globalização econômica, passou a ser a principal preocupação do mercado. O contexto mundial exige que as empresas melhorem cada vez mais seus níveis de qualidade e diminuam os seus custos de produção. A principal forma de diminuir custos de produção, especificamente no desenvolvimento de software, é através do aumento da produtividade, ou seja, desenvolver mais com um menor custo. Um diferencial competitivo para empresas ou grupos que atuam no desenvolvimento de sistemas é a capacidade de liberar novos produtos ou novas versões do produto com maior agilidade e menor custo, trazendo à tona a questão da produtividade no desenvolvimento de software.

Além disto, a diminuição gradual do custo do hardware, aumento da capacidade de processamento e evolução da sociedade atual, fez com que a demanda pelo desenvolvimento de software venha crescendo de forma acelerada nos últimos anos. Conseqüentemente, a demanda por especialistas no desenvolvimento de software também vem crescendo bastante. Por outro lado, as universidades e centros de formação de capital humano não conseguem formar pessoas na velocidade demanda pela área. Segundo Boehm [10], o setor de software cresce anualmente entre \$300 a \$400 bilhões nos Estados Unidos e entre \$600 e \$800 bilhões no mundo.

Muitos avanços já foram feitos no sentido de melhorar a produtividade no desenvolvimento de software com a pesquisa, definição, padronização e divulgação de boas práticas para tal fim. O problema é que estes estudos exploram as práticas ou técnicas de uma maneira muito pontual. Neste contexto não se tem uma visão global do que pode ser feito no intuito de melhorar a produtividade, nem como os investimentos podem ser priorizados ou organizados de forma a se incorporar produtividade em todo ciclo de desenvolvimento do software, de uma maneira segura, planejada e gradual. De acordo com os resultados do estudo do TRW [16], ganhos significativos em produtividade requerem um programa integrado de iniciativas em diversas áreas. Segundo Vosburgh [81], um programa de melhoria da produtividade para obter sucesso deve endereçar as questões de produtividade como um todo. Características-chaves de tal programa são comprometimento da gerência e uma abordagem integrada. Ainda neste estudo defende-se que uma única melhoria pontual não garante grandes ganhos de produtividade. Um programa de sucesso requer que sejam feitas várias coisas ao mesmo tempo.

A literatura abrangente sobre produtividade confirma o interesse da sociedade científica mundial sobre tal assunto. Muitas questões ainda estão sem respostas e muitos mitos ainda existem sobre o problema da produtividade. As principais questões relacionadas ao assunto e que orientam como o problema deve ser desdobrado e tratado são:

- **Como melhorar a produtividade?**
- **O que afeta a produtividade?**

- **Como se mede a produtividade?**

O objetivo deste artigo é investigar os principais estudos publicados na literatura sobre produtividade no desenvolvimento de software e levantar as principais lições aprendidas reportadas sobre o assunto. Para tal, as lições aprendidas serão categorizadas nas seguintes áreas: Métricas e Medição de Produtividade, na Seção 3; Fatores que afetam a produtividade, na Seção 4; e Estratégias para melhoria da produtividade, na Seção 5.

## **2 The Genesis**

Produtividade tem sido um dos assuntos mais discutidos na área de Engenharia de Software, tanto pela academia quanto pela indústria. Uma diversidade muito grande de estudos endereça o tema e muita evolução no estado da arte já foi realizada. Ainda hoje, após diversos estudos publicados sobre o assunto, existe uma grande quantidade de problemas que se mantém em aberto, e a indústria e academia se mantém em busca de formas de se obter ganhos de produtividade no desenvolvimento de software.

Apesar dessa ênfase em produtividade nos dias atuais, este problema não é novo. Já no início da Engenharia de Software em 1968, na NATO Conference, quando aproximadamente 50 especialistas em computação de 11 países se reuniram em Garmisch, Alemanha, para discutir os problemas de desenvolvimento de software da época [62], a preocupação com a produtividade já era bastante citada. Inclusive durante as discussões foram citados diversos fatores que afetam a produtividade, tais como linguagem de programação, experiência do programador, cultura e motivação, que ainda hoje são bem aceitos pelos estudos na área.

Inclusive pesquisas anteriores já demonstravam a importância do problema. Os primeiros estudos empíricos, baseados em análises estatísticas de dados de projetos de desenvolvimento de software do exército americano, surgiram em meados da década de 60 na System Development Corporation [32, 31]. Estes tinham como objetivo definir fatores que afetavam o custo de projetos de software, afim de melhorar seu processo de estimativa. Apesar destes trabalhos não terem tido o objetivo direto de análise de produtividade, eles podem ser considerados partes deste escopo já que identificaram diversos fatores que afetavam a produtividade, além de quantificarem o nível de influência dos mesmos.

Posteriormente, diversos trabalhos na mesma linha, na System Development Corporation, estenderam os trabalhos anteriores explorando o problema com dados de novos projetos e baseados nos resultados da aplicação prática dos resultados anteriores [34, 33, 84, 63, 36, 52].

Em particular, LaBolle [53] publicou um modelo de estimativa de custos resultados de um extenso estudo, resultado da investigação de 169 projetos finalizados. Neste estudo foram analisados estatisticamente diversos fatores que tinham influência no esforço final de desenvolvimento, ou mais especificamente, na produtividade dos projetos de desenvolvimento de software.

Posteriormente, Pietrasanta [64] citou diversos fatores que afetam a produtividade e discutiu sobre o problema de estimativa de projetos de software, Um estudo conduzido por Aron em um grupo de projetos na IBM's Federal Systems Division (FSD) [3] sugerindo pesquisas quantitativas e análises científicas afim de melhorar o conhecimento sobre o assunto.

Estudos conduzidos por Aron em projetos na IBM's Federal Systems Division (FSD) [3, 4] identificaram que fatores como a complexidade do sistema, a duração do projeto e nível da linguagem de programação tinham influência na produtividade.

Outros estudos mais específicos sobre o problema foram surgindo. A maioria destes estudos eram parte de uma grande busca por métodos de estimativas mais eficazes, que viabilizassem uma maior confiabilidade na previsão dos custos dos projetos de softwares e desta forma evitassem os problemas típicos desta área, que ficaram conhecidos como Crise de Software [62]. Por este motivo eles tinham como foco principal definir métricas para medir a produtividade e identificar os fatores que afetam a produtividade.

Ao longo da história da Engenharia de Software, muitos estudos foram realizados relacionados ao problema da produtividade e como pôde ser visto a mais de 40 anos já se discutia sobre o problema. Apesar de existirem diversos aspectos do problema ainda em aberto, existem, por outro lado, uma gama muito grande de resultados relevantes que demonstram as possibilidades de exploração deste assunto. Por este motivo, esta área de pesquisa se mostra bastante promissora e relevante do ponto de vista científico.

### 3 Productivity Measurement and Metrics

Medição de Produtividade é conceitualmente bastante simples e pode ser definida como a razão entre o que é produzido e o que é consumido para produzir isso, ou seja,  $produtividade = output/input$ . Este conceito é largamente usado em muitas outras áreas, desde agricultura até economia. Apesar da idéia de produtividade ser bastante simples e popular em outras áreas, especificamente no desenvolvimento de software ela é um problema historicamente complexo.

Uma outra forma, bastante usada em alguns trabalhos sobre produtividade [34, 63, 9], de se calcular a produtividade é  $produtividade = input/output$ , ou seja, de forma inversa à apresentada anteriormente. Esta fórmula, à princípio, se mostra paradoxial à idéia de produtividade, pois quanto maior o seu valor, pior será a capacidade de produção (produtividade). Apesar de estranha, esta forma facilita os cálculos em estimativas de esforço, que são comumente feitas através do cálculo  $esforco = tamanho \times produtividade$ .

A dificuldade citada para medição de produtividade no desenvolvimento de software não se dá especificamente pela questão de produtividade em si, mas sim pela característica bem mais fundamental que é concretizar o significado de *output* no contexto de projetos de software, ou seja, quantificar o resultado produzido em um projeto de software em termos de tamanho, complexidade ou valor agregado ao cliente.

Apesar desta dificuldade intrínseca, muito esforço já foi realizado na tentativa de se obter formas de se avaliar a produtividade, seguindo a filosofia da lei de Gilbs [38] que diz que “*Anything you need to quantify can be measured in some way that is superior to not measuring it at all*”. Esta seção irá apresentar uma revisão literária e os principais resultados dos estudos publicados sobre o assunto.

Scacchi [71], reporta em seu estudo sobre métricas de produtividade que é evidente que os estudos atuais são fundamentalmente inadequados e potencialmente falhos. Além disto, dependendo de como e quais indicadores são medidos é possível visualizar melhorias na produtividade de um modo bem granular ou de forma muita superficial.

Baley e Basili [7] não concordam com métricas para comparação de produtividade entre organizações. Eles afirmam que as medidas de produtividade devem ser específicas para a organização.

As principais propostas de métricas de produtividade surgiram junto com os primeiros estudos sobre produtividade em software, já reportados na Seção 2. Os primeiros estudos usavam métricas baseadas em linhas de código (SLOC), como medida de *output*. Com o

tempo outras propostas de medidas foram sendo definidas e usadas como alternativa [1, 39, 87, 68].

Em relação às medidas de *input*, há uma certa convergência sobre o uso de determinadas métricas. As mais comumente usadas são **esforço**, ou seja, a medidade de trabalho investida para se produzir o resultado, e o próprio **capital**, ou seja, a quantidade de recursos financeiros aplicados para produzir os resultados, que uma métrica financeira universalmente usada. Existem algumas variação destas medidas, mas todas baseadas nestes dois princípios.

Em particular iremos analisar e organizar os estudos de métricas de produtividade de acordo com a abordagem de medida de *output* na seções subsequentes.

### 3.1 Output Metrics

A Seção 3.1.1 vai compilar os estudos que usaram SLOC como medida de *output*, a Seção 3.1.2 contem os estudos que usam métricas alternativas, mas ainda assim baseadas no código fonte da aplicação e, por fim, a Seção 3.1.3 já analisa estudos que usam uma visão mais moderna em relação à medida e usam métricas de *output* baseada em valor agregado.

#### 3.1.1 Line of Code Based Metrics

Apesar de SLOC ser algo bastante rudimentar ela foi largamente usada nos estudos sobre produtividade como a medida de *output* na avaliação de produtividade (*produtividade = output/input*).

Os estudos iniciais sobre o assunto [34, 63] usavam uma métrica conceitualmente igual ao SLOC, antecessora ao SLOC per man-month, mas que tem o mesmo princípio. A métrica de *man months per machine instructions*, ou mais especificamente, *man months per 10.000 machine instructions* foi amplamente usada como métrica de produtividade nestes estudos.

Lawrence [55], com o objetivo de avaliar a produtividade de programadores usou a métrica de *SLOC/Hours* em 278 projetos desenvolvidos na Austrália.

Bailey and Basili [7] em sua proposta de um meta-modelo de estimativa de software usaram a métrica de *SLOC/ManMonth* como medida de produtividade em projetos realizados no Software Engineering Laboratory (SEL).

Jones [48] reconheceu que métricas de produtividade e qualidade em termos de linhas de código são paradoxais, já que LOC por unidade de esforço tende a enfatizar mais o tamanho dos programas do que a eficiência e qualidade dos mesmos. Segundo ele LOC não é um bom indicador econômico de produtividade, apesar de ser frequentemente utilizado.

Aron [3, 4] realizou estudos em projetos da IBM usando a métrica de LOC, tanto em sistemas de software quanto aplicações de negócios.

Banker [8] realizou estudos em 65 projetos de manutenção de grandes de sistemas bancários nos EUA e utilizou como medida de tamanho a quantidade de LOC, assim como a quantidade de Pontos de Função.

#### 3.1.2 Others Code Based Metrics

Walston e Felix [82] em seus estudos na IBM Federal Systems Division, sob os dados de 60 projetos, com tamanho variando entre 4000 e 467000 DSL e 12 a 11758, usaram *DSL/MM* como métrica de *produtividade*, onde *DSL* (Delivery Source Line) significa somente as

linhas de código entregues como parte do produto e  $MM$  o esforço total em Homens-Mês (Man-Month).

Boehm [14, 16] fez uso extenso da métrica  $DSI/MM$ , sendo  $DSI$  (Delivery Source Instruction), praticamente o mesmo que DSL, na avaliação da produtividade e na criação de modelos de estimativas de custos de projetos de software.

Vosburgh et al [81] estudaram a produtividade em projetos de grande escala e utilizam a métrica de comandos desenvolvidos (developed statement) por pessoa-ano (personyear).

Grady and Caswell [39] usaram a métrica de  $NCSS/personmonth$  para avaliar a produtividade de projetos na HP. A métrica básica  $NCSS$  é baseada em SLOC e é calculada através da eliminação de comentários e espaços em branco.

Cusumano [26] em seu estudo sobre a produtividade de projetos dos U.S e Japão utilizaram a métrica de *non-comment SLOC* como medida de *output*, conceitualmente igual ao  $NCSS$ , e pessoa-ano (*work-years*) como métrica de *input*.

Arthur [5], por outro lado, propõe o uso de uma métrica de produtividade que reflita melhor a perspectiva financeira do *input*. Neste trabalho ele indica a métrica  $ELOC/\$$  como medida de produtividade, tendo  $ELOC$  (Executable Lines of Code) a mesma semântica de DSL.

Putnam e Myers [68] definiram uma métrica denominada *Process Productivity* (Equação 1), que segundo eles é superior à métrica de produtividade padrão baseada em SLOC. Esta medida de produtividade foi baseada em anos de experiências na gestão de projetos de P&D nas forças armadas americanas e, posteriormente na General Electric Co. e envolveu projetos desenvolvidos nos Estados Unidos, Inglaterra, Austrália e Japão.

$$Process\ Productivity = \frac{Size}{\left(\frac{Effort}{\beta}\right)^{\frac{1}{3}} \times Time^{\frac{4}{3}}} \quad (1)$$

onde,

- $Size$  - É o tamanho do produto em ESLOC (Effective Source Lines of Code);
- $\beta$  - É um fator denominado *Skill Factor* que é função do tamanho do projeto de acordo com a Tabela 1;
- $Effort$  - É o esforço total do projeto em pessoas-ano;
- $Time$  - É o prazo total do projeto em anos.

Por outro lado, Maxwell et al [59] demonstram que a métrica padrão de SLOC é superior à *Process Productivity* em um estudo realizado em um banco de dados de projetos da European Space Agency, contendo 99 projetos de 8 países europeus.

### 3.1.3 Value Based Metrics

Duncan [30], critica o uso de métrica baseada em SLOC como forma de aferir a produtividade. Segundo ele, o processo chave do desenvolvimento de software é a transformação de idéias em produtos. Por isto para se medir a real produtividade do desenvolvimento de software, precisamos conseguir medir quanto efetivamente e eficientemente consegue-se transformar idéias em software.

Size (SLOC)	$\beta$
5-15K	0.16
20K	0.18
30K	0.28
40K	0.34
50K	0.37
> 70K	0.39

Tabela 1: Putnam and Myers Skill Factors Table

Nesta mesma linha de pensamento Albrecht [1] desenvolveu uma técnica de quantificação do tamanho do software que tinha como unidade de medida o ponto de função. O objetivo principal desta medida é contar o tamanho e complexidade do sistema segundo a visão do usuário. Uma das características marcantes da técnica de contagem em ponto de função é que o tamanho do sistema sendo desenvolvido independe da tecnologia, linguagem de programação utilizada ou até mesmo de decisões técnicas como escolha de algoritmos ou estrutura de dados. Esta medida foi inicialmente usada para avaliar a produtividade em 24 diferentes projetos na IBM, variando entre 3000 e 318.000 SLOC desenvolvidos entre 1974 e 1978.

Ponto de Função se tornou uma das medidas mais conhecidas e usadas para se quantificar o tamanho de softwares e conseqüentemente a produtividade. Apesar disto, ela é bastante criticada e questionada pois diversas premissas e definições não possuem um aparato empírico que as valide, como é o caso dos pesos atribuídos às funções de dados e de transações [71, 74].

Behrens [9] também utilizou métricas baseadas em ponto de função para aferir a produtividade em vinte e cinco diferentes projetos desenvolvidos entre 1980 e 1980, com o tamanho variando entre 27 e 599 pontos de função e o esforço entre 600 e 28700 horas. Mais especificamente ele usou a métrica de produtividade com unidade *horas/FP*, onde FP é quantidade de pontos de função.

De forma alternativa, outros autores definiram outras formas, não convencional, de medir a produtividade. Vernon [87], por exemplo, sugere o uso de uma nova métrica, chamada *Change-Point*. Ela foi definida pelo autor como sendo “*The intended result of a set of actions on a single new, modified, affected, or deleted program unit used to materialize a particular AWI*”

D’Amore [27] discute sobre o valor da documentação em um projeto e propõe uma medida, denominada *publication unit*, que captura o tamanho e complexidade de documentos e é usada como medida de *output* para avaliar a produtividade na escrita de documentos em projetos de software.

### 3.2 Compilação dos Resultados

... Será incluída uma tabela resumindo tudo que foi explicado anteriormente

## 4 Productivity Factors

Jones [47] realizou um amplo estudo sobre fatores que influenciam na produtividade em projetos de software. Os fatores que afetam a produtividade foram organizados segundo a natureza

dos projetos (Softwares Embarcados, Sistemas de Informação, Sistemas Militares e Softwares de prateleira). Segundo ele, os fatores e a influência dos mesmos diferenciam bastante dependendo da natureza do projeto.

Um estudo [81] realizado em quarenta e quatro projetos de TI de grandes corporações nos Estados Unidos em 1984 investigou projetos de diferentes áreas de negócio, tais como telecomunicações, engenharia e defesa. Estes projetos eram desenvolvidos geograficamente em nove países e tinham tamanho que variava entre 5.000 e 500.000 comandos. O objetivo do estudo foi identificar que fatores influenciavam na produtividade. O estudo identificou quatorze fatores, que foram categorizados como “os que são controlados pelo gerente” e “os que não estão sobre controle do gerente”. Exemplos de fatores que não estão sobre controle do gerente são Experiência do cliente; Participação do cliente; Complexidade do negócio. Entre os que estão sobre controle do gerente temos: Tamanho da equipe; Processos utilizados; Ferramentas e Técnicas adotadas. Neste estudo foi realizada uma análise multivalorada com aproximadamente cem diferentes variáveis.

Nele foi apontado que fatores relacionados ao produto, tais como complexidade e participação do cliente, têm uma influência de aproximadamente 28% na produtividade de um projeto. Já os fatores relacionados ao projeto, tais como, processo de desenvolvimento, experiência dos desenvolvedores, organização do ambiente e ferramentas utilizadas, têm uma influência de aproximadamente 37%.

Jones [44], aponta mais de quarenta variáveis que podem influenciar na produtividade do desenvolvimento de software. Neste livro ele descreve alguns dos problemas e paradoxos relacionados à produtividade de software.

Já Tom DeMarco em seu trabalho que envolveu a análise de aproximadamente 600 desenvolvedores de 92 organizações [29], defende que os principais problemas de produtividade estão relacionados a fatores humanos e organizacionais e não a fatores técnicos como muitos defendem. Inclusive ele apresenta uma série de experimentos realizados com o objetivo de identificar estes fatores e em seus resultados fica explícita tal afirmação. Nestes experimentos, fatores como linguagem de programação adotada, faixa salarial e experiência na função não influenciam na produtividade. Já outros fatores, tais como espaço físico individual e nível de ruído no ambiente de trabalho afetam de um modo muito intenso.

Scacchi [70] aponta aproximadamente dezoito macro variáveis que influenciam na produtividade, elas englobam aspectos tecnológicos, gerenciais e do ambiente do projeto. Entre os fatores apontados estão Linguagem de Programação, Ferramentas e técnicas de desenvolvimento, reuso e organização do time.

Clincy [23], propõe que existem quatro áreas que mais impactam na habilidade da organização aumentar sua produtividade e diminuir o tempo do ciclo de desenvolvimento. As quatro áreas são (1) Estrutura e clima organizacional, (2) Sistemas de recompensa (definição de como recompensar efetivamente os times com alta produtividade), (3) Processos de desenvolvimento de Software e (4) o uso de ferramentas.

Já Behrens [9], conclui em sua análise realizada em vinte e cinco projetos de desenvolvimento de software que o tamanho do projeto, o ambiente de desenvolvimento e linguagem de programação impactam na produtividade. Em particular, ele concluiu que pequenos times produzem mais código do que grandes times. Boehm et al [17] reportaram que pequenos times completaram seus projetos com uma produtividade 39% maior do que grandes times.

Em seu livro clássico, Brooks [37] relata que a adição de pessoas a um projeto, principalmente os que já estão atrasados, não é uma boa estratégia visto que a necessidade de

comunicação e alinhamento aumenta exponencialmente em função do tamanho da equipe. Simmons [73] confirma que a adição de pessoas ao projeto não aumenta a produtividade global na mesma proporção. Inclusive, segundo ele a partir de um determinado ponto, adicionar pessoas a uma equipe faz com que a produtividade global da equipe diminua.

Baley e Basili [7] comentam sobre vários fatores. Em especial eles concluíram que a alta produtividade está diretamente relacionada com o uso de metodologia de desenvolvimento de software disciplinada.

Boehm [14], em seu largo estudo, sobre os fatores que influenciam no custo de um projeto, se baseia fortemente no tamanho dos projetos e variações dos processos de desenvolvimento de software, para avaliar os efeitos de determinadas variáveis nas várias fases do ciclo de desenvolvimento de software.

Boehm defende que a produtividade no desenvolvimento de software é mais fortemente afetada por aqueles que desenvolvem o sistema e pela forma como estes estão organizados e gerenciados como time. DeMarco [29], reafirma essa tese dizendo que o resultado de um projeto é muito mais função de quem o faz, do que como ele é feito. Scacchi [70] complementa revisando alguns relatórios sobre gerenciamento de grande projetos. Nesta revisão ele confirma que quando projetos são mal gerenciados ou organizados, a produtividade é substancialmente mais baixa do que o normal.

Clinicy [23] sugere o uso de recompensas, financeiramente ou não, para aumentar a motivação dos desenvolvedores, embora tal prática não seja recomendada por [29]. Austin [6] também relata os problemas advindos de tal prática e critica o uso de medição de performance de indivíduos.

A preocupação com a alocação e carreira de pessoas também é bastante reportada na literatura. Segundo Jones [47], a definição de papéis específicos para cada funcionário, de modo que o mesmo possa se especializar em determinada área, pode trazer um aumento de produtividade de 18%, enquanto que a não definição pode diminuir a produtividade em 15%. Em [29], o autor enfatiza a necessidade da existência de especialistas nas empresas.

Segundo Bruckhaus et al. [19] as ferramentas de apoio ao processo podem melhorar ou piorar a produtividade, dependendo do tamanho do projeto e o processo de desenvolvimento adotado. Segundo ele, tais características influenciam em como as ferramentas serão utilizadas e com que frequência.

Estudo de estimativas, de uma forma geral, apontam as principais variáveis que afetam a produtividade. Inclusive alguns destes definem até o grau de influência das mesmas. O exemplo mais popular disso é o modelo de estimativa de custo COCOMO II [15]. Nele são apontadas 22 fatores, divididos em quatro grupos: relacionados com o produto, plataforma, pessoal e projeto.

Ao contrário do que Jones [44] defende, Jiang e Comstock [43], em um survey realizado na base de dados do ISBSG, concluíram que o tipo da linguagem interfere largamente na produtividade final de um projeto.

## 4.1 Compilação dos Resultados

... Nesta seção será colocada uma tabela com a compilação dos fatores e estudos.

## 5 Productivity Improvement Strategies

O problema é que estes estudos exploram as práticas ou técnicas de uma maneira muito pontual. Neste contexto não se tem uma visão global do que pode ser feito no intuito de melhorar a produtividade, nem como os investimentos podem ser priorizados ou organizados de forma a se incorporar produtividade em todo ciclo de desenvolvimento do software, de uma maneira segura, planejada e gradual. De acordo com os resultados do estudo do TRW [16], ganhos significativos em produtividade requerem um programa integrado de iniciativas em diversas áreas. Essas áreas incluem melhorias em ferramentas, metodologias, ambiente de trabalho, educação, gestão, incentivos pessoais e reuso de software. Segundo Vosburgh [81], um programa de melhoria da produtividade para obter sucesso deve endereçar as questões de produtividade como um todo. Características chaves de tal programa são comprometimento da gerência e uma abordagem integrada. Ainda neste estudo defende-se que uma única melhoria pontual não garante grandes ganhos de produtividade. Um programa de sucesso requer que sejam feitas várias coisas ao mesmo tempo.

Boehm [18], especifica três estratégias de melhoria de produtividade, são elas:

1. **Trabalhar mais rápido** – Aumentar a velocidade de produção, através de ferramentas e práticas modernas de desenvolvimento;
2. **Trabalhar melhor** – Planejar e controlar melhor a qualidade, de modo a evitar desperdício e retrabalho;
3. **Evitar trabalhar** – Reusar software e documentação sempre que possível.

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.

Em [58], os autores, baseado nos dados disponibilizados no banco de dados Experience, propõem um modelo de produtividade de acordo com o negócio do projeto.

Chiang [22], defende que a melhoria de produtividade se dá através do balanceamento de três pilares do gerenciamento de software: tecnologia, pessoas e processo.

Clincy [23], em seu survey, concluiu que as áreas que mais impactam na habilidade da organização de aumentar sua produtividade são: estrutura e clima organizacional, sistemas de recompensa, processos de desenvolvimento de software e uso de ferramentas.

Taylor [75] conclui que o foco na melhoria da cultura organizacional pode trazer grandes benefícios na produtividade da organização.

## Referências

- [1] Alan Albrecht. Measuring application development productivity. In I. B. M. Press, editor, *IBM Application Development Symp.*, pages 83–92, October 1979.
- [2] Donald Anselmo and Henry Ledgard. Measuring productivity in the software industry. *Commun. ACM*, 46(11):121–125, 2003.
- [3] J. D. Aron. Estimating resources for large systems. In *In NATO Conference Report on Software Engineering Techniques*, pages 68–79, Rome, Italy, October 1969. Brussels: NATO Science Committee.
- [4] J.D. Aron. Estimating resources for large programming systems. In P. Naur J.M. Buxton and B. Randell, editors, *Software Engineering: Concepts and Techniques*, pages 206–217. Litton Education Publishing, 1976.
- [5] Lowell Jay Arthur. Software productivity and quality measurement. In *ACM '85: Proceedings of the 1985 ACM annual conference on The range of computing : mid-80's perspective*, pages 187–192, New York, NY, USA, 1985. ACM.
- [6] Robert Austin. *Measuring and Managing Performance in Organizations*. Dorset House Publishing Company, 1996.
- [7] John W. Bailey and 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.
- [8] Rajiv D. Banker, Srikant M. Datar, and Chris F. Kemerer. A model to evaluate variables impacting the productivity of software maintenance projects. *Manage. Sci.*, 37(1):1–18, 1991.
- [9] C. A. Behrens. Measuring the productivity of computer systems development activities with function points. *IEEE Trans. Softw. Eng.*, 9(6):648–652, 1983.
- [10] B. Boehm and K. Sullivan. Software economics: status and prospects. *Information and Software Technology*, Volume 41(14):937–946., November 1999.
- [11] Barry Boehm. Managing software productivity and reuse. *Computer*, 32(9):111–113, 1999.
- [12] Barry Boehm. Value-based software engineering: reinventing. *SIGSOFT Softw. Eng. Notes*, 28(2):3, 2003.
- [13] Barry Boehm and Li Guo Huang. Value-based software engineering: A case study. *Computer*, 36(3):33–41, 2003.
- [14] Barry W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
- [15] Barry W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, Ray Madachy, and Bert Steece. *Software Cost Estimation with Cocomo II*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

- [16] Barry W. Boehm, James F. Elwell, Arthur B. Pyster, E. Donald Stuckle, and 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 Computer Society Press.
- [17] Barry W. Boehm, Terence E. Gray, and Thomas Seewaldt. Prototyping versus specifying: a multiproject experiment. *IEEE Trans. Softw. Eng.*, 10(3):290–302, 1984.
- [18] B.W. Boehm. Improving software productivity. *Computer*, 20(9):43–57, September 1987.
- [19] Tilmann Bruckhaus, Nazim H. Madhavji, Ingrid Janssen, and John Henshaw. The impact of tools on software productivity. *IEEE Softw.*, 13(5):29–38, 1996.
- [20] J. N. Buxton and B. Randell, editors. *Software Engineering Techniques: Report of a conference sponsored by the NATO Science Committee, Rome, Italy, 27-31 Oct. 1969, Brussels, Scientific Affairs Division, NATO*. 1970.
- [21] James Westland Cain and Rachel Jane McCrindle. An investigation into the effects of code coupling on team dynamics and productivity. In *COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment*, pages 907–913, Washington, DC, USA, 2002. IEEE Computer Society.
- [22] I. Robert Chiang and Vijay S. Mookerjee. Improving software team productivity. *Commun. ACM*, 47(5):89–93, 2004.
- [23] Victor A. Clincy. Software development productivity and cycle time reduction. *J. Comput. Small Coll.*, 19(2):278–287, 2003.
- [24] G. Coleman and R. O'Connor. Power to the programmer - using measurement to optimise the software process at the individual level. In *Proceedings of 11th European Software Control and Metrics conference*, Munich, April 2000.
- [25] Philip B. Crosby. *Quality Is Free*. Mentor, 1980.
- [26] Michael A. Cusumano and Chris F. Kemerer. A quantitative analysis of u.s. and japanese practice and performance in software development. *Manage. Sci.*, 36(11):1384–1406, 1990.
- [27] Glenn M. D'Amore. Using productivity metrics to manage documentation projects. In *SIGDOC '97: Proceedings of the 15th annual international conference on Computer documentation*, pages 39–44, New York, NY, USA, 1997. ACM Press.
- [28] Daniel P. Delorey, Charles D. Knutson, and Scott Chun. Do programming languages affect productivity? a case study using data from open source projects. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*, page 8, Washington, DC, USA, 2007. IEEE Computer Society.

- [29] Tom DeMarco and Timothy Lister. *Peopleware: productive projects and teams*. Dorset House Publishing Co., Inc., New York, NY, USA, 1987.
- [30] A. S. Duncan. Software development productivity tools and metrics. In *ICSE '88: Proceedings of the 10th international conference on Software engineering*, pages 41–48, Los Alamitos, CA, USA, 1988. IEEE Computer Society Press.
- [31] L. Farr and B. Nanus. Factors that affect the cost of computer programming. Technical Report TM-1447/000/02, System Development Corporation, Santa Monica, California, July 1964.
- [32] L. Farr and H. J Zagorski. Factors that affect the cost of computer programming: A quantitative analysis. Technical Report TM-1447/001/00, System Development Corporation, Santa Monica, California, August 1964.
- [33] L. Farr and H. J. Zagorski. A summary of an analysis of computer programming cost factors. Technical Report TM-1447/002/00, System Development Corporation, Santa Monica, California, January 1965.
- [34] Leonard Farr, Victor LaBolle, and Norman E. Willmorth. Planning guide for computer program development. Technical Report TM-2314/000/00, System Development Corporation, Santa Monica, California, May 1965.
- [35] Stuart Faulk, John Gustafson, Philip Johnson, Adam Porter, Walter Tichy, and Lawrence Votta. Measuring high performance computing productivity. *Int. J. High Perform. Comput. Appl.*, 18(4):459–473, 2004.
- [36] T. Fleishman. Current results from the analysis of cost data for computer programming. Technical Report TM-3026/000/01, System Development Corporation, Santa Monica, California, July 1966.
- [37] Jr. Frederick P. Brooks. *The mythical man-month: Essays on softw.* 1978.
- [38] Tom Gilb. *Software Metrics*. Winthrop Publishers, Cambridge, Mass, 1977.
- [39] Robert B. Grady and Deborah L. Caswell. *Software Metrics: Establishing a Company-Wide Program*. Prentice Hall PTR, 1987.
- [40] R. Groth. Is the software industry's productivity declining? *Software, IEEE*, 21(6):92–94, Nov.-Dec. 2004.
- [41] Talha Javed, Manzil e Maqsood, and Qaiser S. Durrani. A study to investigate the impact of requirements instability on software defects. *SIGSOFT Softw. Eng. Notes*, 29(3):1–7, 2004.
- [42] D. R. Jeffrey. A software development productivity model for mis enviroments. *J. Syst. Softw.*, 7(2):115–125, 1987.
- [43] Zhizhong Jiang and Craig Comstock. The factors significant to software development productivity. In *PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY*, volume 21, January 2007.

- [44] Capers Jones. *Programming productivity*. McGraw-Hill, Inc., New York, NY, USA, 1986.
- [45] Capers Jones. *Assessment and control of software risks*. Yourdon Press, Upper Saddle River, NJ, USA, 1994.
- [46] Capers Jones. The economics of software process improvement. *Computer*, 29(1):95–97, 1996.
- [47] Capers Jones. *Software assessments, benchmarks, and best practices*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.
- [48] T. C. Jones. Measuring programming quality and productivity. *IBM System Journal*, 17(1):115–125, 1978.
- [49] Cem Kaner and Walter P. Bond. Software engineering metrics: What do they measure and how do we know?
- [50] Chris F Kemerer. An empirical validation of software cost estimation models. *Commun. ACM*, 30(5):416–429, 1987.
- [51] Barbara Kitchenham. Software productivity measurement using multiple size measures. *IEEE Trans. Softw. Eng.*, 30(12):1023–1035, 2004. Member-Emilia Mendes.
- [52] V. La Bolle. Development of equations for estimating the costs of computer program production. Technical Report TM-2918/000/00, System Development Corporation, Santa Monica, California, April 1966.
- [53] V. LaBolle. Statistical analysis of computer programming costs. In *SIGCPR '66: Proceedings of the fourth SIGCPR conference on Computer personnel research*, pages 29–38, New York, NY, USA, 1966. ACM.
- [54] G. N. Lambert. A comparative study of system response time on program developer productivity. *IBM Systems Journal*, 23(1):36–43, 1984.
- [55] M. J Lawrence. Programming methodology, organizational environment, and programming productivity. *Journal of Systems and Software*, 2(3):257–269, September 1981.
- [56] Wayne C. Lim. Effects of reuse on quality, productivity, and economics. *IEEE Softw.*, 11(5):23–30, 1994.
- [57] Katrina D. Maxwell. Collecting data for comparability: Benchmarking software development productivity. *IEEE Softw.*, 18(5):22–25, 2001.
- [58] Katrina D. Maxwell and Pekka Forselius. Benchmarking software-development productivity. *IEEE Softw.*, 17(1):80–88, 2000.
- [59] Katrina D. Maxwell, Luk Van Wassenhove, and Soumitra Dutta. Software development productivity of european space, military, and industrial applications. *IEEE Trans. Softw. Eng.*, 22(10):706–718, 1996.
- [60] Steve McConnell. Software quality at top speed. *Softw. Dev.*, 4(8):38–42, 1996.

- [61] Siba N. Mohanty. Software cost estimation: Present and future. *Software: Practice and Experience*, 11(2):103–122, 1981.
- [62] Peter Naur and Brian Randell, editors. *Software Engineering: Report of a conference sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 Oct. 1968, Brussels, Scientific Affairs Division, NATO*. 1969.
- [63] E.A. Nelson. Management handbook for the estimation of computer programming costs. Technical Report AD-A648750, System Development Corporation, October 1966.
- [64] Alfred M. Pietrasanta. Current methodological research. In *Proceedings of the 1968 23rd ACM national conference*, pages 341–346, New York, NY, USA, 1968. ACM.
- [65] Vahe Poladian, Shaw Butler, Mary Shaw, and David Garlan. Time is not money: the case for multi-dimensional accounting in value-based software engineering. In *Proceedings of the 5th Int'l Workshop on Economics Driven Software Engineering Research (EDSER-5)*, Portland, OR, USA, 2003.
- [66] Rahul Premraj, Martin Shepperd, Barbara Kitchenham, and Pekka Forselius. An empirical analysis of software productivity over time. In *METRICS '05: Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS'05)*, page 37, Washington, DC, USA, 2005. IEEE Computer Society.
- [67] Rahul Premraj, Bhekispo Twala, Carolyn Mair, and Pekka Forselius. Productivity of software projects by business sector: An empirical analysis of trends. In *10th IEEE International Software Metrics Symposium (Late Break-in Papers)*, September 2004.
- [68] Lawrence H. Putnam and Ware Myers. *Measures for Excellence: Reliable Software on Time, within Budget*. Prentice Hall Professional Technical Reference, 1991.
- [69] Narayan Ramasubbu and 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.
- [70] W. Scacchi. Managing software engineering projects: A social analysis. *IEEE Transactions on Software Engineering*, 10(1):49–59, 1984.
- [71] Walt Scacchi. Understanding software productivity.
- [72] Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using benchmarking to advance research: a challenge to software engineering. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.
- [73] Dick B. Simmons. Software organization productivity.
- [74] C.R. Symons. Function point analysis: difficulties and improvements. *Software Engineering, IEEE Transactions on*, 14(1):2–11, Jan 1988.

- [75] Bruce Taylor. Organizational culture is important in software productivity, 2005.
- [76] Stephanie Teasley, Lisa Covi, M. S. Krishnan, and Judith S. Olson. How does radical collocation help a team succeed? In *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 339–346, New York, NY, USA, 2000. ACM.
- [77] A. J. Thadhani. Factors affecting programmer productivity during application development. *IBM Systems Journal*, 23(1):19–35, 1984.
- [78] Piotr Tomaszewski and Lars Lundberg. Software development productivity on a new platform: an industrial case study. *Information and Software Technology*, 47(4):257–269, March 2005.
- [79] Masateru Tsunoda, Akito Monden, Hiroshi Yadohisa, Nahomi Kikuchi, and Ken ichi Matsumoto. Productivity analysis of japanese enterprise software development projects. In *MSR '06: Proceedings of the 2006 international workshop on Mining software repositories*, pages 14–17, New York, NY, USA, 2006. ACM.
- [80] III Vernon V. Chatman. Change-points: a proposal for software productivity measurement. *J. Syst. Softw.*, 31(1):71–91, 1995.
- [81] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben, and Y. Liu. Productivity factors and programming environments. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 143–152, Piscataway, NJ, USA, 1984. IEEE Press.
- [82] C. E. Walston and C. P. Felix. A method of programming measurement and estimation. *IBM System Journal*, 16(1):54–65, 1977.
- [83] K. R. Weaver. Measure productivity: use a generally accepted metric. In *APL '89: Conference proceedings on APL as a tool of thought*, pages 377–380, New York, NY, USA, 1989. ACM Press.
- [84] G. F. Weinwurm and H. J. Zagorski. Research into the management of computer programming: A transitional analysis of cost estimation techniques. Technical Report TM-2712/000/00, System Development Corporation, Santa Monica, California, November 1965.
- [85] K. S White. Software engineering management for productivity and quality. In *In: International Conference on Accelerator and Large Experimental Physics Control Systems*, Trieste, Italy, 1999.
- [86] W.D. Yu, D.P. Smith, and S.T. Huang. Software productivity measurements. *Computer Software and Applications Conference, 1991. COMPSAC '91., Proceedings of the Fifteenth Annual International*, pages 558–564, September 1991.
- [87] Seok Jun Yun and Dick B. Simmons. Continuous productivity assessment and effort prediction based on bayesian analysis. In *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, pages 44–49, Washington, DC, USA, 2004. IEEE Computer Society.