UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

UM MODELO DE AVALIAÇÃO DE DESEMPENHO PARA SUPORTE AO PLANEJAMENTO DO PROCESSO DE MUDANÇA DE SOFTWARE

MARCELY DANIELA DOS SANTOS DIAS

DISSERTAÇÃO DE MESTRADO

Recife

2010

UM MODELO DE AVALIAÇÃO DE DESEMPENHO PARA SUPORTE AO PLANEJAMENTO DO PROCESSO DE MUDANÇA DE SOFTWARE

A apresentação dessa dissertação é exigência do Programa de Pós-graduação: Ciência da Computação da Universidade Federal de Pernambuco, para obtenção do título de Mestre.
Orientador: Prof. Dr. Paulo Romero Martins Maciel
Co-Orientadora: Profa. Msc. Teresa Maria Medeiros Maciel

Recife

2010

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Recife, ..., ... de 2010.

A Comissão Examinadora, abaixo assinada, aprova a dissertação Um Modelo de Avaliação de Desempenho para Suporte ao Planejamento do Processo de Mudança de Software, elaborada por Marcely Daniela dos Santos Dias, como requisito parcial para a obtenção do Grau de Mestre em Ciência da Computação.

Comissão Examinadora:

Profa. Dra. Maria da Conceição Moraes Batista

Prof. Dr. Ricardo Massa Ferreira Lima

Prof. Dr. Paulo Romero Martins Maciel

Dedico esta dissertação à minha família.

AGRADECIMENTOS

- A Deus por ter conseguido conquistar e vencer mais esta etapa na minha vida.
- Ao meu orientador Prof. Paulo Maciel pela paciência e por ter acreditado acima de tudo em mim e no meu trabalho.
- Agradeço também a amiga Profa. Teresa Maciel por todo apoio, orientação, ajuda e amizade.
- A todos os meus amigos que sempre me deram apoio para concluir este trabalho. Dentre eles meu agradecimento especial a Fabiana Leonel por todo apoio, ajuda e força no momento de fraqueza e dúvidas.
- Aos meus pais Sergio e Sandra, bem como a minha irmã Cyntia, pois mesmo estando distantes me deram todo apoio necessário.
- Ao meu marido Jonatas pela paciência em todos os momentos desta etapa.

"Para realizar grandes tarefas precisa-se de paixão e audácia."

Che Guevara.

RESUMO

A engenharia de software se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção deste sistema, depois que ele entrou em operação (SOMMERVILLE, 2005). Tudo isso acontece através de um processo de software, ou seja, através da execução de um conjunto de atividades e resultados associados que geram um produto de software. Cada organização possui seus processos, a sua forma de desenvolver software, seu conjunto de atividades, métodos e ferramentas de apoio, mas a grande maioria não conhece o desempenho da execução dos mesmos. Um exemplo disso é na área de Gerência de mudanças, geralmente negligenciada da Gerência de configuração, responsável por mapear, para cada mudança efetuada no sistema, qual foi o motivo que gerou esta mudança. Essa área inclui o processo de tomada de decisão sobre que mudanças realmente devem ser realizadas e o processo para executá-las. As organizações geralmente não se preocupam com os custos de execução das atividades desse processo de mudança no software por falta de um planejamento, principalmente na análise de impacto e tomada de decisão de execução dessas mudanças. Dessa forma, este trabalho propõe um modelo em redes de Petri estocástica para avaliar o impacto dos custos na execução do processo de mudança de software. Além disso, uma metodologia de avaliação de desempenho é proposta, com o intuito de auxiliar os processos de modelagem e de avaliação. Por fim, estudos de caso serão apresentados mostrando a aplicabilidade do trabalho em comento.

Palavras-chave: Processo, mudança de software, análise de desempenho, custos, modelagem, SPN.

ABSTRACT

Software engineering deals with all aspects of software production, from the early stages of system specification to maintenance of this system, after it came into operation. All this happens through a process of software, by performing a set of activities and associated results that generate a software product. Each organization has its processes, the way they develop software, its range of activities, methods and tools to support, but most do not know the performance of process execution. An example is the area of management changes, often a neglected area of configuration management, responsible for mapping, for every change made in the system, the reason that generated this change. This area includes the process of deciding on what changes should actually be performed and the process to implement changes. Organizations generally do not care about the costs of implementing the activities in this Software Change Process by a lack of planning, especially in impact analysis and decision to implement those changes. This work proposes a stochastic Petri net model to evaluate the impact of costs in implementing the Software Change Process. Furthermore, a performance evaluation methodology is proposed, with the goal of helping the process of modeling and evaluation. Lastly, case studies will be presented showing the applicability of this work.

Keywords: Process, software change, performance analysis, cost, modeling, SPN.

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO 14

1.1 CONTEXTO 14
1.2 MOTIVAÇÃO 15
1.3 OBJETIVOS 18
1.4 ESTRUTURA DA DISSERTAÇÃO 20

CAPÍTULO 2 - TRABALHOS RELACIONADOS 21

2.1 TRABALHOS 21
2.2 CONSIDERAÇÕES FINAIS 30

CAPITULO 3 - FUNDAMENTOS 31

3.1 PROCESSO DE SOFTWARE 31
3.2 PROCESSO DE MUDANÇA DE SOFTWARE 36
3.2.1 Problemas da Mudança do Software 43
3.3 REDES DE PETRI ESTOCÁSTICA 46
3.3.1 Propriedades das Redes de Petri 55
3.3.2 Aproximação por fases 57
3.4 CONSIDERAÇÕES FINAIS 60

CAPITULO 4 – METODOLOGIA DE AVALIAÇÃO DE DESEMPENHO 61

4.1 INTRODUÇÃO 61
4.2 CONHECIMENTO E LEVANTAMENTO DE INFORMAÇÕES DO PROCESSO DE MUDANÇA DE SOFTWARE 63
4.3 SELEÇÃO DAS MÉTRICAS DE DESEMPENHO 65
4.4 GERAÇÃO DO MODELO DE REDE DE PETRI ABSTRATO 65
4.5 MEDIÇÃO E TRATAMENTO DOS DADOS 66
4.6 REFINAMENTO E ANÁLISE DO MODELO DE REDES DE PETRI 67
4.7 VALIDAÇÃO DO MODELO 68
4.8 AVALIAÇÃO, INTERPRETAÇÃO DOS DADOS E APRESENTAÇÃO DOS RESULTADOS 70
4.9 CONSIDERAÇÕES FINAIS 71

CAPITULO 5 – MODELO PROPOSTO 72

5.1 PROCESSO DE MUDANÇA DE SOFTWARE 72
5.2 MODELOS 73
5.2.1 Solicitação 73
5.2.2 Buffer 74
5.2.3 CCB 75
5.2.4 Baseline 77
5.3 VALIDAÇÃO 80
5.3.1 Compreensão e levantamento de informações do processo e seleção de métricas. 80
5.3.2 Geração do Modelo Abstrato de Desempenho 82
5.3.3 Medição 84
5.3.4 Tratamento dos Dados 84
5.3.5 Refinamento do Modelo 86
5.3.6 Validação do Modelo 88
5.4 CONSIDERAÇÕES FINAIS 92

CAPÍTULO 6 - ESTUDO DE CASO 93

6.1 Cenário 1 93
6.2 Cenário 2 99
6.3 Cenário 3 104
6.4 Considerações Finais 110

CONCLUSÃO 111

7.1 Contribuições, Limitações e Dificuldades 112
7.2 Trabalhos Futuros 113

REFERÊNCIAS 114

APÊNDICE A 120

I. Intervalo de Confiança 120
A. Cálculo do Intervalo de Confiança da Média 121
I. Quando o σ (desvio padrão) é conhecido 121
II. Quando o σ (desvio padrão) é desconhecido 121
III. Para diferença (Teste t-emparelhado) 121
B. Método Bootstrap 122

APÊNDICE B 123

LISTA DE FIGURAS

FIGURA 1: DIAGRAMA DE ATIVIDADES 36
FIGURA 2: PROCESSO DE MUDANÇA DE SOFTWARE 39
FIGURA 3: EXEMPLO DE SOLICITAÇÃO DE MUDANÇA PREENCHIDA – (A) 40
FIGURA 4: EXEMPLO DE SOLICITAÇÃO DE MUDANÇA PREENCHIDA – (B) 41
FIGURA 5: EXEMPLO DE SOLICITAÇÃO DE MUDANÇA PREENCHIDA – (C) 41
FIGURA 6: PROCESSO DE MODELAGEM SIMPLES 46
FIGURA 7: ELEMENTOS DE UMA REDE DE PETRI 48
FIGURA 8: PERÍODOS DO DIA 48
FIGURA 9: LINHA DE PRODUÇÃO 50
FIGURA 10: ANO LETIVO REPRESENTADO GRAFICAMENTE EM REDES DE PETRI 51
FIGURA 11: EXEMPLO DE UMA REDE DE PETRI ESTOCÁTICA 54
FIGURA 12: SUB-REDE GSPN PARA REPRESENTAR DISTRIBUIÇÕES POLINÔMIO-EXPONENCIAIS UTILIZANDO *MOMENT MATCHING* 58
FIGURA 13: FLUXOGRAMA DA METODOLOGIA DE AVALIAÇÃO DE DESEMPENHO. 62
FIGURA 14: MODELO REDES DE PETRI ABSTRATO PARA O PROCESSO DE MUDANÇA DE SOFTWARE. 66
FIGURA 15: MODELO SPN REFINADO PARA O PROCESSO DE MUDANÇA DE SOFTWARE. 68
FIGURA 16: PROCESSO DE MUDANÇA DE SOFTWARE 72
FIGURA 17: MODELO SPN PARA A SOLICITAÇÃO 74
FIGURA 18: MODELO SPN PARA O BUFFER. 75
FIGURA 19: MODELO SPN PARA A ANÁLISE DE IMPACTO FEITA PELO CCB. 76
FIGURA 20: MODELO SPN COMPOSTO COM A SOLICITAÇÃO, BUFFER E ANÁLISE DE IMPACTO FEITA PELO CCB 76
FIGURA 21: MODELO SPN PARA GERAÇÃO DE *BASELINE* PELO CM. 78
FIGURA 22: MODELO SPN ABSTRATO COMPOSTO POR TODOS OS MODELOS PROPOSTOS NESSE TRABALHO 79
FIGURA 23: MODELO ABSTRATO DO PROCESSO DE MUDANÇA DE SOFTWARE 83
FIGURA 24: GRÁFICO *BOXPLOT* 86
FIGURA 25: MODELO REDES DE PETRI REFINADO DO PROCESSO DE MUDANÇA DE SOFTWARE. 88
FIGURA 26: COMPARAÇÃO DA MÉDIA DE SOLICITAÇÕES DE MUDANÇA SISTEMA E MODELO 91
FIGURA 27: CUSTO REAL X CUSTO SUGERIDO – CENÁRIO 1 98
FIGURA 28: CUSTO REAL X CUSTO SUGERIDO – CENÁRIO 2 104
FIGURA 29: CUSTO REAL X CUSTO SUGERIDO – CENÁRIO 3 109

LISTA DE TABELAS

TABELA 1: ATRIBUTOS PARA TOMADA DE DECISÃO SOBRE AS SOLICITAÇÕES DE MUDANÇA... **ERRO! INDICADOR NÃO DEFINIDO.**
TABELA 2: INTERPRETAÇÕES PARA OS LUGARES E TRANSIÇÕES. 49
TABELA 3: *SWITCHING DISTRIBUTION* DA GSPN DESCRITA NA FIGURA 6..... **ERRO! INDICADOR NÃO DEFINIDO.**
TABELA 4: MÉTRICAS DO MODELO CCB. 77
TABELA 5: MÉTRICAS DO MODELO MUDANÇA FINALIZADA. 79
TABELA 6: RESUMO DOS PROJETOS ANALISADOS 81
TABELA 7: RESUMO ESTATÍSTICO DOS PROJETOS. 84
TABELA 8: CONFIGURAÇÃO DOS EXPERIMENTOS. 89
TABELA 9: RESUMO ESTATÍSTICO DOS PROJETOS. 90
TABELA 10: DADOS DO CCB - CENÁRIO 1 95
TABELA 11: DADOS DO CCB PLANEJADOS – CENÁRIO 1 96
TABELA 12: DADOS DO CCB - CENÁRIO 2 100
TABELA 13: DADOS DO CCB PLANEJADOS – CENÁRIO 2 102
TABELA 14: DADOS DO CCB - CENÁRIO 3 106
TABELA 15: DADOS DO CCB PLANEJADOS – CENÁRIO 3 107

LISTA DE ABREVIATURAS

CCB	Comitê do Controle de Mudança
UFPE	Universidade Federal de Pernambuco
CM	*Configuration Manager*
UML	*Unified Modeling Language*
PMLs	*Process Modeling Languages*
RdP	Redes de Petri Estocástica
GSPN	*Generalized Stochastic Petri Nets*
SPN	*Stocasthic Petri Net*
CR	*Change Request* (Solicitação de Mudança)
CCB	*Change Control Board* (Comitê de Controle de Mudança)

CAPÍTULO 1 - INTRODUÇÃO

Este capítulo provê uma breve introdução à área de gerência de configuração e mudança de software e, em seguida uma contextualização sobre a gerência de mudança, destacando a importância de existir um processo de gerência de mudança de software, como também, a necessidade da avaliação de desempenho desse processo para suporte ao planejamento do mesmo no projeto.

Em seguida, são apresentados a motivação, trabalhos relacionados e a proposta deste trabalho. Assim como, seus objetivos.

1.1 Contexto

A engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção deste sistema, depois que ele entrou em operação (SOMMERVILLE, 2005). Tudo isso acontece através de um processo de software, ou seja, através da execução de um conjunto de atividades e resultados associados que geram um produto de software.

De acordo com (SOMMERVILLE, 2005) é impossível produzir sistemas de qualquer tamanho que não precisem ser modificados. Quando o software é colocado em uso, novos requisitos surgem, e os requisitos existentes são modificados. Partes do software podem precisar ser modificadas para corrigir os erros encontrados na operação, melhorar seu desempenho e outras características não funcionais. Isso significa que, depois dos softwares serem entregues aos seus clientes, eles sempre evoluem, em respostas às exigências de mudanças.

A engenharia de software possui a área Gerência de Configuração de Software, Gerência de Configuração ou ainda Gestão de Configuração de Software responsável por fornecer o apoio para o desenvolvimento de

software. Suas principais atribuições são: o controle de versões, o controle de mudanças e a auditoria das configurações (Wikipédia).

Roger Pressman (PRESSMAN, 2000), afirma que a gerência de configuração de software (GCS) é o conjunto de atividades projetadas para controlar as mudanças pela identificação dos produtos do trabalho que serão alterados, estabelecendo um relacionamento entre eles, definindo o mecanismo para o gerenciamento de diferentes versões destes produtos, controlando as mudanças impostas, e auditando e relatando as mudanças realizadas.

Em outras palavras, a Gerência de Configuração de Software tem como objetivo responder a perguntas do tipo o que mudou e quando?, por que mudou?, quem fez a mudança e “podemos reproduzir esta mudança?” (Wikipédia).

Cada uma dessas perguntas corresponde a uma das atividades realizadas pela Gerência de Configuração de Software. O controle de versão é capaz de dizer o que mudou e quando mudou. O controle de mudanças é capaz de atribuir os motivos a cada uma das mudanças. A Auditoria por sua vez responde as duas últimas perguntas: Quem fez a mudança e podemos reproduzir a mudança (Wikipédia)?

Adicionalmente, com o intuito de ajudar no planejamento do processo de mudanças em um projeto de software, este trabalho avaliará o desempenho e a confiabilidade do processo de mudança de software apoiando na redução de custos na execução deste processo de mudança de software.

1.2 Motivação

Hoje em dia, segundo (SOMMERVILLE, 2005), as organizações dependem inteiramente de seus sistemas de software e investem muito neles. Elas precisam investir em alterações nesses sistemas para manter o valor dos mesmos.

A manutenção destes softwares é uma parte integrante do ciclo de vida do software, mas, historicamente, ela não recebeu o mesmo grau de atenção

que as outras fases tiveram. Para realizar o controle dessas mudanças no software é necessário ter um processo de manutenção do software. Segundo (SOMMERVILLE, 2005) os processos de manutenção variam dependendo do tipo de software que está em manutenção, dos processos de desenvolvimento de software utilizados em uma organização e do pessoal envolvido no processo.

Os processos de software formam uma base para o controle gerencial de projetos de software e estabelecem o contexto no qual os métodos técnicos são aplicados, os produtos de trabalho (modelos, documentos, dados, relatórios, formulários, etc) são produzidos, os marcos são estabelecidos, a qualidade é assegurada e as modificações são adequadamente geridas (PRESSMAN, 2000). Dada à complexidade envolvida nos processos de software, a comunidade de Engenharia de Software sentiu a necessidade de desenvolver linguagens para a modelagem de processos de software, denominadas Linguagens de Modelagem de Processos (*Process Modeling Languages - PMLs*), com o objetivo de formalizar a definição dos processos, bem como possibilitar o suporte a todo o ciclo de vida do processo (ABDALA, 2004) (REIS, 2004).

Essas Linguagens de Modelagem de Processo são utilizadas a fim de facilitar a definição de processos, sua compreensão e garantir uma alta qualidade no que diz respeito à aplicação do processo em uma Organização, em um determinado Projeto. Entre as linguagens de modelagem a UML (*Unified Modeling Language*) é um dos mais importantes padrões mantidos pelo grupo OMG. A UML (Booch, et al., 2000; Guedes, 2004; OMG, 2005) vem sendo considerada como a linguagem de modelagem (gráfica) padrão no desenvolvimento orientado a objetos, oferecendo apoio à criação de modelos independentes de plataforma, nos quais os conceitos são separados da semântica existente nos modelos da implementação (OMG, 2005). O diagrama de atividades da UML determina as regras essenciais de seqüência que se deve seguir para a execução de um processo (OMG, 2005). O diagrama de atividades é um dos diagramas da UML utilizados para a modelagem de processo (OMG, 2005).

Definir processos de software adequados que garantam a qualidade do produto e produtividade no desenvolvimento/manutenção tem representado um desafio para as indústrias de software. As organizações possuem uma grande dificuldade na hora de identificar e estabelecer o processo mais adequado a um determinado Projeto. Ao optar por um processo, dependendo das características do projeto, as organizações não têm visibilidade do desempenho do processo, não têm uma garantia de que eles resultarão em um menor custo e qualidade.

Para melhorar e ter visibilidade da execução do processo de software, as organizações buscam se tornar aderentes a modelos de qualidade como a ISO (ISO), ao CMMI – Capability Maturity Model Integration (CMMI, 2006) e ao MPS-Br – Modelo de Processo de Software Brasileiro (MPS-Br). Nas avaliações destes modelos, elas possuem a visão se o processo definido está atendendo ao modelo de qualidade e sugestões de melhorias para o mesmo. Mas, poucas as organizações possuem a informação de quão bom é o desempenho desse processo através de dados estatísticos. As organizações que possuem níveis maiores de maturidade como o nível 5 do CMMI e Nível A do MPS-Br podem possuir estas informações (CMMI, 2006) (MPS-Br).

As redes de Petri foram propostas por *Carl Adam Petri* em 1962 na sua tese de doutoramento (Petri, 1962). Segundo (Marsan, et al., 1995), as redes de Petri permitem a especificação de sistemas concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e estocásticos. Desde então, as redes de Petri vêm sendo aplicadas nas mais diversas áreas, indo desde a ciência da computação, até as áreas de administração de empresas e as biológicas. Desde seu surgimento, diversas extensões foram propostas para esse formalismo. Entre as extensões existentes, estão as *Stochastic Petri Nets* (SPN), que incluem a noção de tempo estocástico. Esse tipo de rede permite a avaliação de sistemas a partir de métricas, por exemplo, a probabilidade de utilização de um recurso computacional (processador, disco, memória, entre outros.), ou ainda a disponibilidade e/ou confiabilidade desses recursos. Essas métricas podem ser computadas para um determinado intervalo de tempo (análises transientes), ou quando o sistema entra em equilíbrio (análises estacionárias) (German, et al., 1995). Alem disso, satisfazendo algumas

propriedades (Murata, 1989), os resultados das métricas podem ser obtidos de forma analítica, como também por simulação do modelo (Chung, 2004).

Através do mapeamento do Modelo utilizando Rede de Petri Estocásticas é possível obter dados estatísticos que ajudarão as organizações a analisar melhor o desempenho dos seus processos e sua aplicabilidade.

No caso deste trabalho, foi possível realizar uma descoberta precoce de como melhor planejar o envolvimento de pessoas na execução do processo de solicitação de mudança de software de uma organização. Esse planejamento poderá reduzir a quantidade de mudanças, reduzindo os custos de pessoal, evitando dessa forma perdas de recursos financeiros e recursos humanos em outras atividades mais prioritárias. Foi realizada a avaliação de desempenho a partir dos resultados estatísticos obtidos na análise e validação dos dados permitindo adquirir conclusões a respeito do desempenho e aplicabilidade do processo de mudança.

Este projeto tem como objetivo principal definir um modelo de avaliação de desempenho para suporte ao planejamento da alocação dos recursos que serão envolvidos na análise de impacto da mudança no processo de mudança de software que foi modelado pelo diagrama de atividade da UML e mapeado em uma rede de Petri estocástica.

1.3 Objetivos

No momento de analisar as mudanças de software deve existir uma atenção da equipe e do gestor do projeto no planejamento da execução do processo de mudança de software ao alocar os papéis corretos para realização da análise de impacto dessas mudanças.

Este trabalho propõe a definição de um modelo de avaliação de desempenho para o suporte ao planejamento do processo de mudança do software modelado no diagrama de atividades e mapeado em uma rede de Petri. Esse mapeamento tem por objetivo a análise e verificação do desempenho do processo de mudança do software, estudando suas dificuldades na execução, prazos, esforço e custo no envolvimento do comitê de controle de mudança na análise de impacto.

Em resumo, este trabalho consiste em um mecanismo de analisar e verificar as dificuldades, esforço e custo investido no processo de mudança do software em uma organização, através de uma metodologia. As métricas identificadas e analisadas servirão de instrumento para melhor entendimento das dificuldades e montar estratégias para facilitar o uso do processo.

No início deste projeto, foram realizados estudos em trabalhos relacionados que propõe o entendimento do processo mudança do software, as dificuldades em implantar e utilizar este processo, a definição da metodologia para avaliação de desempenho do processo de mudança do software, o mapeamento de linguagens semiformais em linguagens formais. Após isso, foi desenvolvido um mecanismo de mapeamento em uma rede de Petri estocástica. Esse mecanismo consiste na derivação dos elementos básicos do diagrama de atividades para uma rede de Petri.

Foi definida a metodologia, o processo de mapeamento foi realizado, analisando e verificando os dados obtidos através de métricas.

Para validação da proposta, foi usado dado de projetos de uma empresa de desenvolvimento de software da cidade de Recife/ Pernambuco.

Mais especificamente este trabalho possui os seguintes objetivos:

- desenvolver um modelo de avaliação de desempenho para realizar análises qualitativas e quantitativas do processo de mudança de *software* para suporte ao planejamento da execução deste processo:
    - dando visibilidade ao gestor do projeto da utilização e custo do comitê de controle de mudança;
    - apoiando o gestor do projeto no planejamento da alocação dos papeis mais adequados no comitê de controle de mudança, reduzindo custos relacionados a execução desta atividade no projeto;
- desenvolver um método para o mapeamento dos diagramas comportamentais da UML em rede de Petri estocástica. Neste

trabalho, o diagrama de atividades é adotado;

- definir uma metodologia que auxilie a modelagem e avaliação de desempenho do processo de mudança de software;
- definir métricas para estimar a utilização e custo do processo de mudança de software.

1.4 Estrutura da Dissertação

O Capítulo 2 Trabalhos relacionados, descreve os trabalhos relacionados, comentando seus objetivos e a que eles propõem, contribuindo para a realização deste trabalho.

O Capítulo 3 Fundamentos, introduz os conceitos fundamentais sobre engenharia de software, processo de software, processo de mudança de software, os conceitos fundamentais sobre redes de Petri.

O Capítulo 4 Metodologia, apresenta uma metodologia de avaliação de desempenho do processo de mudança de software. O Capítulo 5 Modelo Proposto, apresenta os modelos SPN do processo de mudança de software e o estudo de caso utilizado para validar tanto os modelos SPN propostos, quanto a metodologia de avaliação.

O Capítulo 6 Estudo de Caso, apresenta um estudo de caso no qual foi aplicada a metodologia. Finalmente, o capítulo da Conclusão conclui o trabalho e apresenta os trabalhos futuros.

CAPÍTULO 2 - TRABALHOS RELACIONADOS

Este capítulo apresenta alguns trabalhos relacionados a este trabalho no que diz respeito a processo, modelagem de processo, mudança de software, redes de Petri e avaliação de desempenho.

2.1 Trabalhos

Alguns conceitos de engenharia de software, seus processos e manutenção de software foram abordados por Rigo (RIGO, 2008). As Normas ISO/IEC 12207 e ISO/IEC FDIS 14764, também são analisadas neste trabalho de Rigo (RIGO, 2008), as quais definem modelos para auxiliar no processo de manutenção do software. Vale salientar que nesse trabalho investigaram-se as tarefas da Norma ISO/IEC FDIS 14764, a qual trata especificamente do processo de manutenção de software, e realizou-se uma comparação com o processo de manutenção de software da web. Este trabalhou orientou o entendimento de processos de manutenção/ mudanças de software aplicados no mercado.

Nesse trabalho, (RIGO, 2008) cita que o termo “engenharia” engloba conceitos de criação, construção, análise, desenvolvimento e manutenção. A engenharia de software utiliza dos métodos e princípios da engenharia para tornar a produção de software mais eficiente e confiável, além de almejar a redução de custos e prazos no desenvolvimento do software, gerenciando o processo de desenvolvimento e aplicando conceitos de qualidade. A engenharia de software abrange um conjunto de três elementos fundamentais: métodos, ferramentas e processos para auxiliar no controle do processo de desenvolvimento do software.

Rigo (RIGO, 2008) explica que o processo de desenvolvimento de software utiliza atividades ordenadas para a geração de um produto de software, sendo essas atividades a especificação, desenvolvimento, validação e manutenção/evolução do software. Esta última atividade é vista pelos

autores, como sendo uma atividade importante e indispensável no processo de ciclo de vida do software, tanto para aplicações tradicionais como para web.

Segundo citado por Rigo (RIGO, 2008) a manutenção/evolução de software é um processo de modificação de software após ele ser colocado em uso, não dependendo o tamanho ou complexidade da aplicação ele sempre vai evoluir e modificar. Essas modificações podem variar das mais simples, como correção de erros de código, as mais específicas como acomodação de requisitos.

Os engenheiros de softwares preocupados com a demanda emergente no mercado por aplicativos web, sentiram a necessidade de utilizar técnicas e métodos padronizados a esse tipo de aplicação, tendo assim o surgimento da engenharia da web para tratar dessa carência. A engenharia da web aplica princípios científicos sólidos, de engenharia e de gestão, e abordagens disciplinadas e sistemáticas para alcançar o sucesso no desenvolvimento, implantação e manutenção de sistemas de alta qualidade baseados na web (RIGO, 2008).

Rigo (RIGO, 2008) comenta que para tratar do processo de desenvolvimento de software, a literatura apresenta a Norma ISO/IEC 12207. Já a Norma ISO/IEC FDIS 14764 é elaborada com base na Norma ISO/IEC 12207, e trata especificamente do processo de manutenção de software. As Normas estabelecem uma estrutura a ser seguida e orientada para o desenvolvimento do software, a qual pode ser referenciada pela indústria do software.

Rigo (RIGO, 2008) explica que a engenharia da web deixa lacunas em relação ao processo de manutenção para seus aplicativos. Com isso, o objetivo deste trabalho foi investigar o processo de manutenção na engenharia de software através das Normas ISO/IEC 12207 e a ISO/IEC FDIS 14764, para possível adequação no processo de manutenção de software web. A análise foi realizada através da comparação das tarefas da Norma ISO/IEC FDIS 14764 com o processo de manutenção de software na web. Após a análise realizada sugeriu possíveis adequações das tarefas da Norma para uso no processo de manutenção de software na web.

O autor (RIGO, 2008) no primeiro capítulo deste trabalho apresenta definições de engenharia de software, assim como os custos, problemas, previsão e equipe envolvida nessa etapa de manutenção, o que serviu de base para o entendimento do processo de manutenção do software. Já no segundo capítulo apresenta os conceitos de engenharia da web, seus processos, a equipe envolvida. Assim como, algumas definições de atributos para aplicativos web, evidenciando sua importância nessa demanda emergente por produtos ágeis e confiáveis. As camadas da engenharia de aplicativos web também é conceituada neste capítulo juntamente com seus processos. Além de apresentar o processo de gerenciamento de mudança na web segundo visão de Pressman (PRESSMAN, 2000). No terceiro capítulo, o autor (RIGO, 2008) apresenta conceitos relativos as Normas ISO/IEC 12207 e ISO/IEC FDIS 14764 que tratam do processo de manutenção de software. Descreve o processo de manutenção de software estabelecido pelas Normas, como as atividades com suas respectivas entradas, tarefas e saídas. E no quarto e último capítulo apresenta o processo utilizado para análise e as possíveis adequações da Norma para o uso na manutenção de software na web. Assim como, os resultados da análise realizada com a comparação do processo de manutenção de software na web e as tarefas apresentadas pela Norma ISO/IEC FDIS 14764.

Reis (REIS, 2004) explica os conceitos referentes às linguagens de modelagem de processos. Dada à complexidade envolvida nos processos de software, a comunidade de engenharia de software sentiu a necessidade de desenvolver linguagens para a modelagem de processos de software, denominadas linguagens de modelagem de processos (*process modeling languages - PMLs*), com o objetivo de formalizar a definição dos processos, bem como possibilitar o suporte a todo o ciclo de vida do processo. Essas linguagens de processo são utilizadas a fim de facilitar a definição de processos, sua compreensão e garantir uma alta qualidade no que diz respeito à aplicação do Processo em uma Organização. Essas linguagens possibilitam a representação precisa e compreensível dos vários elementos envolvidos no processo (atividade, agente e artefato), ou seja, as PMLs são constituídas de elementos centrais, com semânticas próprias, que representam os elementos

comuns do processo de software, e um conjunto de associações necessárias a construção de um modelo de processo. Incluem a capacidade de gerenciar atividades que necessitam da interação humana para serem executadas. Esse trabalho apoiou no entendimento das linguagens de modelagem de processos de software e na escolha da linguagem de modelagem que iria se utilizada para documentar o processo de mudança de software.

No trabalho de Lachtermacher (LACHTERMACHER, 2008) mapeou-se o diagrama de atividades da UML em uma Rede de Petri. Neste trabalho os autores adotaram um padrão recente para a linguagem de transformação chamado QVT - Query View Transformation adotado pelo OMG (Object Management Group) e o objetivo dele foi fazer transformação entre modelos. O modelo fonte utilizado foi o diagrama de atividades apresentado na UML 2.0 e o modelo alvo foi a rede de Petri. Este trabalho serviu como base para o mapeamento do diagrama de atividades em Redes de Petri que realizou-se nesta dissertação. Para alcançar esse objetivo, objetivos intermediários foram definidos, como: o estudo do QVT, do diagrama de atividades, da Rede de Petri e as das ferramentas utilizadas para realizar a transformação, como por exemplo: *Eclipse Modeling Framework*, para fazer a geração dos metamodelos e do exemplo que será transformado e o MediniQVT para fazer a transformação em si, utilizando como entrada os dois metamodelos, o exemplo de Diagrama de Atividade e um script QVT desenvolvido.

Em Kerzner (KERZNER, 2006) pode-se ter mais uma visão de como as mudanças no software são geradas, os riscos que elas trazem para o projeto e a importância de um processo controlando a chegada e execução destas mudanças. Este trabalho (KERZNER, 2006) é comentado que os projetos ocorrem, inevitavelmente, em um ambiente de mudanças e, nem sempre, tais mudanças são geradas por decisões da equipe do projeto. O autor cita que grande parte dos casos de mudanças que ocorrem são por fatores externos como clientes que adicionam itens ao escopo ou diminuem o prazo, mudança das regulamentações e leis, obsolescência do produto, alterações gerenciais, avanços tecnológicos e mudanças no financiamento por exemplo.

O autor comenta que há, também, uma importante relação entre o gerenciamento de mudanças e os riscos do projeto, pois segundo Kerzner (KERZNER, 2006), se as mudanças não são gerenciadas, então mais tempo e mais dinheiro serão necessários para realizar o gerenciamento dos riscos, que se tornará com maior frequência um processo de gerenciamento de crises.

Para o autor não é possível evitar as mudanças, mas é possível gerenciá-las, através de um processo de gerenciamento de mudanças típico. Para Kerzner (KERZNER, 2006) o processo de gerenciamento de mudanças não pode ser tratado de forma isolada, ou apenas com o controle de mudanças feito em uma ou outra disciplina do projeto em separado. É necessário que haja um controle integrado, que avalie e identifique as mudanças de uma disciplina (escopo, por exemplo) e avalie as conseqüências nas demais disciplinas do projeto (custo e prazo, por exemplo).

O autor sugere que o gerenciamento integrado de mudanças definido pelo PMI/PMBOK é o processo necessário para controlar os fatores que criam mudanças, garantir que essas mudanças sejam benéficas, determinar se ocorreu uma mudança e gerenciar as ações para lidar com as mudanças. Esse processo deve ser realizado durante todo o projeto, desde a iniciação até o seu encerramento. Além disso, é um processo contido no grupo de processos de monitoramento e controle do projeto.

Em Andrade (Andrade, 2009), o autor tem por objetivo obter a integração dos modelos formais e semiformais, este trabalho propõe o mapeamento dos diagramas comportamentais da SysML em uma Rede de Petri Temporizada. As restrições de tempo e anotações energia são representadas pelo novo profile da UML MARTE (*Modeling and Analysis of Real-time and Embedded systems*). Além disso, uma metodologia de avaliação de desempenho das especificações de sistemas críticos é proposta, com o intuito de auxiliar o processo de modelagem e avaliação. Este trabalho serviu como base para o mapeamento do diagrama de atividades da UML em Redes de Petri.

O trabalho de Oliveira (Oliveira, 2006) tem como objetivo o desenvolvimento de um software de apoio ao processo de gerência de

solicitação de mudanças. O software possui funcionalidades aderentes ao processo de gerência de solicitação de mudanças da norma ISO/IEC 15504. O software promove a troca de informações entre os envolvidos no processo de mudança, permitindo que as solicitações sejam gerenciadas, acompanhadas e controladas até a sua conclusão.

Segundo o autor (Oliveira, 2006) a utilização de sistemas de solicitação de mudança é fundamental para a organização gerenciar as solicitações, garantindo que elas serão acompanhadas, controladas e bem documentadas.

A gerência de mudança é uma atividade importante que faz parte de todo processo de desenvolvimento de um sistema. A gerência das mudanças nos sistemas é um fator a ser considerado para a garantia da qualidade de um software. Para um controle efetivo, é necessário produzir uma documentação, registro de histórico das alterações que forneça a uma base de conhecimento para avaliação dos pontos críticos da empresa, e assim estruturar uma solução evitando problemas reincidentes (Oliveira, 2006).

Em relação aos objetivos definidos neste trabalho, Oliveira (Oliveira, 2006) conclui que os mesmos foram alcançados, pois as solicitações podem ser registradas, gerenciadas, acompanhadas. A análise de impacto é um dos itens que o sistema não contempla totalmente, mas existem funcionalidades do sistema que contribuem para que isso aconteça de forma a contribuir para uma análise das dependências e relacionamentos. Funcionalidades como o histórico das solicitações; relatório de solicitações por sistema, módulo e funcionalidade que permite saber quais solicitações estavam ou estão ligadas a uma determinada funcionalidade. O software facilita a troca de informações entre os envolvidos no processo, de forma automática e sempre notifica através de e-mail o solicitante quanto à situação da solicitação. Outro item importante é que a implementação produzida até o momento possui tecnologias amplamente utilizadas atualmente no mercado de software e de forma livre.

O software desenvolvido não atendeu idealmente o quesito análise de impacto, por ser uma atividade que depende muito da gerência de configuração, da necessidade dos itens de configuração. A análise de impacto

pode ser feita detalhando as solicitações até o nível de funcionalidades afetadas.

Este trabalho auxiliou no entendimento da importância de existir uma ferramenta que registre, controle e acompanhe o processo de mudança de software.

O trabalho de Barbaresco (BARBARESCO, 2000) explica o processo de gerência da configuração de software que é um conjunto de atividades definidas para administrar mudanças no ciclo de vida de um software. Um software de apoio a este processo foi desenvolvido a partir do estudo comparativo entre o processo de gerência da configuração existente nas normas de qualidade ISO/IEC 12207, ISO 9000-3, ISO/IEC 15504(SPICE) e no modelo CMM/SEI. O software construído permite o cadastro, controle e consultas sobre os itens de configuração e suas versões produzidas ao longo de um projeto de software, bem como permite a geração de relatórios gerenciais.

Segundo o autor (BARBARESCO, 2000) de forma geral, as normas que foram adotadas para estudo no desenvolvimento deste trabalho, mostraram-se igualmente satisfatórias. Contudo, de todas as normas utilizadas, a norma ISO/IEC 12207 e o modelo CMM/SEI são os que melhor detalham o processo de gerência de configuração, sendo que este último inclusive propõe metas subdividindo o processo, apontando recomendações, verificações e também apresentando alguns conceitos importantes. As outras duas normas a ISO 9000-3 e ISO/IEC 15504/SPICE não apresentam muitos detalhes, sobre o processo estudado.

Este trabalho auxiliou no entendimento da importância de existir uma ferramenta que registre, controle e acompanhe o processo de mudança de software.

Em Araújo (ARAUJO, 2009), o autor tem por objetivo propor um modelo em redes de Petri estocástica para avaliar o impacto da disponibilidade e da confiabilidade dos recursos computacionais. Além disso, propõe uma

metodologia de avaliação de desempenho, com o intuito de auxiliar os processos de modelagem e de avaliação.

O trabalho de Araújo (ARAUJO, 2009) apresenta uma abordagem de sistemas de transferência eletrônica de fundos baseada em *Generalized Stochastic Petri nets* (GSPN), com o objetivo de realizar avaliações de desempenho nos recursos computacionais. Além disso, uma metodologia foi apresentada com o intuito de auxiliar no processo de avaliação de desempenho do sistema TEF. Essa metodologia é composta por uma série de passos que envolvem desde o entendimento do ambiente, modelagem, até a geração e análise do modelo GSPN. Com a utilização das GSPNs como ferramenta de modelagem e análise, é possível aferir métricas de maneira probabilística. Por exemplo, pode-se encontrar a probabilidade de um recurso computacional estar processando transações, indicando assim o seu nível de utilização.

Este trabalho auxiliou na criação do modelo em redes de petri estocástica também utilizada nesta dissertação, no entendimento dos conceitos sobre redes de petri estocástica e na definição da metodologia de avaliação de desempenho.

De acordo com Oliveira (Oliveira, 2006) a gestão de processos de negócio baseada em workflow vem se tornando um importante recurso administrativo no contexto de médias e grandes empresas. Esta abordagem consiste na automatização dos processos empresariais através da implantação de sistemas de informação especializados, que têm por objetivo otimizar o fluxo de informações dentro da organização. A peça-chave nestes sistemas é o desenho do processo da empresa que se quer automatizar, chamado de workflow. Um workflow descreve todas as atividades que são executadas no processo, define seus participantes (os responsáveis por cada atividade), os dados que são transferidos entre eles e a ordem em que tais atividades devem ocorrer.

O trabalho de Oliveira (Oliveira, 2006) tem foco na melhoria de workflow. Este tema aborda a busca por eficiência, produtividade e qualidade em processos de negócio. Esta tarefa deve ser realizada de forma objetiva, sistematizada e alinhada às metas estratégicas da organização.

Segundo o autor (Oliveira, 2006) um grande número de pesquisas propõem o uso de redes de Petri como método de análise qualitativa e quantitativa de workflow, com o fim de oferecer apoio a projetos de melhoria. Este formalismo é amplamente usado para verificação e análise de desempenho de sistemas variados e apresenta características que fazem a sua aplicação em workflow bastante natural. Apesar disso, verificou-se que o uso de redes de Petri neste contexto ainda apresenta limitações e não está devidamente alinhado às preocupações enfrentadas na gestão de processos de negócio. Poucos modelos abordam características realistas do workflow e muitos deles não tiram proveito do potencial das redes de Petri.

Neste trabalho, o autor (Oliveira, 2006) apresentada uma metodologia para a aplicação de redes de petri estocásticas generalizadas (GSPN) em projetos de melhoria de workflow. Os conceitos de workflow são mapeados em estruturas modeladas em GSPN, fornecendo uma representação formal para workflow que pode ser utilizada para a realização de análises qualitativas e de desempenho.

Diversas características encontradas em processos reais são consideradas neste modelo, tais como disputa por recursos e controle de fluxo complexo entre atividades (Oliveira, 2006).

De acordo com o autor (Oliveira, 2006) estas características não estão presentes em outros modelos relacionados. Além disso, a metodologia apresentada alinha este método aos conceitos e necessidades encontrados na gestão de processos, permitindo uma integração com o arcabouço conceitual desta área. A metodologia é utilizada na realização de dois estudos de caso que demonstram sua relevância do ponto de vista prático.

Este trabalho auxiliou na criação do modelo em redes de petri estocástica também utilizada, na modelagem de fluxos de trabalhos e processos, no entendimento dos conceitos sobre redes de petri estocástica e na definição da metodologia de avaliação de desempenho.

2.2 Considerações Finais

Este capítulo apresentou os trabalhos que auxiliaram no entendimento e estão relacionados a este trabalho. Em resumo, os trabalhos citados acima puderam complementar o conteúdo deste trabalho, buscando esclarecer o entendimento de assuntos como as mudanças são geradas e a importância de controlá-las através de um processo de mudança, linguagens de modelagem de processo, mapeamento do diagrama de atividades em uma redes de Petri, a proposta de um modelo de rede de Petri estocástica e duma metodologia de avaliação de desempenho. A grande diferença, é que neste trabalho, será proposto um modelo estocástico para avaliação de desempenho ao suporte do planejamento de um processo de mudança de software, utilizando o mapeamento do diagrama de atividades da UML em redes de Petri estocástica e a definição de uma metodologia que possa apoiar esta avaliação.

CAPITULO 3 - FUNDAMENTOS

Este capítulo apresenta os principais conceitos da dissertação. Primeiramente, conceitos sobre Engenharia de Software são introduzidos. Em seguida, são abordados os conceitos de processo de software e modelagem de processo através do diagrama de atividades da UML – *Unified Modeling Language*. Em seguida processo de mudança de software é introduzido, destacando seus conceitos, benefícios e execução e planejamento. Por fim, é apresentado uma visão geral sobre redes de Petri (RdP), onde exemplos de modelagens e refinamento das RdP são abordados. Além disso, também é apresentada as redes de Petri estocástica (*Stocasthic Petri net - SPN*), uma extensão das RdP, adotadas nesta dissertação e as razões para a escolha das SPN.

3.1 Processo de software

O termo Engenharia de Software foi criado na década de 1960 e utilizado oficialmente em 1968 na Conferência sobre Engenharia de Software da OTAN numa tentativa de contornar a crise do software e dar um tratamento de engenharia (mais sistemático e controlado) ao desenvolvimento de sistemas de software (Buxton, et al., 1969).

Vários autores já desenvolveram suas definições de engenharia de software, as quais são apresentadas a seguir. A definição proposta por Pressman (PRESSMAN, 2000), estabelece que engenharia de software é a aplicação de uma abordagem prática, disciplinável, e quantificável para o desenvolvimento, operação e manutenção de software. Na visão de Pfleeger (PFLEEGER, 2004), engenharia de software “é um processo tanto criativo como sistemático, geralmente, envolvendo muitas pessoas e produzindo diferentes tipos de produtos”. Já Sommerville (SOMMERVILLE, 2005), acredita que a engenharia de software dedica-se a processos técnicos de desenvolvimento de software, atividades de gerenciamento de

projeto de software, desenvolvimento de ferramentas, métodos e teorias que apóiam a produção de software.

O fundamento da engenharia de software é a camada de processo. O processo de engenharia de software mantém unidas as camadas de ferramentas, métodos e processos (PRESSMAN, 2000).

Muitas empresas querem organizar-se por processos, mas não têm uma noção clara dos passos a seguir e das providências que devem ser tomadas. Outras não estão certas da decisão a tomar a respeito da sua estruturação por processos e podem beneficiar-se de um raciocínio que as ajudem a decidir. Existem também as empresas que não sabem ao certo o que significa serem organizadas por processos e as que não têm certeza se a sua forma organizacional atual é adequada para a gestão dos processos (Dias, et al., 2008).

Um processo é definido (RIGO, 2008; Booch, et al., 1999), como sendo um conjunto seqüencial de passos a serem seguidos constituídos por atividades, métodos, práticas e transformações, utilizados para atingir uma meta, a qual geralmente está associada a um ou mais resultados concretos finais, que são os produtos da execução do processo. Um processo é caracterizado por conter documentação detalhada do que é feito (produto), quando (passos), por quem (agentes), o que usa (insumo) e o que produz (resultado).

Na concepção mais freqüente, processo é qualquer atividade ou conjunto de atividades que toma um *input*, adiciona valor a ela e fornece um *output* a um cliente específico. Os processos utilizam os recursos da organização para oferecer resultados objetivos aos seus clientes (Dias, et al., 2008).

Essa idéia de processo como um fluxo de trabalho – com *inputs* e *outputs* claramente definidos e tarefas discretas que seguem uma seqüência e que dependem uma das outras numa sucessão clara – vem da tradição da engenharia. Os *inputs* podem ser materiais – equipamentos e

outros bens tangíveis, mas também podem ser informações e conhecimento (Dias, et al., 2008).

O objetivo da modelagem do processo é garantir o entendimento da estrutura e a dinâmica da organização, objetivando a compreensão entre todos os envolvidos, bem como: cliente, usuários finais e desenvolvedores, para que todos tenham uma visão comum (DIAS, 2008).

Os profissionais que são responsáveis pela modelagem de processos de negócios podem contar com várias técnicas, ferramentas, processos e métodos utilizados para a modelagem dos processos (DIAS, 2008).

Dada à complexidade envolvida nos processos de software a comunidade de Engenharia de Software sentiu a necessidade de desenvolver linguagens para a modelagem de processos de software, denominadas Linguagens de Modelagem de Processos (*Process Modeling Languages – PMLs*), com o objetivo de formalizar a definição dos processos, bem como possibilitar o suporte a todo o ciclo de vida do processo (ABDALA, 2004).

Essas Linguagens de processo são utilizadas a fim de facilitar a definição de processos, sua compreensão e garantir uma alta qualidade no que diz respeito à aplicação do Processo em uma Organização, em um determinado Projeto.

As PMLs oferecem recursos para descrever e manipular os passos do processo. Elas constituem uma categoria específica de linguagens de especificação e programação de computadores voltada à definição e automação do processo de software. Assim sendo, as PMLs possuem algumas características próprias, que as distanciam das linguagens de programação de propósito geral, incluindo, por exemplo, a capacidade de gerenciar atividades que necessitam da interação humana para serem executadas (REIS, 2004).

Muitas linguagens apresentadas na literatura podem ser classificadas simultaneamente como de especificação de projeto e de

implementação de processo, como por exemplo, Merlin, Marvel, SPELL, SLANG e SPEM (REIS, 2004).

A UML trata-se de uma linguagem com uma especificação semântica semiformal, que inclui sintaxe abstrata, regras bem definidas e semântica dinâmica (Dias, et al., 2008).

O RUP (RUP; Booch, et al., 2000) usa UML, uma notação consistente que pode ser aplicada à engenharia de sistemas e à engenharia de negócios para modelar seus fluxos de trabalho.

A modelagem visual é o uso de notações de design gráficas e textuais, semanticamente ricas, para capturar designs de software. Uma notação, como a UML, permite que o nível de abstração seja aumentado, enquanto mantém sintaxe e semântica rígidas.

Este trabalho introduz o diagrama de atividades da UML - *Unified Modeling Language* (Booch, et al., 2000; Guedes, 2004; Fowler, et al., 2000), como um diagrama de modelagem de processos pela sua simplicidade de utilização e representação do mundo real.

O Diagrama de atividades da UML é utilizado para descrever lógica de programação, processos de negócio e *workflows*. Para um processo, este diagrama determina as regras essenciais de seqüência que se deve seguir para a execução, mostrando o fluxo de uma atividade para a outra (OMG, 2005; Booch, et al., 2000).

O diagrama de atividades (Booch, et al., 2000; Guedes, 2004; Fowler, et al., 2000) é formado por alguns componentes tais como: estado inicial e estado final que determinam o início e o encerramento do fluxo de controle do diagrama. Deve haver sempre um estado inicial e podem existir vários estados finais. O componente de decisão é representado por um losango e indica a possibilidade de escolha entre os fluxos disponíveis. Tem um ponto de entrada e 3 pontos de saída. Para a definição das condições, pode-se usar texto livre ou pseudo-código e a condição é expressa na g*uard condition* (condição de guarda). O componente intercalação, também conhecida como merge, é como a decisão,

representada por um losango. Destina-se a marcação do término de um comportamento condicional iniciado por um desvio (Possui várias entradas e uma única saída). A ação é uma tarefa a ser executada dentro de uma atividade. Pode ser do tipo (*entry* – executada imediatamente ao entrar na atividade, *exit* – executada imediatamente antes de sair da atividade, *do* – executada durante a permanência na atividade e evento – executada quando um evento específico acontece. Exige a definição do evento, argumentos e condição. O componente transição é indicado pela seta e representa o relacionamento entre duas atividades e são chamadas de transição não-ativadas, por não representarem um espaço de tempo. As barras de sincronização indicam a execução de fluxos concorrentes ou paralelos e são representadas por barras verticais ou horizontais que mostram a separação ou a união do fluxo. As barras de bifurcação têm um fluxo de entrada e dois ou mais de saída. E as barras de união tem dois ou mais fluxos de entrada e um de saída.

A Figura 1 abaixo ilustra o diagrama de atividades (Booch, et al., 2000).

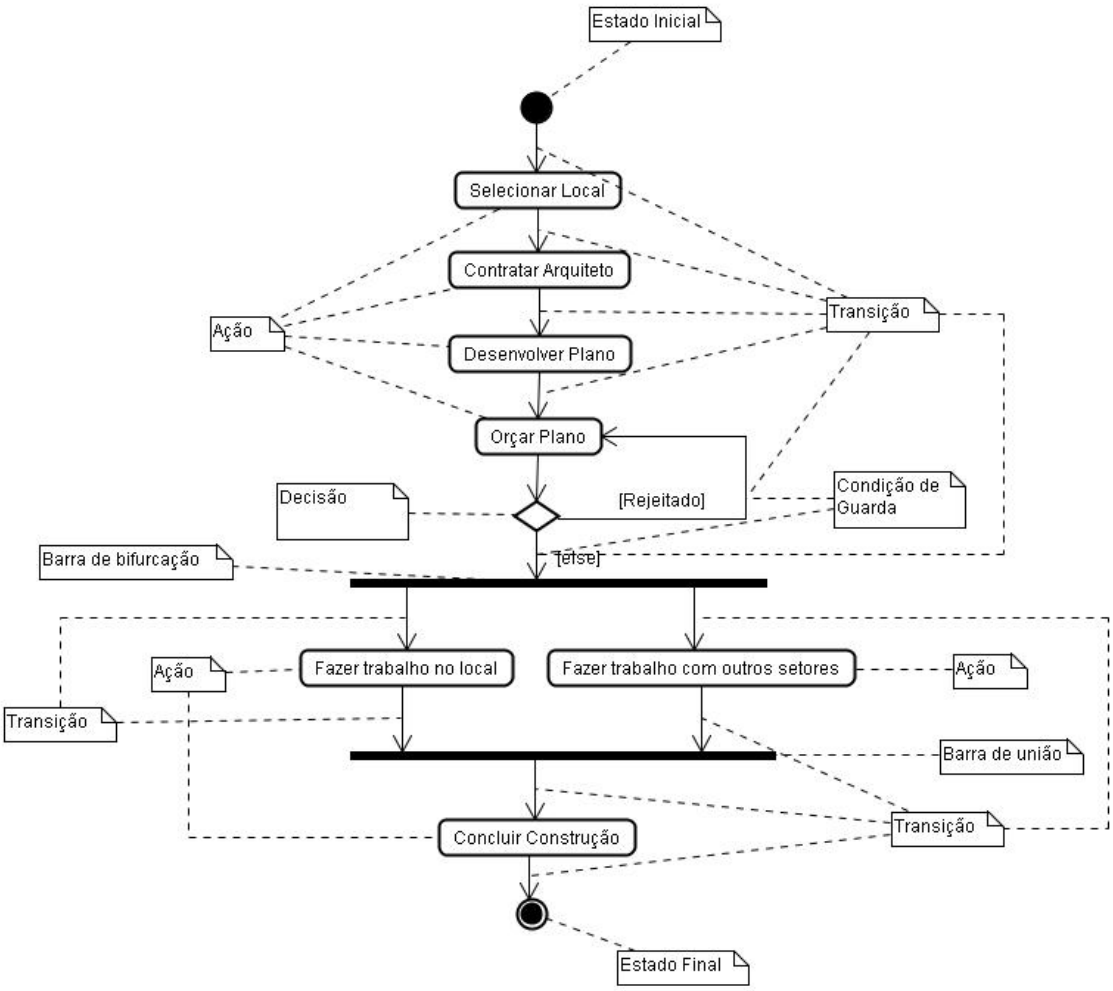


Figura 1: Diagrama de Atividades

3.2 Processo de Mudança de Software

Quando o software é colocado em uso, novos requisitos surgem, e os requisitos existentes são modificados. Os softwares podem precisar ser modificados para corrigir os erros encontrados, melhorar seu desempenho e outras características não funcionais, evoluindo em respostas as exigências de mudanças (SOMMERVILLE, 2005).

A história da gerência de configuração de software surge em meados da década de 1970, quando os microprocessadores se tornaram populares e o software deixou de ser considerado parte integrante do hardware para se tornar um produto independente. Nesta época, os sistemas se tornaram cada vez maiores e sofisticados, ficando claro que seriam necessárias metodologias próprias para controlar o desenvolvimento desses sistemas (Wikipédia).

A gerência de mudanças é uma parte geralmente negligenciada da gerência de configuração. Como ela não tem resultados imediatos para os desenvolvedores e engenheiros de software envolvidos no projeto, estes acabam por não perceber sua importância. Gerência de mudanças entretanto é uma parte importante da gerência de configuração, pois é a atividade que permite se saber o motivo de uma configuração ter sido mudada para outra configuração. Esta atividade também pode ser parcialmente automatizada, e diversos Sistemas de controle de versão já são integrados com sistemas de gerência de mudanças. A gerência de mudanças tem por objetivo mapear, para cada mudança efetuada no sistema, qual foi o motivo que gerou esta mudança (Wikipédia). Inclui o processo de tomada de decisão sobre quais mudanças realmente devem ser realizadas e o processo para executá-las. Envolve outras disciplinas do desenvolvimento de sistemas, como: gerência de requisitos, testes, gerenciamento da liberação de software, suporte ao usuário e gerência de projeto (Schneider, 2001).

É comum ter em sistemas de software, arquivos que listam as melhorias e mudanças ocorridas entre duas versões. Estes arquivos são resultado da gerência de mudanças, identificando o que mudou entre uma versão e outra (Wikipédia).

O registro de uma solicitação de mudança deve incluir informações sobre a origem e o impacto do problema, a solução proposta, e o seu custo e efeito sobre os prazos do projeto. O processo de solução depende de uma série de fatores. As solicitações são tratadas, em geral, de acordo com duas principais categorias: solicitações de melhoria e correção de defeitos (Schneider, 2001).

Uma solicitação de melhoria especifica uma nova característica ou uma mudança no comportamento projetado do sistema. Defeitos são anomalias ou falhas em um produto entregue. Enquanto grande parte dos dados mantidos para o controle de melhorias ou de defeitos é similar, estes dois tipos de solicitações são tratados de forma bem diferente no processo de controle de mudanças (Schneider, 2001).

Segundo Schneider (Schneider, 2001) o processo de gerenciamento das solicitações de mudanças pode variar bastante com a abrangência do problema a ser tratado. Uma mudança significativa na especificação do sistema pode representar até a interrupção ou redefinição por completo do projeto. Para mudanças “rotineiras”, no entanto, pode ser definido um modelo de transição de estados das solicitações de mudanças, definindo eventos e ações correspondentes a cada evento. As modificações podem ser simples, destinadas a corrigir erros de códigos, mais extensas a fim de corrigir erros de projetos, ou significativas, com a finalidade de corrigir erros de especificação ou acomodação de novos requisitos.

Segundo Sommerville (Sommerville, 2005) existem três diferentes tipos de manutenção de software. A manutenção corretiva, para reparar os defeitos no software. A manutenção adaptativa, para adaptar o software a um ambiente operacional diferente ou o software a novos requisitos. E a manutenção evolutiva, para fazer acréscimos à funcionalidade do sistema ou modificá-la.

Os processos de manutenção variam dependendo do tipo de software que está em manutenção, dos processos de desenvolvimento utilizados em uma organização e do pessoal envolvido no processo. Em algumas organizações, a manutenção pode ser um processo informal. A maioria dos pedidos de manutenção surge de conversações entre os usuários e os desenvolvedores do sistema. Em outras organizações, esse é um processo formalizado, com documentação estruturada produzida a cada estágio do processo (SOMMERVILLE, 2005).

Segundo o RUP (RUP) a finalidade de ter um processo padrão de controle de mudanças documentado é assegurar que as mudanças feitas em um projeto sejam consistentes e que os envolvidos adequados sejam informados do estado do produto, das mudanças feitas nele e do impacto de custo e programação gerado por essas mudanças.

A Figura 2 define um processo de mudança de software utilizado na realização deste trabalho. Este processo foi baseado no processo de mudança de software de uma organização de desenvolvimento de software

da cidade do Recife/ Pernambuco e utiliza do diagrama de atividades da UML para sua modelagem.

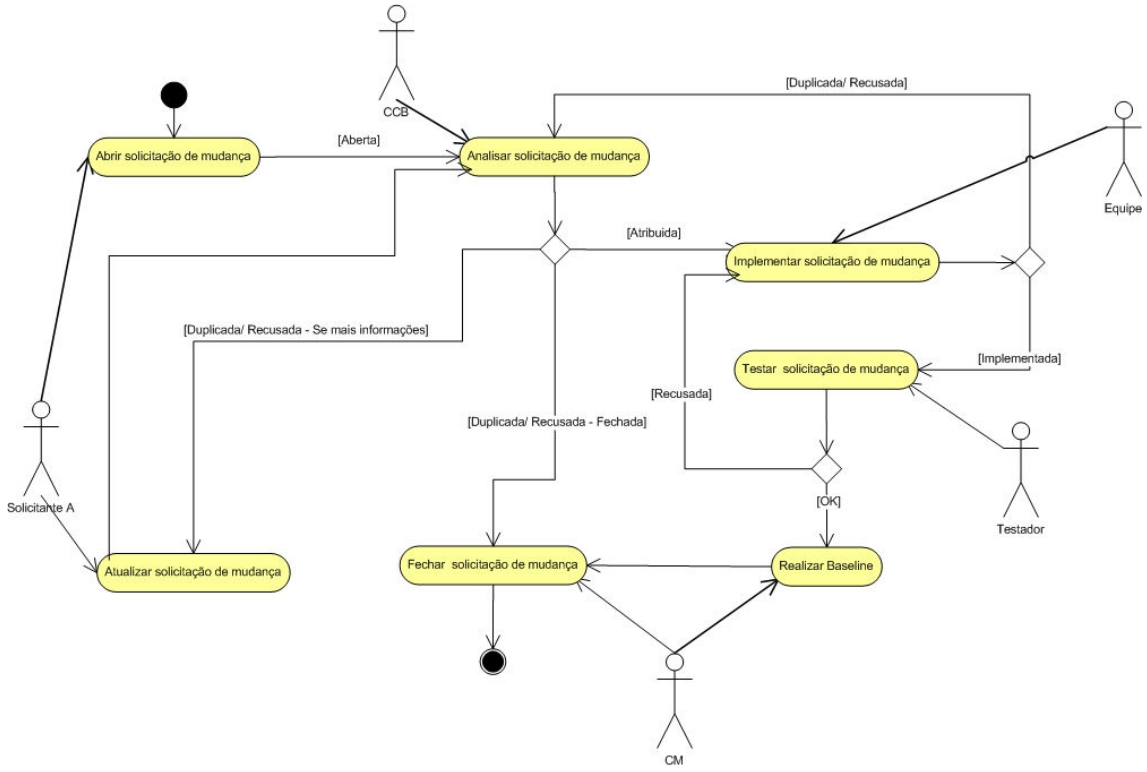


Figura 2: Processo de Mudança de Software

O processo da Figura 2 é iniciado por um conjunto de pedidos de mudança por parte de um dos envolvidos no sistema. O custo e o impacto dessas mudanças são calculados para ver quanto do sistema é afetado pela mudança e quanto pode custar para implementar essa mudança. As mudanças são implementadas e testadas e uma nova versão estável do sistema é liberada (SWEBOK, 2004).

As mudanças nos artefatos de desenvolvimento são propostas através de solicitações de mudança (RUP). A solicitação de mudança é um artefato formalmente submetido que é usado para rastrear, através da ferramenta de controle de mudança, todas as solicitações dos envolvidos (inclusive novas características, solicitações de melhoria, conserto de defeitos, mudança de requisitos etc). Todo o histórico de mudanças será mantido com a solicitação de mudança, o que inclui todas as mudanças de estado, datas e motivos para as mudanças (RUP).

Segundo o RUP (RUP) as solicitações de mudança são usadas para documentar e controlar defeitos, solicitações de melhorias e qualquer outro tipo de solicitação de mudança no produto. A vantagem das solicitações de mudanças é que elas fornecem um registro das decisões e, devido a seu processo de avaliação, garantem que os impactos das mudanças sejam entendidos no projeto através das notas e comentários dos envolvidos na execução da mudança. A Figura 3, Figura 4 e Figura 5 ilustram uma solicitação de mudança preenchida.

Mantis - Projeto 1

Viewing Issue Advanced Details

ID:	Categories/Module:	Severity:	Reproducibility:	Date Submitted:	Last Update:
74572	Bug - Interno	major	always	17-Oct-2010 13:37	17-Oct-2010 13:39

Reporter: Marcely Santos
Assigned To: Marcely Santos
Priority: high
Status: ASSIGNED FOR DEVELOPMENT
Baselined in:
Projection: none
ETA: none
Horas Estimadas para Resolucao:
Origem na Fábrica de Testes: NAO
Tipo de Artefato Envolvido: Codigo
Use Case: 057

Platform:
OS:
OS Version:
Found in: CLIENTE-PROJETO1-SNAPSHOT-20101015-1810
Resolution: open
Released in:

Summary: [CR74572]: Relatório de Custódia de Valores.
Description: O código implementado deve ser ajustado da seguinte forma:

1 - Na tela de Parâmetros, o sistema deve carregar o campo Nº Contrato Microsiga apenas com os Nº vinculados a serviços de Custódia;

2 - Em todos os Relatórios:
- Não está calculando o Repasse ISS
- Caso a query não encontre dados a exibir, o sistema deve exibir a msg de erro MG_Erro261;

3 - No relatório Analítico Geral, os percentuais contratados do cabeçalho estão sendo exibidos multiplicados por 100;

4 - No relatório Sintético Geral:
- O registro de Guarda de Tesouraria deve ser o primeiro a ser impresso (antes dos Pontos de Atendimento);
- O valor da custódia da Guarda de Tesouraria foi calculado errado.

5 - No relatório Analítico Itaú:
- O valor da custódia da Guarda de Tesouraria foi calculado errado;
- No nome do Ponto de Atendimento, quando o sistema deveria exibir 'Guarda Tesouraria', mas está ficando em branco;
- Está quebrando a página indevidamente para exibir o Total Geral. Caso seja necessária a quebra, o cabeçalho deve ser impresso completo;

6 - No relatório Sintético Itaú Clientes:
- O registro de Guarda de Tesouraria deve ser o primeiro a ser impresso (antes dos Pontos de Atendimento);
- O valor da custódia da Guarda de Tesouraria foi calculado errado.

7 - No relatório Sintético Agências:
- Colocar no cabeçalho os percentuais contratados para o Grupo Serviço selecionado ao invés do percentuais de um dos Pontos de Atendimento selecionado;
- O valor da Guarda de Tesouraria não está sendo considerado na totalização do dia.

Steps To Reproduce: Home » Faturamento » >> Relatórios >> Custódia de Valores
CCB Comments: PARTICIPANTES DO CCB DE 2010.06.10

- Marcely Santos
- João da Silva

Figura 3: Exemplo de solicitação de mudança preenchida – (a)

CCB Comments: PARTICIPANTES DO CCB DE 2010.06.10

- Marcely Santos
- João da Silva
- Maria dos Santos

ANÁLISE/IMPACTOS

Escopo: namespaces: Cliente.Projeto1.Web.TransporteValores
Cliente.Projeto1.WebApp.TransporteValores
Cliente.Projeto1.Negocio.TransporteValores

Prazo: 2h.

Custo: Será calculado a partir da estimativa.

COMENTÁRIOS EXTRAS

Não se aplica

Relationships

Sel	Relationship Type	ID	Status	Severity	Categories	Assigned To	Summary	Found in	CCB	Checklist	Branches

Attached Files:

Issue History

Date Modified	Username	Field	Change
17-Oct-2010 13:37	Marcely Santos	New Issue	
17-Oct-2010 13:37	Marcely Santos	Assigned To	=> Marcely Santos
17-Oct-2010 13:37	Marcely Santos	Origem na Fábrica de Testes	=> NAO
17-Oct-2010 13:37	Marcely Santos	Tipo de Artefato Envolvido	=> Codigo
17-Oct-2010 13:37	Marcely Santos	Use Case	=> 057
17-Oct-2010 13:37	Marcely Santos	Status	NEW => ASSIGNED FOR DEVELOPMENT
17-Oct-2010 13:39	Marcely Santos	Note Added: 0179609	
17-Oct-2010 13:39	Marcely Santos	Note Added: 0179610	
17-Oct-2010 13:39	Marcely Santos	Note Added: 0179611	

Notes

(0179609)
Marcely Santos
17-Oct-2010 13:39

Ao retestar a funcionalidade, os seguintes problemas foram detectados:

1) O cálculo do repasse ISS está incorreto. Está sendo exibida a alíquota cadastrada para o município do Ponto de Atendimento (10,0).

Exemplo do cálculo do repasse do ISS (com aplicação do fator embutido).

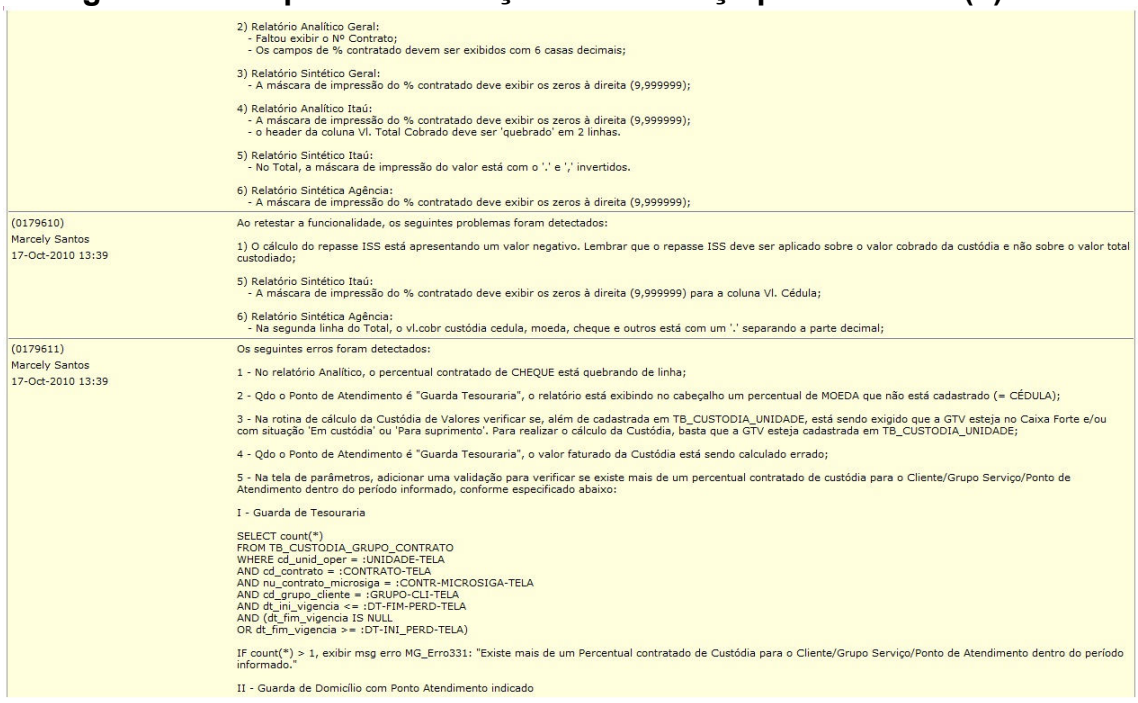
Para o valor de 5% cadastrado para o município: (1,00 - 0,05) = 0,95

Valor calculado da Custódia R$ 1000,00

1000,00 / 0,95 = R$ 1.052,63 - R$ 1.000,00 = repasse ISS R$ 52,63

=> R$ 1.052,63 * 5% = R$ 52,63

Figura 4: Exemplo de solicitação de mudança preenchida – (b)



2) Relatório Analítico Geral:
- Faltou exibir o Nº Contrato;
- Os campos de % contratado devem ser exibidos com 6 casas decimais;

3) Relatório Sintético Geral:
- A máscara de impressão do % contratado deve exibir os zeros à direita (9,999999);

4) Relatório Analítico Itaú:
- A máscara de impressão do % contratado deve exibir os zeros à direita (9,999999);
- o header da coluna Vl. Total Cobrado deve ser 'quebrado' em 2 linhas.

5) Relatório Sintético Itaú:
- No Total, a máscara de impressão do valor está com o '.' e ',' invertidos.

6) Relatório Sintética Agência:
- A máscara de impressão do % contratado deve exibir os zeros à direita (9,999999);

(0179610)
Marcely Santos
17-Oct-2010 13:39

Ao retestar a funcionalidade, os seguintes problemas foram detectados:

1) O cálculo do repasse ISS está apresentando um valor negativo. Lembrar que o repasse ISS deve ser aplicado sobre o valor cobrado da custódia e não sobre o valor total custodiado;

5) Relatório Sintético Itaú:
- A máscara de impressão do % contratado deve exibir os zeros à direita (9,999999) para a coluna Vl. Cédula;

6) Relatório Sintética Agência:
- Na segunda linha do Total, o vl.cobr custódia cedula, moeda, cheque e outros está com um '.' separando a parte decimal;

(0179611)
Marcely Santos
17-Oct-2010 13:39

Os seguintes erros foram detectados:

1 - No relatório Analítico, o percentual contratado de CHEQUE está quebrando de linha;

2 - Qdo o Ponto de Atendimento é "Guarda Tesouraria", o relatório está exibindo no cabeçalho um percentual de MOEDA que não está cadastrado (= CÉDULA);

3 - Na rotina de cálculo da Custódia de Valores verificar se, além de cadastrada em TB_CUSTODIA_UNIDADE, está sendo exigido que a GTV esteja no Caixa Forte e/ou com situação 'Em custódia' ou 'Para suprimento'. Para realizar o cálculo da Custódia, basta que a GTV esteja cadastrada em TB_CUSTODIA_UNIDADE;

4 - Qdo o Ponto de Atendimento é "Guarda Tesouraria", o valor faturado da Custódia está sendo calculado errado;

5 - Na tela de parâmetros, adicionar uma validação para verificar se existe mais de um percentual contratado de custódia para o Cliente/Grupo Serviço/Ponto de Atendimento dentro do período informado, conforme especificado abaixo:

I - Guarda de Tesouraria

```
SELECT count(*)
FROM TB_CUSTODIA_GRUPO_CONTRATO
WHERE cd_unid_oper = :UNIDADE-TELA
AND cd_contrato = :CONTRATO-TELA
AND nu_contrato_microsiga = :CONTR-MICROSIGA-TELA
AND cd_grupo_cliente = :GRUPO-CLI-TELA
AND dt_ini_vigencia <= :DT-FIM-PERD-TELA
AND (dt_fim_vigencia IS NULL
OR dt_fim_vigencia >= :DT-INI_PERD-TELA)
```

IF count(*) > 1, exibir msg erro MG_Erro331: "Existe mais de um Percentual contratado de Custódia para o Cliente/Grupo Serviço/Ponto de Atendimento dentro do período informado."

II - Guarda de Domicílio com Ponto Atendimento indicado

Figura 5: Exemplo de solicitação de mudança preenchida – (c)

As práticas de gerenciamento de mudanças normalmente são institucionalizadas ou estabelecidas no início do ciclo de vida do projeto. Desse modo, as solicitações de mudança, que integram o processo de

mudança, podem surgir a qualquer momento durante o curso do projeto (RUP).

Segundo o RUP (RUP) a principal origem de mudanças (defeitos, melhoria e mudanças) consiste nos resultados da execução dos testes — integração, sistema e desempenho. Contudo, os defeitos podem aparecer em qualquer ponto do ciclo de vida de desenvolvimento do software e abranger a identificação de documentação, casos de teste ou casos de uso ausentes ou incompletos.

Qualquer pessoa da equipe de projeto ou o cliente deve ser capaz de iniciar uma solicitação de mudança. No entanto, a mesma precisa ser analisada. A arbitragem sobre uma solicitação de mudança é realizada por uma equipe de revisão ou um comitê de controle de mudança (*change control board* - CCB). Esse comitê que supervisiona o processo de mudanças consiste em representantes de todas as partes interessadas, inclusive clientes, desenvolvedores e usuários (RUP).

De acordo com o RUP (RUP), após aberto a mudança, é feita a análise da mesma pelo CCB. A finalidade desta atividade é determinar se a solicitação de mudança deve ser aceita, duplicada ou recusada. No caso de solicitações de mudanças aceitas, esta atividade avalia a prioridade, o esforço, a programação e outros aspectos como o custo de execução da mudança, a gravidade e complexidade, a fim de determinar se a mudança será implementada.

Se houver suspeita de uma solicitação de mudança estar duplicada (por exemplo, se a solicitação de mudança já foi registrada) ou ter sido abortada (por exemplo, a solicitação de mudança não conseguiu ser reproduzida pelo responsável da sua execução, mudança não pertinente) pode haver a necessidade de mais informações para a avaliação. O CCB atribui a solicitação de mudança ao membro da equipe adequado - de acordo com o tipo de solicitação (por exemplo, solicitação de melhoria, conserto de defeito, mudança de documentação, defeito de teste etc) - e faz as atualizações necessárias na programação do projeto (RUP).

O membro da equipe designado executará a implementação das mudanças solicitadas. Depois de implementada pelo membro da equipe atribuído (analista, desenvolvedor, testador, redator técnico), as mudanças são testadas. Após os testes as mudanças entram em uma versão estável do sistema gerada pelo gerente de configuração (c*onfiguration management* – CM) e a solicitação é fechada (RUP).

3.2.1 Problemas da Mudança do Software

Grande parte dos problemas associados às mudanças de software pode estar relacionado à maneira pela qual o software foi planejado e desenvolvido. A falta de disciplina nas atividades de desenvolvimento da engenharia de software, quase sempre traduzem-se em problemas na mudança do software.

Segundo Pressman (PRESSMAN, 2000), outros problemas são associados à manutenção de software, como:

- a falta de documentação das mudanças e de todo os outros artefatos, dificultando assim, o rastreamento do software em suas versões e lançamentos;
- a elevada mobilidade entre o pessoal da área de software. Muitas vezes não podendo contar com uma explicação pessoal do desenvolvimento quando a manutenção for necessária;
- a maioria dos softwares não são projetados para sofrer mudanças, a menos que um método de projeto acomode mudanças mediante conceitos como, independência funcional ou classes de objetos.

A modificação no desenvolvimento de software é imprescindível, com objetivo do resultado final ficar próximo ao desejado, minimizando problemas futuros. Após o sistema estar em funcionamento, dificulta ainda mais as realizações de modificações que vierem ser necessárias sem a geração de inconveniência para os usuários.

Na visão de Pfleeger (PFLEEGER, 2004), “a manutenção de software é difícil. Como o software está em funcionamento, a equipe de manutenção equilibra a necessidade de modificação com a necessidade de

mantê-lo acessível aos usuários". Para um maior entendimento do que o autor expressa, classificou os problemas de manutenção em dois tipos: problemas com o pessoal e problemas técnicos. Os problemas com o pessoal, estão relacionado com aspectos pessoais e organizacionais. É essencial a interação do pessoal com o problema e a solução, a fim de corrigir e ajustar o software. Os problemas com o pessoal subdividem em:

- entendimento limitado: além de equilibrar as necessidades dos usuários com as relativas de software e hardware, a equipe de manutenção trata das limitações do entendimento humano. Existe um limite para o potencial de cada pessoa quanto a estudar a documentação e a obter material relevante para o problema a ser resolvido. A distração no ambiente de trabalho limita a produtividade. A falta de habilidade e o entendimento incorreto do usuário em relação ao funcionamento do sistema também geram problemas;
- prioridade de gerenciamento: são comparados os desejos dos clientes com as necessidades do sistema. Algumas vezes, os gerentes consideram mais importantes à manutenção e o aprimoramento do que a construção de novas aplicações. Enquanto eles estimulam os responsáveis pela manutenção a reparar um sistema antigo, os usuários pedem novas funções ou um novo sistema. No mesmo modo que a pressa em disponibilizar um produto, pode levar os desenvolvedores e mantedores a implementar uma modificação rápida, ineficiente e sem ter sido adequadamente testada, em vez de utilizar o tempo necessário para as boas práticas de engenharia de software. O resultado é um produto difícil de ser entendido e alterado;
- ânimo: uma das principais razões para o pouco ânimo é o status de trabalho "de segunda classe", freqüentemente relacionado com a equipe de manutenção. Algumas vezes, os programadores pensam que é necessário ter mais habilidades para projetar e desenvolver um sistema do que para mantê-lo em funcionamento. Porém, os programadores que trabalham na manutenção lidam com problemas que os desenvolvedores nunca observarão. Além da habilidade para

> programar, os responsáveis pela manutenção devem ter a capacidade de trabalhar com os usuários, a fim de antecipar mudanças e investigá-las. Alguns grupos fazem “rodízio” de programadores entre diversos projetos de desenvolvimento e manutenção para dar a eles a oportunidade de realizar um pouco de cada atividade. Esse “rodízio” ajuda a evitar o estigma da manutenção. Contudo, freqüentemente
>
> os programadores precisam trabalhar em vários projetos ao mesmo tempo. A demanda do tempo de um programador resulta em um conflito de prioridades.

Os problemas técnicos também interferem na produtividade da manutenção. Muitas vezes, eles são conseqüência do que desenvolvedores fizeram anteriormente. Ou então, resultam de paradigmas ou processos específicos, adotados na implementação. Os problemas técnicos subdividem em, segundo (RIGO, 2008; PFLEEGER, 2004):

- recursos e paradigmas: se a lógica do projeto não for óbvia, a equipe poderá ter dificuldades em determinar se o projeto pode lidar com as mudanças propostas. Um projeto inconsistente ou inflexível pode exigir tempo extra para que seja entendido, modificado e testado;
- dificuldades para a realização de testes: os testes podem ser um problema quando não se dispõe de tempo para a sua realização. Quando o sistema realiza uma função vital, pode ser impossível desligar o sistema para testá-lo. Nesses casos, freqüentemente os testes são realizados em ambientes de testes. Em seguida, as mudanças testadas são transferidas para o sistema de produção.

Na visão de Sommerville (SOMMERVILLE, 2005), há dificuldade de implementar mudanças rapidamente nos sistemas, pois os problemas nos reparos de emergência podem acarretar inconsistências nos requisitos, projeto de software e o código. Os engenheiros de manutenção podem ser orientados no sentido de tratar de novos reparos de emergência no software, e a solução adequada é adiada. Se o engenheiro que fez a mudança deixar a equipe, será mais trabalhoso para o substituto readequar as mudanças aos requisitos e ao projeto.

Outro problema com os reparos de emergência em sistema é que eles devem ser resolvidos o mais breve possível. Muitas vezes não é escolhida a melhor solução, com isso acelerando o envelhecimento do software, e futuras mudanças se tornam progressivamente mais difíceis e o custo da manutenção mais elevado (SOMMERVILLE, 2005).

3.3 Redes de Petri Estocástica

Um modelo é uma representação de um ou mais pontos de vista de um sistema em um determinado nível de abstração. Em um modelo formal, um sistema é representado através de um conjunto de variáveis, funções e relações matemáticas que representa uma ou mais perspectivas do sistema. Enquanto um sistema é “algo real", o modelo é uma “abstração", como, por exemplo, um conjunto de equações matemáticas (ARAUJO, 2009). A Figura 6 mostra a representação de um sistema através de um modelo.

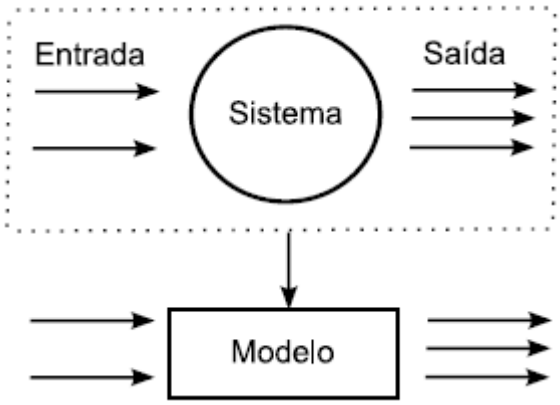


Figura 6: Processo de modelagem simples

Segundo Cassandras (CASSANDRAS, 1999) diversos modelos de representação têm sido utilizados para representar sistemas computacionais. Dentre eles, a redes de Petri é uma das mais utilizadas.

Redes de Petri (ou simplesmente RdP) foram criadas a partir da tese de doutorado de Carl Adam Petri, intitulada *Kommunication mit Automaten* (Comunicação com Autômatos), apresentada à Universidade de *Bonn* em 1962 (Marranghello, 2005). Desde então, esse formalismo tem sido

amplamente utilizado em diferentes áreas, tais como a Ciência da Computação, Engenharia Elétrica, Administração, Química, entre outras.

Diversas variantes do modelo de RdP clássico têm sido desenvolvidas ao longo do tempo, tais como redes temporizadas (Merlin, 1976), estocásticas (Marsan, 1989; Haverkort, 1998), alto-nível (ARAUJO, 2009) e orientadas a objetos (Janousek, 1998). Isso é devido à necessidade de suprir as diferentes áreas de aplicação, alem de prover facilidades de comunicação e transferência de métodos e ferramentas de uma área para outra (ARAUJO, 2009).

Diversas características fazem das redes de Petri um formalismo atrativo para a modelagem e análise de sistemas concorrentes e distribuídos, dentre elas é possível destacar as seguintes (GIRAULT, 2003):

- redes de Petri são um conjunto de formalismos que tem uma representação gráfica;
- fornecem mecanismos de refinamento e abstração que são de grande importância para o projeto de sistemas complexos;
- existe uma grande variedade de ferramentas computacionais disponíveis para as RdP, tanto comerciais quanto acadêmicas, para modelagem, analise e verificação;
- têm sido utilizadas em muitas áreas das ciências aplicadas e engenharias. Portanto, vários resultados são encontrados na literatura para os diferentes domínios da sua aplicação;
- existem varias extensões do modelo básico das RdP, que permitem tanto a representação de características básicas no estudo da concorrência, como também possibilita a analise de problemas práticos das organizações.

Graficamente, as RdP são representadas por lugares (**Figura 7** (a)), transições (**Figura 7**(b)), arcos (**Figura 7**(c)) e *tokens* (**Figura 7**(d)). Uma RdP é um grafo dirigido, onde os lugares e transições são seus vértices, interligados através de arcos dirigidos. Se a origem de um arco for um lugar, seu destino precisa necessariamente ser uma transição, e vice-versa.

A distribuição de *tokens* nos lugares da RdP determinam o estado do sistema.

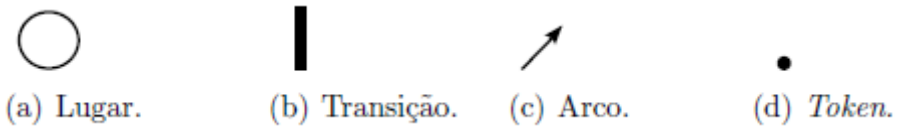


Figura 7: Elementos de uma rede de Petri

A Figura 8 apresenta uma RdP, que representa o ciclo repetitivo dos turnos (períodos) de um dia. O dia foi dividido em três períodos: **manhã**, **tarde** e **noite**, ou seja, há três condições. A transição de uma dessas condições para uma outra, por exemplo – **amanhecer** (noite → manhã), são os eventos. O modelo que representa o ciclo operacional desse sistema é formado pelas três condições, representadas por três variáveis de estado (lugares), e por três eventos (transições): **amanhecer**, **entardecer** e **anoitecer**. Para representar a situação atual, ou seja, em que condição encontra-se o sistema modelado, usa-se uma marca grafada (um ponto) no lugar que corresponde a essa situação, por exemplo: a condição atual é manhã (Figura 8 (a)).

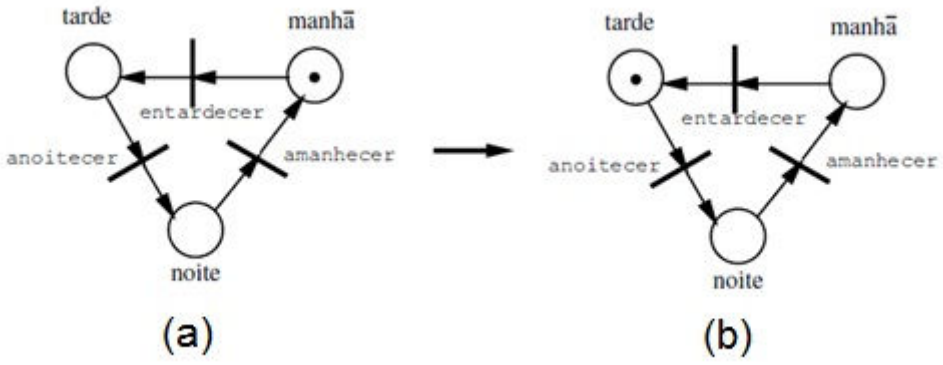


Figura 8: Períodos do dia

Nesse modelo, têm-se a condição atual representada pela marca no lugar **manhã** (Figura 8 (a)). Estando nessa condição o único evento que poderá ocorrer é **entardecer**, que é representado pela transição de mesmo nome. Na ocorrência desse evento têm-se uma nova situação atual, ou

seja: **tarde**, que é representada na (Figura 8 (b)) por uma marca no lugar tarde.

Dependendo do sistema modelado, as transições e os lugares de saída e entrada podem ter significados diferentes (Murata, 1989), conforme descritos na Tabela 1 (ARAUJO, 2009).

Tabela 1: Interpretações para os lugares e transições.

Lugares de Entrada	Transições	Lugares de Saída
Pré-condições	Eventos	Pós-condições
Dados de entrada	Passo e computação	Dados de saída
Sinal de entrada	Processamento de sinal	Sinal de saída
Disponibilidade de recursos	Tarefa	Liberação de recursos
Condição	Clausula lógica	Conclusões
Buffers	Processador	*Buffers*

Fonte: (ARAUJO, 2009; Murata, 1989)

As informações contidas nas colunas lugares de entrada, transições e lugares de saída na Tabela 1, representam respectivamente possíveis interpretações de lugares de entrada, transições e lugares de saída ao mapear em redes de Petri um sistema.

A Figura 9 ilustra um exemplo dessas interpretações. Os lugares parafusos e porcas são interpretações de lugares de entrada, ou seja, dados de entrada para a tarefa de montagem de pacote representada pela transição monta pacote. O lugar pacote é interpretação de um lugar de saída, ou seja, dados de saída da tarefa montagem de pacote. O mesmo também é interpretação de um lugar de entrada, isto é, dados de entrada para a tarefa de envio de pacote representado pela transição envia pacote. E o lugar depósito é interpretação de um lugar de saída, ou seja, dado de saída da tarefa de envio de pacote.

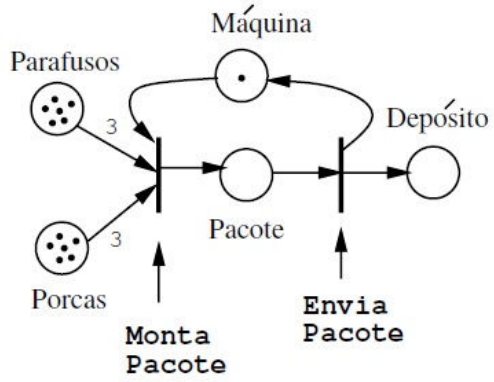


Figura 9: Linha de Produção

Uma Rede de Petri (RdP) é uma quíntupla, RP = (P, T, I, O, μ0), onde (ARAUJO, 2009):

- P é o conjunto de lugares.
- T é conjunto de transições, $P \cap T = \emptyset$;
- I, O : T X P → $\mathbb{N}$ são funções que denotam os lugares de entrada e saída das transições, respectivamente;
- μ0 : P → $\mathbb{N}$ é uma função que denota a marcação inicial dos lugares da rede.

As funções I e O descrevem, respectivamente, os arcos de entrada e saída de uma dada transição t Є T para um lugar Є P. Assim, para se obter o peso do arco que conecta p a t, utiliza-se a notação I (t, p). Essa notação também pode ser utilizada para a função de arcos de saída O. É usual representar estas funções através de uma notação matricial. Nesta notação, I é denominada matriz de entrada e O e a matriz de saída (ARAUJO, 2009).

Para exemplificar, supõe-se que se deseje representar um ano letivo de uma Universidade. O ano letivo começa com o primeiro período (semestre) letivo, seguido das primeiras férias (de julho), logo após, tem-se o segundo período letivo, e finalmente as férias de final de ano. Assim, o ano letivo poderia ser representado conforme a Figura 10.

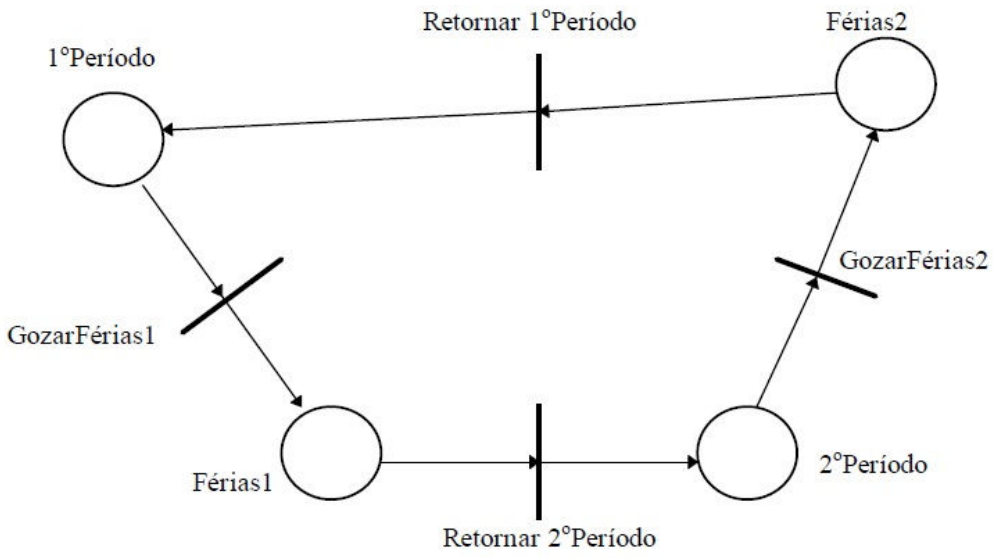


Figura 10: Ano Letivo Representado Graficamente em Redes de Petri

RAno_Letivo = (P, T, I, O, μ0), onde

o conjunto de lugares P é

P = {1oPeríodo, Férias1, 2oPeríodo, Férias2};

o conjunto de transições T é

T = {GozarFérias1, Retornar2oPeríodo, GozarFérias2, Retornar1oPeríodo};

o conjunto de transições para lugares de entrada I é

I = { I (GozarFérias1) = [1oPeríodo], I (Retornar2oPeríodo) = [Férias1],

I (GozarFérias2) = [2oPeríodo], I (Retornar1oPeríodo) = [Férias2] };

conjunto de transições para lugares de saída O é

O = {O (GozarFérias1) = [Férias1], O (Retornar2oPeríodo) = [2oPeríodo],

O (GozarFérias2) = [Férias2], O (Retornar1oPeríodo) = [1oPeríodo] };

e o conjunto de capacidades dos lugares é

μ0 = { μ01oPeríodo = 1, μ0Férias1 = 1, μ02oPeríodo = 1, μ0Férias2 = 1}.

As redes de Petri possui algumas extensões, a qual uma delas são as redes temporizadas. As redes de Petri temporizadas são extensões que buscam acrescentar às redes de Petri a possibilidade de análise no domínio de tempo (Marranghello, 2005).

Segundo Marrangnello (Marranghello, 2005) nestas extensões o tempo pode estar associado às marcas, aos arcos, aos lugares ou às transições. Quando associado às marcas, elas carregam uma informação indicando, geralmente, quando a marca estará disponível para ser considerada para a habilitação de transições. Quando a informação de tempo está associada aos arcos, a cada arco é associado um tempo de disparo. Quando a informação de tempo está associada aos lugares, usualmente, corresponde a uma indicação do tempo que a marca deve permanecer naquele lugar antes de ser utilizada para a habilitação das transições sucessoras do lugar em questão. Finalmente, quando o tempo é associado às transições a indicação refere-se, de modo geral, tempo que a ação leva para ser executada (Marranghello, 2005).

Estas extensões temporais podem ser determinísticas ou estocásticas. As extensões determinísticas surgiram na primeira metade da década de setenta e indicam tempos absolutos relativos à execução dos eventos correspondentes. As extensões estocásticas, por sua vez, permitem considerar incertezas nos instantes de execução de eventos do sistema associando a eles funções de probabilidade para a determinação de sua execução (Marranghello, 2005).

Neste trabalho foi usada a extensão temporal estocástica, pois o sistema avaliado no capitulo 5 é limitado, possui estados que ocorrem de forma temporal e imediata e associa tempo somente para alguns eventos que acredita-se ter um grande impacto na avaliação do sistema. Dessa forma a melhor extensão que se adéqua as características do sistema é o estocástico.

Segundo Vieira (Vieira, 2008) as redes de Petri estocásticas são dadas por:

- SPN = (P; T; F;M_0;R),
  onde (P; T; F;M_0) é uma rede de Petri, e
- R = {r_1; r_2;...; r_m}

é um conjunto de taxas de ativação (execuções por unidade de tempo) associadas com as transições da rede de Petri.

O tempo de espera em uma marcação M é dado por uma variável aleatória distribuída exponencialmente com média dada por

$$\left[\sum_{i \in H} r_i\right]^{-1}$$

onde H é o conjunto de transições ativas na marcação (Vieira, 2008).

A taxa de uma transição, a qual altera o estado da rede de Mi para Mj é

$$\sum_{k \in H_{ij}} r_k \ .$$

Onde Hij é o conjunto de transições que levam do estado Mi ao estado Mj . Este conjunto de transições é geralmente formado por apenas uma transição (Vieira, 2008).

Segundo Sales (SALES, 2002) uma característica importante do formalismo de redes de Petri estocásticas é que ele pode ser facilmente compreendido por pessoas que não tenham familiaridade com métodos de modelagem probabilística. Entretanto, a representação gráfica de sistemas utilizando este formalismo torna-se limitada à medida que aumentam o tamanho e a complexidade do sistema em questão.

Como dito anteriormente o formalismo de redes de Petri estocásticas permite duas classes diferentes de transições no modelo: transições imediatas e transições temporizadas (Haverkort, 1998).

As transições imediatas são aquelas que possuem um tempo de disparo igual a zero com prioridade superior às transições temporizadas e são representadas graficamente no modelo por barras finas. Estas transições disparam assim que são habilitadas. Níveis de prioridade podem

ser atribuídos às transições. Associam-se pesos às transições imediatas a fim de solucionar conflitos de disparos das transições (FAGUNDES, 2006; Haverkort, 1998).

As transições temporizadas são aquelas que disparam após um tempo aleatório, distribuído exponencialmente, associado à mesma quando habilitada. Estas transições são representadas graficamente no modelo por barras espessas. As taxas de disparos estão associadas somente às transições temporizadas, e estas podem ser dependentes da marcação da SPN para serem habilitadas e disparadas (SALES, 2002; Haverkort, 1998; Barros, 2001).

Na Figura 11, tem-se um exemplo de um modelo de rede de Petri estocástica.

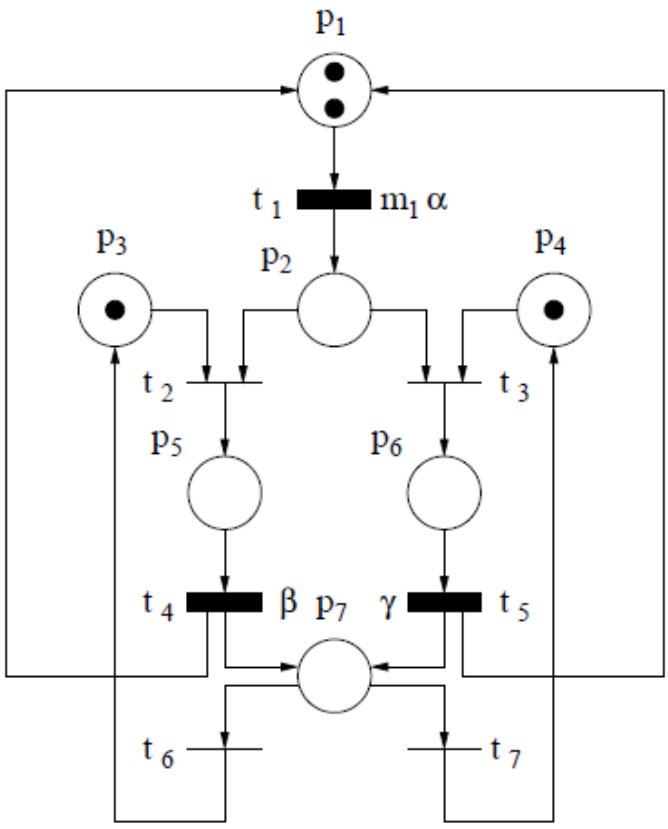


Figura 11: Exemplo de uma Rede de Petri Estocática

A SPN descrita na Figura 11 possui sete lugares e sete transições. Esta SPN possui três transições temporizadas: t1, t4 e t5, as quais

possuem como taxas de disparo $m_1\alpha$, β e γ respectivamente. As transições t2, t3, t6 e t7 são transições imediatas, e por definição possuem taxas de disparo de tempo zero. A transição t1 é dependente da marcação de p1.

3.3.1 Propriedades das Redes de Petri

Segundo Maciel (MACIEL, 1996) redes de Petri não se restringem apenas a uma ferramenta que possibilita a modelagem de problemas que tenham atividades concorrentes. Foi desenvolvida, em paralelo, uma série de métodos que permitem a análise de um grande número de propriedades em sistemas.

Diversas propriedades podem ser obtidas a partir dos modelos, permitindo assim revelar as mais diversas características do sistema. Essas propriedades podem ser subdivididas em comportamentais e estruturais, as quais são descritas nas seções seguintes.

3.3.1.1 Propriedades Comportamentais

As propriedades comportamentais são aquelas que dependem da marcação. Subscrevem-se aqui as principais propriedades comportamentais baseadas em (Murata, 1989; MACIEL, 1996).

Alcançabilidade (Reachability)

Alcançabilidade ou *Reachability* é fundamental para o estudo de propriedades dinâmicas de qualquer sistema. A alcançabilidade indica a possibilidade de atingirmos uma determinada marcação pelo disparo de um úmero finito de transições, a partir de uma marcação inicial (MACIEL, 1996).

Uma marcação M_0 é dita alcançável a partir de M_i, se existir uma seqüência de disparo que transforme M_0 em M_i. A seqüência de disparo é denotada pelo conjunto σ = {t1, t2,..., tn}. Nesse caso, M_i é alcançável a partir de M_0 por σ. Onde σ é formalmente descrito por M_0 [σ > M_i (ARAUJO, 2009).

Limitação e Safeness

Uma rede é dita *k-limitada*, se todos os seus lugares forem limitados, ou seja, o número de *tokens* em cada lugar não deve ultrapassar um número finito k, para qualquer marcação alcançável a partir de M_0. Uma rede de Petri é dita *safeness*, se k = 1 (ARAUJO, 2009).

Liveness

Uma rede é dita *live* se não importam quais marcações sejam alcançáveis a partir de uma marcação inicial m_0, se for possível disparar qualquer transição através do disparo de alguma seqüência de transições L(M_0). O conceito de *deadlock* está fortemente conectado ao conceito de *liveness*. No entanto, o fato de um sistema ser livre de *deadlock* não resulta que este seja *liveness*. Contudo, um sistema *liveness* implica em um sistema livre de *deadlocks*. A análise de *liveness* de uma rede permite verificar se os eventos modelados efetivamente ocorrem durante o funcionamento do sistema, ou se foram definidos eventos mortos no modelo (ARAUJO, 2009; Murata, 1989).

Cobertura (Coverability)

Segundo Maciel (MACIEL, 1996) a propriedade de cobertura está fortemente conectada ao conceito de alcançabilidade e *liveness*, apresentados nesta seção, respectivamente. Quando se deseja saber se alguma marcação M_i pode ser obtida a partir de uma marcação M_j, tem-se o problema denominado cobertura de uma marcação. Uma marcação M_i é dita coberta se existe uma marcação M_j tal que M_j = M_i. Fora isso, em alguns sistemas, deseja-se apenas observar o comportamento de determinados lugares. Para isso, restringe-se a pesquisa a apenas um conjunto de lugares de particular interesse (cobertura de submarcações).

Reversibilidade e Home State

Uma rede é dita reversível se, para cada marcação M em $R(M_0)$, M_0 é alcançável a partir de M. Assim, a rede possui a capacidade de retornar à marcação inicial. Além disso, em algumas aplicações não é necessário voltar à marcação inicial, mas sim a uma marcação específica. Essa marcação específica é denominada *Home State* (ARAUJO, 2009).

3.3.1.2 Propriedades Estruturais

As propriedades estruturais são aquelas que refletem características independentes de marcação (MACIEL, 1996). Ou seja, as propriedades estruturais possuem dependência exclusivamente da topologia da rede (ARAUJO, 2009). Abaixo serão descritas as principais propriedades estruturais baseadas em (Murata, 1989; MACIEL, 1996)

Limitação Estrutural

Uma rede é dita limitada estrutural, se o número de *tokens* é limitado para qualquer marcação inicial.

Conservação

É uma importante propriedade das Redes de Petri, pois permite a verificação de não destruição de recursos através da simples conservação de marcas.

Repetitividade

Uma rede é considerada como repetitiva se para uma marcação e uma seqüência de transições disparáveis, todas as transições dessa rede são disparadas ilimitadamente.

Consistência

Uma rede é considerada consistente se disparando uma seqüência de transições a partir de uma marcação inicial M_o, ele retorna a M_o, porém todas as transições da rede são disparadas pelo menos uma vez.

3.3.2 Aproximação por fases

A utilização da técnica de aproximação por fases (Desrochers, 1995; Malhotra, et al., 1993) tem sido muito comum para representar o comportamento de uma distribuição desconhecida. Essa distribuição desconhecida pode ser aproximada para outra distribuição conhecida, por exemplo, Erlang, Hipo-exponencial e Hiper-exponencial (Trivedi, 2006). É importante mencionar que essa técnica tem sido usada com sucesso na modelagem de atividades não-exponenciais (ARAUJO, 2009).

As transições das redes de Petri estocásticas podem ser imediatas ou estocásticas, cujo tempo apresenta um atraso exponencialmente distribuído. Portanto, a técnica de aproximação por fases é utilizada para representar tempos que sigam outra distribuição (Desrochers, 1995; Malhotra, et al., 1993).

A aproximação é realizada através da construção de uma sub-rede cujo throughput segue a distribuição desejada, conforme mostra a Figura 12 (ARAUJO, 2009). Esta sub-rede é construída utilizando-se o moment matching, que consiste em aproximar os dois primeiros momentos da distribuição, calculando-se a média (μD) e o desvio padrão (σ).

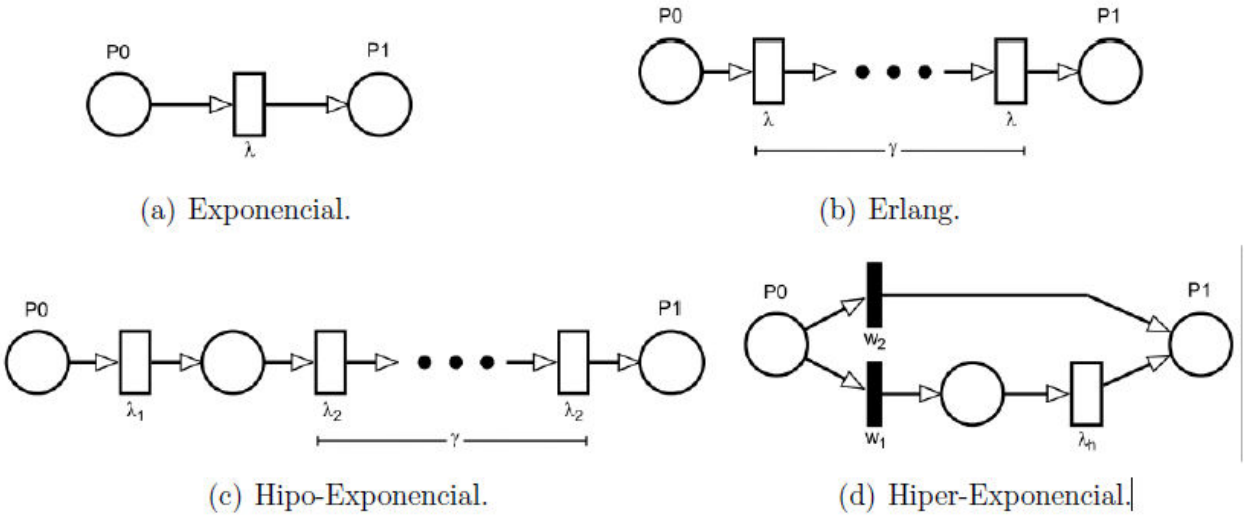


Figura 12: Sub-rede GSPN para representar distribuições polinômio-exponenciais utilizando *moment matching*.

A partir dos valores calculados para μD e σD, utilizam-se os passos e equações a seguir para realizar a aproximação por fases (Desrochers, 1995).

1. Se $\mu D/\sigma D = 1$, utiliza-se a subrede que representa uma Distribuição Exponencial (Ver Figura 12(a)). É usada uma transição exponencial com taxa $\lambda = \mu D$;
2. Se $\mu D/\sigma D \in \mathbb{Z} \wedge \mu D/\sigma D \neq 1$, utiliza-se a subrede que representa uma Distribuição Erlang (Ver Figura 12(b)) pela

Equação $\gamma = \left(\frac{\mu D}{\sigma D}\right)^2$. A taxa da transição exponencial é calculada pela Equação $\lambda = \frac{\gamma}{\mu D}$.

3. Se $\mu D/\sigma D > 1$, utiliza-se a sub-rede que representa uma Distribuição Hipo-Exponencial (Ver Figura 12 (c)) pela Equação $\left(\frac{\mu D}{\sigma D}\right)^2 - 1 \leq \gamma < \left(\frac{\mu D}{\sigma D}\right)^2$. A taxa das transições exponenciais é calculada pelas Equações $\lambda_{1} = \frac{1}{\mu_1}$ e $\lambda_{2} = \frac{1}{\mu_2}$. Os respectivos *delays* (valores medidos) atribuídos à transição exponencial são calculados pelas Equações:

$$\mu_1 = \mu D \mp \frac{\sqrt{\gamma\,(\gamma+1)\,\sigma^2} - \gamma\mu^2}{\gamma+1} \text{ e}$$

$$\mu_2 = \gamma\mu D \pm \frac{\sqrt{\gamma\,(\gamma+1)\,\sigma^2} - \gamma\mu^2}{\gamma+1}$$

4. Se $\mu D/\sigma D < 1$, utiliza-se a sub-rede que representa uma Distribuição Hiper-Exponencial (Ver Figura 12 (d)). A taxa da transição exponencial deve ser calculada com a Equação $\lambda_h = \frac{2\mu D}{\mu D^2 + \sigma D^2}$ e os pesos das transições imediatas são calculados pelas Equações: $\omega_1 = \frac{2\mu D^2}{\mu D^2 + \sigma D^2}$ e $\omega_2 = 1 - \omega_1$.

3.4 Considerações Finais

Este capítulo apresentou fundamentos referentes ao procedimento de controle de mudança de software e redes de Petri. Através desses fundamentos foi possível concluir que o procedimento de controle de mudança garante que as mudanças propostas a um sistema sejam avaliadas e implementadas de forma controlada e consistente. Assim, o fluxo de trabalho da mudança de requisitos é definido e pode ser repetido, as solicitações de mudança registradas e analisadas facilitam a comunicação clara entre equipe e cliente, e pode-se avaliar objetivamente o status do projeto através das estatísticas de taxa de mudanças. Concluiu-se, também, que as redes de Petri permitem a modelagem de sistemas de diferentes áreas de aplicação, possibilitando a análise de domínio do tempo que uma ação leva para ser executada, refinamento e abstração do que são de grande importância para o sistema e permitem o estudo de concorrência e análise de problemas práticos de uma organização.

CAPITULO 4 – METODOLOGIA DE AVALIAÇÃO DE DESEMPENHO

Este capítulo apresenta uma metodologia para auxiliar a modelagem e avaliação de desempenho do processo de mudança de software. A metodologia proposta apresenta diversas etapas, desde o entendimento do ambiente, seleção de métricas de desempenho até a validação do modelo e interpretação dos resultados.

4.1 Introdução

As organizações buscam melhorar continuamente seus processos e melhorar continuamente é estabelecer processos organizacionais, racionalmente planejados, tendo em vista a evolução e o aperfeiçoamento ininterrupto de práticas de gestão, perceptíveis por todas as partes interessadas na busca constante pela excelência de resultados (Filho, 2010).

Com intuito de prover um melhor entendimento dos processos das organizações, a melhoria contínua destes processos e a avaliação de desempenho dos mesmos este capítulo apresenta uma metodologia para avaliação de desempenho de um processo de mudança de software a fim de apoiar projetos de software das organizações no planejamento do processo de mudança.

Mudanças no software são imprescindíveis e podem ser solicitadas a qualquer momento. As mudanças no software precisam ser controladas através de um processo de mudança de software. Elas são armazenadas e a análise de impacto é realizada pelo comitê de controle de mudança para decisão de realização da mudança e sua prioridade (RUP). É de particular interesse deste trabalho analisar métricas relacionadas a esforço e custo de utilização do CCB, baseado na quantidade de mudanças que são solicitadas para suporte ao planejamento de execução desse processo.

Para tanto, é proposta a modelagem do processo de mudança de software em modelos de desempenho baseados em *redes de Petri estocástica*.

A Figura 13 ilustra a metodologia concebida e contextualiza o ambiente no qual este trabalho está inserido, destacando suas principais atividades em: compreender o processo e o problema, definir as métricas, mapear o modelo do processo em redes de Petri, gerar modelo abstrato, medir e tratar os dados, refinar modelo, validar o modelo e avaliar o modelo considerando o cenário especificado, interpretar e apresentar os resultados. É importante ressaltar que esta metodologia pode ser utilizada para avaliação de sistemas que apresentem características semelhantes.

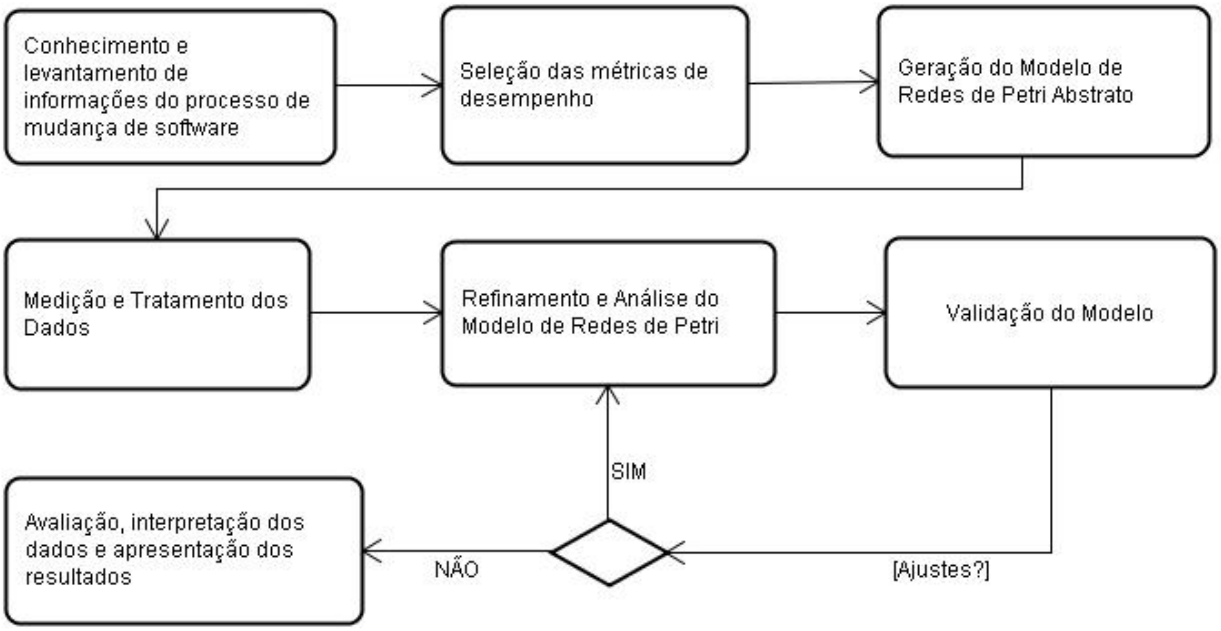


Figura 13: Fluxograma da metodologia de avaliação de desempenho.

A metodologia é composta das seguintes atividades:

1. Conhecimento e levantamento de informações do processo de mudança de software: compreende o entendimento geral do processo de mudança de software. Nesta atividade, realiza-se um estudo e entendimento do processo de solicitação de mudança no software utilizado por um projeto ou organização. É importante conhecer cada etapa do processo, as suas atividades, os artefatos, pré-condições e pós-condições, papéis e ferramentas. Nesta atividade, é realiza-se o entendimento do problema, o que é necessário melhorar. Quais as

falhas do processo do sistema analisado estão gerando custos à organização ou ao projeto? Existem retrabalhos? Existe algum planejamento para execução do processo? Quais os papéis alocados para participar da análise do CCB?

2. Seleção das métricas de desempenho: corresponde à seleção das métricas de desempenho relevantes para avaliação do processo de mudança, tais como utilização do CCB e custo para análise de impacto.
3. Geração do modelo de redes de Petri abstrato: o fluxo do processo de mudança de software é transformado em modelo de redes de Petri abstratos, permitindo assim, que este fluxo seja validado. Além disso, são especificadas as métricas selecionadas.
4. Medição: corresponde à configuração do ambiente para medição. Os dados obtidos são utilizados para modelagem, validação e avaliação de cenário representado no modelo.
5. Tratamento dos dados: Esta atividade corresponde ao tratamento estatístico das informações, garantindo, assim, que as mesmas sejam confiáveis.
6. Refinamento e análise do modelo de redes de Petri: o modelo de redes de Petri abstrato é refinado, com a finalidade de tornar o modelo mais representativo.
7. Validação do modelo: Esta atividade analisa a rede de Petri Estocástica gerada. A validação é conduzida da seguinte forma: Dadas as métricas que se têm e as métricas que se obtém no modelo é feita uma comparação dos resultados através de teste t-emparelhado (Apêndice A). Esta comparação deve esta dentro de um erro de no mínimo 15%.
8. Avaliação, interpretação e apresentação dos resultados: os dados produzidos pelos experimentos são interpretados e os resultados serão apresentados.

Nas seções seguintes as atividades da metodologia serão detalhadas.

4.2 Conhecimento e levantamento de informações do processo de mudança de software

A primeira etapa da metodologia concebida nesta dissertação tem como finalidade o entendimento e levantamento de informações do processo de mudança de software. Essa etapa proverá uma base de conhecimento inicial sobre o processo de mudança a ser modelado. É

importante salientar que nesta fase são adquiridos os insumos necessários (atividades, papéis que executam as atividades, artefatos de entrada e saída de cada atividade, entre outros) para a modelagem do processo de mudança de software. Caso não sejam bem definidos esses insumos, o avaliador provavelmente cometerá erros de interpretação e, conseqüentemente, comprometerá a execução das etapas seguintes.

A metodologia ora apresentada baseia-se na modelagem de um processo de mudança de software através de redes de Petri. A partir dessas redes de Petri podem-se obter métricas que nos auxiliam no entendimento do funcionamento do processo de mudança. Uma vez analisado o ambiente, pode-se avaliar diferentes cenários visando encontrar uma configuração adequada. Com o estudo consolidado acerca do ambiente, o avaliador terá insumos para a elaboração dos cenários a serem adotados no processo de mudança.

O tamanho do projeto, o tamanho da equipe, a severidade das solicitações de mudança e os papéis que participam da análise de impacto (CCB) das solicitações de mudança são fundamentais na definição dos cenários a serem avaliados. Projetos com grandes equipes, por exemplo, comumente possuem um número maior de solicitações de mudança e com diversos níveis de severidade e normalmente possuem a participação de papéis que possuem um custo alto na análise de impacto. Portanto, na execução da atividade de análise de impacto das mudanças do processo de solicitações de mudanças, os projetos podem gerar custos altos, caso envolva os papéis incorretos e de valor alto (como, por exemplo, envolver o arquiteto na análise de impacto de uma solicitação de mudança de severidade baixa), indicando um mau planejamento da execução do processo de mudança de software.

Na atividade de análise de impacto, o CCB possui a participação de papéis como: analista de sistemas, arquiteto, engenheiro de software e gerente de projeto. Cada papel possui um custo. Quando um determinando papel é alocado para realização da análise de impacto de uma solicitação

de mudança, e esta análise não necessitaria da sua atuação, ele poderá gerar um custo desnecessário ao projeto.

As informações adquiridas nesta etapa da metodologia servirão de suporte para as próximas etapas, mais especificamente na escolha do fluxo do processo de mudança de software que será abordado durante a execução da metodologia.

4.3 Seleção das Métricas de Desempenho

Após o entendimento do processo de mudança de software, o avaliador deve identificar as métricas de interesse para representar o desempenho do processo de mudança. A seguir, são listadas algumas métricas relevantes identificadas para este trabalho:

- Utilização do CCB: é o percentual de tempo utilizado pelo CCB para realizar a análise de impacto da solicitação de mudança registrada.
- Custo CCB: é o custo que o projeto possui com o CCB quando o mesmo realiza a análise de impacto da solicitação de mudança registrada.
- Utilização do CM: é o percentual de tempo utilizado pelo CM para realizar a geração da *baseline*.
- Custo CM: é o custo que o projeto possui com o CM quando o mesmo realiza a geração da *baseline*.

As métricas escolhidas para avaliar o desempenho do processo de mudança de software serão posteriormente especificadas e formuladas segundo a sintaxe suportada pelas ferramentas de avaliação adotadas.

4.4 Geração do Modelo de Rede de Petri Abstrato

Esta fase concerne à geração do modelo redes de Petri abstrato que represente o sistema a ser avaliado. Esse modelo é denominado abstrato, pois não possui os detalhes que descrevem o desempenho do sistema. Posteriormente, estas características serão incluídas no modelo para que esse tenha uma representação. Estas características serão representadas através da utilização da técnica de aproximação por fases (ARAUJO, 2009; Desrochers, 1995).

Os parâmetros necessários para o processo de mudança serão descritos de maneira mais detalhada no Capítulo 5. A Figura 14 apresenta uma rede de Petri abstrata para o processo de mudança de software. Esse modelo representa a abertura da solicitação da mudança, a análise de impacto realizada pelo CCB, a implementação da mudança pela equipe, a verificação da mudança pelo testador e a realização da *baseline* pelo CM.

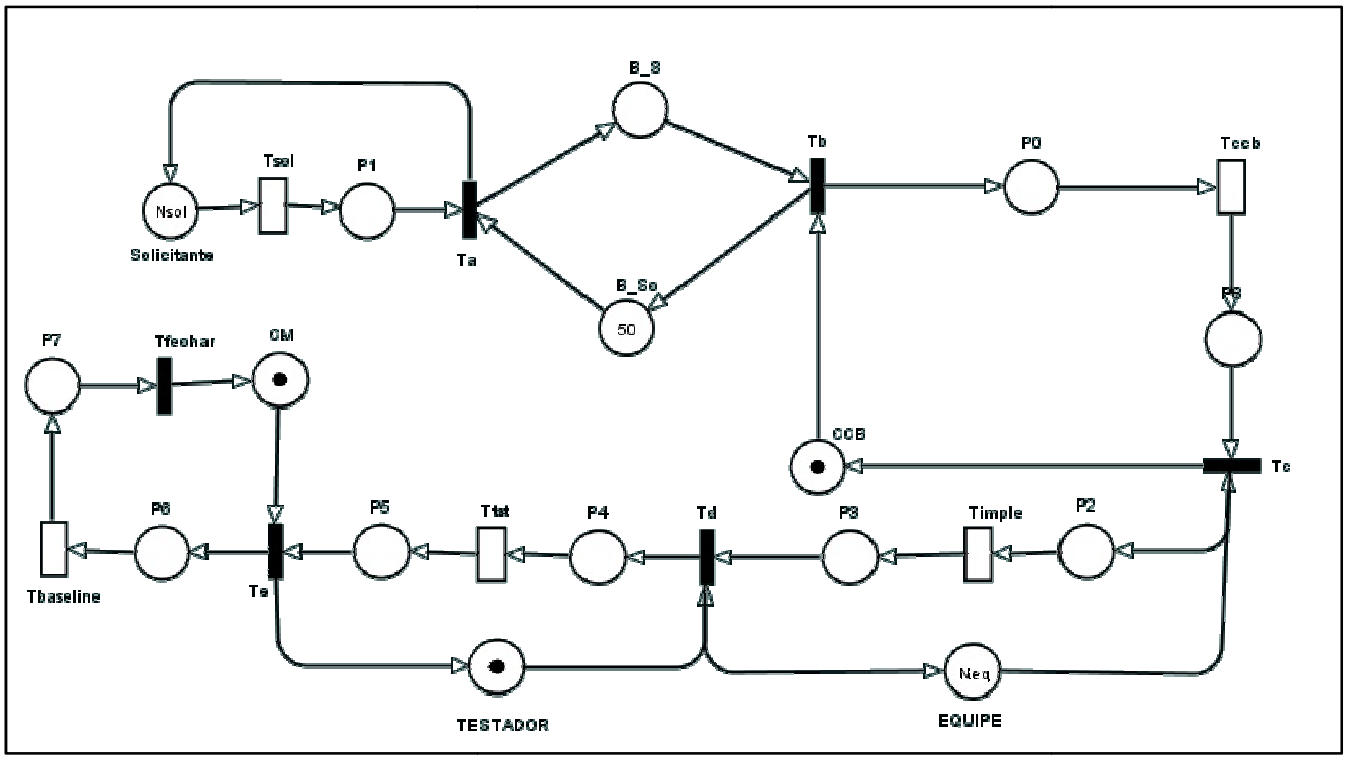


Figura 14: Modelo redes de Petri abstrato para o processo de mudança de software.

A geração do modelo abstrato é definida com base na fase de entendimento do ambiente. Após a criação do modelo abstrato, detalhado no capítulo 5, pode-se realizar um *token game* (Zimmerman, 2001) do modelo em redes de Petri, visando validar o seu comportamento abstrato.

4.5 Medição e Tratamento dos Dados

Para que o sistema seja refinado, é necessário que se obtenha ou que se colete dados acerca do sistema, como por exemplo a média, o desvio padrão, mediana, entre outros. Os dados coletados influenciarão nas demais etapas da metodologia.

Para a obtenção dos dados, é possível encontrar dados históricos disponíveis, indisponíveis, confiáveis e não confiáveis. Quando os dados históricos estão disponíveis e foram coletados de maneira apropriada, a análise se torna mais fácil (ARAUJO, 2009). O avaliador usará estes dados para medir o processo de mudança de software.

Antes mesmo de medir os dados, o avaliador deve escolher as ferramentas a serem utilizadas para a coleta dos dados e especificar os requisitos referentes à medição, tais quais o tamanho da amostra e o formato de armazenamento dos dados medidos (ARAUJO, 2009).

Após a obtenção dos dados, sejam obtidos de dados históricos ou de medição específica, é fundamental que estes estejam organizados e armazenados para análise (ARAUJO, 2009), para que sejam tratados em seguida.

A análise dos dados coletados na fase de medição da metodologia, permite que os dados medidos não apresentem anormalidades, ou seja, discrepâncias estatísticas (Triola., 2005). Nesta atividade, realiza-se o tratamento estatístico das informações, mas antes da análise estatística, faz-se uma análise exploratória delas. A análise exploratória permite avaliar distorções, e inconsistência nos dados.

4.6 Refinamento e Análise do Modelo de Redes de Petri

Esta etapa trata sobre o refinamento do modelo abstrato. Para o refinamento do modelo abstrato, o avaliador deve observar se o nível de abstração utilizado é adequado à avaliação do processo de forma precisa. As estatísticas média (μ) e desvio-padrão (σ) são utilizadas para encontrar uma distribuição de probabilidade que represente as respectivas atividades. Esta abordagem é conhecida como aproximação por fases (ARAUJO, 2009; Desrochers, 1995) e tem sido usada com sucesso para a modelagem de atividades não exponenciais.

A Figura 15 ilustra o modelo em redes de Petri refinado equivalente a uma parte do modelo abstrato do processo de mudança de software

apresentado anteriormente. A técnica de aproximação por fases foi aplicada à transição Tccb. Visando aproximar o valor do tempo de análise de impacto da solicitação de mudança, a transição Tccb foi aproximada para uma distribuição erlang.

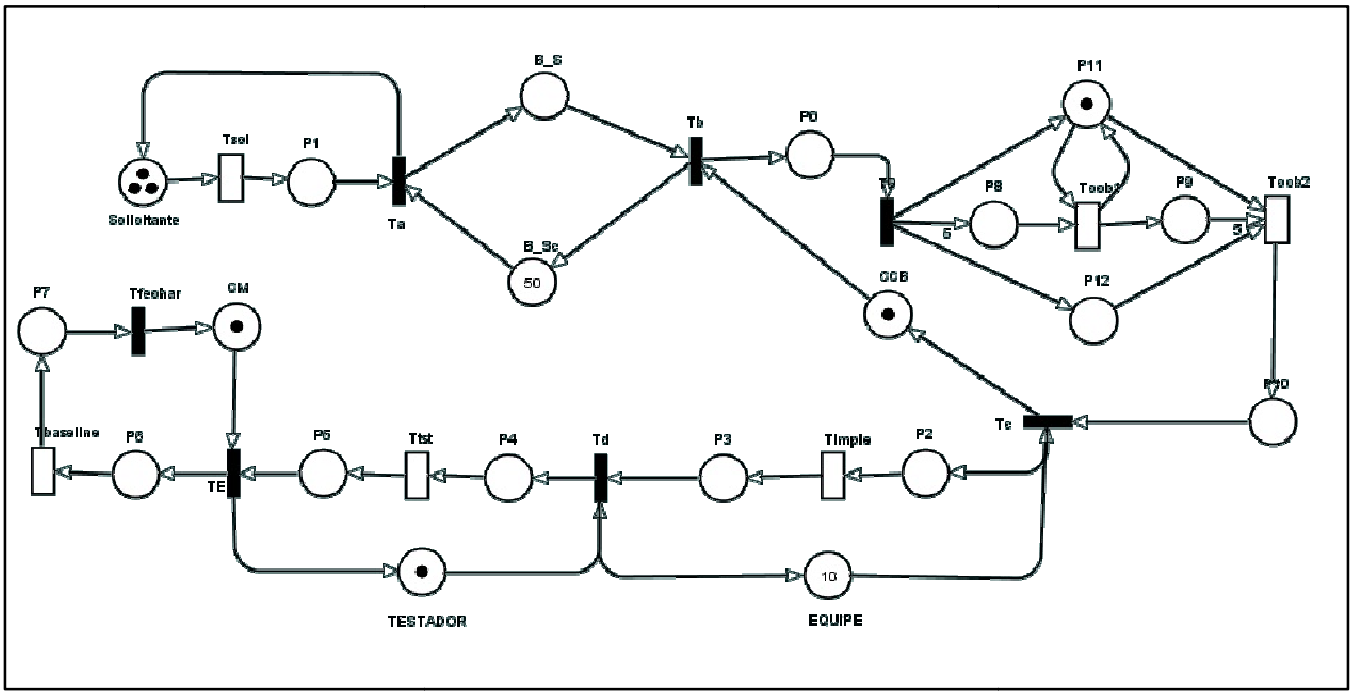


Figura 15: Modelo SPN refinado para o processo de mudança de software.

Existem pontos considerados importantes na escolha de uma aproximação por fase, como a qualidade da aproximação, o número de estados da aproximação e a facilidade de obtenção do modelo resultante, visto que, alguns modelos usados para a aproximação podem produzir resultados mais satisfatórios em relação aos outros, pois pode não ser fácil a integração no modelo resultante (ARAUJO, 2009).

4.7 Validação do Modelo

Nesta etapa, o modelo refinado é analisado, verificado e validado. Esta atividade visa analisar e verificar um conjunto de propriedades estruturais e comportamentais associadas ao modelo de rede de Petri e

também validar o modelo abstrato da rede de Petri através do *token game*[1] (Zimmerman, 2001).

A análise qualitativa captura aspectos lógicos do sistema. Entre as propriedades de interesse, pode-se ressaltar a análise da existência de *deadlock*, *liveness*, limitação e reversibilidade. Essas propriedades estão detalhadas no Capítulo 3.

A validação quantitativa tem por objetivo prover evidências que corroborem a representatividade do sistema pelo modelo (ARAUJO, 2009). A representatividade do modelo diz respeito à capacidade de se replicar os valores das métricas do sistema através do modelo concebido. Para isso, cenários especialmente planejados devem ser especificados. A quantidade de solicitações de mudança abertas e resolvidas em um determinado período de tempo é um valor que pode ser obtido, tanto no modelo quanto através de medição no sistema. A quantidade de solicitações abertas e resolvidas foi considerada como uma métrica para validação do modelo.

Essa validação pode ser realizada considerando o valor médio estimado no modelo em relação aos dados obtidos no sistema real. Os resultados dessas métricas devem ser obtidos utilizando uma avaliação estacionária, ou uma avaliação transiente (ARAUJO, 2009). Nem todos os modelos de rede de Petri gerados propiciam o cálculo de métricas em regime estacionário, dado que nem todos os modelos alcançam esse estado. Para que as métricas possam ser calculadas no estado estacionário, é necessário que a rede de Petri seja limitada. Os modelos propostos neste trabalho são estruturalmente limitados, permitindo, assim, o cálculo de métricas no estado estacionário.

Na avaliação de estado transiente, os valores obtidos das métricas são calculados considerando-se um intervalo de tempo. Já a avaliação de estado estacionário, computa as métricas desprezando os efeitos transientes iniciais do modelo (ARAUJO, 2009).

[1] Simulação da Rede, também chamada de *token game*, é a tarefa de avaliar as transições habilitadas e de proceder os seus disparos, levando a rede a passar por uma seqüência de estados ao longo do tempo (Oliveira, 2006).

Para que o modelo seja considerado válido, é necessário que este proporcione o cálculo de métricas com erros de exatidão dentro de níveis considerados aceitáveis. Caso não se consiga obter os valores próximos aos medidos, é necessário revisar o modelo e o processo de refinamento, visando encontrar erros cometidos nessa etapa (ARAUJO, 2009).

4.8 Avaliação, Interpretação dos dados e apresentação dos resultados

Esta última etapa tem o objetivo de avaliar os cenários e situações de interesse através dos modelos das redes de Petri concebidos.

Os cenários criados devem representar as situações de interesse do processo de mudança. A fase de entendimento do ambiente deve fornecer insumos suficientes para que o avaliador tenha o entendimento do processo de mudança de software, provendo uma base de conhecimento inicial sobre o mesmo e insumos necessários para a modelagem do processo de mudança de software, como, por exemplo, o conhecimento das atividades do processo de mudança de software, os papéis que executam cada atividade, o tempo que cada papel executa cada atividade, entre outras. A análise dos diversos cenários tem como objetivo avaliar as situações de interesse para tomada de decisão, no que diz respeito, por exemplo, ao planejamento do gerente do projeto na alocação dos papéis que devem compor a formação do comitê de controle de mudança que estará avaliando as solicitações de mudança que chegam e são de severidade alta. Portanto, é responsabilidade do avaliador interpretar os dados e apontar as possíveis soluções para o problema analisado. As métricas podem, por exemplo, apontar um possível gargalo no sistema como, por exemplo, o comitê de controle de mudança não está dando conta de analisar em tempo hábil as solicitações de mudança que chegam causando um atraso no desenvolvimento das mesma, ou ainda, a necessidade de melhorias no desempenho do sistema. Além disso, a conclusão dos cenários com gráficos e tabelas são apresentados em documento, auxiliando a apresentação desses resultados.

4.9 Considerações Finais

Este capítulo apresentou a metodologia adotada para avaliação de desempenho do processo de mudança de software. Em resumo, essa metodologia iniciou a partir da compreensão geral do processo de mudança de software, em seguida as métricas de desempenho foram selecionadas e o modelo abstrato foi gerado. Com o modelo abstrato gerado a medição e tratamento dos dados são realizados, e o modelo é refinado para validação. Após modelo validado, é feita a avaliação, interpretação e apresentação dos resultados. Dessa forma, a metodologia proposta apresenta um conjunto de atividades definidas para o processo de avaliação do desempenho do processo de mudança.

CAPITULO 5 – MODELO PROPOSTO

Este capítulo apresenta modelos SPN do processo de mudança de software e o estudo de caso utilizado para validar tanto os modelos SPN propostos, quanto a metodologia de avaliação.

5.1 Processo de Mudança de Software

Esta seção apresenta o processo de mudança de software utilizado na realização deste trabalho.

A Figura 16 do diagrama de atividades da UML apresenta o processo de mudança de software que foi modelado em redes de Petri. Este processo foi baseado no processo de mudança de software de uma organização de desenvolvimento de software da cidade do Recife/ Pernambuco.

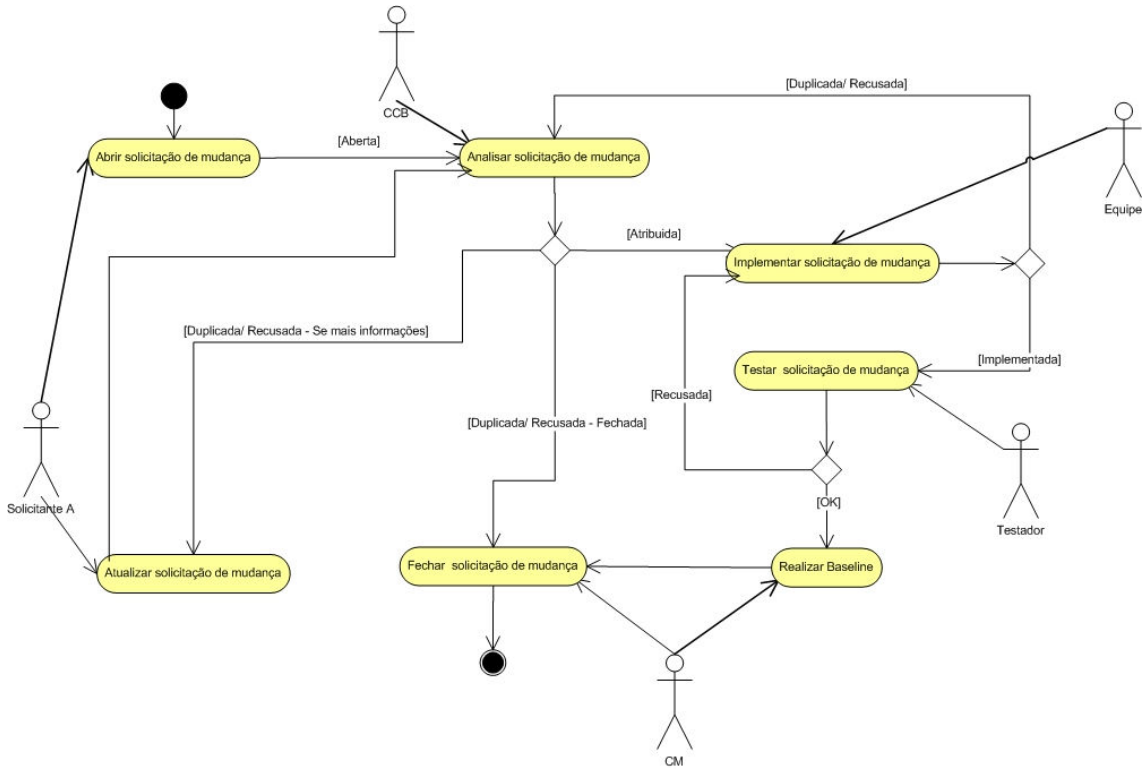


Figura 16: Processo de Mudança de Software

O processo da Figura 16 é iniciado por um conjunto de pedidos de mudança por parte de um dos envolvidos no sistema. O custo e o impacto dessas mudanças são calculados para ver quanto do sistema é afetado pela mudança e quanto pode custar para implementar essa mudança. As mudanças são implementadas e testadas e uma nova versão estável do sistema é liberada (SWEBOK, 2004).

Optou-se por utilizar as redes de Petri, pois elas fornecem mecanismos de refinamento e abstração que são de grande importância para projetos de sistemas complexos, existe uma grande variedade de ferramentas disponíveis para modelagem, análise e verificação e possui várias extensões para representação de características de estudo de concorrência e análise de problemas práticos das organizações (GIRAULT, 2003).

5.2 Modelos

Esta seção apresenta os modelos que representam a Solicitação, *Buffer*, CCB e *Baseline*. Para cada modelo, será apresentada a sua representação gráfica e a sua descrição. Como também suas métricas de desempenho.

5.2.1 Solicitação

O modelo Solicitação compreende parte do processo de mudança, que tem por finalidade a abertura das solicitações de mudança de um projeto.

O modelo SPN Solicitação (Ver Figura 17) representa a abertura e chegada das solicitações de mudança a serem analisadas pelo comitê do controle de mudança (CCB).

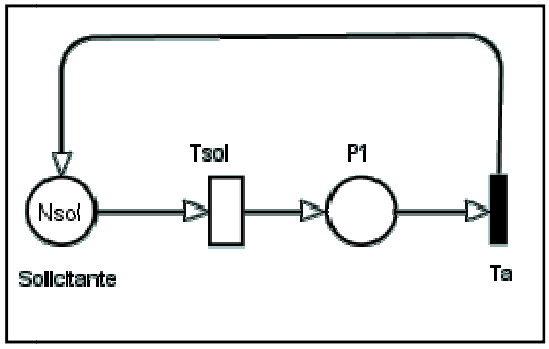


Figura 17: Modelo SPN para a Solicitação

A transição Tsol representa o envio das solicitações de mudança para o *buffer*[2]. O disparo dessa transição é gerado a uma taxa de λ_{Tsol}. Adicionalmente, com o disparo da transição Tsol, um *token* é armazenado no lugar P1, em seguida é disparado pela transição Ta, sendo armazenado no lugar B_S do modelo *buffer*.

A partir do modelo que representa a Solicitação, o avaliador pode adicionar outros modelos Solicitação para as mudanças abertas por projetos diferentes.

5.2.2 Buffer

O modelo *buffer* representa o mecanismo de armazenamento e repasse das solicitações de mudanças abertas para o CCB. É possível, através desse modelo, verificar a quantidade de solicitações de mudança abertas que chegam e ainda estão em espera de análise pelo CCB. A Figura 18 apresenta o modelo que representa o buffer do processo de solicitação de mudança.

[2] *É* uma região de memória temporária utilizada para escrita e leitura de dados. Os dados podem ser originados de dispositivos (ou processos) externos ou internos ao sistema.

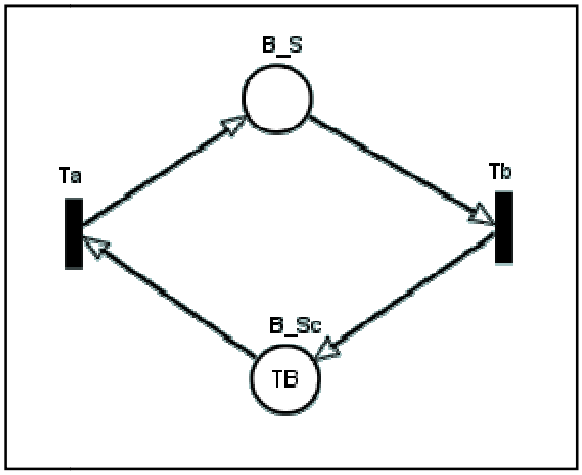


Figura 18: Modelo SPN para o buffer.

A transição Ta representa o recebimento das solicitações de mudança para armazenamento no *buffer*. O disparo dessa transição representa o armazenamento das solicitações que chegam para análise de impacto realizada pelo CCB. Nesse momento um *token* é armazenado no lugar B_S e debitado do lugar B_Sc que representa o tamanho do *buffer*. Adicionalmente, com o disparo da transição Tb, um *token* retorna ao lugar B_Sc.

5.2.3 CCB

A atuação do CCB é considerada determinante no desempenho do processo de mudança de software, pois este recurso é que recebe uma maior demanda de atividades. Este recurso é responsável pela análise de impacto das solicitações de mudança que são abertas. A Figura 19 apresenta o modelo para representar a análise de impacto feita pelo CCB no processo de mudança de software e a Figura 20 apresenta a composição dos modelos solicitação, buffer e CCB.

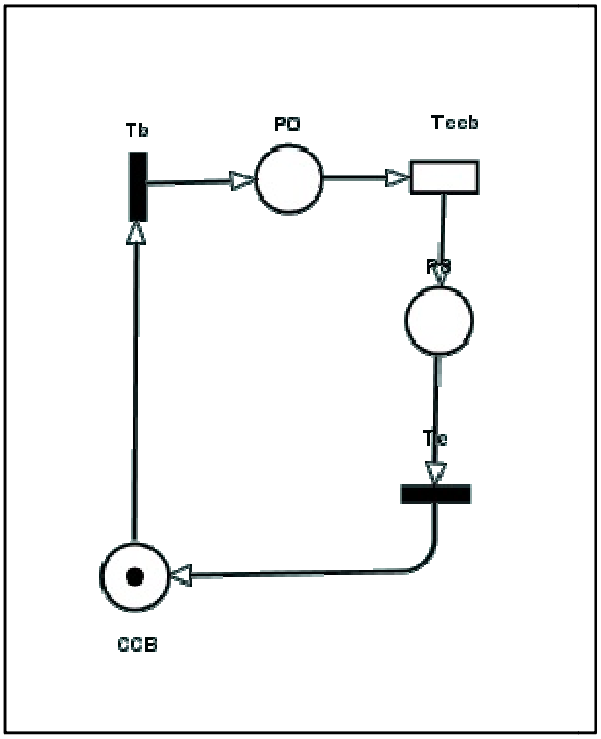


Figura 19: Modelo SPN para a Análise de Impacto feita pelo CCB.

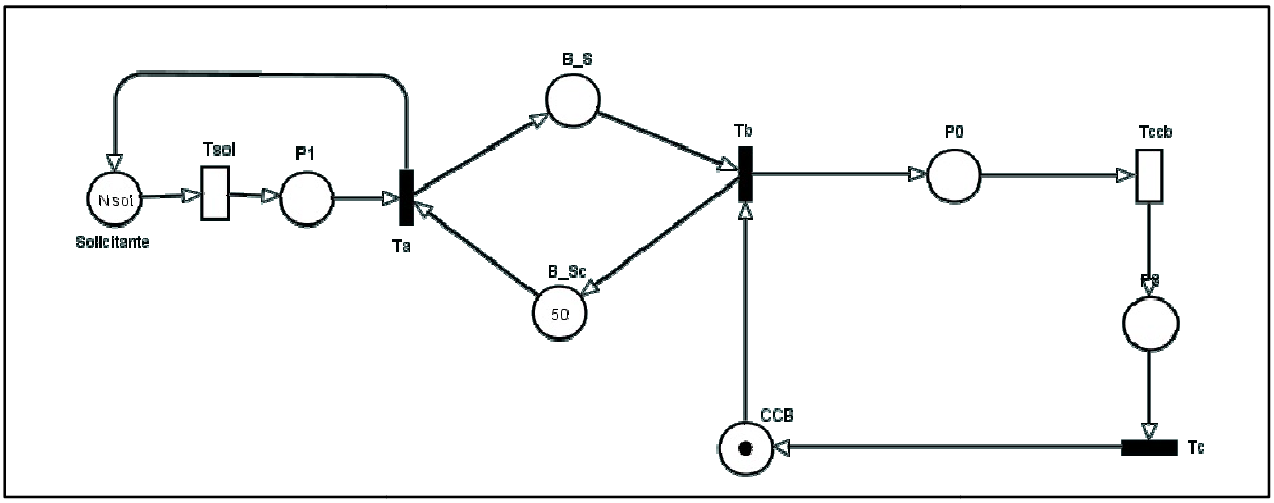


Figura 20: Modelo SPN composto com a Solicitação, Buffer e Análise de Impacto feita pelo CCB

Um *token* no lugar CCB denota que o recurso CCB está ocioso. O disparo da transição Tb representa a solicitação para analisar uma solicitação de mudança pelo CCB e o disparo de Tccb indica a análise de impacto da solicitação de mudança pelo CCB e é realizada a uma taxa de λ_{Tccb}. Por conseguinte, quando o recurso CCB estiver em uso, o lugar CCB

estará sem *token*. O disparo da transição Tc indica que a atividade de análise de impacto foi concluída e que uma nova solicitação de mudança pode ter ser impacto analisado pelo CCB, pois um *token* é armazenado no lugar CCB. É relevante mencionar que a técnica de aproximação por fases apresentada no Capítulo 3 pode ser aplicada à transição Tccb, pois observou-se que ao comparar os dados do sistema com o do modelo em redes petri, os valores estavam muito distantes. O modelo CCB é utilizado pelo avaliador através da fusão deste modelo com o modelo buffer. A transição Ta referente ao modelo buffer e a transição Tb referente ao modelo CCB evidenciam esta fusão.

Após definido o modelo que representa a análise de impacto, é necessário definir as métricas relacionadas a esse modelo. A Tabela 2 apresenta as métricas utilizadas.

A expressão da métrica utilização do CCB apresentada na Tabela 2 calcula a probabilidade do papel do CCB, representado pelo lugar CCB, ser igual a zero, identificando a sua utilização. E a expressão da métrica custo do CCB, calcula do custo do CCB através da sua utilização, multiplicado pela soma do valor/hora dos papéis que formam o CCB, multiplicado pelo tempo que está sendo analisado a execução do processo de mudança de software.

Tabela 2: Métricas do modelo CCB.

Métrica	Expressão
Utilização do CCB	P{#CCB=0}
Custo do CCB	(P{#CCB=0})*CCCB*T

5.2.4 *Baseline*

A atuação CM (*Configuration Manager*) ou Gerente de Configuração é considerada importante no desempenho do processo de mudança de software, pois este recurso é que recebe as solicitações de mudança para geração de uma *baseline*. A Figura 21 apresenta o modelo para representar a geração da *baseline* feita pelo CM no processo de mudança de software e

a Figura 22 apresenta o modelo abstrato composto dos modelos propostos nesse trabalho.

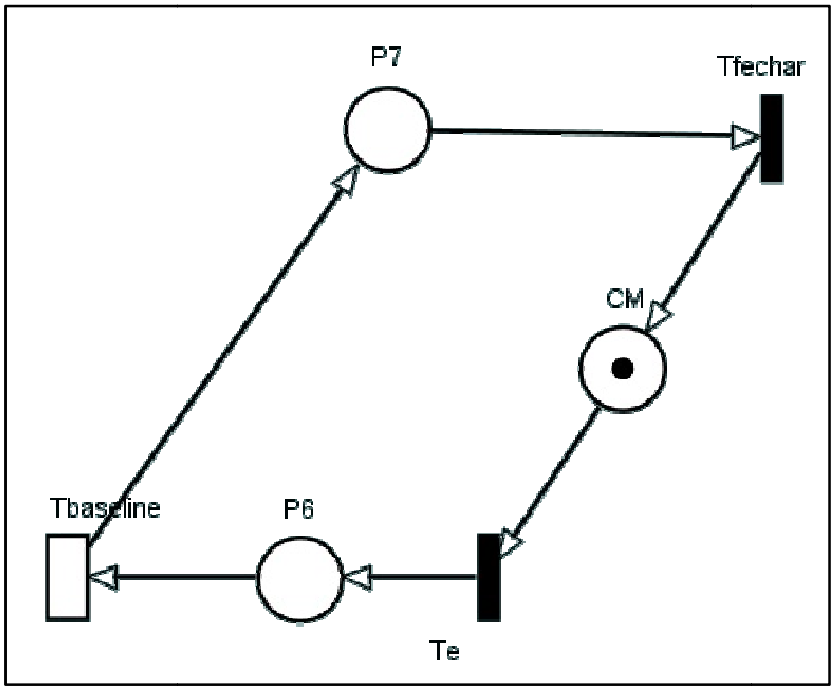


Figura 21: Modelo SPN para Geração de *Baseline* pelo CM.

Um *token* no lugar CM denota que o recurso CM está ocioso. O disparo da transição Te representa a solicitação para geração da *baseline* pelo CM e o disparo de Tbaseline indica que a geração da *baseline* está sendo realizada. Essa geração de *baseline* é realizada a uma taxa de $\lambda_{Tbaseline}$. Por conseguinte, quando o recurso CM estiver em uso, o lugar CM estará sem *token* armazenado. Para que uma nova geração de *baseline* seja realizada, é necessário haver o disparo da transição Tfechar. Portanto, o disparo da transição Tfechar indica que a atividade geração de *baseline* foi concluída. Um *token* armazenado no lugar CM indica também essa atividade.

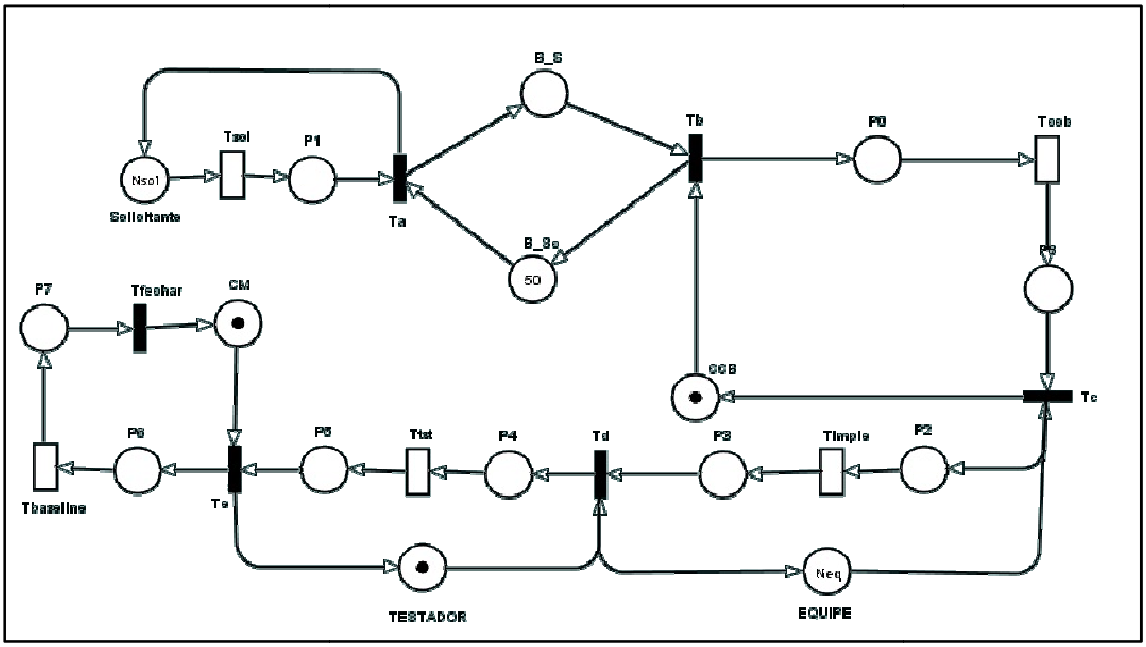


Figura 22: Modelo SPN abstrato composto por todos os modelos propostos nesse trabalho

Após definido o modelo que representa a geração de *baseline* feita pelo CM, é necessário definir a métrica relacionada a esse modelo. A Tabela **3** apresenta as métricas utilizadas.

Tabela 3: Métricas do modelo Mudança Finalizada.

Métrica	Expressão
Utilização do CM	P{#CM=0}
Custo do CM	(P{#CM=0})*CCM*T

A expressão da métrica utilização do CM apresentada na Tabela 3 calcula a probabilidade do papel do CM, representado pelo lugar CM, ser igual a zero, identificando a sua utilização. E a expressão da métrica custo do CM, calcula do custo do CCB através da sua utilização, multiplicado pelo valor/hora do CM, multiplicado pelo tempo que está sendo analisada a execução do processo de mudança de software.

5.3 Validação

Esta seção apresenta o estudo de caso relativo à validação do modelo e metodologia de avaliação.

O objetivo do estudo de caso é validar tanto os modelos SPN propostos, quanto a metodologia de avaliação. Para tanto, todas as etapas da metodologia foram utilizadas para auxiliar na avaliação desses estudos, a fim de demonstrar a aplicabilidade destes.

O método adotado para validar o modelo SPN consiste na comparação dos resultados obtidos através dos modelos SPN propostos com os dados medidos no processo de mudança de software real. Nesse sentido, procura-se demonstrar a adequação do modelo SPN e da metodologia desenvolvida.

5.3.1 Compreensão e levantamento de informações do processo e seleção de métricas.

Nessa primeira etapa foi compreendido o processo de mudança de nos projetos selecionados para validação.

Foi analisado o processo de mudança, referido na seção 5.1 de 4 (quatro) projetos de uma empresa de desenvolvimento de software da cidade do Recife, Pernambuco, a partir da coleta de dados nas ferramentas, documentos dos projetos e conversa com especialistas.

O Projeto 1, é um projeto que utiliza o ciclo de vida cascata e possui um tamanho de 1871 pontos de função (PF)[3]. A equipe deste projeto é composta de 15 pessoas, contendo 1 gerente de projeto, 2 analistas de sistemas, 1 arquiteto, 10 engenheiros de software e 1 engenheiro de configuração (CM). No período de 6 meses analisado, o projeto encontrava-se na execução da fase de testes. Já o Projeto 2, usa o ciclo iterativo e

[3] É uma unidade de medida de software reconhecida pela ISO para estimar o tamanho de um sistema de informação baseando-se na funcionalidade percebida pelo usuário do sistema, independentemente da tecnologia usada para implementá-lo. O método para medir o tamanho de um sistema de informação e expressá-lo em um número de pontos de função é chamado de Análise de Pontos de Função (APF) (NESMA).

incremental, também se encontrava na execução da fase de testes da iteração 3 e foi analisado um período de 3 meses. O seu tamanho é de 1059 pontos de função e a equipe é formada por 7 pessoas, contendo 1 gerente de projeto, 1 analista de sistemas, 1 arquiteto, 3 engenheiros de software e 1 engenheiro de configuração (CM). O terceiro projeto, com um tamanho de 1628 pontos de função possui uma equipe formada por 11 pessoas, contendo 1 gerente de projeto, 1 analista de sistemas, 1 arquiteto, 7 engenheiros de software e 1 engenheiro de configuração (CM). Utiliza o ciclo de vida iterativo e incremental e o período analisado também foi de 3 meses e estava na execução da fase de testes da iteração 2. O Projeto 4 possui um tamanho de 458 pontos de função e sua equipe é composta de 6 pessoas, contendo 1 gerente de projeto, 1 analista de sistemas, 1 arquiteto, 2 engenheiros de software e 1 engenheiro de configuração (CM). O projeto usa também o ciclo iterativo e incremental e no período de 3 meses analisado, o projeto estava na execução da fase de testes.

Tabela 4: Resumo dos Projetos analisados

Projeto	Tamanho	Tamanho equipe	Situação período analisado
Projeto 1	1871	15	Fase de Teste
Projeto 2	1059	7	Fase de Teste
Projeto 3	1628	11	Fase de Teste
Projeto 4	458	6	Fase de Teste

A Tabela 4 apresenta um resumo dos projetos analisados na validação do modelo.

As métricas utilizadas para a avaliação do processo de mudança foram: a utilização do CCB que é o percentual de tempo de utilização do CCB ao realizar a análise de impacto, o custo do CCB que é o custo que o projeto possui na utilização do CCB na análise de impacto, a utilização do CM que é o percentual de tempo de utilização do CM ao realizar a geração da *baseline* e o custo do CM que é o custo que o projeto possui na utilização do CM na geração da *baseline*. Estas métricas foram escolhidas para analisar o custo da atividade de análise de impacto realizada pelo

CCB e do CM na atividade de geração da *baseline* no processo de mudança de software.

5.3.2 Geração do Modelo Abstrato de Desempenho

As informações obtidas na fase de entendimento do processo de mudança de software devem ser suficientes para a construção do modelo SPN.

O modelo abstrato da SPN foi construído a partir do processo de mudança compreendido e modelado através do mapeamento do diagrama de atividades da UML em redes de Petri. No mapeamento o elemento Ação no diagrama de atividades representa uma especificação de um comportamento que tem diversos inputs que serão transformados em um conjunto de *outputs*. Na Rede de Petri o elemento transição é o componente que representa a ação e que pode alterar o estado do sistema. Por possuírem a mesma funcionalidade o elemento Ação é transformado no elemento transição. Como o elemento atividade Inicial do diagrama de atividade apenas passa o fluxo para o próximo elemento, ele será representado como um elemento Lugar sem nenhuma marca, pois não demonstra nenhuma mudança de estado. O elemento Decisão direciona o fluxo em diversas direções que são determinadas de acordo com uma ação anterior. Esse elemento é mapeado como um Lugar sem Marca na rede de Petri. O elemento União é mapeado com um elemento Lugar, assim como o elemento Decisão. O elemento Separação tem como objetivo duplicar o fluxo e com isso representa uma ação. Logo ele será mapeado como uma transição na rede de Petri, que tem a mesma função de produzir uma ação. Como o elemento de Junção tem como objetivo unir o elemento Separação, ele também representa uma ação que deve ser representado com uma Transição. Como já mencionado, os elementos Final de Fluxo e Final de Atividade finalizam o fluxo de *tokens*. O conceito de finalização de *tokens* não existe em uma rede de Petri e por isso ele não será mapeado na transformação. O último elemento é o Objeto, ele representa no diagrama de atividade inputs e outputs para as Ações. Na Rede de Petri isso é

mapeado como um Lugar com uma Marca que representa a mudança de estado do Objeto em questão (LACHTERMACHER, 2008).

A Figura 23 ilustra o modelo abstrato do processo de mudança de software.

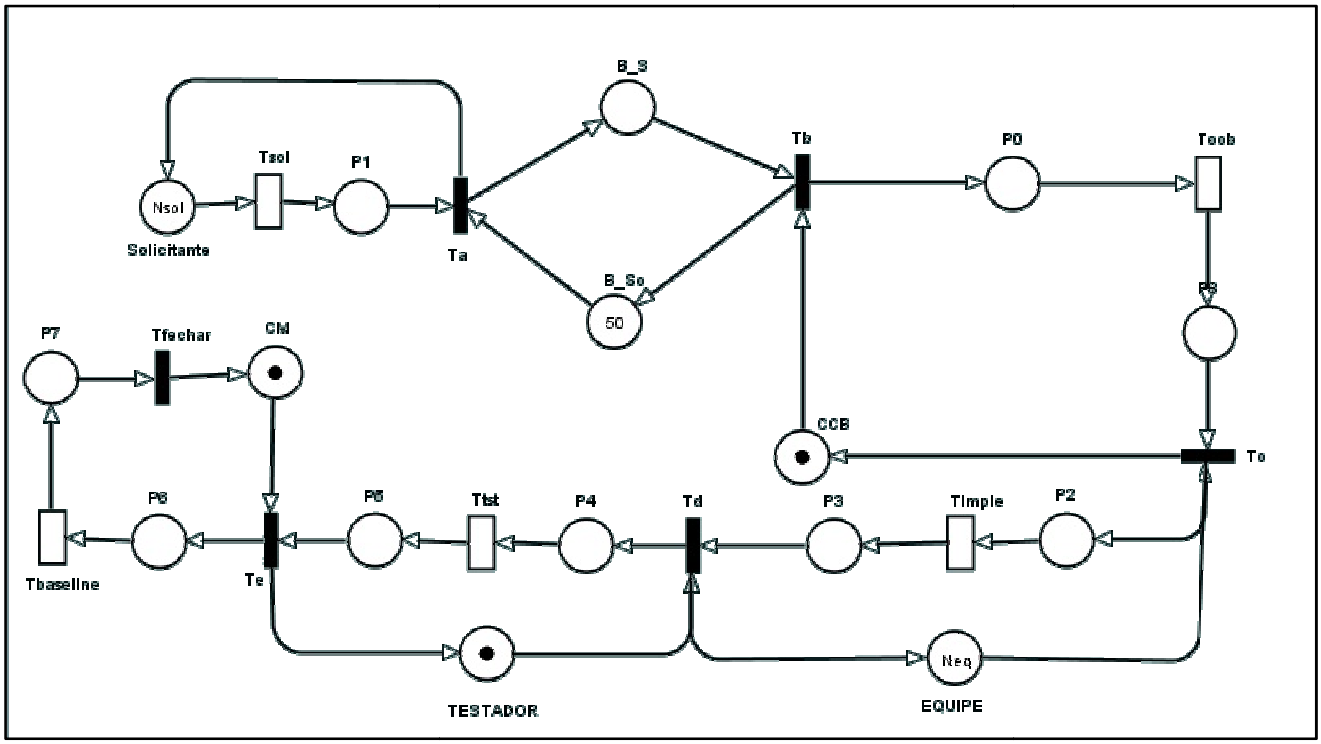


Figura 23: Modelo abstrato do processo de mudança de software

A validação qualitativa do modelo abstrato foi realizada através de um *token game*, cujo intuito foi reproduzir o funcionamento genérico das transações sob ponto de vista do desempenho (Zimmerman, 2001) (Murata, 1989). O modelo abstrato apresenta o processo geral de mudança de software, desde a solicitação de mudança ao momento que a mudança entra formalmente em uma versão estável do software. O modelo exibe a abertura de uma solicitação de mudança (Tsol) por um solicitante, representado pelo lugar Solicitante, a mudança é armazenada em um *buffer*, representado pelo lugar B_S, até que a solicitação tenha seu impacto analisado (Tccb) pelo CCB, representado pelo lugar CCB. Após aprovada a implementação da mudança a mesma é implementada (Timple) pela equipe, representada pelo lugar equipe e testada (Ttst) pelo testador representado pelo lugar testador e formalmente relacionada a uma versão

estável do software (Tbaseline) realizada pelo CM representada pelo lugar CM.

5.3.3 Medição

Na quarta etapa da metodologia os dados relativos aos parâmetros do sistema do processo de mudança de software são coletados. A coleta consiste na medição eventos de interesse no sistema, na adoção de dados históricos e na obtenção de informações de especialistas no processo de mudança. Os resumos estatísticos ou as respectivas informações providas pelos especialistas são atribuídos a elementos (lugares, transições, arcos e marcações) do modelo de desempenho. Os resumos estatísticos ou as respectivas informações providas

Vale salientar que os dados dos projetos utilizados para analisar o processo de mudança de software, foram extraídos de ferramentas e documentos utilizados por projetos reais de uma organização de desenvolvimento de software da cidade do Recife. Por motivos de privacidade, não será citado o nome dos projetos e da organização. Após extração e conduziu-se uma análise exploratória cujo objetivo foi, eventualmente, identificar as solicitações de mudanças abertas que tornavam o tempo médio de chegada das solicitações muito distorcido.

5.3.4 Tratamento dos Dados

Após coleta dos dados, os dados são analisados. A Tabela 5 apresenta um breve resumo estatístico referentes aos dados coletados na fase de medição. Os parâmetros (μ_D), (σ_D), (C_V), M_i, Q_1, Q_3, IQ, H foram calculados e representam, respectivamente, a média, desvio-padrão, coeficiente de variação, mediana, quartil 1, quartil 3, intervalo inter-quartil e amplitude referentes ao tempo de chegada das solicitações de mudanças abertas. O tempo de chegada das solicitações de mudanças abertas está associado à atividade de abertura de solicitação de mudança.

Tabela 5: Resumo Estatístico dos Projetos.

Projetos	μ_D	σ_D	C_V	M_i	Q_1	Q_3	IQ	H
Projeto 1	2,28	4,98	2,18	0,37	0,10	1,28	1,18	23,66
Projeto 2	3,95	7,02	1,78	0,49	0,16	2,81	2,65	23,87
Projeto 3	4,41	7,06	1,60	0,57	0,15	4,51	4,36	23,01
Projeto 4	2,99	5,8	1,94	0,53	0,23	1,75	1,52	22,58

Os dados coletados referentes ao tempo de chegada das solicitações de mudanças abertas que apresentam um grande afastamento das restantes ou são inconsistentes com elas são habitualmente designadas por *outliers*, designados por observações “anormais”, contaminantes, estranhas ou extremas (Figueira, 1998). Antes de decidir o que deverá ser feito com estes dados é conveniente ter conhecimento das causas que levam ao seu aparecimento. Em muitos casos as razões da sua existência determinam as formas como devem ser tratados. Assim, as principais causas que levam ao aparecimento de *outliers* são erros de medição, erros de execução e variabilidade inerente dos elementos da população.

Inicialmente foi realizada a identificação dos dados que são potencialmente estranhos. A identificação foi realizada por observação direta dos dados. Em seguida foi eliminado da subjetividade destes dados, pois sua correção é inviável.

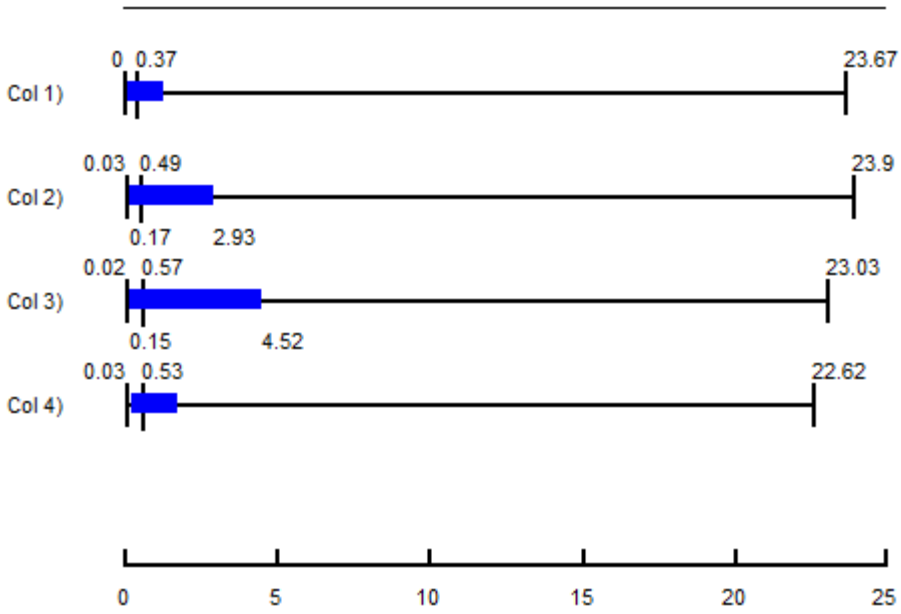


Figura 24: Gráfico *Boxplot*

As Figura 24 mostra o gráfico *Boxplot* referente aos valores mínimo, Q_1 Q_3 e máximo dos 4 (quatro) projetos analisados. As bases do retângulo no *boxplot* têm alturas correspondentes aos primeiro e terceiro quartis da distribuição. O retângulo é cortado por um segmento paralelo às bases, na altura correspondente ao segundo quartil. Assim, o retângulo do *boxplot* corresponde aos 50% valores centrais da distribuição. No gráfico são traçados dois segmentos. Um segmento paralelo ao eixo, partindo do ponto médio da base superior do retângulo até o maior valor observado que não supera o valor de Q3+(1,5). E ponto médio da base inferior do retângulo, até o menor valor que não é menor do que Q1-(1,5). No gráfico *boxplot* as observações que estiverem acima de Q3+(1,5) ou abaixo de Q1-(1,5) são chamadas pontos exteriores e são destoantes das demais. No caso do gráfico representado na Figura 24 não existem pontos destoantes, logo não possui *outliers*.

5.3.5 Refinamento do Modelo

Esta etapa da metodologia concerne ao refinamento do modelo de redes de Petri abstrato. Os parâmetros μ_D, σ_D,C_V, foram utilizados para refinar o modelo abstrato visando determinar, empiricamente, distribuições

de probabilidades que aproximem satisfatoriamente o comportamento das variáveis aleatórias correspondentes aos tempos de disparo das transições temporizadas representadas no modelo. A técnica de aproximação por fases foi utilizada para determinar qual distribuição de probabilidade expolinomial era mais adequada para representar os dados medidos. Essa técnica utilizou a média (μ) e o desvio padrão (σ) para obter os tempos a serem associados à transição genérica Tccb do modelo abstrato. Conforme apresentado no Capítulo 2, a técnica de aproximação por fases adotada utiliza as seguintes funções de probabilidade Erlang, Hipo-exponencial e Hiper-exponencial.

A métrica selecionada para a avaliação do modelo foi: número de solicitações de mudanças abertas e resolvidas no período analisado, utilizando a seguinte expressão: (P{#P0>=1}*(1/TA))*T. Essa métrica representa o número de solicitações de mudanças que foram abertas e entraram em *baseline, ou seja,* foi incorporada a uma versão estável do sistema.

Na aproximação por fases da transição Tccb correspondente a análise de impacto foi escolhida a distribuição Erlang para modelar uma transição temporizada com peso nos arcos. O tempo médio 0,61h da transição temporizada Tccb foi obtido através do conhecimento e experiência de gestores da organização, pois a organização não registra o tempo de análise de impacto dessa etapa, logo não possui dados históricos.

Essa distribuição foi modelada através de duas transições temporizadas. A Figura 25 apresenta o modelo refinado do processo de mudança de software. A transição Tccb foi refinada para representar distribuição Erlang com o intuito de aproximar satisfatoriamente o comportamento a variável aleatória correspondente ao tempo de disparo da transição temporizada. O modelo refinado apresenta o processo geral de mudança de software, desde a solicitação de mudança ao momento que a mudança entra formalmente em uma versão estável do software. O modelo exibe a abertura de uma solicitação de mudança (Tsol) por um solicitante,

representado pelo lugar Solicitante, a mudança é armazenada em um *buffer,* representado pelo lugar B_S até que a solicitação tenha seu impacto analisado pelo CCB, onde depois de refinado o modelo representado pelas transições Tccb-1 e Tccb-2 e pelo lugar CCB. Após aprovada a implementação da mudança a mesma é implementada (Timple) e testada (Ttst) e formalmente relacionada a uma versão estável do software (Tbaseline).

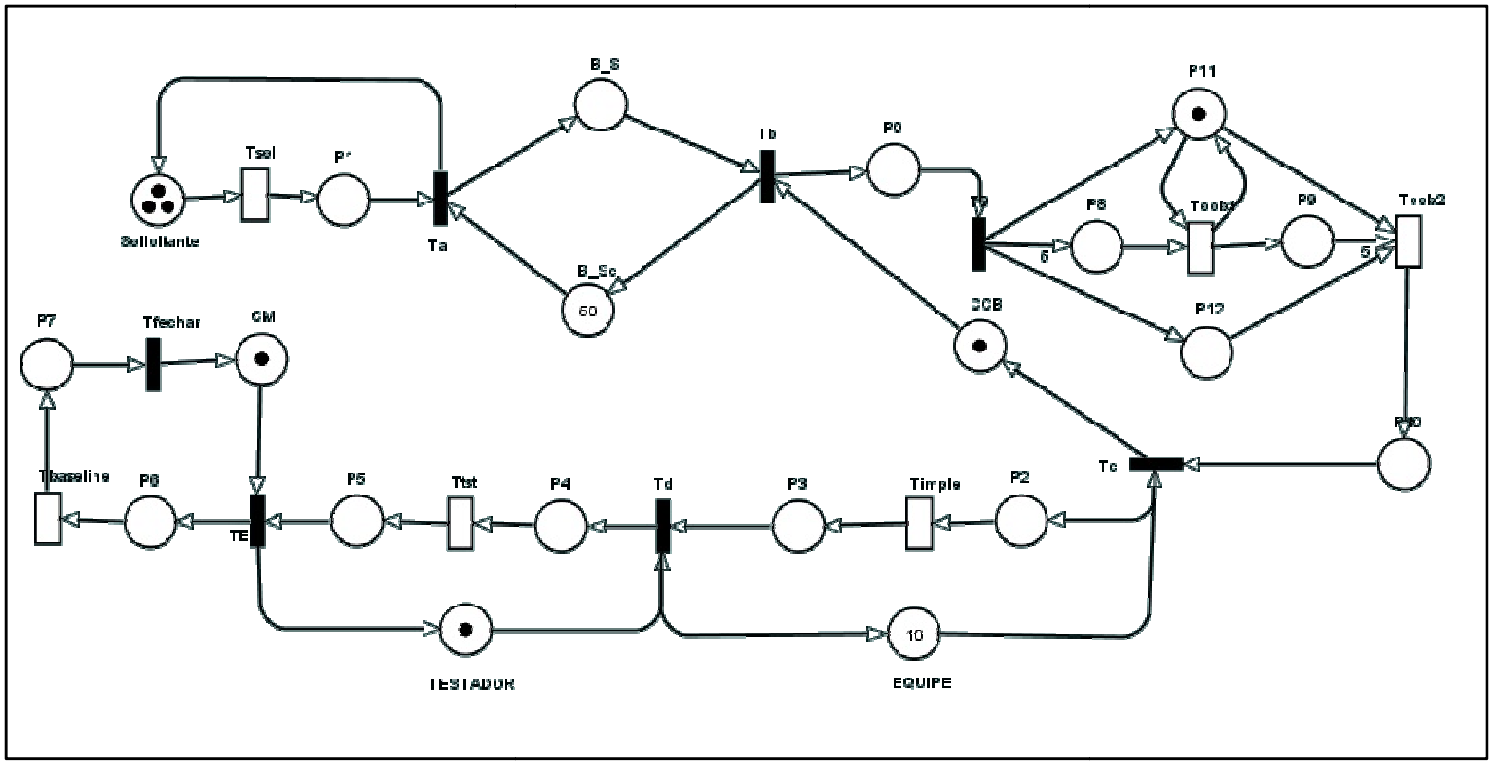


Figura 25: Modelo redes de Petri refinado do processo de mudança de software.

5.3.6 Validação do Modelo

Esta etapa consiste na validação do modelo refinado foi analisado e verificado. Para validação, adotou-se a análise estacionária do modelo, pois a rede é limitada. Esse método de avaliação deve ser utilizado quando as redes geradas são estruturalmente limitadas e as métricas de interesse sejam independentes do tempo. Um *tokengame* também foi executado para validar o comportamento da rede de Petri.

Além do refinamento por fases, um modelo de desempenho, considerando apenas os tempos médios, também foi gerado. Este modelo normalmente é menos exato, contudo o espaço de estado é sempre significativamente menor que o correspondente obtido a partir do modelo expolinomial.

Para a validação, foram especificados 4 cenários representativos. O número de solicitações de mudanças abertas coletadas foi de um período de 3 meses de cada projeto. Os dados coletados foram armazenados em uma planilha.

Os quatro experimentos tiveram por objetivo comparar o número de solicitações de mudança abertas e realizadas no período de 3 meses do modelo do processo de mudança com o numero de solicitações abertas se realizadas do processo real.

As médias de abertura de solicitação de mudança utilizadas em cada cenário foram 2,28; 3,95; 4,41 e 2,99 horas. A Tabela **6** apresenta a configuração referente aos experimentos.

Tabela 6: Configuração dos Experimentos.

Projetos	Tempo Médio entre Chegadas	Tempo Médio Análise do CCB
Projeto 1	2,28	0,61
Projeto 2	3,95	0,61
Projeto 3	4,41	0,61
Projeto 4	2,99	0,61

Para validar o modelo refinado, os dados medidos pelo sistema foram comparados com os resultados da métrica de interesse obtidos através da avaliação do modelo. A métrica adotada para a validação do modelo foi o número de solicitações abertas e resolvidas em um determinando período.

A Figura 26 apresenta a comparação dos valores medidos do processo de mudança de software real com os valores calculados do modelo, referentes a métrica número de solicitações abertas e resolvidas em um determinando período. O erro médio encontrado na métrica foi de -16%.

Os resultados obtidos através do modelo e das medições do sistema foram comparados por meio do teste T-emparelhado. Pode-se constatar com 95% de grau de confiança, que os resultados não evidenciam diferença significativa entre os dados medidos e os valores obtidos através do modelo.

Tabela 7: Resumo Estatístico dos Projetos.

Projetos	μ_D	σ_D	*IC*	C_V	M_i	Qtd Abertas[4]	Qtd Resolvidas[5]	Qtd Extra[6]	Total[7]	Modelo	% erro
Projeto 1	2,28	4,98	1.987899 < mean <2.572101	2,18	0,37	1119	875	24	899		
Projeto 2	3,95	7,02	2.653207 < mean <5.246793	1,78	0,49	115	107	19	126		
Projeto 3	4,41	7,06	3.328079 < mean <5.491921	1,60	0,57	166	110	229	339	1719,99	-16%
Projeto 4	2,99	5,8	1.904006 < mean <4.075994	1,94	0,53	112	103	12	115		
							Total dos Projetos		1479		

A Tabela 7 apresenta o resumo estatístico dos projetos. Os valores estatísticos referentes à amostra das solicitações de mudanças abertas no projeto 1, projeto 2, projeto 3 e projeto 4 foram adquiridos através de ferramentas estatísticas. As colunas μ_D, σ_D, IC, C_V e M_i referem-se à média, desvio padrão, intervalo de confiança, coeficiente de variação e

[4] Quantidade de solicitações que foram abertas no período.
[5] Quantidade de solicitações que foram abertas no período e foram resolvidas no período.
[6] Quantidade de solicitações que foram abertas antes do período e foram resolvidas no período.
[7] Soma da quantidade de solicitações abertas e resolvidas no período e a da quantidade de solicitações que foram abertas antes e resolvidas no período.

mediana das solicitações de mudança abertas em cada projeto. Os valores exibidos nas colunas qtd abertas, qtd resolvidas, qtd extra e total foram obtido através do sistema e referem também as solicitações de mudança abertas em cada projeto. A coluna qtd abertas refere-se à quantidade de solicitações que foram abertas no período, a coluna qtd resolvidas contém o valor da quantidade de solicitações que foram abertas e resolvidas no período e a qtd extra contém o valor referente à quantidade de solicitações que foram abertas antes do período analisado e foram resolvidas no período. A coluna total contém a soma da quantidade de solicitações abertas e resolvidas no período e a da quantidade de solicitações que foram abertas antes e resolvidas no período. O total de cada projeto é somado e obtém-se o total dos projetos, evidenciado no campo total dos projetos e comparado a coluna modelo. O valor da coluna modelo foi calculado através da equação (P{#P0>=1}*(1/TA))*T através do modelo refinado, usado para comparar com os valores do sistema. Essa comparação gera um percentual de erro de -16%, evidenciado na coluna % Erro.

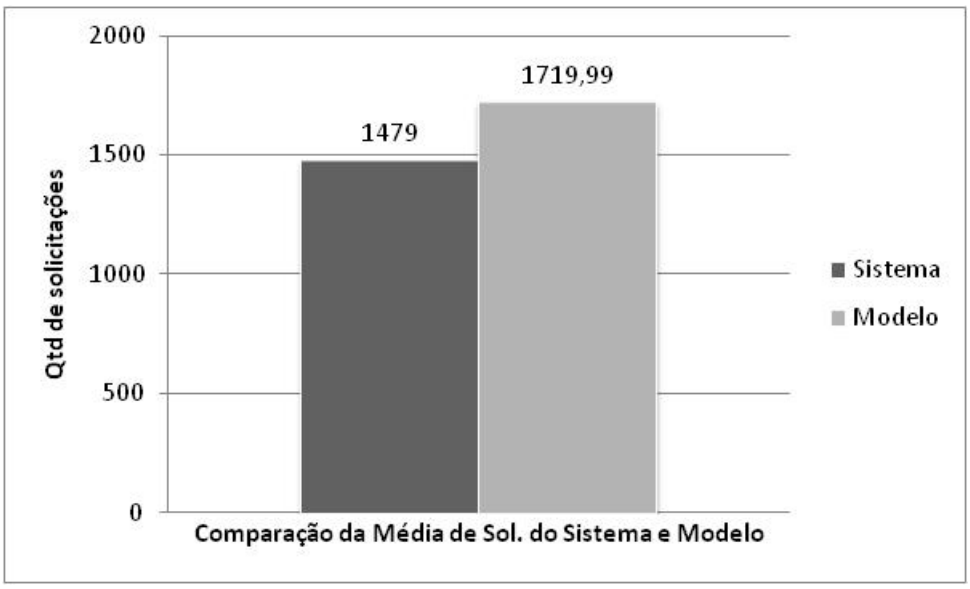


Figura 26: Comparação da Média de Solicitações de Mudança Sistema e Modelo

5.4 Considerações Finais

O capítulo apresentado mostrou os modelos propostos para modelagem do processo de mudança de software. Foram apresentados os modelos: solicitação, buffer, CCB, Mudança Finalizada, *Baseline* e suas respectivas métricas, e a respectiva validação do modelo.

CAPÍTULO 6 - ESTUDO DE CASO

Este capítulo apresenta os resultados da avaliação de desempenho de um estudo de caso, relativos a três cenários específicos de um projeto que ocorreu muitas mudanças no decorrer do seu desenvolvimento. Estes resultados são informações importantes ao planejamento de processos de mudança de software de uma empresa de desenvolvimento de software (ver Figura 16) e modelado em redes de Petri (ver Figura 25).

6.1 Cenário 1

O primeiro cenário descreve o projeto, cujo tamanho é de 1871 Pontos de Função e utilizou ciclo de vida cascata no desenvolvimento do sistema. No período analisado o projeto estava na fase de testes. A equipe de desenvolvimento deste projeto é composta de 9 (nove) engenheiros de software (E), 1 (um) arquiteto (A), 2 (dois) analistas de sistemas (AS) e 1 (um) gerente de projeto (G). Neste projeto o período planejado para a fase de especificação dos requisitos foi curto, ou seja, o período planejado não foi suficiente para realização desta atividade e na validação das especificações dos requisitos não houve o envolvimento dos *stakeholders*[8] relevantes. Em levantamento de lições aprendidas deste projeto, a equipe de desenvolvimento destaca este o principal motivo da grande quantidade de solicitações de mudança abertas. O objetivo da avaliação deste cenário

[8] **Stakeholder** (em português, **parte interessada**), é um termo usado em administração que refere-se a qualquer pessoa ou entidade que afeta ou é afetada pelas atividades do projeto.

foi analisar o custo que o projeto teve com o CCB ao realizar a análise de impacto de solicitações de mudança abertas de severidade alta[9].

No período de 6 (seis) meses houve um total de 1119 (mil cento e dezenove) solicitações de mudança abertas para código, incluindo defeitos e melhorias/mudanças de requisitos. Identificou-se que 875 (oitocentos e setenta e cinco) solicitações de mudança foram abertas e resolvidas no período. Entretanto, 332 (trezentos e trinta e duas) destas 875 (oitocentos e setenta e cinco) solicitações de mudança foram de severidade alta. Estas solicitações de mudança de severidade alta foram abertas em um intervalo médio de 4,60hs.

Cada papel como o gerente do projeto, o analista de sistemas, arquiteto e o engenheiro de software tiveram participação na análise de impacto dessas solicitações de mudança de severidade alta, compondo o CCB.

Entre as 332 (trezentos e trinta e duas) solicitações de mudança analisadas pelo CCB, houve um esforço médio de 0,61 horas por solicitação de mudança.

Das 875 (oitocentos e setenta e cinco) abertas e resolvidas no período, 38% foram de severidade alta. Um percentual alto para solicitações de mudança dessa classe para este projeto. Esse percentual elevado pode ter sido causado pela utilização de um ciclo de vida cascata, pois todos os requisitos foram validados ao final da fase de requisitos, podendo ter passado detalhes despercebidos pelos seus *stakeholders*. Dois outros motivos são o intervalo reduzido de tempo, que foi planejado pelo gerente do projeto, para especificação dos requisitos e a não participação dos *stakeholders* relevantes (pessoas que conheçam do negócio e com poder de decisão) na validação da mesma.

[9] Problema que impeça a continuidade dos trabalhos sobre a aplicação até que seja resolvido ou ainda um problema que impossibilite a utilização ou testes da aplicação. Esse tipo de mudança exige ação corretiva imediata. Ou ainda, um problema sério que produza a perda intermitente das funcionalidades ou degrade o seu desempenho ou ainda que reflita a ausência de um requisito essencial. Uma mudança dessa gravidade exige ação corretiva tão logo seja possível. Ex: não gravação da informação na base de dados, execução de ações de fluxos principais e subfluxos sem possibilidade de concretização, execução de ações de fluxos principais e subfluxos com falhas, execução de ações de fluxos alternativos ou de exceção sem possibilidade de concretização, regras de negócio com erros, gravação de informações (base de dados) incompletas.

A Tabela 8 exibe os dados do projeto verificado no cenário 1 referentes ao CCB utilizado em um conjunto de solicitações de mudança. A coluna Qtd sol. exibe a quantidade de solicitações de mudança que utilizou a formação do CCB exibida na coluna CCB. O valor/hora dessa formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 91% do CCB obtida através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 91%. O custo total do CCB é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 15.836,26 (ver Tabela 8). Com o intuito de melhor planejar a formação do CCB e reduzir os custos do mesmo, sugerem-se algumas alterações.

Tabela 8: Dados do CCB - Cenário 1

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
108	2 E	R$ 46,26	R$ 42,10	R$ 4.546,43
8	2 E + 1 A	R$ 90,09	R$ 81,98	R$ 655,86
20	1 E + 1 A	R$ 66,96	R$ 60,93	R$ 1.218,67
12	2 E + 1 AS	R$ 89,78	R$ 81,70	R$ 980,40
6	1 A + 1 E + 1 AS	R$ 110,48	R$ 100,54	R$ 603,22
1	2 E + 1 G	R$ 114,47	R$ 104,17	R$ 104,17
5	1 A + 1 AS	R$ 87,35	R$ 79,49	R$ 397,44
66	1 E + 1 AS	R$ 66,65	R$ 60,65	R$ 4.003,00
44	1 A	R$ 43,83	R$ 39,89	R$ 1.754,95
58	1 E	R$ 23,13	R$ 21,05	R$ 1.220,80
1	1 A + 1 G	R$ 112,04	R$ 101,96	R$ 101,96
3	1 E + 1 G	R$ 91,34	R$ 83,12	R$ 249,36
		Total	**857,57**	**15.836,26**

Com o intuito de planejar uma melhor alocação para a formação do CCB para análise de impacto das solicitações de mudança, sugere-se a

formação do mesmo baseado na análise dos dados de outros projetos dentro da organização em que os mesmo obtiveram um custo conforme planejado com relação a alocação dos papéis utilizados na formação do comitê de controle de mudança (CCB).

Para as 108 solicitações de mudanças, sugere-se que na formação do CCB o gerente do projeto envolva apenas 1 arquiteto (A) ao invés de 2 engenheiros de software (E). Com essa alteração economizou-se, pois o valor hora de 2 engenheiros de software (E) é de R$ 46,26 e o de um arquiteto (A) é de R$ 43,83.

Para as 8 solicitações de mudança, sugere-se que na formação do CCB envolva apenas 1 arquiteto (A) ao invés de 2 engenheiros de software (E) e 1 arquiteto (A). Com essa alteração o gerente do projeto economizou, pois o valor hora de 2 engenheiros de software (E) e 1 arquiteto (A) é de R$ 90,09 e o de um arquiteto (A) é de R$ 43,83. A mesma alteração para as 20 solicitações de mudança que envolve 1 engenheiro de software (E) e 1 arquiteto (A) na formação do CCB. Com essa alteração, o custo de R$ 66,96 com 1 engenheiro de software (E) e 1 arquiteto (A) reduz para R$ 43,83.

Tabela 9: Dados do CCB Planejados – Cenário 1

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
108	1 A	R$ 43,83	R$ 18,85	R$ 2.035,47
8	1 A	R$ 43,83	R$ 18,85	R$ 150,78
20	1 A	R$ 43,83	R$ 18,85	R$ 376,94
12	1 A + 1 AS	R$ 87,35	R$ 37,56	R$ 450,73
6	1 A + 1 AS	R$ 87,35	R$ 37,56	R$ 225,36
1	1 A + 1 G	R$ 112,04	R$ 48,18	R$ 48,18
5	1 A + 1 AS	R$ 87,35	R$ 79,49	R$ 397,44
66	1 E + 1 AS	R$ 66,65	R$ 60,65	R$ 4.003,00
44	1 A	R$ 43,83	R$ 39,89	R$ 1.754,95
58	1 E	R$ 23,13	R$ 21,05	R$ 1.220,80
1	1 A + 1 G	R$ 112,04	R$ 101,96	R$ 101,96

3	1 E + 1 G	R$ 91,34	R$ 83,12	R$ 249,36
		Total	**565,99**	**11.014,96**

Para as 12 solicitações de mudança que envolve na formação do CCB 2 engenheiros de software (E) e 1 analista (AS) e para as 6 solicitações de mudança que envolve 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS), sugere-se que o gerente do projeto envolva nesta formação apenas 1 arquiteto (A) e 1 analista (AS). Com essa alteração, o custo de R$ 89,78 de 2 engenheiros de software (E) e 1 analista (AS) reduz para R$ 87,35. E o custo de R$ 110,48 de 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS) também reduz para R$ 87,35.

E por fim, para a única solicitação de mudança, sugere-se que envolva na formação do CCB apenas 1 arquiteto (A) e 1 gerente de projeto (G) ao invés de 2 engenheiros de software (E) e 1 gerente de projeto (G). Com essa alteração o gerente do projeto obtém uma economia, pois o valor de 2 engenheiros de software (E) e 1 gerente de projeto (G) é de R$ 114,47 e o de 1 arquiteto (A) e 1 gerente de projeto (G) é de R$ 112,04.

No geral, com essas alterações, o projeto teria um custo de apenas R$ 11.014,96, economizando R$4.821,30.

A Tabela 9 exibe os dados do projeto com as alterações sugeridas no cenário 1 referentes ao CCB utilizado em um conjunto de solicitações de mudança. Com as alterações sugeridas o valor/hora dessa nova formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 43% obtida através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 43%. O custo total do CCB também foi alterado, pois é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo novo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 11.014,96 (ver Tabela 9).

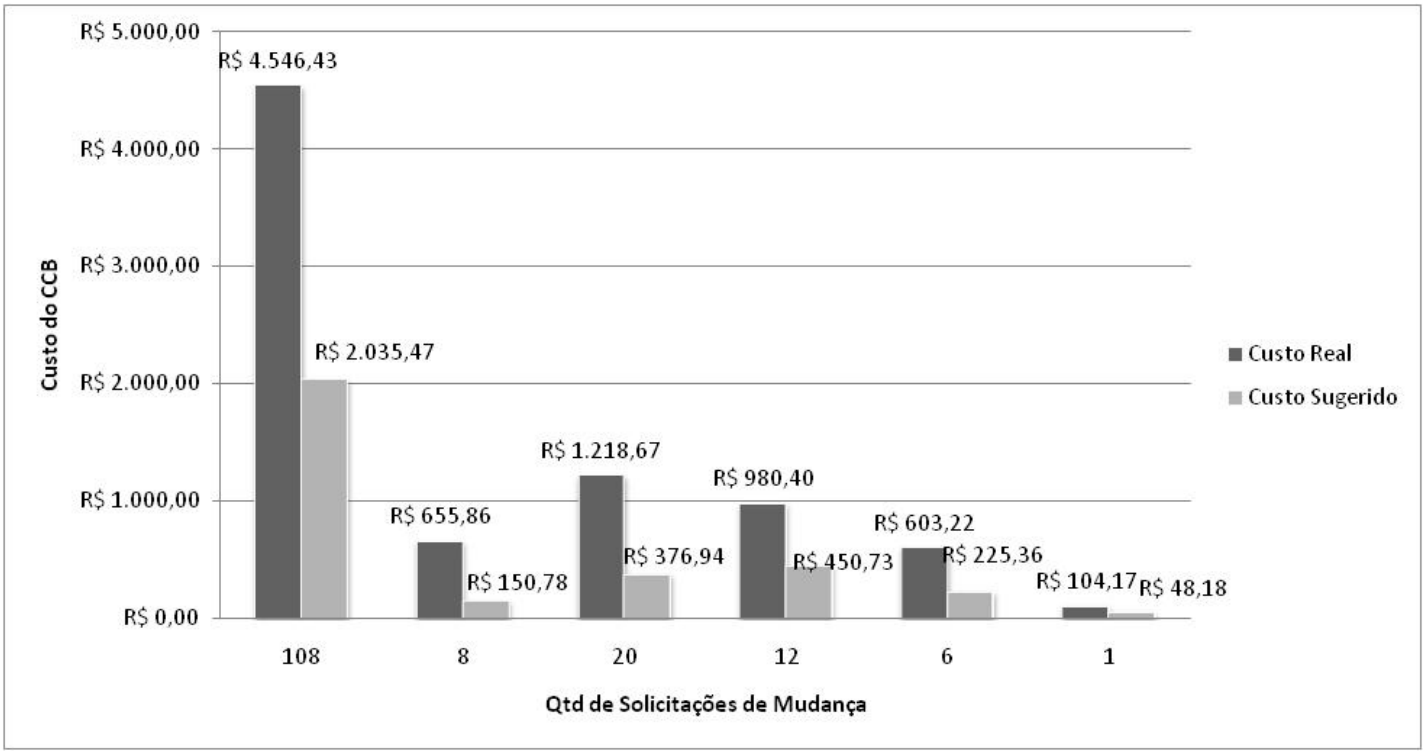


Figura 27: Custo Real X Custo Sugerido – Cenário 1

Observa-se na Figura 27 que com as alterações sugeridas nos papéis envolvidos na formação do CCB, o gerente do projeto obteve uma redução do custo, acarretando uma economia total de R$4.821,30.

A avaliação deste cenário constatou que para o planejamento do processo de mudança de software é importante verificar quais papéis da equipe da equipe de desenvolvimento formarão o CCB para execução da análise de impacto das solicitações de mudança de software de severidade alta. Ainda na avaliação, constatou-se que como o papel arquiteto possui atribuições de implementação do código complexas, tem uma formação técnica mais abrangente e uma maior experiência, o mesmo possui competência e condições técnicas de analisar a solicitação de mudança, não havendo a necessidade de envolver outro engenheiro de software nas solicitações de mudança que ele participa na formação do CCB. Com isso, existe uma redução dos custos desta atividade no processo de mudança de software, conseqüentemente no projeto.

Em outros três projetos de desenvolvimento de software que utilizaram o ciclo de vida iterativo e incremental, observou-se, que no planejamento da formação do CCB, os gerentes dos projetos utilizaram a mesma formação do CCB sugerida neste cenário 1, sendo o custo de utilização do CCB conforme planejado.

6.2 Cenário 2

O segundo cenário descreve o projeto cuja equipe é composta por 13 recursos pessoais, sejam 9 (nove) engenheiros de software (E), 1 (um) arquiteto (A), 2 (dois) analistas de sistemas (AS) e 1 (um) gerente de projeto (G). O tamanho deste projeto é de 1871 Pontos de Função e utilizou ciclo de vida cascata no desenvolvimento do sistema. No período analisado o projeto se encontrava na fase de testes.

O período planejado para a fase de especificação dos requisitos, deste projeto, foi curto, ou seja, o período planejado não foi suficiente para realização desta atividade e não houve na validação dessas especificações dos requisitos o envolvimento dos *stakeholders* relevantes. A equipe de desenvolvimento, em levantamento de lições aprendidas deste projeto, destaca estes os principais motivos da grande quantidade de solicitações de mudança abertas. O objetivo da avaliação deste cenário foi analisar o custo que o projeto teve com o CCB ao realizar a análise de impacto de solicitações de mudança abertas de severidade baixa[10].

Um total de 1119 (mil cento e dezenove) solicitações de mudança foram abertas para código, no período de 6 (seis) meses, incluindo defeitos e melhorias/mudanças de requisitos. Identificou-se que 875 (oitocentos e setenta e cinco) solicitações de mudança foram abertas e resolvidas no período. Entretanto, 542 (quinhentos e quarenta e duas) destas 875

[10] Problema menor que não impeça o usuário de realizar as funções desejadas e transparentes para o cliente. A correção destes problemas pode ser adiada para o próximo release sem justificação formal. Geralmente problemas com características de interface visual. Ex: execução de ações de fluxos alternativos ou de exceção com falhas, mas que não comprometem a concretização das operações, erros de validação nas entradas de dados, erros no fluxo de navegação de telas, erros de ordenação e na paginação das informações, problemas de alinhamentos e distorção de informação, campos aceitando caracteres inadequados, campos com tamanhos inadequados, erros na ação do botão voltar, caminho de navegação ou código de tela errados, erros em máscaras de edição e textos com erros de português, inclusive mensagens.

(oitocentos e setenta e cinco) solicitações de mudança foram de severidade baixa e foram abertas em um intervalo médio de 3,12hs.

Cada papel como o gerente do projeto, o analista de sistemas, arquiteto e o engenheiro de software tiveram participação na análise de impacto dessas solicitações de mudança de severidade baixa, compondo o CCB.

Entre as 542 (quinhentos e quarenta e duas) solicitações de mudança analisadas pelo CCB, houve um esforço médio de 0,61 horas por solicitação de mudança. Estas 542 (quinhentos e quarenta e duas) solicitações de mudança de severidade baixa equivalem 62% das 875 (oitocentos e setenta e cinco) abertas e resolvidas no período. Um percentual aceitável para solicitações de mudança dessa classe. Mas apesar do percentual razoável, 542 (quinhentos e quarenta e duas) solicitações é uma grande quantidade de mudanças. Assim como no Cenário 1, essa quantidade de mudanças de severidade baixa pode ter sido causado pela utilização de um ciclo de vida cascata, pois todos os requisitos foram validados ao final da fase de requisitos, podendo ter passado detalhes despercebidos pelos seus *stakeholders*. Dois outros motivos são o intervalo reduzido de tempo, que foi planejado pelo gerente do projeto, para especificação dos requisitos e a não participação dos *stakeholders* relevantes (pessoas que conheçam do negócio e com poder de decisão) na validação da mesma.

Tabela 10: Dados do CCB - Cenário 2

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
155	2 E	R$ 46,26	R$ 45,80	R$ 7.098,60
4	2 E + 1 A	R$ 90,09	R$ 89,19	R$ 356,76
22	1 E + 1 A	R$ 66,96	R$ 66,29	R$ 1.458,39
10	2 E + 1 AS	R$ 89,78	R$ 88,88	R$ 888,82
10	1 A + 1 E + 1 AS	R$ 110,48	R$ 109,38	R$ 1.093,75

10	1 A + 1 AS	R$ 87,35	R$ 86,48	R$ 864,77
78	1 E + 1 AS	R$ 66,65	R$ 65,98	R$ 5.146,71
29	1 A	R$ 43,83	R$ 43,39	R$ 1.258,36
222	1 E	R$ 23,13	R$ 22,90	R$ 5.083,51
1	1 E + 1 AS + 1 G	R$ 134,86	R$ 133,51	R$ 133,51
1	1 E + 1 G	R$ 91,34	R$ 90,43	R$ 90,43
		Total	**842,22**	**23.473,60**

A Tabela 10 exibe os dados do projeto verificado no cenário 2 referentes ao CCB utilizado em um conjunto de solicitações de mudança. A coluna Qtd sol. exibe a quantidade de solicitações de mudança que utilizou a formação do CCB exibida na coluna CCB. O valor/hora dessa formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 99% obtida através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 99%.

O custo total do CCB é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 21.576,75 (ver Tabela 10). Com o intuito de melhor planejar a formação do CCB e reduzir os custos do mesmo, sugerem-se algumas alterações.

Com o intuito de melhor planejar a formação do CCB para a análise de impacto das solicitações de mudança e reduzir os custos do mesmo, sugerem-se algumas alterações. Estas sugestões são baseadas na análise dos dados de outros projetos dentro da organização em que os mesmos obtiveram um custo conforme planejado com relação a alocação dos papéis utilizados na formação do comitê de controle de mudança (CCB).

Com o intuito de melhor planejar a formação do CCB para a análise de impacto das solicitações de mudança e reduzir os custos do mesmo,

sugerem-se algumas alterações. Para o CCB formado por 2 engenheiros de software (E), sugere-se alterar esta formação para apenas 1 arquiteto (A). Com essa alteração, aplicada nas 155 solicitações de mudança (ver Tabela 10), o gerente do projeto garante uma economia no projeto, pois o valor hora de 2 engenheiros de software (E) é de R$ 46,26 e o de um arquiteto (A) é de R$ 43,83.

Tabela 11: Dados do CCB Planejados – Cenário 2

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
155	1 A	R$ 43,83	R$ 14,03	R$ 2.173,97
4	1 A	R$ 43,83	R$ 14,03	R$ 56,10
22	1 A	R$ 43,83	R$ 14,03	R$ 308,56
10	1 A + 1 AS	R$ 87,35	R$ 27,95	R$ 279,52
10	1 A + 1 AS	R$ 87,35	R$ 27,95	R$ 279,52
10	1 A + 1 AS	R$ 87,35	R$ 27,95	R$ 279,52
78	1 E + 1 AS	R$ 66,65	R$ 21,33	R$ 1.663,58
29	1 A	R$ 43,83	R$ 14,03	R$ 406,74
222	1 E	R$ 23,13	R$ 7,40	R$ 1.643,16
1	1 E + 1 AS + 1 G	R$ 134,86	R$ 43,16	R$ 43,16
1	1 E + 1 G	R$ 91,34	R$ 29,23	R$ 29,23
		Total	**241,07**	**7.163,06**

Para as 4 solicitações de mudança, sugere-se que envolva na formação do CCB apenas 1 arquiteto (A) ao invés de 2 engenheiros de software (E) e 1 arquiteto (A). Com essa alteração o gestor do projeto verifica uma economia, pois o valor hora de 2 engenheiros de software (E) e 1 arquiteto (A) é de R$ 90,09 e o de um arquiteto é de R$ 43,83. A mesma alteração para as 22 solicitações de mudança que envolve 1 engenheiro de software (E) e 1 arquiteto (A) na formação do CCB. Com essa alteração, o custo de R$ 66,96 com 1 engenheiro de software (E) e 1 arquiteto (A) reduz para R$ 43,83.

Para as 10 solicitações de mudança que envolve na formação do CCB 2 engenheiros de software (E) e 1 analista (AS) e para as 10 solicitações de mudança que envolve 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS), sugere-se que o gestor do projeto planeje envolver na formação do CCB apenas 1 arquiteto (A) e 1 analista (AS). Com essa alteração, o custo de R$ 89,78 de 2 engenheiros de software (E) e 1 analista (AS) reduz para R$ 87,35. E o custo de R$ 110,48 de 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS) também reduz para R$ 87,35.

No geral, com essas alterações, o projeto teria um custo de apenas R$ 7.163,06, economizando R$14.413,69.

A Tabela 11 exibe os dados do projeto com as alterações sugeridas no cenário 2 referentes ao CCB utilizado em um conjunto de solicitações de mudança. Com as alterações sugeridas o valor/hora dessa nova formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 32% obtido através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 32%.

O custo total do CCB também foi alterado, pois é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo novo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 7.163,06 (ver Tabela 11).

Observa-se na Figura 28 que com as alterações sugeridas nos papéis envolvidos na formação do CCB, o gerente do projeto obteve uma redução do custo, acarretando uma economia total de R$16.310,54.

A avaliação deste cenário pode se verificar uma redução dos custos da atividade de análise de impacto realizada pelo CCB no processo de mudança de software, conseqüentemente no projeto. Para o planejamento do processo de mudança de software, pode se constatar neste segundo cenário, que é importante verificar quais papéis da equipe de

desenvolvimento formará o CCB para execução da análise de impacto das solicitações de mudança de software de severidade baixa. Constatou-se, ainda na avaliação, que como o papel arquiteto possui atribuições de implementação do código complexas, tem uma formação técnica mais abrangente e uma maior experiência, o mesmo possui competência e condições técnicas de analisar a solicitação de mudança, não havendo a necessidade de envolver outro engenheiro de software nas solicitações de mudança que ele participa na formação do CCB.

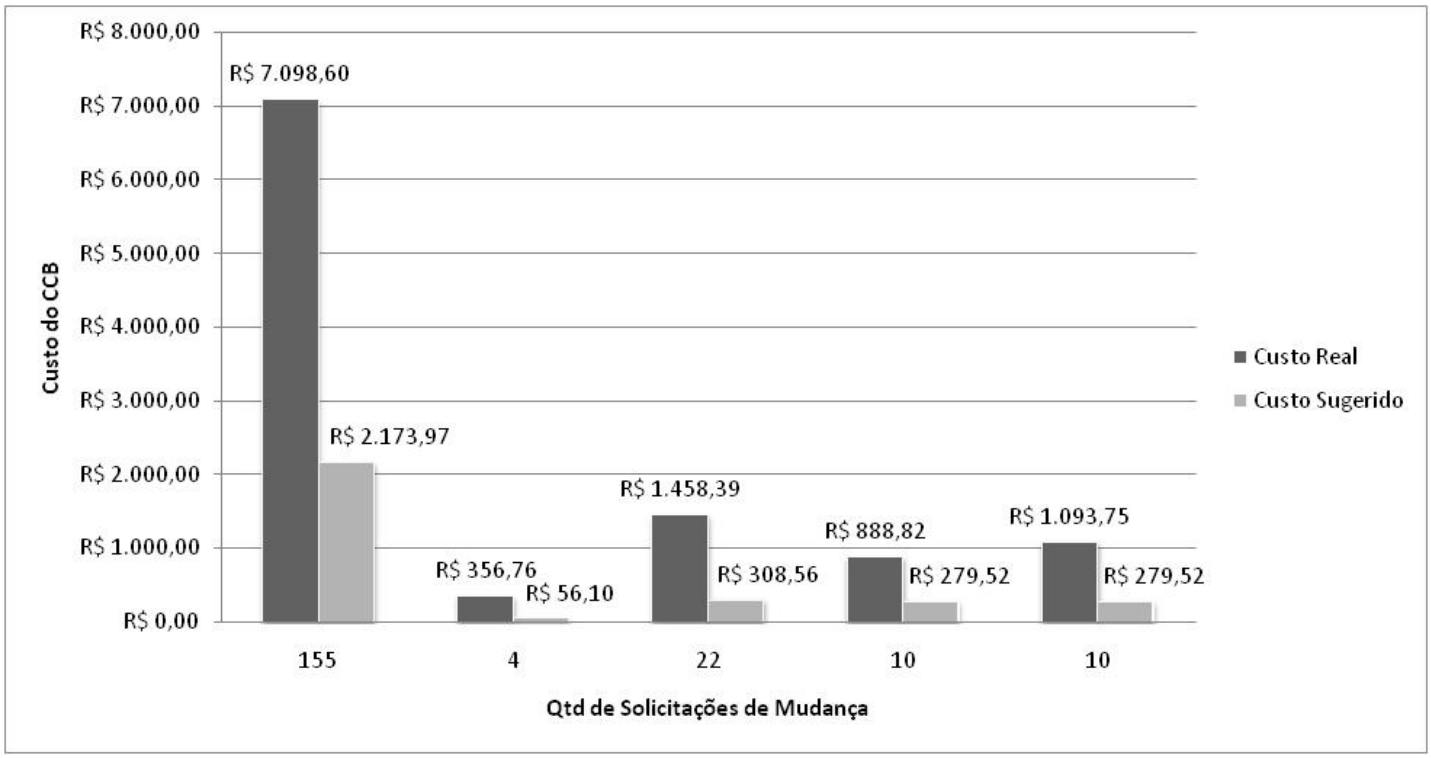


Figura 28: Custo Real X Custo Sugerido – Cenário 2

6.3 Cenário 3

O terceiro cenário descreve o projeto cujo tamanho é de 1871 Pontos de Função e utilizou ciclo de vida cascata no desenvolvimento do sistema. No período analisado o projeto estava na fase de testes. A equipe de desenvolvimento deste projeto é composta de 9 (nove) engenheiros de software (E), 1 (um) arquiteto (A), 2 (dois) analistas de sistemas (AS) e 1 (um) gerente de projeto (G). Neste projeto o período planejado para a fase de especificação dos requisitos foi curto e na validação das especificações

dos requisitos não houve o envolvimento dos *stakeholders* relevantes. Em levantamento de lições aprendidas deste projeto, a equipe de desenvolvimento destaca este o principal motivo da grande quantidade de solicitações de mudança abertas. O objetivo da avaliação deste cenário foi analisar as despesas que o gestor do projeto obteve com a formação do CCB ao realizar a análise de impacto de solicitações de mudança abertas de severidade alta.

Um total de 1119 (mil cento e dezenove) solicitações de mudança abertas para código, no período de 6 (seis) meses, incluindo defeitos e melhorias/mudanças de requisitos. Identificou-se que 875 (oitocentos e setenta e cinco) solicitações de mudança foram abertas e resolvidas no período. Entretanto, 332 (trezentos e trinta e duas) destas 875 (oitocentos e setenta e cinco) solicitações de mudança foram de severidade alta e foram abertas em um intervalo médio de 4,60hs.

Cada papel como o gerente do projeto, o analista de sistemas, arquiteto e o engenheiro de software tiveram participação na análise de impacto dessas solicitações de mudança de severidade alta, compondo o CCB. Entre as 332 (trezentos e trinta e duas) solicitações de mudança analisadas pelo CCB, houve um esforço médio de 0,61 horas por solicitação de mudança.

Um percentual de 38% das 875 (oitocentos e setenta e cinco) solicitações de mudanças abertas e resolvidas no período foi de severidade alta. Um percentual alto para solicitações de mudança dessa classe. Esse percentual elevado pode ter sido causado por alguns motivos relevantes como a validação dos requisitos apenas ao final da fase de requisitos, podendo ter passado detalhes despercebidos pelos seus *stakeholders*, por conta da utilização de um ciclo de vida cascata, o intervalo reduzido de tempo, que foi planejado pelo gerente do projeto, para especificação dos requisitos e a não participação dos *stakeholders* relevantes (pessoas que conheçam do negócio e com poder de decisão) na validação da mesma.

Os dados do projeto verificado no cenário 3 referentes ao CCB utilizado em um conjunto de solicitações de mudança é verificado na Tabela

12. A coluna Qtd sol. exibe a quantidade de solicitações de mudança que utilizou a formação do CCB exibida na coluna CCB. O valor/hora dessa formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 91% obtida através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 91%. O custo total do CCB é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 15.836,26 (ver Tabela 12). Com o intuito de melhor planejar a formação do CCB e reduzir os custos do mesmo, sugerem-se algumas alterações.

Tabela 12: Dados do CCB - Cenário 3

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
108	2 E	R$ 46,26	R$ 42,10	R$ 4.546,43
8	2 E + 1 A	R$ 90,09	R$ 81,98	R$ 655,86
20	1 E + 1 A	R$ 66,96	R$ 60,93	R$ 1.218,67
12	2 E + 1 AS	R$ 89,78	R$ 81,70	R$ 980,40
6	1 A + 1 E + 1 AS	R$ 110,48	R$ 100,54	R$ 603,22
1	2 E + 1 G	R$ 114,47	R$ 104,17	R$ 104,17
5	1 A + 1 AS	R$ 87,35	R$ 79,49	R$ 397,44
66	1 E + 1 AS	R$ 66,65	R$ 60,65	R$ 4.003,00
44	1 A	R$ 43,83	R$ 39,89	R$ 1.754,95
58	1 E	R$ 23,13	R$ 21,05	R$ 1.220,80
1	1 A + 1 G	R$ 112,04	R$ 101,96	R$ 101,96
3	1 E + 1 G	R$ 91,34	R$ 83,12	R$ 249,36
		Total	**857,57**	**15.836,26**

Se essas 332 solicitações de severidade alta fossem de severidade baixa, a combinação dos papéis na formação do CCB seria diferenciada e o projeto teria uma redução nos custos. Com o intuito de melhor planejar a

execução do CCB e economizar os custos do mesmo, sugerem-se algumas alterações.

Com o intuito de planejar uma melhor alocação para a formação do CCB para análise de impacto das solicitações de mudança, sugere-se a formação do mesmo baseado na análise dos dados de outros projetos dentro da organização em que os mesmo obtiveram um custo conforme planejado com relação a alocação dos papéis utilizados na formação do comitê de controle de mudança (CCB).

Para as 108 solicitações de mudança, sugere-se que envolva na formação do CCB apenas 1 engenheiro de software (E) mais experiente ao invés de 2 engenheiros de software (E). Existe, com essa alteração, uma redução de custo, pois o valor de 2 engenheiros de software (E) é de R$ 46,26 e o de um engenheiro de software (E) é de R$ 23,13.

Tabela 13: Dados do CCB Planejados – Cenário 3

Qtd Sol.	CCB	Valor/Hora	Custo CCB/Sol.	Custo Total CCB
108	1 E	R$ 23,13	R$ 9,95	R$ 1.074,16
8	1 A	R$ 43,83	R$ 18,85	R$ 150,78
20	1 A	R$ 43,83	R$ 18,85	R$ 376,94
12	1 E+ 1 AS	R$ 66,65	R$ 28,66	R$ 343,91
6	1 A+ 1 AS	R$ 87,35	R$ 37,56	R$ 225,36
1	1 E + 1 G	R$ 91,34	R$ 39,28	R$ 39,28
5	1 A + 1 AS	R$ 87,35	R$ 79,49	R$ 397,44
66	1 E + 1 AS	R$ 66,65	R$ 60,65	R$ 4.003,00
44	1 A	R$ 43,83	R$ 39,89	R$ 1.754,95
58	1 E	R$ 23,13	R$ 21,05	R$ 1.220,80
1	1 A+ 1 G	R$ 112,04	R$ 101,96	R$ 101,96
3	1 E + 1 G	R$ 91,34	R$ 83,12	R$ 249,36
		Total	**R$ 539,29**	**R$ 9.937,93**

Envolver apenas 1 arquiteto (A) ao invés de 2 engenheiros de software (E) e 1 arquiteto (A) nas 8 solicitações de mudança. Com essa alteração há uma redução de custos na formação do CCB, pois o valor hora de 2 engenheiros de software (E) e 1 arquiteto (A) é de R$ 90,09 e o de um arquiteto (A) é de R$ 43,83. A mesma alteração para as 20 solicitações de mudança que envolve 1 engenheiro de software (E) e 1 arquiteto (A) na formação do CCB. Com essa alteração, o custo de R$ 66,96 com 1 engenheiro de software (E) e 1 arquiteto (A) reduz para R$ 43,83.

Sugere-se envolver na formação do CCB o engenheiro de software mais experiente (E) e 1 analista (AS) nas 12 solicitações de mudança que compõe seu CCB de 2 engenheiros de software (E) e 1 analista (AS). Com essa alteração, o custo de R$ 89,78 de 2 engenheiros de software (E) e 1 analista (AS) reduz para R$ 66,65.

E para as 6 solicitações de mudança que envolvem na formação do CCB 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS), sugere-se envolver apenas 1 arquiteto (A) e 1 analista (AS). Com essa alteração, o custo de R$ 110,48 de 1 arquiteto (A), 1 engenheiro de software (E) e 1 analista (AS) também reduz para R$ 87,35.

E por fim, para a única solicitação de mudança, sugere-se que envolva apenas o engenheiro de software (E) mais experiente e 1 gerente de projeto (G) ao invés de 2 engenheiros de software (E) e 1 gerente de projeto (G). Com essa alteração existe uma economia, pois o valor de 2 engenheiros de software (E) e 1 gerente de projeto (G) é de R$ 114,47 e o de 1 engenheiro de software (E) mais experiente e 1 gerente de projeto (G) é de R$ 91,34.

No geral, com essas alterações, o projeto teria um custo de apenas R$ 9.937,93, economizando R$5.898,32.

A Tabela 13 exibe os dados do projeto com as alterações sugeridas no cenário 3 referentes ao CCB utilizado em um conjunto de solicitações de mudança. Com as alterações sugeridas o valor/hora dessa nova formação está representado na coluna Valor/Hora. A coluna Custo CCB/Sol. exibe o

custo que o projeto possui com a formação do CCB baseando-se no percentual de utilização de 43% obtida através do modelo no período analisado, ou seja, o valor/hora do CCB multiplicado pelo percentual de 43%. O custo total do CCB também foi alterado, pois é baseado na quantidade de solicitações (Qtd sol.) multiplicado pelo novo custo do CCB por solicitação (Custo CCB/Sol.). A análise de impacto teve um custo de no total R$ 9.937,93 (ver Tabela 13).

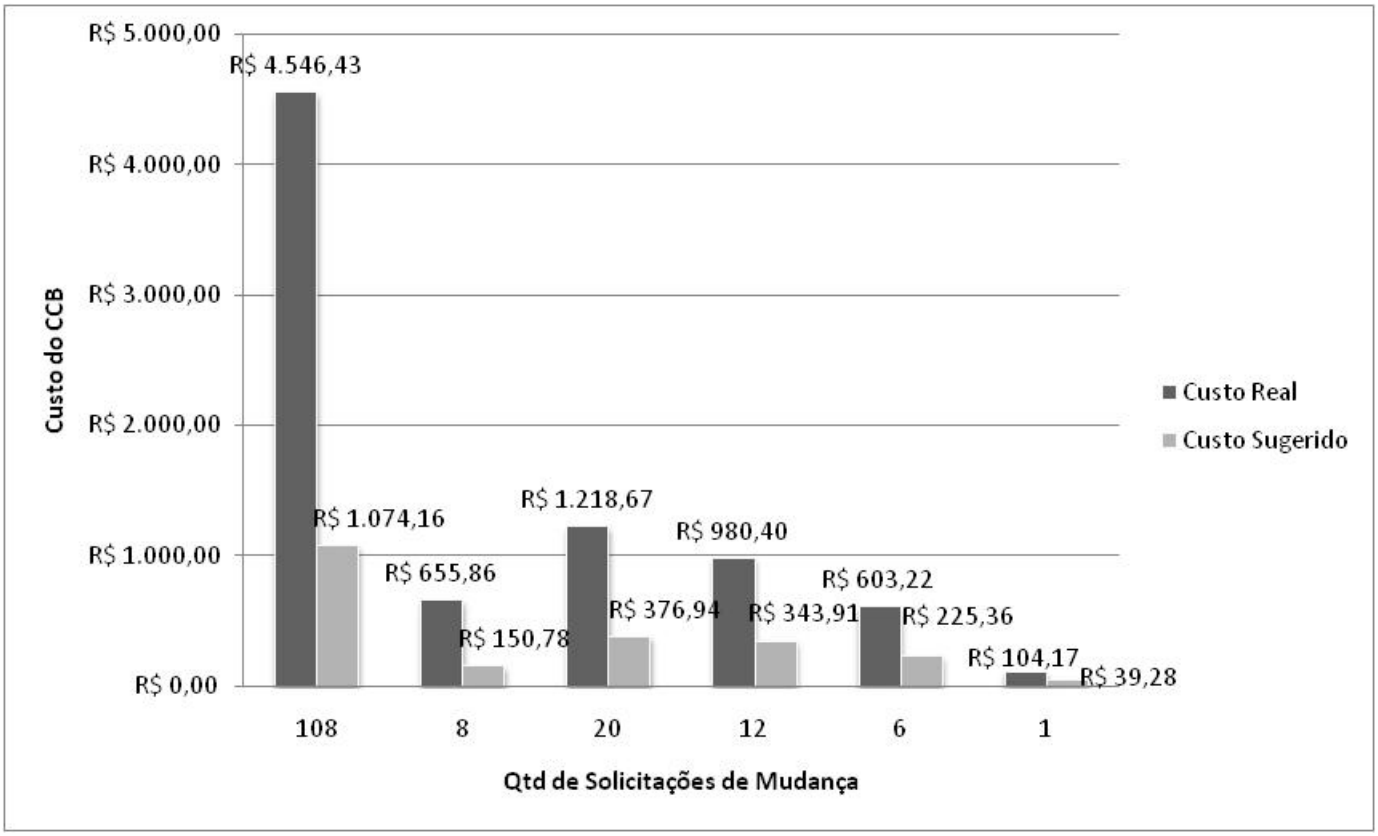


Figura 29: Custo Real X Custo Sugerido – Cenário 3

Observa-se na Figura 29 que com as alterações sugeridas nos papéis envolvidos na formação do CCB, o gerente do projeto obteve uma redução do custo, acarretando uma economia total de R$5.898,32.

A avaliação deste cenário constatou que a equipe de desenvolvimento de software ao desenvolver seus artefatos, seja código ou documentação, deve ter muita atenção para evitar a geração de mudanças de severidade alta. Outra constatação importante, é que gerente do projeto, ao planejar a equipe do CCB, deve ter muita atenção ao selecionar

os papéis envolvidos, pois o envolvimento de papéis com o custo elevado nas mudanças de severidade baixa gera um custo alto ao projeto. Ainda na avaliação, constatou-se, que nas solicitações de mudança de severidade baixa, o gerente do projeto deve envolver na formação do CCB o engenheiro de software mais experiente no lugar de 2 engenheiros de software na atividade de análise de impacto feita pelo CCB. Com isso, existe uma redução dos custos desta atividade no processo de mudança de software, conseqüentemente no projeto.

6.4 Considerações Finais

Este capítulo apresentou os resultados obtidos na realização do estudo de caso. Através do estudo aqui apresentado, foi possível avaliar cenários diversos interessantes para o modelo SPN proposto por este trabalho. Com os resultados dos experimentos, foi possível avaliar o processo de mudança de software, mas especificamente a atividade de análise de impacto realizada pelo CCB, em termos de utilização e custo através dos cenários propostos, dando suporte ao planejamento do processo de mudança de software e tomada de decisão.

CONCLUSÃO

As solicitações de mudanças ocorreram no software e elas precisam ser registradas, seu impacto analisado, ser implementadas e incluída em uma versão estável do software. Esse procedimento gera custos para o projeto, logo a execução do mesmo precisa ser planejada para gerar o menor custo possível ao projeto. Por isso, faz-se necessária a análise da execução do processo de mudança de software, visto que projetos de desenvolvimento de software não se preocupam em planejar a execução deste procedimento de mudança, principalmente com relação à alocação dos papéis na formação do CCB para execução da análise de impacto da mudança. Determinados papéis, que são alocados na formação do CCB para análise de impacto da mudança, possuem um custo alto para o projeto. Mas, dependendo da mudança, esta alocação poderia ser de um papel com o custo menor.

A aplicação de métodos formais na avaliação de desempenho de processos, por serem matematicamente fundamentados, pode representar uma boa estratégia, pois permitem avaliações de qualquer fluxo de execução. Portanto, modelos de desempenho são bastante úteis para o entendimento e previsão do comportamento de processo de mudança de software.

O trabalho em comento apresentou uma modelagem do processo de gerência de mudança de software baseada em *Stochastic Petri Nets* (SPN), com o objetivo de realizar avaliações de desempenho do processo de gerenciamento de mudanças de software. Além disso, uma metodologia foi apresentada com o intuito de auxiliar no processo de avaliação de desempenho do processo de mudança de software. Essa metodologia foi composta por uma série de passos que envolvem desde o entendimento do ambiente, seleção de métricas de desempenho até a validação do modelo e interpretação dos resultados. Com a utilização das SPNs como ferramenta

de modelagem e análise, foi possível aferir métricas de maneira probabilística.

Com o intuito de validar tanto o modelo SPN proposto, quanto a metodologia de avaliação, foram realizados três estudos de caso. Os resultados obtidos através do modelo SPN demonstrou a grande aplicabilidade em análise de desempenho do processo de mudança de software, visto que a avaliação desses cenários sem a utilização dos modelos seria uma tarefa complexa, dispendiosa financeiramente e custosa.

7.1 Contribuições, Limitações e Dificuldades

As contribuições deste trabalho são as seguintes:

- Proposição de modelo SPN para representar o funcionamento do processo de mudança de software. Através desse modelo, foi possível aferir métricas de desempenho.
- Desenvolvimento de uma metodologia para auxiliar o processo de avaliação do modelo de desempenho do processo de mudança de software. Essa metodologia é composta por uma série de etapas que envolvem desde o entendimento do ambiente, seleção de métricas de desempenho até a validação do modelo e interpretação dos resultados.
- Com a aplicação da metodologia proposta, vários problemas relacionados à modelagem do processo de mudança de software poderão ser solucionados;
- Suporte ao planejamento do processo de mudança de software. Com o modelo proposto, é possível prever o custo da execução do processo de mudança de software. Através dos resultados obtidos, o avaliador pode tomar algumas medidas para melhorar o processo de mudança de software.
- A metodologia desenvolvida pode ser aplicada para suporte ao planejamento a qualquer outro tipo de processo de software, onde com a aplicação da metodologia proposta, vários problemas relacionados à modelagem do processo analisado podem ser solucionados;

As limitações deste trabalho foram às seguintes:

- A medição do processo de mudança de software é realizada de forma manual, portanto, falhas podem ocorrer no registro dos dados coletados e na medição, e conseqüentemente, erros podem ser inclusos nos resultados obtidos;

As dificuldades deste trabalho foram às seguintes:

- Encontrar trabalhos relacionados à avaliação de desempenho de processos de mudança de software.
- Coleta dos dados na validação, pois nos projetos reais as equipes não registram as informações no momento exato e o esforço médio de análise de impacto teve que ser coletado através do *feeling* dos gestores da organização.
- Na fase de validação do modelo de desempenho refinado, foram necessários diversos ajustes nos parâmetros do modelo até que os erros em relação aos dados medidos fossem aceitáveis. Sendo assim, custosa essa fase;

7.2 Trabalhos Futuros

Outros estudos podem ser produzidos através da metodologia de avaliação propostas. A seguir, alguns deles são apresentados:

- Avaliação de desempenho para suporte ao planejamento da execução de outros processos de desenvolvimento de software.
- Construção de uma ferramenta que auxilie o processo de avaliação de desempenho dos dados medidos na geração dos modelos SPN para processos no desenvolvimento de software. Assim, após o processo de medição utilizando a ferramenta, seria possível guardar, organizar e tratar os dados que foram medidos de forma mais eficiente, possuindo uma base histórica para comparações futuras. Assim como também, os tempos na análise dos dados e o refinamento dos modelos seriam reduzidos.

REFERÊNCIAS

1. ABDALA, Martha A. D.; SANT'ANNA, Nilson. "Modelagem do Processo de Gerenciamento da Configuração de Software para um Ambiente Integrado." São Paulo/2003.

2. ABDALA, Martha Adriana Dias "Uma Abordagem para a Gerência das Modificações e da Configuração em um Ambiente Integrado para o Desenvolvimento e Gestão de Projetos de Software", São José dos Campos, 2004.

3. Andrade, Ermeson Carneiro. Modelagem e Análise de Especificação de Sistemas Embarcados de Tempo-Real Crítico com Restrições de Energia. Recife/ 2009.

4. ARAUJO, Carlos Julian Menezes. Avaliação e Modelagem de Desempenho para Planejamento de Capacidade do Sistema de Transferência Eletrônica de Fundos utilizando Tráfego em Rajada. Recife/2009.

5. Barbaresco, Eduardo Alexandre. SOFTWARE DE APOIO AO PROCESSO DE GERÊNCIA DA CONFIGURAÇÃO SEGUNDO NORMAS E MODELOS DA QUALIDADE. Blumenau/ 2000.

6. Barros, João Paulo M. P. Ramos. Introdução à modelação de sistemas utilizando redes de Petri. Beja/2001.

7. Booch, G. Rumbaugh, J. Jacobson, I. UML. Guia do Usuário. Campus, 2000.

8. Booch, G. Rumbaugh, J. Jacobson, I.The Unified Software Development Process. Addisn-Wesley, 1999.

9. Buxton,J.N., Randell,B. Software Engineering Techniques: Report on a Conference sponsored by the NATO science committee. 27-31 October, Rome, Italy, 1969.

10. C.G. CASSANDRAS. LAFORTUNE; Stéphane; "Introduction to Discrete Event Systems", book, 1999.

11. CMMI® for Development, Version 1.2, August 2006

12. Chung, Christopher A. Simulation Modeling. Handbook. A Practical Approach. CRC Press, 2004.

13. DIAS, Ursulino P, Faria, Celso Luis. Z. Como Modelar Processos de Negócios Utilizando Diagrama de Atividades da UML. 2008.

14. D Mongomery, G Runger. Applied Statistics and Probability for Engineers. John Wiley and Sons, 3nd Edition, 2002.

15. A.A. Desrochers and R.Y. Al-Jaar. Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis. IEEE Press, 1995.

16. FAGUNDES, Roberta Andrade de Araújo. Avaliação de Desempenho do Serviço de Controle de Concorrência usando Rede de Petri Estocástica. Recife/2006.

17. Filho, José Silvino. Documento do site: http://www.melhoriacontinua.com.br/index.php?option=com_rokdownloads&view=file&Itemid=56&id=142. Último acesso em 01 de outubro de 2010.

18. Figueira, M.M.C, Identificação de Outliers, MILLENIUM nº12 – Outubro de 1998.

19. FRANCES, Carlos Renato Lisboa. Introdução às Redes de Petri. Pará/2003.

20. Fowler, M. Scott, K. UML Essencial: Um breve guia para a linguagem-padrão de modelagem de objetos. Bookman, 2000.

21. GUEDES, Gilleans T. A. UML – Uma Abordagem Prática. Novatec, 2004.

22. R. German, C. Kelling, A. Zimmermann, and G. Hommel. TimeNET a toolkit for evaluating non-Markovian stochastic Petrinets. In Petri Nets and Performance Models, 1995.

23. C. Girault and R. Valk. Petri nets for systems engineering: a guide to modeling, verification, and applications. Springer, 2003.

24. Haverkort, Boudewijn R. Performance of Computer Communication Systems: A Model-Based Approach. John Wiley & Sons, 1998.

25. ISO. http://www.iso.org/iso/home.html. Último acesso em 01 de julho de 2010.

26. KERZNER, H. Project Management: A Systems Approach to Planning, Scheduling, and Controlling. New Jersey: John Wiley & Sons, 2006, 9th Edition.

27. Lachtermacher, L., Silveira, D., Paes, R., Lucena, C. Transformando o Diagrama de Atividade em uma Rede de Petri. Rio de Janeiro, 2008.

28. Maciel, P., Lins, R., Cunha, P. Introdução às Redes de Petri e Aplicações. Campinas - SP: Sociedade Brasileira de Computação, 1996.

29. Marranghello, Norian. Redes de Petri: Conceitos e Aplicações. DCCE/IBILCE/UNESP, 2005

30. P. Merlin, D. Farber, I.B.M.T.J.W.R. Center, and Y. Heights. Recoverability of communication protocols {implications of a theoretical study. Communications, IEEE Transactions on [legacy, pre-1988], 24(9):1036{1043,1976.

31. M. Malhotra and A. Reibman. Selecting and implementing phase approximations for semi-Markov models. Communications in statistics. Stochastic models, 1993.

32. C.V. Janousek. Modelling Objects by Petri Nets. PhD thesis, PhD thesis, Department of Computer Science and Engineering, Technical University of Brno, Czech Republic,(In Czech). CiteSeer. IST-Copyright Penn State and NEC, 1998.

33. T. Murata. Petri nets: Properties, Analysis and Applications. Proceedings of the IEEE, 1989.

34. M.A. Marsan. Stochastic petri nets: an elementary introduction. Advances in Petri Nets, 424:1{29, 1989.

35. M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. Modelling with Generalized Stochastic Petri Nets. John Wiley and Sons, 1995.

36. Montgomery, Douglas C. e Runger, George C. Applied Statistics and Probability for Engineers. s.l. : John Wiley & Sons, Inc., 2002.

37. MPS-Br. http://www.softex.br/mpsbr/_home/default.asp. Último acesso em 03 de julho de 2010.

38. NESMA. http://www.nesma.nl/section/fpa/earlyfpa.htm. Último acesso em 23 de setembro de 2010.

39. OMG, Object Management Group - Unified Modeling Language (UML) Superstructure Specification, version 2.0, 2005. Disponível em: http://www.omg.org/cgi-bin/doc?formal/05-07-04, Acessada em: 02/2008.

40. Oliveira, César Augusto Lins. SIMULAÇÃO DE REDES DE PETRI EM AMBIENTE JAVA. Recife, 2006.

41. Oliveira, Fabrício. SOFTWARE DE APOIO À GERÊNCIA DE SOLICITAÇÃO DE MUDANÇAS. Blumenau, 2006.

42. C. A. Petri. Kommunikation mit Automaten. PhD thesis, Bonn: Institut fÄur Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962.

43. PRESSMAN, Roger S.. *Engenharia de Software*. São Paulo: Pearson Makron Books, 2001.

44. PROJECT MANAGEMENT INSTITUTE - PMI. A Guide to the Project Management Body of Knowledge - PMBoK. 2004, Pennsylvania: USA.

45. PFLEEGER, Shari Lawrence. Engenharia de Software: Teoria e Prática. 2.ed. São Paulo: Prentice Hall, 2004.

46. R. German, C. Kelling, A. Zimmermann, and G. Hommel. TimeNET a toolkit for evaluating non-Markovian stochastic Petrinets. In Petri Nets and Performance Models, 1995.

47. RIGO, Fernanda. Adequação da Norma ISO/IEC FDIS 14764 para manutenção de software da web. Passo Fundo/2008.

48. REIS, Carla Alessandra Lima. Introdução à Modelagem de Processos de Software. Belém/2004.

49. SOMMERVILLE, Ian. Engenharia de Software. Addison-Wesley, 2005.

50. Schneider, Ricardo Luiz. UM SISTEMA DE GERÊNCIA COOPERATIVA DE CONFIGURAÇÃO DE SOFTWARE. Rio de Janeiro, 2001.

51. SALES, Afonso Henrique Corrêa de. Um Estudo sobre Redes de Petri Estocásticas Generalizadas. Porto Alegre/2002.

52. SWEBOK. Guide to the Software Engineering Body of Knowledge. IEEE, 2004.

53. TimeNET 3.0 User Manual. A Software Tool for the Performability Evaluation with Stochastic Petri Nets. Performance Evaluation Group, TU Berlin, 2001.

54. K.S. Trivedi. Probability and Statistics with Reliability, Queuing, and Computer Science Applications. John Wiley and Sons, 2nd Edition, 2006.

55. M.F. Triola. Introdução à estatística. (Tradução Vera Regina de Farias e Flores; revisão técnica Ana Maria Lima de Farias e Flores). Livros técnicos e Científicos, 2005.

56. Vieira, Hugo Vares. GERAÇÃO DE CASOS DE TESTE PARA A INTERFACE DE USUÁRIO DE SISTEMAS DE GERÊNCIA DE WORKFLOW. Porto Alegre/2008.

57. Wthreex. http://www.wthreex.com/rup/portugues/index.htm. Último acesso em 01 de maio de 2010.

58. Wikipédia. http://pt.wikipedia.org/wiki/Gerência_de_Configuração_de_Software. Último acesso em 03 de julho de 2010.

APÊNDICE A

I. Intervalo de Confiança

Em estatística, um intervalo de confiança (IC) (Montgomery, et al., 2002) é um intervalo estimado de um parâmetro estatístico. Em vez de estimar o parâmetro por um único valor, é dado um intervalo de estimativas prováveis. Quão prováveis são estas estimativas é determinado pelo coeficiente de confiança. Quanto maior a probabilidade do intervalo conter o parâmetro, maior será o intervalo. Mongomery (D Mongomery, 2002) explica que uma estimativa de intervalo para um parâmetro populacional é chamado de intervalo de confiança.

Os intervalos de confiança são amplamente utilizados na engenharia e as ciências. Intervalos de confiança são usados para indicar a confiabilidade de uma estimativa. Por exemplo, um IC pode ser usado para descrever quão confiáveis são os resultados de uma pesquisa. Sendo todas as outras coisas iguais, uma pesquisa que resulte num IC pequeno é mais confiável do que uma que resulte num IC maior.

Em sentido estrito, um IC para um parâmetro populacional é um intervalo com uma proporção p associada a qual é gerada por uma amostra aleatória de uma população subjacente, de tal forma que se a amostragem for repetida inúmeras vezes e o intervalo de confiança for recalculado para cada amostra de acordo com o mesmo método, uma proporção p dos intervalos de confiança conteria o parâmetro estatístico em questão. Intervalos de confiança são a forma predominante de estimativa por intervalo.

Se U e V são estatísticas (isto é, variáveis aleatórias) cuja distribuição de probabilidade dependa de algum parâmetro não observável θ, e $\Pr(U < \theta < V|\theta) = x$(onde *x* é um número entre 0 e 1) então o intervalo aleatório (U, V) é um intervalo de confiança "100x% para θ". O número x é chamado de nível de confiança ou coeficiente de confiança. Na

prática moderna aplicada, a maioria dos intervalos de confiança estão no nível de 95%.

a. Cálculo do Intervalo de Confiança da Média

$\overline{x}$ ± margem de erro

$$\overline{x} \pm \varepsilon_{\overline{x}}$$

i. Quando o σ (desvio padrão) é conhecido

Se $Z_{\frac{\alpha}{2}} = \frac{\pm \varepsilon_x}{\sigma_x}$, então

$$\varepsilon_{\overline{x}} = z_{\frac{\alpha}{2}} . \sigma_{\overline{x}}$$ e

$$\varepsilon_{\overline{x}} = z_{\frac{\alpha}{2}} . \frac{\sigma}{\sqrt{n}}$$

ii. Quando o σ (desvio padrão) é desconhecido

Temos que

$$\varepsilon_{\overline{x}} = t_{n-1gl;\frac{\alpha}{2}} . \sigma_{\overline{x}}$$ e

$$\varepsilon_{\overline{x}} = t_{n-1gl;\frac{\alpha}{2}} . \frac{s}{\sqrt{n}}$$

iii. Para diferença (Teste t-emparelhado)

Para um só conjunto de dados com n observações:

$$S_{Erro} = \frac{S}{\sqrt{n}}$$

Para dois conjuntos de medições com n1 e n2 observações cada e desvios padrões s1 e s2 respectivamente:

$$S_{Erro} = S \times \sqrt{\frac{1}{n1} + \frac{1}{n2}}$$

$$S^2 = \frac{(n1-1)S1^2 + (n2-1)S2^2}{n1+n2-2}$$

b. Método Bootstrap

Seja uma amostra original e a estatística de interesse abaixo (Montgomery, et al., 2002):

x ={x1, x2, x3,..., xn-1, xn}.

$$\hat{\theta} = F(X)$$

Geram-se as amostras bootstrap x(1), x(2), x(3),... , x(n*) com reposição de x.

Calculam-se as estimativas da estatística de interesse:

$\hat{\theta}$(b) = F[x(b)] , b=1, ..., B

Calcula-se o erro padrão bootstrap, $\hat{S}$boot, dado por:

$$\hat{S}_{boot} = \frac{1}{B-1}\left\{\sum_{B-1}^{B}\left[\hat{\theta}_b - \hat{\theta}_{(*)}\right]^{2\,1/2}, sendo\right\}$$

$$\hat{\theta}_{(*)} = \sum_{B-1}^{B}\theta_{(b)}$$

APÊNDICE B

A sintaxe das métricas utilizadas nas métricas deste trabalho foram baseadas na sintaxe da ferramenta TimeNet (Zimmerman, 2001). Os elementos especiais para definições das métricas utilizadas são explicadas a seguir. Todos eles mostram os resultados de uma análise ou uma simulação, tanto em estado estacionário ou transiente (Zimmerman, 2001).

- ”P{” <condição lógica> ”}” = Probabilidade de <condição lógica>
- ”P{” <condição lógica 1> ”IF” <condição lógica 2> ”}” = Probabilidade de <condição lógica 1> sob a condição de <condição lódica 2> (probabilidade condicional)
- ”E{” <função marcada> ”}” = valor esperado da expressão dependente da marcação <função marcada>
- ”E{” <função marcada> ”IF” <condição lógica> ”}” = valor esperado da expressão dependente da marcação <função marcada>; apenas marcações quando <condição lógica> avaliada como verdadeiro são levados em consideração.