Universidade Federal de Pernambuco
Centro de Informática

Pós-graduação em Ciência da Computação

AVALIAÇÃO DE DESEMPENHO DE PROCESSOS DE TESTES DE SOFTWARE

Marcelo Luiz Monteiro Marinho

DISSERTAÇÃO DE MESTRADO

Recife
Agosto de 2010

Universidade Federal de Pernambuco
Centro de Informática

Marcelo Luiz Monteiro Marinho

AVALIAÇÃO DE DESEMPENHO DE PROCESSOS DE TESTES DE SOFTWARE

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: *Prof. Dr. Paulo Romero Martins Maciel*
Co-orientador: *Prof. Msc. Teresa Maria Medeiros Maciel*

Recife
Agosto de 2010

A Deus.
À Minha Família.
Aos Meus Amigos.

AGRADECIMENTOS

- A Deus, meu amparo e meu refúgio, o grande responsável por mais uma vitória em minha vida.
- Aos meus amados pais Luiz e Fátima e minha irmã Patrícia, são o suporte de minha caminhada e o alicerce de minha vida.
- À minha linda namorada Mariana, por estar ao meu lado sempre me apoiando.
- Ao meu orientador Paulo Maciel, pela paciência, dedicação e por ter acreditado, acima de tudo, em mim e no meu trabalho.
- À Teresa Maciel, pela orientação, apoio e enorme paciência, elementos essenciais para o desenvolvimento deste trabalho.
- À Erica Souza, uma amiga que me deu um grande apoio na minha pesquisa.
- Ao Professor Manoel Eusébio, um amigo que sempre me incentivou no meu trabalho.
- Aos professores Alexandre Vasconcelos e Eduardo Tavares, por terem aceitado o convite de compor a banca e por terem contribuído na melhoria deste trabalho.
- Ao meu tio Aristoteles Veríssimo, que, de uma forma simples, tem me ajudado na minha caminhada acadêmica.
- A Renan Freitas, um mestre e amigo dos tempos de colégio, que deu um apoio bastante significativo na minha dissertação.
- A todos os meus amigos, pelo carinho e apoio nos momentos mais difíceis desta jornada.

A mente que se abre a uma nova idéia jamais voltará ao seu tamanho original.

—ALBERT EINSTEIN

RESUMO

A procura por softwares com maior qualidade tem motivado a definição de métodos e técnicas para o desenvolvimento de softwares que atinjam os padrões de qualidade impostos. Com isso, o interesse pela atividade de teste de software vem aumentando nos últimos anos. As fábricas de software enfrentam dificuldades na elaboração de processos de testes adequados ao projeto de maneira que sejam efetivos com relação à qualidade do produto e, ao mesmo tempo, tenham execução eficiente. Esses aspectos concorrentes podem afetar os níveis de qualidade almejado ou induzir o desenvolvimento de processos rebuscados e ineficientes. Parte desse problema ocorre tanto devido à dificuldade enfrentada pelas organizações na definição de processos a cada projeto particular, quanto pela ausência de mecanismos que possibilitem a provisão de meios para escolha das alternativas mais convenientes a cada projeto particular em termos de desempenho e critérios de qualidade.

Dessa forma, ambientes que proporcionem a avaliação do desempenho dos processos e que possibilitem estimativa do uso de recursos são mecanismos que concorrem para melhoria dos índices de qualidade e produtividade das organizações. Modelos de execução de processo voltados para estimativa de desempenho que levem em consideração combinações de cenários diversos e ativos podem trazer ganhos substanciais de produtividade tanto na customização dos processos quanto na efetividade do processo definido para o projeto.

Este trabalho propõe uma metodologia de avaliação de desempenho aplicada a Processos de Testes de Software. Com a aplicação da metodologia proposta, é possível verificar o impacto de mudanças no processo, avaliar o desempenho do processo de testes, realizar simulações com o objetivo de obter estimativas mais precisas e, principalmente, ajudar na garantia da qualidade do produto. Além disso, essa metodologia possibilitará a avaliação de diferentes alternativas de implementações, bem como a verificação de melhor composição de recursos pessoais para as atividades do processo. Isso tudo pode ser realizado sem a necessidade da real implementação do processo, tornando mais ágil e barato todo o processo.

Palavras-chave: Avaliação de Desempenho, Testes de Software, Redes de Petri Estocásticas, Planejamento de Testes de Software, Modelos Estocásticos Expolinomiais, Processos de Testes de Software, *Performance engineering*.

ABSTRACT

The demand for higher quality software has motivated the definition of methods and techniques for the development of software that meet the imposed quality standards. As a result, in recent years, interest in software testing activities has increased. Software manufacturers have been facing difficulties in relation to develop appropriate testing procedures for projects that meet the required product quality and at the same time perform efficiently. Such competing concerns can affect the quality levels desired, or cause the development process to become convoluted and ineffective.

The problem occurs in part because of the difficulty faced by organizations during the design process for each particular project, and also because the absence of mechanisms that allow for the provision of means for selecting substitutes more suitable to each project in terms of performance and quality criteria. For that reason, environments that provide for the evaluation of process development and enable the assessment of resource use contribute to the improvement of the quality level and productivity of organizations. Models for implementing processes that are aimed at estimating performance and consider a variety of diverse and dynamic scenarios can bring significant productivity gains in the customization of the processes as well as in the effectiveness of the procedure defined for the project.

This work proposes a performance evaluation methodology for Software Testing processes. Employing such a methodology makes it possible to validate the impact of process changes, assess the performance of the testing process, perform simulations to obtain more accurate estimates, and particularly help towards guaranteeing product quality. Furthermore, this methodology will allow for the evaluation of different implementation alternatives, as well as identify the best composition of personal resources for process activities. This can all be accomplished without requiring the actual implementation of the process, making the whole process faster and more efficient.

Keywords: Performance Evaluation, Software Testing, Stochastic Petri Net, Test Planning, Expolinomial Stochastic Models, Test Process, Performance engineering.

SUMÁRIO

Capítulo 1—Introdução 1

1.1 Contexto 1
1.2 Motivação 2
1.3 Objetivos e Contribuições 4
1.4 Estrutura da Dissertação 5

Capítulo 2—Trabalhos Relacionados 6

2.1 Contextualização dos Trabalhos Relacionados 6
2.2 Considerações Finais 10

Capítulo 3—Redes de Petri 11

3.1 Introdução 11
3.1.1 Propriedades das Redes de Petri 13
3.1.1.1 Propriedades Comportamentais 13
3.1.1.2 Propriedades Estruturais 14
3.2 Rede de Petri Estocástica 16
3.3 Moment Matching 20
3.4 Considerações Finais 24

Capítulo 4—Metodologia de Avaliação 25

4.1 Contexto 25
4.1.1 Planejamento de Projetos 25
4.1.2 Testes de Software 26
4.1.2.1 Processo de Testes de Software 27
4.1.2.2 Planejamento de Testes de Software 28

4.2 Metodologia de Avaliação de Desempenho 29
4.3 Considerações Finais 32

Capítulo 5—Modelagem 33

5.1 Visão Geral 33
5.2 Mapeamento dos Diagramas de Atividades - UML em SPN 34
5.2.1 Mapeamento de Atividades 34
5.2.2 Mapeamento de Transições 35
5.2.3 Mapeamento do Estado Inicial e Final 35
5.2.4 Barras de Sincronização 36
5.2.5 Mapeamento Decisão 37
5.3 Modelos Para Alocação de Recursos 38
5.3.1 Modelo de Alocação de Recursos 39
5.3.2 Modelos de Alocação de Diferentes Recursos 40
5.4 Validação 41
5.5 Considerações Finais 53

Capítulo 6—Estudos de caso 55

6.1 Introdução 55
6.2 Estudo de Caso: Alocação de Engenheiros 55
6.3 Estudo de Caso: Alocação de Engenheiros e *Trainees* 58
6.4 Estudo de Caso: Alocação de Engenheiros, *Trainee* e Estagiário 63
6.5 Estudo de Caso: Desempenho da Atividade Implementar Testes 65
6.6 Estudo de Caso: Quantidade de Defeitos escapados 68
6.7 Considerações Finais 70

Capítulo 7—Conclusão 72

7.1 Considerações Finais e Trabalhos Futuros 75

Apêndice A—Apêndice A 83

A.1 Sintaxe das expressões suportadas pelo TimeNet 4.0 83

A.1.1 Símbolos Usados 83
A.1.2 Definições da Sintaxe 84
A.1.3 Expressões especiais para definições de métricas 86

LISTA DE FIGURAS

3.1 Elementos de rede de Petri 12
3.2 Exemplo de rede de Petri 12
3.3 Períodos do Dia 12
3.4 Técnicas de Redução 16
3.5 Geração de Gráfico de Alcançabilidade 19
3.6 *Throughput Subnets* 20
3.7 Distribuição Empírica 22
3.8 Distribuição Erlang 23
3.9 Distribuição Hipoexponencial 23
3.10 Distribuição Hiperexponencial 24

4.1 Metodologia de Avaliação de Desempenho. 30

5.1 Exemplo de um DA. 34
5.2 Mapeamento de Diagramas de Atividades para modelo SPN. 35
5.3 Exemplo do fluxo de Transições. 35
5.4 Mapeamento de Transições. 36
5.5 Mapeamento do Estado Inicial. 36
5.6 Mapeamento do Estado Final. 36
5.7 Barra de sincronia. 37
5.8 Mapeamento do Fork. 38
5.9 Mapeamento do Join. 38
5.10 Decisão. 39
5.11 Mapeando Decisão 39
5.12 Alocação de Recursos. 40
5.13 Alocação de Colaboradores Com Diferentes Qualificações 41
5.14 Processo de Testes 42

5.15 Modelo Abstrato do Processo de Testes. 48
5.16 Modelo Abstrato Para Estimativa de Recurso Pessoal. 50
5.17 Modelo Refinado Para Estimativa de Recurso Pessoal. 54

6.1 Gráfico das Composições Avaliadas. 57
6.2 Modelo Abstrato de Alocações de Recursos Pessoais. 60
6.3 Modelo Refinado de Alocações de Recursos Pessoais. 61
6.4 Modelo Refinado de Atividade de Alocações de Recursos Pessoais. 61
6.5 Gráfico das Composições Avaliadas - Engenheiros e *Trainees*. 62
6.6 Gráfico das Composições Avaliadas - Engenheiros, *Trainees* e Estagiários. 64
6.7 Tempo do Processo de Testes Com e Sem a Atividade Implementar Testes - Engenheiros e *trainees*. 66
6.8 Custo do Processo de Testes Com e Sem a Atividade Implementar Testes - Engenheiros e *trainees*. 66
6.9 Tempo do Processo de Testes, Com e Sem a Atividade Implementar Testes - engenheiros, *trainees* e estagiários 68
6.10 Custo do Processo de Testes Com e Sem a Atividade Implementar Testes - engenheiros, *trainees* e estagiários 68
6.11 Modelo Qualidade do Processo de Testes. 69
6.12 Número Defeitos Escapados. 70
6.13 Número de Defeitos Escapados Acumulados. 71
6.14 Custo X Tempo das atividades do processo. 71

LISTA DE TABELAS

5.1 Tempo das Atividades de Testes 47
5.2 Intervalos de Confiança dos Tempos de Execução 47
5.3 Média e Desvios Padrão das Amostras 47
5.4 Utilização dos Colaboradores 52

6.1 Tabela de Composições 1 56
6.2 Tabela de Composições 2 56
6.3 Tabela de Composições 3 57
6.4 Engenheiros e *Trainees* 61
6.5 Composições 1: Engenheiros, Trainee e Estagiário 63
6.6 Composições 2:Engenheiros, Trainee e Estagiário 64
6.7 Implementar Testes: Engenheiros e *Trainees* 65
6.8 Implementar Testes: Engenheiros, *Trainees* e Estagiário 67
6.9 Quantidade de Defeitos Escapados Por Nível de Colaboradores Envolvidos 70

LISTA DE ABREVIATURAS

CMMI - *Capability Maturity Model Integration.*

DA - Diagrama de Atividades.

IEEE - *Institute of Electrical and Electronics Engineers.*

OMG - *Object Management Group.*

PE - *Performance Engineering.*

RdP - Redes de Petri.

RUP - *Rational Unified Process.*

SEI - *Software Engineering Institute.*

SPN - *Stochastic Petri Nets.*

TimeNET - *Timed Net Evalution.*

TI - *Tecnologia da Informação.*

UML - *Unified Model Language.*

CAPÍTULO 1

INTRODUÇÃO

Este capítulo traz uma breve introdução ao processo de testes de software, destacando-se as principais restrições a serem avaliadas neste contexto, apresenta também a necessidade da integração dos modelos semiformais e formais. Em seguida, são apresentadas a motivação e a proposta deste trabalho bem como o seu escopo.

1.1 CONTEXTO

Computadores e sistemas de software estão se tornando onipresentes nas modernas sociedades. Usuários contam com computadores individuais e interligados, a fim de satisfazer as diversas necessidades de processamento de informações, armazenamento de dados, busca e recuperações. Todas essas necessidades são atendidas com apoio de programas de software [Tia05]. Essa dependência exige um correto funcionamento do software durante todo o tempo de uso, porém, apesar dos inúmeros avanços na Engenharia de Software, muito ainda é discutido acerca da qualidade e da produtividade da indústria de software. Níveis de qualidade inadequados, assim como baixa produtividade, geram insatisfação e prejuízo às organizações. A utilização ubíqua dos computadores proporciona o surgimento de demandas cuja complexidade é crescente, bem como exigem estruturas ágeis para o atendimento aos requisitos de desempenho.

O desenvolvimento de sistemas de software envolve uma série de atividades de produção em que as oportunidades de injeção de falhas humanas são enormes. Erros podem começar a acontecer logo no começo do processo, no qual os objetivos podem estar errônea ou imperfeitamente especificados, além de erros que venham a ocorrer em fases de projetos e desenvolvimento posteriores. A atividade de testes de software é um elemento crítico de garantia de qualidade de software e representa a última revisão de especificação, projeto e codificação [PMM02].

Testes de software têm como objetivo identificar a integridade e a qualidade do software desenvolvido. Possuem o objetivo de demonstrar a validade do software e a autenticidade do sistema final no que diz respeito ao requisito do cliente[Per06]. No entanto, em muitos projetos de desenvolvimento de aplicações, não são usados recursos suficientes na fase de testes. Com a pressão da concorrência, as organizações começaram a aplicar testes em estágios iniciais de desenvolvimento de software, é visto que muitas empresas se esforçam para encontrar estratégias de teste eficazes. Poucas organizações estabelecem métricas para medir a eficácia dos testes. Sem métricas e estratégia de gestão adequada para os testes, a eficácia deles não pode ser medida ou melhorada.

O destaque crescente do software nas organizações e os "custos"envolvidos associados

às falhas de software são incentivos para uma atividade de teste cuidadosa e bem planejada. Não é incomum que uma empresa de software gaste 40 por cento do esforço de projeto total em testes[PMM02]. No entanto, a atividade de teste de software de sistemas, dos quais dependem vidas humanas (por exemplo, controle de vôo, monitoração de reatores nucleares), pode custar três a cinco vezes mais que todos os outros passos da engenharia de software juntos [Mye04].

Com o objetivo de atender requisitos de qualidade, as fábricas de softwares procuram definir processos a fim de atender os critérios de qualidade dos produtos gerados e eficiência do processo produtivo. Definir processos adequados que garantam a qualidade do produto e produtividade no desenvolvimento tem representado um desafio para as indústrias de software. Nesse contexto, a estruturação de processos flexíveis compostos de ativos independentes, de forma a serem facilmente adequados a cada projeto específico, tem sido uma abordagem amplamente adotada pelas organizações. Buscando o aumento da qualidade, notações como a Linguagem de Modelagem Unificada - UML [Lar08] especificada pela OMG proporcionam avanços consideráveis para modularização e manutenção dos processos software. No entanto, esses modelos por si só não fornecem suporte para avaliação de desempenho das especificações dos processos, assim, faz-se necessário o mapeamento desses modelos semiformais para modelos formais.

Dessa forma, ambientes que proporcionem a avaliação do desempenho dos processos e que possibilitem estimativa do uso de recursos são mecanismos que concorrem para melhoria dos índices de qualidade e produtividade das organizações. Modelos de execução de processo voltados para estimativa de desempenho que levem em consideração combinações de cenário diversos e ativos podem trazer ganhos substanciais de produtividade tanto na customização dos processos quanto na efetividade do processo definido para o projeto.

A customização de processos de testes de software é feita a partir da seleção e combinação de ativos (ex.: atividades, modelos de artefatos, guias etc) armazenados em um repositório. Será validada através de um modelo estocástico para avaliação de desempenho de processo de testes que deve estimar métricas a partir de critérios pré-estabelecidos, viabilizando inclusive a comparação de mais de uma solução para o processo customizado.

1.2 MOTIVAÇÃO

A procura por softwares com maior qualidade tem motivado a definição de métodos e técnicas para o desenvolvimento de softwares que atinjam os padrões de qualidade impostos. Com isso, o interesse pela atividade de teste de software vem aumentando nos últimos anos. Diversos pesquisadores têm investigado os diferentes critérios de teste, buscando obter estratégias de teste com baixo custo de aplicação, mas, ao mesmo tempo, com capacidade para revelar erros [MBV+04].

"Os processos de software formam a 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"[PMM02]. Neste contexto, as organizações de software procuram instituir processos que garantam níveis de qualidade compatíveis com as demandas particulares de cada contexto.

O processo de desenvolvimento de software envolve uma série de atividades nas quais, apesar das técnicas, métodos e ferramentas empregados, os erros no produto ainda podem ocorrer. Atividades agregadas sob o nome de Garantia de Qualidade de Software são introduzidas ao longo de todo o processo de desenvolvimento, entre elas atividades de Teste, também conhecidas por Processo de Testes de Software, cujo objetivo é minimizar a ocorrência de erros e riscos associados.

As fábricas de software enfrentam dificuldades na elaboração de processos de testes adequados ao projeto de maneira que sejam efetivos com relação à qualidade do produto e, ao mesmo tempo, tenham execução eficiente. Esses aspectos concorrentes podem afetar os níveis de qualidade almejados ou induzir o desenvolvimento de processos rebuscados e ineficientes. Parte desse problema ocorre tanto devido à dificuldade enfrentada pelas organizações na definição de processos a cada projeto em particular, quanto pela ausência de mecanismos que possibilitem a provisão de meios para escolha das alternativas mais convenientes a cada projeto em termos de desempenho e critérios de qualidade.

Metodologias e métodos para avaliação de processos têm sido amplamente adotados em diversos contextos da engenharia de sistemas. *Performance Engineering* é o termo utilizado para se referir ao conjunto de políticas, atividades, práticas e ferramentas aplicadas às fases do ciclo de vida dos processos almejando garantir que as soluções implementadas atendam os requisitos não-funcionais definidos. Envolve, portanto, a monitoração dos sistemas existentes, considerando-se cargas apropriadas à representação de cenários de forma a possibilitar a reprodução do comportamento temporal. Na *Performance Engineering*, ênfase particular é dada à modelagem estocástica nas diversas etapas de desenvolvimento [Smi93].

Representações típica de modelos de desempenho são baseadas em mecanismos de simulação estocásticas, redes de fila, redes de Petri temporizadas (mais notadamente as redes estocásticas) e álgebras de processo estocásticas. Alguns desses meios de representação possibilitam tanto a avaliação através de simulação quanto via análise numérica.

A disciplina de testes de software é uma das mais importantes do processo de software, pois é voltada para o alcance de um nível de qualidade do produto. Teste de software é a disciplina que irá validar e garantir a qualidade do produto para o usuário, porém é uma das etapas do processo de software que consome mais tempo.

Apesar das diversas técnicas de estimativas de atividades no processo de testes, são observáveis atrasos nos diversos projetos. Uma grande preocupação das empresas de software é gerar produtos com qualidade, para isso precisam de um bom processo de testes. A atividade de testes é uma das etapas do processo de software que utiliza uma boa parte do tempo de todo o projeto, por isso, quanto melhor for a estimativa das atividades de testes, melhor a previsão de entrega do produto. A gerência precisa avaliar

se as etapas do processo de testes são eficientes com relação ao tempo, ao custo, como também verificação da qualidade do processo de testes.

1.3 OBJETIVOS E CONTRIBUIÇÕES

As organizações procuram seguir os modelos definidos, porém enfrentam dificuldades no que diz respeito à adoção de processos efetivos e eficientes. A maioria destes modelos procura garantir a corretude do sistema, porém não apresentam garantias de que o processo possuirá um desempenho satisfatório [SSR00a]. Nesse contexto, empresas de desenvolvimento de software procuram metodologias, métodos e ferramentas para avaliação de desempenho de processos de software, porém esses mecanismos são reduzidos para o mercado de software atual.

Este trabalho propõe uma metodologia de avaliação de desempenho aplicada a Processos de Testes de Software. Essa metodologia apresenta uma série de etapas que envolvem desde o estudo do processo de testes e modelagem dos diagramas de atividades da UML até a geração e análise dos modelos de redes de Petri estocástica (SPN). Com a aplicação da metodologia proposta, é possível verificar o impacto de mudanças no processo, avaliar o desempenho do processo de testes, realizar simulações com o objetivo de obter estimativas mais precisas e, principalmente, ajudar na garantia da qualidade do produto. Além disso, essa metodologia possibilitará a avaliação de diferentes alternativas de implementações do processo de testes, bem como a verificação de melhor composição de recursos pessoais para as atividades do processo. Isso tudo pode ser realizado sem a necessidade da real implementação do processo, tornando mais ágil e barato todo o processo.

Este trabalho também propõe alguns modelos estocásticos para auxiliar o planejamento dos testes de software, considerando o processo adotado e o perfil de colaboradores envolvidos no projeto. A partir desses modelos, é possível realizar análises quantitativas (ex.: tempo de execução das atividades e custo das atividades) e verificações. Além disso, é proposto um modelo SPN baseado em um processo de testes adotado por uma instituição de software brasileira. A validação da metodologia é realizada através da comparação dos resultados do modelo SPN, e dos dados do processo da organização. Baseado no modelo SPN, são criados alguns cenários para avaliar o desempenho do processo de testes, os resultados avaliados irão auxiliar na atividade de planejamento dos testes de software.

Ainda, neste trabalho, foi elaborado uma série de modelos SPN básicos gerados a partir do processo de mapeamento dos diagramas de atividades da UML para SPN. Através da composição sistemática, é possível obter um modelo SPN que representa a especificação do processo de testes e assim é possível aplicar técnicas de análises e verificações ainda durante as fase de planejamento. Em outras palavras, a partir dos modelos formais, será possível realizar análises/verificações, antes mesmo do *software* ser implementado.

Mais especificamente, este trabalho possui os seguintes objetivos e contribuições:

- Definir uma metodologia que auxilie no processo de avaliação de desempenho das especificações dos Processos de Testes de Softwares;
- Desenvolver um método para o mapeamento dos diagramas comportamentais da UML para modelos SPN. Adotam-se, aqui, os diagramas de atividades;
- Desenvolver um modelo SPN para realizar análises (qualitativas e quantitativas) de um processo de testes de software.

Parte do trabalho descrito nesta dissertação, bem como os trabalhos relacionados podem ser encontrados nas seguintes publicações: [MMS+10a, MMS+10b, MMSM10].

1.4 ESTRUTURA DA DISSERTAÇÃO

Além deste capítulo introdutório, este trabalho está estruturado da seguinte maneira: O Capítulo 2 apresenta os trabalhos relacionados ao tema, no qual se destacam as relações entre os trabalhos. O Capítulo 3 apresenta uma visão geral dos modelos de desempenho e conceitos fundamentais das SPNs. O Capítulo 4 apresenta uma metodologia de avaliação de desempenho para o processos de testes de software. O Capítulo 5 apresenta as regras de mapeamento do diagrama de atividades da UML para SPN, como também o mapeamento (modelos básicos) adotado para representar Processo de Teste de Software. Apresenta também o modelo de avaliação de desempenho de processo de testes de software bem como a validação do modelo. O Capítulo 6 apresenta um estudo de caso no qual foi aplicada a metodologia e o modelo proposto. Finalmente, o Capítulo 7 conclui o trabalho e apresenta os trabalhos futuros.

CAPÍTULO 2

TRABALHOS RELACIONADOS

Este capítulo tem por finalidade apresentar alguns dos trabalhos relacionados a esta dissertação, destacando-se algumas das principais contribuições relacionadas ao contexto deste documento, como também a necessidade de propor melhorias e a importância da avaliação de desempenho dos processos de testes através dos modelos formais.

2.1 CONTEXTUALIZAÇÃO DOS TRABALHOS RELACIONADOS

Atualmente, desenvolver produtos de software de alta qualidade é de primordial importância, considerando a relevância do processo de testes diante da garantia de qualidade do produto. A adoção da disciplina de testes no processo de software nas empresas vem crescendo, e é uma das atividades do processo de software que toma uma boa parte do tempo [Mye04]. É observado que as diversas técnicas de estimativas existentes e utilizadas pelos gerentes de softwares não geram estimativas coerentes para as atividades de testes de um determinado tipo de projeto. Esta Seção apresenta uma breve descrição dos trabalhos relacionados a esta dissertação.

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.

Workflow é um conjunto coordenado de atividades que são interligadas com o objetivo de alcançar uma meta comum. São utilizados para automação e gestão de processos de negócio em todo o mundo. Esta abordagem consiste em manter modelos dos processos e fornecê-los a um Sistema de Gestão de *Workflow*, que é capaz de entendê-los, executá-los e monitorá-los.

A Melhoria de *Workflow*, uma das preocupações da Melhoria de Processos de Negócio (*Business Process Improvement*), ou BPI, tem o objetivo de realizar mudanças no processo de forma a reduzir os custos de sua execução e aumentar a qualidade de seus produtos. Um processo mais eficiente fornece maior lucratividade para a empresa. O trabalho de Oliveira et al [dO] aborda o tema da Melhoria de *Workflow* sob o ponto de

vista da análise de desempenho, aplicando neste cenário as Redes de Petri Estocásticas Generalizadas (Generalized Stochastic Petri Nets) ou GSPN.

O objetivo de [dO] é a criação de uma metodologia para melhoria do *Workflow* utilizando modelos em redes de Petri estocásticas generalizadas (GSPN). É demonstrada a melhoria do *Workflow*, o aumento de eficiência, aumento de produtividade, aumento de qualidade, adequação a normas e padrões e mudanças nas estratégias. O trabalho está bastante associado com esta dissertação, pois apresenta um modelo para avaliação de desempenho de processos de negócios e traz um estudo de caso hipotético sobre possíveis configurações de trabalho com equipes.

Esta dissertação apresenta a modelagem e a metodologia voltada para os processos de testes. Nos estudos de caso realizados utilizou-se o processo de teste e a equipe de uma organização. Os modelos que serão descritos mais à frente apresentam o uso do tratamento estatístico [DAJ95] para calcular os dois primeiros momentos da distribuição empírica (dados coletados), as estatísticas obtidas permitem a seleção da distribuição expolinomial (hipoexponencial, hiperexponencial ou *erlang*) que melhor se adapta à distribuição empírica. Essa técnica em [dO] não é abordada.

Schmietendorf, em [SSR00b], apresenta um modelo de desenvolvimento para processo na área de *performance engineering*. O modelo é denominado *The Performance Engineering Maturity Model - PEMM*. O PEMM dá um apoio sistemático e orientações para melhoria do processo de software e suporte para identificação de possíveis deficiências no processo de software existente adotado pelas organizações. A utilização deste modelo permite a avaliação de níveis de integrações e aplicações dentro da engenharia de performance, ele se apóia no modelo CMMI do SEI e é baseado em um catálogo de questionários para avaliar o desempenho dos processos de softwares. Com este modelo, os autores pretendem avaliar processos de softwares com o termo de desempenho, o modelo proposto auxilia no estabelecimento de processos para prover um bom desempenho das atividades desse processo.

Segundo Schmietendorf, a qualidade dos sistemas de softwares depende decididamente da qualidade do processo de software adotado. É comum os compradores exigirem um grau de maturidade dos softwares, com isso os clientes preferem escolher comprar produtos de organizações que, além de softwares, seguem padrões de qualidade que possuam um cuidado com performance desse processo. Como o modelo é baseado em CMMI, os autores propuseram no PEMM cinco níveis de maturidade em performance.

O modelo criado possui uma série de questionários para avaliar o desempenho dos processos de software, esses questionários foram passados para um portal web para que as organizações pudessem responder as questões e avaliar o nível de performance no processo de software.

Esta dissertação é voltada para o desempenho do processo de testes, mas, se comparada com o trabalho de [SSR00b], apresenta algumas vantagens. O trabalho aqui descrito apresenta, através da metodologia proposta e dos mapeamentos realizados, a avaliação de desempenho do processo de testes através de modelos formais. O desempenho do

processo de testes pode ser verificado antes que este seja executado. Além disso, busca ajudar na melhoria desse processo de testes através de avaliações, sem a necessidade de ter que executar todo o processo e fazer várias alterações como em [SSR00b]. Um outro ganho que a metodologia mostra a possibilidade de criar medidas de desempenho e, através dessas, estimar custo, tempo e qualidade independente do modelo de maturidade adotado pela organização.

Os gerentes de projeto devem considerar cuidadosamente os recursos necessários para concluir um projeto no prazo e no orçamento estipulado. Qualquer plano pode ser geralmente posto em prática se os referidos recursos estão disponíveis.

A tomada de decisão adequada nos projetos de software depende muito da habilidade do gerente e de sua experiência. No entanto, a modelagem de processos e ferramentas de análise podem auxiliar nas estimativas de tempo, custo e quantidade de defeitos. Existem diferentes tipos de ferramentas de apoio para o gerenciamento de projetos. Elas vão desde simples notações gráficas para a modelagem do fluxo de atividades, tais como gráficos de Gannt [Var], até ferramentas matemáticas bem sofisticadas que podem predizer, por meio de análise ou simulação, os resultados decorrentes das hipóteses do gerente sobre os recursos envolvidos e do fluxo de eventos.

Uma rede de Petri é capaz de representar a interdependência de recursos, atribuição parcial, a substituição dos recursos e a exclusividade mútua. Usando propriedades comportamentais, tais como a acessibilidade e delimitações, o planejamento de projeto pode ser melhor realizado (com restrições de recursos). Em [JKCPR00] é descrita a aplicação de redes de Petri (PNs) em gerenciamento de projeto e nivelamento de recursos usando um algoritmo heurístico.

Jeetendra [JKCPR00] utiliza redes de petri coloridas e algoritmos heurísticos como base para desenvolver um software que auxilie o gerente de projetos a analisar e definir os melhores relacionamentos do recursos pessoais para o projeto. O software conhecido por *PLS-net* é utilizado para lidar com gerenciamento de projetos utilizando tanto atividades de tempo determinísticas quanto estocásticas.

Em [JKCPR00] é apresentado o uso de modelos formais, especificamente Redes de Petri, para auxiliar os gerentes de projetos na definição de atividades com base no relacionamento entre os recursos pessoais. O trabalho dessa dissertação, baseado em processo de testes de software, possibilita a escolha de composições de equipes e, através das composições, verificar qual o melhor tempo e custo em nível de desempenho, o que não é mostrado nem verificado em [JKCPR00].

O trabalho de Bertolino et al [BMM02] aborda em particular o uso de *Performance Engineering* (PE) para ajudar os gerentes de projeto. Entre as vantagens do uso de PE para gerenciamento é destacada a capacidade de lidar com vários projetos e suas interferências mútuas em escalonamento e utilização de recursos. Também da solidez das previsões fornecidas pelas ferramentas de PE, que podem contar com um formalismo matemático e ter uma validade estatística.

Em [BMM02] é elaborada uma metodologia, que segundo os autores, pode ser útil em

qualquer estágio de desenvolvimento, quando o gerente de projetos é chamado para tomar uma decisão mais apropriada baseada no atual estado e nas circunstâncias emergentes. Técnicas de PE podem ajudar a prever os resultados sob o fluxo de trabalho atual. O trabalho utiliza técnicas de UML e RT-UML para modelar os processos e redes de fila para criar um ambiente de simulação.

Em particular, o estudo de caso que foi realizado em [BMM02] tem a ver com a decisão de lançamento de um produto de software. Foi considerado que no início da fase de testes, o gerente quer prever o tempo esperado para *release* do produto. Como primeiro passo foi modelada a estrutura da organização da empresa considerando o processo de testes e o gerenciamento dos problemas relatados. No estudo de caso, foi assumido que a composição da equipe é formada por um engenheiro de testes, duas pessoa desenvolvendo, um arquiteto de software e um gerente. As pessoas responsáveis pelos testes começam a executar os casos de testes planejados e para cada número de dias (foram assumidos três dias), inserem os relatórios de problemas em uma base de dados online chamada de sistema de detecção (TS), que apenas os responsáveis pelos testes e o gerente de projetos podem modificar. Para cada atualização de TS, o gerente de projetos analisa os relatórios de problema e toma uma decisão da liberação do produto ou de uma prorrogação do prazo para a liberação do produto.

Segundo Bertolino, com o modelo baseado em uma rede de filas e conforme a configuração da equipe, o gerente pode realizar suposições diferentes para os parâmetros do modelo derivado e imediatamente obter uma previsão confiável de quais serão os resultados conseqüentes. Foi considerado que antes do início da fase de teste, o gerente quer uma estimativa da data de release. Foi assumindo como uma primeira suposição que 10 relatórios de problemas são emitidos, a análise de PE estima que na média, o produto estará pronto para lançamento (ou seja, não tem mais a existência de relatórios de problema em abertos) após 17 dias desde o início da fase de teste. Assim, por meio de simples análises de PE, o gerente recebe previsões estatísticas que podem apoiar seu processo decisório.

O trabalho é bastante relevante e assemelha-se a esta dissertação quando é possível através de configurações, prever o tempo gasto no processo, porém não prever o tempo total do processo de testes para a tomada de decisões prévia do gerente, ou seja, só é previsto quando concluídos os testes e a partir de um relatório, o tempo de duração para a entrega do produto é estabelecido. Em [BMM02] não é analisado o desempenho em termos de qualidade do processo de testes.

Teste é um componente vital em um processo de qualidade de software e é uma das atividades mais desafiadoras e custosas cumpridas durante o desenvolvimento e manutenção de um produto de software.

No trabalho de Vale et al [Val09] foi proposta uma formalização de testes funcionais, através da especificação destes usando redes de Petri orientada a objetos e *Workflows*-Nets no contexto de softwares orientados a objetos. O trabalho mostra que é possível, a partir dos diagramas da UML, em especial o diagrama de atividades, transformar o processo apresentado nesses diagramas em uma rede de Petri e, a partir dessa rede

de Petri transformar em *workflow*-Net onde é possível executar o testes funcionais do software a partir de uma seleção de dados de entrada e, no final, comparar os dados de saída com os dados esperados, automatizando os testes funcionais.

Em [Val09], é definido o conceito de redes de Petri orientado a objetos, para representar tanto estruturas de dados quanto fluxos de controle. Por outro lado, as *workflow*-Nets apresentam uma estruturação do controle que é adaptada à execução de testes funcionais. O contexto do trabalho consiste em oferecer, de maneira formal, a execução e simulação de um modelo de teste que dê suporte ao paralelismo, à concorrência, à representação de fluxo de controle e principalmente a estruturas de dados na forma de objetos que as redes de Petri orientada a objetos permitem representar.

O modelo de teste funcional proposto em [Val09] não é voltado somente para a verificação das funcionalidades do software do ponto de vista de ações externas do usuário, mas também pode ser usado para avaliar o comportamento interno de certas partes do software, como a atualização de um banco de dados, por exemplo, já que a classe de testes pode oferecer mecanismos para que o procedimento ocorra.

O trabalho mostra uma aplicação das Redes de Petri de forma a auxiliar na execução dos testes funcionais. Talvez uma boa contribuição para o trabalho seria adicionar medidas de desempenho, bem como associar ao processo de testes mapeado em um modelo formal como mostra nessa dissertação para avaliação tanto do processo de testes como dos resultados obtidos nos testes executados.

2.2 CONSIDERAÇÕES FINAIS

Este capítulo apresentou alguns dos trabalhos relacionados a esta dissertação. Dentre estes, Oliveira et al [OLR09] propõem um modelo de avaliação de desempenho para os processos de negócios utilizando Redes de Petri Estocásticas Generalizadas (GSPN); Jeetendra et al [JKCPR00] apresenta as vantagens de utilizar Redes de Petri em gerenciamento de projeto; Schmietendorf et al [SSR00b] apresenta um modelo para avaliar o desempenho dos processos de softwares; Nóbrega et al [Nób08] apresenta um modelo para avaliação e melhoria de desempenho de equipes de testes; Vale et al [Val09] apresenta uma metodologia de desenvolvimento de modelos de teste funcional utilizando redes de Petri orientada a objetos; finalmente, Bertolino et al [BMM02] apresenta uma proposta de metodologia baseada no uso de técnicas da *Performance Engineering* para apoio na gestão de pessoas realizando um estudo de caso no processo de testes para estabelecer estimativas de entregas do produto.

Nesta dissertação, é apresentada uma metodologia como também um modelo voltado para avaliação de desempenho dos processos de testes. A partir dos modelos SPN, é possível verificar as melhores composições de equipes em termos de desempenho, tempo e custo, bem como realizar avaliações para verificar as melhores estimativas. Também é verificada a qualidade do processo de testes, através do modelo SPN gerado.

CAPÍTULO 3

REDES DE PETRI

Este capítulo apresenta os principais conceitos sobre redes de Petri (*Petri Nets* - PNs), assim como características, propriedades e técnicas de análise. Em seguida, são apresentadas as redes de Petri estocásticas (*Stochastic Petri Nets* - SPNs), que são uma extensão à teoria inicial das redes de Petri. Finalmente, são introduzidos o *moment matching* e a técnica de aproximação de fases.

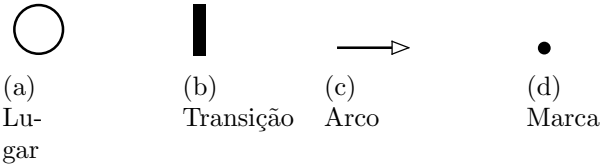
3.1 INTRODUÇÃO

O conceito de redes de Petri foi introduzido por Carl Adam Petri, no ano de 1962, com a apresentação da sua tese de doutorado "*Kommunikation mit Automaten*" (comunicação com autômatos) [Mur89] na faculdade de Matemática e Física da Universidade Darmstadf na Alemanha. Redes de Petri são ferramentas gráficas e matemáticas usadas para descrição formal de sistemas caracterizados pelas propriedades de concorrência, paralelismo, sincronização, distribuição, assincronismo e não-determinismo.

A aplicabilidade das Redes de Petri como ferramenta para estudo de sistemas é importante por permitir representação matemática, análise dos modelos e também por fornecer informações úteis sobre a estrutura e o comportamento dinâmico dos sistemas modelados. As aplicações das Redes de Petri podem se dar em muitas áreas (sistemas de manufatura, desenvolvimento de *software*, sistemas administrativos, entre outros).

As redes de Petri são formadas por lugares (Figura 3.1(a)), transições (Figura 3.1(b)), arcos (Figura 3.1(c)) e marcas (Figura 3.1(d)). Os lugares correspondem às variáveis de estado e às transições, às ações ou eventos realizados pelo sistema. A realização de uma ação está associada a algumas pré-condições, ou seja, existe uma relação entre os lugares e as transições que possibilitam ou não a realização de uma determinada ação. Após a realização de uma determinada ação, alguns lugares terão suas informações alteradas, ou seja, a ação criará uma pós-condição. Os arcos representam o fluxo das marcas pela rede de Petri, e as marcas representam o estado em que o sistema se encontra em determinado momento. Graficamente, os lugares são representados por elipses ou círculos; as transições, por retângulos, os arcos, por setas e as marcas, por meio de pontos. A Figura 3.2 mostra um exemplo de rede de Petri.

Os dois elementos, lugar e transição, são interligados por meio de arcos dirigidos. Os arcos que interligam lugares às transições (Lugar $\longrightarrow$ Transição) correspondem à relação entre as condições verdadeiras (pré-condição), que possibilitam a execução das ações. Os arcos que interligam as transições aos lugares (Transição $\longrightarrow$ Lugar) representam a relação entre as ações e as condições que se tornam verdadeiras com a execução das ações



(a) Lugar (b) Transição (c) Arco (d) Marca

Figura 3.1: Elementos de rede de Petri

(pós-condição).

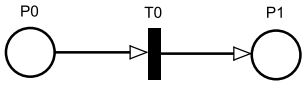
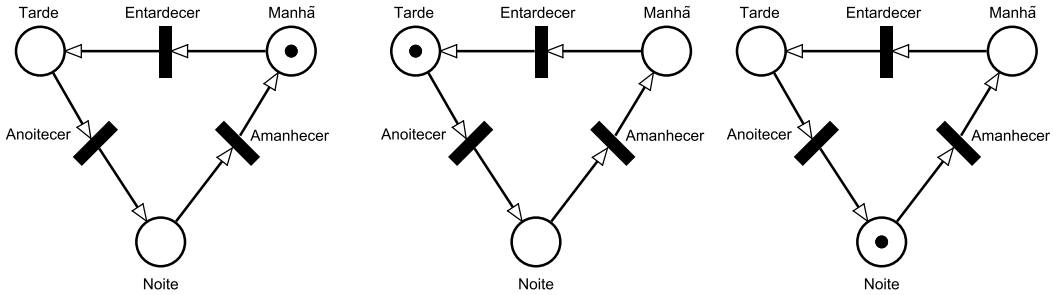


Figura 3.2: Exemplo de rede de Petri

A Figura 3.3 [MLC96] apresenta o ciclo repetitivo dos períodos do dia através de um modelo PN. Um dia pode ser dividido em três períodos distintos: manhã, tarde e noite. As transições entre os períodos do dia devem respeitar algumas condições. O período anterior à tarde é a manhã, o anterior à noite é a tarde e assim por diante. Dessa forma, podem ser estabelecidas as pré-condições e as pós-condições. Para modelar esse sistema através de uma rede de Petri são necessários três lugares, os quais representam os três períodos do dia, e três transições, as quais representam as três mudanças de período do dia. Esse modelo tem o seu estado inicial com uma marca (ponto) no lugar Manhã (Figura 3.3(a)). Com essa marcação, o único evento possível de ocorrer é o evento entardecer, representado pela transição entardecer. Após a execução desse evento, é depositada uma marca no lugar Tarde (Figura 3.3(b)). Com uma marca no lugar Tarde, o próximo evento que ocorrerá é anoitecer, representado pela transição anoitecer. A execução desse evento deposita uma marca no lugar Noite (Figura 3.3(c)). Uma marca no lugar Noite possibilita a ocorrência do evento amanhecer, este representado pela transição amanhecer. A execução desse evento reinicia o processo.



(a) Dia (b) Tarde (c) Noite

Figura 3.3: Períodos do Dia

A representação formal de um modelo PN é a quíntupla $PN = \{P, T, F, W, \mu_0\}$, onde:

- P é o conjunto finito de lugares;

- T é o conjunto finito de transições, $P \cap T = \emptyset$;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de arcos;
- $W : F \rightarrow IR^+ \cup \{0\}$ é a função de atribuição de peso aos arcos;
- $\mu_0 : P \rightarrow IN$ é a função de marcação inicial, onde $P \cap T = \varnothing$ e $P \cup T \neq \varnothing$.

3.1.1 Propriedades das Redes de Petri

O estudo das propriedades de redes de Petri permite a análise do sistema modelado. Os tipos de propriedades podem ser divididos em duas categorias: as propriedades dependentes de marcação inicial, conhecidas como propriedades comportamentais, e as propriedades não dependentes de marcação, conhecidas como propriedades estruturais [MLC96, Mur89].

3.1.1.1 Propriedades Comportamentais

As propriedades comportamentais são aquelas que dependem apenas da marcação inicial da rede de Petri. As propriedades abordadas são alcançabilidade, limitação, segurança, liveness e cobertura.

Alcançabilidade ou *reachability* indica a possibilidade de uma determinada marcação ser atingida pelo disparo de um número finito de transições a partir de uma marcação inicial. Dada uma rede de Petri marcada $RM = (R; M_0)$, o disparo de uma transição t_0 altera a marcação da rede. Uma marcação M' é acessível a partir de M_0 se existe uma sequência de transições que, disparadas, levam à marcação M'. Ou seja, se a marcação M_0 habilita a transição t_0, disparando-se esta transição, atinge-se a marcação M_1. A marcação M_1 habilita t_1 a qual, sendo disparada, atinge-se a marcação M_2 e assim por diante até a obtenção da marcação M'.

Seja $M_i[t_j > M_k$ e $M_k[t_h > M_1$, então $M_i[t_j t_h > M_1$, por transitividade, o disparo de uma sequência $s \in T^*$ é designado por $M[s > M'$. O conjunto de todas as possíveis marcações obtidas a partir da marcação M_0 na rede $RM = (R; M_0)$ é denotado por $CA(R; M_0) = \{M' \in IN^m | \exists s, M_0[s > M'\}$, onde m é a cardinalidade do conjunto de lugares da rede.

A análise da alcançabilidade de uma marcação consiste em determinarmos uma dada marcação $M' \in CA(R; M_0)$ da rede marcada RM. Em alguns casos, deseja-se observar apenas alguns lugares específicos da rede em estudo. Esse problema é denominado submarcação alcançável.

Seja um lugar $p_i \in P$, de uma rede de Petri marcada $RM = (R; M_0)$, esse lugar é k-limitado ($k \in IN$) ou simplesmente limitado se para toda marcação acessível $M \in CA(R; M_0)$, $M(p_i) \leq k$.

O limite k é o número máximo de marcas que um lugar pode acumular. Uma rede de

Petri marcada $RM = (R; M_0)$ é k-limitada se o número de marcas de cada lugar de RM não exceder k em qualquer marcação acessível de $RM(max(M(p)) = k, \forall p \in P)$.

Segurança ou *safeness* é uma particularização da propriedade de limitação. O conceito de limitação define que um lugar p_i é k-limitado se o número de marcas que esse lugar pode acumular estiver limitado ao número k. Um lugar que é 1-limitado pode ser simplesmente chamado de seguro.

Seja $p_i \in P$ um lugar de uma rede de Petri marcada $RM = (R; M_0)$, p_i é seguro se para toda marcação $M' \in CA(R; M_0), M(p_i) \leq 1$. Uma rede é segura se todos os lugares pertencentes a essa rede forem seguros, ou seja, todos os lugares dessa rede podem conter no máximo uma única marca.

Vivacidade ou *liveness* está definida em função das possibilidades de disparo das transições. Uma rede é considerada *live* se, independentemente das marcações que sejam alcançáveis a partir de M_0, for sempre possível disparar qualquer transição da rede através de uma sequência de transições $L(M_0)$. A ausência de bloqueio (*deadlock*) em sistemas está fortemente ligada ao conceito de vivacidade, pois *deadlock* em uma rede de Petri é a impossibilidade do disparo de qualquer transição da rede. O fato de um sistema ser livre de *deadlock* não significa que seja *live*, entretanto um sistema *live* implica um sistema livre de *deadlocks*.

Uma rede $RM = (R; M_0)$ é viva (*live*) se para toda $M \in CA(R; M_0)$ for possível disparar qualquer transição de RM através do disparo de alguma sequência de transições.

O conceito de cobertura está associado ao conceito de alcançabilidade e *live*. Uma marcação M_i é coberta se existir uma marcação $M_j \neq M_i$, tal que $M_j \geq M_i$.

3.1.1.2 Propriedades Estruturais

As propriedades estruturais são aquelas que dependem apenas da estrutura da rede de Petri. Essas propriedades refletem características independentes de marcação. As propriedades analisadas neste trabalho são limitação estrutural e consistência.

Uma rede de Petri $R = (P, T, F, W, \mu_0)$ é classificada como estruturalmente limitada se for limitada para qualquer marcação inicial.

Ela será considerada consistente se, disparando uma sequência de transições habilitadas a partir de uma marcação M_0, retornar a M_0, porém todas as transições da rede são disparadas pelo menos uma vez.

Seja $RM = (R; M_0)$ uma rede marcada e s uma sequência de transições, RM é consistente se $M_0[s > M_0$ e toda transição T_i, disparar pelo menos uma vez em s.

Os métodos de análise das propriedades das redes de Petri são classificados como análise baseada na geração do espaço de estados, análise baseada na equação de estado, métodos baseados na estrutura da rede (análise de invariantes) e técnicas de redução. A validação dos modelos pode ser realizada através de simulação [MLC96, Mur89].

O método de análise baseada na geração do espaço de estados envolve essencialmente

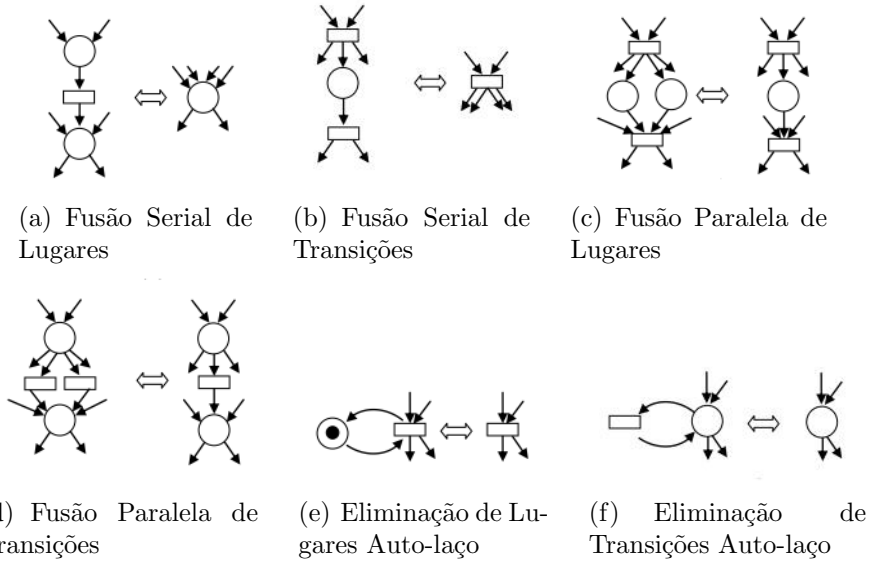
a enumeração das marcações alcançáveis (marcações cobertas). Esse método é aplicável a todas as classes de redes, mas é limitado a redes pequenas devido à complexidade do problema de explosão de espaço de estados. Esse método baseia-se na construção de um grafo que representa todas as marcações que a rede de Petri pode alcançar. Cada nó corresponde a uma marcação, e cada arco corresponde ao disparo de um conjunto não vazio de transições. Se a rede de Petri for limitada, é possível construir este tipo de grafo e, nesse caso, ele denomina-se grafo de ocorrências. Caso a rede de Petri não seja limitada, o grafo de ocorrências é infinito. Nesse caso, ainda é possível construir um grafo que se denomina grafo de cobertura [MLC96, Mur89].

O método de análise baseada na equação de estado ou equação fundamental possibilita a verificação da acessibilidade das marcações, assim como o número de vezes que cada transição tem que ser disparada para atingir determinada marcação. Uma rede de Petri pode ser representada por duas matrizes, uma indicando os conjuntos de lugares que servem de entrada para cada uma das transições da rede e outra indicando os conjuntos de lugares que servem de saída para tais transições. A primeira delas é a matriz de entrada, também chamada de matriz de incidência reversa. A matriz de incidência A de uma rede de Petri é uma matriz $n \times m$ de inteiros, definida como $A = [a_{ij}]$ e $a_{ij} = a_{ij}^{+} - a_{ij}^{-}$, onde $a_{ij}^{+} = w(i,j)$ é o peso do arco da transição i para seu lugar de saída j e $a_{ij}^{-} = w(i,j)$ é o peso do arco do lugar de entrada j para a transição i [MLC96, Mur89].

O método baseado na estrutura da rede (análise de invariantes) verifica a existência de componentes repetitivos estacionários nos modelos, onde esses componentes correspondem a comportamentos cíclicos da rede. Os invariantes em uma rede de Petri representam os componentes conservativos e repetitivos da rede. Há conjuntos de lugares e de transições da rede, cujo comportamento não se altera durante o seu funcionamento. A identificação e a interpretação de cada um destes conjuntos são importantes, pois eles refletem certas propriedades da rede que podem ser de interesse para a análise do sistema modelado. Os componentes conservativos da rede são representados em seus invariantes de lugar, ou seja, são conjuntos de lugares da rede nos quais a soma das marcas é constante durante todo o seu funcionamento. Os componentes repetitivos são representados em seus invariantes de transição, isto é, são conjuntos de transições da rede que, ao serem disparadas em determinada sequência, retornam à marcação de partida [MLC96, Mur89].

As técnicas de redução são transformações aplicadas ao modelo de um sistema com o objetivo de simplicá-lo, preservando as propriedades do sistema analisado. Normalmente essas técnicas são utilizadas para facilitar a análise de sistemas complexos. Essas técnicas são baseadas nas transformações de redes originais em um modelo mais abstrato de tal maneira que as propriedades como *liveness*, *boundedness* e *safeness* são preservadas nos modelos obtidos por estas reduções. A transformação reversa (refinamento) pode ser usada para processos de síntese. As regras de transformação das redes podem ser obtidas a partir de aplicação das fusões, tanto de lugares, quanto de transições [Mur89]. A Figura 3.4 apresenta algumas das técnicas de redução [MLC96, Mur89].

A simulação é utilizada quando o sistema é relativamente complexo e sua análise através de outros métodos analíticos se mostra inviável.



(a) Fusão Serial de Lugares

(b) Fusão Serial de Transições

(c) Fusão Paralela de Lugares

(d) Fusão Paralela de Transições

(e) Eliminação de Lugares Auto-laço

(f) Eliminação de Transições Auto-laço

Figura 3.4: Técnicas de Redução

3.2 REDE DE PETRI ESTOCÁSTICA

Rede de Petri estocástica (SPN) [Ger00] é uma das extensões de rede de Petri (PN) [Mur89] utilizada para a modelagem de desempenho. Uma rede de Petri estocástica adiciona tempo ao formalismo de redes de Petri, com a diferença de que os tempos associados às transições temporizadas são distribuídos exponencialmente, enquanto o tempo associado às transições imediatas é zero. As transições temporizadas modelam atividades através dos tempos associados, de modo que o período de habilitação da transição temporizada corresponde ao período de execução da atividade, e o disparo da transição temporizada corresponde ao término da atividade. Níveis diferentes de prioridade podem ser atribuídos às transições. A prioridade de disparo das transições imediatas é superior à das transições temporizadas. As prioridades podem solucionar situações de confusão [MBC+98]. As probabilidades de disparo associadas às transições imediatas podem solucionar situações de conflito [Bal01, MBC+98].

Uma SPN é definida pela 9-tupla $SPN = \{P, T, I, O, H, \Pi, G, M_0, Atts\}$, onde:

- $P = \{p_1, p_2, ..., p_n\}$ é o conjunto de lugares;
- $T = \{t_1, t_2, ..., t_m\}$ é o conjunto de transições imediatas e temporizadas, $P \cap T = \emptyset$;
- $I \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de entrada (que podem ser dependentes de marcações);
- $O \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ é a matriz que representa os arcos de saída (que podem ser dependentes de marcações);
- $H \in (\mathbb{N}^n \rightarrow \mathbb{N})^{n \times m}$ é a matriz que representa os arcos inibidores (que podem ser dependentes de marcações);

- $\Pi \in \mathbb{N}^m$ é um vetor que associa o nível de prioridade a cada transição;
- $G \in (\mathbb{N}^n \rightarrow \{true, false\})^m$ é o vetor que associa uma condição de guarda relacionada a marcação do lugar a cada transição;
- $M_0 \in \mathbb{N}^n$ é o vetor que associa uma marcação inicial de cada lugar (estado inicial);
- $Atts = (Dist, Markdep, Policy, Concurrency, W)^m$ compreende o conjunto de atributos associados às transições, onde:
  - $Dist \in \mathbb{N}^m \rightarrow \mathcal{F}$ é uma possível função de distribuição de probabilidade associada ao tempo de uma transição (esta distribuição pode ser dependente de marcação) (o domínio de $\mathcal{F}$ é $[0, \infty)$);
  - $Markdep \in \{constante, enabdep\}$, onde a distribuição de probabilidade associada ao tempo de uma transição pode ser independente (*constante*) ou dependente de marcação (*enabdep*− a distribuição depende da condição de habilitação atual);
  - $Policy \in \{prd, prs\}$ define a política de memória adotada pela transição (*prd*− *preemptive repeat different*, valor padrão, de significado idêntico à *race enabling policy*; *prs*− *preemptive resume*, corresponde ao *age memory policy*);
  - $Concurrency \in \{ss, is\}$ é o grau de concorrência das transições, onde *ss* representa a semântica *single server* e *is* representa a semântica *infinity server*.
  - $W : T \rightarrow IR^+ \cup \{0\}$ é a função peso, que representa o peso (w_t) de transições imediatas e a taxa λ_t de transições temporizadas, onde:

    $$\pi(t) = \begin{cases} \geq 1, & \text{se t é uma transição imediata;} \\ 0, & \text{caso contrário.} \end{cases}$$

    Se t é uma transição temporizada, então λ_t será o valor do parâmetro da função densidade probabilidade exponencial;

    Se t é uma transição imediata, então W_t será um peso, que é usado para o cálculo das probabilidades de disparo das transições imediatas em conflitos.

    Os arcos inibidores são usados para prevenir transições de serem habilitadas quando certa condição é verdadeira.

Os modelos SPN possuem dois tipos de estados (marcações), os estados tangíveis (*tangible*) e os estados voláteis (*vanish*). Os estados voláteis são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição imediata. O termo *vanish* é usado porque as marcações chegam a esses lugares e são instantaneamente consumidas. O tempo de permanência das marcações nesses lugares é zero. Os estados tangíveis são criados em decorrência da marcação dos lugares que são pré-condições de habilitação de uma transição temporizada [MBC+98].

As transições temporizadas podem ser caracterizadas por diferentes políticas de memória tais como *Resampling*, *Enabling memory* e *Age memory* [MBC+98].

Resampling: A cada disparo de toda e qualquer transição do modelo, todos os temporizadores existentes são reiniciados (*Restart*), e, sendo assim, não há memória. O temporizador de cada transição será reiniciado sempre que a transição tornar-se habilitada;

Enabling memory: A cada disparo de transição, os temporizadores das transições que estavam desabilitadas são reiniciados, enquanto os temporizadores das transições que estavam habilitadas mantêm o valor atual (*Continue*). Assim que essas transições se tornarem habilitadas novamente, seus temporizadores continuam do ponto onde foram parados. Uma variável (*enabling memory variable*) mede o tempo que a transição passou habilitada desde o último instante de tempo em que ela se tornou habilitada;

Age memory: Após cada disparo, os temporizadores de todas as transições mantêm seus valores atuais (*Continue*). Uma memória do passado é mantida por uma variável (*age memory variable*) associada a cada transição temporizada. Esta variável contabiliza o tempo gasto na atividade modelada pela transição, medindo o tempo cumulativo de habilitação, desde o instante do seu último disparo.

As transições temporizadas podem ser caracterizadas por diferentes semânticas de disparo conhecidas como *single server*, *multiple server* e *infinite server* [MBC+98].

Na semântica *single server*, as marcações são processadas serialmente. Após o primeiro disparo da transição temporizada, o temporizador é reiniciado como se a transição temporizada tivesse sido habilitada novamente. Esse tipo de semântica é utilizada nos modelos de disponibilidade, considerando-se que haja apenas uma única equipe de manutenção, quando vários componentes do sistema entram numa condição de falha;

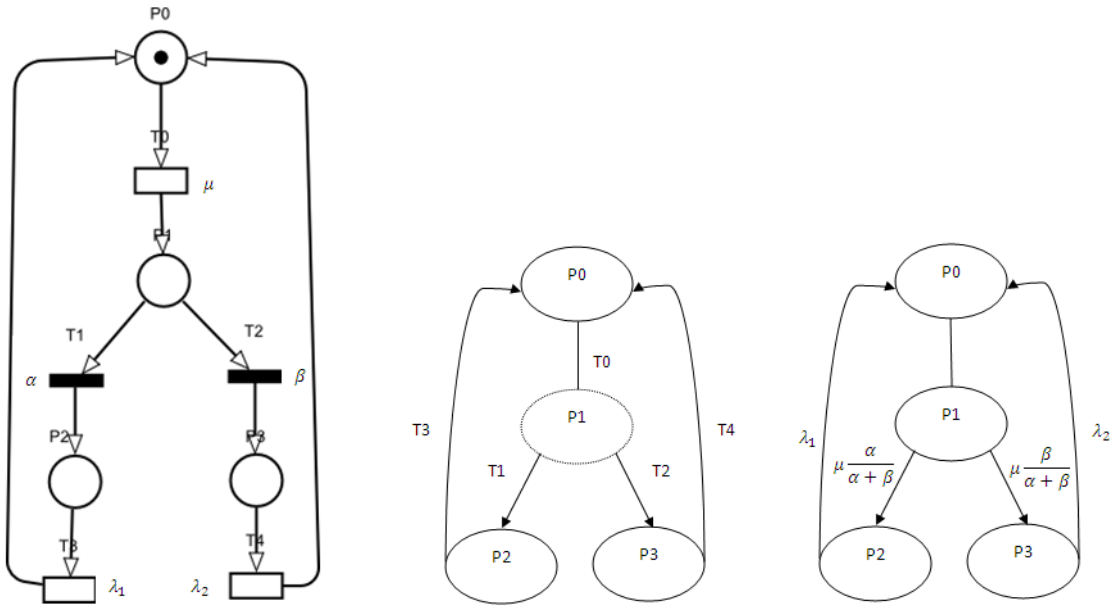
Na semântica *multiple server*, as marcações são processadas com um grau máximo K de paralelismo. Caso o grau de habilitação seja maior do que K, não será criado nenhum novo temporizador para processar o tempo para o novo disparo até que o grau de habilitação tenha diminuído abaixo de K. Esse tipo de semântica é utilizado nos modelos de disponibilidade considerando-se que haja um número de equipes de manutenção menor do que o número de componentes na condição de falha. Os componentes em excesso ficarão em fila;

Na semântica *infinite server*, o valor de K é infinito, todas as marcações são processadas em paralelo, e as temporizações associadas são decrementadas a zero em paralelo. Esse tipo de semântica é utilizada nos modelos de disponibilidade, considerando-se que haja tantas equipes de manutenção quantos sejam os componentes em falha. Para cada componente existe uma equipe de manutenção exclusiva e independente. Nesse tipo de semântica, todas as marcações são processadas em paralelo.

Nos modelos SPN, as transições são disparadas obedecendo à semântica *interleaving* de ações [MBC+98]. Essa semântica define que as transições são disparadas uma a uma, mesmo que o estado compreenda transições imediatas não conflitantes. A análise de um modelo SPN requer a solução de um sistema de equações igual ao número de marcações tangíveis. O gerador infinitesimal Q da cadeia de Markov de tempo contínuo (CTMC) associado ao modelo SPN é derivado de uma redução de um gráfico de alcançabilidade,

rotulado com as taxas das transições temporizadas ou pesos das transições imediatas.

Modelos SPN permitem a geração de gráficos de alcançabilidade a partir dos quais cadeias de Markov de tempo contínuo (CTMC) são diretamente derivadas. A Figura 3.5 apresenta uma exemplo de geração de gráfico de alcançabilidade a partir de um modelo SPN. No modelo SPN mostrado na Figura 3.5(a), existe um conflito entre duas transições imediatas ($T1$ e $T2$). A Figura 3.5(b) mostra o gráfico de alcançabilidade com a indicação de que o estado $P1$ é volátil. O disparo da transição temporizada $T0$ torna o lugar $P1$ marcado, habilitando as duas transições imediatas, $T1$ e $T2$, gerando o estado $P1$. Há uma mudança imediata (tempo zero) para o estado $P2$ ou $P3$, através do disparo da transição imediata $T1$ ou $T2$, com probabilidades $\frac{\alpha}{\alpha+\beta}$ e $\frac{\beta}{\alpha+\beta}$, respectivamente. A Figura 3.5(c) mostra o gráfico de alcançabilidade tangível após a eliminação do estado volátil $P1$.



(a) Modelo SPN (b) Gráfico de Alcançabilidade (c) Gráfico de Alcançabilidade Tangível

Figura 3.5: Geração de Gráfico de Alcançabilidade

A taxa na qual o sistema se move do estado $P0$ para $P2$ ou $P3$ é obtida pelo produto da taxa λ da transição do estado $P0$ para o estado volátil $P1$, com a probabilidade de ir do estado $P1$ para o estado $P2$ ou $P3$.

Redes de Petri estocásticas marcadas, com um número finito de lugares e transições, são isomórficas as cadeias de Markov [Mur89]. O isomorfismo de um modelo SPN com uma cadeia de Markov é obtido a partir do gráfico de alcançabilidade reduzido, que é dado através da eliminação dos estados voláteis e rótulo dos arcos com as taxas das transições temporizadas e pesos das transições imediatas. As medições de desempenho e dependabilidade são obtidas através de simulações e análises em estado estacionário e

transiente baseadas na cadeia de Markov embutida no modelo SPN [BGdMT06].

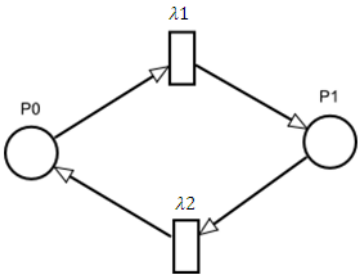
Os modelos SPN são usados para análise de desempenho de sistemas, visto que permitem a descrição das atividades de sistemas através de gráficos de alcançabilidade. Esses gráficos podem ser convertidos em modelos Markovianos, que são utilizados para avaliação quantitativa do sistema analisado.

3.3 MOMENT MATCHING

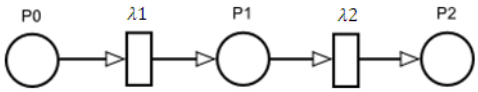
Modelos SPN consideram somente transições imediatas e transições temporizadas com tempos de disparo distribuídos exponencialmente. Essas transições modelam ações, atividades e eventos.

Uma variedade de atividades pode ser modelada através do uso dos construtores *throughput subnets* e *s-transitions*. Esses construtores são utilizados para representar distribuições expolinomiais, tais quais as distribuições Erlang, Hipoexponencial e Hiperexponencial [DAJ95].

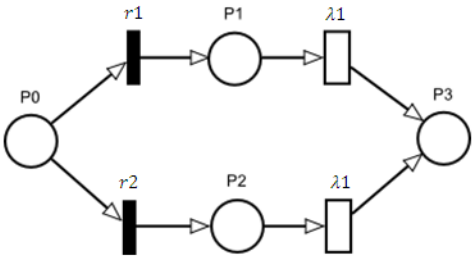
Combinações de lugares, transições exponenciais e transições imediatas podem ser usadas entre dois lugares para representar diferentes tipos de distribuições. As Figuras 3.6(a), 3.6(b) e 3.6(c) representam três *throughput subnets*.



(a) Conexão paralela



(b) Conexão série



(c) Conexão paralela

Figura 3.6: *Throughput Subnets*

A Figura 3.6(a) descreve uma *throughput subnet* formada por duas transições expo-

nenciais em paralelo com taxas λ_1 e λ_2, respectivamente.

Uma marcação no lugar $P0$ aparecerá no lugar $P1$ após o disparo de uma das transições exponenciais que estão em paralelo as quais têm tempos associados τ_1 e τ_2, respectivamente, (Equação (3.1)). A função de densidade para esses tempos é dada pela Equação (3.2).

$$\tau = min(\tau_1 + \tau_2) \tag{3.1}$$

$$f_\tau(t) = (\lambda_1 + \lambda_2)\exp^{(\lambda_1+\lambda_2)t}, t \geq 0 \tag{3.2}$$

Essas transições exponenciais em paralelo são equivalentes a uma transição exponencial com taxa $\lambda_1 + \lambda_2$.

A Figura 3.6(b) descreve uma *throughput subnet* formada por duas transições exponenciais em série com os parâmetros λ_1 e λ_2, respectivamente. Uma marcação no lugar $P0$ aparecerá no lugar $P2$ após o disparo das transições exponenciais, as quais têm um tempo associado $\tau = \tau_1 + \tau_2$, cuja função de densidade é dada pela Equação (3.3).

$$f_\tau(t) = (f_{\tau 1} * f_{\tau 2})(t) = \frac{\lambda_1\lambda_2(\exp^{-\lambda_1 t} - \exp^{-\lambda_2 t})}{\lambda_2 - \lambda_1}, t \geq 0 \tag{3.3}$$

* é o operador de convolução. Para o caso onde λ_1, $\lambda_2 = ... = \lambda_n$, a função densidade é dada pela Equação (3.4).

$$f_\tau(t) = \frac{\lambda^n t^{n-1}\exp^{-\lambda t}}{(n-1)!}, t > 0 \tag{3.4}$$

Essa expressão representa uma distribuição do tipo Erlang de ordem N. Uma distribuição do tipo Erlang é especificada por dois parâmetros $\lambda > 0$ e $n > 0$.

A Figura 3.6(c) descreve uma *throughput subnet* formada por duas sub-redes paralelas, cada uma contendo uma transição imediata e uma transição exponencial. Uma marcação no lugar $P0$ aparecerá no lugar $P3$ após o disparo das transições imediatas e exponenciais em cada sub-rede. A probabilidade de cada sub-rede é determinada pelos pesos r_1 e r_2 das transições imediatas. A função de densidade dos tempos associados às transições exponenciais é dada pela Equação (3.5), que é uma distribuição hiperexponencial.

$$f_\tau(t) = r_1 f_\tau(t) + r_2 f_\tau(t) = r_1\lambda_1^{\exp^{-\lambda_1 t}} + r_2\lambda_2^{\exp^{-\lambda_2 t}}, t > 0 \tag{3.5}$$

Essa *throughput subnet* implementa uma função de densidade com tempo hiperexponencial, cuja distribuição hiperexponencial é descrita pela Equação (3.6).

n, a ordem

$r_j, j = 1...n$,

$\lambda_j, j = 1...n.$

$$\sum r_j = 1 \tag{3.6}$$

A técnica de aproximação de fases pode ser aplicada para modelar ações, atividades e eventos não-exponenciais através do *moment matching*. O método apresentado calcula o primeiro momento em torno da origem (média) e o segundo momento central (variância) e estima os momentos respectivos da *s-transition* [DAJ95].

Dados de desempenho medidos ou obtidos de um sistema (distribuição empírica) com média μ_D e desvio-padrão σ_D podem ter seu comportamento estocástico aproximados através da técnica de aproximação de fases. O inverso do coeficiente de variação dos dados medidos (Equação (3.7)) permite a seleção da distribuição expolinomial que melhor se adapta à distribuição empírica.

$$\frac{1}{CV} = \frac{\mu_D}{\sigma_D} \tag{3.7}$$

A rede de Petri descrita na Figura 3.7 representa uma atividade com distribuição de probabilidade genérica.

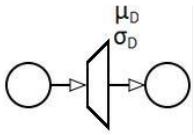


Figura 3.7: Distribuição Empírica

Dependendo do valor do inverso do coeficiente de variação dos dados medidos (Equação (3.7)), a respectiva atividade tem uma dessas distribuições atribuídas: Erlang, Hipoexponencial ou Hiperexponencial.

Quando o inverso do coeficiente de variação é um número inteiro e diferente de um, os dados devem ser caracterizados através da distribuição Erlang, que é representada por uma sequência de transições exponenciais, cujo tamanho é calculado através da Equação (3.8). A taxa de cada transição exponencial é calculada através da Equação (3.9). Os modelos de Redes de Petri descritos na Figura 3.8 representam uma distribuição Erlang.

$$\gamma = \left(\frac{\mu}{\sigma}\right)^2 \tag{3.8}$$

$$\lambda = \frac{\gamma}{\mu} \tag{3.9}$$

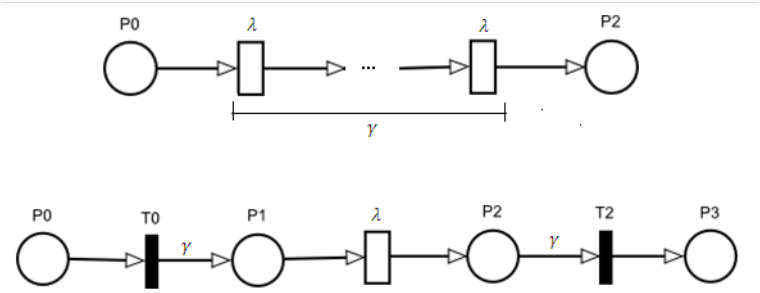


Figura 3.8: Distribuição Erlang

Quando o inverso do coeficiente de variação é um número maior que um (mas não é um número inteiro), os dados são representados através da distribuição hipoexponencial, a qual é representada por uma sequência de transições exponenciais, cujo tamanho é calculado através da Equação (3.10). As taxas das transições exponenciais são calculadas através das Equações (3.11) e (3.12), e os tempos médios atribuídos às transições exponenciais são calculados através das Equações (3.13) e (3.14). Os modelos de Redes de Petri apresentados na Figura 3.9 descrevem uma distribuição hipoexponencial.

$$(\frac{\mu}{\sigma})^2 - 1 \leq \gamma < (\frac{\mu}{\sigma})^2 \tag{3.10}$$

$$\lambda_1 = \frac{1}{\mu_1} \tag{3.11}$$

$$\lambda_2 = \frac{1}{\mu_2} \tag{3.12}$$

$$\mu_1 = \mu \mp \frac{\sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{3.13}$$

$$\mu_2 = \gamma\mu \pm \frac{\sqrt{\gamma(\gamma+1)\sigma^2 - \gamma\mu^2}}{\gamma+1} \tag{3.14}$$

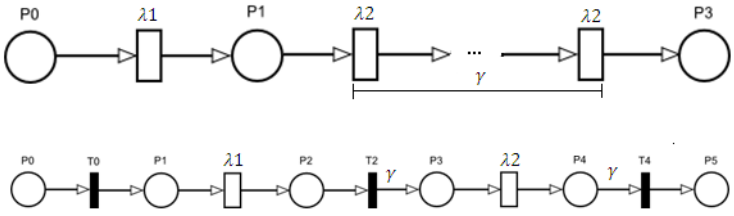


Figura 3.9: Distribuição Hipoexponencial

Quando o inverso do coeficiente de variação é um número menor que um, os dados devem ser caracterizados através de uma distribuição hiperexponencial. A taxa da transição exponencial deve ser calculada através da Equação (3.15), e os pesos das transições imediatas são calculados através das Equações (3.16) e (3.17). O modelo de Redes de Petri que representa essa distribuição hiperexponential é descrito na Figura 3.10.

$$\lambda_h = \frac{2\mu}{\mu^2 + \sigma^2} \tag{3.15}$$

$$r_1 = \frac{2\mu^2}{\mu^2 + \sigma^2} \tag{3.16}$$

$$r_2 = 1 - r_1 \tag{3.17}$$

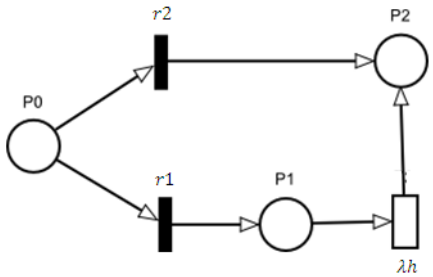


Figura 3.10: Distribuição Hiperexponencial

3.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou uma introdução sobre redes de Petri, assim como definições, conceitos básicos e propriedades, as quais podem ser divididas em duas categorias: propriedades comportamentais e propriedades estruturais. Em seguida, foram apresentadas as redes de Petri estocásticas (SPNs), que são de particular interesse deste trabalho. As SPNs possuem transições com tempos exponenciamente distribuídos e transições imediatas. Finalmente, foram apresentados o *moment matching* e a técnica de aproximação de fases, que são técnicas para aproximar o comportamento estocástico de medidas não-exponenciais.

CAPÍTULO 4

METODOLOGIA DE AVALIAÇÃO

Este capítulo apresenta uma metodologia para avaliação de desempenho de processos de testes de software. Inicialmente, apresenta-se uma breve introdução dos fundamentos de planejamento de projetos de software e dos testes de software que dão apoio teórico à metodologia, posteriormente a metodologia e sua etapas são apresentadas.

4.1 CONTEXTO

A metodologia de avaliação de desempenho apresentada neste Capítulo na Seção *Metodologia de Avaliação de Desempenho* é voltada para criação de modelos estocásticos para previsão de estimativas (tempo, custo, qualidade) das atividades do processo de testes. Para isso, como fundamentação teórica, nas próximas subseções é apresentada uma breve introdução sobre planejamento de projetos de software e dos testes de softwares.

4.1.1 Planejamento de Projetos

Atualmente, mudanças em diversos aspectos da vida humana (culturais, tecnológicos, políticos, econômicos, sociais, etc) estão ocorrendo em velocidade cada vez maior. De uma maneira geral, é comum associarmos as mudanças significativas ao resultado de projetos [Vie02]. Como conseqüência, gerenciar projetos de forma eficiente nessa era de grandes mudanças é um dos grandes desafios do executivo dos tempos modernos [Ker09]. Superar este desafio é estar preparado para gerenciar projetos de forma planejada e profissional. Os próximos parágrafos apresentam uma visão geral do planejamento do projeto e das atividades de planejamento de tempo, custo e recursos.

A ausência de um processo de gerenciamento apropriado, aliado a estimativas deficientes de custo e de tempo, é uma das principais causas das falhas dos projetos de software [Jon98]. Segundo o PMI [Ins08] um projeto é um empreendimento único, com início e fim definidos, que utiliza recursos limitados e é conduzido por pessoas, visando atingir metas e objetivos pré-definidos estabelecidos dentro de parâmetros de prazo, custo e qualidade [Ins08].

Gerenciar consiste em executar atividades e tarefas que têm como propósito planejar e controlar atividades de outras pessoas para atingir objetivos que não podem ser alcançados caso as pessoas atuem por conta própria, sem o esforço sincronizado dos subordinados [KD82].

Segundo Vidger [VK94] os gerentes de projeto estão desacostumados a estimar. Boas estimativas de custo, esforço e prazo de software requerem um processo ordenado que

defina a utilização de métricas de software, método e ferramenta de estimativas.

O bom planejamento é essencial para o sucesso de um projeto. A natureza intangível do software causa problemas para o gerenciamento. Gerentes têm várias atividades, sendo as mais importantes o planejamento e a elaboração de estimativas e de cronogramas. Essas atividades são processos interativos que continuam ao longo do curso do projeto.

No planejamento de projetos o tempo, o custo e os recursos humanos são itens necessários que devem ser verificados pelos gerentes. Segundo o PMI [Ins08] a gerência de tempo do projeto objetiva garantir o término do projeto no tempo certo. Consiste da definição, ordenação e estimativa de duração das atividades, e de elaboração e controle de cronogramas. A gerência de custo tem por objetivo garantir que o projeto seja executado dentro do orçamento aprovado. Consiste no planejamento dos recursos, estimativa, orçamento e controle de custos.

O planejamento de recursos humanos é usado para determinar e identificar quais recursos humanos possuem habilidades necessárias para o êxito do projeto. No planejamento é importante considerar a disponibilidade de recursos humanos escassos ou limitados, ou a concorrência por eles. Considerando esses fatores, os custos do projeto, cronogramas, riscos, qualidade e outras áreas podem ser significativamente alteradas. Um planejamento de recursos humanos eficaz deve considerar e planejar esses fatores, e desenvolver opções dos recursos.

4.1.2 Testes de Software

Nas décadas de 60 e 70, os desenvolvedores dedicavam a maior parcela dos seus esforços e tempos na parte de codificação e testes de unidade. Estima-se que 80% deste esforço eram desprendidos nestas atividades [MFR03]. A partir da década de 90, e principalmente nos dias atuais, houve uma mudança na abrangência e complexidade das aplicações, cujos fatores como segurança e desempenho passaram a ser relevantes, tornando a atividade de testar cada vez mais especializada [MFR03].

No propósito de melhorar a qualidade e desenvolvimento de seus softwares, as empresas têm voltado suas atenções para os processos de testes, visando uma grande economia em correções de softwares mal produzidos. Quanto antes a presença do defeito é revelada, menor o custo de correção e maior a probabilidade de corrigi-lo corretamente. Os custos de descobrir e corrigir defeitos no software aumentam exponencialmente na proporção em que o trabalho evolui através das fases do projeto [MFR03]. O Processo de Testes de Software representa uma estruturação de etapas, atividades, artefatos, papéis e responsabilidades que buscam a padronização dos trabalhos e controle dos projetos de testes.

A qualidade de um produto de software está muito relacionada à satisfação do cliente, e o teste de software é uma das formas de validar se o produto desenvolvido atende às necessidades de seus usuários [KFN99].

O processo de teste de software é voltado para o alcance de um nível de qualidade

de produto, que, durante o processo de desenvolvimento de software, muda conforme o avanço das atividades - requisitos, análise e projeto, implementação e implantação, dessa forma deve-se garantir que testes não serão feitos apenas depois da finalização do processo de implementação, serão realizados durante todo o processo. Teste de software é um processo ou um conjunto de processos projetado para garantir que o código faça tudo o que foi projetado para fazer. Software deve ser previsível e consistente, não oferecendo surpresas para os usuários, por isso a importância da disciplina de testes de software [Pat05].

O processo de teste deve estar associado a objetivos quantificáveis, que possam ser mensurados e monitorados, e deve ser capaz de evoluir a um nível em que haja mecanismos que o levem a promover melhorias contínuas. Essas melhorias no processo de teste, por sua vez, estão diretamente relacionadas com a qualidade do software à medida que garantem a satisfação e a confiança do cliente, ao lhe entregar um produto dentro do prazo e orçamento acordados e com a segurança de que o mesmo atende às suas exigências.

O principal objetivo do teste de software é descobrir defeitos e, em conseqüência, é garantir que o processo de desenvolvimento de software atinja níveis de qualidade especificados e um produto de alta qualidade. Quando um software é testado, é adicionado um valor a esse sistema, ou seja, é aumentado a qualidade e confiabilidade ao software. As atividades de testes de componentes de softwares, dentro do processo de desenvolvimento de softwares, têm grande importância, pois através delas é apresentado o aval para a garantia dos produtos que são entregues aos usuários ou clientes.

A finalidade dos testes de software é garantir que os sistemas de software funcionem como esperado quando executado pelos usuários-alvo [Tia05]. Myers [Mye04] para alcançar os objetivos dos testes é preciso:

- Executar um programa com a intenção de descobrir um erro;
- Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto;
- Um teste bem-sucedido é aquele que revela um erro ainda não descoberto.

As próximas subseções apresentam uma breve introdução dos processos de testes de software e do planejamento de testes.

4.1.2.1 Processo de Testes de Software Teste é também o processo de avaliar a qualidade de um software. O processo de testes define como os testes serão planejados, projetados, implementados, executados e avaliados através de um conjunto de atividades, artefatos e papéis. A seguir são apresentadas as principais atividades de um processo de testes extraídas de [GVVEB08] e descritas seqüencialmente.

- **Planejar e Controlar**: Determina e acompanha, de forma geral, escopo e objetivos dos testes, os riscos que serão endereçados, a abordagem, recursos, cronograma e critérios de saída.

- **Análise e Projeto**: Nesta atividade, objetivos gerais de teste são transformados em cenários e projetos de testes tangíveis; são revisados as análises de riscos, requisitos, arquiteturas, projeto e interface.
- **Implementação e Execução**: Os cenários de testes são transformados em procedimento; o ambiente é criado incluindo os dados para execução dos testes bem como os casos de teste implementados.
- **Avaliação do Critério de Saída e Resultado**: A execução dos testes é avaliada de acordo com o planejado. O critério de saída determina quando uma dada atividade de teste é concluída.
- **Atividades de Encerramento dos Testes**: Dados são coletados com o intuito de consolidar experiência; inclui analisar ambiente, dados, procedimentos de testes.

Essas atividades devem ser desenvolvidas ao longo do próprio processo de desenvolvimento de software e, em geral, concretizam-se em quatro estágios de teste: de unidade, de integração, de sistema e de aceitação. O teste de unidade concentra esforços na menor unidade do projeto de software, ou seja, procura identificar erros de lógica e de implementação em cada módulo do software, separadamente. O teste de integração é uma atividade sistemática aplicada durante a integração da estrutura do programa visando descobrir erros associados às interfaces entre os módulos; o objetivo é encontrar defeitos de interfaceamento entre os módulos e os subsistemas. O teste de sistema, realizado após a integração do sistema, visa a identificar erros de funções e características de desempenho que não estejam de acordo com a especificação. O teste de aceitação é realizado pelo usuário final a fim de demonstrar a conformidade com os requisitos do software [PMM02].

4.1.2.2 Planejamento de Testes de Software Devido ao tamanho e à complexidade dos sistemas de softwares produzidos atualmente, os testes informais sem muito planejamento e preparação se tornam insuficientes. Importantes funções, características e detalhes de implementação podem ser esquecidos nos testes informais. Portanto, existe uma forte necessidade do planejamento de testes a fim de controlar, monitorar e definir estratégias com base nas orientações de qualidade, modelos formais e técnicas afins [Tia05]. O primeiro passo para a execução de um teste bem-feito é seu planejamento [BRCM06], um bom planejamento evita perdas de recursos e atrasos no cronograma [Bei02]. O planejamento garante que os testes sejam preparados antes da conclusão do produto e evita testes tendenciosos [Bei02].

Um planejamento cuidadoso é necessário para se obter o máximo de casos de testes e para controlar o custo do processo de testes [Som95]. Esta etapa caracteriza-se pela definição de uma proposta de testes baseada nas expectativas do cliente em relação a prazos, custos e qualidade esperada, possibilitando dimensionar a equipe e estabelecer um esforço de acordo com as necessidades apontadas pelo cliente.

O planejamento de testes se concentra em estabelecer padrões para o processo de testes, não apenas em descrever testes de produtos. Assim como ajudam os gerentes a

alocar recursos e estimar cronograma de testes. Como parte do processo de planejamento, deve-se estabelecer as estratégias, especificar padrões, procedimentos de testes, estabelecer *checklist* e definir o plano de testes. O plano de testes deve conter a descrição do escopo, a estratégia de testes, os recursos utilizados, os marcos das atividades de testes do projeto. Este documento está diretamente ligado ao plano do projeto.

O plano de testes apresenta o planejamento para execução do teste, incluindo abrangência, abordagem, recursos e cronograma das atividades de teste. Identifica, também, os itens e as funcionalidades a serem testados, bem como as tarefas a serem realizadas e os riscos associados com a atividade de teste. Isso é útil para gerentes de sistemas responsáveis por garantir que esses recursos estarão disponíveis para a equipe de testes. Existe um padrão mundialmente aceito, definido pelo IEEE, que lista o conteúdo de um plano de testes [CSB+04].

4.2 METODOLOGIA DE AVALIAÇÃO DE DESEMPENHO

Atualmente, verifica-se um interesse por parte de equipes de desenvolvimento de softwares em produzir sistemas incluindo o quesito qualidade [PMM02]. Para isso, tem-se dado importância à atividade de teste que contribui por construir sistemas com maior qualidade e confiabilidade [BMV+00].

Além disso, a complexidade das tecnologias utilizadas e dos softwares produzido tem crescido, tornando-se indispensável a utilização de processos, métodos, técnicas e ferramentas que permitam a realização de teste de software de maneira sistematizada e com fundamentação teórica, com o propósito de aumentar a qualidade do produto de software e com um menor custo possível [DMJ07]. O objetivo principal desta metodologia de avaliação de desempenho de processos de testes de software é propor um procedimento que possibilite avaliar o desempenho dos processos e, desta forma, dar suporte às atividades de planejamento dos recursos empregados nas atividades do processo de testes.

A estratégia adotada para avaliação de desempenho de processos de testes é composta de dez etapas: compreensão do problema e entendimento do processo de testes, estabelecimento de métricas, coleta de dados, análise dos dados coletados, mapeamento do processo, geração de modelo abstrato, tratamento estatístico dos dados, geração do modelo refinado, validação do modelo proposto e avaliação de cenários. A Figura 4.1 mostra o diagrama de atividades do método para avaliação de desempenho de processo de testes.

A primeira etapa, compreensão do problema e entendimento do processo de testes, corresponde ao estudo do processo de testes a ser analisado, a identificação das funções estabelecidas no processo de testes, a identificação de artefatos (pré-requisitos, opcionais e liberados), quais insumos devem ser liberados, entendimento das diversas etapas e atividades do processo adotado pela organização ou por um projeto específico. Nesta etapa, é realizado um estudo do projeto, para observar possíveis pontos de melhoria referentes ao processo adotado. Com intuito de obter mais informações, realiza-se uma pesquisa sobre os principais problemas que a organização vem enfrentando, relacionados

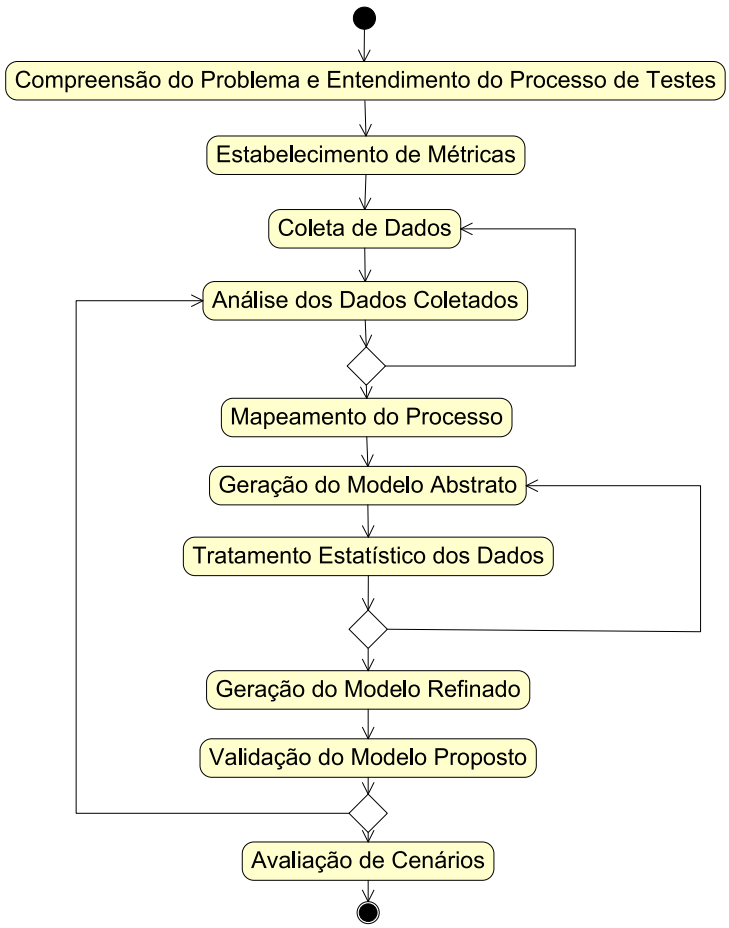


Figura 4.1: Metodologia de Avaliação de Desempenho.

às atividades do processo de testes de software adotado. Ainda nesta etapa, são obtidas informações das equipes envolvidas que executam o processo de teste. Deve se analisar as características do software a ser testado. Verifica-se o nível de qualificação das pessoas envolvidas no projeto, o número de liberações do produto para o cliente, complexidade do sistema e as restrições de prazo. O pré-requisito para que esta atividade seja realizada é a autorização da organização para acesso às informações do processo de testes e também de todos os envolvidos no projeto. A conclusão da atividade é a geração de um documento descrevendo todo o cenário (atividades do processo, documentos, problemas e envolvidos).

A segunda etapa, estabelecimento de métricas, ocorre pela definição das métricas de interesse, ou seja, métricas são estabelecidas para a realização do estudo e demonstração dos resultados obtidos para a instituição. O pré-requisito que deve ser satisfeito para iniciar a execução desta atividade é o documento gerado na atividade *compreensão do problema e entendimento do processo de testes*, pois, baseado nas informações relatadas, serão estabelecidas as métricas de interesse. Como produto desta etapa, são definidas as métricas de interesse.

A fase de coleta dos dados compreende a definição do ambiente de medição, calibração

de ferramental utilizado na coleta de dados, a seleção das informações a serem mensuradas, o procedimento de medição, os formatos de armazenamento e, por fim, a execução do procedimento de medição. Os pré-requisitos que devem ser satisfeitos para se iniciar a execução da atividade são as métricas estabelecidas na atividade anterior para que seja realizada uma seleção adequada das informações a serem coletadas. Após realização da atividade, é gerado um relatório contendo as informações sobre "o quê", "onde", "como", "quem", "quando". **O quê**: as informações que serão coletadas. **Onde** as informações são armazenadas. **Como** realizar as medições. **Quem** é a pessoa responsável pela coleta. **Quando** é o período de coleta.

A fase de análise dos dados corresponde, primeiro, à verificação de possíveis distúrbios nos dados coletados e, em seguida, à aplicação de métodos estatísticos nos dados coletados com o objetivo de fornecer informações precisas a respeito do sistema em avaliação. Entre as estatísticas de interesse, a média, o desvio padrão e o coeficiente de variação são de importância particular, por serem essenciais para definição das distribuições expolinomiais adequadas para representar os dados mensurados. O relatório gerado na etapa de *coleta dos dados* é o pré-requisitos que deve ser satisfeito para se iniciar a execução da atividade. A atividade é finalizada quando é gerada uma análise dos dados coletados.

Os processos de software são usualmente representados através de modelos semiformais (UML por exemplo). Esses modelos, por si só, não fornecem suporte para a avaliação de desempenho das especificações dos processos, assim, faz-se necessário o mapeamento desses modelos para modelos formais. A conclusão da etapa se dá pelo mapeamento do modelo semiformal utilizado em um modelo formal.

A etapa de geração do modelo abstrato corresponde à geração de um primeiro modelo de desempenho. Os modelos podem ser expressos em diferentes níveis. A escolha da granularidade determina as análises que podem ser realizadas, dependendo do detalhamento dos componentes do sistema [MA05]. Os pré-requisitos que devem ser satisfeitos para se iniciar a execução da atividade são os mapeamentos realizados na etapa anterior. O produto liberado será o modelo abstrato de desempenho.

A fase de tratamento estatístico dos dados utiliza o *moment matching* [DAJ95], para calcular os dois primeiros momentos da distribuição empírica (dados coletados), a média (μ_D) e o desvio padrão (σ_D), e os associa aos respectivos primeiros momentos da *s-transition*. As estatísticas obtidas permitem a seleção da distribuição expolinomial (hipoexponencial, hiperexponencial e *erlang*) que melhor se adapta à distribuição empírica.

A etapa de geração do modelo refinado corresponde à geração do modelo de desempenho em função do modelo abstrato e das estatísticas obtidas na fase de medição. Essas estatísticas sugerem o tipo de distribuição expolinomial que melhor se adapta à distribuição empírica (dados coletados). Essa adequação é realizada com o auxílio da técnica de *moment matching* [DAJ95], que calcula os dois primeiros momentos da distribuição empírica, a média (μ_D) e o desvio-padrão (σ_D), e os associa aos dois primeiros momentos da *s-transition*. Dessa forma, primeiro, determina-se qual o tipo de distribuição expolinomial que melhor representa os dados coletados. Em seguida, encontram-se os valores dos parâmetros numéricos da distribuição expolinomial escolhida. Para o início da execução

desta atividade, é preciso que as outras etapas sejam satisfeitas. O artefato gerado na conclusão desta atividade é o modelo de desempenho refinado.

A etapa de validação do modelo proposto analisa se os resultados das métricas de desempenho calculadas pelo modelo são equiparáveis aos dos obtidos através de medições no sistema, considerando um erro de exatidão aceitável. Como pré-requisito, é necessário que as outras etapas sejam satisfeitas. A finalização da atividade é a apresentação e a comparação dos resultados obtidos.

A etapa de avaliação de cenários corresponde à análise de diferentes cenários com o objetivo de encontrar configurações adequadas em termos de custo, tempo, qualidade. A análise pode compreender o estudo do processo em função de variações dos níveis de qualificação dos colaboradores envolvidos no processo de testes, da quantidade de defeitos escapados, do interesse de verificar o desempenho da adoção de uma determinada atividade do processo de testes. O pré-requisito que deve ser satisfeito para se iniciar a execução da atividade é um modelo de desempenho refinado, e o resultado da atividade é a análise de diferentes cenários.

4.3 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os conceitos básicos de planejamento de projeto de software e de conceitos de testes de software. Após isso, foi apresentada uma metodologia de avaliação de desempenho de processo de software. Em resumo, esta metodologia baseia-se, principalmente, na composição de dados históricos do processo de testes, como também a composição sistemática dos modelos básicos gerados pelo processo de mapeamento do diagrama de atividades da UML; para, então, se obter um modelo de desempenho que representa a especificação do processo de testes. A metodologia é composta por diversos passos que vão desde do estudo do processo de testes, passando pelo mapeamento do diagrama de atividades da UML em SPN, até a análise de resultados.

CAPÍTULO 5

MODELAGEM

Este capítulo apresenta os modelos SPN de desempenho do processo de Testes. Inicialmente, apresenta-se uma visão geral. Em seguida, os mapeamentos realizados entre os diagramas de atividades da Unified Modeling Language (UML) em SPN. Após isso, são apresentados modelos para alocação de recursos e, finalmente, apresenta-se a aplicação da metodologia de avaliação apresentada, como também o modelo de desempenho do processo de teste proposto.

5.1 VISÃO GERAL

Atualmente, a forma mais amplamente usada para especificar processos de software é através das linguagens semiformais, tais como UML (*Unified Modeling Language*), devido principalmente a sua notação amigável e intuitiva. A UML foi desenvolvida para especificar, visualizar e modelar sistemas de softwares, bem como para modelagem de negócio e processos de software. A UML representa uma notação que tem sido aprovada com sucesso na modelagem de sistemas [UML10]. No entanto, os modelos semiformais gerados por essas linguagens, por si só não fornecem suporte para avaliação de desempenho das especificações dos sistemas, assim, faz-se necessário o mapeamento destes modelos semiformais para modelos formais. Pois modelos formais são apoiados por fundamentos matemáticos sólidos, que suportam sua semântica precisa, estimulam a avaliação de desempenho e fornecem suporte para verificações das propriedades qualitativas e análises.

Diagramas de atividades são utilizados para descrever lógica de programação, processos de negócio e *workflows*. Além disso, esse diagrama possui muitos recursos que permitem representar estruturas complexas, tais como processos paralelos e condicionais, exceções, eventos, entre outros. Os diagramas de atividades representam ações pré e pós-condições e seus resultados e fornecem uma representação gráfica do fluxo das interações. Analogamente ao fluxograma, um diagrama de atividade usa retângulos arredondados para descrever uma função específica do sistema, setas para representar relações de precedência entre os componentes do sistema, losangos para representar decisões e linhas horizontais sólidas para indicar a execução das atividades paralelas [UML10, Pen03].

Nas próximas seções, os elementos dos diagramas de atividades da UML são mapeados em SPN, a fim de adotá-los para a construção do modelo de processo de testes. Além disso, são apresentados modelos estocásticos para previsão de estimativas (tempo, custo, qualidade) das atividades do processo de testes, considerando a qualificação dos envolvidos nas atividades e, finalmente, é apresentado um estudo de caso da modelagem de um processo de testes adotado por uma empresa de desenvolvimento de software. O modelo proposto para avaliação de desempenho irá tratar o problema da previsão de estimativas

dada a composição da equipe desejada; com o modelo criado, é realizada uma validação da metodologia proposta no Capítulo 4.

5.2 MAPEAMENTO DOS DIAGRAMAS DE ATIVIDADES - UML EM SPN

Os diagramas de atividades da Unified Modeling Language (UML), pela sua simplicidade de utilização e grande aplicação nas diferentes empresas, podem representar o processos de testes. O diagrama de atividades oferece uma rica notação para demonstrar uma seqüência de atividades, como também atividades em paralelo e é bastante aplicado em fluxos de trabalho, processos de software e negócios [Lar08]. Esta seção apresenta os principais elementos que compõem o Diagrama de Atividades (DA), assim como as regras de mapeamento em um modelo SPN [MMSM10].

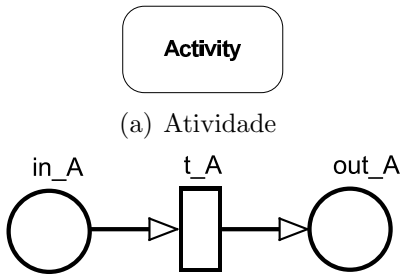
5.2.1 Mapeamento de Atividades

As atividades representam a execução de um processamento não atômico, envolvendo uma ou mais ações. Uma ação consiste em um processamento que resulta em mudança de estado no sistema.

A Figura 5.1, apresenta um fluxo simples de atividades representado através do diagrama de atividades. Esse diagrama descreve um exemplo simplificado do processo de publicação de um artigo. A atividade de publicar um artigo se inicia com o estudo do assunto que se deseja publicar. Uma vez entendido o assunto e obtidos os resultados necessários, inicia-se a atividade de escrita. Por fim, a última atividade consiste na submissão do artigo.

Figura 5.1: Exemplo de um DA.

As atividades são representadas por retângulos com os cantos arredondados (observe a Figura 5.2(a)). As atividades retratam uma invocação de uma operação que pode ser física ou eletrônica e uma ação representa um passo dentro de uma única atividade. No Modelo SPN gerado pelo processo de mapeamento, as atividades são representadas por lugares e transições. Os lugares in_A e out_A representam, respectivamente, as pré-condições e as pós-condições da atividade A. A transição t_A pode representar a duração das atividades ou um conjunto de condições para atribuição de um determinado tempo, como por exemplo, a variação do tempo da atividade de acordo com o número de pessoas envolvidas. A Figura 5.2(b) ilustra o mapeamento de uma atividade para o modelo SPN.



(a) Atividade

(b) Atividade SPN

Figura 5.2: Mapeamento de Diagramas de Atividades para modelo SPN.

5.2.2 Mapeamento de Transições

Segundo [Gue08], uma transição representa um evento que causa uma mudança no estado da entidade, gerando uma nova atividade. As transições são graficamente representadas por uma seta apontando para a atividade destino. Adicionalmente, uma transição pode ter várias origens (nesse caso, ela representa uma junção de várias atividades simultâneas) e vários alvos (nesse caso, ela representa uma forquilha para várias atividades simultâneas). A transição representa o relacionamento entre duas ou mais atividades e são chamadas de transição não-ativadas, por não representarem um espaço de tempo.

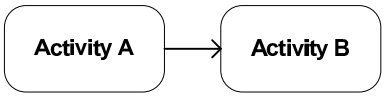
Na Figura 5.3, é apresentada uma transição relativa ao processo de abertura de conta. Neste exemplo, existem duas atividades, a primeira, *Consultar Pessoa* pode executar uma ação de consulta de CPF, por exemplo. Após a ação ser realizada, causa uma transição para a atividade *Atualizando Pessoa*. A transição é representada por uma seta ligando as duas atividades da Figura 5.3.

Figura 5.3: Exemplo do fluxo de Transições.

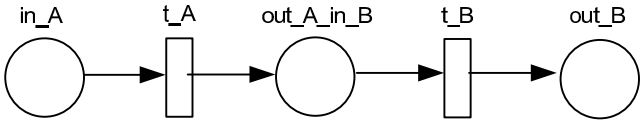
A Figura 5.4(a) representa a transição da atividade A para atividade B no diagrama de atividades da UML. No modelo SPN gerado pelo processo de mapeamento; as transições são representadas por um lugar. O lugar ($out\_A\_in\_B$) representa, a transição da atividade A para atividade B (ver a Figura 5.4(b)).

5.2.3 Mapeamento do Estado Inicial e Final

O estado inicial é um pseudo-estado, cuja função é indicar (apontar para) o ponto de partida do diagrama de atividades, esse pseudo-estado, é representado por um círculo preenchido (ver Figura 5.5(a)). Esse estado inicial é representado por um lugar composto de uma marca no modelo SPN, a Figura 5.5(b) representa o estado inicial, em que o lugar $staIni\_A$ contém a marcação. A transição-PN t_in_A representa a transição entre o



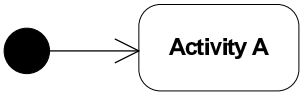
(a) Transição UML



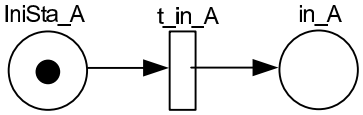
(b) Transição SPN

Figura 5.4: Mapeamento de Transições.

estado inicial e o início de uma atividade A. A Figura 5.5 apresenta o mapeamento.



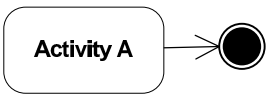
(a) Estado Inicial UML



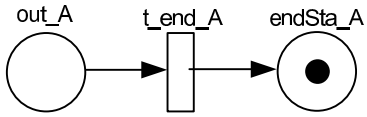
(b) Estado Inicial SPN

Figura 5.5: Mapeamento do Estado Inicial.

O estado final é um pseudo-estado, cuja função é indicar a finalização do fluxo no diagrama de atividades, esse pseudo-estado é representado por um círculo não preenchido envolvendo um segundo círculo preenchido, como pode ser observado na Figura 5.6(a). O estado final é mapeado em um lugar (*endSta_A*) no modelo SPN, no qual a presença de uma marca representa o fim do DA, como pode ser observado na Figura 5.6(b). Além disso, a transição-PN *t_end_A* é usada para representar a transição entre o término da atividade A e o estado final. A Figura 5.6 apresenta o mapeamento.



(a) Estado Final UML



(b) Estado Final SPN

Figura 5.6: Mapeamento do Estado Final.

5.2.4 Barras de Sincronização

As barras de sincronização indicam a execução de fluxos concorrentes ou paralelos e são representadas por barras verticais ou horizontais que mostram a separação (*fork*) ou união (*join*) do fluxo. As barras de sincronizações são representadas por retângulos com múltiplas transições de entradas e/ou saídas. A Figura 5.7 apresenta um exemplo. No exemplo ao ser finalizada a atividade *Separar Produtos* é realizado o *fork* para que as atividades *Embalar Produto* e *Emitir Nota Fiscal* sejam executadas em paralelo, quando

as ações das atividades finalizarem, um *join* é realizado, pois a execução da atividade *Despachar Produtos* depende da finalização de *Embalar Produto* e *Emitir Nota Fiscal*.

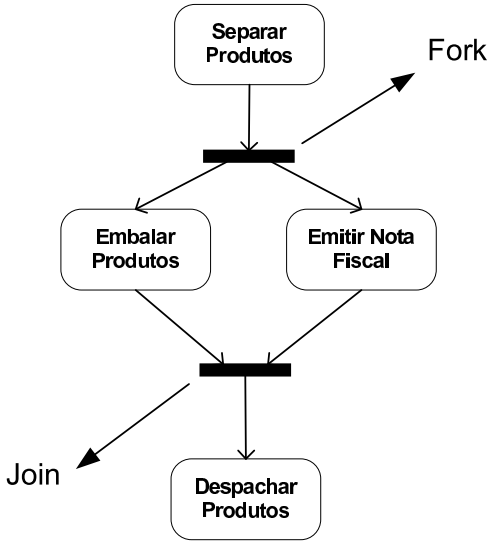


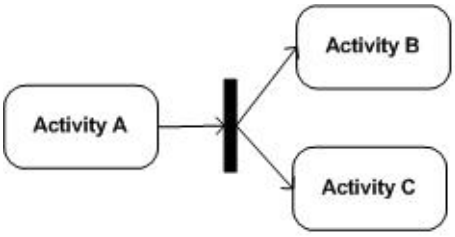
Figura 5.7: Barra de sincronia.

A Figura 5.8(a) representa um *fork*, em que a conclusão da atividade A é uma pré-condição para que as atividades B e C sejam executadas. No processo de mapeamento, a transição imediata SPN *t_fork_B_C*,representa o *fork*-SPN, ou seja, o início do paralelismo das atividades B e C. A Figura 5.9(a) mostra a junção (*join*) entre as atividades A e B. No processo de mapeamento, a transição imediata SPN *t_join_A_B* é usada para representar a junção das atividades A e B, para que a atividade C seja realizada. O disparo da transição *t_join_A_B* só é possível quando existir uma marca em ambos lugares *out_A* e *out_B*.

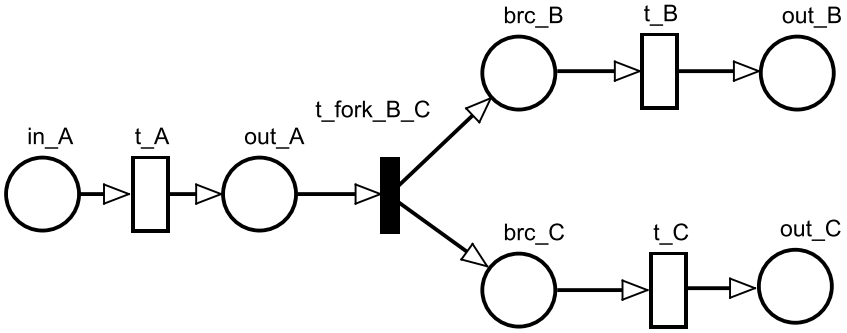
5.2.5 Mapeamento Decisão

A decisão representa um ponto do fluxo de controle, onde deve ser realizada uma tomada de decisão. Normalmente é representado por um losango com uma entrada e várias saídas. As saídas possuem condições de guarda que controlam quais transições (de um conjunto de transições alternativas) sucede a atividade que será concluída. A Figura 5.10 representa um exemplo, em que na finalização da atividade *Inclusão de Produtos no Pedido*, são verificadas duas possibilidades: Produto disponível ou Produto não disponível. Se for escolhida produto disponível, o fluxo irá para a atividade *Separar Produtos*, caso seja escolhido Produto não disponível é encaminhado para a atividade *Adquirir Produtos*.

Na Figura 5.11(a), o resultado da atividade A será verificado pelo elemento decisão através das expressões de condições e assim escolhida uma das atividades B ou C. No mapeamento para SPN, a decisão é definida por duas transições *t_Yes* ou *t_No* , veja a Figura 5.11(b).

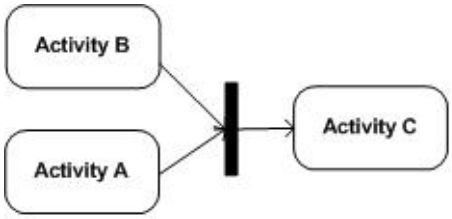


(a) Fork UML

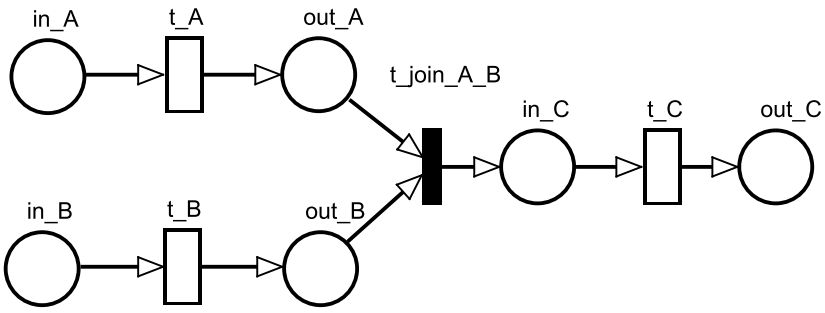


(b) Fork SPN

Figura 5.8: Mapeamento do Fork.



(a) Join UML



(b) Join SPN

Figura 5.9: Mapeamento do Join.

5.3 MODELOS PARA ALOCAÇÃO DE RECURSOS

O gerenciamento de recursos humanos do projeto inclui atividades que organizam e gerenciam a equipe do projeto. A equipe do projeto consiste de pessoas, com papéis e res-

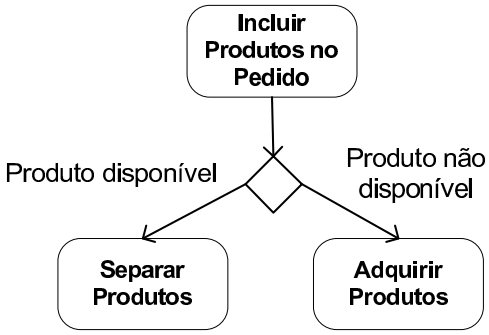
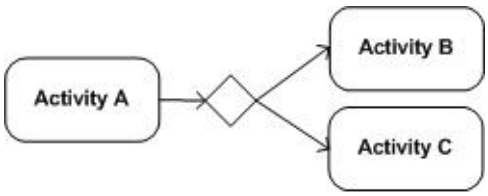
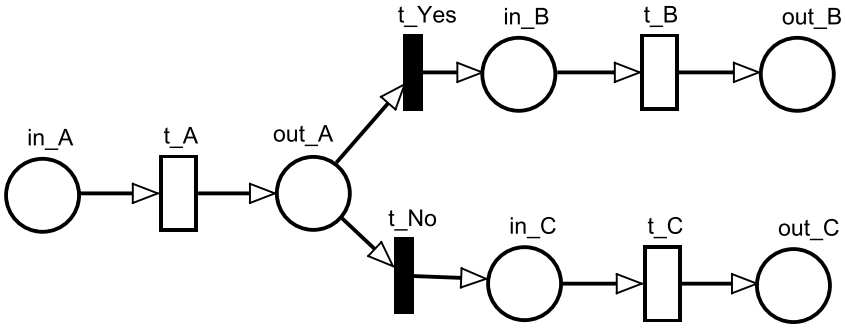


Figura 5.10: Decisão.



(a) Decisão UML



(b) Decisão SPN

Figura 5.11: Mapeando Decisão

ponsabilidades, designadas para conclusão do projeto. A gerência de recursos humanos objetiva garantir o melhor aproveitamento das pessoas envolvidas no projeto. Consiste no planejamento organizacional, alocação de pessoal e definição de equipe [Ins08].

5.3.1 Modelo de Alocação de Recursos

O número de recursos utilizados no projeto para as atividades de testes pode ser encontrado no plano de testes. O plano de testes descreve os recursos que estarão disponíveis para as atividades de testes no projeto, bem como o cronograma das atividades, as estratégias de testes e elementos como níveis de cobertura de código exigidos.

A Figura 5.12 apresenta um mapeamento de uma atividade do DA, adicionada das informações de alocações de recursos que podem ser obtidas no plano de teste. Na Figura 5.16, o place *in_A* representa o início da execução das atividades; o tempo é atribuído

a transição *Time*; a finalização da atividade é representado por *out_A*; o número total de pessoas na equipe é representado pela marcação *Team*, ou seja, o número de marcas no lugar *P*1, o número de pessoas alocadas para essa atividade é representado por *NP*, o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de pessoas que estão executando a atividade é representado pelo número de marcas no lugar *P*0.

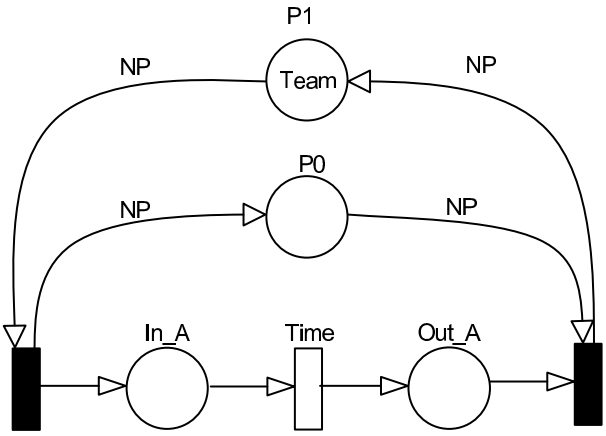


Figura 5.12: Alocação de Recursos.

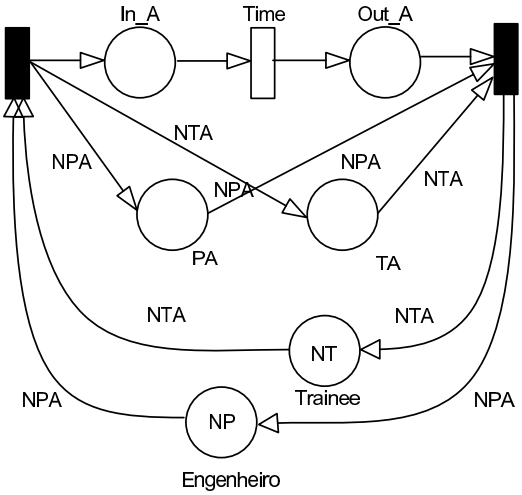
5.3.2 Modelos de Alocação de Diferentes Recursos

Os recursos alocados no plano de testes podem ter características distintas com relação a quantidade de horas alocadas para a atividade ou com relação ao nível de qualificação. As organizações tentam classificar o nível do colaborador envolvido em um determinado cargo [Res02] e dentre esses cargos há uma variação do tempo de trabalho. A Figura 5.13(a) apresenta um modelo SPN de alocação para dois tipos de recursos.

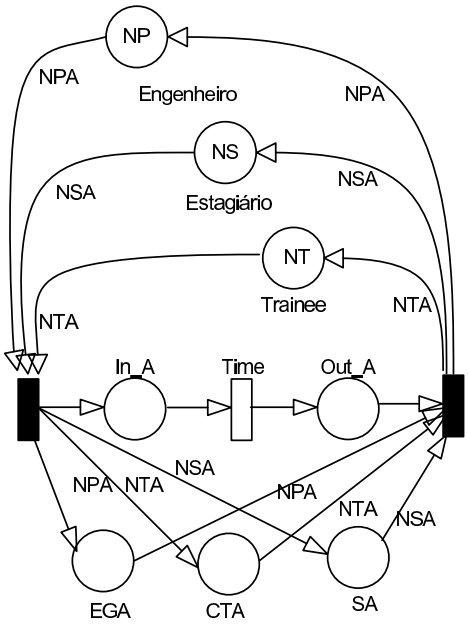
Para contextualizar o modelo proposto, representado na Figura 5.13(a), considerem-se dois recursos: *trainee* e engenheiro. O número total de engenheiros na equipe é representado pela marcação *NP*, ou seja, o número de marcas no lugar *Professional*, o lugar *Trainee* possui o número total de *trainees* que é representado pela marcação *NT*. *NPA* é o peso do arco que representa a quantidade de engenheiros que serão alocados para a atividade, e *NTA* o peso do arco que representa o número de *trainees* que serão alocados. O número de marcas nos lugares *PA* e *TA* correspondem respectivamente ao número de engenheiros alocados e número de *trainees* alocados que estão realizando a atividade. O lugar *in_A* corresponde ao início da atividade, a transição *t_A* corresponde ao tempo de execução da atividade e este é modificado de acordo com os recursos que são alocados, o lugar *out_A* corresponde ao término da atividade.

Para o planejamento com três tipos de recursos de níveis de qualificações diferentes é apresentado o modelo da Figura 5.13(b). O modelo contém, além dos elementos apresentados no modelo da Figura 5.13(a) os seguintes elementos: o número total de estagiários na equipe é representado pela marcação *NS*, ou seja, o número de marcas no

lugar *Estagiário*, *NSA* o peso do arco que representa o número de estagiários alocados para executar a atividade e o lugar *SA* que indica o número total de estagiário que estão executando a determinada tarefa.



(a) Alocação de dois tipos de Recursos.



(b) Alocação de três tipos de Recursos.

Figura 5.13: Alocação de Colaboradores Com Diferentes Qualificações

5.4 VALIDAÇÃO

Esta seção apresenta um estudo de caso com o objetivo de validar o modelo SPN, concebido para avaliação de desempenho de processo de software, como também a validação da metodologia proposta no Capítulo 4.

O estudo foi realizado com base no processo de testes de software adotado por um Instituto de software do Recife, por motivos de contratos entre o instituto e os clientes, não é permitido divulgar o nome da empresa, porém todos os dados utilizados aqui são reais.

O diagrama de atividades do processo de testes é apresentado na Figura 5.14. Foi realizado um estudo do processo de testes da empresa com o objetivo de entender o funcionamento do processo, quais artefatos são pré-requisitos, quais são opcionais, qual a função de cada atividade de teste e quais artefatos são liberados no final de cada atividade.

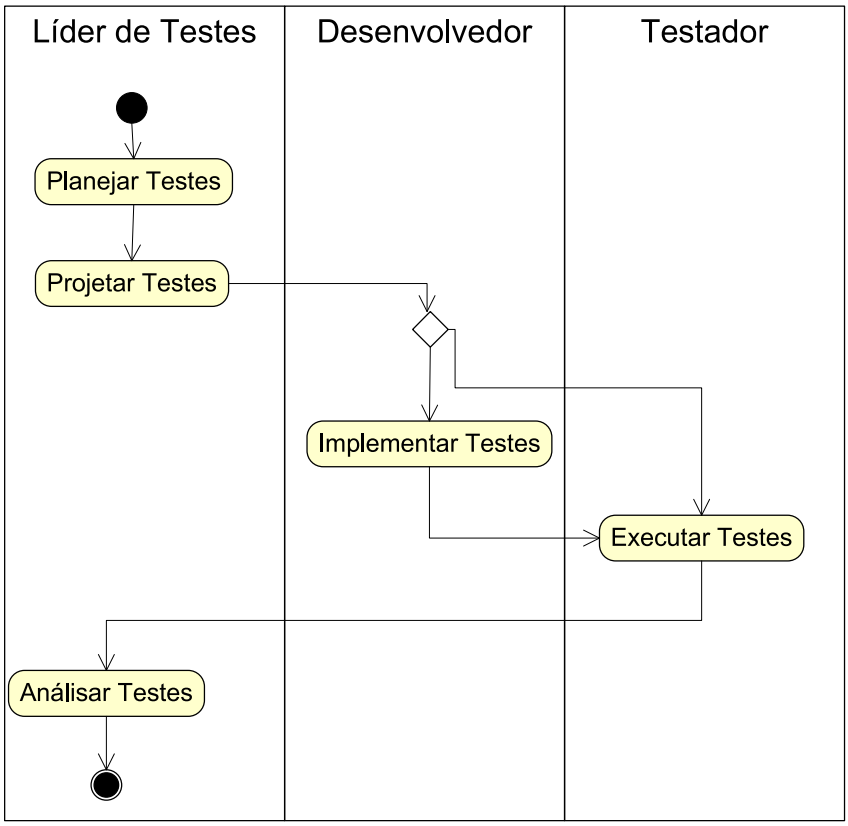


Figura 5.14: Processo de Testes

A primeira atividade do processo de testes, **planejar testes**, tem como objetivo planejar as atividades de testes que serão executadas ao longo de todo o ciclo de vida do projeto. As entradas da atividade são: o plano do projeto, cronograma do projeto, planilha de riscos, documento de requisitos dos clientes e documento de requisito do produto. As saídas são: O plano de testes, cronograma atualizado, se necessário, e planilha de riscos atualizada. Nesta atividade os gerentes de projetos ou líderes de testes devem:

- Elaborar o Plano de Testes considerando os seguintes pontos:
  - Selecionar e priorizar os requisitos, de acordo com os riscos, funcionalidades e

características do produto a ser testado;
  - Determinar os ciclos de testes e o conjunto de requisitos que devem ser testados em cada ciclo;
  - Especificar os critérios de entrada e conclusão, bem como de bloqueio/suspensão, para acompanhamento dos ciclos de teste;
  - Planejar os recursos, tanto humanos como materiais;
  - Estabelecer os marcos das atividades de teste do projeto;
  - A forma como os defeitos serão acompanhados também pode ser definida no planejamento;
  - Identificar que tipos de riscos podem vir a acontecer e reportar ao gerente estes riscos, caso eles aconteçam;

- Atualizar o Cronograma do projeto de acordo com os marcos definidos para as atividades de teste.

É importante lembrar que esta atividade deve está ligada à atividade de planejamento de todo o projeto de desenvolvimento de software.

A atividade **projetar testes** tem como objetivo identificar e especificar os casos de testes baseando-se nos requisitos e na estratégia descrita no Plano de Testes. Os documentos de entradas desta atividade são: Documentos de requisitos do cliente e do produto, documento de arquitetura, plano de testes, plano do projeto, manual de instalação. Os documentos de saída são: projeto de teste, plano de testes, planilha de resultados. Nesta atividade o projetista de testes ou engenheiro de testes deve:

- Especificar cada caso de teste, informando:
  - Tipo de execução: manual, automática ou semi-automática;
  - Tipo do teste: funcional, desempenho, carga, etc;
  - Objetivo do caso de teste;
  - Pré-condições para execução do caso de teste;
  - Passos com respectivos resultados esperados;
  - Pós-condições a serem verificadas após a execução do caso de teste;
  - Recursos necessários, caso haja algum especial;
  - Severidade do Teste;
- Mapear a rastreabilidade dos casos de teste:
  - Requisitos Funcionais x Casos de Teste;
  - Requisitos Não-Funcionais x Casos de Teste ;

- Definir os ciclos de teste, com o conjunto de casos de teste que devem ser executados em cada ciclo, assim como a seqüência de execução dos mesmos.

    - Atualizar a estratégia definida no Plano de Testes, caso haja discrepância com o que foi planejado inicialmente;

A atividade **implementar testes**, conhecida também como **automatizar teste**, tem como objetivo implementar *scripts* de automação para utilizar na execução dos casos de teste. Os documentos de entradas são: documentos de requisitos do cliente e do produto, documento de arquitetura, plano de testes, projeto de testes, especificações de componentes. A saída corresponde ao projeto de testes atualizado se necessário e aos *scripts* de automação. Nessa atividade, os desenvolvedores ou engenheiros de testes devem:

- Configurar a ferramenta que será usada para automação;
- Executar manualmente o caso de teste, a fim de melhorar o entendimento do que deverá ser automatizado;
- Identificar os dados de entrada e saída dos testes, definidos no Projeto de Testes, que serão utilizados para a execução dos mesmos;
- Implementar o *script* de automação de acordo com os padrões estabelecidos no Documento de Arquitetura;
- Se necessário, integrar os *scripts* de automação ao ambiente de testes, realizando ajustes e disponibilizando para a execução.

A atividade de **executar testes** tem como objetivo: executar os casos de teste projetados e registrar os seus resultados. As entradas para essa atividade são documento de requisito do cliente e do produto, projeto de testes, planilha de resultados, *scripts* de testes (caso haja automação), produto a ser testado (versão baseline) e manual de instalação. As saídas são a planilha de resultados atualizada e *log* de automação dos testes (caso exista automação). Os engenheiros de testes devem:

- Analisar cada caso de teste selecionado para a execução do ciclo, identificando o objetivo do teste, a funcionalidade do sistema a ser validada e a seqüência de execução;
- Para execução manual, deve ser analisado o resultado obtido durante a execução de cada caso de teste, comparando com o resultado esperado descrito no caso de teste:

    - Caso os resultados esperados não sejam atingidos, realizar novamente o teste para identificar se há problema do sistema, do ambiente ou do caso de teste;

  - Registrar os resultados obtidos na Planilha de Resultados. Documentar qualquer evento que ocorra durante a atividade de teste e que requeira análise posterior;

- Para execução automática, deve ser executado o(s) *script(s)* de teste associados ao caso de teste:
  - Avaliar a execução do(s) *script(s)* de teste, identificando os eventuais incidentes e ocorrências que serão registradas por uma ferramenta, escolhida na organização, através de um *log*. Este *log* deve ser usado como base para registrar o resultado na Planilha de Resultados;
  - Caso os resultados esperados não sejam atingidos, realizar novamente o teste para identificar se é problema do produto testado, do ambiente, do caso de teste ou da ferramenta;
  - Após a análise dos resultados da execução do caso de teste, abrir ocorrência de erro na ferramenta de Gerenciamento de Mudanças.

- Ao término de cada ciclo/rodada:
  - O Projetista deverá analisar a necessidade de uma nova rodada de testes para o mesmo ciclo, conforme critérios de conclusão e êxito definidos no Plano de Testes.
  - O Testador deverá dar início ao procedimento de testes para a nova rodada, repetindo os passos descritos anteriormente (análise e execução de casos de teste).

A atividade **analisar teste** tem como objetivo avaliar os resultados dos testes. Como pré-requisito desta atividade tem-se: planilha de testes, plano de testes, projeto de testes. A saída é uma atualização dos documento citados na entrada. O engenheiro de testes nesta atividade deve:

- Verificar se a abordagem e tipos de testes planejados foram adequados;
- Verificar se os ciclos de testes foram realizados com sucesso e priorizar as solicitações de mudança;
- Verificar se os critérios de conclusão e êxito descritos no Plano de Testes foram satisfeitos;

Após o entendimento do processo de testes, é preciso compreender quais os problemas, entender quais as preocupações relacionadas às atividades de testes de softwares dos gerentes de projetos e líderes de teste. Dentre as preocupações, algumas são apontadas abaixo:

- Custo do processo de testes para organização;
- Tempo total da atividade de testes dado um conjunto de pessoas com qualificações iguais;
- Tempo total da atividade de testes dado um conjunto de pessoas com diferentes qualificações;
- Viabilidade da implementação de testes automáticos, para reduzir a execução de testes manuais;
- Distribuição de defeitos por fase;
- Análise da cobertura de requisitos.

O cenário composto pelas equipes da organização, os níveis de qualificações dos trabalhadores envolvidos e a responsabilidade de cada membro da equipe nas atividades devem ser entendidos.

Alinhado com as necessidades das empresas, é importante definir que métricas de desempenho devem ser estabelecidas. As métricas de interesse para essa organização foram: tempo, custo e nível de qualidade do processo de testes.

Na coleta de dados, foi solicitada a organização em que esse estudo de caso foi aplicado a seleção das informações a serem mensuradas: tempo planejado e realizado, das atividades do processo de testes.

Entre maio e dezembro de 2008, foram coletados o tempo de execução das atividades do processo de testes. Inicialmente, foram realizadas entrevistas com o gerente e com o líder de testes de cada equipe de projeto. Foi perguntado o tamanho da equipe, a qualificação dos membros das equipes, o tempo total envolvido nas atividades de testes, o tipo de projeto e tipos de testes abordados. Em seguida, foram coletados os tempos de execução das atividades de testes, adquirida da seguinte maneira: no final de cada mês, o gerente fornecia o tempo realizado das atividades de testes, esses dados são compilados em uma ferramenta que contém o tempo de execução das atividades do processo de testes dos projetos. Para obter o tempo planejado (estimado) das atividades de testes a instituição forneceu um acesso à base de dados da ferramenta de controle de mudanças, assim, foi extraído os valores de tempos planejados das atividades.

Coletado os dados, é preciso analisar a coerência dos tempos coletados, verificação de possíveis distúrbios nos dados coletados, ou seja, verificar se os dados que foram coletados não apresentam erros que possam interferir nos resultados.

O tempo em horas do período de coleta é apresentado na Tabela 5.1, em que apresenta o tempo realizado das execuções das atividades do processo de testes.

Após a medição dos tempos de execução das atividades Planejar, Projetar, Implementar, Executar e Analisar, aplicou-se bootstrap [DH97] para se estimar os tempos de execução médios destas atividades. A técnica de bootstrap é uma abordagem ao cálculo

Tabela 5.1: Tempo das Atividades de Testes

Período/Atividade	Planejar	Projetar	Automatizar	Executar	Analisar
Maio 2008	$1,5h$	$9,25h$	$159h$	$44,5h$	$8,25h$
junho 2008	$2,5h$	$134,25h$	$247,25h$	$212,50h$	$25,25h$
julho 2008	$9h$	$103h$	$411,25h$	$219,50h$	$9,25h$
Agosto 2008	$10,25h$	$233h$	$491,75h$	$105,25h$	$8,25h$
Setembro 2008	$9h$	$32h$	$590,75h$	$296,75h$	$14,75h$
Outubro 2008	$31h$	$2h$	$90,25h$	$42,50h$	$39,50h$
Novembro 2008	$0h$	$6,25h$	$134h$	$87,5h$	$6,25h$
Dezembro 2008	$0h$	$9,25h$	$134h$	$30,75h$	$17,25h$

de intervalos de confiança de parâmetros, que pode ser aplicada nos casos em que o número de amostras são reduzidos. Os intervalos de confiança relativo ao tempo médio das atividades são apresentados na Tabela 5.2. Os intervalos foram calculados com 95% de confiança.

Tabela 5.2: Intervalos de Confiança dos Tempos de Execução

Atividade	Intervalo de Confiança (hora)
Planejar	$(1,31;14,43)$
Projetar	$(17,68;123,25)$
Implementar	$(150,75;416,50)$
Executar	(68.18;199,37)
Análisar	$(9,5;23,37)$

As médias das re-amostras e os respectivos desvios-padrão das atividades são apresentados na Tabela 5.3.

Tabela 5.3: Média e Desvios Padrão das Amostras

Atividade	Média (hora)	Desvio-Padrão (hora)
Planejar	$6,76$	$3,58$
Projetar	$65,83$	$28,53$
Implementar	$274,30$	$71,76$
Executar	$129,40$	$33,41$
Análisar	$15,81$	$3,67$

Com base nos mapeamentos apresentado neste capítulo, na Seção *Mapeamento dos*

Diagramas de Atividades - UML em SPN, foi elaborado um modelo genérico de desempenho para o processo de testes, conforme a Figura 5.15.

Neste modelo, a transição *t_in* representa o início das atividades, e uma marca em *in_Model* indica o início das atividades. O lugar *in_plan* representa o início da atividade de planejamento. O tempo da atividade de planejar está associado à transição *t_plan*, o lugar *out_plan_in_proj* representa o término da atividade de planejar e início da atividade projetar testes. O tempo da atividade projetar está associado à transição *t_proj*, o lugar *out_proj* representa o término da atividade de projetar. *D_imp* e *D_exe* representam transições de decisão. O disparo da transição *D_imp* indica que a atividade implementar testes será executada, caso contrário, o disparo de *D_exe* indica que a atividade de executar testes será a próxima do processo. O lugar *in_imp* representa a início da atividade de implementar testes, o tempo da atividade implementar está associado à transição *t_imp*, *out_imp_in_exe* representa a saída da atividade de implementar testes e o início da atividade executar testes. O tempo da atividade executar está associado à transição *t_exe*, *out_exe_in_anal* representa o término da atividade de executar e início da atividade de planejar, o tempo da atividade analisar está associado à transição *t_anal*, o lugar *out_anal* representa a fim da atividade de analisar. O disparo da transição imediata *t_out*, deposita-se uma marca no lugar *out_Model* que representa o estado final do modelo.

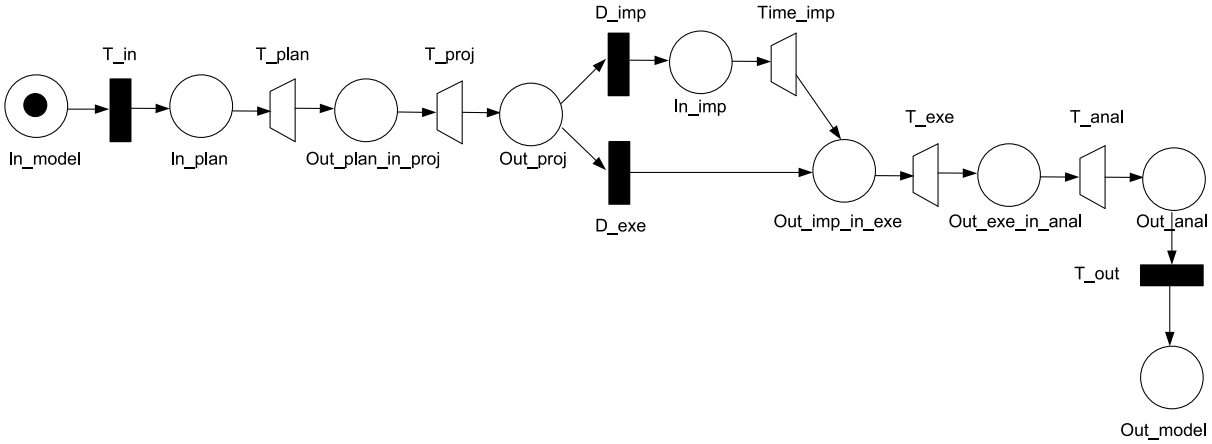


Figura 5.15: Modelo Abstrato do Processo de Testes.

Após a geração do modelo abstrato, é preciso adaptá-lo para realizar as análises de tempo, custo e quantidade de recursos utilizados, para isso é preciso transformar as cinco atividades do modelo em atividades do modelo representado na Figura 5.16 apresentada na Seção 5.3.1. O modelo é apresentado na Figura 5.16.

Neste modelo, *P*0 representa o início do processo de testes. O primeiro bloco representa a atividade "planejamento de testes"em que: *in_Plan* representa o início da atividade de planejamento; o tempo da atividade de planejar está associado à transição *t_Plan*; *out_Plan* representa o término da atividade de planejar. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar *P*3, *Team* representa o total de pessoas que podem executar a atividade. O

número de pessoas alocadas para essa atividade é representado por $NP1$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P2$ representa o número de pessoas que estão executando a atividade.

O segundo bloco representa a atividade "projetar testes"em que: *in_Proj* representa a entrada da atividade de projetar; o tempo da atividade projetar está associado à transição *t_Proj*; *out_Proj* representa o término da atividade de projetar. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P5$. O número de pessoas alocadas para essa atividade é representado por $NP2$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P6$ representa o número de pessoas que estão executando a atividade. As transições *for_imp* e *for_exe* representam transições de decisão. O disparo da transição *for_imp* indica que a atividade implementar testes será executada, caso contrário, *for_exe* pode ser disparada e a próxima atividade do processo será "executar testes".

O terceiro bloco representa a atividade "implementar testes", em que: *in_Imp* representa a entrada da atividade de automatizar; o tempo da atividade implementar está associado à transição *t_Imp*; *out_Imp* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P8$. $NP3$ representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P9$ representa o número de pessoas que estão executando a atividade.

O quarto bloco representa a atividade "executar testes", em que: *in_Exe* representa a entrada da atividade; o tempo da atividade executar está associado à transição *t_Exe*; *out_Exe* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P11$. $NP4$ representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P12$ representa o número de pessoas que estão executando a atividade.

O quinto bloco representa a atividade "analisar testes", em que: *in_Anl* representa a entrada da atividade; o tempo da atividade analisar está associado à transição *t_Anl*; *out_Anl* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P13$. $NP5$ representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P14$ representa o número de pessoas que estão executando a atividade.

Com o modelo abstrato criado, os dados obtidos (distribuição empírica) foram analisados para decidir qual a distribuição expolinomial que melhor se adapta aos tempos do processo de testes de software. A distribuição expolinomial adequada aos dados obtidos foi a hipoexponencial, escolhida com base nos cálculos dos valores das médias (μ_D) e desvios padrão (σ_D) dos tempos de execução das atividades do processo de testes, conforme o processo descrito no Capítulo 3. Com a escolha da distribuição hipoexponencial

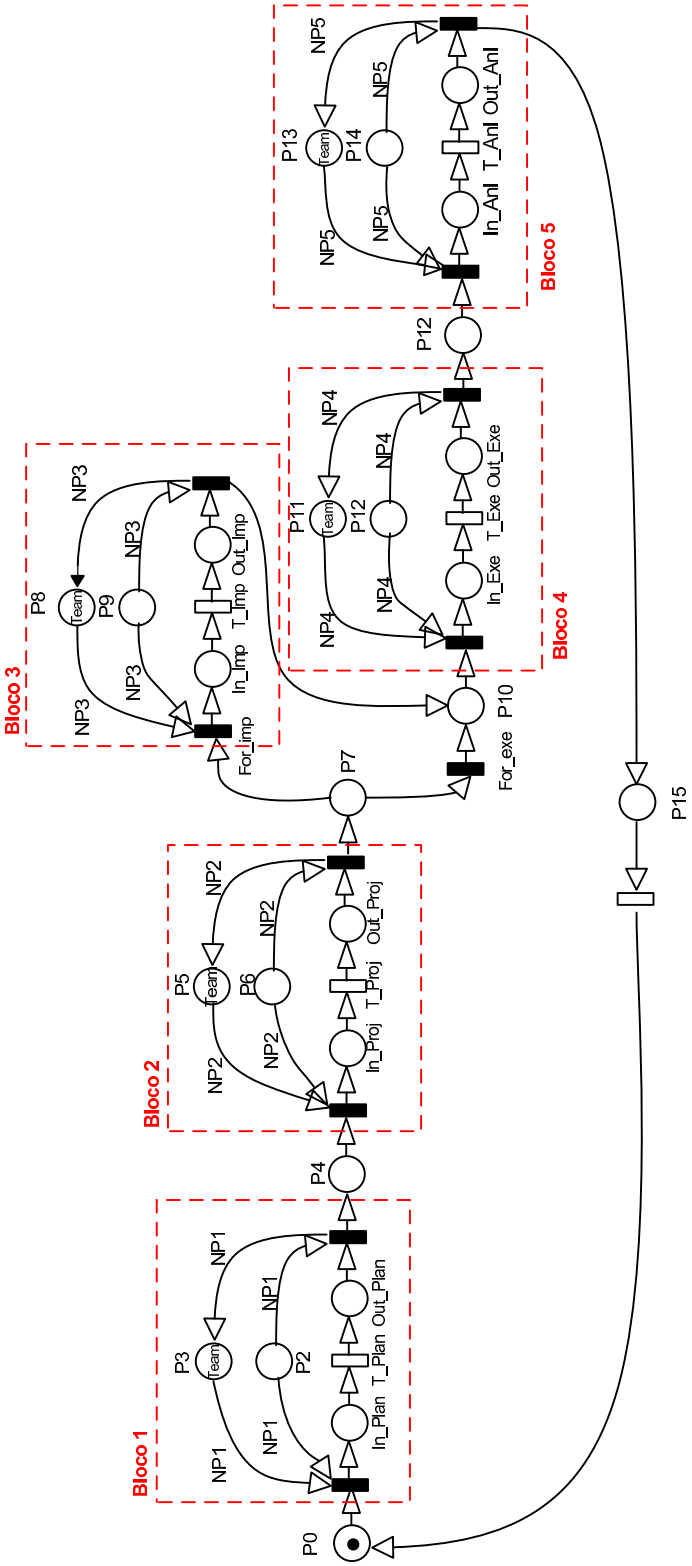


Figura 5.16: Modelo Abstrato Para Estimativa de Recurso Pessoal.

para refinar as transições estocásticas genéricas foi gerado o modelo de processo de testes refinado apresentado na Figura 5.17.

Na Figura 5.17, $P0$ representa o início do processo de testes. O primeiro bloco representa a atividade "planejamento de testes"em que: *in_Plan* representa o início da atividade; as transições *t_Plan*1 representa a primeira fase da hipoexponencial e *t_Plan*2 representa a segunda fase; *out_Plan* representa o término da atividade de planejar. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P2$. O número de pessoas alocadas para essa atividade é representado por $NP1$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P3$ representa o número de pessoas que estão executando a atividade.

O segundo bloco representa a atividade "projetar testes"em que: *in_Proj* representa o início da atividade; as transições *t_Proj*1 representa a primeira fase da hipoexponencial e *t_Proj*2 representa a segunda fase; *out_Proj* representa o término da atividade de projetar. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P4$. O número de pessoas alocadas para essa atividade, é representado por $NP2$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P5$ representa o número de pessoas que estão executando a atividade. As transições *for_imp* e *for_exe* representam transições de decisão. O disparo da transição *for_imp* indica que a atividade implementar testes será executada, caso contrário, *for_exe* pode ser disparada e a próxima atividade do processo será de executar testes.

O terceiro bloco representa a atividade "implementar testes", em que *in_Imp* representa o início da atividade de automatizar; as transições *t_Imp*1 representa a primeira fase da hipoexponencial e *t_Imp*2 representa a segunda fase; *out_Imp* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P6$. O número de pessoas alocadas para essa atividade, é representado por $NP3$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P7$ representa o número de pessoas que estão executando a atividade.

O quarto bloco representa a atividade "executar testes", em que *in_Exe* representa o início da atividade;as transições *t_Exe*1 representa a primeira fase da hipoexponencial e *t_Exe*2 representa a segunda fase; *out_Exe* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas no lugar $P8$. O número de pessoas alocadas para essa atividade, é representado por $NP4$ o qual representa o peso do arco e varia de acordo com o número de pessoas necessárias para executar essa atividade. O número de marcas no lugar $P9$ representa o número de pessoas que estão executando a atividade.

O quinto bloco representa a atividade "analisar testes", em que *in_Anl* representa o início da atividade; as transições *t_Anl*1 representa a primeira fase da hipoexponencial e *t_Anl*2 representa a segunda fase; *out_Anl* representa o término. O número total de pessoas na equipe que podem executar a atividade é representado pelo número de marcas

no lugar $P10$. O número de marcas no lugar $P11$ representa o número de pessoas que estão executando a atividade.

Foi analisado e verificado um conjunto de propriedades estruturais e comportamentais associadas ao modelo refinado. As seguintes propriedades de interesse foram satisfeitas: alcançabilidade (*reachability*), limitada (*bounded*), segura (*safe*), viva (*live*) e ausência de *deadlocks*.

Na fase de coleta de dados foram obtidos: o tempo estimado e o tempo de execução das atividades. Através da Equação (5.1) foi obtido a utilização dos colaboradores em cada atividade. UT representa o tempo realizado da atividade, e ET representa o tempo planejado da atividade. A expressão (5.2) representa a métrica computada no modelo de desempenho por meio da ferramenta *TimeNET* 4.0 [ZK07] (ver Apêndice A).

$$UP = (100 * UT)/ET \tag{5.1}$$

$$Pr = P\{\#Place = TeamSize\} \tag{5.2}$$

A Tabela 5.4 apresenta os resultados obtidos de cada atividade do processo de testes. Na Tabela 5.4 é apresentado a utilização dos colaboradores. A coluna sistema, apresenta os dados reais do processo e a coluna modelo apresenta os resultados da métrica de desempenho obtidas através do modelo de desempenho.

Tabela 5.4: Utilização dos Colaboradores

Atividade	Sistema	Modelo
Planejar	$98,28\%$	$97,28\%$
Projetar	$78,62\%$	$68,93\%$
Implementar	$67,58\%$	$58,60\%$
Executar	$80,68\%$	$80,11\%$
Análisar	$97,75\%$	$95,69\%$

Os resultados obtidos com o modelo e as respectivas medições do sistema foram comparados por meio do Teste T-emparelhado [Lil05]. Pôde-se constatar com 95% de grau de confiança, intervalo de confiança $(-1.12;10,04)$ e a estatística do teste T igual a $2,22$, que os resultados não evidenciam qualquer discrepância entre os dados medidos e os valores obtidos do modelo.

É importante destacar que validação é um processo de convencimento e que para isso mais estudos poderão e deverão ser conduzidos para que se tenha maior confiança no processo. No entanto, como dito para o estudo realizado, não temos evidências que refutem o processo.

5.5 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os mapeamentos do diagrama de atividades da UML para modelos SPN. Dentre os elementos mapeados, pode-se citar:atividade,transições, barras de sincronizações, estado inicial e final. Em seguida foram apresentados os modelos para alocações de recursos. Finalmente, um modelo de avaliação de desempenho de processo de testes foi elaborado para verificação de medidas de desempenho de um processo de testes adotado por uma empresa, ainda foi apresentada a validação do modelo proposto.

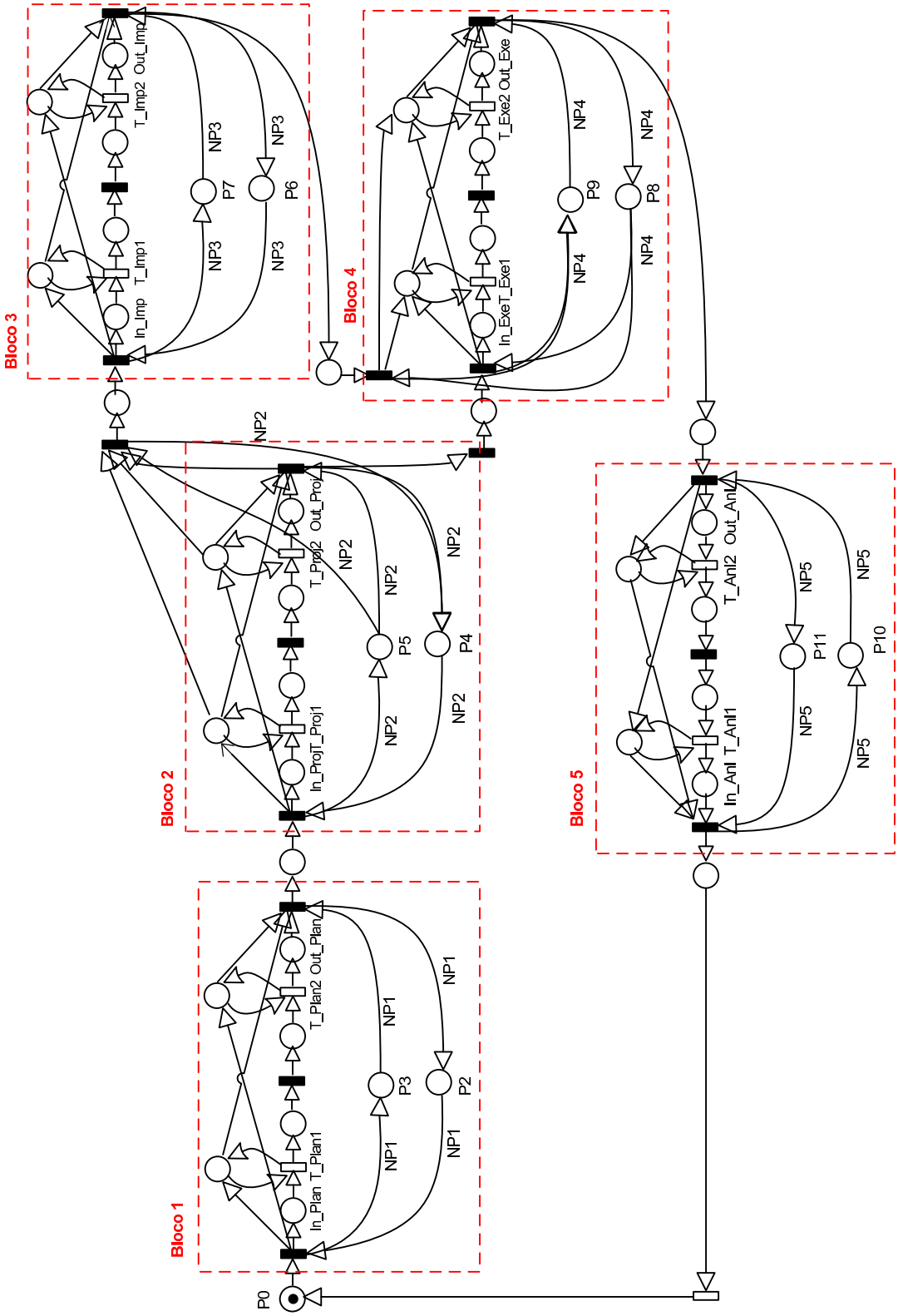


Figura 5.17: Modelo Refinado Para Estimativa de Recurso Pessoal.

CAPÍTULO 6

ESTUDOS DE CASO

Neste capítulo, apresentam-se os estudos de caso com intuito de avaliar situações de interesse prático. Inicialmente, são apresentados três estudos voltados ao apoio de previsão de melhores estimativas de custo e tempo das atividades do processo de testes de software. Após isso, segue uma comparação do processo de testes com e sem a atividade "automatizar testes"a fim de verificar custo e tempo da aplicação da atividade e, finalmente, o quinto cenário verifica a qualidade do processo de testes executado por colaboradores de diferentes qualificações.

6.1 INTRODUÇÃO

Os estudos de casos deste capítulo baseiam-se no processo de teste adotado por uma instituição de software do Recife-PE, a mesma do Capítulo 5.

A equipe do projeto em que se aplica o estudo de caso compõe-se de sete engenheiros de testes com qualificação e salários semelhantes. Esses profissionais, quando não utilizados neste projeto, são realocados para outro e, além das atividades do processo de testes de software, participam de outras atividades do processo de software como um todo. Esse projeto tem as seguintes características: o tempo de duração é de três anos, realiza em média, um *release* com os seguintes testes: funcionais, de configuração, de segurança, desempenho e concorrência. Mensalmente, a *suite* de testes apresenta mesmo tamanho.

Há cinco estudos neste capítulo: os três primeiros direcionam-se para a melhoria de estimativa de custo e tempo das atividades do processo de testes de software. O quarto mostra uma comparação da execução das atividades do processo de testes com a presença da atividade de automação de testes e sem a execução desta. Finalmente, o quinto estudo apresenta uma estimativa de quantidades de defeitos escapados associando a diferentes níveis de colaboradores que executam as atividades de testes, avaliando assim, o processo de testes, considerado por diferentes colaboradores.

6.2 ESTUDO DE CASO: ALOCAÇÃO DE ENGENHEIROS

O objetivo deste estudo é prever estimativas mais exatas para as atividades de testes, através de modelos estocásticos. Neste estudo, realiza-se a estimativa do tempo das atividades do processo de testes, que visa a adoção de modelos estocásticos para realizar estimativas das atividades do processo, variando a quantidade de engenheiros em cada uma das atividades do processo

Alguns cenários da equipe foram estabelecidos a fim de verificar custo e tempo da composição de engenheiros de testes executando as atividades do processo de testes. As Tabelas (6.1), (6.2), (6.3) mostram os cenários criados. Cada cenário possui as composições de colaboradores, ou seja, em cada atividade do processo de testes existe um número de colaboradores que irão executar a atividade. A junção dessas composições forma um cenário.

Nas Tabelas (6.1), (6.2), (6.3), as colunas mostram cada cenário. Os cenários são compostos por um número de engenheiros em cada atividade do processo de testes da instituição. Nas últimas duas linhas, computa-se o tempo e o custo obtido no modelo de desempenho proposto na Figura 5.17. Os valores obtidos nas duas últimas linhas das tabelas são extraídos das seguintes métricas: **Tempo de execução** apresentado na expressão (6.1), por meio da ferramenta *TimeNET* 4.0 [ZK07], que serve para simular o tempo das atividades do processo de teste, em que P{#P15>0} representa a utilização, μ representa o tempo da transição exponencial. O **Custo Médio** definido pela expressão (6.2), em que N representa o número de colaboradores, C(Collaborator) o custo da hora do colaborador, *Time* as horas trabalhadas e K(Team)a média de colaboradores envolvidos nas atividades.

Tabela 6.1: Tabela de Composições 1

Atividade	C1	C2	C3	C4	C5	C6
Planejar	7	7	7	7	1	7
Projetar	7	7	7	1	1	7
Implementar	7	3	7	7	3	7
Executar	7	7	3	7	5	1
Analisar	1	1	1	1	1	1
Tempo	$81,43h$	$126,65h$	$103,16h$	$234,17h$	$299,24h$	$179,21h$
Custo Médio	$R\$175,45$	$R\$316,62$	$R\$257,90$	$R\$636,33$	$R\$1700,00$	$R\$468,96$

Tabela 6.2: Tabela de Composições 2

Atividade	C7	C8	C9	C10	C11	C12
Planejar	7	7	7	1	1	1
Projetar	7	7	7	1	1	1
Implementar	7	3	6	1	2	6
Executar	6	7	7	7	4	6
Analisar	1	1	1	1	1	1
Tempo	$84,15h$	$126,65h$	$87,08h$	$451,61h$	$344,11h$	$255,87h$
Custo Médio	$R\$187,83$	$R\$316,62$	$R\$194,37$	$R\$2565,96$	$R\$2389,65$	$R\$1066,12$

$$Time = (1/P\{\#P15 > 0\}) * \mu \tag{6.1}$$

Tabela 6.3: Tabela de Composições 3

Atividade	C13	C14	C15	C16	C17	C18
Planejar	1	1	2	1	1	5
Projetar	2	1	3	2	4	7
Implementar	2	6	3	2	4	2
Executar	4	0	4	6	7	5
Analisar	1	1	1	1	1	1
Tempo	$255,42h$	$249,53h$	$177,12h$	$245,91h$	$139,29h$	$173,63h$
Custo Médio	$R\$1596,37$	$R\$1732,85$	$R\$851,53$	$R\$1280,78$	$R\$512,09$	$R\$542,59$

$$AverageCost = [(N * C(Collaborator) * Time)/K(Team)] \tag{6.2}$$

Os resultados de tempo e custo da composição dos diferentes números de engenheiros podem ser apresentados na forma gráfica na Figura 6.1, que mostra a relação entre custo e tempo das diferentes composições do processo de testes. Cada coluna equivale a um cenário diferente de número de engenheiros. Com base neste gráfico, gerentes de projetos e líderes de testes podem definir a estimativa, considerando o custo e o tempo da execução das atividades, ou seja, é possível estimar custo, tempo e utilização associados ao processo de acordo com a formação de quadro pessoal envolvido nas atividades do processo. Quando os gerentes e líderes forem estimar tempo e custo das atividades, podem verificar, através do modelo de desempenho, observando as medidas com a adição ou remoção de número de colaboradores nas atividades do processo de testes.

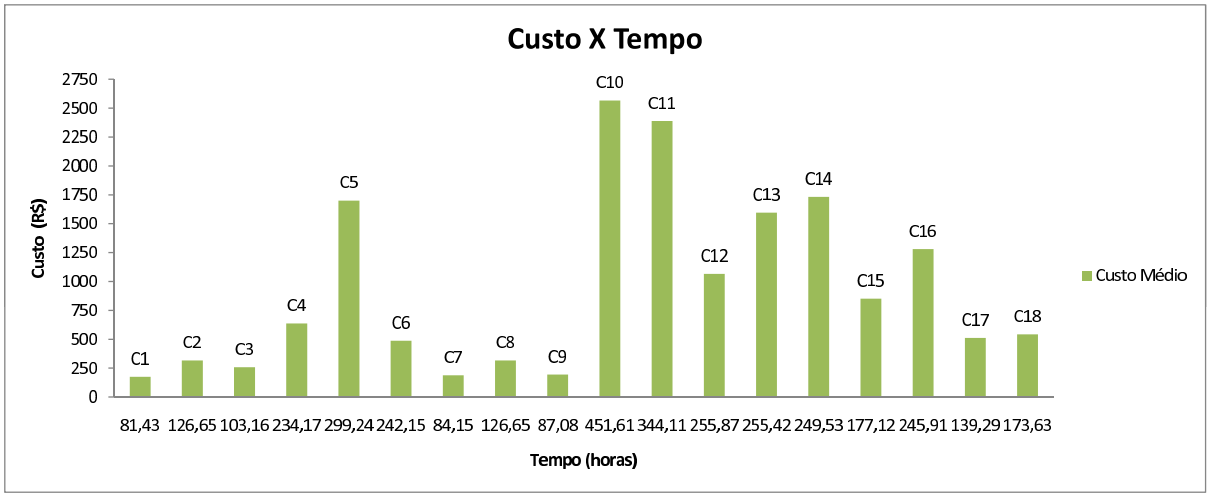


Figura 6.1: Gráfico das Composições Avaliadas.

Na fase de planejamento, os gerentes podem avaliar qual melhor cenário ou propor configurações a serem avaliadas para verificação do custo e tempo. Nesse estudo de caso, foram obtidos as estimativas de tempos das atividades do processo e o custo real do processo executado.

Dentre as configurações avaliadas, o melhor tempo de execução e custo foi do cenário C1, que é composto por sete engenheiros, estes executam todas as atividades do processo em paralelo com exceção da atividade de analisar testes que é executado por apenas um engenheiro. O custo médio do cenário C1 por colaborador é de $175,49$ (cento e setenta e cinco reais e quarenta e nove centavos), com o tempo de $81,43$ horas. O cenário que apresentou o pior resultado foi C10, em que o tempo foi de $451,61$ horas, a um custo médio de $2.565,96$ (Dois mil, quinhentos e sessenta e cinco reais e noventa e seis centavos).

6.3 ESTUDO DE CASO: ALOCAÇÃO DE ENGENHEIROS E TRAINEES

Neste estudo, o objetivo é verificar tempo e custo, alocando profissionais com níveis de experiência diferentes e, por conseqüência, valor de salários distintos. Os níveis que serão tratados aqui será do engenheiro (um cargo mais senior) e do trainee (um cargo junior).

Os salários em reais recebidos pelas duas categorias são: $2.000,00$(dois mil reais) para o engenheiro e $1200,00$ (mil e duzentos reais) para o trainee. O tempo de trabalho é equivalente a oito horas diárias para os dois cargos, porém, com base em informações dos especialistas do instituto de software em que se verifica esse estudo, a cada uma hora que o especialista realiza uma atividade de testes, o trainee executa a mesma atividade num tempo médio de uma hora e quinze minutos.

Seguindo os passos da metodologia proposta no Capítulo 4, para verificar os diferentes níveis, elaborou-se o modelo abstrato que representa os três níveis de qualificações (Engenheiro, *Trainee* e Estagiário). Este é representado pela Figura 6.2. Neste modelo, $P0$ representa o início das atividades. O número total de engenheiros, *Trainees* e estagiário na equipe, que podem executar a atividade, é representado pelo número de marcas nos lugares *Profissional*, *Trainee* e *Estagiário* respectivamente.

O primeiro bloco representa a atividade "planejamento de testes"em que: *in_Plan* representa o início atividade; o tempo da atividade está associado à transição *t_Plan*; *out_Plan* representa o término da atividade. O número de engenheiros, *Trainee* e estagiários alocados para essa atividade está representado por $NPA1$,$NTA1$ e $NSA1$ respectivamente, os quais representam os pesos dos arcos e varia de acordo com o número de engenheiros, *Trainees* e estagiários necessários para executar essa atividade. O número de marcas nos lugares $EGA1$,$CTA1$, $SA1$ representa a quantidade de engenheiros, *Trainee* e estagiários respectivamente, que estão executando a atividade.

O segundo bloco representa a atividade "projetar testes"em que: *in_Proj* representa o início da atividade; o tempo da atividade está associado à transição *t_Proj*; *out_Plan* representa o término da atividade. O número de engenheiros, *Trainee* e estagiários alocados para essa atividade está representado por $NPA2$,$NTA2$ e $NSA2$ respectivamente, os quais representam os pesos dos arcos e variam de acordo com o número de engenheiros, *Trainees* e estagiários necessários para executar essa atividade. O número de marcas nos lugares $EGA2$,$CTA2$, $SA2$ representa a quantidade de engenheiros, *Trainee* e estagiários respectivamente, que estão executando a atividade. As transições *for_imp* e *for_exe* representam transições de decisão. O disparo da transição *for_imp* indica que a

atividade implementar testes será executada, caso contrário, *for_exe* pode ser disparada, e a próxima atividade do processo será "executar testes".

O terceiro bloco representa a atividade "implementar testes"em que: *in_Imp* representa o início da atividade; o tempo da atividade está associado à transição *t_Imp*; *out_Imp* representa o término da atividade. O número de engenheiros, *Trainees* e estagiários alocados para essa atividade está representado por $NPA3$,$NTA3$ e $NSA3$ respectivamente, os quais representam os pesos dos arcos e variam de acordo com o número de engenheiros, *Trainee* e estagiários necessários para executar essa atividade. O número de marcas nos lugares $EGA3, CTA3$, $SA3$ representa a quantidade de engenheiros, *Trainee* e estagiários respectivamente, que estão executando a atividade.

O quarto bloco representa a atividade executar testes em que: *in_Exe* representa o início da atividade; o tempo da atividade está associado à transição *t_Exe*; *out_Exe* representa o término da atividade. O número de engenheiros, *Trainees* e estagiários alocados para essa atividade está representado por $NPA4$,$NTA4$ e $NSA4$, respectivamente, os quais representam os pesos dos arcos e variam de acordo com o número de engenheiros, *Trainee* e estagiários necessários para executar essa atividade. O número de marcas nos lugares $EGA4, CTA4$, $SA4$ representa a quantidade de engenheiros, *Trainees* e estagiários respectivamente, que estão executando a atividade.

O quinto bloco representa a atividade analisar testes em que: *in_Anl* representa o início da atividade; o tempo da atividade está associado à transição *t_Anl*; *out_Anl* representa o término da atividade de planejar. O número de engenheiros, *Trainees* e estagiários alocados para essa atividade está representado por $NPA5$,$NTA5$ e $NSA5$ respectivamente, os quais representam os pesos dos arcos e variam de acordo com o número de engenheiros, *Trainee* e estagiários necessários para executar essa atividade. O número de marcas nos lugares $EGA4, CTA4$, $SA4$ representam a quantidade de engenheiros, *Trainee* e estagiários respectivamente, que estão executando a atividade.

Após a construção do modelo abstrato e seguindo os passos da metodologia, foi gerado o modelo refinado, que é representado na Figura 6.3, em que os blocos representam as cinco atividades do processo de testes, e cada bloco é representado pelo modelo refinado da Figura 6.4.

O número de marcas nos lugares *Profissional*, *Trainees* e *Estagiário*, representa o número total de engenheiros, *Trainee* e estagiários respectivamente. *In_A* representa o início da atividade. A transição t_A1 representa a primeira fase da hiporexponencial, já t_A2 representa a segunda fase da hiporexponencial. *Out_A* representa o término da atividade. O número de engenheiros, *Trainee* e estagiários, alocados para essa atividade, representado por NPA,NTA e NSA, respectivamente, representa os pesos dos arcos e varia de acordo com o número de engenheiros, *Trainee* e estagiários necessários para executar essa atividade. O número de marcas nos lugares PA, TA, EA representa o número de engenheiros, *Trainees* e estagiários, respectivamente, que estão executando a atividade.

Este estudo de caso tem como objetivo verificar o tempo de execução total em um mês e o custo equivalente das atividades do processo de testes considerando que dois perfis de

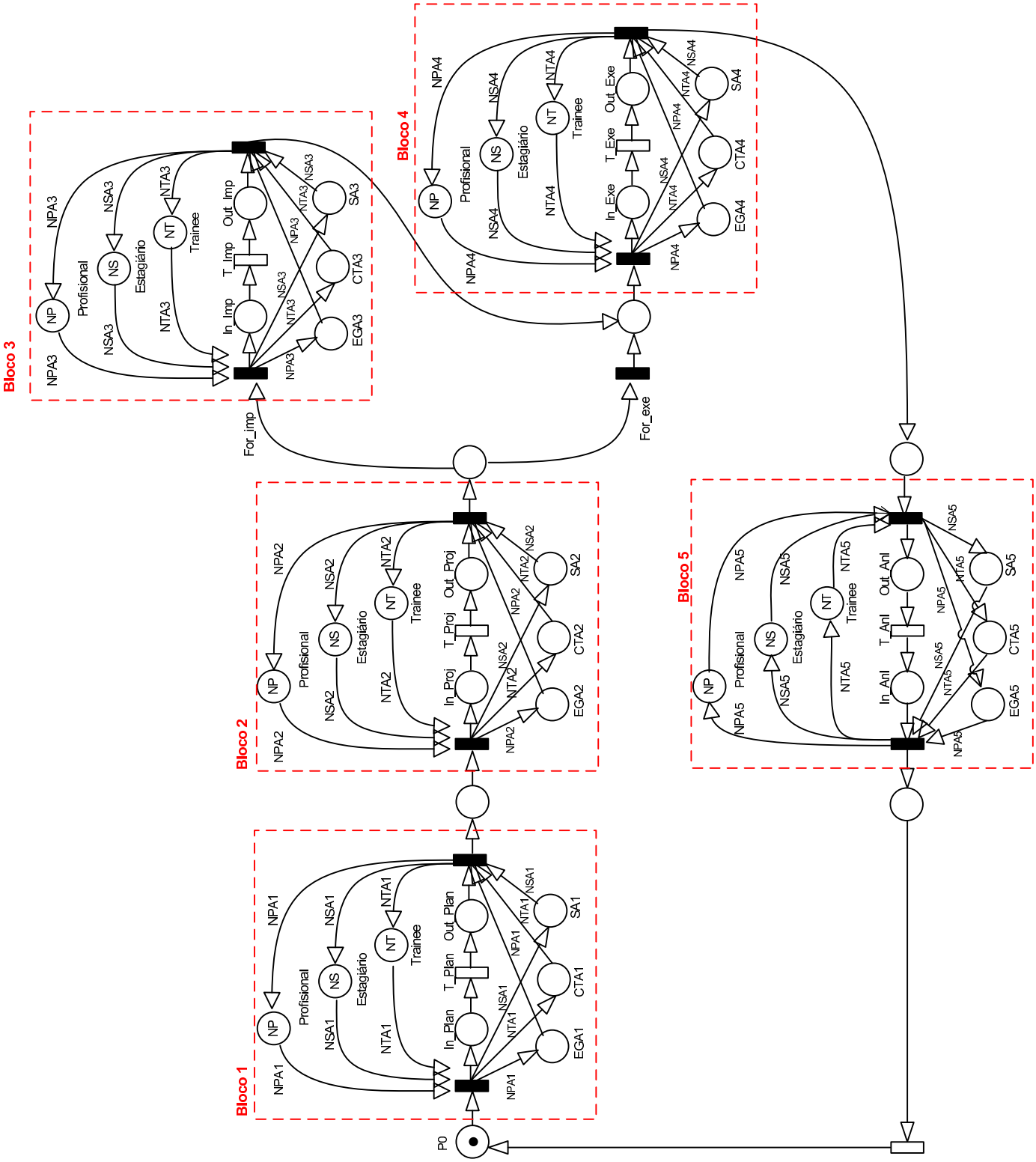


Figura 6.2: Modelo Abstrato de Alocações de Recursos Pessoais.

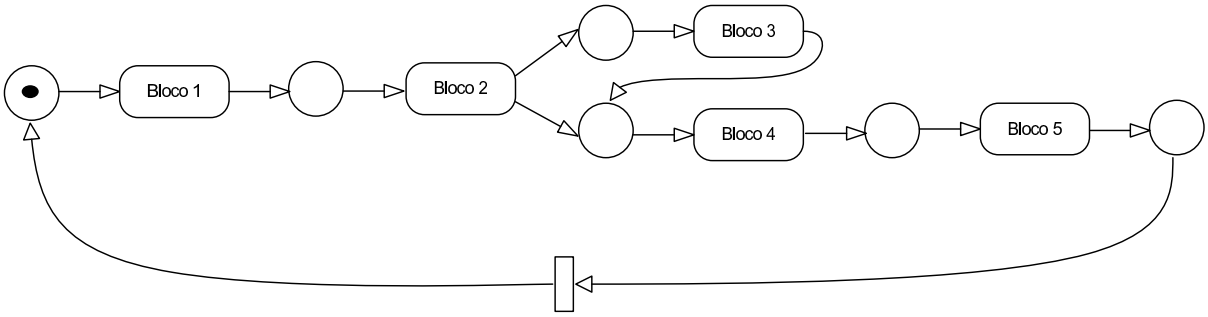


Figura 6.3: Modelo Refinado de Alocações de Recursos Pessoais.

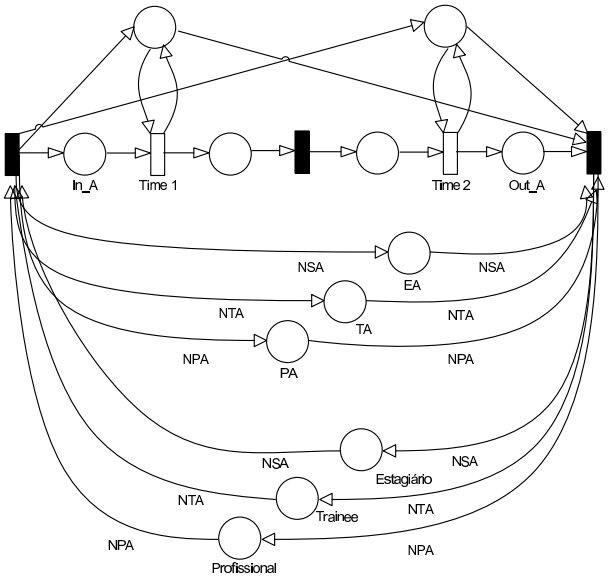


Figura 6.4: Modelo Refinado de Atividade de Alocações de Recursos Pessoais.

colaboradores, engenheiro e *trainee*, estarão executando as atividades do processo. Nesse estudo, foi considerado que toda a equipe estava envolvida em todas as cinco atividades do processo com exceção da atividade de analisar testes, que deve ser executado por um engenheiro que exerça a função de líder de testes. A Tabela 6.4 mostra os cenários com as composições avaliadas. Cada cenário possui as composições de colaboradores, ou seja, em cada coluna da Tabela 6.4 existe a quantidade de colaboradores que estarão executando todas as atividades do processo de testes.

Tabela 6.4: Engenheiros e *Trainees*

Atividade	C 1	C 2	C 3	C 4	C 5	C 6	C 7
Engenheiros	7	6	5	4	3	2	1
Trainees	0	1	2	3	4	5	6
Tempo	$81,43h$	$176,19h$	$153,49h$	$167,11h$	$205,74h$	$292,50h$	$562,47h$
Custo Total	$1017,87$	$2075,51$	$1699,13$	$1731,26$	$1983,33$	$2612,02$	$4617,87$

Os resultados de tempo e custo da composição das diferentes quantidades de engenheiros e *trainees* podem ser apresentados na forma gráfica na Figura 6.5, que mostra a relação entre custo e tempo das diferentes composições do processo de testes. As barras duplas equivalem ao custo e tempo de um cenário com diferente números de engenheiros e *trainees*.

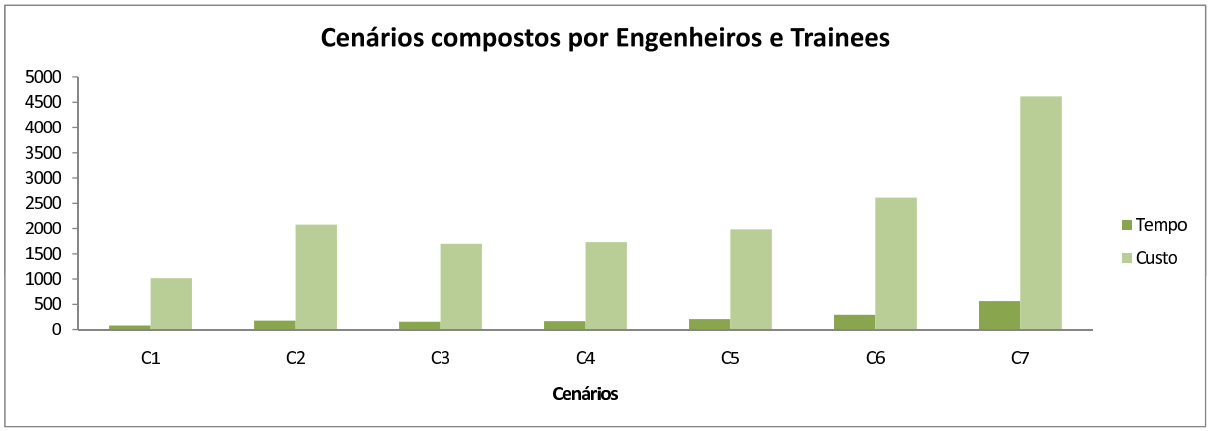


Figura 6.5: Gráfico das Composições Avaliadas - Engenheiros e *Trainees*.

O tempo da execução do processo de testes é calculado através da expressão 6.1. O valor da hora pode ser obtido através da expressão 6.3, em que *NE* significa o número de engenheiros envolvidos que pode variar até m Engenheiros, C(*Engenheiro*) representa o custo de um engenheiro, *NT* representa o número de *Trainees*, C(*Trainee*) o custo de um *trainee*, T(Engenheiro) representa o tempo de trabalho mensal de um engenheiro e T(*Trainee*) o tempo de trabalho mensal de um *trainee*. O custo da execução do processo pode ser obtido pelo resultado da expressão 6.1 multiplicado pela 6.3 e é expresso por 6.4.

$$ValorHora = \frac{(\sum_{i=1}^{m} NE_i \times C(Engenheiros) + \sum_{j=1}^{n} NT_j \times C(Trainee)}{(\sum_{i=1}^{m} NE_i \times T(Engenheiros) + \sum_{j=1}^{m} NT_j \times T(Trainee)} \tag{6.3}$$

$$Custo = Time * ValorHora \tag{6.4}$$

Dentre as configurações avaliadas, o melhor tempo de execução e melhor custo foi do cenário C1, que é composto apenas por colaboradores de nível engenheiros. É importante verificar que mesmo a hora do colaborador *trainee* sendo mais barata do que o engenheiro, ainda assim, comparando os dois colaboradores, o tempo de execução das atividades pelo engenheiro compensa o custo da mão de obra deste perfil, ou seja, investir em profissionais de nível engenheiro é melhor do que o investimento no *trainee*. O custo médio do cenário C1 foi 1.017, 87 (Um mil, dezessete reais e oitenta e sete centavos), em que o tempo foi 81, 43 horas. O cenário que apresentou o pior resultado foi C7, em que o tempo foi de 562, 47 horas, a um custo médio de 4.617, 87 (Quatro mil, seiscentos e dezessete reais

e oitenta e sete centavos). Dentre as configurações experimentadas com a participação de um *trainee*, a melhor foi C3, composta por 5 engenheiros e 2 *trainees* executando as atividades de testes. Esta configuração apresenta um tempo de $153,49$ horas, a um custo de $1.699,13$ (Um mil, seiscentos e noventa e nove reais e treze centavos).

6.4 ESTUDO DE CASO: ALOCAÇÃO DE ENGENHEIROS, TRAINEE E ESTAGIÁRIO

Neste estudo, também são verificadas questões de desempenho, tempo e custo, porém para três níveis de profissionais com qualificações diferentes envolvidos nas atividades do processo de testes. Os profissionais são: engenheiro, *trainee* e estagiário. O tempo de trabalho do engenheiro e do *Trainee* é de oito horas ao dia, já o estagiário é de seis horas por dia. O salário de um Engenheiro é equivalente a $2.000,00$ reais, do trainee $1.200,00$ reais e do estagiário $540,00$ reais. Com base em informações dos especialistas do instituto de software, em que esse estudo de caso foi aplicado, a cada uma hora que o engenheiro realiza uma atividade de testes, o trainee executa a mesma atividade em média uma hora e mais quinze minutos, o estagiário em uma hora e trinta minutos a mais do que o engenheiro.

O modelo abstrato está representado na Figura 6.2, e o refinado na Figura 6.3. Foram avaliados dez cenários, constantes nas Tabelas (6.5) e (6.6). Para as composições, é definido que todos estão participando das cinco atividades do processo de testes com exceção da atividade de analisar testes, que deve ser executado por um engenheiro que exerça a função de líder de testes. As Tabelas mostram (6.5) e (6.6) o número de engenheiro, *Trainee* e estagiário que estão executando o processo, as últimas linhas apresentam o tempo de execução e o custo do cenário.

Tabela 6.5: Composições 1: Engenheiros, Trainee e Estagiário

Perfil	C 1	C 2	C 3	C 4	C 5
$Engenheiros$	7	5	4	3	2
$Trainees$	0	1	2	3	4
$Estagiarios$	0	1	1	1	1
$Tempo$	$81,43h$	$373,04h$	$359,03h$	$390,82h$	$474,88h$
$Custo\ Total$	$R\$1017,87$	$R\$4054,94$	$R\$3636,97$	$R\$3669,80$	$R\$4107,71$

Os resultados de tempo e custo da composição das diferentes quantidades de engenheiros, *trainees* e estagiários podem ser apresentados na forma gráfica na Figura 6.6, que mostra a relação entre custo e tempo das diferentes composições do processo de testes. As barras duplas equivalem ao custo e tempo de um cenário com diferença de números de engenheiros, *trainees* e estagiários.

Dos cenários avaliados, é verificado que C1, possui sete engenheiros, é a composição com um melhor custo e tempo. O tempo de C1 foi de $81,43$ horas e o custo de $1.017,87$ (Um mil, dezessete reais e oitenta e sete centavos). Dos cenários avaliados com diferentes

Tabela 6.6: Composições 2:Engenheiros, Trainee e Estagiário

Perfil	C 6	C 7	C 8	C 9	C 10
Engenheiros	2	2	3	4	1
Trainees	3	2	1	0	3
Estagiarios	2	3	3	3	4
Tempo	$382,86h$	$376,73h$	$275,98h$	$194,77$	$620,90h$
Custo Total	$R\$3193,05$	$R\$3021,37$	$R\$2434,14$	$R\$1873,68$	$R\$4302,83$

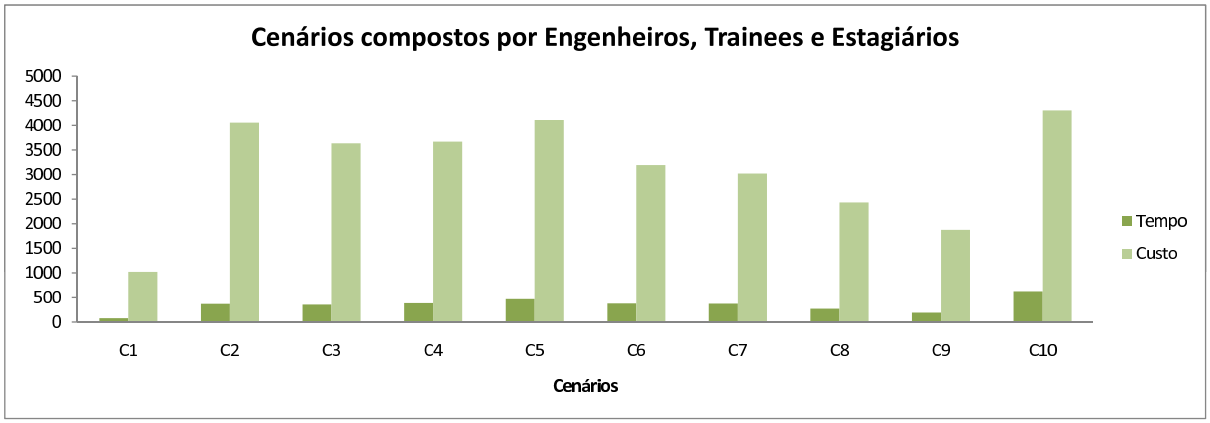


Figura 6.6: Gráfico das Composições Avaliadas - Engenheiros, *Trainees* e Estagiários.

recursos, é verificado que C9, possui quatro engenheiros e três estagiários, é a composição com um melhor custo e tempo possuindo recursos diferentes. O tempo de C9 foi de $194,77$ horas e o custo de $1.873,68$ (Um mil, oitocentos e setenta e três reais e sessenta e oito centavos). O cenário que apresentou um pior resultado foi C10 com custo de $4.302,84$ (quatro mil, trezentos e dois reais e oitenta e quatro centavos), com $620,90$ horas.

Avaliando os cenários, verifica-se que é melhor investir em termos de custo nos estagiários do que investir em profissionais do nível de *trainee* para as atividades do processo de testes. Comparando os cenários, verifica-se que o investimento no profissional de nível *trainee* não tem sido muito eficiente. Por exemplo, ao comparar os cenários C6 e C7, que possuem o mesmo número de engenheiros, observa-se que em C7, o tempo e o custo é menor que C6. Nesse cenário ocorre, pois, que uma grande parte dos *trainees* desta instituição são pessoas que terminaram, em pouco tempo, a graduação. Esses, apesar de ter um nível de maturidade e cuidado na execução dos testes, o tempo de execução de uma atividade de testes não é muito distante de um estagiário. Este mesmo não sendo formado, está há um certo período na organização e, mesmo executando as atividades em tempo mais longo que os outros colaboradores de níveis superior, o baixo custo da hora paga ao estagiário compensa. Verifica-se que para a instituição, investir em estagiários é melhor do que nos *trainees*, cuja maior parte compõe-se de pessoas recém-formadas com pouca experiência na atividade do processo de testes.

6.5 ESTUDO DE CASO: DESEMPENHO DA ATIVIDADE IMPLEMENTAR TESTES

A automação dos testes de software há muito é reconhecida por seu potencial para melhoria dos testes, reduzindo testes manuais redundantes e maximizando a precisão dos testes e sua capacidade de repetição. A implementação de teste tem como objetivo, automatizar algumas tarefas manuais com o uso de algumas ferramentas de software.

A demanda por automação de teste é forte, porque o teste puramente manual do início ao *release* pode ser tedioso e propenso a erros. Um nível de automação para algumas regras de negócios é possível, através de várias ferramentas comerciais ou ferramentas desenvolvidas no âmbito das grandes organizações [Tia05].

A atividade *implementar testes* do processo utilizado, descrito no Capítulo 5, é opcional, ou seja, pode ser executada ou não de acordo com uma escolha do gerente do projeto. Esta atividade foi executada pela equipe do projeto, mas não foi verificado se a inclusão da atividade implementar testes, conhecida também como automatizar testes, no projeto utilizado, pôde reduzir o número de horas das atividades do processo e se a implantação dessa atividade compensa o investimento. O objetivo deste estudo é verificar se, de acordo com o projeto e os colaboradores envolvidos nas atividades, a implementação de testes pode diminuir o tempo de realização da atividades, especificamente das atividade *executar testes*. É importante destacar que o estudo foi aplicado nas fases iniciais de implantação do processo de testes dentro do projeto e que para isso mais estudos poderão e deverão ser conduzidos para que se tenha maior confiança no processo.

Com base nos estudos anteriores, é verificado o tempo total do processo de testes com a atividade de automatização, o tempo sem a automatização e também o custo total com ou sem a atividade de automatizar testes.

- **Implementar Testes: Engenheiros e *Trainees***:

  A Tabela 6.7 apresenta os cenários de todas as composições mostrada em 6.3. Na Tabela 6.7, é apresentada uma comparação dos resultados, tempo e custo, com e sem a atividade de implementação de testes.

Tabela 6.7: Implementar Testes: Engenheiros e *Trainees*

Composição	Tempo c/Imp	Tempo s/Imp	Custo c/Imp	Custo s/Imp
$C1$	$81,43h$	$105,86h$	$1.017,87$	$1.323,25$
$C2$	$176,19h$	$374,05h$	$2.075,52$	$4.406,30$
$C3$	$153,49h$	$361,24h$	$1.699,13$	$3.998,92$
$C4$	$167,11h$	$368,93h$	$1.731,25$	$3.822,11$
$C5$	$205,74h$	$390,73h$	$1.983,33$	$3.766,63$
$C6$	$292,50h$	$439,51h$	$2.612,02$	$3.924,82$
$C7$	$562,47h$	$592,16h$	$4.617,87$	$4.861,63$

Os resultados de tempo com e sem a atividade de implementar testes podem ser

apresentados na forma gráfica na Figura 6.7, que mostra a relação entre tempo do processo de testes com e sem a atividade implementar testes. As barras duplas equivalem ao tempo, com e sem a atividade implementar testes, de um cenário com diferentes números de engenheiros e *trainees*. De acordo com os resultados, o tempo do processo de testes é menor quando a atividade implementar testes é executada.

Já os resultados do custo, com e sem a atividade de implementar testes, podem ser apresentados na forma gráfica na Figura 6.8, que mostra a relação entre tempo do processo de testes, com e sem a atividade implementar testes. As barras duplas equivalem ao custo, com e sem a atividade implementar testes, de um cenário com diferentes números de engenheiros e *trainees*. De acordo com os resultados, o custo do processo de testes é menor quando a atividade implementar testes é executada.

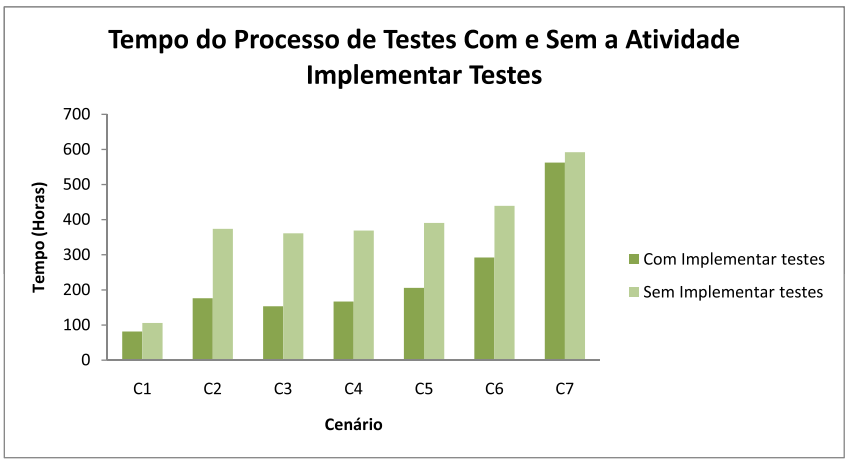


Figura 6.7: Tempo do Processo de Testes Com e Sem a Atividade Implementar Testes - Engenheiros e *trainees*.

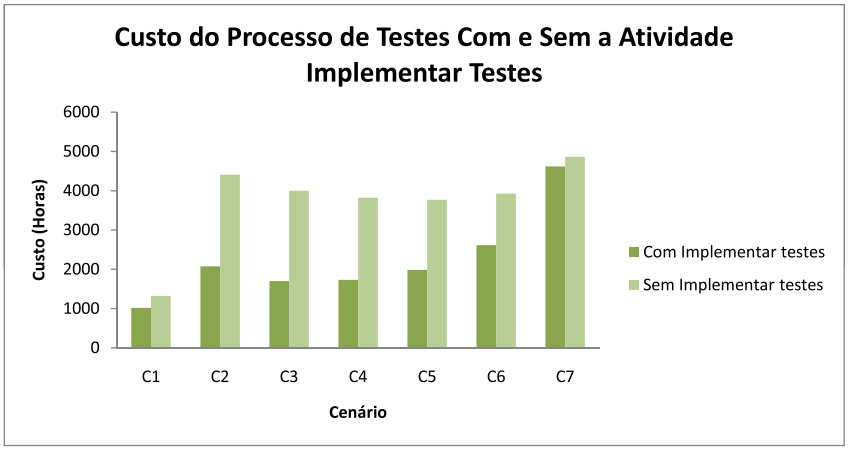


Figura 6.8: Custo do Processo de Testes Com e Sem a Atividade Implementar Testes - Engenheiros e *trainees*.

Na Tabela 6.7, verifica-se válido o investimento nas atividades de implementação

de testes para todas as composições.

- **Implementar Testes: Engenheiros, *Trainees* e Estagiário**:

  A Tabela 6.8 apresenta os cenário de todas as composições mostradas em 6.5. Na Tabela 6.8, é apresentada uma comparação dos resultados, tempo e custo, com e sem a atividade de implementação de testes.

Tabela 6.8: Implementar Testes: Engenheiros, *Trainees* e Estagiário

Composição	Tempo c/Imp	Tempo s/Imp	Custo c/Imp	Custo s/Imp
$C1$	$81,43h$	$105,86h$	$1.017,87$	$1.323,25$
$C2$	$373,04h$	$485,80h$	$4.054,94$	$5.280,64$
$C3$	$359,03h$	$477,74h$	$3.636,97$	$4.839,50$
$C4$	$390,82h$	$495,68h$	$3.669,79$	$4.654,43$
$C5$	$474,88h$	$543,14h$	$4.107,71$	$4.698,16$
$C6$	$382,86h$	$481,12h$	$3.193,05$	$4.012,54$
$C7$	$376,73h$	$483,57h$	$3.021,37$	$3.878,23$
$C8$	$275,98h$	$390,14h$	$2.434,14$	$3.441,03$
$C9$	$194,77h$	$373,68h$	$1.873,68$	$3.594,80$
$C10$	$620,90h$	$620,90h$	$4.302,83$	$4.302,83$

Os resultados de tempo, com e sem a atividade de implementar testes, podem ser apresentados na forma gráfica na Figura 6.9, que mostra a relação entre tempo do processo de testes, com e sem a atividade implementar testes. As barras duplas equivalem ao tempo, com e sem a atividade implementar testes, de um cenário com diferentes números de engenheiros, *trainees* e estagiários. De acordo com os resultados, o tempo do processo de testes é menor quando a atividade implementar testes é executada.

Já os resultados do custo, com e sem a atividade de implementar testes, podem ser apresentados na forma gráfica na Figura 6.10, que mostra a relação entre tempo do processo de testes, com e sem a atividade implementar testes. As barras duplas equivalem ao custo, com e sem a atividade implementar testes, de um cenário com diferentes números de engenheiros, *trainees* e estagiários. De acordo com os resultados, o custo do processo de testes é menor quando a atividade implementar testes é executada.

Após a avaliação, é verificado que as atividades de automatização com colaboradores de diferentes categorias é eficiente, pois reduz o custo do processo de teste.

Dentre as composições avaliadas, os resultados mostram que, para esta equipe de testes e para este tipo de projeto, mesmo com pessoas de diferentes níveis de conhecimento, a adoção da atividade implementar testes, reduz tempo e custo do processo de testes.

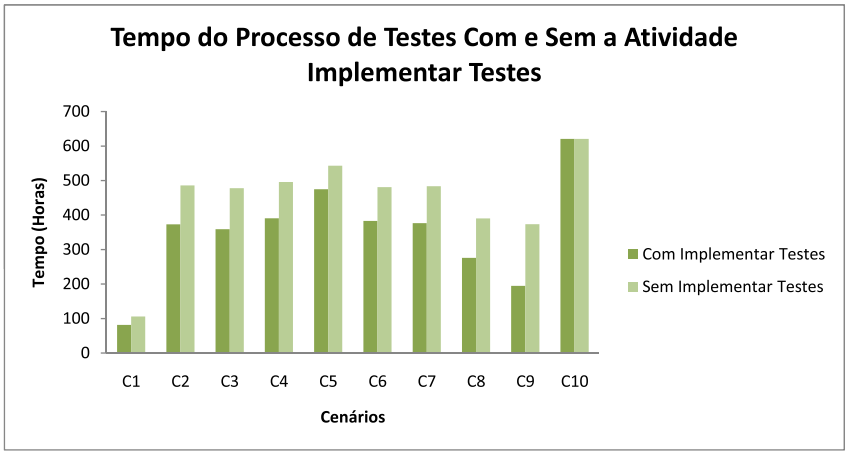


Figura 6.9: Tempo do Processo de Testes, Com e Sem a Atividade Implementar Testes - engenheiros, *trainees* e estagiários

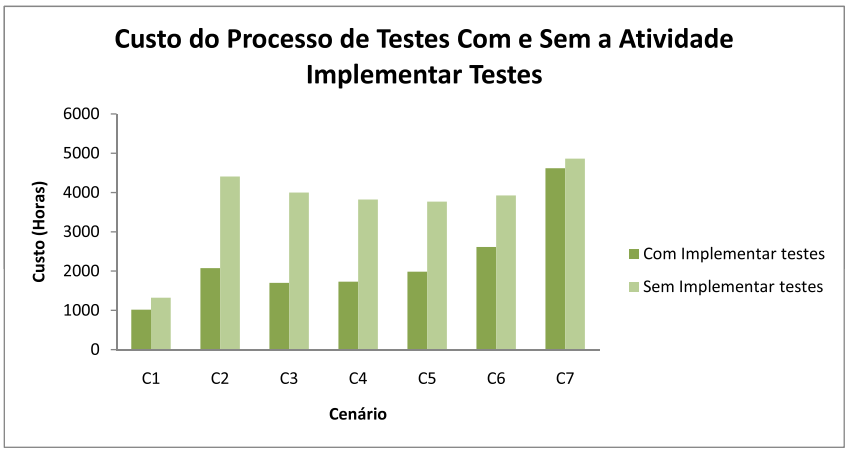


Figura 6.10: Custo do Processo de Testes Com e Sem a Atividade Implementar Testes - engenheiros, *trainees* e estagiários

6.6 ESTUDO DE CASO: QUANTIDADE DE DEFEITOS ESCAPADOS

Defeito escapado é a definição para uma falha do software não encontrado pela equipe de teste e, sim, pelo cliente. Quanto mais cedo encontra-se um defeito no ciclo de vida de um software, mais barato é o custo da sua correção. De acordo com Myers, em [Mye04], corrigir um defeito que se propagou até o ambiente de produção, pode chegar a ser cem vezes mais caro, do que corrigir este mesmo defeito em fases iniciais do projeto.

O objetivo de uma análise dos *defeitos escapados* é garantir melhoria contínua tanto do software entregue ao cliente, quanto do processo de testes de software. Esse estudo tem como objetivo realizar uma estimativa dos números de defeitos escapados do processo de testes, comparando número de defeitos encontrados, quando o processo é executado por engenheiros ou por *trainees* ou pelos estagiários.

A avaliação desse estudo, verifica a quantidade de defeitos escapados que o cliente pode encontrar ao longo de seis meses do projeto. Verifica-se através do modelo representado na Figura 6.11, em que cada bloco (ver Figura 6.4) representa as atividades do modelo de desempenho. De acordo com o gerente da instituição, é possível definir percentuais de defeitos escapados pelo nível de qualificação da equipe de testes que executa as atividades. A taxa percentual de defeitos escapados nas funcionalidades liberada para o cliente é de 20% para os engenheiros; 40% para os *trainee* e 50% para os estagiários. Com base nessa informação, foram adicionados na finalização do modelo proposto, duas transições imediatas: *NOk* e *Ok* que representam a taxa de defeitos escapados, encontrados no produto liberado e a taxa de aceitação do *release*, respectivamente. A expressão 6.5 serve para computar o número de defeitos escapados, em que P{#P6>0} representa a probabilidade, *W1* representa o peso da transição *NOk*, *W*2 representa o peso da transição *Ok* e Δt representa o período de tempo.

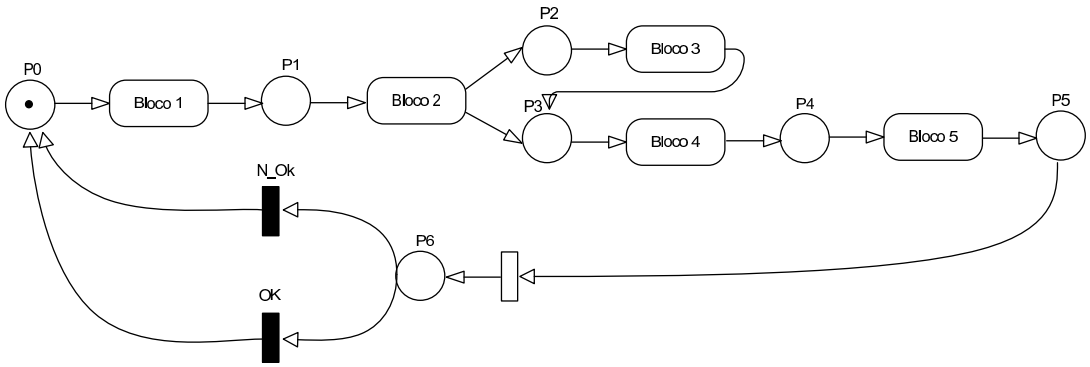


Figura 6.11: Modelo Qualidade do Processo de Testes.

$$Defeitosescapados = [(P\{\#P6 > 0\} * W1)/(W1 + W2)] * \Delta t \tag{6.5}$$

Foi verificada a execução do processo de testes considerando os seguintes cenários: apenas engenheiros, apenas *trainees* e apenas estagiários executando as atividades. Foi considerada a quantidade de casos de testes padrão, o intervalo de 30 dias para que o cliente avaliasse o produto e um grupo de sete colaboradores executores das atividades de testes.

A Tabela 6.9 mostra uma estimativa de quantidade de defeitos escapados. A estimativa de defeitos, mostrada na Tabela 6.9 foi verificado para cada nível de profissional que estava envolvido no processo.

A Figura 6.12 representa de forma gráfica a quantidade de defeitos escapados encontrados por cada perfil de colaborador no período de seis meses. Já na Figura 6.13 é representado o acumulado de defeitos encontrados durante todo o período verificado. Segundo esses gráficos, observa-se que a qualidade dos testes executados pelo grupo de colaboradores de nível engenheiro é bem superior ao dos outros. Mostrando a competência que esses profissionais possuem, mas ainda assim é preciso tomar um certo cuidado, pois houve uma variação do número de defeitos.

Tabela 6.9: Quantidade de Defeitos Escapados Por Nível de Colaboradores Envolvidos

Período	Engenheiro	Trainee	Estagiário
$Mês1$	2	3	5
$Mês2$	1	4	5
$Mês3$	2	3	5
$Mês4$	2	4	5
$Mês5$	2	3	6
$Mês6$	1	3	5

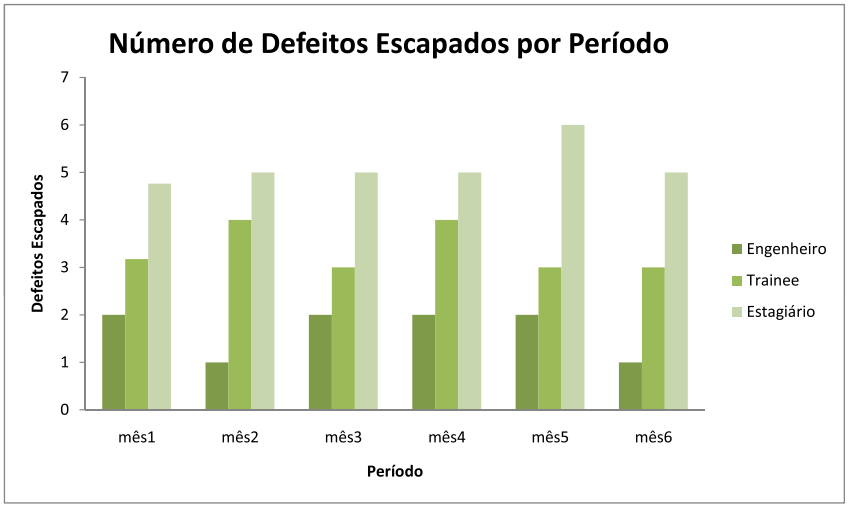


Figura 6.12: Número Defeitos Escapados.

O tempo e o custo da atividade do processo de testes de cada perfil de profissional, pode ser observado na Figura 6.14. A Figura 6.14 mostra que o tempo e o custo com as atividades do processo de testes com o engenheiro é bem menor que os outros perfis e que, apesar de os estagiários encontrarem um número maior de defeitos escapados e o tempo de execução das atividades um pouco maior, o custo das atividades de testes é menor que as dos *trainees*.

6.7 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os resultados obtidos na realização dos estudos de caso (avaliação de desempenho de processos de testes de softwares). Neste capítulo, cinco estudos foram realizados. Inicialmente, um estudo considerou diferentes alocações de engenheiros com números diferentes nas atividades do processo de testes. O objetivo desse foi utilizar o modelo proposto para obter tempo e custo gastos nas atividades do processo de testes, alocando números de engenheiros diferentes nas atividades do processo e ajudando a obter estimativas mais reais. Em seguida, foi avaliada a alocação de profissionais com dois níveis diferentes de qualificações(engenheiro e *trainee*). Também, no terceiro estudo, o custo e

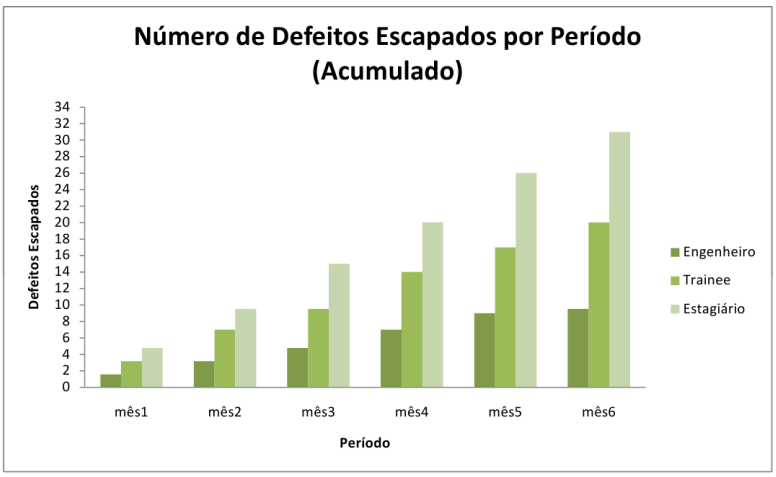


Figura 6.13: Número de Defeitos Escapados Acumulados.

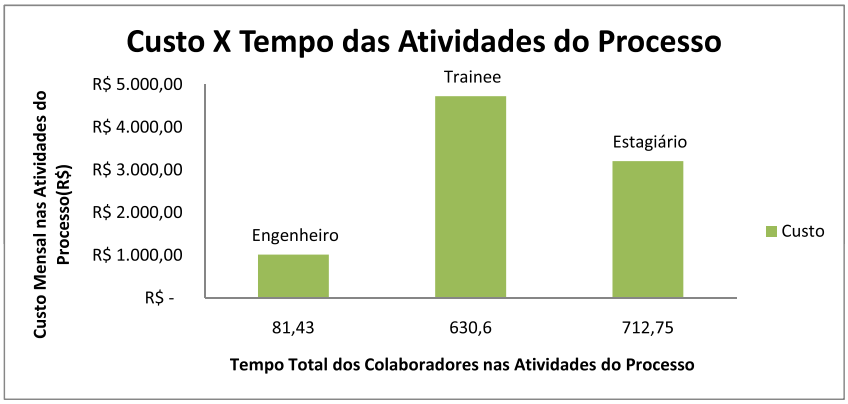


Figura 6.14: Custo X Tempo das atividades do processo.

o tempo foram avaliados com três níveis de qualificações diferentes(engenheiro, *trainee* e estagiário). Um outro estudo avaliado, foi o da necessidade de executar a atividade de implementação de testes automáticos, verificando se haveria ganho de tempo e custo no processo. Finalmente, um estudo que tinha como objetivo simular o número de defeitos escapados do processo de testes, verificando a quantidade de defeitos produzidos pelos diferentes níveis de colaboradores da instituição.

CAPÍTULO 7

CONCLUSÃO

O presente trabalho propôs uma metodologia de avaliação de desempenho de processos de testes software. Com o uso da metodologia, é possível elaborar modelos formais para auxiliar os gerentes nas análises de medidas de desempenho do processo de testes. Foram apresentados alguns mapeamentos dos diagramas de atividades da UML para os modelos SPN. Apresentaram-se uns modelos para alocação de recursos. Um modelo de avaliação de desempenho de processo de testes foi elaborado para verificação de medidas de desempenho de um processo de testes adotado por uma empresa. A metodologia e o modelo proposto foi validado com 95% de grau de confiança, em que os resultados não evidenciaram nenhuma discrepância entre os dados medidos e os valores obtidos do modelo.

Cinco estudos foram apresentados, os três primeiros voltados ao apoio de previsão de melhores estimativas de custo e tempo das atividades do processo de testes de software. No quarto, realizou-se uma comparação do processo de testes com e sem a atividade de automatizar testes; no quinto cenário, verifica-se a qualidade do processo de testes executado por colaboradores de diferentes qualificações. Os estudos realizados servem para apoiar os gerentes nas previsões de estimativas mais exatas de tempo, custo, quantidade de defeitos escapados no processo de testes de software. Com uma estimativa mais exata, o software a ser testado irá passar por todas as etapas do ciclo de testes, garantindo, assim, a qualidade do produto a ser liberado no tempo determinado.

A metodologia proposta, proporciona a avaliação do desempenho dos processos de testes e possibilitam estimativa do uso de recursos, apoiando para melhoria dos índices de qualidade e produtividade das organizações. Os modelos de execução de processo voltados para estimativa de desempenho, levam em consideração combinações de cenários diversos trazendo ganhos substanciais de produtividade, tanto na customização dos processos, quanto na efetivação do processo definido para o projeto.

Este trabalho proporciona a avaliação do desempenho dos processos de testes e possibilita previsões de estimativas do uso de recursos, apoiando para a melhoria dos índices de qualidade e produtividade das organizações. A dissertação apresentou modelos de execução de processo voltados para estimativa de desempenho que levam em consideração combinações de cenários diversos, trazendo ganhos substanciais de produtividade tanto na customização dos processos, quanto na efetividade do processo definido para o projeto.

Nesta dissertação, o Capítulo 5 apresentou os mapeamentos dos diagramas de atividades da UML para modelos SPN. Em seguida, evidenciam-se os modelos para alocação de recursos. Finalmente, elaborou-se um modelo de avaliação de desempenho de processo de testes para verificação de medidas de desempenho de um processo de testes adotado

por uma empresa, como também a validação do modelo proposto.

Os estudos de caso com objetivo de avaliar situações de interesse prático, encontram-se no Capítulo 6. Foram realizados cinco estudos com base no processo de testes de software, adotado por um Instituto de software do Recife.

O primeiro estudo verificou as configurações de equipe com profissionais de nível igual e com diferentes composições, nas cinco atividades do processo de testes. No segundo, foi verificado o tempo e o custo do processo, executado por profissionais com níveis de experiência diferentes e, por conseqüência, valor de salários distintos. Os níveis mostrados foram do engenheiro(um cargo mais senior) e do trainee (um cargo junior). O terceiro estudo verificou as configurações de equipes com três níveis de qualificações diferentes. A partir das composições dos cenários, foram apresentadas as medidas de desempenho.

O quarto estudo tratou o caso de avaliação de desempenho, comparado o custo e o tempo do processo de testes com e sem a atividade implementar testes. Foram verificadas algumas composições e analisado que, para o processo de testes adotado pela empresa na qual se aplicou o estudo de caso, a atividade de automatização de testes com colaboradores de diferentes categorias é válida, pois reduz o custo e o tempo do processo de teste.

O quinto estudo verificou questões de qualidade do processo de testes utilizado na organização. Através do modelo de desempenho, foi realizado um comparativo de tempo, custo e número de defeitos encontrados por categoria de profissionais envolvidos nas atividades de testes.

Para realização deste trabalho, algumas atividades foram realizadas

- estudo-da-arte sobre avaliação de desempenho, testes de software, planejamento de projetos, redes de petri;
- criação de uma metodologia para avaliação de desempenho dos processos de testes;
- escolha de um processo para ser utilizado como base para aplicação e validação da metodologia e dos modelos propostos;
- criação de modelos formais para extrair medidas de desempenho;
- entrevistas e coletas de dados históricos com gerentes de projetos e líderes de testes;
- estudos de casos com intuito de avaliar situações de interesse prático.

As contribuições deste trabalho são as seguintes:

- a criação de uma metodologia para auxiliar o processo de avaliação de desempenho das especificações dos processos de testes de software. Essa metodologia compõe-se de uma série de etapas que envolvem desde a compreensão do problema e entendimento do processo de testes, até a geração e análise dos modelos SPN. Com a aplicação da metodologia proposta, é possível avaliar o impacto de mudanças no

processo, avaliar o desempenho do processo de testes, realizar simulações com o objetivo de obter estimativas mais precisas e ajudar na garantia da qualidade do produto.

- o mapeamento dos diagramas de atividades da UML para modelos SPN, além disso, o desenvolvimento de modelos SPN, levando em consideração a quantidade e a qualificação dos membros da equipe do projeto para, a partir de composições de diferentes pessoas, verificar estimativas de tempo, custo e qualidade do processo de testes;

- um modelo estocástico expolinomial para avaliação de desempenho. O modelo de desempenho dos processos de testes permite realizar análises do processo de testes de software, proporcionando a avaliação do desempenho dos processos, que possibilitem estimativa do uso de recursos, um mecanismo que concorre para melhoria dos índices de qualidade e produtividade das organizações, além da identificação de processos adequados ao projeto de maneira que sejam efetivos com relação à qualidade do produto e, ao mesmo tempo, tenham execução eficiente;

- a realização de análises/verificações na etapa do planejamento das atividades de testes com o intuito de realizar estimativas mais precisas, estimando o tempo e o custo para todo o processo baseado nos diferentes colaboradores envolvidos no projeto;

- a integração dos modelos formais e semiformais. Essa integração permite uma especificação completa, rigorosa (sem ambiguidades nem inconsistências) e verificável às atividades do processo de testes. Os modelos formais são importantes porque a análise detalhada e o rigor necessário à sua construção permitem prevenir e detectar avaliações de desempenho mais cedo e, consequentemente, a prevenção de erros de estimativas. Contudo os modelos formais não são intuitivos e requerem um considerável esforço por parte dos projetistas para entenderem à notação usada. Por outro lado, as anotações dos modelos semiformais são amigáveis e intuitivas. Assim, é importante adotar o uso colaborativo dos modelos semiformais e formais a fim de encontrar erros ou inconsistências.

Durante a elaboração deste trabalho, foram encontradas algumas dificuldades que tornaram o processo mais desafiador. Dentre elas:

- obtenção dos dados do processo de teste, dos dados históricos do processo, como também das autorizações de acesso nas diferentes ferramentas, para coleta dos dados pela empresa em que os estudos foram adotados;

- participação dos gerentes e líderes de testes nas entrevistas realizadas;

- processo de validação e aproximação de fases, este não é um processo trivial, exige, portanto, um bom tempo do trabalho para alcançar um resultado satisfatório.

7.1 CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

O presente trabalho auxilia na identificação de processos de testes adequados ao projeto de maneira que sejam efetivos com relação à qualidade do produto e, ao mesmo tempo, tenham execução eficiente na estimativa de tempo, custo e na verificação da qualidade do processo de testes adotado pelas empresas.

Como trabalho futuro, pode-se incluir a implementação de uma ferramenta para a geração automática dos modelos SPN, a partir dos diagramas comportamentais da UML. Essa ferramenta poderá contribuir em diversos aspectos. O tempo do processo de mapeamento poderá ser reduzido, uma vez que o modelo SPN será gerado automaticamente. Devido a esse processo automático, também poderá ser garantida a não ocorrência de falhas no processo de conversão do modelo de alto-nível para o domínio das SPN. Essa ferramenta também permitirá uma completa abstração com relação à utilização das SPN. Além disso, as estimativas de tempo de execução, de custo do processo e da qualidade serão realizadas diretamente na ferramenta.

Este trabalho também poderá ser estendido para cobrir outras medidas de desempenho do processo de testes, como também a possibilidade de verificar os resultados esperados da execução dos testes, seja automáticos, seja manuais a fim de encontrar informações significantes. Adicionalmente, pode-se incluir, também, a realização de outros estudos de caso.

REFERÊNCIAS BIBLIOGRÁFICAS

[ASRM07] R.D. Arteiro, F.N. Souza, N.S. Rosa, and P.R.M. Maciel. *Utilizando Redes de Petri para Modelagem de Desempenho de Middleware Orientado a Mensagem. WPerformance*, pages 1–21, 2007.

[Bac04] J. Bach. What is exploratory testing. *E. van Veenendaal, The Testing Practitioner–2nd edition, UTN Publishing, ISBN*, pages 90–72194, 2004.

[Bal01] G. Balbo. *Introduction to Stochastic Petri Nets. Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, Berg en Dal, The Netherlands, July 3-7, 2000: Revised Lectures*, 2001.

[BDM02] S. Bernardi, S. Donatelli, and J. Merseguer. From UML sequence diagrams and statecharts to analysable petri net models. *Proceedings of the 3rd international workshop on Software and performance*, 2002.

[Bei02] B. Beizer. *Software testing techniques*. Dreamtech Press, 2002.

[BGdMT06] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Wiley-Interscience, 2006.

[BMM02] A. Bertolino, E. Marchetti, and R. Mirandola. Real-time UML-based performance engineering to aid manager's decisions in multi-project planning. In *Proceedings of the 3rd international workshop on Software and performance*, pages 251–261. ACM, 2002.

[BMV+00] E.F. Barbosa, J.C. Maldonado, A.M.R. Vincenzi, M.E. Delamaro, SRS Souza, and M. Jino. Introdução ao Teste de Software. *XIV Simpósio Brasileiro de Engenharia de Software. Anais... João Pessoa/PB*, 2000.

[BRCM06] A. Bastos, E. Rios, R. Cristalli, and T. Moreira. Base de conhecimento em teste de software. *Traço & Photo*, 2006.

[Bur03] I. Burnstein. *Practical software testing: a process-oriented approach*. Springer Verlag, 2003.

[Cam03] M.S. Campos. Desvendando o Minitab. Qualitymark, 2003.

[CKS] MB Crisis, M. Konrad, and S. Shrum. CMMI® Second Edition. Guidelines for Process Integration and Product Improvement.

[CL08] C. Cassandras and S. Lafortune. Introduction to Discrete Event Systems. Springer, 2008.

[CSB+04] A.N. Crespo, O.J. Silva, C.A. Borges, C.F. Salviano, M.T. ARGOLLO Jr, and M. Jino. Uma metodologia para teste de Software no Contexto da Melhoria de Processo. *Simpósio Brasileiro de Qualidade de Software*, 2004.

[DAJ95] A.A. Desrochers and R.Y. Al-Jaar. Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis. IEEE Press, 1995.

[DdSS] L.M. Döll, J.U.F. de Souza, and P.C. Stadzisz. Verificação e Validação de Sistemas Orientados a Objetos Usando Redes de Petri.

[DH97] A.C. Davison and DV Hinkley. *Bootstrap methods and their application*. Cambridge Univ Pr, 1997.

[DMJ07] M.E. Delamaro, J.C. Maldonado, and M. Jino. Introdução ao teste de software. *Rio de Janeiro, RJ: Editora Campus*, 2007.

[dO] C.A.L. de Oliveira. Uma Abordagem para Melhoria de Workflow Baseada em Redes de Petri Estocásticas Generalizadas.

[dSLJ+06] A.N. da Silva, F.A.A. Lins, J.C.S. Júnior, N.S. Rosa, N.C. Quental, and P.R.M. Maciel. Avaliação de Desempenho da Composição de Web Services Usando Redes de Petri. Brazilian Symposium on Computer Networks. Curitiba, Paraná, Brazil, 2006.

[dVRMdM06] A.M.L. de Vasconcelos, A.C. Rouiller, C.Â.F. Machado, and T.M.M. de Medeiros. *Introdução à Engenharia de Software e à Qualidade de Software*. Lavras: UFLA/FAEPE, 2006.

[ESU97] T. Ericson, A. Subotic, and S. Ursing. TIM - A Test Improvement Model. *Software Testing Verification and Reliability*, 7(4):229–246, 1997.

[FBK+91] S. Fujiwara, G. Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test selection based on finite state models. *IEEE Transactions on software engineering*, 17(6):591–603, 1991.

[GAS05] V. Gupta, KK Aggarwal, and Y. Singh. Objectively Managing Software Testing Projects Using Software Test Metrics. *Journal of Conceptual Modeling*, 34, 2005.

[Ger00] R. German. Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets. John Wiley & Sons, Inc. New York, NY, USA, 2000.

[GLR+03] V. Gupta, V. Lam, H.G.V. Ramasamy, W.H. Sanders, and S. Singh. *Dependability and Performance Evaluation of Intrusion-tolerant Server Architectures*. *Lecture Notes in Computer Science*, pages 81–101, 2003.

[Gra91] WK Grassmann. *Finding Transient Solutions in Markovian Event Systems Through Randomization*. *Numerical Solution of Markov chains*, pages 357–371, 1991.

[Gro08] D. Gross. Fundamentals of Queueing Theory. Wiley India Pvt. Ltd., 2008.

[Gue08] GTA Guedes. UML: Uma abordagem prática, 3a Edição, 2008.

[GVVEB08] D. Graham, E. Van Veenendaal, I. Evans, and R. Black. *Foundations of Software Testing: ISTQB Certification*. Thomson Learning Emea, 2008.

[Her01] U. Herzog. *Formal Methods for Performance Evaluation*. *Lecture Notes in Computer Science*, 2090:1–37, 2001.

[HMRT01] B.R. Haverkort, R. Marie, G. Rubino, and K.S. Trivedi. Performability Modelling: Techniques and Tools. John Wiley & Sons Inc, 2001.

[Ins08] Project Management Institute. *A Guide to the Project Management Body of Knowledge:(PMBOK Guide)*. Project Management Institute, Inc., 2008.

[Jai91] R. Jain. The Art of Computer Systems Performance Analysis. John Wiley & Sons New York, 1991.

[JKCPR00] VA Jeetendra, OV Krishnaiah Chetty, and J. Prashanth Reddy. Petri nets for project management and resource levelling. *The International Journal of Advanced Manufacturing Technology*, 16(7):516–520, 2000.

[JM90] F. Jay and R. Mayer. IEEE standard glossary of software engineering terminology. *IEEE Std*, 610:1990, 1990.

[Jon98] C. Jones. Software Project Management in the 21st Century. *American Programmer*, 11:24–30, 1998.

[KD82] H. Koontz and C. DONNELL. Princípios de administração: uma análise das funções administrativas. *São Paulo: Pioneira*, 1982.

[Ker09] H. Kerzner. *Project management: a systems approach to planning, scheduling, and controlling*. Wiley, 2009.

[KFN99] C. Kaner, J.L. Falk, and H.Q. Nguyen. *Testing computer software*. John Wiley & Sons, Inc. New York, NY, USA, 1999.

[KKC06] S. Kumanan and OV Krishnaiah Chetty. Estimating product development cycle time using Petri nets. *The International Journal of Advanced Manufacturing Technology*, 28(1):215–220, 2006.

[KN97] R.S. Kaplan and D.P. Norton. Estratégia em ação: balanced scorecard. 1997.

[KP] P. King and R. Pooley. Using UML to derive stochastic Petri net models. In *Proceedings of the 15th UK Performance Engineering Workshop*, pages 45–56. Citeseer.

[Lar08] C. Larman. *Utilizando UML e padrões*. Livraria Tempo Real Inform, 2008.

[LGMC04] J.P. López-Grao, J. Merseguer, and J. Campos. From UML activity diagrams to Stochastic Petri nets: application to software performance engineering. In *Proceedings of the 4th international workshop on Software and performance*, page 36. ACM, 2004.

[Lil05] D.J. Lilja. *Measuring computer performance: a practitioner's guide*. Cambridge Univ Pr, 2005.

[MA05] D.A. Menascé and V.A.F. Almeida. Performance by Design: Computer Capacity Planning by Example. Prentice Hall PTR, 2005.

[MAFM00] D.A. Menascé, V.A.F. Almeida, R. Fonseca, and M.A. Mendes. *Business-oriented Resource Management Ppolicies for E-commerce Servers*. *Performance Evaluation*, 42(2-3):223–239, 2000.

[Mar89] M.A. Marsan. Stochastic Petri Nets: An Elementary Introduction. *Advances in Petri Nets*, 424:1–29, 1989.

[MBC+98] M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets. ACM SIGMETRICS Performance Evaluation Review*, 26(2), 1998.

[MBV+04] J.C. Maldonado, E.F. Barbosa, A.M.R. Vincenzi, M.E. Delamaro, C.U.E. de Marilia, S.R.S. de Souza, and M. Jino. Introduçao ao teste de software. *Instituto de Ciências Matemáticas e de Computacão–ICMC-USP, San Carlos, Brasil, Versión*, 1, 2004.

[MCBD02] J. Merseguer, J. Campos, S. Bernardi, and S. Donatelli. A compositional semantics for UML state machines aimed at performance evaluation. *Discrete Event Systems, 2002. Proceedings. Sixth International Workshop on*, pages 295–302, 2002.

[MCM02] J. Merseguer, J. Campos, and E. Mena. Performance evaluation for the design of agent-based systems: A Petri net approach. *21st International Conference on Application and Theory of Petri Nets*, pages 1–20, 2002.

[MFR03] T.R. MOREIRA FILHO and E. RIOS. Projeto & engenharia de software: teste de software, 2003.

[MLC96] P.R.M. Maciel, R.D. Lins, and P.R.F. Cunha. *Introduction of the Petri Net and Applied. X Escola de Computação, Campinas, SP*, 1996.

[MMS+10a] M. Marinho, P. Maciel, E. Sousa, T. Maciel, and E. Andrade. Performance Evaluation Model For Test Process. *11th IEEE Latin American Test Workshop (LATW'10),Punta Del Este*, 2010.

[MMS+10b] M. Marinho, P. Maciel, E. Sousa, T. Maciel, and E. Andrade. Performance Evaluation of Test Process Based on Stochastic Models. *Symposium On Theory of Modeling and Simulation - DEVS Integrative M&S Symposium (DEVS'10), part of 2010 Spring Simulation Multiconference (SpringSim'10), Orlando*, 2010.

[MMSM10] M. Marinho, P. Maciel, E. Sousa, and T. Maciel. Stochastic Model for Performace Evaluation of Test Planning. *IEEE International Conference on Systems, Man, and Cybernetics, Turquia*, 2010.

[MR03] D.C. Montgomery and G.C. Runger. Estatística Aplicada e Probabilidade para Engenheiros. Livros Técnicos e Científicos, 2003.

[Mur89] T. Murata. *Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE*, 77(4):541–580, 1989.

[Mye04] G.J. Myers. *The art of software testing.* Wiley, 2004.

[Nób08] R.O. Nóbrega. Balanced testing scorecard: um modelo paraavaliação e melhoria de desempenho de equipes detestes de software. 2008.

[OLR09] C. Oliveira, R. Lima, and B. Recife. Performance analysis of resourcecontrained business processes: A formal approach based on stochastic petri nets. 2009.

[Pat05] R. Patton. *Software testing.* Sams Indianapolis, IN, USA, 2005.

[Pen03] T. Pender. *UML Bible.* John Wiley & Sons, Inc. New York, NY, USA, 2003.

[Per06] W. Perry. Effective methods for software testing. 2006.

[PMM02] R.S. Pressman and G. Mônica Maria. *Engenharia de software.* McGraw-Hill, 2002.

[PSST08] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. *CPU Demand for Web Serving: Measurement Analysis and Dynamic Estimation. Performance Evaluation*, 65(6-7):531–553, 2008.

[Res02] Ê. Resende. *Cargos, salários e carreira: novos paradigmas conceituais e práticos.* Grupo Editorial Summus, 2002.

[RK07] K. Raja and S. Kumanan. Resource Leveling Using Petrinet and Memetic Approach. *American Journal of Applied Sciences*, 4(5):317–322, 2007.

[rup10] Rational Unified Process. Disponível em. http://www.wthreex.com/rup, Acessado em: 02/04/2010.

[SARM06] F.N. Souza, R.D. Arteiro, N.S. Rosa, and P.R.M. Maciel. *Using Stochastic Petri Nets for Performance Modelling of Application Servers*. *Performance Modelling, Evaluation, and Optimisation of Parallel and Distributed Systems*, pages 1–8, 2006.

[Smi93] C. Smith. Software performance engineering. *Performance Evaluation of Computer and Communication Systems*, pages 509–536, 1993.

[Som95] I. Sommerville. *Software Engineering*. Addison Wesley, 1995.

[SSR00a] A. Schmietendorf, A. Scholz, and C. Rautenstrauch. Evaluating the performance engineering process. In *Proceedings of the 2nd international workshop on Software and performance*, pages 89–95. ACM, 2000.

[SSR00b] Andreas Schmietendorf, André Scholz, and Claus Rautenstrauch. Evaluating the performance engineering process. In *WOSP '00: Proceedings of the 2nd international workshop on Software and performance*, pages 89–95, New York, NY, USA, 2000. ACM.

[Tia05] J. Tian. *Software quality engineering: testing, quality assurance, and quantifiable improvement*. Wiley-IEEE Computer Society Pr, 2005.

[UML10] OMG UML. 2.0 Superstructure Specification. http://www.uml.org/, Acessado em: 08/04/2010.

[Val09] L.D.N. Vale. Especificação de Testes Funcionais usando Redes de Petri a Objetos para Software Orientados a Objetos. 2009.

[Van03] MA Vandermark. Defect Escape Analysis: Test Process Improvement, 2003.

[Var] R. Vargas. *Manual prático do plano de projeto: utilizando o PMBOK Guide*. Brasport.

[Vie02] E. Vieira. Gerenciando Projetos na Era de Grandes Mudanças-Uma breve abordagem do panorama atual. *PMI Journal–PMI-RS*, 3:7–16, 2002.

[VK94] M.R. Vigder and A.W. Kark. Software cost estimation and control. *Institute for Information Technology, National Research Council Canada, Ottowa, Ontario, Canada, NRC*, 37116:21, 1994.

[ZK07] A. Zimmermann and M. Knoke. A Software Tool the Performability Evaluation with Stochastic and Colored Petri Nets. Technische Universitt Berlin. Real-Time Systems and Robotics Group, 2007.

APÊNDICE A

APÊNDICE A

A.1 SINTAXE DAS EXPRESSÕES SUPORTADAS PELO TIMENET 4.0

O apêndice apresenta a sintaxe das expressões suportada pelo TimeNet 4.0 [ZK07] para especificação das métricas.

A.1.1 Símbolos Usados

"symbol" = símbolo terminal

<symbol> = não - símbolo terminal

expr1 | expr2 = Expressão 1 ou Expressão 2

Expression = Uma expressão ou ocorrências

[Expression] = Uma expressão opcional

<md exp delay> = marcação dependente de uma transição exponencial

<md det delay> = marcação dependente de uma transição determinística

<md weight> = disparo de peso dependente de marcação de uma transição imediata

<md enable> = guarda de uma transição imediata marcação-dependente ou dependente de uma marcação

<md arc mult> = multiplicidade dependente da marcação de uma entrada, saída ou arco do inibidor

<reward def> = Definição da medida de recompensa de um parâmetro de atraso dependente

<param def> = Definição de parâmetros de atrasos dependentes

<pmf def> = Atraso de disparo de uma transição generalizada

A.1.2 Definições da Sintaxe

<md exp delay> : *<if expr>*

<md det delay> : *<if expr>*

<md weight> : *<if expr>*

<md enable> : *<logic condition>";"*

<md arc mult> : *<if expr>*

<reward def> : *<expression>";"*

<param def> : *<expression>";"*

<pmf def> : *<pmf defnition>";"*

<pmf definition> : *"DETERMINISTIC("<real constant>)";"*
| *"UNIFORM("<real constant> ","<real constant> ");"*
| *<pmf expression> ";"*

<pmf definition> : *"<pmf expression>"+"<pmf expression>*
| *<pmf expression> "-"<pmf expression>*
| *<impulse>*
| *<rectangule>*

<impulse> : *[<real constant>["*"] "I["<real constant>"]"*

<if expr> : *"IF"<logic condition> ";"<expression>*
"ELSE"<expression> ";"| <expression> ";"

<expression> : *<real value>*

| *"-"<expression>*
| *"("<expression> ")"*
| *<expression> <num op> <expression>*

<real value> : *<real parameter>*
| *<real constant>*
| *"("<expression> ")"*
| *<expression> <num op> <expression>*

<real parameter> : *<identifier>*

<rew item> : *"P {"<logic condition> "}"*
| *"P"{"<logic condition> "IF"<logic condition>*
|*' 'E {"<marc func> "}"logic condition* |*' 'E {"<marc func> "IF"<logic condition> "}"*

<logic condition> : *<comparison>*
| *"NOT"<logic condition>*
| *<logic condition> "OR" <logic condition>*
| *<logic condition> "AND" <logic condition>*

<comparison> : *<mark func> <comp oper> <mark func>*

<comp oper> : *"="| "/ ="| ">"| "<"| ">="| "<="*

<mark func> : *<mark func> <num op> <mark func>* | *"("<mark func> ")"*| *<integer value>*

<num op> : *"+"| "−"| "∗"| "/"| "?"*

<integer value> : *<integer constant>* | *<integer parameter>* | *<marking>*

<integer constant> : *<digit>* { *<digit>* }

<integer parameter> : *<identifier>*

<marking> : *"♯"<place name>*

<letter> : *"a"| "..."| "z"| "A"| "..."| "Z"*

<digit> : "0"| "1"| "2"| "3"| "4"| "5"| "6"| "7"| "8"| "9"| "10"

A.1.3 Expressões especiais para definições de métricas

"P"{ <logic condition> "}"= Probabilidade de <logic condition>

"P"{ <logic condition1> "IF" <logic condition2> "}"= Probabilidade de <logic condition1> como pré-condução <logic condition2> (Probabilidade Condicional)

"E"{ <marc func> }"= Valor esperado da expressão dependente da marcação <marc func>

"E"{ <marc func> "IF" <logic condition2> "}"= Valor esperado da expressão dependente da marcação <marc func> ; as marcações são consideradas se <logic condition2> for avaliada como verdadeira

Este volume foi tipografado em LaTeX na classe UFPEThesis (www.cin.ufpe.br/~paguso/ufpethesis).