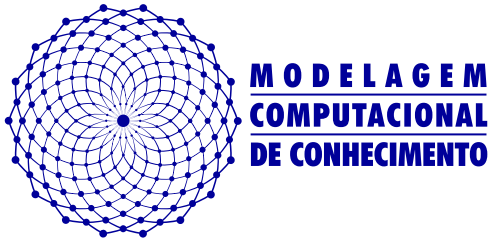


Dissertação de Mestrado

Um estudo para identificar fatores de influência na produtividade de desenvolvedores em projetos de software

Rafael Fernandes Pugliese de Morais
rafaelfpm@gmail.com

Orientadores:

Evandro de Barros Costa
Patrick Henrique da Silva Brito

Coorientador:

Reinaldo Cabral Silva Filho

Maceió, Março de 2018

Rafael Fernandes Pugliese de Morais

Um estudo para identificar fatores de influência na produtividade de desenvolvedores em projetos de software

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Curso de Mestrado em Modelagem Computacional de Conhecimento do Instituto de Computação da Universidade Federal de Alagoas.

Orientadores:

Evandro de Barros Costa

Patrick Henrique da Silva Brito

Coorientador:

Reinaldo Cabral Silva Filho

Maceió, Março de 2018

Catalogação na fonte
Universidade Federal de Alagoas
Biblioteca Central
Bibliotecária Responsável: Janis Christine Angelina Cavalcante

M827e Morais, Rafael Fernandes Pugliese de.

Um estudo para identificar fatores de influência na produtividade de desenvolvedores em projetos de software / Rafael Fernandes Pugliese de Morais. – 2018.
68. : il., grafs., tabs.

Orientadores: Evandro de Barros Costa e Patrick Henrique da Silva Brito.
Coorientador: Reinaldo Cabral Silva Filho.
Dissertação (Mestrado em Modelagem Computacional de Conhecimento) – Universidade Federal de Alagoas. Instituto de Computação. Programa de Pós–Graduação em Modelagem Computacional de Conhecimento. Maceió, 2018.

Bibliografia: f. 64-68.

1. Modelagem computacional. 2. Produtividade de software. 3. Controle estatístico de processos. 4. Regressão linear. 5. Gráfico de pareto. I. Título.

CDU: 004.4

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Modelagem Computacional de Conhecimento
Avenida Lourival Melo Mota, Km 14, Bloco 12, Cidade Universitária
CEP 57.072-900 – Maceió – AL – Brasil
Telefone: (082) 3214-1364/1825

Membros da Comissão Julgadora da Dissertação de Mestrado de Rafael Fernandes Pugliese de Morais, intitulada: "Um estudo para identificar fatores de influência na produtividade de desenvolvedores em projetos de software", apresentada ao Programa de Pós-Graduação em Modelagem Computacional de Conhecimento da Universidade Federal de Alagoas, em 7 de março de 2018, às 14h00min, na sala de reuniões do Instituto de Computação da Ufal.

COMISSÃO JULGADORA

Prof. Dr. Evandro de Barros Costa
Ufal – Instituto de Computação
Orientador

Prof. Dr. Patrick Henrique da Silva Brito
Ufal – Instituto de Computação
Orientador

Prof. Dr. Reinaldo Cabral Silva Filho
Ufal – Pró-Reitoria de Gestão Institucional
Coorientador

Prof. Dr. Arturo Hernández-Domínguez
Ufal – Instituto de Computação
Examinador

Prof. Dr. Baldoino Fonseca dos Santos Neto
Ufal – Instituto de Computação
Examinador

Maceió, março de 2018.

Resumo

Procurando aumentar a competitividade, muitas empresas investem em melhorias em processos de software. A aplicação de melhorias em processos de software trás vários benefícios (Silveira et al., 2013; Simões et al., 2011; Tonini et al., 2008; Bouer & Carvalho, 2005), dentre eles, o aumento na qualidade dos dados de predição. Na literatura (Hamdan & Madi, 2011*a*; Jorgensen, 2011; Lopez-Martin et al., 2012) são identificados vários aspectos negativos por conta de imprecisões em estimativas de projetos de software. Diante deste cenário, este trabalho descreve procedimentos para identificação de fatores que afetam a produtividade em projetos de software modernos, visto que esta medida é fundamental para a elaboração de estimativas de prazo, custo e esforço para projetos. Com a identificação dos fatores que afetam a produtividade de software, é possível entender o comportamento dos desenvolvedores e realizar ações para diminuir a instabilidade da produtividade e consequentemente reduzir os problemas enfrentados por empresas neste contexto, problemas esses que são citados na literatura por autores como Hamdan & Madi (2011*b*), Jørgensen & Sjøberg (2004) e Lopez-Martin et al. (2012). Os procedimentos elaborados utilizam como ferramentas controle estatístico de processo, regressão linear, gráfico de Pareto e são baseados no arcabouço proposto por Florac & Carleton (1999) e Montoni et al. (2007). Os procedimentos foram aplicados em projetos atuais de uma empresa de desenvolvimento de software de Alagoas e os resultados trouxeram benefícios para o processo de desenvolvimento adotado na empresa.

Palavra-chave: Produtividade de software; controle estatístico de processo; regressão linear; gráfico de pareto.

Abstract

Looking to increase competitiveness, many companies invest in improvements in software processes. The application of improvements in software processes brings several benefits (Silveira et al., 2013; Simões et al., 2011; Tonini et al., 2008; Bouer & Carvalho, 2005), such as improvements on process prediction. In literature (Hamdan & Madi, 2011*a*; Jorgensen, 2011; Lopez-Martin et al., 2012) several negative aspects are identified due to inaccuracies in software projects estimates. Given this scenario, this master work describes procedures to identify factors that affect productivity in modern software projects, which is fundamental for the elaboration of time, cost and effort estimation in software projects. By identifying the factors that affect software productivity, it is possible to understand the behavior of developers and perform actions to reduce productivity instability and consequently reduce the problems faced by companies in this context, which are related in literature by authors such as Hamdan & Madi (2011*b*), Jørgensen & Sjøberg (2004) and Lopez-Martin et al. (2012). The elaborated procedures use statistical process control, linear regression, Pareto graph and are based on the framework proposed by Florac & Carleton (1999) and Montoni et al. (2007). The procedures were applied in current projects of a software development company in Alagoas and the results brought benefits to the development process used by the company.

Agradecimentos

Aos meus pais e irmão, Maria Fernandes, Paulo Jorge e Thássio Fernandes, que sempre estiveram comigo me dando forças para enfrentar as dificuldades encontradas.

Á minha esposa, Rosa Luna, por estar ao meu lado, acreditando na minha capacidade, me dando forças, incentivando e acreditando que todas as dificuldades serão superadas. Devido ao companheirismo, amizade e seu amor.

À Maria Júlia, por todo o carinho e contribuição para que este trabalho fosse realizado.

À minha filha, Maria Gabriela, que é minha princesa onde busco inspiração. Pelo seu amor, carinho e alegria que se faz presente em minha vida.

À minha sogra e cunhado, pelo exemplo de superação e apoio que demonstraram quando necessário.

Aos meus familiares, primos, primas, tios e tias, pelo apoio que me deram diretamente e indiretamente quando preciso.

Aos amigos, André, Bruno, David, Felipe, Henrique Gomes, Jefferson Jr., Keven, Levi, Louanderson, Manuel Wagner, Pereira, Ramires e Sávio que demonstraram preocupação pelos momentos difíceis e sempre me incentivaram.

Aos professores do MMCC que me forneceram a base do conhecimento necessária para concluir o mestrado e prosseguir na minha vida profissional.

A Reinaldo Cabral, pela contribuição neste trabalho. Pelos seus ensinamentos, paciência, compreensão e disposição apresentada.

Ao professor Patrick Henrique, pela boa vontade, disposição, ensinamentos e auxílio nos momentos necessários para a conclusão deste trabalho.

Ao professor Evandro, por toda a contribuição e apoio dado para conclusão deste trabalho.

À Inform Sistemas, 4Techlabs e aos envolvidos, pelos momentos de conhecimento e contribuição deste trabalho.

À Universidade Federal de Alagoas, pelas oportunidades que tive, tanto na graduação como no mestrado.

Rafael Fernandes Pugliese de Morais

Sumário

Lista de Figuras

Lista de Tabelas

1 INTRODUÇÃO

1.1 Contexto

A importância de melhorar a predição de dados em organizações de software está aumentando cada vez mais. Realizar predições precisas é importante desde as fases iniciais de um projeto, quando deseja-se realizar estimativas de prazo, custo e esforço para o planejamento de um projeto. A importância de se ter predições de qualidade continua nas demais fases, quando pode-se realizar, por exemplo, estimativas de números de *bugs*. Ter uma boa predição dos dados não é fácil de se obter, porém, estimativas muito imprecisas aumentam as chances de levar projetos ao fracasso (Hamdan & Madi, 2011*b*), fazendo com que gerem frustrações que, em um caso mais extremo, podem inviabilizar a conclusão dos projetos (Jorgensen, 2011).

Na literatura são encontradas diversas consequências que organizações enfrentam por não encontrarem boas estimativas de prazo, custo e esforço. Lopez-Martin et al. (2012) menciona que as estimativas geralmente estão abaixo dos valores necessários para o desenvolvimento de um projeto. Com isso, muitas vezes a participação de equipes em outros projetos são comprometidas devido às equipes passarem mais tempo do que o planejado em projetos que, de acordo com o cronograma, já deveriam ter sido finalizados. Jorgensen & Sjoberg (2004) e Jorgensen (2010) também ressaltam que estimativas erradas podem causar desânimo na equipe e sensação de frustração, além de causar insatisfação dos clientes, especialmente quando o prazo de execução do projeto é superior ao acordado.

Outras consequências decorrentes de estimativas ruins são levantadas por Lopez-Martin et al. (2012). O autor informa que são gerados planejamentos de baixa eficácia, baixa rentabilidade e consequentemente produtos com baixa qualidade (Jorgensen & Sjoberg, 2004). Lopez-Martin et al. (2012) afirma também que quando um projeto é subestimado, pode levar a atribuição de mais recursos sem planejamento, fazendo com que o projeto seja ainda mais custoso para as organizações, podendo influenciar negativamente inclusive em outros projetos.

Em busca de melhorar sua capacidade preditiva, organizações estão cada vez mais buscando a melhoria contínua em seus processos (Barcellos, 2009). Segundo Simões

et al. (2011), melhoria contínua em processos de software é essencial para os gerentes de projetos, uma vez que eles são os responsáveis por realizar previsões de desempenho dos processos.

Garantir uma boa predição dos dados é importante, uma vez que o cliente espera que seu produto seja entregue no prazo acordado e sem custos adicionais (Tonini et al., 2008).

A aplicação de práticas de melhoria contínua em seus processos pode levar organizações a aumentarem o nível de maturidade de seus processos. Silveira et al. (2013) menciona que os níveis de maturidade representam os passos para o aperfeiçoamento em gerenciamento de projeto. Para o autor, os passos estão associados ao domínio da aplicação de ferramentas, processos, metodologias, conhecimentos e habilidades aplicadas em projetos para atingir os objetivos organizacionais. Seguindo na mesma linha, Bouer & Carvalho (2005) descreve que maturidade em organizações pode ser vista como o estado em que uma organização se encontra em perfeitas condições para alcançar seus objetivos. Para o autor, maturidade em projetos também pode indicar como uma organização está condicionada para gerenciar seus projetos.

A cada nível de maturidade alcançado, organizações incorporam em seus modelos novos conceitos e práticas. Barcellos (2009) informa a aplicação de melhoria contínua em organizações de alta maturidade requer conhecimento consistente do comportamento dos processos em suas execuções nos projetos da organização.

Buscando alcançar a alta maturidade, organizações estão utilizando modelos de qualidade como MR-MPS-SW, CMMI (*Capability Maturity Model*) e normas ISO. Segundo Weber et al. (2004) e Montoni (2010), nos últimos anos houve um aumento no número de organizações que estão implementando diretrizes de modelos de qualidade em seus processos. Em geral, organizações iniciam nos níveis mais básicos dos modelos de qualidade, isto é, nível G do MPS-BR e nível 1 do CMMI, e buscam alcançar os níveis mais altos, como o nível A do MPS-BR e o nível 5 do CMMI.

Modelos de qualidade, geralmente possuem como um de seus objetivos melhorar a qualidade dos processos de software (Guia Geral, 2016), tornando organizações mais qualificadas e competitivas.

A busca para alcançar processos de alta maturidade não é uma atividade trivial, uma vez que é necessário lidar com diversos fatores e um número limitado de dados de projetos que constituem a base histórica de empresas. Por conta da limitação da base de dados, organizações buscam iniciar melhorias em processos utilizando medições tradicionais, ou seja, coletando dados de execuções de projetos e realizando comparações destes valores com valores planejados (Barcellos, 2009). Segundo Barcellos (2009), utilizar medições tradicionais não é suficiente para organizações que buscam alta maturidade para seus processos. Para alcançar a alta maturidade, é preciso que as organizações conheçam o comportamento de seus processos, determinem o seu de-

sempenho a partir de execuções anteriores, a fim de posteriormente fazer uma melhor previsão do desempenho de projetos correntes e futuros.

A aplicação de práticas de melhoria contínua em processos requer um esforço concentrado (Weber et al., 2004), entretanto, Salviano (2006) afirma que, apesar do esforço necessário, esta abordagem é viável, eficaz e eficiente para a melhoria de organizações.

Segundo Salviano (2006), existe uma estatística informal que apenas 1 (um) a cada 3 (três) iniciativas de melhorias chegam ao sucesso. Segundo o autor, existem alguns fatores que podem contribuir para o sucesso; alguns desses fatores são citados a seguir, não necessariamente em ordem de importância:

- Comprometimento de toda a equipe envolvida nos processos de melhoria, inclusive da alta gerência;
- Alinhamento das melhorias com os objetivos estratégicos da organização;
- Conhecimento do processo atual existente na organização;
- Entendimento das características, limitações e implicações de abordagens utilizadas;
- Estabelecimento de metas relevantes, viáveis e mensuráveis;
- Considerar melhorias para os aspectos gerenciais, técnicos e humanos;

1.2 Objetivo

Este trabalho tem como proposta aplicar práticas de alta maturidade para identificar os fatores que mais influenciam a produtividade em projetos modernos reais de uma empresa de software, uma vez que a produtividade é um indicador essencial para realizar estimativa de prazo, custo e esforço de projetos.

Com a identificação dos fatores mais influentes da produtividade em um projeto, será possível realizar ações corretivas nos processos de estimativas utilizados atualmente. Tais indicadores também têm grande potencial para ajudar na evolução do processo de desenvolvimento, com o intuito de estabilizá-lo e consequentemente deixá-lo mais previsível.

1.3 Trabalhos Relacionados

Muitos são os autores que investigam os fatores que influenciam a produtividade de um projeto de software. Wagner & Ruhe (2008) realizaram uma revisão sistemática para identificar quais foram os fatores mais listados dentre todos os trabalhos encontrados,

seguindo os critérios estabelecidos pelos autores. Como resultado de sua pesquisa, os autores identificaram 51 fatores, dentre esses fatores, 24 fatores foram classificados como fatores técnicos e 27 como fatores não técnicos.

Diferente da grande maioria dos pesquisadores, DeMarco & Lister (1999) caminha em uma direção "contrária". Segundo o autor, os fatores que influenciam a produtividade não estão relacionados com questões tecnológicas e sim com questões sociológicas. No trabalho de Wagner & Ruhe (2008), são listados os seis fatores (gerenciamento defensivo, burocracia, separação física, fragmentação do tempo das pessoas, redução de qualidade do produto, prazos falsos e controle de cliques) que são considerados os principais por DeMarco & Lister (1999).

Dentre os modelos de estimativas para projetos de software existentes na literatura, dois possuem grande destaque, um deles é o modelo *COCOMO* que foi desenvolvido por *Boehm* (Boehm et al., 1995) e o outro é o modelo de *Putnam*, também conhecido como modelo SLIM (Putnam & Myers, 2013), ambos possuem como foco realizar estimativas de prazo e esforço para um projeto, entretanto, na construção dos respectivos modelos, análises da produtividade foram realizadas.

Segundo Boehm et al. (1995), o modelo *COCOMO* inicialmente possuía 15 fatores de influência nas estimativas de projetos de software. *Boehm* utilizou esses fatores em conjunto com outras variáveis e formulou o modelo *COCOMO*.

O modelo de *Putnam* (Putnam & Myers, 2013) foi elaborado por *Lawrence H. Putnam* e é baseado na *Curva de Rayleigh-Norden*. O modelo realiza análises de produtividade olhando por uma perspectiva de processos de software, realizando comparações e considerações sobre a produtividade com base no modelo que foi gerado pelo autor.

Cunha et al. (2008) realizou uma avaliação da produtividade de projetos de software baseado em uma abordagem multicritério. Nesta avaliação, o autor identificou 14 fatores de influência da produtividade baseado em modelos cognitivos. Além da identificação dos fatores, o autor realizou agrupamentos dos fatores, considerados semelhantes, e elaborou escalas de influência dos fatores, baseado na metodologia MACBETH (Measuring Attractiveness by a Categorical Based Evaluation Technique).

Oliveira et al. (2016) realizou uma análise da produtividade de desenvolvedores focada na perspectiva de gerentes de projeto. Utilizando uma metodologia de pesquisa qualitativa, baseada em entrevistas semiestruturadas, o autor identificou 4 fatores. Segundo Oliveira et al. (2016), os fatores humanos desempenham um papel importante nas percepções dos gerentes sobre a produtividade.

1.4 Estrutura do Trabalho

O restante do trabalho está organizado da seguinte forma:

- **Capítulo 2 - Fundamentação Teórica:** são abordados neste capítulo os conceitos e ferramentas utilizadas ao longo do desenvolvimento deste trabalho.
- **Capítulo 3 - Identificação de fatores que afetam a produtividade de Software:** descreve métodos existentes na literatura para identificação de fatores que influenciam a produtividade de desenvolvimento de *software*. Também é descrito no capítulo aspectos sobre o comportamento da produtividade com relação a outras medidas de *software*.
- **Capítulo 4 - Procedimentos para Identificação dos Fatores que Influenciam a Produtividade:** descreve os procedimentos realizados neste trabalho para identificar os fatores que influenciam a produtividade em projetos de desenvolvimento de software. Os procedimentos são divididos em duas etapas, que são descritas no capítulo.
- **Capítulo 5 - Estudo de Caso:** apresenta os procedimentos descritos no Capítulo 4, aplicados no contexto de projetos reais atuais de uma empresa de desenvolvimento de software de Alagoas.
- **Capítulo 6 - Conclusão e Trabalhos Futuros:** aborda as conclusões e considerações finais sobre os procedimentos realizados e aponta direcionamentos para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Regressão Linear

Um cenário existente no dia-a-dia de organizações de *software*, é a falta de conhecimento necessário para analisar e interpretar os seus dados.

Com o passar do tempo, os dados existentes nas organizações, independente de qual área esteja se falando, estão aumentando (Wheeler, 1993). Segundo Wheeler (1993), para analisar e interpretar os dados, é imprescindível a existência de um método (procedimento), independente da forma que os dados estejam agrupados e apresentados. Pois, caso não exista tal método, as análises e interpretações serão realizadas somente baseadas no conhecimento e experiência de pessoas, assim, as análises e interpretações estarão mais suscetíveis a falhas, pois serão tão boas quanto a experiência de quem estiver realizando-as.

Procedimentos estatísticos são mecanismos que estão sendo utilizados cada vez mais por pesquisadores para analisar, interpretar e realizar predições de dados em diversas áreas, dentre elas, a Engenharia de Software. Segundo Hair et al. (2005), com procedimentos estatísticos pode-se analisar dados gerais, assim como dados específicos, variando de um contexto de negócio, assim como questões técnicas de uma organização. Ainda segundo Hair et al. (2005), muitas organizações adotam a estatística como recurso para construir desde sua estratégia de mercado (*marketing*), até checar a viabilidade de novos produtos ou retorno esperado de um novo empreendimento.

Segundo Montgomery et al. (2015) um dos procedimentos estatísticos amplamente utilizado para a análise de dados, é a *regressão linear*. Através da *regressão linear* é possível elaborar modelos preditivos baseados nas correlações[1] entre variáveis.

Conforme consta em Montgomery et al. (2015), o procedimento utilizado para identificar as correlações entre duas variáveis é chamado de *regressão linear simples*, onde a intenção é explicar a relação existente entre duas variáveis, em outras palavras, podemos dizer que ela busca descrever o comportamento de uma variável chamada

[1] Segundo Hair et al. (2005), correlação é o "valor que mede a força da relação entre uma variável dependente e uma única variável independente quando os efeitos preditivos das demais variáveis independentes no modelo de regressão são removidos."

de *variável dependente* em função de uma outra variável, chamada de *variável independente* (também chamada de variável preditora). Ainda segundo Montgomery et al. (2015), é definido como *regressão linear múltipla* o procedimento para identificação da correlação entre uma variável dependente com outras variáveis independentes.

A *correlação* entre duas variáveis (variável dependente e variável independente) pode ser proporcional ou inversamente proporcional. Para saber se a *correlação* é proporcional ou inversamente proporcional, utiliza-se uma medida chamada de *Coeficiente de Correlação (r)*. O valor de *r* pode variar entre +1 e -1, onde +1 indica uma perfeita relação positiva (correlação proporcional), 0 (zero) indica nenhuma relação entre as variáveis e -1 significa uma relação perfeita negativa (inversamente proporcional) ou reversa entre as variáveis (Hair et al., 2005).

Para encontrar o valor de *r*, a seguinte equação 2.1 é utilizada:

$$Coeficiente\ de\ correlação\ (r) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.1}$$

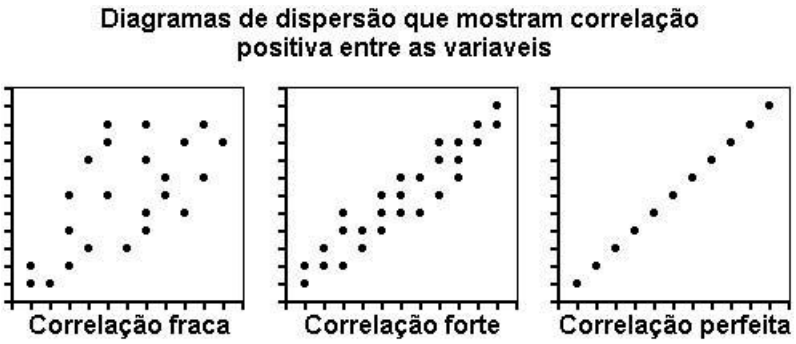
Fonte: Hair et al. (2005)

onde:

x_i = valor individual i da amostra da variável dependente.
y_i = valor individual i da amostra da variável independente.
n = número de amostras.
$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, ou seja, é a média dos valores da variável dependente.
$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$, ou seja, é a média dos valores da variável independente.

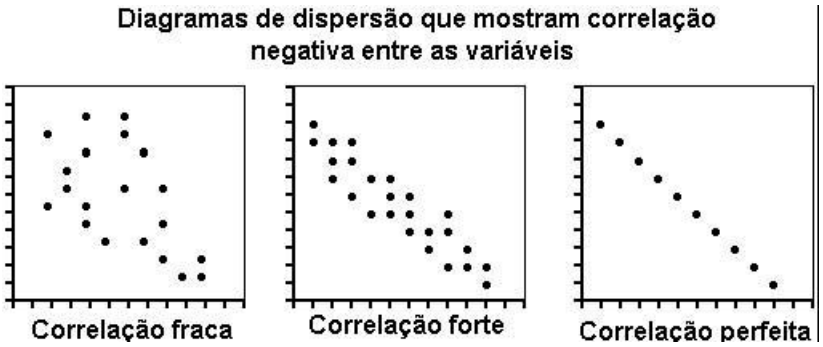
Nas Figuras 2.1 e 2.2 são exibidas correlações entre variáveis em diagramas de dispersão. Em cada diagrama, são informados os tipos de correlações entre as variáveis.

Figura 2.1: Diagramas de dispersão que mostram correlação positiva entre variáveis



Fonte: Adaptada de Hair et al. (2005)

Figura 2.2: Diagramas de dispersão que mostram correlação negativa entre variáveis



Fonte: Adaptada de Hair et al. (2005)

Outra medida utilizada para constatar a relação entre variáveis é o *Coeficiente de Determinação* (R^2). O *Coeficiente de Determinação* representa a variância da variável dependente com relação a variável independente. Os valores de R^2 variam entre 0 e 1, quanto maior for o valor de R^2, maior será o poder de explicação das variáveis, ou seja, mais previsível será o valor da variável dependente Hair et al. (2005).

Utiliza-se a seguinte equação 2.2 para encontrar o valor de R^2:

$$Coeficiente\ de\ Determinação\ (R^2) = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.2}$$

Fonte: Hair et al. (2005)

onde:

n = número de observações.

$\bar{y}$ = média de todas as obervações.

y_i = valor da observação individual i.

$\hat{y}$ = valor previsto da observação i.

Um critério para encontrar resultados adequados é a quantidade de amostras que são utilizadas nas análises, pois o número de amostras influencia diretamente no resultado. Segundo Hair et al. (2005), a razão mínima entre amostras e variáveis deve ser 5 para 1 respectivamente, ou seja, para cada variável independente, deve existir pelo menos 5 amostras. Apesar da existência da proporção mínima, Hair et al. (2005) ainda menciona que a proporção desejada é entre 15 e 20 amostras (observações) para cada variável independente.

2.2 Controle Estatístico de Processo

Controle Estatístico de Processo (CEP) é uma ferramenta relativamente nova no contexto de Engenharia de Software, mas não para a industria em geral. Foi originalmente utilizado na área de manufatura, mais especificamente em melhoria contínua nos processos de linha de produção (Rocha et al., 2012). Tem como objetivo apoiar a análise de processos, afim de realizar prevenções de cenários futuros (cenários que envolvam falhas e defeitos em produtos e/ou serviços), atuar na melhoria da qualidade de produtos e serviços, assim como reduzir custos nas organizações.

Segundo Nomelini et al. (2009), CEP é uma ferramenta simples e sua efetividade é testemunhada por uma repetição fisicamente estabelecida nas indústrias por todo o mundo. O autor complementa que o CEP é uma das mais poderosas metodologias desenvolvidas, visando auxiliar no controle eficaz da qualidade do produto e seus processos produtivos.

Com a aplicação de *Controle Estatístico de Processo*, é possível não só melhorar execuções de projetos futuros, assim como os atuais (Rocha et al., 2012). Para aplicalo é fundamental um acompanhamento muito frequente da execução dos projetos, certamente, quanto mais frequente, mais cedo será a predição de cenários.

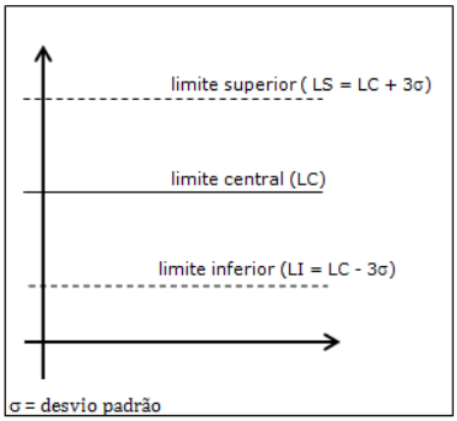
Dentre os vários conceitos existentes no contexto de *Controle Estatístico de Processo*, temos a *Voz do Cliente* e a *Voz do Processo*. Entende-se por *Voz do Cliente* os valores desejáveis pelas organizações e os clientes propriamente ditos, um exemplo de um possível valor para a *Voz do Cliente* é o prazo que o cliente espera que seja atendido ao solicitar um novo projeto de software. Já a *Voz do Processo* é a capacidade que o processo tem de atender os mesmos valores desejáveis pela *Voz do Cliente*. É importante mencionar que o *Controle Estatístico de Processo* trabalha em função da *Voz do Processo* e não da *Voz do Cliente*. Para deixar mais claro o entendimento, segue um exemplo do que seria *Voz do Cliente* e *Voz do Processo*. Digamos que na solicitação de um projeto, o cliente gera a expectativa que o projeto seja entregue em até 90 dias (*Voz do Cliente*), porém ao realizar o planejamento do projeto, o gerente do projeto verificou que seria possível entregar o projeto somente com 150 dias (*Voz do Processo*). Ou

seja, a expectativa gerada pelo cliente (*Voz do Cliente*), pode não ser alcançada pelo desempenho máximo do processo (*Voz do Processo*), embora o ideal é que a *Voz do Processo* consiga alcançar as expectativas geradas da *Voz do Cliente*, desde que estas sejam realistas.

A análise de capacidade do processo envolve os objetivos de qualidade e desempenho definidos pela organização (Barreto, 2011). Segundo Silva Filho (2012), um processo é dito capaz quando: o processo é estável e sua variação não excede os limites de controle. Silva Filho (2012) afirma que quanto menor a variabilidade do processo, maior sua capacidade.

Para entender o comportamento do processo aplicando o *Controle Estatístico de Processo*, é fundamental ter o entendimento sobre gráfico de controle, a Figura 2.3 exibe um exemplo descrito por Rocha et al. (2012).

Figura 2.3: Layout Básico de um Gráfico de Controle



Fonte: Rocha et al. (2012)

A Figura 2.3 exibe as seguintes informações: limite superior ($LS = LC + 3\sigma$), limite inferior ($LI = LC - 3\sigma$) e limite central (LC). Os limites superior e inferior ficam a uma distância de três desvios padrões (σ) da linha central, a linha central por sua vez é a média dos valores que são plotados no gráfico.

Utilizando testes de estabilidade, aplicados ao gráfico de controle, é possível identificar se o processo está sofrendo influência de causas comuns (variações aceitáveis) ou se ele está sofrendo influência de causas especiais. Quando os testes são aplicados e não é encontrada nenhuma violação, é dito que um processo sofre apenas influencia de causas comuns, ou seja, afirma-se que ele está estável (ou repetível), caso contrário, afirma-se que ele está instável, ou seja, existe influencia de causas especiais. Tornar um processo estável, é um dos primeiros passos realizados no *Controle Estatístico de Processo*.

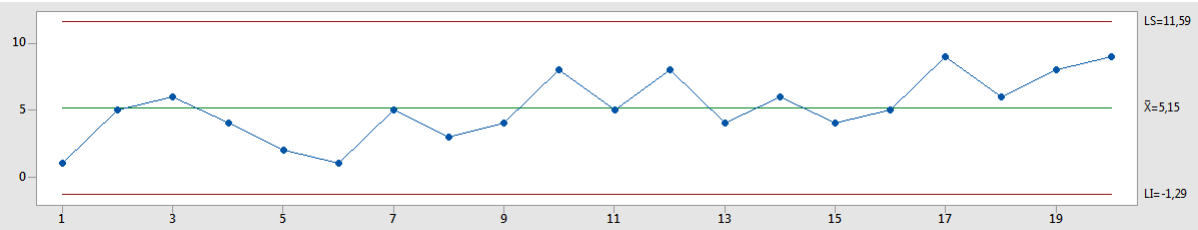
Barcellos (2009) e Chaves Lessa Schots et al. (2014) destacam que um processo

estável não é aquele que não sofre variações, mas sim um processo que apresenta variações aceitáveis e que ocorrem dentro dos limites previsíveis, caracterizando assim, um comportamento repetível.

Silva Filho (2012) menciona um exemplo, existente na literatura, onde demonstra causas comuns e causas especiais que atuam na variação da produtividade de um programador. Segundo o autor, no contexto citado, a produtividade do programador é afetada por causas comuns como capacidade do programador, padrões de programação, linguagem, disponibilização de máquinas e ferramentas. Por outro lado, o autor cita surtos de gripe e baixa taxa de utilização do computador por conta de excessivas quedas de energia como exemplos de causas especiais que podem atuar na produtividade do programador.

A Figura 2.4 ilustra um gráfico de controle onde é possível verificar a estabilidade de um processo, ou seja, onde não há causas especiais.

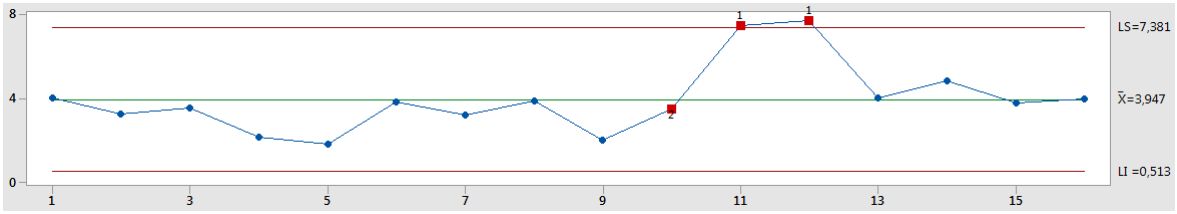
Figura 2.4: Processo estável



Fonte: Documentação do site do Minitab

Na Figura 2.5 é apresentado um gráfico que ilustra um processo cujo comportamento violou testes de estabilidade, sendo identificados pontos cujas causas de variação (causas especiais) devem ser investigadas.

Figura 2.5: Processo com causas especiais



Fonte: Documentação do site do Minitab

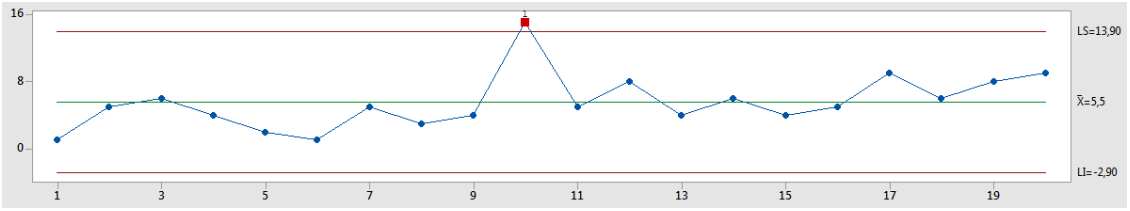
A aplicação de *Controle Estatístico de Processo* apoia análise sobre a estabilidade e sobre a capacidade. Com o *Controle Estatístico de Processo* não é possível identificar o que ocasionou uma variação de causa comum ou variação de causa especial, porém é uma ferramenta que indica onde esta a variação ocorreu para que se possa investigar.

Na literatura são encontrados até 8 testes de estabilidade, entretanto, Wheeler (1993) menciona os teste 1, 5, 6 e 8 como sendo os testes mais efetivos.

- **Teste 1: um ponto a mais do que 3σ da linha central**

  O Teste 1 identifica subgrupos que são atípicos se comparados a outros subgrupos. Esse teste é reconhecido universalmente como necessário para a detecção de situações fora de controle. Se pequenas mudanças no processo forem de interesse, pode-se usar o Teste 2 para suplementar o Teste 1, a fim de criar um gráfico de controle que tenha maior sensibilidade. Um gráfico de controle com violação do teste 1, é apresentado na Figura 2.6.

Figura 2.6: Gráfico de controle representando a violação do teste 1

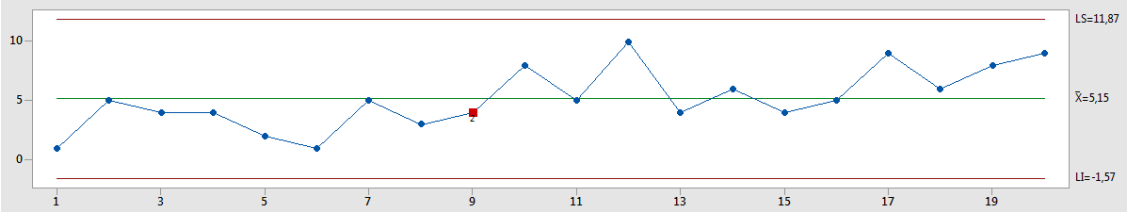


Fonte: Documentação do site do Minitab

- **Teste 2: nove pontos em uma linha no mesmo lado da linha central**

  O Teste 2 identifica mudanças na centralização ou na variação do processo. Se pequenas mudanças no processo forem de interesse, pode-se usar o Teste 2 para suplementar o Teste 1, a fim de criar um gráfico de controle que tenha maior sensibilidade. A Figura 2.7 representa uma gráfico de controle com a violação do teste 2.

Figura 2.7: Gráfico de controle representando a violação do teste 2

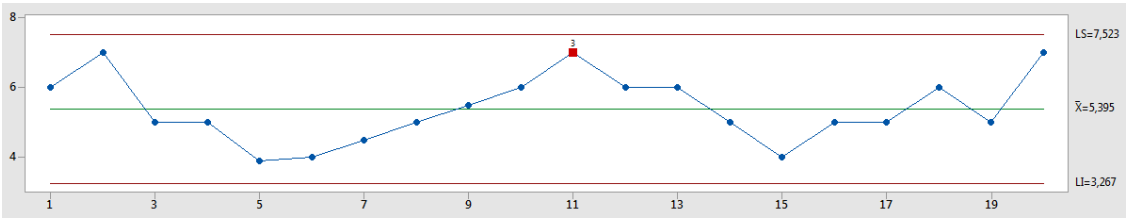


Fonte: Documentação do site do Minitab

- **Teste 3: seis pontos em uma linha, todos crescentes ou todos decrescentes**

  O Teste 3 detecta tendências. Este teste procura uma longa série de pontos consecutivos que aumentam consistentemente em valor ou que diminuem em

Figura 2.8: Gráfico de controle representando a violação do teste 3



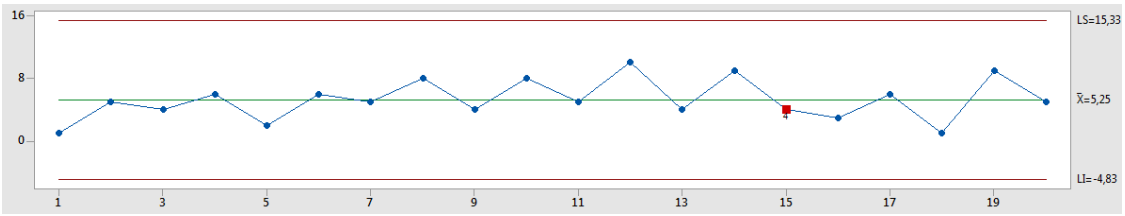
Fonte: Documentação do site do Minitab

valor. Na Figura 2.8 é encontrado um gráfico de controle com a violação do teste 3.

- **Teste 4: quatorze pontos em uma linha, alternando para cima e para baixo**

  O Teste 4 detecta a variação sistemática. Caso deseje-se que o padrão de variação em um processo seja aleatório, mas um ponto que falha no Teste 4 pode indicar que o padrão de variação é previsível. A Figura 2.9 apresenta um gráfico de controle com violação do teste 4.

Figura 2.9: Gráfico de controle representando a violação do teste 4

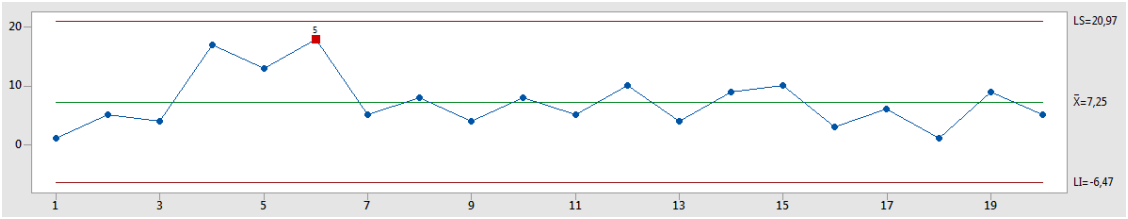


Fonte: Documentação do site do Minitab

- **Teste 5: dois dos três pontos a mais do que 2σ da linha central (mesmo lado)**

  O Teste 5 detecta pequenas mudanças no processo. Um gráfico de controle com violação do teste 5 é encontrado na Figura 2.10.

Figura 2.10: Gráfico de controle representando a violação do teste 5

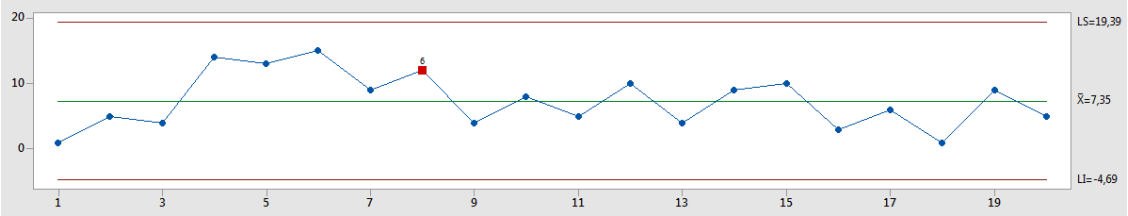


Fonte: Documentação do site do Minitab

- **Teste 6: quatro dos cinco pontos a mais do que 1σ da linha central (mesmo lado)**

  O Teste 6 detecta pequenas mudanças no processo. Na Figura 2.11, é exibido um gráfico de controle com violação do teste 6.

Figura 2.11: Gráfico de controle representando a violação do teste 6

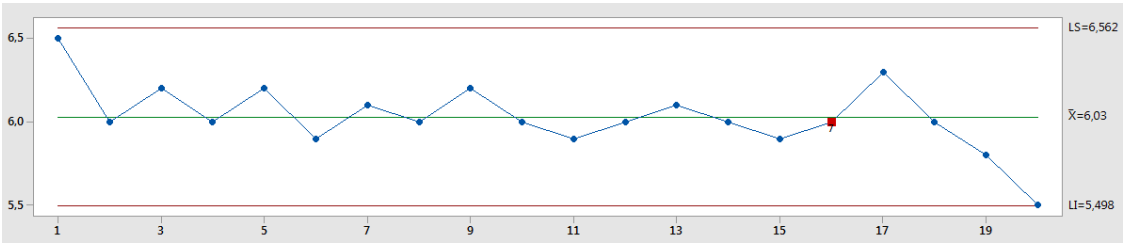


Fonte: Documentação do site do Minitab

- **Teste 7: quinze pontos em uma linha dentro de 1σ da linha central (qualquer dos lados)**

  O Teste 7 detecta um padrão de variação que é algumas vezes confundido como evidência de bom controle. Esse teste detecta limites de controle que são muito amplos. Os limites de controle, que são muito amplos, são frequentemente causados por dados estratificados, que ocorrem quando uma fonte sistemática de variação está presente dentro de cada subgrupo. Na Figura 2.12, é exibido um gráfico de controle com violação do teste 7.

Figura 2.12: Gráfico de controle representando a violação do teste 7

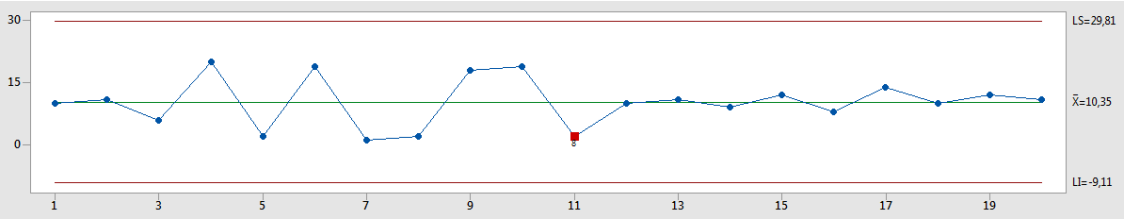


Fonte: Documentação do site do Minitab

- **Teste 8: oito pontos em uma linha a mais do que 1σ da linha central (em qualquer lado)**

  O Teste 8 detecta um padrão de mistura. Em um padrão de mistura, os pontos tendem a se encaixar longe da linha central e, em vez disso, se encaixam perto dos limites de controle. Um gráfico de controle com violação do teste 8 é exibido na Figura 2.13.

Figura 2.13: Gráfico de controle representando a violação do teste 8



Fonte: Documentação do site do Minitab

Existem diversos tipos de Gráficos de Controle, entretanto, todos seguem o mesmo *layout* básico apresentado na Figura 2.3.

Segundo o autor Rocha et al. (2012), para cada tipo de Gráfico de Controle existe um conjunto de expressões que são utilizadas nos cálculos necessários para construção de cada respectivo Gráfico. O autor complementa que nem todos os testes são aplicáveis em todos os tipos de Gráficos e que os gráficos são divididos em duas categorias: *dados de variáveis* e *dados de atributos*.

2.2.1 Gráfico de Controle

Os gráficos de controle são divididos em duas categorias: Gráficos de Controle para Dados de Variáveis e Gráficos de Controle para Dados de Atributos.

2.2.1.1 Gráficos de Controle para Dados de Variáveis

- **Gráficos X-bar e R (XbarR)**

  São utilizados para analisar o comportamento do processo através de subgrupos de medidas obtidas sob as mesmas condições. Enquanto o gráfico X-bar analisa a média dos valores em cada subgrupo, o gráfico R indica a variação entre os subgrupos. Esses gráficos apresentam a limitação de trabalhar apenas com 10 observações no máximo.

- **Gráficos X-bar e S (XbarS)**

  São utilizados nas mesmas condições que os gráficos *XbarR*, porém, não possuem a limitação de trabalhar com no máximo 10 observações. A diferença básica entre os gráficos *XbarS* e *XbarR* está nos cálculos da variação de cada subgrupo. Enquanto os gráficos *XbarR* considera a média dos valores em cada subgrupo, os gráficos *XbarS* considera o desvio padrão de cada subgrupo.

- **Gráficos Individuals and Moving Range (XmR)**

  São utilizados para monitorar apenas uma variável (diferente dos gráficos *XbarR* e *XbarS*). Enquanto o gráfico *X* representa os valores individuais coletados, o gráfico *mR* representa a variação de um valor em relação ao valor anterior, chamada de varição móvel. Para os dois pontos que são utilizados no calculo da variação móvel, é utilizado o termo *two-point moving range*.

- **Gráficos Individuals and Median Moving Range (XMmR)**

  São utilizados nas mesmas condições que os gráficos *XmR*, entretanto, em seus cálculos é utilizada a mediana, ao invés da média. Entende-se que a mediana pode ser mais sensível com relação as causas especiais, assim, utilizando a mediana, nos cálculos, é possível identificar causas especiais que não são identificadas, em alguns cenários, quando a média é utilizada.

- **Gráficos Moving Average and Moving Range (mXmR)**

  São gráficos que além de utilizar a variação móvel, como os gráficos *XmR* e *XMmR*, utilizam também a média móvel (média entre dois valores consecutivos). Assim, possuem como objetivo avaliar as tendências do desempenho dos processos ao longo do tempo.

2.2.1.2 Gráficos de Controle para Dados de Atributos

- **Gráfico C (C Chart)**

  São utilizados para monitorar a ocorrência de eventos em uma mesma área de observação, ou seja, a área de observação utilizada para realizar a monitoração dos eventos deve ser constante.

- **Gráfico U (U Chart)**

  Assim como os gráficos *C*, são utilizados para monitorar a ocorrência de eventos em áreas de observação, entretanto, no gráfico U as áreas de observações devem ser diferentes, ou seja, a área de observação não deve ser constante.

- **Gráfico Z (Z Chart)**

  São utilizados para converter os valores de um gráfico *U* para a escala baseada no desvio padrão (σ). Essa conversão, em alguns casos, pode facilitar a visualização de tendências à instabilidade no comportamento do processo. Após a conversão, os valores estão na unidade *sigma*, os limites inferior e superior são definidos como -3 e 3, respectivamente, e o limite central é definido como zero.

2.3 Gráfico de Pareto

Elaborado por Vilfredo Pareto, a partir de estudos econômicos, o gráfico de Pareto é utilizado para identificar quais são as principais causas atuantes em um problema (Lins, 1993).

Peinado & Graeml (2007) informa que quando existem várias causas para um problema, normalmente, uma ou duas destas causas são responsáveis pela maior parte do problema. Assim, é preferível que ao invés de buscar a eliminação de todas as causas, é possível e prático, inicialmente, agir para eliminar apenas a causa principal.

Também encontrado na literatura como *diagrama de Pareto*, o gráfico de Pareto é composto por outros dois gráficos: gráfico de barras verticais e gráfico de linha. Segundo Lins (1993), cada causa é quantificada em termos da sua contribuição para o problema e é colocada em um gráfico de barras em ordem decrescente de influência ou de ocorrência. O gráfico de linha, por sua vez, apresenta o valor total acumulado.

Por outro lado, Sales (2006) descreve um gráfico de Pareto como sendo um gráfico onde várias classificações de dados são organizadas em ordem decrescente, da esquerda para a direita por barras simples depois de reunir os dados para qualificar as causas. Assim, é possível atribuir uma ordem de prioridades.

A construção de um gráfico de Pareto pode ser observada através de quatro passos descritos em um exemplo colocado por Peinado & Graeml (2007). O autor descreve uma situação onde uma empresa que fabrica e entrega produtos para lojas de varejo, deseja reduzir o número de devoluções. Para isso, a empresa listou o número de ocorrências (vide Tabela 2.1) geradoras de devolução da entrega no último semestre.

Tabela 2.1: Número de casos de devolução de entrega (Peinado & Graeml, 2007)

Razões	**Número de ocorrências**
Separação errada	45
Faturamento incorreto	60
Atraso da transportadora	125
Pedido errado	30
Atraso na entrega	140
Preço errado	20
Produto danificado	65
Outros	15
Total	**500**

Fonte: Peinado & Graeml (2007)

Primeiro passo: realizar a ordenação dos valores por ordem decrescente (com base na coluna *Número de ocorrência*) (vide Tabela 2.2).

Tabela 2.2: Número de casos em ordem decrescente (Peinado & Graeml, 2007)

Razões	**Número de ocorrências**
Atraso na entrega	140
Atraso da transportadora	125
Produto danificado	65
Faturamento incorreto	60
Separação errada	45
Pedido errado	30
Preço errado	20
Outros	15
Total	**500**

Fonte: Peinado & Graeml (2007)

Segundo passo: acrescentar mais uma coluna (*Casos acumulados*) indicando os valores acumulados (vide Tabela 2.3).

Tabela 2.3: Número de casos acumulados (Peinado & Graeml, 2007)

Razões	**Número de ocorrências**	**Casos acumulados**
Atraso na entrega	140	140
Atraso da transportadora	125	265
Produto danificado	65	330
Faturamento incorreto	60	390
Separação errada	45	435
Pedido errado	30	365
Preço errado	20	385
Outros	15	500
Total	**500**	

Fonte: Peinado & Graeml (2007)

Terceiro passo: acrescentar mais uma coluna (*Percentual unitário %*) onde serão colocados os valores percentuais referentes a cada tipo de ocorrência (vide tabela 2.4).

Tabela 2.4: Valores percentuais unitários (Peinado & Graeml, 2007)

Razões	**Número de ocorrências**	**Casos acumulados**	**Percentual unitário %**
Atraso na entrega	140	140	28
Atraso da transportadora	125	265	25
Produto danificado	65	330	13
Faturamento incorreto	60	390	12
Separação errada	45	435	9
Pedido errado	30	365	6
Preço errado	20	385	4
Outros	15	500	3
Total	**500**		100

Fonte: Peinado & Graeml (2007)

O cálculo é feito dividindo-se o número de ocorrências de um determinado tipo pelo total de ocorrências no período. Por exemplo:

% de atraso na entrega = $\frac{140}{500}$ = 0,28 => 28%

Quarto passo: a última coluna (*Percentual acumulado %*) é acrescentada para colocar o % acumulado (vide Tabela 2.5).

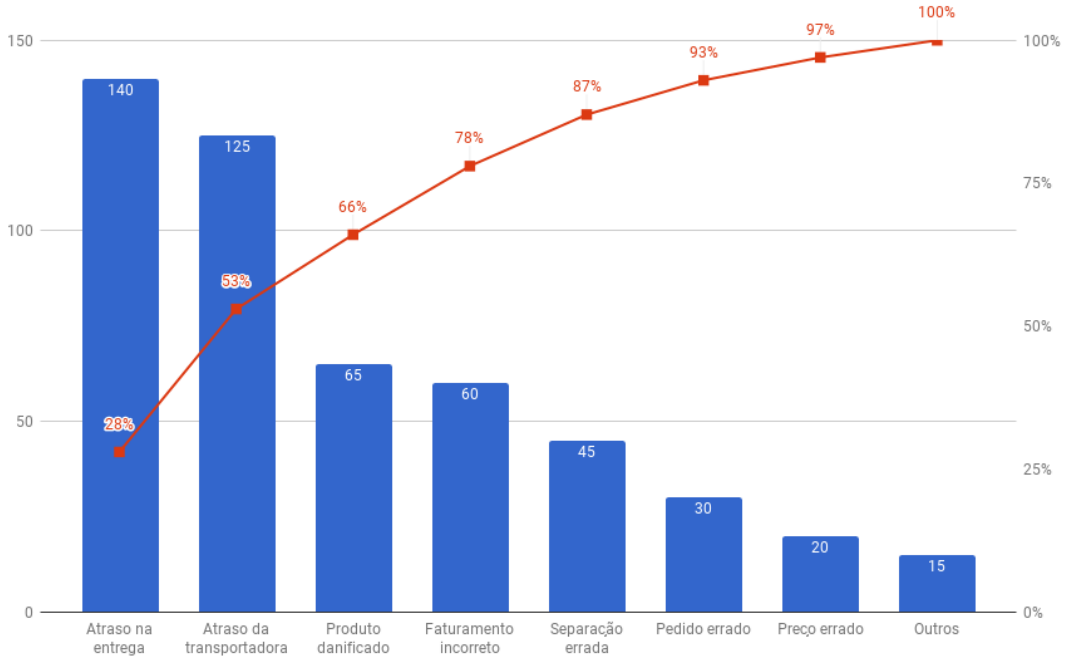
Tabela 2.5: Dados completos para a construção do gráfico de Pareto (Peinado & Graeml, 2007)

Razões	**Número de ocorrências**	**Casos acumulados**	**Percentual unitário %**	**Percentual acumulado %**
Atraso na entrega	140	140	28	28
Atraso da transportadora	125	265	25	53
Produto danificado	65	330	13	66
Faturamento incorreto	60	390	12	78
Separação errada	45	435	9	87
Pedido errado	30	365	6	93
Preço errado	20	385	4	97
Outros	15	500	3	100
Total	**500**		100	

Fonte: Peinado & Graeml (2007)

A partir da Tabela 2.5, o gráfico de Pareto foi construído (vide Imagem 2.14).

Figura 2.14: Gráfico de Pareto dos casos de devolução (Peinado & Graeml, 2007)



Fonte: Peinado & Graeml (2007)

Peinado & Graeml (2007) afirma que observando o gráfico de Pareto é possível concluir que para diminuir o problema de devolução de produtos, será necessário criar um programa de ação para a empresa diminuir os atrasos de entrega da fábrica e da transportadora. Com a realização apenas desta ação, observa-se que 53% do problema será resolvido.

3 IDENTIFICAÇÃO DE FATORES QUE AFETAM A PRODUTIVIDADE DE SOFTWARE

3.1 Modelo de Custo Construtivo (COCOMO)

A preocupação em realizar estimativas de custos de softwares baseadas em especialistas, foi a motivação para *Boehm* elaborar o Modelo de Custo Construtivo (COnstructive COst MOdel - COCOMO) (Menzies et al., 2016). Utilizando uma abordagem empírica em dados dos projetos da TRW, *Boehm* construiu a primeira versão do modelo utilizando 63 projetos de software (Nguyen et al., 2010; Gulezian, 1991; Boehm, 1984; Nguyen et al., n.d.) de diversas áreas, diversas linguagens de programação e que variavam entre 2.000 e 100.000 linhas de código (LOC) (Teixeira & Sanches, 2000).

Até o momento, são encontradas na literatura duas versões do COCOMO. A primeira versão (COCOMO I), também encontrado na literatura como COCOMO 81, durou em torno de 15 anos (1964 - 1979) e foi oficialmente publicada em 1981 (Teixeira & Sanches, 2000). Devido a estar obsoleto por não atender as necessidades de ciclos interativos e lidar com componentes de terceiros, que estavam sendo utilizados cada vez mais a partir da década de 90 durante o desenvolvimento de projetos de softwares, foi elaborada a versão COCOMO II. *Boehm* e a *University of Southern California*, com o apoio de diversas empresas, elaboraram o COCOMO II utilizando um conjunto de 161 projetos (Menzies et al., 2016).

As duas versões (COCOMO I e COCOMO II) possuem o mesmo objetivo: realizar estimativas de custo, prazo, esforço e tamanho da equipe para desenvolvimento de um projeto de software Haufe (2001).

O modelo COCOMO é baseado em equações matemáticas e aplicação de métodos estatísticos. Segundo Teixeira & Sanches (2000) para obter os primeiros parâmetros das equações do modelo COCOMO, *Boehm* utilizou opiniões subjetivas de gerentes experientes e combinação de resultados de outros modelos. Posteriormente os parâmetros

foram refinados e calibrados utilizando um subconjunto da base de dados (Teixeira & Sanches, 2000).

Na elaboração do COCOMO I, *Boehm* utilizou 15 fatores[1] de software. Segundo Teixeira & Sanches (2000) os 15 fatores foram extraídos dos 29 fatores identificados em Walston & Felix (1977). A seleção dos fatores foi realizada considerando o alto grau de correlação com os custos de um projeto de software. Já na segunda versão, o COCOMO II, foram utilizados 23 fatores (Nguyen et al., n.d.).

Segundo Nguyen et al. (n.d.), os fatores de software são utilizados para realizar a calibração do modelo COCOMO e consequentemente aumentar a predição das estimativas de custo do projeto de software. Ainda segundo Nguyen et al. (n.d.), a literatura informa que modelos que utilizam calibração com dados locais, são mais precisos do que modelos que utilizam dados genéricos.

Segundo Menzies et al. (2016), cada fator é classificado em até 5 níveis propostos por *Boehm*. Posterior a classificação de todos os fatores, os valores desses fatores são multiplicados e o resultado ($m(X)$) dessas multiplicações é utilizado na equação 3.1 (Teixeira & Sanches, 2000).

$$E = a_i \cdot S^{b_i} \cdot m(X) \tag{3.1}$$

Onde:

$E =$ Esforço estimado em pessoa mês.
a_i e $b_i =$ São constantes relacionadas com o nível[2] e modelo[3] de projetos de software.
$S =$ Tamanho do software estimado em linhas de código[4].
$m(X) =$ Função multiplicadora[5] de todos os fatores de software após classificação.

Nas tabelas 3.1 e 3.2 são apresentados os valores dos fatores de software, em cada respectivo nível, utilizados na elaboração do modelo COCOMO para as versão COCOMO I e COCOMO II respectivamente.

[1] Fatores de software são características identificadas nos projetos de software

[2] Segundo Boehm (1984), foram definidos três níveis de projetos de software: básico, intermediário e avançado

[3] Segundo Boehm (1984), foram definidos três modelos de projetos: orgânico, embutido e semidestacado

[4] Na contagem de linhas de código não são considerados os comentários, somente as linhas de código propriamente ditas.

[5] Para os projetos identificados como *tipo básico*, a função multiplicadores ($m(X)$) sempre deverá ser igual a 1.

Tabela 3.1: Fatores do COCOMO I

Fatores	Muito Baixo	Baixo	Nominal	Alto	Muito Alto	Extremamente Alto
ACAP	1.46	1.19	1.00	0.86	0.71	-
PCAP	1.42	1.17	1.00	0.86	0.70	-
AEXP	1.29	1.13	1.00	0.91	0.82	-
MODP	1.24	1.10	1.00	0.91	0.82	-
TOOL	1.24	1.10	1.00	0.91	0.83	-
VEXP	1.21	1.10	1.00	0.90	-	-
LEXP	1.14	1.07	1.00	0.95	-	-
DATA	-	0.94	1.00	1.08	1.16	-
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TURN	-	0.87	1.00	1.07	1.15	-
VIRT	-	0.87	1.00	1.15	1.30	-
STOR	-	-	1.00	1.06	1.21	1.56
TIME	-	-	1.00	1.11	1.30	1.66
RELY	0.75	0.88	1.00	1.15	1.40	-
SCED	1.23	1.08	1.00	1.04	1.10	-

Tabela 3.2: Fatores do COCOMO II

Fatores	Muito Baixo	Baixo	Nominal	Alto	Muito Alto	Extremamente Alto
PREC	6.20	4.96	3.72	2.48	1.24	-
FLEX	5.07	4.05	3.04	2.03	1.01	-
RESL	7.07	5.65	4.24	2.83	1.41	-
TEAM	5.48	4.38	3.29	2.19	1.10	-
PMAT	7.80	6.24	4.68	3.12	1.56	-
RELY	0.82	0.92	1.00	1.10	1.26	-
DATA	-	0.90	1.00	1.14	1.28	-
CPLX	0.73	0.87	1.00	1.17	1.34	1.74
RUSE	-	0.95	1.00	1.07	1.15	1.24
DOCU	0.81	0.91	1.00	1.11	1.23	-
TIME	-	-	1.00	1.11	1.29	1.63
STOR	-	-	1.00	1.05	1.17	1.46
PVOL	-	0.87	1.00	1.15	1.30	-
ACAP	1.42	1.19	1.00	0.85	0.71	-
PCAP	1.34	1.15	1.00	0.88	0.76	-
PCON	1.29	1.12	1.00	0.90	0.81	-
AEXP	1.22	1.10	1.00	0.88	0.81	-
PEXP	1.19	1.09	1.00	0.91	0.85	-
LTEX	1.20	1.09	1.00	0.91	0.84	-
TOOL	1.17	1.09	1.00	0.90	0.78	-
SITE	1.22	1.09	1.00	0.93	0.86	0.80
SCED	1.43	1.14	1.00	1.00	1.00	-

Como perceptível nas tabelas 3.1 e 3.2, alguns fatores entre a versão COCOMO I e COCOMO II foram removidos e outros adicionados, segundo Selby (2007); Boehm et al. (1995), os critérios para essas mudanças entre os fatores da versão COCOMO I para a versão COCOMO II foram:

- **Continuidade:** A menos que exista algum fator forte de lógica, as escalas de classificação e os multiplicadores de esforço do COCOMO II são consistentes com a versão anterior.
- **Parsimonia:** Os fatores estão incluídos no modelo de linha de base do COCOMO II somente se houver uma forte justificativa que explicariam de forma independente uma fonte significativa de esforço de projeto ou variação da produtividade.

Para melhor compreensão de cada fator, todos os fatores das versão COCOMO I e COCOMO II foram descritos.

- **ACAP - Analyst Capability**

  Capacidade do Analista: fator relacionado com a capacidade do análise e o design, eficiência e rigor nas definições, e a capacidade de se comunicar e cooperar com o projeto.

- **PCAP - Programmer Capability**

  Capacidade do programador: fator relacionado com a capacidade, eficiência e rigor nas definições, e a capacidade de se comunicar e cooperar com o projeto. Esse fator leva em consideração a equipe e não os indivíduos de forma específica.

- **AEXP - Applications Experience**

  Experiência em aplicativos: avalia o nível de experiência em aplicativos da equipe do projeto. Essa experiência engloba diversas situações, como: experiência com plataforma, recursos gráficos de interface de usuário, banco de dados, redes e *meddleware distribuídos*.

- **MODP - Use of Modern Programming Practices**

  Uso de práticas modernas de programação: segundo Boehm et al. (1995), a definição de "práticas modernas de programação"evoluiu para um termo mais amplo de "práticas maduras de engenharia de software", exemplificado pelo modelo de qualidade *Capability Maturity Model Integration* (CMMI).

- **TOOL - Use of Software Tools**

  Uso de ferramentas de software: captura o impacto na produtividade das ferramentas que vão desde ferramentas simples de edição e código até ferramentas de suporte integradas e pró-ativas.

- **VEXP - Virtual Machine Experience**

  Experiência da Máquina Virtual: esse fator representa a experiência da equipe do projeto com relação a complexidade de hardware e software que o produto de software exige para realizar suas tarefas, como exemplo, Boehm et al. (1995) cita: sistema operacional e / ou sistema de gerenciamento de banco de dados.

- **LEXP - Language and Tool Experience**

  Experiência em linguagem e ferramenta: fator referente a experiência na linguagem de programação e ferramentas de software utilizadas pela equipe.

- **DATA - Data Base Size**

  Tamanho da base de dados: captura o efeito que os grandes requisitos de dados têm no desenvolvimento do produto.

- **CPLX - Product Complexity**

  Complexidade do produto: caracteriza a complexidade do produto em cinco áreas; Operações de controle, operações computacionais, operações dependentes do dispositivo, operações de gerenciamento de dados e operações de gerenciamento de interface do usuário. Este fator contempla as várias mudanças que ocorreram (entre a versão COCOMO I e COCOMO II) com relação as tecnologias e aplicações.

- **TURN - Computer Turnaround Time**

  Tempo de resposta do computador: segundo Boehm et al. (1995), o tempo de resposta do computador foi um fator significativo durante o período inicial de calibração COCOMO na década de 1970, já que muitos desenvolvedores de software utilizavam computadores com processamento em lote. Atualmente, os computadores realizam processamentos utilizando *thread*. Como resultado, o fator TURN perdeu a maior parte do seu significado e não foi utilizado no COCOMO II.

- **VIRT - Virtual Machine Volatility**

  Volatilidade da Máquina Virtual: Isso reflete o nível de volatilidade da máquina virtual subjacente ao produto de software a ser desenvolvido. A máquina virtual é definida como o complexo de hardware e software que o produto recorrerá para realizar suas tarefas.

- **STOR - Main Storage Constraint**

  Restrição de armazenamento: fator relacionado com o grau de restrição de armazenamento principal imposta a um sistema de software. Boehm et al. (1995), com o aumento notável no tempo de execução do processador disponível e no armazenamento principal, pode-se questionar se essas variáveis de restrição ainda são

relevantes. No entanto, muitas aplicações continuam a expandir para consumir quaisquer recursos disponíveis, tornando esses drivers de custo ainda relevantes.

- **TIME - Execution Time Constraint**

  Restrição de tempo: Medida da restrição de tempo de execução imposta a um sistema de software.

- **RELY - Required Software Reliability**

  Confiabilidade de software exigida: grau de segurança para o qual o software executará a função pretendida.

- **SCED - Required Development Schedule**

  Calendário de desenvolvimento necessário: mede a restrição de programação imposta à equipe do projeto.

- **PREC - Precedentedness of Application**

  Procedência da Aplicação: este fator considera a diferença existentes entre os tipos de projetos colocados por Boehm (1984), segundo o autor, existem três tipos: orgânico, semidestacado e embutido.

- **FLEX - Development Flexibility**

  Flexibilidade de desenvolvimento: este fator considera a flexibilidade do desenvolvimento para atender as necessidades do projeto, necessidades essas que estão relacionadas com os tipos (orgânico, semidestacado e embutido) de projetos colocados por Boehm (1984).

- **RESL - Risk Resolution**

  Resolução de Risco: fator relacionado com a capacidade de solucionar os riscos identificados para o projeto.

- **TEAM - Team Cohesion**

  Equipe coesa: fator que contempla o quanto a equipe está empenhada e sincronizada para desenvolvimento do projeto. Neste contexto, considera-se parte da equipe: usuários, clientes, desenvolvedores e outros participantes que façam parte de forma direta ou indireta do projeto.

- **PMAT - Equivalent Process Maturity Level**

  Nível Equivalente de Maturidade do Processo: fator que representa a classificação dos processos com relação ao CMMI.

- **RUSE - Required Reusability**

  Reutilização Requerida: Contas para o esforço adicional necessário para construir componentes destinados a reutilização nos projetos atuais ou futuros.

- **DOCU - Documentation match to life-cycle needs**

  Correspondência de documentação às necessidades do ciclo de vida: avaliado em termos de adequação da documentação do projeto às suas necessidades de ciclo de vida.

- **PVOL - Platform Volatility**

  Volatilidade da Plataforma: volatilidade do complexo de hardware e software (Sistema Operacional - SO, Sistema de Gerenciamento de Banco de Dados - SGBD, etc.) que o produto de software exige para executar suas tarefas.

- **PCON - Personnel Continuity**

  Continuidade do pessoal: volume de negócios anual do pessoal do projeto.

- **PEXP - Platform Experience**

  Experiência de Plataforma: Experiência em usar o complexo de hardware e software (Sistema de Gerenciamento de Banco de Dados - SGBD, etc.) que o produto de software exige para executar suas tarefas.

- **SITE - Multisite Development**

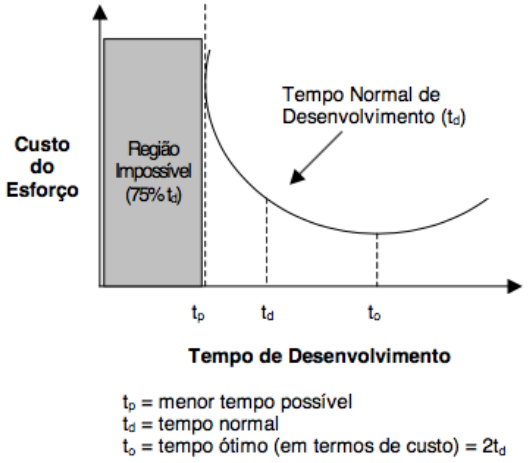
  Desenvolvimento Geograficamente Distribuído: distribuição geográfica das equipes e toda a complexidade relacionada, como por exemplo a comunicação.

3.2 Modelo de Putnam

Buscando aperfeiçoar a atividade que realiza estimativas de projetos de software, realizadas geralmente por líderes de projetos, na década de 70, por volta de 1978, *Lawrence H. Putnam* elaborou um modelo baseado na *Curva de Rayleigh-Norden* (ver figura 3.1).

Utilizando múltiplas variáveis, o modelo de *Putnam* busca estimar o esforço e o tempo necessário para um projeto de desenvolvimento de software. Para chegar na equação utilizada em seu modelo, *Putnam* inicialmente realizou análises entre dois relacionamentos (Putnam & Myers, 2013): (i) tempo com tamanho e (ii) esforço com tamanho.

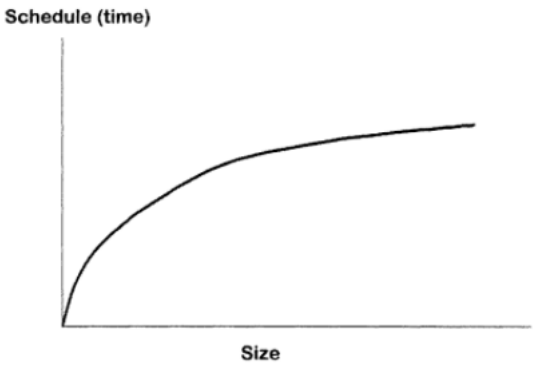
Figura 3.1: Curva de Rayleigh



Fonte: Putnam & Myers (2013)

- **Tempo com tamanho:** segundo Putnam & Myers (2013), na relação entre tempo de programação e tamanho, a medida que uma variável aumenta a outra também aumenta no mesmo sentido, porém não proporcionalmente. Segundo o autor, a medida que o esforço aumenta, o tempo também aumenta, porém de forma mais lenta. Expressando a relação entre as variáveis (tempo e tamanho) em um plano cartesiano, a ligação entre os pontos não seria uma reta, mas sim uma curva côncava para baixo, conforme exibido na figura 3.2.

Figura 3.2: Tempo (eixo y) X Tamanho (eixo x)

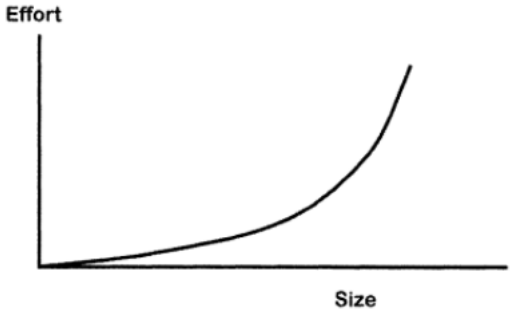


Fonte: Putnam & Myers (2013)

- **Esforço com tamanho:** na relação entre esforço e tamanho, Putnam & Myers (2013) afirma que a medida que uma variável aumenta a outra aumenta na direção contrária, porém não sendo um aumento proporcional. A medida que o tamanho aumenta, o esforço aumenta na direção contrária, porém de forma mais

lenta. Exibindo a relação entre as variáveis (esforço e tamanho) em um plano cartesiano, a relação será expressa em uma curva côncava para cima, conforme exibido na figura 3.3.

Figura 3.3: Esforço (eixo y) X Tamanho (eixo x)



Fonte: Putnam & Myers (2013)

Após realizar análises das relações entre de tempo com tamanho e esforço com tamanho, *Putnam* chegou a seguinte equação (Putnam, 1978; Putnam & Myers, 2013):

$$Tamanho = (Esforço/\beta)^{(1/3)} \cdot Tempo^{(4/3)} \cdot (produtividade\ do\ processo) \quad (3.2)$$

Onde:

$Tamanho =$ Variável que expressa o tamanho do software, *Putnam* utilizou o tamanho de um software expresso em linhas de código-fonte(LOC) em seu modelo.

$Esforço =$ Quantidade de horas utilizadas em pessoas-ano.

$\beta =$ Fator de habilidade (com relação a tecnologia) que é utilizado de acordo com o tamanho do projeto.

$Tempo =$ Cronograma total do projeto.

$Produtividade\ do\ processo =$ produtividade mensurada pela razão entre linha de código e pessoa mês.

A equação apresentada em seu modelo foi elaborada realizando calibrações em 200 projetos, onde, desses 200, 50 projetos eram do comando do exército (Sage & Rouse, 2011). Putnam & Myers (2013) afirma que o modelo foi validado na prática utilizando dados de centenas de empresas e milhares de estimativas. O autor complementa que

após a aplicação do modelo em dados de projetos que foram realizados por organizações independentes e que nem sabiam da existência do modelo, foram obtidos resultados que seguiram a equação muito de perto. Ainda segundo o autor, a equação trás resultados aproximados, não exatos, pois ela não é uma lei da natureza e nem física.

Em seu livro (Putnam & Myers, 2013), *Putnam* informa que a produtividade é melhor convencionada na literatura como sendo a quantidade de linhas de código-fonte entregues por pessoa-mês e a classifica em seu livro como sendo a *produtividade convencional*. Segundo Putnam & Myers (2013), o autor ressalta que não é confiável utilizar a *produtividade convencional* em projetos de software, pois dentre vários motivo, para o autor a produtividade de cada desenvolvedor não é igual e pode variar em até 10 vezes de um programador para outro, ocasionando assim grande variação.

Em Putnam & Myers (2013), é definido o conceito de *produtividade do processo*, o autor define a *produtividade do processo* como sendo a reorganização da equação do modelo de *Putnam* em função da produtividade, assim, tem-se a seguinte equação:

$$produtividade\ do\ processo = \frac{Tamanho}{(Esforço/\beta)^{(1/3)} \cdot Tempo^{(4/3)}} \tag{3.3}$$

Para o autor, visto que a *produtividade do processo* e *tamanho* são variáveis constantes no momento de um planejamento de um projeto, as variáveis de *esforço* e *tempo* devem ficar em equilíbrio, a medida que o valor de uma variável aumenta (considerando as potências de cada variável), a outra deve diminuir para que a equação seja respeitada. Caso as alterações dos valores dessas variáveis (esforço e tamanho) sejam realizadas sem esse conhecimento, certamente problemas no *esforço* ou *tempo* de projetos serão encontrados.

Após a definição da *produtividade do processo*, algumas considerações foram colocadas em Putnam & Myers (2013). Uma delas é que o autor menciona que as pessoas tiveram dificuldade para visualizar que a *produtividade do processo* é influenciada pelo planejamento inicial do projeto de software, uma vez que a *produtividade do processo* é uma função dependente do tamanho, esforço e tempo estimado.

Reorganizando a equação do modelo de *Putnam* em função do tamanho pelo esforço (produtividade convencional), tem-se:

$$\frac{(produtividade\ do\ processo)^3 \cdot Tempo^4 \cdot \beta}{Tamanho^2} = \frac{Tamanho}{Esforço} = produtividade\ convencional \tag{3.4}$$

Segundo *Putnam*, com essa reorganização da equação algumas conclusões e comparações (entre a *produtividade convencional* e *produtividade do processo*) são identificadas mais facilmente:

- Devido a potência de valor 3 da *produtividade do processo*, qualquer aumento,

mínimo que seja, na *produtividade do processo* a *produtividade convencional* terá um aumento significativo, entretanto, qualquer diminuição na *produtividade do processo*, a *produtividade convencional* também terá uma grande queda.

- Quanto menor for o cronograma do projeto, a produtividade convencional tende a ser menor ainda, visto que o tempo do projeto possui uma potência de valor 4.
- Diferente da *produtividade do processo* e do *tempo* previsto para um projeto, o tamanho do software atua de forma inversa na *produtividade convencional*, ou seja, quanto maior o tamanho do software, menor será a *produtividade convencional* e quanto menor o tamanho do software, maior será a *produtividade convencional*.

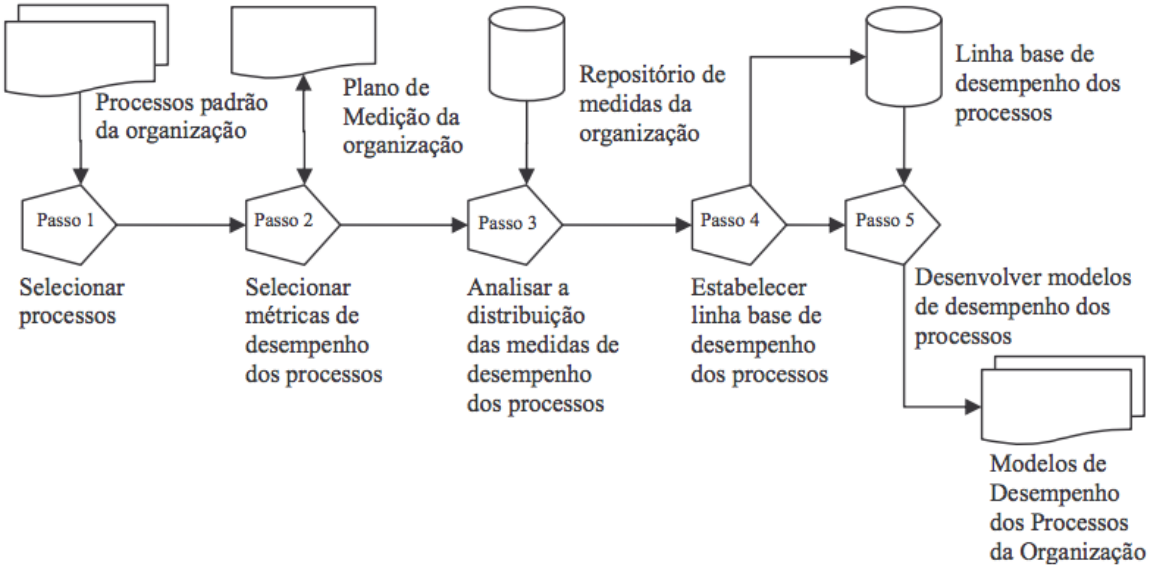
4 PROCEDIMENTOS PARA IDENTIFICAÇÃO DOS FATORES QUE INFLUENCIAM A PRODUTIVIDADE

Os procedimentos realizados com o intuito de identificar os fatores que afetam a produtividade foram baseados no arcabouço proposto por Florac & Carleton (1999) e Montoni et al. (2007).

Florac & Carleton (1999) propuseram um arcabouço para mensurar o comportamento dos processos e caracterizá-los quanto à estabilidade e capacidade. Os autores sugerem que a organização inicie por uma reflexão sobre os seus objetivos de negócio e do entendimento de quais processos podem dar maior contribuição. Em seguida, deve-se identificar e priorizar as questões a serem tratadas para que os processos atinjam os objetivos de negócio. Medidas devem ser identificadas, definidas, coletadas e analisadas para auxiliar na caracterização dos processos ou produtos. Ao avaliar o comportamento do processo com relação à estabilidade, as causas especiais de variação são identificadas e removidas. Ao atingir a estabilidade, o processo deve ser continuamente melhorado em prol do aumento da capacidade.

Montoni et al. (2007) propuseram uma metodologia para criação de modelos de desempenho constituída por um processo para desenvolver modelos de desempenho para gerência quantitativa de projetos de software (Figura 4.1). Os autores ilustraram a aplicação da metodologia no contexto de uma empresa de software CMMI nível 3, resultando a construção de um modelo para monitorar quantitativamente os projetos de desenvolvimento de software a partir da predição da qualidade do produto sob o ponto de vista do seu conteúdo.

Figura 4.1: Processo de desenvolvimento de modelos de desempenho de processos para gerência quantitativa de projetos de software (Montoni et al., 2007)



Fonte: Montoni et al. (2007)

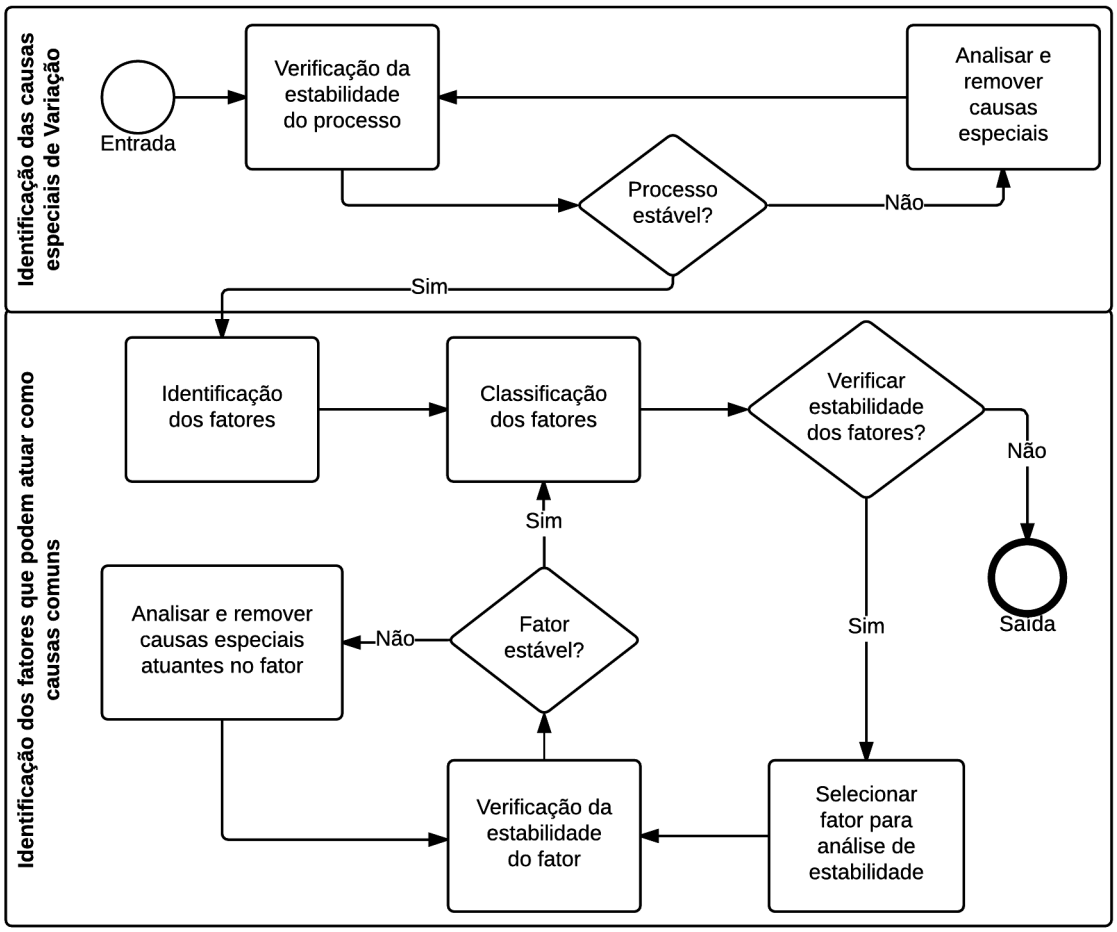
4.1 Procedimentos desenvolvidos

Os procedimentos necessários para a identificação dos fatores que afetam a produtividade estão exibidos no fluxograma 4.2. Os procedimentos estão divididos em duas etapas:

- Identificação das causas especiais de variação: etapa que é verificada a estabilidade do processo com relação a produtividade dos projetos. No caso da constatação de instabilidade, deve-se analisar a causa e realizar ações corretivas.
- Identificação dos fatores que podem atuar como causas comuns: etapa que identifica quais são os fatores influentes na produtividade do projeto. Nesta etapa, também é possível verificar a estabilidade dos fatores e realizar ações corretivas, quando necessário, para aumentar a precisão da produtividade.

Inicialmente deve-se coletar a medida de produtividade de cada um dos projetos existentes na base histórica. Vale ressaltar que os dados devem ser ordenados cronologicamente, pois a ordem influenciará nos procedimentos que serão realizados.

Figura 4.2: Representação dos procedimentos para identificação dos fatores que afetam a produtividade



Fonte: Elaborado pelo autor

4.1.1 Verificação da estabilidade do processo

Neste momento são realizados os testes de estabilidade. Pode-se utilizar até 8 (oito) testes (apresentados no capítulo capítulo 2). Cada teste é utilizado para verificar um cenário de violação diferente, eles abrangem violações desde pontos ultrapassando os limites aceitáveis, até a existência de n pontos consecutivos de um mesmo lado da linha central.

Um *Gráfico de Controle* é utilizado para verificar o resultado da execução dos testes. É indicado o uso do gráfico *XmR* ou *XMmR* conforme a necessidade de precisão dos testes de estabilidade. O gráfico do tipo *XMmR* é o mais preciso. Um outro critério que pode ser considerado para a utilização de um gráfico do tipo *XmR* é o tamanho da amostra. Pode não ser possível alcançar a estabilidade de um processo, dependendo do tamanho da amostra, com um gráfico *XMmR*, visto que este tipo de gráfico é mais

sensível as causas especiais.

É possível constatar a estabilidade da produtividade verificando se algum dos testes executados foi violado. Se houver violação, sabe-se que existem *causas especiais* atuando na produtividade, com isso, deve-se analisar e identificar quais são essas *causas especiais* (Moreira & Souza, 2008). Quando os testes são executados e é constatado que nenhum teste foi violado, pode-se afirmar que o processo está estável e só existe atuação de *causas comuns* no processo, ou seja, só exitem variações aceitáveis e que ocorrem dentro de limites previsíveis, caracterizando assim, a repetitividade de seu comportamento (Barcellos, 2009).

O intuito de alcançar a estabilidade do processo é garantir que não existam *outliers*, ou seja, projetos que pudessem ter sofrido a incidência de causas especiais de variação e que, por esta razão, poderiam dificultar a análise.

4.1.2 Processo estável?

Quando é verificada a estabilidade do processo, é possível iniciar o procedimento para identificação dos fatores que afetam a produtividade. Caso o processo não esteja estável, mesmo que alguma causa especial tenha sido removida, deve-se realizar uma análise para identificar o que está ocasionando o desvio que provoca a instabilidade, para que seja realizada a remoção.

4.1.3 Analisar e remover causas especiais

A partir do momento que é constatada a violação de algum dos testes que foram executados, é possível afirmar que existem causas especiais atuando no processo (Barcellos, 2009). Identificar causas especiais, em geral, requer esforço significativo e uma análise minuciosa dos dados, entretanto, segundo Barcellos (2009), é importante que as causas especiais sejam identificadas e removidas para alcançar a estabilidade do processo e assim aumentar a capacidade preditiva.

Para identificar as causas especiais, é necessário um profundo conhecimento em CEP e em processos (Schots & Rocha, 2012). Segundo Schots & Rocha (2012), a identificação das causas especiais é realizada através dos dados organizacionais e requer, normalmente, um conhecimento implícito da organização, ou seja, um conhecimento que não está documentado e encontra-se na experiência das pessoas que executam os processos. O autor complementa que após a identificação das causas, procura-se buscar uma ação que possa tanto corrigir o desvio identificado, como possa prevenir a sua recorrência. Uma análise das ações anteriormente adotadas em situações semelhantes na organização poderia auxiliar nesta identificação.

4.1.4 Identificação dos fatores

A partir do momento em que é verificada a estabilidade do processo, é iniciado o procedimento para identificação dos fatores candidatos à causa comum de variação na produtividade dos projetos. Os fatores são identificados a partir de uma análise empírica nos projetos da base histórica da organização.

4.1.5 Classificação dos fatores

Os fatores são classificados de acordo com o nível de influência que exercem na produtividade dos projetos. O objetivo de realizar a classificação é identificar quais são os fatores mais influentes.

O nível de influencia dos fatores são baseados no coeficiente de determinação (R^2), quanto maior o coeficiente, maior será a influencia do fator na produtividade.

No caso da identificação de fatores que não são mensuráveis, como por exemplo plataforma do software e tipo de cliente, deve-se considerar uma constante numérica para cada valor diferente encontrado. O coeficiente de determinação é calculado conforme a equação:

$$Coeficiente\ de\ Determinação\ (R^2) = \frac{\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \tag{4.1}$$

onde:

n = número de observações.

$\overline{y}$ = média de todas as obervações.

y_i = valor da observação individual i.

$\hat{y}$ = valor previsto da observação i.

4.1.6 Verificar estabilidade dos fatores?

A partir do momento que os fatores estão identificados e classificados de acordo com o nível de influencia que exercem na produtividade, é possível verificar a estabilidade dos fatores. O intuito de verificar a estabilidade dos fatores é identificar quais são os fatores que estão instáveis para realizar remoções das causas especiais atuantes. A proporção que mais fatores identificados estiverem estáveis, a tendência é que as estimativas de projetos, derivados da produtividade, fiquem mais precisas.

O procedimento de estabilizar os fatores influentes na produtividade podem ser repetidos quantas vezes forem necessário, entretanto, é importante observar que para realizar uma análise satisfatória, um conjunto mínimo de projetos é necessário. Segundo autores, o número desejado de observações para cada medida analisada pode

variar de 15 (Wheeler, 1993) a 45 (Florac & Carleton, 1999).

4.1.7 Selecionar fator para análise de estabilidade

Durante as análises de estabilidade dos fatores, o esforço deve ser empregado a fim de analisar os principais fatores de influencia sobre a produtividade, pois analisar todos os fatores pode não ser economicamente viável. Alcançar a estabilidade dos fatores mais influentes, ao invés dos menos influentes, faz com que a produtividade seja calculada de forma mais precisa. Assim, conhecer o contexto organizacional torna-se uma habilidade útil.

As análises para identificação e remoção das causas especiais atuantes nos fatores, para alcançar a estabilidade, deve ocorrer com um fator por vez.

4.1.8 Verificação da estabilidade do fator

Para verificar a estabilidade de um fator, utiliza-se o mesmo procedimento utilizado para verificar a estabilidade da produtividade dos projetos, ou seja, executa os testes de estabilidade, com o auxílio do gráfico de controle, e verifica se houve alguma violação. No caso de existir violações, sabe-se que o fator em questão não está estável.

A escolha do gráfico (gráfico do tipo *XmR* ou *XMmR*) deve seguir o mesmo raciocínio utilizado no momento da seleção do gráfico para analisar a produtividade geral.

4.1.9 Fator estável?

Quando é constatado que o fator que está sendo analisando, quanto a sua estabilidade, está estável, é iniciado o procedimento para classificação dos fatores novamente. No caso da constatação da instabilidade do fator, inicia-se o procedimento para análise e remoção das causas especiais atuantes no fator. É necessário analisar e remover as causas especiais atuantes no fator até constatar que o fator está estável.

Caso seja constatada a estabilidade do fator desde início, ou seja, desde a *seleção do fator*, devido aos dados estarem preservados, considera-se a última classificação dos fatores realizada.

4.1.10 Analisar e remover causas especiais atuantes no fator

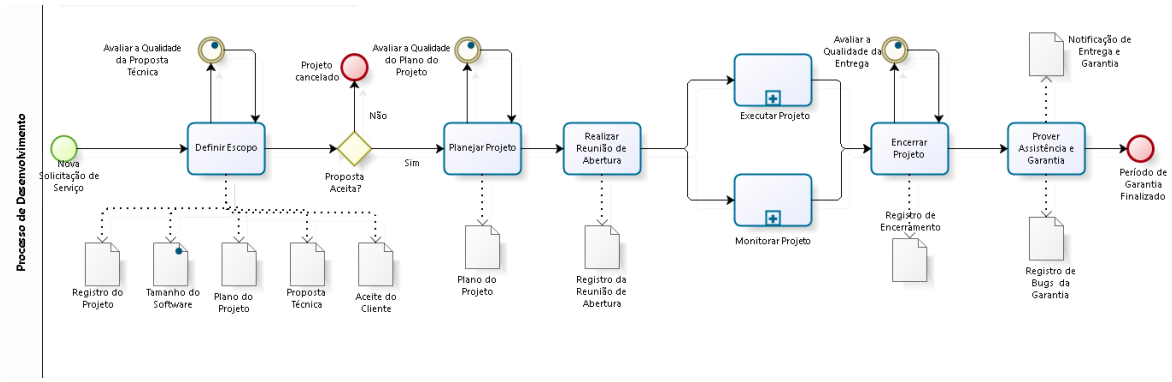
Com a identificação de instabilidade de um fator, deve-se identificar, realizando uma análise empírica, e remover a causas especiais atuantes no fator, assim como é feito em *analisar e remover causas especiais* 4.1.3.

5 ESTUDO DE CASO

5.1 Contexto

Os dados que foram utilizados para as análises, foram extraídos de uma base histórica de dados existente na empresa Inform Sistemas Ltda. A Inform Sistemas Ltda. é uma empresa Alagoana que foi fundada em 1991, e nos seus últimos anos têm investido um esforço sistemático em melhoria dos seus processos utilizando o modelo MR-MPS como referência desde 2009, ano em que obteve o MPS nível F. Com a melhoria de seus processos e aplicando práticas de alta maturidade, a empresa alcançou, em 2013, o nível C do modelo MPS do MR-MPS-SW. Dentre suas frentes de negócio, a empresa possui uma unidade de fábrica de software que atua no desenvolvimento de aplicações nas plataformas Android, iOS e Web. Na imagem 5.1 é exibida a visão macro do Processo fabril.

Figura 5.1: Processo Fabril



Fonte: Elaborado pela empresa do Estudo de Caso

A fábrica de software da Inform Sistemas Ltda. trabalha oferecendo três serviços para os seus clientes.

- Desenvolvimento de aplicações
- Especificação de aplicações

- Especificação e desenvolvimento de aplicações

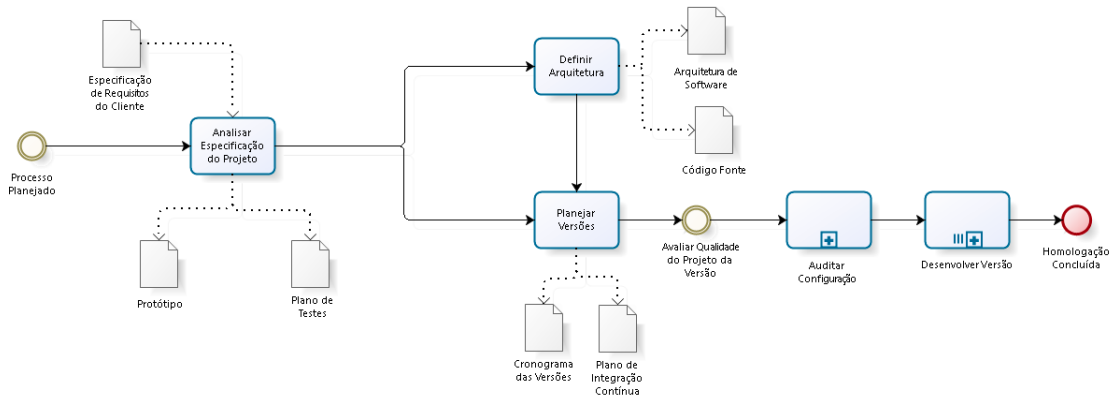
Vale ressaltar que todos os processos utilizados nos três tipos de serviços foram avaliados e aprovados no nível C do modelo MR-MPS-SW. Estes processos eram evoluídos com base em melhorias que eram identificadas pela própria equipe da fábrica de software. Na ocasião do estudo os processos estavam na versão 1.18.

5.1.1 Desenvolvimento de aplicações

Partindo do pressuposto que existe uma especificação de software, este serviço, em síntese, atua da seguinte forma (Figura 5.2 ilustra o processo utilizado para este tipo de serviço):

1. Realiza análise da especificação previamente elaborada;
2. Realiza definições da arquitetura do sistema;
3. Realiza o planejamento de suas versões;
4. Desenvolve cada versão, identificada no passo anterior, realizando homologações internas e externas

Figura 5.2: Processo Desenvolvimento de aplicações

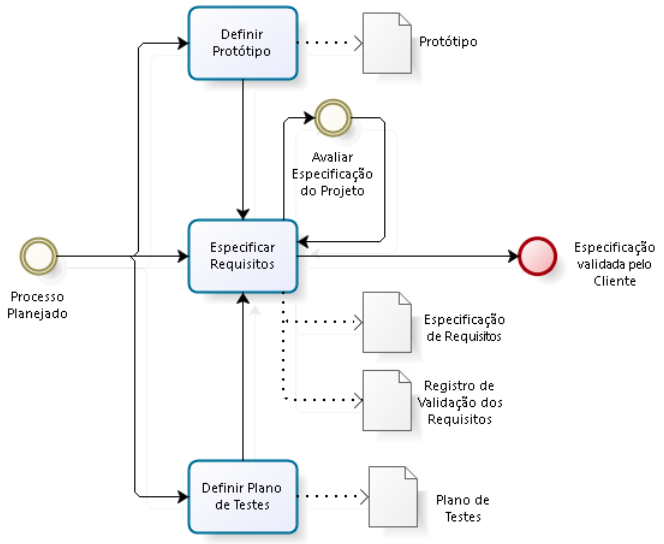


Fonte: Elaborado pela empresa do Estudo de Caso

5.1.2 Especificação de aplicações

Neste serviço são realizadas ações para a elaboração de uma documentação onde contemple todos os requisitos funcionais e não funcionais desejáveis pelo cliente. Todos os requisitos elaborados são revisados em conjunto com o cliente para garantir que a documentação reflita a real necessidade do cliente. Na imagem 5.3 é ilustrado o processo utilizado na elaboração da documentação de um projeto.

Figura 5.3: Processo Especificação de aplicações



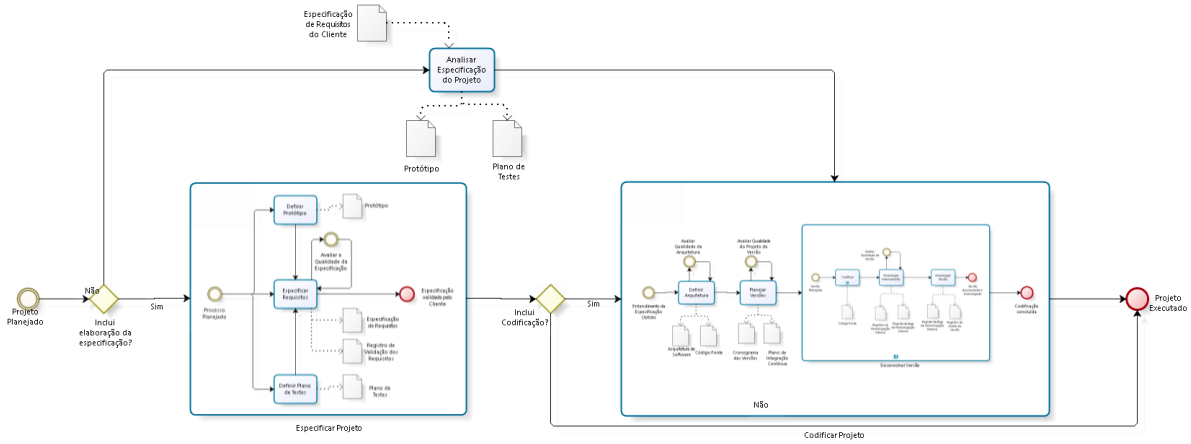
Fonte: Elaborado pela empresa do Estudo de Caso

5.1.3 Especificação e desenvolvimento de aplicações

Este serviço visa atender as duas necessidades do cliente, (i) elaborar uma documentação de uma aplicação de software e (ii) realizar o desenvolvimento desta aplicação. Neste serviço é realizada a união das atividades existentes nos processos dos dois serviços anteriores (serviço de especificação de aplicações e serviço de desenvolvimento de aplicações), realizando adequações afim de evitar execução de atividades duplicadas e acrescentando atividades necessárias para melhor execução do processo.

Neste tipo de serviço, o cliente é envolvido desde os primeiros passos da elaboração da especificação até o momento em que as versões são elaboradas e entregues para homologação. Na imagem 5.4 é ilustrado o processo utilizado para este tipo de serviço.

Figura 5.4: Processo de Especificação e Desenvolvimento de aplicações



Fonte: Elaborado pela empresa do Estudo de Caso

5.2 Análises

As análises encontradas neste trabalho foram realizadas com um dos principais indicadores existente na Fábrica de Software da Inform Sistemas, a produtividade. A produtividade é mensurada, pela Inform Sistemas, como sendo razão entre o esforço realizado e o tamanho do projeto, esta razão também é utilizada como medida de produtividade pelo ISBSG[1], entretanto, para referenciar a produtividade, o ISBSG utiliza o termo de taxa de entrega de projetos.

O estudo contou com dezesseis projetos[2] (Tabela 5.1) da base histórica da unidade fabril. Dentre os projetos, são encontrados projetos de diversas linguagens (java, Objective-c, entre outras) e nas plataformas Android, iOs e web. Em busca de identificar os fatores que poderiam influenciar a produtividade, os projetos foram caracterizados de acordo com a visão dos líderes de projeto e com os fatores relacionados por Wagner & Ruhe (2008) que foram considerados pertinentes ao contexto organizacional da unidade fabril.

Os projetos utilizados no estudo foram ordenados de acordo com a data de término dos projetos, esta ordenação foi utilizada para que fosse seguida a ordem cronológica das produtividades apresentadas. Os projetos estão identificados como InformXXX (onde XXX representa um numeral) para preservar a confidencialidade dos dados.

Seguindo os procedimentos iniciais descritos no capítulo 4, inicialmente foram utilizados dezesseis projetos existentes na base histórica da unidade fabril.

[1] International Software Benchmarking Standards Group. http://www.isbsg.org

[2] Os projetos atuam nas seguintes áreas: suporte a feiras de idiomas e gestão empresarial.

Tabela 5.1: Projetos da base histórica da unidade fabril

Ordem	Projeto	Produtividade	Ordem	Projeto	Produtividade
1	Inform003	4,0120000000	9	Inform012	2,0069035533
2	Inform004	3,2700000000	10	Inform009	3,5285637823
3	Inform005	3,5321428571	11	Inform014	7,5087719298
4	Inform006	2,1692307692	12	Inform016	7,7339449541
5	Inform010	1,8486842105	13	Inform015	4,0114136126
6	Inform007	3,8501408451	14	Inform020	4,8378947368
7	Inform013	3,2253839205	15	Inform018	3,7729242622
8	Inform011	3,8763513514	16	Inform019	3,9651929825

Fonte: Dados da pesquisa

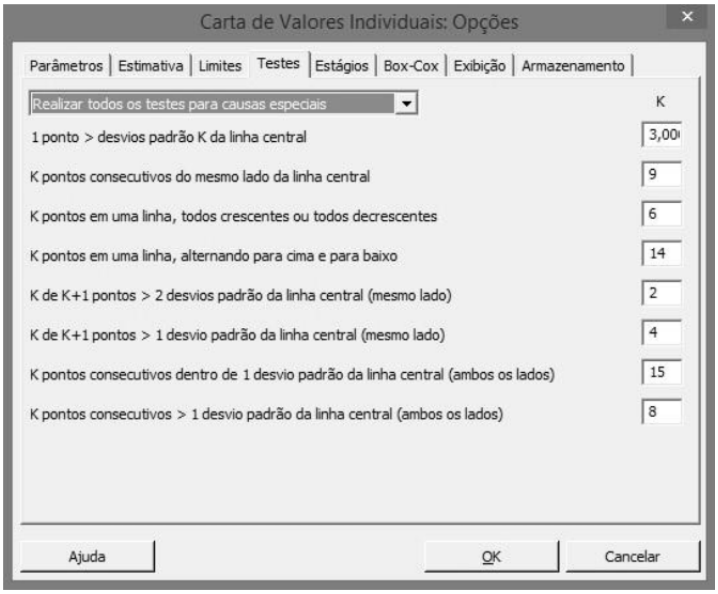
5.2.1 Identificando as causas especiais de variação

Durante a aplicação dos testes de estabilidade, foi identificado que existiam 2 (duas) causas especiais que atuavam na produtividade dos projetos. As causas especiais foram removidas para que fosse alcançada a estabilidade do processo com relação a produtividade dos projetos. Os detalhes da identificação e remoção das causas especiais, estão descritos abaixo.

Os testes realizados nas próximas seções foram executados através da ferramenta Minitab[3], a escolha da ferramenta Minitab se deu por conta de ser uma ferramenta prática, para sua utilização, e atender as necessidades desejadas, visto que a mesma também possui o objetivo em realizar análises de variáveis. A execução dos testes, para verificar a estabilidade do processo, é realizada através da elaboração do gráfico de controle para variáveis individuais (gráfico do tipo XmR, apresentado em 2.2.1).Na configuração da elaboração do gráfico, são utilizados todos os parâmetros padrões da ferramenta, inclusive os valores sugeridos na configuração dos testes (Figura 5.5).

[3]Minitab® Statistical Software, www.minitab.com

Figura 5.5: Testes de estabilidade oferecidos na ferramenta Minitab com os respectivos parâmetros



Fonte: Elaborada pelo autor

Todos os gráficos de controle que foram utilizados nas análises, descritas nas próximas seções, foram do tipo *XmR* (apresentado em 2.2.1).

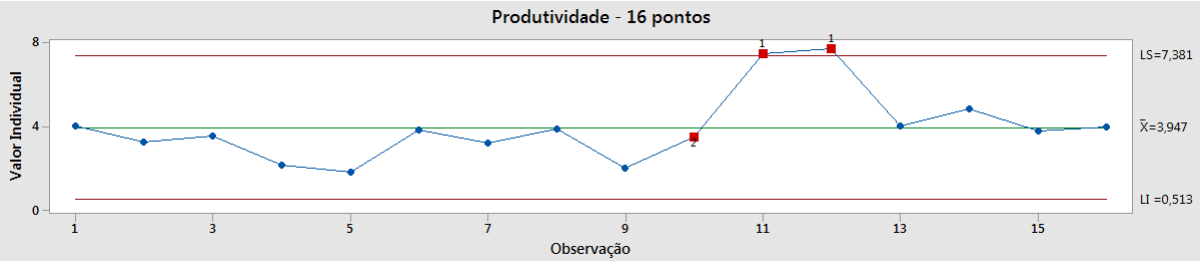
O motivo da utilização do tipo de gráfico foi devido a necessidade de monitorar apenas uma variável (produtividade) e pela precisão do gráfico ser suficiente para atender as necessidades das análises.

Com o resultado da execução dos testes de estabilidade utilizando os 16 pontos, foram identificadas duas violações:

- Teste 1 - Identificação de 2 projetos (projetos *Inform014* e *Inform016*) que possuem a produtividade acima de 3 desvios padrão da linha central.
- Teste 2 - Identificação de 9 projetos consecutivos do mesmo lado da linha central;

Os limites inferior e superior para o conjunto de projetos utilizados foram 0,513 HH/PF e 7,381 HH/PF, respectivamente (vide Figura 5.6).

Figura 5.6: Gráfico de controle para valores individuais



Fonte: Dados da pesquisa

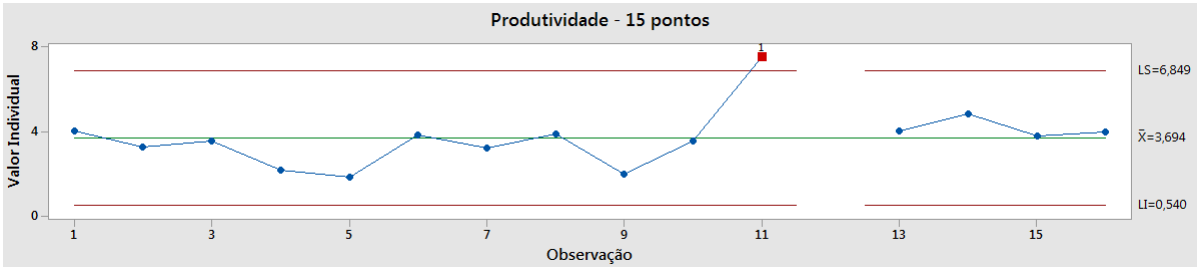
Analisando o projeto *Inform016* para identificar a causa da variação da produtividade, foi observado um investimento de esforço adicional para a implementação de testes de aceitação automáticos na plataforma Android. Isto implicou no aumento da razão de esforço para cada ponto por função do projeto. O projeto *Inform016* foi excluído da amostra de projetos e uma nova rodada de testes foi executada.

Após remover o projeto, foi feita uma segunda verificação da estabilidade do processo com relação a produtividade, utilizando os 15 pontos restantes. Foi identificada uma violação:

- Teste 1 - Identificação de um projeto (projeto *Inform014*) ter ficado acima de três desvios padrão a partir da linha central.

Após a exclusão do projeto *Inform016*, os limites inferior e superior foram atualizados para 0,540 HH/PF e 6,849 HH/PF, respectivamente, e o valor da linha central passou de 3,947 HH/PF para 3,694 HH/PF (vide Figura 5.7).

Figura 5.7: Gráfico de controle com a exclusão do projeto *Inform016*

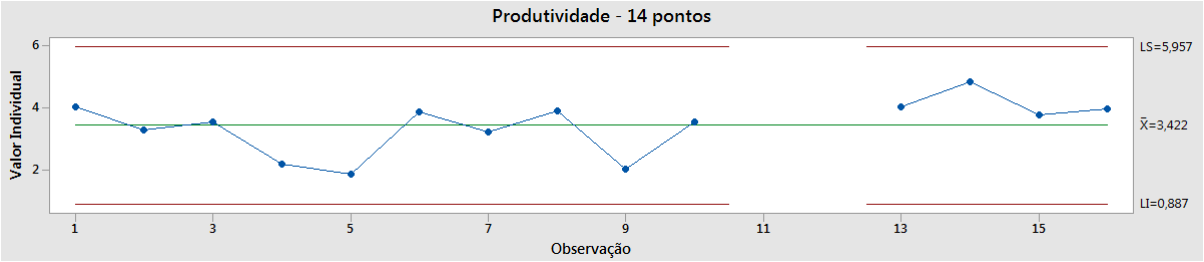


Fonte: Dados da pesquisa

Ao analisar o projeto *Inform014*, constatou-se a existência de um requisito não-funcional que foi responsável por aproximadamente 50% do esforço adicional investido no projeto. O projeto *Inform014* foi excluído da amostra de projetos e uma nova rodada de testes foi executada.

Utilizando os 14 pontos (com os projetos *Inform014* e *Inform016* excluídos), para verificar a estabilidade do processo, não foi identificada nenhuma violação dos testes de estabilidade com relação à produtividade. Após recalculados, os limites inferior e superior foram atualizados para 0,887 HH/PF e 5,957 HH/PF, respectivamente, e o valor da linha central passou para 3,422 HH/PF (vide Figura 5.8). É importante considerar que a aplicação deste procedimento depende da existência de um número significativo de projetos.

Figura 5.8: Gráfico de controle para valores individuais com a exclusão dos projetos *Inform014* e *Inform016*



Fonte: Dados da pesquisa

5.2.2 Identificando fatores que podem atuar como causas comuns

Após alcançar a estabilidade do processo com relação a produtividade, foi iniciada a busca para identificar os fatores que podem atuar como causas comuns. Nesta etapa, a análise foi realizada inicialmente com 14 pontos derivados do passo anterior.

A busca pelos potenciais fatores foi baseada nos fatores identificados por Wagner & Ruhe (2008) e em outros fatores que foram identificados, a partir de estudos empíricos na base de dados dos projetos, pelos líderes de projeto da unidade fabril. Como resultado foram identificados os seguintes fatores:

- **Plataforma:** foram considerados 3 (três) tipos de plataforma: iOS, Android e Web. A plataforma é referente ao ambiente em que o software é executado.
- **Linguagem de programação:** foram consideradas duas linguagem de programação: Java e Objective-C. A linguagem de programação é referência a linguagem que foi utilizada para desenvolver os projetos.
- **Tipo Cliente:** foram considerados dois tipos de clientes: interno e externo. Os clientes internos foram demandas existentes dentro da própria empresa, já as demandas externas, foram clientes de outras empresas.

- **Tipo Serviço:** neste fator foram considerados três tipos de serviço contratados nos projetos: especificação de aplicações, desenvolvimento de aplicações e especificação e desenvolvimento de aplicações. Detalhes destes tipos de serviços foram apresentados no início deste capítulo.

- **Testes Automáticos:** este fator representa a existência ou não de testes automáticos para os projetos que foram utilizados no estudo.

- **Projeto Novo/Evolução:** este fator referencia se o projeto foi desenvolvido pela primeira vez, ou se é uma evolução de algum dos projetos já existentes na base histórica.

- **Prazo Projeto:** considera o prazo real que foi utilizado para o desenvolvimento dos projetos.

- **Local Físico (ST/1A):** durante a execução dos projetos, a equipe foi realocada fisicamente para outro local. Este fator considera o local onde os projetos foram desenvolvidos.

- **Período de Início:** considera o período de início do projeto de acordo com os meses do ano.

- **Produtividade na Codificação:** este fator representa a produtividade da codificação (atividade existente no processo fabril) de cada projeto. Na atividade de codificação é realizado todo o trabalho de implementação da aplicação.

- **Produtividade na Análise da especificação:** este fator representa a produtividade na atividade de análise da especificação (atividade existente no processo fabril) de cada projeto. Na atividade de realizar análise da especificação, é realizado todo o trabalho de levantamento e definição dos requisitos que fazem parte do projeto e identificação dos atores que interagem com a aplicação.

- **Produtividade na Arquitetura:** este fator representa a produtividade na atividade de definição da arquitetura (atividade existente no processo fabril) de cada projeto. Na atividade de arquitetura, é realizada toda a documentação necessária da aplicação (a nível de arquitetura) e a implementação básica de toda a estrutura da aplicação.

- **Densidade *Bugs* Homologação Externa:** este fator representa a densidade de *bugs* na atividade de homologação externa (atividade existente no processo fabril) de cada projeto. A atividade de homologação externa representa as ações que são realizadas pelos clientes no momento de homologação do projeto e as

ações dos desenvolvedores para corrigir os itens que são identificados pelo cliente classificados como *bug*.

- **Densidade de *Bugs*:** este fator representa a densidade de *bugs* somados das atividades de homologação interna (homologação realizada pela própria equipe antes de liberar o projeto para o cliente) e homologação externa (atividades existentes no processo fabril) de cada projeto.

- **Mudanças:** durante a execução do projeto, o cliente pode a qualquer momento solicitar as mudanças desejadas. Esta fator, representa o número de mudanças que foram solicitadas pelo cliente em cada projeto.

- **Componentes:** este fator representa a quantidade de componentes que foram desenvolvidos em cada projeto. Foram classificados como 1 (um) para os projetos que desenvolveram somente o *back-end* ou *front-end* e 2 (dois) para os projetos que desenvolveram os 2 (dois) componentes *back-end* e *front-end*.

- **Linhas de Código (LOC):** considera a quantidade de linhas de código de cada projeto.

- **Número de classes filhas:** considera a quantidade de classes filhas utilizadas em cada projeto.

- **Linhas de código duplicadas:** considera a quantidade de linhas de código duplicadas de cada projeto.

- **Métodos por classe:** considera a média de números de métodos por classe de cada projeto.

- **Número de telas:** considera o número de telas que foram desenvolvidas ou alteradas em cada projeto.

- **Número de Requisitos de Negócio:** considera o número de requisitos de negócio que foram identificados em cada projeto.

Após a identificação, os fatores foram classificados (Tabela 5.2) considerando o coeficiente de determinação (R^2) para verificar qual o nível de influência dos fatores na produtividade dos projetos. Os coeficientes de determinação foram calculado utilizando a equação 2.2.

Para um melhor entendimento da influencia de cada fator com relação a produtividade, o coeficiente de correlação também foi calculado, utilizando a equação 2.1, para cada fator.

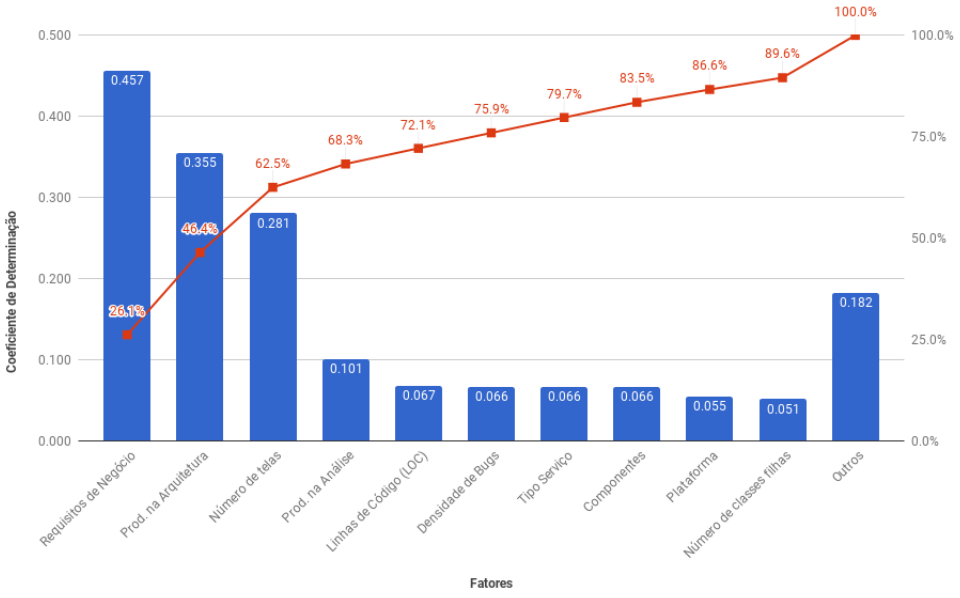
Tabela 5.2: Correlação e coeficiente de determinação entre a produtividade e os Fatores Candidatos

	Fatores Candidatos	Correlação (*r*)	Coeficiente de Determinação (R^2)
1	Plataforma	-0,234	0,055
2	Linguagem de programação	-0,207	0,043
3	Tipo Cliente	-0,166	0,028
4	Tipo Serviço	0,257	0,066
5	Testes Automáticos	-0,070	0,005
6	Projeto Novo/Evolução	0,140	0,020
7	Prazo Projeto	0,170	0,029
8	Local Físico (ST/1A)	-0,115	0,013
9	Período de Início	0,087	0,008
10	Produtividade na Codificação	-0,106	0,011
11	Produtividade na Análise da especificação	0,318	0,101
12	Produtividade na Arquitetura	0,595	0,355
13	Densidade *Bugs* Homologação Externa	-0,033	0,001
14	Densidade de *Bugs*	-0,258	0,066
15	Mudanças	-0,031	0,001
16	Componentes	0,257	0,066
17	Linhas de Código (LOC)	-0,260	0,067
18	Número de classes filhas	-0,226	0,051
19	Linhas de código duplicadas	-0,103	0,011
20	Métodos por classe	0,118	0,014
21	Número de telas	-0,530	0,281
22	Número de Requisitos de Negócio	-0,676	0,457

Fonte: Dados da pesquisa

Para observar mais claramente quais os fatores mais influentes na produtividade dos projetos, foi elaborado um gráfico de *Pareto* (Figura 5.9) a partir dos valores obtidos na tabela 5.2.

Figura 5.9: Gráfico de Pareto para seleção dos fatores mais influentes



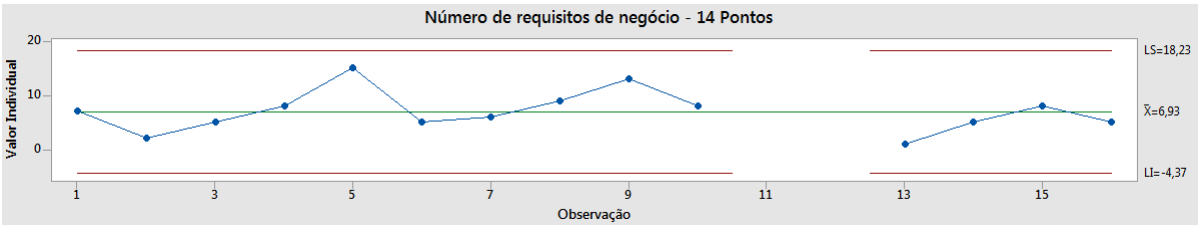
Fonte: Dados da pesquisa

Como observa-se no gráfico de *Pareto* (Figura 5.9), os fatores mais influentes na produtividade dos projetos são o número de requisitos de negócio, a produtividade na arquitetura e o número de telas que são manipuladas em cada projeto, cada fator explica aproximadamente, de acordo com o coeficiente de determinação, 46%, 36% e 28%, respectivamente, da variação da produtividade.

Com os fatores identificados e classificados, com relação a influencia na produtividade, foi iniciado o procedimento de analisar a estabilidade dos fatores. Conforme mencionado em *selecionar fator para análise de estabilidade* 4.1.7, a análise de estabilidade foi iniciada a partir dos fatores mais influentes.

Ao analisar a estabilidade do *número de requisitos de negócio* dos projetos, fator mais influente na produtividade com aproximadamente 46% de influência, foi constatado que o fator estava estável. A estabilidade do *número de requisitos de negócio* pode ser visualizada através do gráfico exibido na Figura 5.10.

Figura 5.10: Gráfico de controle para valores individuais do número de requisitos de negócio para os 14 projetos



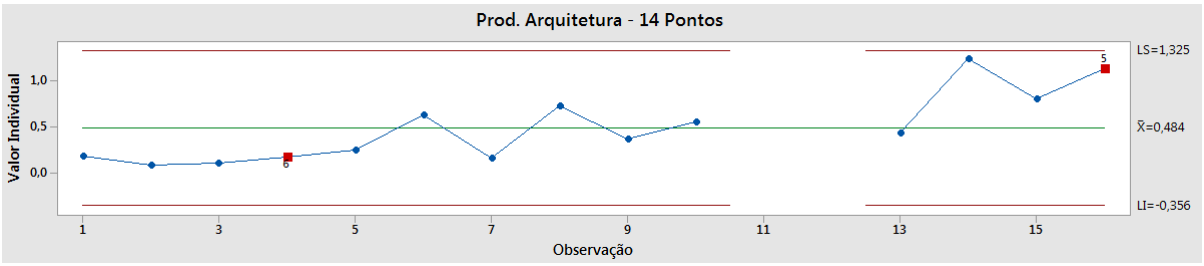
Fonte: Dados da pesquisa

Constatada a estabilidade do fator mais influente, foi realizada a análise de estabilidade do segundo fator mais influente, a produtividade da arquitetura. Após análise de estabilidade da produtividade da arquitetura, foi constatado que este fator não estava estável, para os 14 projetos. Assim, os mesmos procedimentos, aplicados para estabilizar a produtividade geral dos projetos, foram aplicados para a produtividade da arquitetura. Com a execução dos testes de estabilidade aplicados para a produtividade da arquitetura, foram identificadas duas violações:

- Teste 6 - Identificação de 4 projetos (projetos *Inform003*, *Inform004*, *Inform005* e *Inform006*) a mais do que 1 desvio padrão da linha central e do mesmo lado.
- Teste 5 - Identificação de 2 projetos (projetos *Inform020* e *Inform019*) que possuem a produtividade acima de 2 desvios padrão da linha central e do mesmo lado.

Os limites inferior e superior para o conjunto de projetos utilizado foram -0,356 HH/PF e 1,325 HH/PF, respectivamente (vide Figura 5.11). O valor da linha central ficou em 0,484 HH/PF.

Figura 5.11: Gráfico de controle para valores individuais da produtividade na arquitetura para os 14 projetos



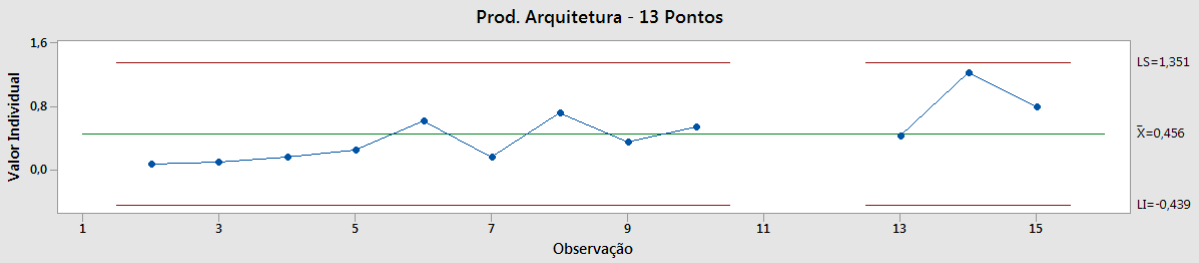
Fonte: Dados da pesquisa

Ao analisar o projeto *Inform019*, observou-se que no projeto foi economizado esforço devido a existência de outro projeto, executado anteriormente, que possuía os mesmos

requisitos, assim, não foi necessário adquirir novos conhecimentos para a elaboração da arquitetura e parte da documentação da arquitetura foi reutilizada. Com a identificação desta causa especial, o projeto *Inform019* foi excluído e os testes de estabilidade foram executados com os 13 projetos restantes.

Com a exclusão do projeto *Inform019*, um novo gráfico de controle foi elaborado (vide Figura 5.12) e novos limites foram calculados. Os limites inferior e superior foram -0,439 HH/PF e 1,351 HH/PF, respectivamente. O valor da linha central ficou em 0,456 HH/PF.

Figura 5.12: Gráfico de controle para valores individuais da produtividade na arquitetura para os 13 projetos

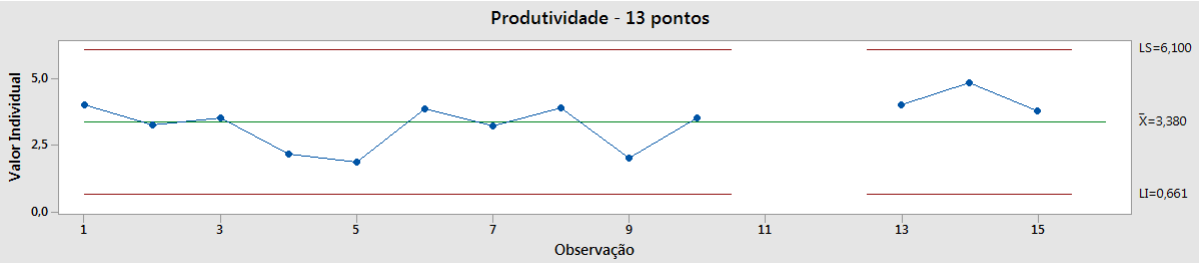


Fonte: Dados da pesquisa

Após a exclusão do projeto *Inform019*, foi constatada não só a estabilidade da produtividade da arquitetura dos projetos, mas também a estabilidade nos demais fatores mais influentes na produtividade geral dos projetos.

Após a remoção das causas especiais identificadas a partir dos fatores mais influentes, observa-se que a produtividade geral do projeto continua estável, conforme o gráfico de controle apresentado na Figura 5.13 e que os limites inferior, superior e linha central foram recalculados para 0,661 HH/PF, 6,100 HH/PF e 3,380 HH/PF, respectivamente.

Figura 5.13: Gráfico de controle para valores individuais com a exclusão do projeto *Inform019*



Fonte: Dados da pesquisa

Alcançada a estabilidade da produtividade geral dos projetos e de seus fatores mais influentes, todos os fatores foram novamente analisados e classificados quanto ao co-

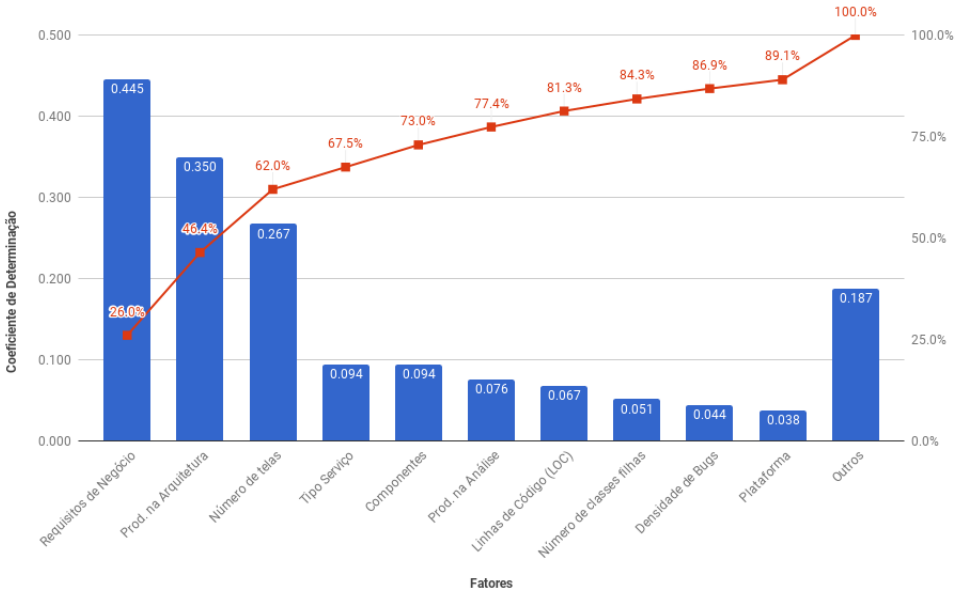
eficiente de determinação, de cada respectivo fator, com relação a produtividade geral dos projetos (Tabela 5.3). A partir da tabela 5.3, observa-se que os fatores número de requisitos de negócio, a produtividade na arquitetura e o número de telas que são manipuladas em cada projeto, passaram a explicar aproximadamente, de acordo com o coeficiente de determinação, 45%, 35% e 27%, respectivamente, da variação da produtividade.

Tabela 5.3: Correlação e coeficiente de determinação entre a produtividade e os Fatores Candidatos após alcançada estabilidade dos fatores mais influentes

	Fatores Candidatos	Após estabilidade dos fatores mais influentes		Antes da estabilidade dos fatores mais influentes	
		Correlação (r)	**Coeficiente de Determinação (R^2)**	**Correlação (r)**	**Coeficiente de Determinação (R^2)**
1	Plataforma	-0,194	0,038	-0,234	0,055
2	Linguagem de programação	-0,176	0,031	-0,207	0,043
3	Tipo Cliente	-0,241	0,058	-0,166	0,028
4	Tipo Serviço	0,306	0,094	0,257	0,066
5	Testes Automáticos	-0,039	0,002	-0,070	0,005
6	Projeto Novo/Evolução	0,079	0,006	0,140	0,020
7	Prazo Projeto	0,195	0,038	0,170	0,029
8	Local Físico (ST/1A)	-0,145	0,021	-0,115	0,013
9	Período de Início	0,064	0,004	0,087	0,008
10	Produtividade na Codificação	-0,051	0,003	-0,106	0,011
11	Produtividade na Análise da especificação	0,275	0,076	0,318	0,101
12	Produtividade na Arquitetura	0,591	0,350	0,595	0,355
13	Densidade *Bugs* Homologação Externa	0,008	0,000	-0,033	0,001
14	Densidade de *Bugs*	-0,209	0,044	-0,258	0,066
15	Mudanças	0,006	0,000	-0,031	0,001
16	Componentes	0,306	0,094	0,257	0,066
17	Linhas de Código (LOC)	-0,260	0,067	-0,260	0,067
18	Número de classes filhas	-0,226	0,051	-0,226	0,051
19	Linhas de código duplicadas	-0,103	0,011	-0,103	0,011
20	Métodos por classe	0,118	0,014	0,118	0,014
21	Número de telas	-0,517	0,267	-0,530	0,281
22	Número de Requisitos de Negócio	-0,667	0,445	-0,676	0,457

Para observar mais claramente os fatores que mais influenciam na produtividade com os valores atuais, após remoção das causas especiais dos fatores mais influentes, novamente foi elaborado um gráfico de *Pareto* 5.14.

Figura 5.14: Gráfico de *Pareto* para seleção dos fatores mais influentes, após alcançada estabilidade dos fatores mais influentes



Fonte: Dados da pesquisa

Um dos aspectos que chamaram atenção foi o fato da produtividade da codificação exercer menos influência do que a produtividade na arquitetura, uma vez que a codificação responde a 41% e a atividade de arquitetura corresponde a aproximadamente 14% de esforço realizado em um projeto, assim, destacamos que uma má escolha nessa atividade poderá impactar em várias atividades seguintes do projeto. Outra questão interessante é que fatores que poderiam indicar a existência de subgrupos de projetos, a exemplo da plataforma, linguagem de programação, tipo de serviço, projeto novo/-manutenção, apresentaram baixa correlação.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

Este trabalho apresentou um mecanismo para identificação de fatores que afetam a produtividade de projetos de software atuais. O mecanismo de identificação dos fatores descrito neste trabalho é apresentado em duas etapas: identificação das causas especiais de variação e identificação dos fatores que podem atuar como causas comuns.

A *identificação das causas especiais de variação* foi guiada pela utilização de controle estatístico de processos, já a *identificação dos fatores que podem atuar como causas comuns* foi guiada utilizando regressão linear.

No estudo de caso, realizado em uma empresa Alagoana, primeiramente foram realizadas ações para alcançar a estabilidade dos projetos. Após alcançada a estabilidade dos projetos, diversos fatores que influenciam a produtividade dos projetos de software foram identificados, dentre estes, três foram classificados como os fatores mais influentes (seguindo a ordem): número de requisitos de negócio, produtividade na atividade de projeto arquitetural e número de telas.

Análises de estabilidade dos fatores mais influentes foram realizadas utilizando controle estatístico de processos e causas especiais foram identificadas. A estabilidade dos fatores mais influentes foi alcançada após remoção de causas especiais. Foi observado que a influência na produtividade geral dos projetos foi reduzida.

Os resultados preliminares corroboram com a opinião de Florac & Carleton (1999), quanto ao benefício de se ter um controle estatístico de processos, mesmo quando a organização está em níveis iniciais de maturidade.

6.2 Contribuições

As principais contribuições deste trabalho foram:

i. Os procedimentos para identificação e classificação dos fatores que influenciam a produtividade em projetos de *software*.

Além dos procedimentos propriamente ditos, cada um de seus passos caracteriza uma contribuição particular, uma vez que, além de serem utilizados e evoluídos no contexto dos procedimentos como um todo, também podem ser utilizados e evoluídos de forma independente. Assim, são também contribuições deste trabalho:

ii. O procedimento para identificação dos fatores candidatos;

iii. O procedimento para classificação e remoção das causas especiais atuantes nos fatores identificados;

Além das contribuições diretamente relacionadas, ao longo do desenvolvimento deste trabalho, outras contribuições relevantes foram produzidas:

iv. A lista de fatores que influenciam a produtividade de projetos de software. Vale lembrar, que a influencia de cada fator dependente do contexto em que os projetos são executados, porém, a lista identificada pode ser utilizada como base para análise em outros trabalhos.

v. A lista de causas especiais que foram identificadas nos projetos. As causas especiais de variação são importantes para que seja realizada uma reflexão e que ações possam ser realizadas afim de evitar ou minimizar recorrências futuras das mesmas.

6.3 Limitações Observadas

Também é importante destacar algumas limitações da abordagem desenvolvida neste trabalho.

No contexto da execução dos procedimentos, pode ser considerada uma limitação deste trabalho a necessidade de existir uma base histórica de projetos para que seja realizada uma análise dos fatores de influência no contexto de uma organização. Caso a organização não possua dados históricos, pode-se utilizar inicialmente dados identificados na literatura, entretanto, a utilização desta abordagem compromete a precisão dos resultados encontrados.

Quanto ao escopo dos resultados apresentados neste trabalho, eles são restritos ao contexto do caso de uso apresentado. Os procedimentos que foram descritos neste trabalho, são aplicáveis em qualquer organização, desde que exista uma base histórica, ou que seja utilizada uma alternativa como a apresentada no parágrafo anterior, entretanto, os resultados encontrados são satisfatórios para o contexto em que os procedimentos são aplicados.

6.4 Trabalhos Futuros

Como trabalhos futuros, um aspecto relevante a ser considerado neste estudo é a potencial existência de outros fatores que não foram considerados ou atendidos pelos projetos utilizados no experimento. Acredita-se que outros fatores, que não foram considerados na lista de fatores candidatos, podem atuar como causas comuns de variação na produtividade. Para se certificar disso, se faz necessário aprofundar ainda mais a compreensão sobre o comportamento dos processos em diferentes tipos de projetos, tais como: jogos eletrônicos, aplicações científicas, outros sistemas de informação de diferentes plataformas, desenvolvimento de software básico, tais como sistemas operacionais e compiladores, etc.

Referências bibliográficas

Barcellos, M. P. (2009), 'Uma estratégia para medição de software e avaliação de bases de medidas para controle estatístico de processos de software em organizações de alta maturidade', *Doctoral, Engenharia de Sistemas e Computação, COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro*.

Barreto, A. (2011), 'Uma abordagem para definição de processos baseada em reutilização visando à alta maturidade em processos', *Dsc Dissertation, Universidade Federal do Rio de Janeiro*.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R. & Selby, R. (1995), 'Cost models for future software life cycle processes: Cocomo 2.0', *Annals of Software Engineering* **1**(1), 57–94. **DOI** 10.1007/BF02249046. ISSN 1573-7489. URL https://doi.org/10.1007/BF02249046.

Boehm, B. W. (1984), 'Software engineering economics', *IEEE Transactions on Software Engineering* **SE-10**(1), 4–21. **DOI** 10.1109/TSE.1984.5010193. ISSN 0098-5589.

Bouer, R. & Carvalho, M. M. d. (2005), 'Metodologia singular de gestão de projetos: condição suficiente para a maturidade em gestão de projetos?', *Production* **15**, 347 - 361. ISSN 0103-6513.

Chaves Lessa Schots, N., Gonçalves, T., Figueiredo Magalhaes, R., Regina Rocha, A., Santos, G. & Oliveira, K. (2014), 'Supporting software process performance analysis through a knowledge-based environment'.

Cunha, A., Fernandes Thomaz, J. & Moura, H. (2008), 'Um modelo de avaliação da produtividade de projetos de software baseado em uma abordagem multicritério'.

DeMarco, T. & Lister, T. (1999), *Peopleware: Productive Projects and Teams*, Dorset House Pub.

Florac, W. A. & Carleton, A. D. (1999), *Measuring the Software Process: Statistical Process Control for Software Process Improvement*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Guia Geral, M. (2016), 'Mps. br-melhoria de processo do software brasileiro'.

Gulezian, R. (1991), 'Reformulating and calibrating cocomo'.

Hair, J., Anderson, R. & Tatham, R. (2005), *Analise Multivariada de Dados*, Bookman.

Hamdan, K. & Madi, M. (2011*a*), Software project effort: Different methods of estimation, *in* 'Communications and Information Technology (ICCIT), 2011 International Conference on', pp. 15–18.
DOI 10.1109/ICCITECHNOL.2011.5762670.

Hamdan, K. & Madi, M. (2011*b*), Software project effort: Different methods of estimation, *in* 'Communications and Information Technology (ICCIT), 2011 International Conference on', IEEE, pp. 15–18.

Haufe, M. I. (2001), 'Estimativa da produtividade no desenvolvimento de software'.

Jorgensen, M. (2010), 'Identification of more risks can lead to increased over-optimism of and over-confidence in software development effort estimates', *Information and Software Technology* **52**(5), 506 – 516.
DOI http://dx.doi.org/10.1016/j.infsof.2009.12.002. ISSN 0950-5849. URL http://www.sciencedirect.com/science/article/pii/S0950584909002146, TAIC-PART 2008 TAIC-PART 2008.

Jorgensen, M. (2011), 'Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation', *Information and Software Technology* **53**(12), 1382 – 1390.
DOI http://dx.doi.org/10.1016/j.infsof.2011.07.001. ISSN 0950-5849. URL http://www.sciencedirect.com/science/article/pii/S0950584911001595.

Jorgensen, M. & Sjoberg, D. I. (2004), 'The impact of customer expectation on software development effort estimates', *International Journal of Project Management* **22**(4), 317 – 325. **DOI** http://dx.doi.org/10.1016/S0263-7863(03)00085-1. ISSN 0263-7863. URL http://www.sciencedirect.com/science/article/pii/S0263786303000851.

Jørgensen, M. & Sjøberg, D. I. (2004), 'The impact of customer expectation on software development effort estimates', *International Journal of Project Management* **22**(4), 317 – 325. **DOI** http://dx.doi.org/10.1016/S0263-7863(03)00085-1. ISSN 0263-7863. URL http://www.sciencedirect.com/science/article/pii/S0263786303000851.

Lins, B. F. (1993), 'Ferramentas básicas da qualidade', *Ciência da Informação* **22**(2), 153–161.

Lopez-Martin, C., Isaza, C. & Chavoya, A. (2012), 'Software development effort prediction of industrial projects applying a general regression neural network', *Empirical Software Engineering* **17**(6), 738–756. **DOI** 10.1007/s10664-011-9192-6. ISSN 1382-3256. URL http://dx.doi.org/10.1007/s10664-011-9192-6.

Menzies, T., Yang, Y., Mathew, G., Boehm, B. & Hihn, J. (2016), 'Negative Results for Software Effort Estimation', *ArXiv e-prints*.

Montgomery, D. C., Peck, E. A. & Vining, G. G. (2015), *Introduction to linear regression analysis*, John Wiley & Sons.

Montoni, M. A. (2010), 'Uma investigação sobre os fatores críticos de sucesso em iniciativas de melhoria de processos de software', *Doutorado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro*.

Montoni, M., Kalinowski, M., Lupo, P., Abrantes, J. F., Ferreira, A. I. F. & Rocha, A. R. (2007), 'Uma metodologia para desenvolvimento de modelos de desempenho de processos para gerencia quantitativa de projetos de software', *Proc. of the VI Simpósio Brasileiro de Qualidade de Software (SBQS), Porto de Galinhas*.

Moreira, P. & Souza, C. (2008), 'Utilização de gráficos de controle para gerência quantitativa de processos de software', *UFAP*.

Nguyen, V., Huang, L. & Boehm, B. (2010), 'An analysis of trends in productivity and cost drivers over years'.

Nguyen, V., Steece, B. & Boehm, B. (n.d.), A constrained regression technique for cocomo calibration, *in* 'Proceedings of the 2nd ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM', pp. 213–222.

Nomelini, Q. S. S., Ferreira, E. B. & Oliveira, M. S. d. (2009), 'Estudos dos padrões de não aleatoriedade dos gráficos de controle de Shewhart: um enfoque probabilístico', *Gestão & Produção* **16**, 414 – 421. ISSN 0104-530X.

Oliveira, E., Conte, T., Cristo, M. & Mendes, E. (2016), Software project managers' perceptions of productivity factors: Findings from a qualitative study, *in* 'Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement', ESEM '16, ACM, New York, NY, USA, pp. 15:1–15:6. **DOI** 10.1145/2961111.2962626. URL http://doi.acm.org/10.1145/2961111.2962626.

Peinado, J. & Graeml, A. R. (2007), 'Administração da produção', *Operações industriais e de serviços. Unicenp*.

Putnam, L. & Myers, W. (2013), *Five Core Metrics: The Intelligence Behind Successful Software Management*, Dorset House eBooks, Pearson Education.

Putnam, L. H. (1978), 'A general empirical solution to the macro software sizing and estimating problem', *IEEE Transactions on Software Engineering* **SE-4**(4), 345–361. **DOI** 10.1109/TSE.1978.231521. ISSN 0098-5589.

Rocha, A. d., Souza, G. d. S. & Barcellos, M. (2012), *Medição de software e controle estatístico de processos*.

Sage, A. & Rouse, W. (2011), *Economic Systems Analysis and Assessment: Intensive Systems, Organizations,and Enterprises*, Wiley Series in Systems Engineering and Management, Wiley.

Sales, M. (2006), 'Diagrama de pareto', *Recuperado el*.

Salviano, C. (2006), Uma proposta orientada a perfis de capacidade de processo para evolução da melhoria de processo de software (in Portuguese)(A proposal oriented by process capability profiles for the evolution of software process improvement), PhD thesis, PhD thesis, FEEC Unicamp, Campinas, SP, Brazil.

Schots, N. C. L. & Rocha, A. R. (2012), Um workflow para controle estatístico de processos em software, *in* 'VIII Workshop Anual do MPS, Campinas, São Paulo'.

Selby, R. (2007), *Software Engineering: Barry W. Boehm's Lifetime Contributions to Software Development, Management, and Research*, Practitioners Series, Wiley.

Silva Filho, R. C. (2012), 'Avaliação do desempenho potencial de projetos de software com simulação de processos', *Doutorado, Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro, Rio de Janeiro*.

Silveira, G. d. A., Sbragia, R. & Kruglianskas, I. (2013), 'Fatores condicionantes do nível de maturidade em gerenciamento de projetos: um estudo empírico em empresas brasileiras', *Revista de Administração* **48**(3), 574 – 591. **DOI** https://doi.org/10.5700/rausp1107. ISSN 0080-2107. URL http://www.sciencedirect.com/science/article/pii/S0080210716302941.

Simões, C. A., da Rocha, A. R. C., Alberto, C., Rocha, I., Cavalcanti, A. R., Mello, M., PARA, M., MATURIDADE, O. D. A. & SOFTWARE, E. P. D. (2011), Repositório de medidas para organizações de alta maturidade em processos de software.

Teixeira, W. J. & Sanches, R. (2000), Modelos de estimativas de custo de software cocomo & cocomo ii, Technical report, São Carlos.

Tonini, A. C., Carvalho, M. M. d. & Spinola, M. d. M. (2008), 'Contribuição dos modelos de qualidade e maturidade na melhoria dos processos de software', *Production* **18**, 275 – 286. ISSN 0103-6513.

Wagner, S. & Ruhe, M. (2008), 'A systematic review of productivity factors in software development', *Proceedings of 2nd International Workshop on Software Productivity Analysis and Cost Estimation (SPACE 2008)*.

Walston, C. E. & Felix, C. P. (1977), 'A method of programming measurement and estimation', *IBM Systems Journal* **16**(1), 54–73. **DOI** 10.1147/sj.161.0054. ISSN 0018-8670.

Weber, K. C., Rocha, A. R., Alves, Â., Ayala, A. M., Gonçalves, A., Paret, B., Salviano, C., Machado, C. F., Scalet, D., Petit, D. et al. (2004), Modelo de referência para melhoria de processo de software: uma abordagem brasileira, *in* 'XXX Conferencia Latinoamericana de Informatica (CLEI2004), Sesión', Vol. 13, pp. 20–10.

Wheeler, D. (1993), *Understanding Variation: The Key to Managing Chaos*, SPC Press.

Este trabalho foi redigido em LaTeX utilizando uma modificação do estilo IC-UFAL. As referências bibliográficas foram preparadas no JabRef e administradas pelo BibTeX com o estilo LaCCAN. O texto utiliza fonte Kerkis e os elementos matemáticos a família tipográfica Fourier-GUTenberg, ambas em corpo de 12 pontos. A numeração dos capítulos segue com a familia tipográfica Art Nouveau Initialen.
As citações que iniciam cada capítulo estão disponíveis no pacote fortunes do R.