PUCRS

FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO
CIÊNCIAS DE COMPUTAÇÃO

ILDO INÊS ROBÃO MASSITELA

APLICAÇÃO DO FORMALISMO SAN PARA AVALIAÇÃO DE DESEMPENHO DE UMAEQUIPE DEDESENVOLVIMENTO DE SOFTWARE BASEADA NO MODELO WATERFALL

Porto Alegre
2016

PÓS-GRADUAÇÃO - *STRICTO SENSU*

Pontifícia Universidade Católica
do Rio Grande do Sul

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APLICAÇÃO DO FORMALISMO SAN PARA AVALIAÇÃ DE DESEMPENHO DE UMA EQUIPE DE DESENVOLVIMENTO DE SOFTWARE BASEADA NO MODELO WATERFALL

ILDO INÊS ROBÃO MASSITELA

Dissertação apresentada como requisito à obtenção do grau de Mestre em Ciência da Computação na Pontifícia Universidade Católica do Rio Grande do Sul.

Orientador: Prof. Paulo Henrique Lemelle Fernandes

Porto Alegre
2016

Ficha Catalográfica

M418a Massitela, Ildo Inês Robão

Aplicação do Formalismo SAN para Avalição de Desempenho de uma Equipe de Desenvolvimento de Software baseada no Modelo Waterfall / Ildo Inês Robão Massitela . – 2016.

76 f.

Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Paulo Henrique Lemelle Fernandes.

1. Modelagem Estocástica, Modelo Waterfall. I. Fernandes, Paulo Henrique Lemelle. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS com os dados fornecidos pelo(a) autor(a).

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada "*Aplicação do Formalismo SAN para Avaliação de Desempenho de uma Equipe de Desenvolvimento de Software Baseada no Modelo WATERFALL*" apresentada por Ildo Inês Robão Massitela como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, aprovada em 23 de março de 2016 pela Comissão Examinadora:

Prof. Dr. Paulo Henrique Lemelle Fernandes– Orientador — PPGCC/PUCRS

Prof. Dr. Rafael Prikladnicki – — PPGCC/PUCRS

Prof. Dr. Afonso Henrique Corrêa de Sales - — FACIN/PUCRS

Homologada em 08/12/2016, conforme Ata No. 025 pela Comissão Coordenadora.

Prof. Dr. Luiz Gustavo Leão Fernandes
Coordenador.

DEDICATÓRIA

Dedico este trabalho a minha esposa Manuela da Gloria Emília Mondlhane,aos meus filhos Cyndee Wami, Ricky Keytan, Cristyan Sholani, aos meus pais e irmãos

“O sucesso é ir de fracasso em fracasso sem perder entusiasmo"
(Winston Churchill)
“O único lugar aonde o sucesso vem antes do trabalho é no dicionário"
(Albert Einstein)

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Paulo Fernandes por ter me aceita como orientando, pelo suporte, incentivo, suporte e acima de tudo pela paciência de esclarecimento nas minhas enumeras dificuldade. O meu muito obrigado aos Prof. Dr. Rafael Prikladnicki e Prof. Dr. Afonso Sales por aceitarem compor a minha banca de defesa de mestrado. Agradeço A CNPq por acreditar no meu projeto e me fornecer o suporte financeiro para o mestrado na PUCRS. Agradeço a todos os professores do PPGCC/PUCRS pelo apoio e pela paciência de dar suporte e respostas as minhas dificuldade. Agradeço aos colegas de mestrado Alan dos Santos, Marcelo Gomes, Antônio Leaes, Maurício, Ramon Pereira. Agradeço aos funcionários do PPGCC/PUCRS pela constante e pontual colaboração. O agradecimento muitíssimo especial a minha esposa Manuela e a minha filha Cyndee por suportarem a minha ausência neste período que estive fora e longe delas. Por me darem forças de caminhar até ao fim da jornada. Este trabalho é muito especialmente dedicado a elas.

APLICAÇÃO DO FORMALISMO SAN PARA AVALIAÇÃO DE DESEMPENHO DE UMA EQUIPE DE DESENVOLVIMENTO DE SOFTWARE BASEADA NO MODELO WATERFALL

RESUMO

O proliferação do uso de softwares no dia a dia de diversas instituições é um fato incontestável, quer seja instituições acadêmicas, industriais, comerciais, dentre outras. Esta grande procura de softwares cria um ambiente de grande pressão na academia e na indústria de software no que se refere a capacidade de resposta da grande demanda. A avaliação de desempenho pode ser um dos grandes benefícios para as instituições, na tomada de decisões que podem contribuir no aumento e melhoria da capacidade de resposta da crescente demanda. O presente trabalho tem como objetivo mostrar, com base num estudo de caso, a aplicação de modelagem estocástica para avaliação de desempenho de uma equipe de desenvolvimento de software que tem o modelo de processos Waterfall como base do projeto de software. Apesar de ser um modelo que vem sofrendo críticas, tem sido um recurso em vários projetos de software pela sua estrutura rígida e muito focada no gerenciamento. Alguns dos fatores que garantem um bom gerenciamento são as fases do projeto bem distintas, onde uma fase não se inicia sem que a outra tenha terminado e o foco para documentação que ajuda no histórico institucional ao longo do tempo.

Palavras Chave: Modelos de Desenvolvimento de Software, Waterfall, Modelagem Estocástica, Stochastic Automata Network, Avaliação de Sistemas.

FORMALISM SAN APPLICATION FOR PERFORMANCE EVALUATION OF A SOFTWARE DEVELOPMENT TEAM BASED IN WATERFALL MODEL

ABSTRACT

The proliferation of the use of software on a daily basis from various institutions is an indisputable fact, whether academic institutions, industrial, commercial, among others. This great looking software creates a great pressure environment in academia and software industry regarding the responsiveness of the great demand. The performance assessment can be a major benefit to the institutions, decision-making can contribute in increasing and improving the responsiveness of growing demand. This paper aims to show, based on a case study, the use of stochastic modeling for performance evaluation of a software development team that has the waterfall model based software design. Despite being a model that has been criticized, it has been a feature in various software projects for its rigid structure and focused management. Some of the factors that ensure a good management are very different design phases where one phase does not start without the other has finished and documentation that helps to focus the institutional history over time.

Keywords: Software Development Models, Waterfall, Stochastic Modeling, Stochastic Automata Network, Systems Evaluation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

SAN – Stocastic Automata Networks

GSD – Global Sofware Development

FTS – Follow The Sun

SDLS – Software Development Life Cycle

PEPS – Perfomance Evaluation pf Parallel Systems

PERHAPS – Perfomce Evaluation of 'rha' Parallel Systems

TPS – Transacoes por segundo

FPA – Function Point Analysis

IBM – International Business Machines

TI – Tecnologias de Informação

LISTA DE ABREVIATURAS

P&D. – Pesquisa e Desenvolvimento

SW. – Software

XP. – Extreme Programing

Tr. – Trabalhando

Pa. – Parado

Co. – Colaborando

Re. – Retrabalhando

AS. – Analista de Sistemas

Dev. – Desenvolvedor

AT. – Analisata de Testes

Tst. – Tester

T.. – Tempo

Prob. – Probabilidade

Bps. – Bits por segundo

TPS. – Transações por segundo

SUMÁRIO

1. INTRODUÇÃO

É claramente evidente que o uso de software está tomando conta de vários processos do dia-a-dia. É quase impossível imaginar o mundo moderno sem a existência de softwares para realização de algumas tarefas. O funcionamento pleno de várias infraestruturas e diversos serviços é garantido por um computador e um software. Daí que, incontornavelmente, o mercado de softwares tem vindo o conhecer um ambiente de muita procura o que conduz a uma necessidade de rápida resposta à procura de softwares. Para fazer face a este ambiente, as equipes de desenvolvimento de softwares tem investido esforço para garantir que os projetos sejam viáveis em termos de tempo de conclusão. Os Engenheiros de Software, que são os responsáveis por gerir todos os aspectos de produção de softwares, dentre vários aspectos, eles devem ter em conta questões práticas de custo, prazos do projeto e a garantia de confiabilidade do produto.

Para o projeto de desenvolvimento de software existem modelos de processos prescritivos e as ágeis. O presente trabalho foca para a avaliação de desempenho de uma equipe que utiliza o modelo de processos prescritivo Waterfall para o desenvolvimento de softwares que é um dos mais importantes modelos e é referência e ainda serve de base para muitos projetos de software modernos. O modelo de processo Waterfall, que também é conhecido por abordagem "top-down", foi proposto por Royce em 1970. Até meados da década de 1980 foi o único modelo com aceitação geral. Este modelo foi derivado de modelos de atividade de engenharia com o fim de estabelecer ordem no desenvolvimento de grandes produtos de software [20].

O modelo de processos Waterfall considera aspectos que tem a ver com especificação do projeto, desenvolvimento, validação e evolução. E todas essas fases são representadas como distintas. É um modelo muito apropriado para situações em que se tem um entendimento muito claro dos requisitos. Apesar de ser um modelo que oferece muita previsibilidade de prazos e custos existe a necessidade de aperfeiçoar ainda mais o nível de previsibilidade de tempo de execução de projetos nas distintas fases que caracterizam o modelo de processos Waterfall, de modo a garantir maior performance da equipe de desenvolvimento. Para análise do desempenho de uma equipe que usa o modelo Waterfall várias alternativas de modelos teóricos podem ser usados como por exemplo a análise por ponto de função (Function Point Anaysis – FPA) e Use Case Points [18].

São abordados neste trabalho alguns modelos ágeis mais usados de modo a elucidar e sustentar a escolha de modelo de processos Waterfall para este estudo, apesar de haver uma clara tendência de se optar pelos modelos processos ágeis principalmente o eXtreme Programing (XP) e Scrum pelo fato de se considerar o modelo Waterfall inflexível e inaplicável na prática. Segundo Beck [39], Extreme Programming (ou simplesmente XP) "é um modelo de processos ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante e rápida mudança". O Scrum é um modelo ágil que objetiva desenvolver e entregar o software com a maior qualidade possível dentro de séries, compostas por Sprints, intervalos de tempos definidos para o desenvolvimento [42].

O presente trabalho recorre ao modelo de Redes de Autômatos Estocásticos (Stochastic Automata Networks - SAN) para modelagem analítica de equipes de desenvolvimento de softwares. A equipe modelada neste trabalho usa o modelo de processos Waterfall para o desenvolvimento do projeto. O modelo SAN [1, 12, 13] é um poderoso formalismo de modelagem baseado em cadeias de Markov [14] que fornece uma descrição de alto nível (abstração) de um modelo. Trabalhos relacionados para a modelagem estocástica e simulações são desenvolvidos através da especificações dinâmicas do projeto de software , como uma modelagem analítica de uso para a análise da variabilidade de desempenho de equipes de desenvolvimento de software [11, 23, 47].

Novas metodologias são necessárias para avaliação e simulação de cenários que podem incrementar ou ilustrar a performance de projetos, principalmente a performance dos elementos participantes nas equipes de projetos de software. Este trabalho tem o objetivo de aplicar o formalismo SAN em projetos de software com base no modelo de processos Waterfall. Apesar de haver muitos questionamentos na aplicabilidade real de modelos de processos Waterfall, muitas organizações não abrem mão do seu uso pela sua forte orientação ao planejamento.

Para a execução do modelo SAN recorre-se ao software PEPS. O software PEPS (Performance Evaluation of Parallel Systems) é um pacote de ferramentas para modelar e resolver modelos expressos em Redes de Autômatos Estocásticos (SAN). O projeto PEPS teve o seu inicio nos finais dos anos 80 com vista a prover uma ferramenta de software para modelar e determinar soluções numéricas para o formalismo SAN [15]. Neste trabalho usa-se a versão PEPS2007.

1.1 Motivação

Atualmente verifica-se uma grande queda do custo de equipamentos de hardware, mas o custo de desenvolvimento de softwares não vem obedecendo esta mesma tendência. Muito pelo contrário, os custos de desenvolvimento de software vem, a cada dia que passa, correspondendo a maior parte do custo do desenvolvimento dos sistemas. O tempo de duração dos projetos de desenvolvimento de softwares complexos tem se caracterizado por ser longo e de difícil previsão. As vezes motivado pela complexidade do sistema mas, em outros casos, motivado pela previsão deficiente do tempo de culminação do projeto de software de diferentes áreas de Pesquisa e Desenvolvimento (P&D).

Apesar de a literatura dizer que o modelo de processos Waterfall oferece maior previsibilidade de tempo e custos na prática equipes que usam este modelo tem deparado com situações de atraso na entrega de projetos de software, estouro de orçamento, má alocação dos técnicos disponíveis para responder efetivamente o tempo de execução do projeto.

Com o uso do modelo SAN , os gestores poderão simular, de forma fácil e intuitiva, diversos cenários com diferentes configurações de equipes. Apesar da existência de diversas ferramentas e técnicas que apoiam o Gerente de Projetos de Softwares a monitorar e controlar os prazos de execução de projetos, a utilização de um modelo matemático é extremamente útil e poderá apoiar

os gestores em termos de mais recursos para a tomada de decisão refente a questões de previsão de tempo de execução de projetos e extração de índices de desempenho.

1.2 Objetivos

A presente dissertação tem como objetivo avaliar o desempenho de uma equipe de desenvolvimento de software que usa o modelo de processos Waterfall para o projeto de software. Para avaliar o desempenho de equipes é gerado manualmente o respetivo modelo SAN e é executado no software PEPS de modo a devolver índices de desempenho para análise. De modo a garantir o alcance do objetivo foi necessário: Fazer um levantamento bibliográfico sobre as SAN; Estudar com mais profundidade os modelos de desenvolvimento de software com maior ênfase para o modelo de processos Waterfall (etapas de execução, formação da equipe de execução do projeto, previsão do tempo de execução do projeto); Aprofundar os conhecimentos do software PEPS; Construir o modelo SAN referente à equipe de desenvolvimento de software com base nos dados de um estudo de caso previamente identificado; Executar o modelo no software PEPS; Interpretar os índices de desempenho devolvidos pelo PEPS.

2. TRABALHOS RELACIONADOS

2.1 Avaliação de Sistemas

Avaliar um sistema é pronunciar-se sobre as características de um certo sistema. Dado um sistema real qualquer, uma avaliação deste sistema pode ser definida como toda e qualquer observação feita sobre ele. Existem dois tipos de avaliação [18]:

- Avaliação qualitativa - quando existe a necessidade de uma comparação com o senso comum, ou ainda uma comparação com um referencial de base;
- Avaliação quantitativa - que se baseia na formulação de valores específicos, sem expressar considerações dos méritos dos valores obtidos.

Em princípio, toda avaliação tem por objetivo o estabelecimento de um julgamento qualitativo sobre o sistema avaliado. No entanto, toda avaliação científica é feita sobre resultados quantitativos. Tais resultados devem ser objetivamente apresentados ao usuário final, de forma que se possa tomar decisões acerca de seu sistema [18]. A aplicação prática da avaliação de desempenho é o conhecimento da situação (estado) do sistema avaliado. Tanto situações anteriores como situações atuais podem ser avaliadas para tornar possível a observação da evolução do sistema. Além disso, a observação do comportamento do sistema ajuda a entender o seu funcionamento. Podem ser ainda avaliadas situações futuras, com a finalidade de previsão e planejamento [18]. No contexto da avaliação de sistemas, salienta-se que é sempre recomendável um estudo da confiabilidade do método aplicado para um determinado processo de avaliação de desempenho de sistema; para levar a cabo este objetivo é frequente realizar-se a comparação de resultados de diversas técnicas aplicadas ou de uma mesma técnica em cenários diferentes.

2.1.1 Técnicas de Avaliação de Sistemas

A avaliação é tradicionalmente feita com base em três técnicas que são: a monitoração, a simulação e métodos analíticos [18].

2.1.1.1 Monitoração

A Monitoração consiste na observação (monitoração) de sistemas reais. Dentre as técnicas citadas, esta é a que propicia maior fidelidade dos índices obtidos, pois não é feita nenhuma abstração (modelagem) do sistema em questão. Todavia, há algumas desvantagens visíveis desta técnica de avaliação, como é o caso da necessidade clara da existência real do sistema a ser avaliado. Isto pode gerar problemas em relação ao custo e ao tempo, pois o sistema implementado pode não satisfazer

as necessidades, tendo que ser abandonado [18]. Uma outra desvantagem consiste na necessidade de ter que se prestar muita atenção na questão da amostragem. É necessário que se faça o uso correto de técnicas de estatística de amostragem para que os dados recolhidos tenham validade e sejam realmente representativos.

2.1.1.2 Simulação

A simulação é uma ferramenta poderosa, se entendida e usada corretamente. Simulação é "o processo de concepção de um modelo de um sistema real e a realização de experimentos com base no modelo concebido com a finalidade de se compreender o comportamento do sistema ou para avaliação de várias estratégias (dentro dos limites impostos por um critério ou conjunto de critérios) para o funcionamento do sistema ". O modelo deve descrever as características funcionais do sistema dentro de uma escala de tempo apropriada [31, 35]. O modelo obtido para efeitos de simulação contém apenas as informações relevantes que se pretende que sejam modeladas e que sejam possíveis de modelar. Pelo fato de, durante a fase da modelagem, não se obter um modelo que contenha todas as características do sistema deve se considerar um certo nível de abstração que o modelo deve abranger [1, 32, 33]. Alguns estudos relacionados [1, 11, 31] basearam-se fortemente na simulação para abstrair equipes de desenvolvimento de software. Dafoulas *et al* [34] apresentou uma pesquisa com uma abordagem para a criação de estudos-piloto simulando principais características que fazem com que equipes de Desenvolvimento de Software Globais sejam susceptíveis a um bom gerenciamento.

Comparativamente à monitoração, a simulação costuma ser menos dispendiosa e consome menos tempo para que os índices sejam calculados, permitindo que sejam feitos quantos experimentos forem necessários. Porém, por se tratar de uma abstração da realidade, a fidelidade das medidas tende a ser menor na simulação se compararmos com a monitoração. Além disso, da mesma forma que na monitoração, a quantidade e representatividade das amostras consideradas é muito importante para a obtenção de resultados corretos [18].

2.1.1.3 Métodos Analíticos

Esta técnica de avaliação de desempenho, do mesmo modo que a simulação, também se baseia no desenvolvimento de um modelo do sistema real, porém com um nível de abstração do sistema real muito mais alto. Neste caso, o modelo é puramente matemático. Neste tipo de modelo, o funcionamento do sistema real é reduzido a relações matemáticas. Modelos analíticos podem ser determinísticos ou estocásticos. Em um modelo determinístico, todos os parâmetros do sistema são previamente determinados. Já em um modelo estocástico, o comportamento do sistema é analisado probabilisticamente, ou seja, os parâmetros do sistema são descritos por variáveis aleatórias, com distribuições de probabilidade convenientes [18]. No último caso, o sistema é descrito em termos de um conjunto de estados em que o mesmo pode se encontrar e as transições estocásticas entre esses estados (uma transição estocástica é aquela cuja ocorrência é descrita por uma variável aleatória)

[13]. Desenvolver modelos analíticos normalmente exige maior abstração de aspectos da realidade, se comparado a modelos de simulação. Ainda, em alguns casos, não se consegue obter uma resolução numérica, mas sim uma resolução analítica. Algumas vezes a complexidade computacional do modelo pode tornar a resolução muito cara, ficando mais dispendiosa que uma resolução igualmente aceitável em simulação [18].

Segundo referenciado no estudo de Chanin *et al* [18], apesar de em algumas situações os métodos analíticos poderem ser mais dispendiosos, eles podem ser empregues com maior facilidade que outros métodos em vários casos. Uma vantagem desta técnica em relação as outras descritas é que não há a necessidade de se preocupar com um conjunto específico de amostras de funcionamento do sistema para a obtenção dos índices de desempenho o que torna o domínio de métodos analíticos para avaliação de sistemas constituir uma ferramenta importante para profissionais da área de Informática.

2.1.2 Medidas de Desempenho

Desempenho pode ser definido como a maneira como um sistema se comporta. Isto é, o desempenho de um sistema é determinado por suas características de execução. Portanto, avaliar o desempenho de um sistema demanda definir quais características comportamentais interessam ser consideradas [18]. Para sistemas computacionais, em geral consideram-se 4 fatores para medida de desempenho:

- Vazão (throughput): taxa de atendimento de pedidos pelo sistema. Ex.: Sistemas em lotes: jobs por segundo; Sistemas interativos: requisições por segundo; CPU: MIPS ou MFLOPS; Redes: pacotes por segundo (pps) ou bits por segundo (bps); Sistemas de processamento de transações: transações por segundo (TPS);
- Utilização: fatia de tempo em que o sistema permanece ocupado, atendendo a pedidos;
- População: quantidade de atendimentos a serem feitos em um determinado instante;
- Tempo de resposta: intervalo de tempo entre o pedido e o início/conclusão do serviço.

2.2 Modelos de Desenvolvimento de Software

Modelos de desenvolvimento de software têm como objetivo descrever o que é necessário fazer para que um software seja produzido, não lhes cabendo dizer como implementar, ficando a tarefa de como implementar o software a cargo de um framework de desenvolvimento, muitas vezes estes frameworks levam o mesmo nome do modelo em si mas podem haver diferentes frameworks para um mesmo modelo.

Embora existam muitos processos de desenvolvimento de software diferentes, há atividades fundamentais comuns a todos eles [2]:

- Especificação de softwares que consiste em definir as funcionalidades do software e as restrições em sua operação.
- Projeto de implementação de software que garante que o software deve ser produzido de modo que cumpra as suas especificações.
- Validação de software onde é feita a validação do software para garantir que ele faz o que o cliente solicitou.
- Evolução do software que garante a evolução de software de modo a atender ás necessidades mutáveis do cliente.

2.2.1 Modelos de Processos Prescritivos

São considerados modelos de processos prescritivos (ou modelos pesados, ou ainda modelos centrados no planejamento) de desenvolvimento de software aqueles que tentam prever todos os requisitos do sistema, ou seja, se caracterizam por focar a análise de sistemas, investindo esforço para a obtenção de artefatos de projeto tão completos e precisos quanto possível: relatórios, documentos e diagramas. O principal modelo prescritivo e muito utilizado até hoje é o modelo de processos Waterfall que é brevemente descrito na subsecção a seguir.

2.2.1.1 Modelo de Processos Waterfall

O modelo de processos Waterfall, que também é conhecido por abordagem "top-down", foi proposto por Royce em 1970. Até meados da década de 1980 foi o único modelo com aceitação geral. Esse modelo foi derivado de modelos de atividade de engenharia com o fim de estabelecer ordem no desenvolvimento de grandes produtos de software.

No modelo de processos Waterfall as etapas são apresentadas em sequência, como em uma cascata, dai a designação Waterfall. O desenvolvimento de uma fase deve terminar antes da próxima iniciar e assim por diante [20], ou seja, este modelo argumenta que cada atividade apenas deve ser iniciada quando a outra estiver terminada e verificada. Ele é considerado monolítico por não introduzir a participação de clientes e usuário durante as atividades do desenvolvimento, eles apenas são introduzidos quando o software ter sido implementado e entregue [20]. Não existe como o cliente verificar antecipadamente qual o produto final para detectar eventuais problemas.

O modelo de processos Waterfall facilita o entendimento do cliente não familiarizado com o desenvolvimento de software, facilitando deste modo, a obtenção de informações desejadas dos

clientes. Porém, este modelo peca pela falta de orientação para os gerentes e desenvolvedores de como tratar as mudanças que ocorrem durante o desenvolvimento. Segundo Sommerville [2], as etapas de desenvolvimento do modelo de processos Waterfall são: definição de requisitos, projeto de sistema e de software, implementação e testes de unidades, operação e manutenção. A Figura 2.1 ilustra o esquema gráfico do modelo de processos Waterfall.

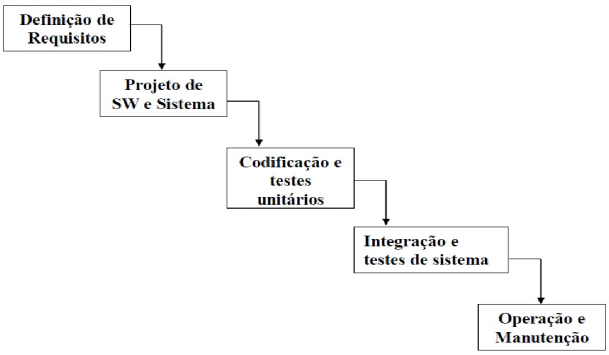


Figura 2.1 – Modelo Waterfall. Fonte: [2]

A descrição mais detalhada de cada fase do modelo de processos Waterfall pode ser encontrada no trabalho [20]. Salientar que a fase de operação e manutenção, não necessariamente é a fase mais longa do modelo de processos Waterfall. A manutenção envolve corrigir erros que não foram descobertos em fases anteriores do modelo Waterfall, melhorando a implementação e aumentando as funcionalidades do sistema em causa á medida que novos requisitos são descobertos e isso pode implicar novos projetos que em função da sua natureza podem determinar o uso de um modelo de processos especifico e não necessariamente a que foi usada até esta fase.

2.2.1.2 Vantagens do Modelo Waterfall

- Torna o processo de desenvolvimento estruturado e tem uma ordem sequencial de fases;
- Cada fase cai em cascata na próxima e cada fase deve estar terminada antes do início da seguinte;
- Todas as atividades identificadas nas fases do modelo são fundamentais e estão na ordem certa;
- Esta abordagem é na maior parte das vezes a norma de desenvolvimento de software de grande porte;
- Permite a gerência do baseline, que identifica um conjunto fixo de documentos produzidos como resultado de cada fase do ciclo de vida.

2.2.1.3 Desvantagens do Modelo Waterfall

- Não fornece feedback entre as fases e não permite a atualização ou redefinição das fases anteriores;
- Não é flexível, pois, não suporta modificações nos requisitos;
- Não permite a re-utilização;
- Se ocorrer um atraso numa das fases de desenvolvimento todo o processo é afetado;
- A entrega de um software neste modelo é muito demorada;
- Este processo possui dificuldades em acomodar mudanças depois de começado, pois um nível depende do outro, sendo assim, se alterarmos o segundo nível da cascata tendo já completado o terceiro nível, por exemplo, provavelmente teremos que refazer o primeiro nível [28].

2.2.2 Modelos de Processos Ágeis

Devido a cada vez mais crescente procura de softwares, há necessidade de desenvolver software de forma mais rápida, mas com qualidade necessária. Esse desenvolvimento pode ser obtido utilizando modelos de processos ágeis. Nos últimos anos, modelos de processos ágeis passaram a ser usados na academia e na indústria de softwares. Dentre os vários modelos ágeis propostos no mercado de softwares o eXtreme Programing (XP) e Scrum são os modelos mais usados para desenvolvimento de Sofware. O XP é mais focado em boas práticas de desenvolvimento e o Scrum é um framework focado principalmente em planejamento e gerência.

O surgimento dos modelos ágeis (também chamadas de modelos leves ou lightweight methodologies) deveu-se aos questionamentos que eram levantados em torno dos problemas que os modelos traicionais apresentavam no final da década de 90 [38]. Dos principais motivos desse questionamento se destacou a alta frequência com que os projetos de software deixavam de cumprir seus cronogramas e extrapolavam seus orçamentos e a dificuldade de uso dos modelos prescritivos para determinados tipos de projetos de softwares.

2.2.2.1 Modelo Extreme Programing (XP)

Segundo Beck [39], Extreme Programing (ou simplesmente XP) "é um modelo ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante e rápida mudança".

O XP destaca-se perante os outros modelos de processos pelo fato de dar feedback constantes, a abordagem de desenvolvimento é incremental e encoraja uma constante comunicação podendo até não ser uma comunicação formal. As regras do XP, aplicadas isoladamente não fazem

nenhum sentido e não surtem o efeito desejado pelo modelo. Para se garantir o sucesso na aplicação do modelo XP as suas regras devem ser aplicadas em conjunto [39]. Apesar de os modelos ágeis (XP) trazerem uma revolução no desenvolvimento de software muita organizações resistem a aderir aos modelos ágeis e se mantendo fieis aos modelos prescritivos.

Embora estejamos falando de processos durante todo o tempo até agora, XP não se apresenta como um processo, mas sim como um conjunto de práticas que, ao ser usado por completo, tem a capacidade de produzir resultados melhores do que o que seria produzido pela soma dos resultados alcançados por cada prática aplicada individualmente. Os seguidores do modelo XP são conduzidos por 4 valores: comunicação, simplicidade, feedback e coragem [39].

A comunicação permite estabelecer o melhor relacionamento possível entre as partes envolvidas no projeto. A XP estabelece que deve haver uma comunicação frequente, independentemente dela ser formal ou não. Encoraja a comunicação face to face o máximo possível em vez de recorrer o telefone e outros meios de comunicação.

A simplicidade sustenta-se no uso de código simples evitando o máximo possível o uso de classes e métodos. A simplicidade garante ainda que se implemente requisitos atuais desde que os objetivos sejam alcançados, evitando implementar requisitos que possam ser úteis no futuro.Melhor implementar o útil agora e pagar "um pouco mais" para futuras modificações.

O constante feedback, ou retorno, determina a resposta para uma ação ou prática adotada no projeto, como testes de unidade e planejamento. O feedback permite que o cliente tenha acesso a uma parte funcional do software e possa dar sugestões de mudanças na funcionalidade do software.

O quarto valor que é a coragem garante que os três valores referenciados sejam aplicados. Por exemplo, mesmo que uma parte do código esteja a devolver o resultado, se haver uma oportunidade de simplificar o código, o desenvolvedor do código deve tomar coragem para tal. Segundo Teles [40], os valores devem tornar-se um hábito dos envolvidos no projeto e devem ser colocados em prática em conjunto para o sucesso de um projeto XP.

2.2.2.2 Práticas da XP

A seguir são apresentadas as 12 práticas da XP, conforme descrito por Teles [41]:

- Jogo de planejamento: No XP, as funcionalidades são descritas em pequenos cartões e são chamados de estórias. No planejamento define-se quais os requisitos atuais que precisam ser implementados. Também visa estabelecer as regras do jogo de desenvolvimento para os clientes e os desenvolvedores. Na fase do planejamento são acordados os custos e definidos os prazos para entrega do projeto de modo similar aos modelos prescritivos.

- Small releases (entregas frequentes): A cada pequeno ciclo de desenvolvimento é liberada uma versão funcional do sistema. Os small releases melhoram a participação do cliente e evitam surpresas no fim do projeto, garantindo, deste modo, que o produto final esteja de acordo com os requisitos dos clientes.

- Metáfora: Metáforas ou histórias de usuários são descrições do funcionamento do sistema, baseada em uma linguagem não técnica para ajudar os envolvidos no projeto a ter uma visão de seu funcionamento.
- Projetos simples: Processos complexos devem ser removidos, bem como o uso de classes e métodos deve ser o mínimo possível.
- Testes: Os programadores escrevem testes unitários que são combinados em conjuntos de testes, os quais devem ser rodados com frequência.
- Refactoring (Refatoração): É o melhoramento contínuo do projeto. O código vai sendo reestruturado, removendo-se duplicações e adicionando-se flexibilidade.
- Programação em pares: O código do sistema é escrito por dois programadores na mesma máquina e os pares devem ser formados dinamicamente onde um digita e o outro vai acompanhando e auxiliando na correção de erros.
- Propriedade coletiva: O código pode ser alterado por qualquer programador a qualquer momento.
- Integração contínua: O código é integrado aos diversos módulos do projeto diversas vezes ao dia. A cada integração o código deve ser testado.
- Semana de 40 horas: Trabalhar por períodos muito longos não é produtivo. O bem estar da equipe é fator importante para uma boa produtividade e qualidade no trabalho.
- Cliente presente: Todos os envolvidos no projeto dividem o mesmo ambiente, formando uma equipe, incluindo o cliente.
- Padrões de codificação: "Os programadores escreverão código respeitando as regras que enfatizam comunicação através do código".

A Figura 2.2 ilustra o conjunto das práticas do modelo XP. O círculo maior ilustra as práticas da organização no seu todo, o círculo médio representando as práticas da equipe envolvida num determinado projeto aplicando o modelo XP e por fim, o círculo menor, representa especificamente as boas práticas do par de programadores.

2.2.2.3 Modelo Scrum

O Scrum é um modelo de processos ágil que objetiva desenvolver e entregar o software com a maior qualidade possível dentro de séries, compostas por Sprints, intervalos de tempos definidos para o desenvolvimento [42].

Enquanto a XP focalize a programação, o Scrum dá ênfase ao gerenciamento do projeto. Scrum, usado principalmente para projetos de software, também pode ser usado para projetos sem relação com software, pois seus princípios são aplicados a qualquer projeto [42].

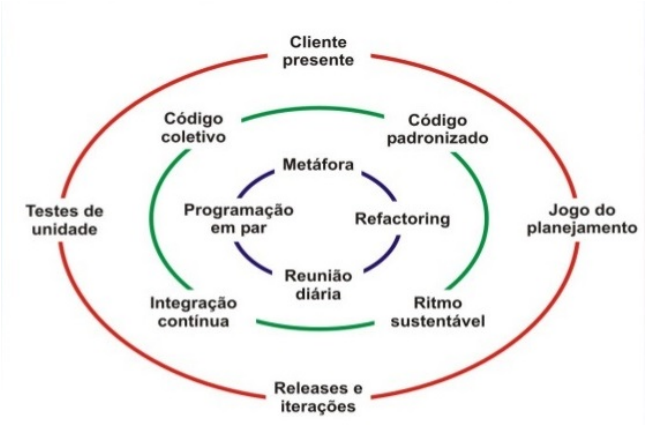


Figura 2.2 – Esquema das práticas do XP. Fonte: [43]

Ao contrário dos processos prescritivos, no Scrum os requisitos não precisam ser detalhados no começo do projeto, eles vão sendo definidos com sua evolução. Com recurso ás reuniões diárias são definidas novas tarefas do processo de desenvolvimento. Nas reuniões diárias busca-se identificar problemas referentes ao andamento do projeto o mais cedo possível e propor as respetivas soluções.

A seguir são detalhados alguns fundamentos e conceitos do modelo Scrum segundo Schwaber *et al* [43]:

- Scrum Master (Mestre Scrum): Garante a utilização dos valores, práticas e regras do Scrum. Relaciona-se com o cliente e com cada membro da equipe, escutando o que cada um tem pra relatar; é o representante do modelo.
- Backlog do produto: É uma lista das funcionalidades que estarão no software. Nesta lista também serão especificados os novos requisitos, tarefas, funções e tecnologias que poderão ser utilizadas.
- Sprint: É o ciclo de desenvolvimento do Scrum e geralmente dura trinta dias.
- Times Scrum: Ou equipes Scrum, incluem desenvolvedores e usuários. Estes times são pequenos e geralmente possuem sete componentes.
- Sprint Backlog: É o backlog que será executado durante o ciclo do Sprint, ou seja, é uma lista com funcionalidades, atividades e tarefas a serem executadas e desenvolvidas durante o Sprint.
- Reuniões de Scrum diárias: Nesta reunião os integrantes da equipe analisam e falam o que foi feito desde a última reunião e o que pretendem fazer até a próxima.
- Reunião de Planejamento (Sprint Planning Meeting): Nesta reunião, realizada antes do Sprint, serão definidos os itens que vão compor a lista de backlog do Sprint, bem como a ordem de prioridade que eles terão. Aqui os requisitos podem ser alterados, sendo que após o início do Sprint só os desenvolvedores podem realizar mudanças.

O modelo Scrum não possui atividades e práticas que orientam o processo na hora do desenvolvimento, seu enfoque é para o controle do gerenciamento do projeto. A Figura 2.3 reproduz graficamente o ciclo do Scrum. O ciclo do Scrum pode ser dividido, em três partes [44], são elas: pré-sprint (pontos 1,2 e 3 na Figura 2), Sprint (pontos 4 e 5) e pós-sprint (ponto 6). De acordo com Highsmith *et al* [45], o pré-sprint inicia-se com a definição da lista do backlog de produto, através de uma reunião de planejamento, onde o usuário vai definir e classificar as funcionalidades desejadas para o software por nível de prioridade.A fase de Sprint completa as tarefas definidas para ela no pré - sprint e entregar uma pequena parte funcional do software, também conhecida como incremento. Para tanto a equipe é livre para desenvolver, desde que os objetivos sejam alcançados. Uma vez iniciada só os desenvolvedores envolvidos poderão fazer mudanças nas atividades especificadas [46]. Para fechar o ciclo (pós-sprint) é apresentado ao cliente o produto da iteração, ou seja, o pedaço executável de software ou incremento é analisado pelos usuários. Também deverá ter a sua documentação, que será entregue ao cliente [44].

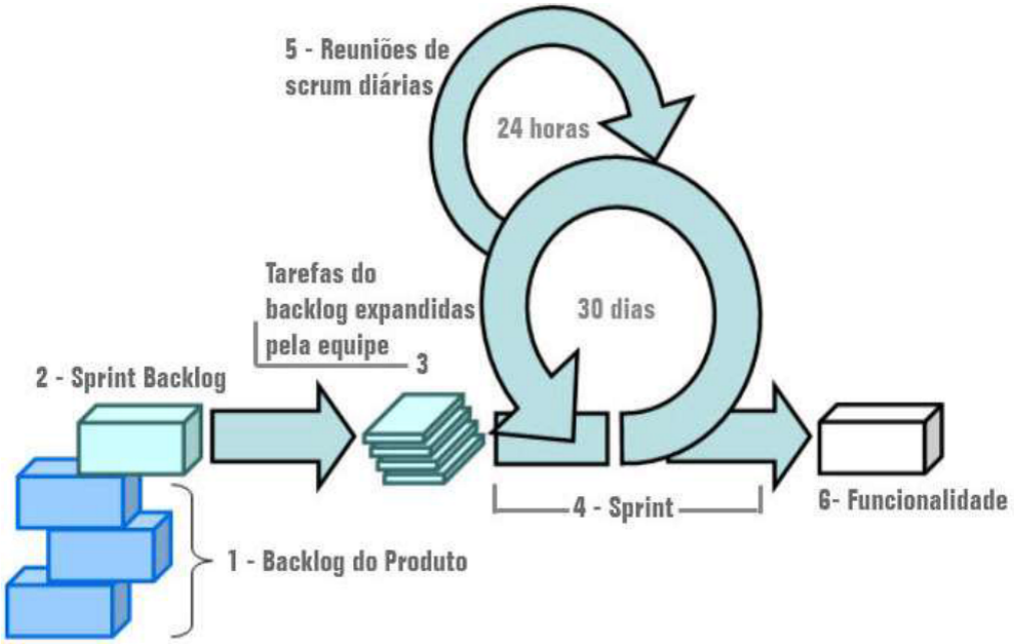


Figura 2.3 – Modelo Scrum. Fonte :Adaptado www.controlchaos.com

2.3 Justificativa da Escolha do Modelo de Processos Waterfall

Existem diversas variantes do modelo Waterfall propostas por diferentes pesquisadores ou empresas de desenvolvimento de software adaptadas a diferentes tipos de softwares em função dos requisitos aos quais o software deve responder. As várias modalidades do modelo Waterfall convergem numa característica comum que consiste no fluxo linear e sequencial das atividades. Um dos aspetos que mais chama atenção e que é uma das maiores razões do seu uso é a maneira fortemente disciplinada como as atividades do projeto de desenvolvimento de software são conduzidos. Cada fase inicia apenas quando a anterior termina e é aprovada pelos integrantes do projeto.

Muitas empresas de software mostram-se resistentes em aderir os modelos ágeis pelo fato de estes modelos não terem foco no gerenciamento de risco e a data fim do projeto não é determinada no início do projeto, ou seja, ela vai se desenhando ao longo do desenvolvimento do projeto. Os custos do projeto são conhecidos apenas ao longo do projeto o que obriga os gestores a gastarem mais tempo no controle dos custos resultantes das possíveis mudanças de requisitos [22].

Com as etapas tão bem definidas, claras e sequenciais, os projetos Waterfall ficam mais simples de serem compreendidos pela equipe e o fluxo das atividades é mais organizado e bem administrável. Para combater a falta de flexibilidade do modelo Waterfall foi definido um novo processo baseado no Waterfall, designado Waterfall revisto que se diferencia do Waterfall clássico pelo fato de dar a possibilidade de, a partir de qualquer tarefa do ciclo de vida, se poder regressar a uma tarefa anterior de modo a contemplar alterações funcionais e/ou técnicas que eventualmente tenham surgido em consequência da dinâmica que sujeita os requisitos a possíveis alterações significativas. Porém, as possíveis mudanças são sujeitas a uma profunda análise e impacto de modo a não comprometer o propósito inicial do projeto.

Apesar de o modelo Waterfall ser bastante criticado por ser linear, rígido e monolítico, várias propostas têm sido avançadas no sentido de complementar a sua inflexibilidade com base nas boas práticas previstas pelos modelos ágeis. Devido à grande resistência das grandes empresas de software abrirem mão do modelo Waterfall vários estudos atuais, como por exemplo o estudo [22], defendem que para tornar ágil o modelo Waterfall, todas as partes envolvidas no projeto (fornecedor e cliente) devem estar de acordo que, na elaboração da proposta do projeto, a fase de definição de requisitos requer refinamentos até a obtenção de um nível de detalhamento de requisitos implementável [22].

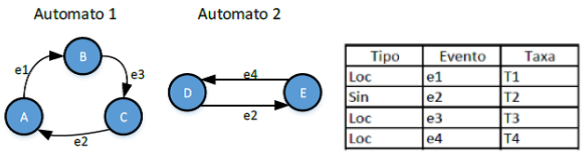
Agilização do modelo Waterfall sugere a promoção de comunicação (um dos principais pontos fortes dos modelos ágeis) entre o fornecedor e o cliente ao longo do projeto de modo facultar ao cliente entendimento e andamento do projeto.

O modelo Waterfall é ideal para projetos em que os requisitos são bem conhecidos e não sofrem mudanças significativas ao longo do tempo. Os resistentes a adotar os modelos ágeis acreditam que ser ágil é mudar toda hora. Pelo fato de os modelos ágeis serem bem aplicáveis em equipes com poucos integrantes, os grandes projetos de software ainda optam pelo modelo de processos Waterfall que pela sua estruturação permitem a gestão de grandes equipes de desenvolvimento, inclusive a gestão de equipes geograficamente distribuídas [22].

2.4 Redes de Autômatos Estocásticos (Stochastic Automata Networks – SAN)

Redes de Autômatos Estocásticos (SAN) são um formalismo Markoviano modular e compacto [1, 12, 13]. SAN é usado para modelar sistemas de pequenos componentes (autômatos) com interações pontuais e definidos entre eles de uma forma estruturada. A solução de um modelo SAN (extração dos resultados numéricos) é geralmente realizada por algoritmos específicos, projetados

para lidar com espaços de estados bastante grandes [1, 17]. Por esta razão, SAN facilita a modelagem de equipes de desenvolvimento de software (em um contexto de engenharia de software) pela descrição individual de cada comportamento da entidade, onde apenas algumas atividades representam sincronizações entre algumas entidades. Neste contexto, uma entidade pode ser modelado como um participante especifico da equipe ou uma equipe inteira [1]. A Figura 2.5 exemplifica uma Redes de Autômatos Estocásticos com dois automas.



Tipo	Evento	Taxa
Loc	e1	T1
Sin	e2	T2
Loc	e3	T3
Loc	e4	T4

Figura 2.4 – Modelo SAN com dois autômatos

2.4.1 Autômatos Estocásticos

Autômato estocástico é um modelo matemático de um sistema que possui entradas e saídas discretas. O sistema pode se encontrar em qualquer um dentre o número finito dos estados do sistema ou das conFigurações internas. O estado interno em que o sistema se encontra sumariza as informações sobre as entradas anteriores e indica o que é necessário para determinar o comportamento do sistema para as entradas seguintes [49]. Baseado nessa definição, pode-se descrever um autômato estocástico como um conjunto finito de estados e um conjunto finito de transições entre esses estados. A denominação de estocásticos atribuída a esses autômatos dá-se pela razão do tempo ser tratado como uma variável aleatória, a qual obedece a uma distribuição exponencial à escala de tempo contínua e uma distribuição geométrica à escala de tempo discreta [49].

2.4.2 Eventos Locais e Sincronizantes

O formalismo SAN modela dois tipos de eventos, nomeadamente: eventos locais que são utilizados em SAN para alterar o estado local de um único autômato sem que essa alteração ocasione uma mudança de estado em qualquer outro autômato do modelo e eventos sincronizantes alteram o estado local de dois ou mais autômatos simultaneamente, ou seja, a ocorrência de um evento sincronizante em um autômato força a ocorrência deste mesmo evento nos outros autômatos envolvidos [49].

A classificação de um evento como local ou sincronizante é dada pela aparição do identificador do evento e no conjunto de eventos de um autômato. Caso o identificador do evento apareça apenas no conjunto de eventos de um único autômato, o evento é classificado como evento local. Se

o mesmo identificador aparecer no conjunto de eventos de vários autômatos, o evento é classificado como evento sincronizante [49].

2.5 Software PEPS (Performance Evaluation of Parallel Systems)

PEPS (Performance Evaluation of Parallel Systems) é um pacote de ferramentas para modelar e resolver modelos expressos em Redes de Autômatos Estocásticos (SAN). O projeto PEPS teve o seu inicio nos finais dos anos 80 com vista a prover uma ferramenta de software para modelar e determinar soluções numéricas para o formalismo SAN [15]. O formalismo SAN foi proposto por Plateau [1] e a sua ideia básica era de representar um sistema no seu todo com recurso a um conjunto de subsistemas com um comportamento independente (eventos locais) e com algumas interdependências ocasionais através de taxas de transição e eventos sincronizantes. No ano 2000 surgiu uma nova versão do PEPS (PEPS2000) que implementou um conjunto de algoritmos propostos por Fernandes *et al* [15]. O principal incremento nesta nova versão foi o método de multiplicação do vetor-descritor.

A versão PEPS2003 basicamente, trouxe uma nova interface de compilação, uma esparsa manipulação do vetor e uma avaliação de funções Just-in-time. A nova interface do compilador é mais intuitiva e compacta. Outro recurso implementado no PEPS2003 é a avaliação da função just-in-time. Este método gera, para cada função descrita no modelo SAN, um código de C ++.Os códigos de função C ++ são compilados e relacionados com os métodos de solução PEPS e eles são chamados toda vez que uma função é avaliada [15].

Mais tarde o PEPS evoluiu para a versão PEPS2007 na qual foi adotada uma nova estrutura do software. PEPS divide-se num conjunto de módulos independentes onde cada modulo implementa à parte os procedimentos de solução. Com esta abordagem, novos métodos podem ser desenvolvidos e testados independentemente da versão estável PEPS. Especificamente, ele permite diferentes versões de desenvolvimento localmente e / ou logicamente distribuída sem uma preocupação particular com correções de bugs recentes e melhorias em outras partes do software PEPS [15].

Os módulos básicos PEPS2007 são montados em três grupos: Interface, estrutura de dados, e métodos de solução. Juntamente com os módulos anteriores de implementação dos recursos das versões anteriores, também estão sendo implementados novos recursos da versão PEPS2007. Atualmente já está disponível a versão PEPS2009 que nas suas novas funcionalidades consta um compilador novo, aceitando nova sintaxe para definição dos autômatos e o Split que é um novo método de multiplicação vetor-descritor [15].

O PErhaPS (Performance Evaluation of 'rha' Parallel Systems) trata-se da versão do PEPS para a plataforma Windows. O software é compilado com o DevC++ e possui os últimos recursos em termos de novas funcionalidades que foram agregadas na ferramenta ao longo do tempo inclusive o Split.

3. METODOLOGIA

3.1 Estratégia de Pesquisa

Ao se iniciar a pesquisa dos estudos primários, sugere-se que uma estratégia deve ser usada, através da definição das palavras chaves, bibliotecas digitais, jornais e conferências [36]. A presente pesquisa teve seu início baseado na revisão da literatura sobre o assunto a fim de amadurecer a ideia e definir claramente o objeto de estudo. A revisão literária fez-se em duas partes, nomeadamente: primeira - modelos de desenvolvimento de software e a segunda - modelagem estocástico para avaliação de desempenho de equipes. Foram identificadas várias publicações (teses, jornais, conferencias, etc). Esta pesquisa, dadas as suas características, ela tem bases que lhe dão potencial para ser considerada exploratória porque baseou-se na revisão bibliográfica, entrevistas com pessoas mais experientes no assunto relacionado com avaliação de desempenho de equipes de desenvolvimento de software de modo a poder proporcionar maior familiaridade com o assunto. Duma forma geral este estudo assumiu uma forma de revisão bibliográfica e estudo de caso que segundo Gil [37] são as características básicas de um estudo exploratório.

3.2 Fases da Pesquisa

Esta seção consiste da descrição das fases consideradas para alcançar os objetivos definidos neste trabalho. O primeiro estágio é composto pela revisão da literatura (trabalhos relacionado) que compôs o Capitulo 2. Ainda no primeiro estágio foi feita a descrição do estudo de caso concreto de projeto de desenvolvimento de software com base no modelo waterfall que foi identificado e selecionado para este trabalho. O segundo estágio é composto pelo exercício de concepção do modelo e testes da sua execução no software PEPS. O terceiro e último estágio do modelo é composto pelo modelo final, resultados da sua execução e a respetiva discussão dos resultados.

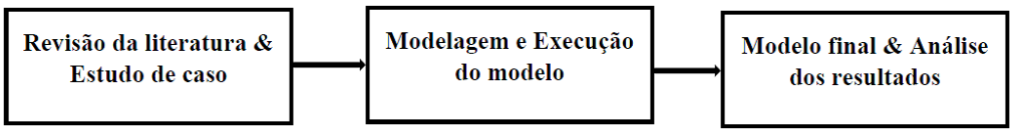


Figura 3.1 – Fases da pesquisa

3.3 String de Busca

As strings de pesquisa foram definidas a partir das palavras-chave relevantes para o assunto. Usando diferentes combinações de strings foram buscadas algumas publicações relacionadas com

os modelos de desenvolvimento de software. Estudou-se com mais profundidade os modelos de desenvolvimento de modo a adquirir uma base para sustentar a escolha da avaliação de desempenho de uma equipe que usa o modelo de processos waterfall nos projetos de software. As strings mais usadas para modelos de desenvolvimento de software foram:

- Modelos de desenvolvimento;
- Modelos de processos de desenvolvimento;
- Software Development Model;
- Heavy models;
- lightweight models/Methodology;
- prescriptive models/Methodology;
- Modelos/Métodos pesados;
- Modelos/Métodos prescritivos;
- waterfall model/Methodology;
- Modelo/Método/Metodologia cascata;
- Modelo/Método/Metodologia espiral;
- Modelos/Métodos/Metodologias ágeis;
- Agile Models/Methodology;
- Xtreme Programing;
- Scrum.

Em muitas situações alternou-se a palavra metodologia ou método no lugar de modelo para as buscas em Português assim com para Inglês. De salientar que as strings de busca foi usadas considerando também a sua tradução em português ou inglês. No lugar de modelo também foi muitas vezes usado modelo de processos. Essa alternância das palavras modelo, método, metodologia e modelos de processos deveu-se ao fato de existir na literatura uma controvérsia nos conceitos das mesmas.

A busca de literatura para segunda parte da revisão literária que aborda modelos estocásticos para avaliação de desempenho de equipes foi feita quase que de uma maneira direta com temas das publicações uma vez que o grupo de pesquisa sobre o assunto já vem efetuando estudos sobre o assunto e possui um número considerável de artigos relacionados ao assunto, simplesmente teve lugar uma leitura nos títulos e resumos de modo a validar o uso de parte do material fornecido.

3.4 Busca da Literatura

De modo a selecionar as fontes foram definidos, a disponibilidade dos artigos na web, mecanismos de busca através das diferentes palavras-chave como critérios de busca. Uma boa parte dos artigos foi obtido através de pessoas com experiência no assunto e com várias publicações credíveis sobre o assunto. Os idiomas de busca foram o inglês e o português, tendo sido na sua maior parte os resultados credíveis obtidos no idioma inglês. As principais fontes digitais usadas foram:

- IEEEXplore Digital Library,
- ACM Digital Library,
- Elsevier ScienceDirect,
- ResearchGate.

Para além da busca digital foi incluído no estudo a consulta a 3 livros que abordam o assunto sobre modelos de desenvolvimento [2, 19, 21].

3.5 Critério de Inclusão de Artigos

Para a definição de critérios de inclusão de artigos relevantes ao estudo foi feita uma análise com base no título, palavras-chave, resumo, introdução e conclusão de cada artigo identificado. A análise feita foi para uma verificação e validação uma vez que na sua maioria os artigos foram obtidos por meio de pessoas experientes no assunto que disponibilizaram uma boa parte dos artigos.

3.6 Revisão da Literatura

Esta revisão de literatura tem como objetivo encontrar estudos existentes relacionados com os objetivos definidos para esta monografia. Esta pesquisa teve como base de realização uma revisão da literatura com foco em: Modelos de Processos de Desenvolvimento de software. A revisão da literatura foi feita com recurso a 4 bibliotecas digitais listadas na Seção 3.4. A pesquisa foi iniciada no primeiro semestre de 2015 e terminou em meados do segundo semestre do mesmo ano. Usando as diferentes strings mencionadas na Seção 3.2 foram listados 91 estudos e 20 deles foram selecionados e analisados. A busca obedeceu a seguinte distribuição:

- IEEEXplore: foram encontrados 46 artigos com base na alternância dos strings de busca definidos para o efeito. Dos 46, apenas 4 foram revistos, nomeadamente: [15], [17], [22] e [25].

- ACM Digital Library:Nesta livraria digital o resultados de busca não foram satisfatórios, ou seja, nenhum resultado significativo foi encontrado.
- Elsevier ScinceDiret: foram encontrados 8 artigos mas neste caso nenhum dos artigos foi aprofundados porque os 8 artigos conduziam para os livros sobre engenharia de software de Pressman [19], Sommerville [2].
- ResearchGate: Nesta ferramenta foram encontrados 28 artigos dos quais foram selecionados e revistos 5, que são: [8], [14], [16], [20] e [32].

Para além das ferramentes de busca acima mencionadas, serviram de fontes de material bibliográfico alguns profissionais e estudantes da área do grupo de pesquisa sobre o assunto. De forma direta do grupo de pesquisa obteve-se os seguintes trabalhos nomeadamente: [1], [11], [13], [17], [23], [24], [31], [32] e [36].

Com base na revisão dos 20 estudos selecionados e com base no nome dos autores e os títulos das obras referenciados nos artigos, foram identificados e selecionados outros estudos que complementaram a lista das referências bibliográficas apresentada no final desta monografia.

As fases subsequentes da metodologia (estudo de caso e modelagem) são bem descritas nos Capítulos 4 e 5.

4. ESTUDO DE CASO

Este capitulo dedica-se à descrição do estudo de caso que é base para concepção do modelo proposto no Capitulo 5. O projeto consiste de um Portal do Concessionário de um banco que será designado de XYZ por razões de confidencialidade. Neste capitulo usa-se o modelo proposto no capitulo 5 para instanciar as fases do estudo de caso para evitar separar as instanciações das descrições de cada uma das fazes do estudo de caso de para melhor enquadramento e compressão. O ideal seria usar modelo depois de ter sido introduzido e descrito mas, uma vez que o estudo de caso é fundamental para a abstração do modelo surge esta necessidade de flexibilização e iteração entre os Capítulos 4 (Estudo de caso) e 5 (Modelo SAN). Os dados quantitativos do estudo de caso XYZ são apresentados na Seção 6.1.1 do Capitulo 6 onde são usados para efeitos de comparação com os resultados numéricos obtidos neste trabalho. A Figura 4.1 do modelo SAN é usada neste capitulo apenas para percepção da instanciação das fases do estudo de caso. Mais detalhes sobre a concepção do modelo é abordado no Capitulo 5 onde aparece a mesma Figura 4.1 mas em outro contexto.

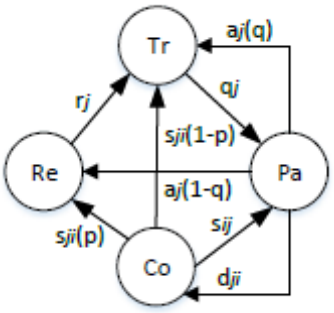


Figura 4.1 – Modelo SAN

4.1 Estudo de Caso XYZ

No paper [1] foi apresentado um modelo analítico SAN para um estudo de caso prático de uma companhia de Tecnologias de Informação (TI) que consistia de múltiplos locais com participante que desempenhavam diferentes papeis na equipe e com diferentes níveis de experiencia. Nesse estudo foi feita uma confrontação entre os prazos obtidos através do modelo SAN contra os prazos atuais (observados na realidade) do estudo de caso considerado no paper. O estudo de caso considerado consistia de uma equipe central onde se variou o nível de disponibilidade e o nível do suporte técnico providenciado pela equipe central de modo a verificar o seu impacto na performance da equipe. O estudo [1] foi resumido com novas discussões de resultados numéricos e possíveis extensões do modelo proposto. No sentido de determinar o nível de acurácia, os resultados numéricos obtidos no modelo proposto foram comparados com as horas atuais (observadas na realidade) gastas nas fases

concretas de um projeto concreto. Foi feito também a análise de possíveis variações dos cenários do projeto considerando diferente comportamentos e habilidade.

Segundo Santos [31] no mesmo contexto das pesquisas do paper [11] ainda existem oportunidades de capturar outros aspetos do GSD usando configurações FTS, tais como a eficiência de handoff, controle das fases do projeto, capacidade das equipes, dentre outros aspetos inerentes ao GSD. Na sequência dessas oportunidades Santos [31] identificou e catalogou aspetos que podem influenciar o processo de tomada de decisão nos projetos de desenvolvimento de softwares. Na sua pesquisa a sua questão era de compreender como qualificar os processos de tomada de decisão nos projetos FTS, avaliando fatores como, custo, tempo e qualidade com base num estudo de caso pratico.

Nessa ordem, de modo a substanciar a usabilidade da modelagem analítica para análise de performance de equipes de desenvolvimento de softwares, o presente trabalho apresenta um de estudo de caso prático que vai se designar XYZ por questões de confidencialidade. O dados do estudo de caso XYZ servem para instanciação do modelo SAN proposto neste trabalho. No projeto foram usados 8 recursos (elementos da equipe) nomeadamente:

- 2 Analistas de Sistemas,
- 3 Desenvolvedores,
- 1 Analista de Testes e
- 2 Testers.

O projeto do estudo de caso foi desenvolvido com base no modelo de processos prescritivo Waterfall. Uma vez que o modelo de processos Waterfall considera as fases do projeto como distintas, ou seja, cada fase só inicia depois que a anterior termina, as fases serão modeladas como distintas. O projeto teve lugar de 10/05/2011 a 18/11/2011 (6 meses). A carga horária padrão em todas as fases com exceção da desenvolvimento foi de 9 horas/dia. Durante a fase de desenvolvimento usou-se uma carga horaria média de 12 horas/dia. As atividades eram realizadas nos dias uteis. Na subsecção que se segue serão ilustradas as fazes que compõem o projeto XYZ bem como a alocação do tempo (parcial ou inteiro) de cada elemento em cada fase do projeto. A fase de homologação o tempo foi alocado a 25%.

4.1.1 Fases do Estudo de Caso

A instanciação das diferentes fases do projeto XYZ é com recurso ao modelo SAN proposto neste trabalho, onde os integrantes da equipe são abstraídos como autômato elemento. No projeto XYZ as equipes tem no minimo 2 elementos e no maximo 5 elementos de acordo com a fase. Conforme ilustrado na Figura 4.1 o autômato elemento consiste de 4 estados, nomeadamente: Trabalhando (Tr), Parado (Pa), Colaborando (Co) e Retrabalhando. Os valores de j e i variam de 1

a 5 desconsiderando as situações em que j = i. Neste caso, o j corresponde ao elemento instanciado considerado no momento e o i corresponde aos restantes elementos que se sincronizam com o j nas condições que são bem detalhadas no capitulo 5 onde se descrevem os eventos do modelo SAN.

Esses estados representam as possíveis atividades de cada elemento em cada dia de trabalho. É importante ressalvar que neste trabalho pretende-se avaliar a produtividade das equipes que é determinado com base nas taxas do estado Trabalhando (Tr).

4.1.1.1 Especificação dos Requisitos

Esta fase foi realizada por 1 analista de sistemas sênior e 1 analista de sistemas júnior e decorreu de 11/05/2011 a 07/06/2011 que é um período equivalente a 21 dias. Nesta fase foram realizadas tarefas referentes ao levantamento e detalhamento de requisitos, construção de modelo de dados lógico, especificação de casos de uso. A Figura 4.2 representa a instanciação da fase de requisitos com base no modelo ilustrado na Figura 4.1.

Tabela 4.1 – Elementos da fase de especificação dos requisitos

Papel do elemento	Alocação de tempo
Analista de sistemas sênior	100%
Analista de sistemas júnior	100%

O conceito de requisito, na Ciência de Computação, Engenharia de Software e Engenharia de Sistemas, é utilizado para referir-se à definição de uma característica, atributo, habilidade ou qualidade que um sistema (ou qualquer um de seus módulos e sub-rotinas) deve necessariamente prover para ser útil a seus usuários.

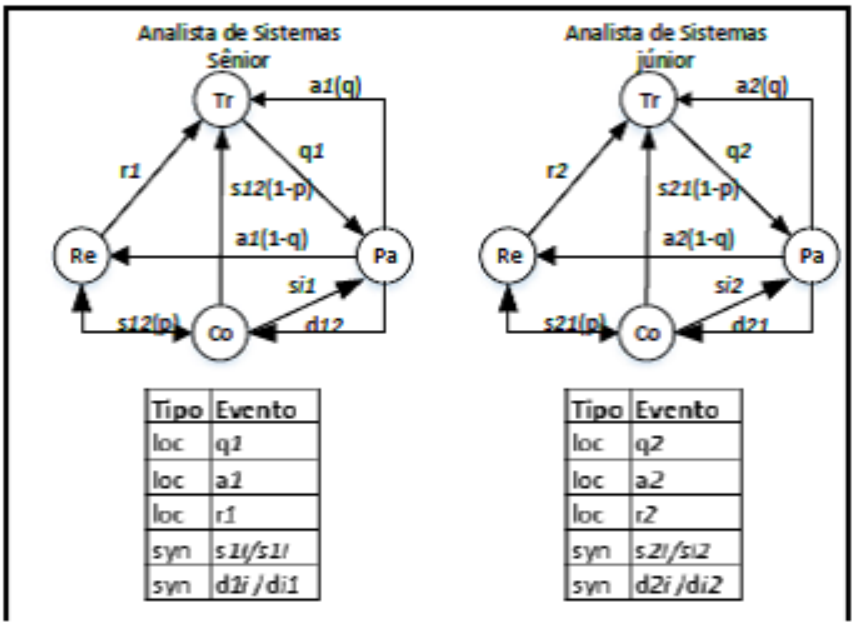


Figura 4.2 – Instanciação da fase de requisitos

4.1.1.2 Fase de Prototipagem

Esta fase foi realizada pelos 2 analistas de sistemas num intervalo que vai de 09/06/2011 a 17/06/2011, um período que corresponde a 7 dias uteis. Esta fase responsabilizou-se pela prototipação de interfaces. A Figura 4.3 representa a instanciação da fase de prototipagem com base no modelo ilustrado na Figura 4.1.

Tabela 4.2 – Elementos da fase de prototipagem

Papel do elemento	Alocação de tempo
Analista de sistemas sênior	100%
Analista de sistemas júnior	100%

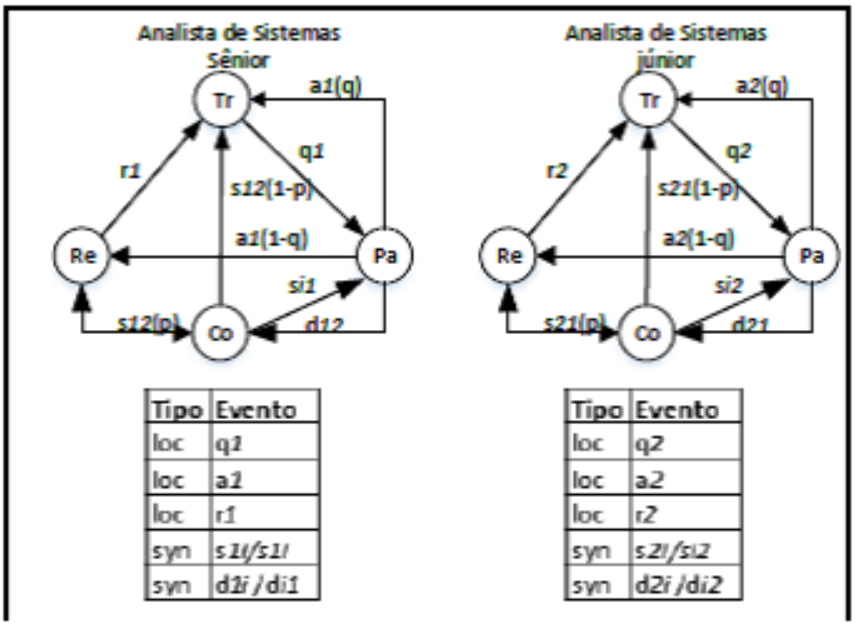


Figura 4.3 – Instanciação da fase de prototipagem

A prototipagem é a atividade de desenvolvimento de uma versão inicial do sistema baseada no atendimento dos requisitos ainda pouco definidos, permitindo a descoberta de falhas difíceis de serem encontradas na comunicação verbal. Um processo que propõe a criação de um protótipo de software tem o objetivo de apoiar a fase de levantamento de requisitos de modo a prevenir as possíveis falhas no sistema em fases posteriores.

4.1.1.3 Especificação de Programas

Esta fase do projeto foi realizada por 2 analistas de sistemas e 1 analista de testes. Ela foi levada a cabo de 20/06/2011 a 11/07/2011, que é um período que corresponde a 16 dias. A principal responsabilidade desta fase foi a construção das especificações técnicas, dos diagramas UML (classes, sequência, estados, etc) e também foi responsável pela construção dos casos de

Tabela 4.3 – Elementos da fase de especificação de programas

Papel do elemento	Alocação de tempo
Analista de sistemas sênior	100%
Analista de sistemas júnior	100%
Analista de testes pleno	100%

testes. A Figura 4.4 representa a instanciação da fase de especificação de programas com base no modelo ilustrado na Figura 4.1.

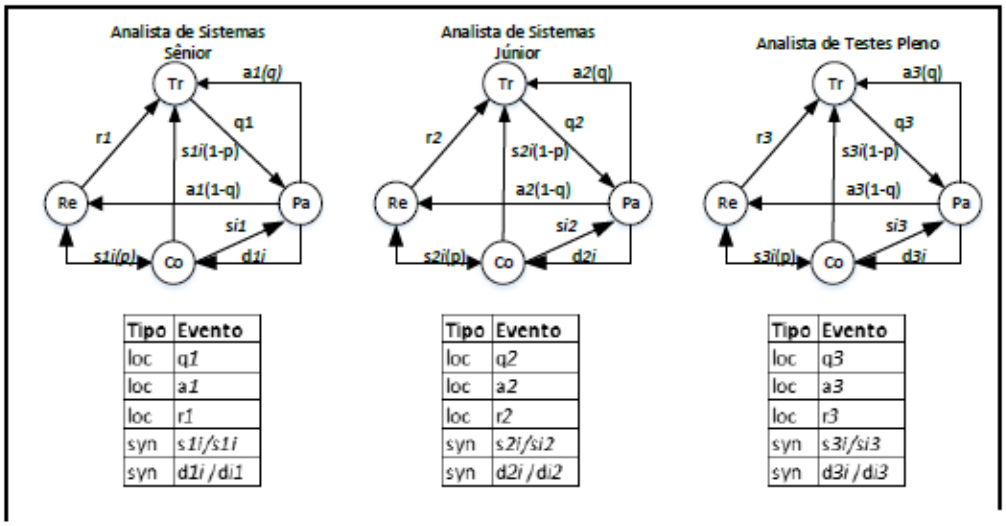


Figura 4.4 – Instanciação da fase de especificação de programas

4.1.1.4 Fase de Desenvolvimento

Tabela 4.4 – Elementos da fase de desenvolvimento

Papel do elemento	Alocação de tempo
Desenvolvedor sênior	100%
Desenvolvedor pleno	100%
Desenvolvedor júnior	100%
Tester sênior	100%
Tester júnior	100%

Esta é a fase mais longa de todas fases neste estudo de caso especifico. Realizada por uma equipe formada por 3 desenvolvedores e 2 testers. Foi realizada num período que foi de 27/06/2011 a 16/09/2011, que é um período que corresponde a 76 dias. Nesta fase foi construído o código e também foram feitos testes unitários. A Figura 4.5 representa a instanciação da fase de desenvolvimento.

A fase de desenvolvimento em algumas literaturas é designada por fase de implementação e consiste, como se afirmou de uma forma breve no paragrafo anterior, na transformação de um

projeto para um código deve e é, geralmente, a parte mais evidente do trabalho da engenharia de software, mas não necessariamente a sua maior porção. Coincidentemente no estudo de caso considerado para este trabalho a fase de desenvolvimento consome a maior porcão do projeto.

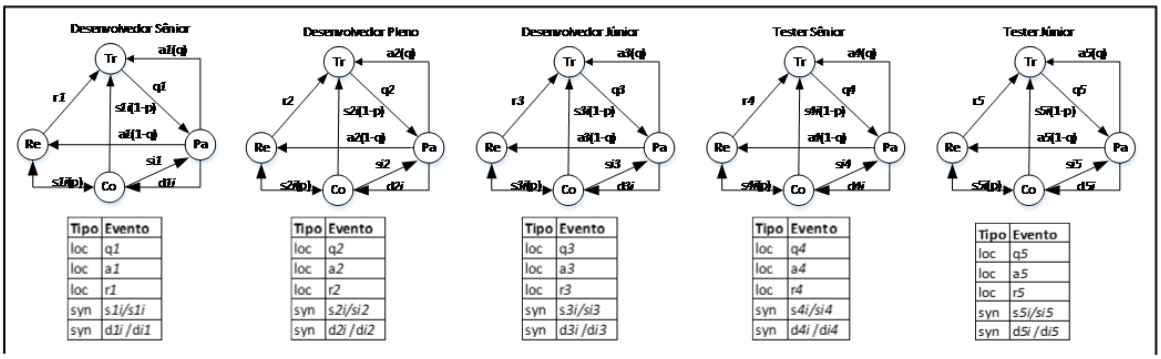


Figura 4.5 – Instanciação da fase de desenvolvimento

4.1.1.5 Fase de Documentação

Realizada por 2 analistas de sistemas e 1 tester. De 30/08/2011 a 14/10/2011 – 19 dias. Esta fase encarregou-se pela elaboração da documentação do usuário e help online do portal. A Figura 4.5 representa a instanciação da fase de documentação.

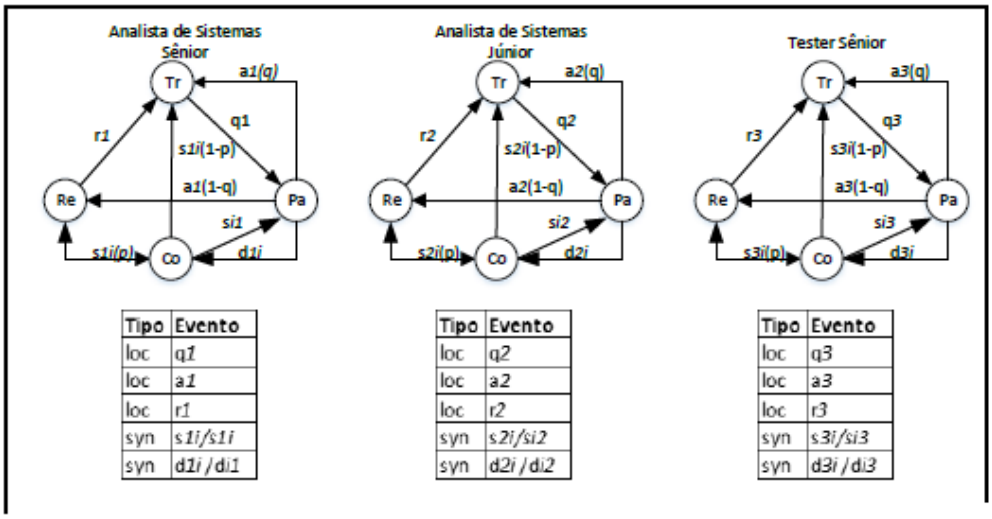
Tabela 4.5 – Elementos da fase de Documentação

Papel do elemento	Alocação de tempo
Analista de sistemas sênior	100%
Analista de sistemas júnior	100%
Tester sênior	100%

A documentação de software ou, em alguns bibliografias, documentação do código fonte, é um texto escrito que acompanha o software e geralmente explica como utilizá-lo. O termo pode significar coisas diferentes para pessoas de várias funções. Os textos podem acompanhar uma função, uma classe, um simples trecho ou módulo, ou podem consistir de num conjunto de manuais gerais e técnicos, além de diagramas explicando o funcionamento do programa como um todo ou cada parte dele.

4.1.1.6 Homologação

Realizada pela equipe do cliente de 14/09/2011 a 24/10/2011 – 28 dias. Atividades de homologação do sistema no ambiente do cliente. Durante a homologação a equipe fica alocada a tempo parcial (25%) para suporte necessário foram alocados 1 analista de sistemas e 1 analistas de testes e para correção de bugs encontrados pelo cliente foi alocado 1 desenvolvedor. A Figura 4.6 representa a instanciação da fase homologação.



Tipo	Evento
loc	q1
loc	a1
loc	r1
syn	s1i/si1
syn	d1i / di1

Tipo	Evento
loc	q2
loc	a2
loc	r2
syn	s2i/si2
syn	d2i / di2

Tipo	Evento
loc	q3
loc	a3
loc	r3
syn	s3i/si3
syn	d3i / di3

Figura 4.6 – Instanciação da fase de documentação

Tabela 4.6 – Elementos da fase de Homologação

Papel do elemento	Alocação de tempo
Analista de sistemas sênior	25%
Analista de testes pleno	25%
Desenvolvedor sênior	25%

Um sistema operacional homologado para servidores nada mais é que um software que foi testado em determinadas condições e recebeu um selo ou aval do fabricante de hardware (ou servidor) que funciona sem problemas. As empresas que homologam o servidor junto aos sistemas operacionais buscam garantir que as tarefas demandadas serão executadas. Caso um servidor tenha um sistema operacional não homologado, o usuário (ou administrador de rede) não terá qualquer tipo de suporte dos fornecedores do servidor. Os benefícios criados pela homologação de softwares visam o pagamento do tributo ao governo unicamente.

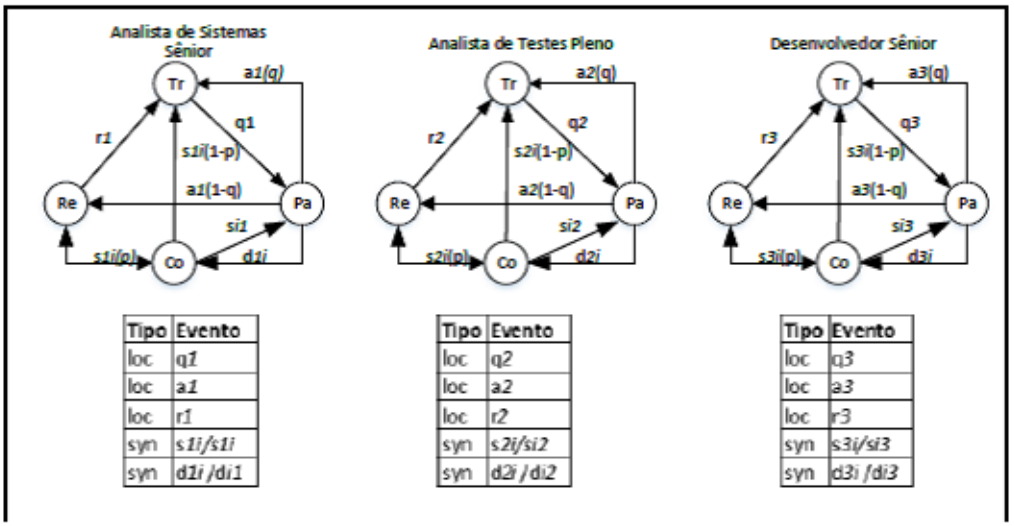


Figura 4.7 – Instanciação da fase de homologação

5. MODELO SAN

Esta seção apresenta um modelo analítico de uma equipe de desenvolvimento de softwares onde os seus elementos estão localizados em um único local. Corresponde a um ambiente auto-suficiente, modelando os elementos que assumem diferentes papéis em um projeto de software baseado no modelo de processos Waterfall, tais como desenvolvedor, analista de sistemas, tester e analista de testes que são os papeis que constituem as equipes do estudo de caso considerado neste trabalho. Os papeis são classificados em função do nível de experiência. Para instanciar o modelo usa-se o dados do estudo de caso descrito com mais detalhes no Capitulo 4.

A abstração do modelo consiste de uma entidade principal designada elemento que é representada no modelo SAN por um autômato elemento com quatro estados nomeadamente: Trabalhando (Tr), Parado (Pa), Colaborando (Co) e Retrabalhando (Re). O número de elementos suportados pelo autômato vai de j=1 até n. O valor de n podendo assumir no máximo 10 porque aconselha-se que numero de elementos de uma equipe de desenvolvimento não seja maior que 10 participantes [11, 47]. A Figura 5.1 ilustra o padrão de iterações entre os elementos da equipe de cada fase do estudo de caso XYZ

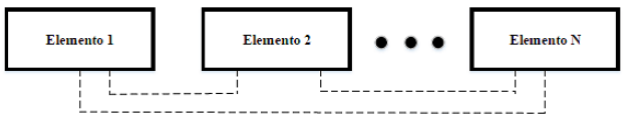


Figura 5.1 – Padrão de iterações do equipe

Assumindo que determinado elemento da equipe está no estado trabalhando (Tr) pode transitar para o estado (Pa) quando precisa de mais informações ou ajuda relacionada com suas tarefas. No estado (Pa) o elemento pode estar não necessariamente aguardando por uma colaboração. Pode estar parado por não conseguir seguir com suas próprias tarefas por algum tipo de impedimento. Ou ainda o estado (Pa) pode significar disponibilidade para colaborar. Para transitar para o estado (Co) de modo a colaborar com outro elemento, os elementos devem sincronizar de modo a permitir comunicação (que pode ser uma colaboração de dar ou receber ajuda ou ainda outro tipo de colaboração). Considera-se uma colaboração horizontal uma vez que se considera que os elementos da equipe são os principais responsáveis pelas suas tarefas e pelo sucesso do projeto e não possuem um gerente que fornece suporte técnico em cada fase do desenvolvimento Waterfall.

Nos casos em que se aplica o modelo de processos de desenvolvimento Waterfall em que as equipes envolvidas não possuem um líder as tarefas são fortemente dependentes da experiência e decisão dos elementos da equipe daí a necessidade de comunicação (colaboração) entre eles (é necessário que o elemento esteja disponível para colaborar para melhor desempenho da equipe). Depois da colaboração (Co) o elemento pode continuar trabalhando (Tr) ou perceber que algumas partes das suas tarefas precisam de melhorias ou por exemplo algum bug pode ter ocorrido e necessita de uma correção, neste caso, o elemento transita para o estado retrabalho (Re) para melhorias e

correções em suas tarefas, só então começa a trabalhar novamente (Tr), repetindo este processo de forma iterativa. Se se tratar dum elemento que estava no estado (Pa) e transitou para (Co), depois da colaboração pode retomar para o estado (Pa) continuando disponível para outras colaborações ou continuando focado noutro assunto que lhe levou ao estado (Pa).

A abstração dos elementos da equipe é ilustrada na Figura 5.2 que consiste de um modelo SAN com apenas um autômato que pode gerar N elementos em função do tamanho da equipe. De salientar que os valores dos índices variam de j = 1..N e i = 1 N, onde para cada instanciação exclui-se o caso em que j = i, que representa uma situação em que um elemento colabora consigo mesmo o que na prática não faz sentido. Estes índices ajudam a fazer a instanciação e o controle das sincronizações entre os elementos das equipes de cada fase do projeto XYZ. O j representa o elemento instanciado e o elemento i representa os restantes elementos da equipe que em determinadas condições sincronizam-se com o j.

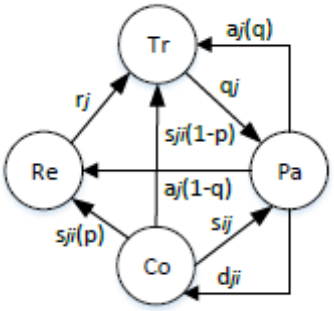


Figura 5.2 – Modelo SAN

Na abstração do modelo assume-se que todos os elementos da equipe tem as tarefas igualmente distribuídas sem levar em conta o nível de detalhes e de complexidade de cada tarefa. Usando a entidade representada na abstração obtida na modelagem, pode-se construir um modelo analítico representando um projeto de desenvolvimento de software com base no modelo de processos Waterfall com N participantes que colaboram em cada fase do projeto do software. Neste estudo apresenta-se uma instanciação de um estudo de caso de um projeto de desenvolvimento de software real no qual a equipe de desenvolvimento usou o modelo de processos Waterfall para condução dos seus processos no desenvolvimento.

5.1 Estados do Modelo

- Trabalhando (Tr): Inclui a execução de todas as tarefas relacionadas com o papel de cada elemento da equipe alocada para cada fase do projeto Waterfall de acordo com o cronograma do projeto.
- Parado (Pa): Representa o estado em que o elemento da equipe precisa esperar para cooperar com outros elementos da equipe. Pode estar parado por não conseguir prosseguir com suas

tarefas ou então, pode estar parado nas suas tarefas para pesquisar de modo a ultrapassar alguma dificuldade. Este estado também pode significar que o elemento está disponível para colaborar.

- Colaborando (Co): Neste estado o elemento da equipe se encontra colaborando sincronicamente com outro membro da equipa de modo a resolver determinada questão relativa a fase em que se encontram (dando ou recebendo ajuda).

- Retrabalhando (Re): Consiste na revisão de requisitos, correções de bugs, reescrita do código e as revisões de projeto conforme a fase em que se encontra que pode ser resultado duma colaboração ou fruto de uma pesquisa individual.

  A Tabela 4.1 que se segue ilustra de uma forma mais breve e sucinta a descrição de cada estado do modelo.

Tabela 5.1 – Nomes e descrição dos estados

Estado	Abstração
(Tr)	Trabalhando: Execução das tarefas de acordo com o papel de cada elemento.
(Pa)	Parado: Pausa para solucionar impedimentos através de pesquisa individual ou esperar colaboração.
(Co)	Colaborando: Colaboração sincronizada entre os elementos.
(Re)	Retrabalhando: Consiste no retrabalho apos uma colaboração ou pesquisa.

5.2 Eventos do Modelo

O modelo Waterfall proposto possui seis eventos (d*ji*, r*j*, s*ji*, s*ij*, q*j* e a*ji*). Para representar a transição do elemento ir ao estado (Pa) na busca de solução para determinada questão ou para esperar uma colaboração temos o evento q*j*. O evento s*ji* corresponde ao ato de colaborar na resolução das questões que surgem ao longo das tarefas dos elementos da equipe, caso especifico de quem precisou de ajuda. O evento s*ij* corresponde ao regresso (Pa) depois de fornecer ajuda a outros elementos. A transição dos elementos da equipe para colaborar com outros na busca de respostas para questões pertinentes é representada pelo evento d*ji* representando disponibilidade do elemento *i* para ajudar ou receber ajudar do elemento j) e o evento representando transição do elemento do estado retrabalhando (Re) para o estado trabalho é o evento r*j*. O evento a*ji* representa a transição de determinado elemento do estado (Pa) para o estado (Re) ou o regresso para o estado (Tr) sem ter precisado de colaborar com outros elementos da equipe. Cada mudança de estado e interações entre os elementos da equipe acontecem devido à ocorrência destes eventos. A Tabela 5.2 apresenta os eventos e suas respetivas descrições de um modo mais resumidas.

Tabela 5.2 – Descrição dos eventos do modelo SAN

Evento	Descrição
q*j*	Expressa a transição do elemento *j* do estado trabalhando (Tr) para o estado parado (Pa)
s*ji*	Representa o ato de colaboração do elemento *j* com o elemento *i* (elemento *j* recebendo ajuda do elemento *i*).
s*ij*	Representa o ato de colaboração do elemento *i* com o elemento *j* (elemento *i* ajudando o elemento *j*).
d*ji*	Representa o ato do elemento *j* disponibilizar-se para colaborar com *i* e vice -versa.
a*j*	Representa o retorno do elemento *j* para o trabalho (Tr) ou ida para retrabalho (Re) saindo do estado parado (Pa).
r*j*	Consiste na transição do elemento *j* do estado retrabalhando (Re) retornando para o estado trabalho (Tr).

5.3 Definição de Parâmetros

A definição de parâmetros é em conformidade com Czekster *et al* [11] que propôs modelos para 2 contextos (locais geograficamente distribuídos e equipes localizadas no mesmo espaço geográfico). O estudo de caso do presente trabalho se enquadra no contexto único local. Segundo Czekster *et al* [11] descrever apenas as entidades e seus relacionamentos em um modelo é insuficiente, faltando as durações que cada entidade se apoia em um determinado estado. Estas informações são também denominados como parâmetros do modelo e são fundamentais na modelagem analítica, devido à sua expressividade. O estudo do paper referenciado baseou-se nos resultados de experiências empíricas onde o grupo autor do estudo fez atribuições das durações de cada estado no modelo.

No presente estudo é seguida a mesma abordagem do Czekster *et al* [11] mas aplicado a um estudo de caso concreto referente a um projeto de software que se baseia no modelo de processo prescritivo Waterfall e que as equipes estão localizadas no mesmo espaço geográfico. Os resultados da modelagem (tempos calculados) são comparados com os tempos observados e os tempos estimados de modo avaliar até que ponto o modelo SAN poderá contribuir na previsão de prazos de projetos, neste caso especifico, em projetos com base no modelo de processos Waterfall que tem uma característica básica que consiste nas fases do projeto serem bem distintas (uma fase só inicia com o termino doutra) o que necessita uma boa pre-definição de prazos para o sucesso do projeto e minimização de riscos.

As taxas dos eventos são determinadas com base na razão entre a frequência de ocorrência de cada evento (X*i*) e a carga horária diária de trabalho considerada para cada dia de trabalho (Z), ou seja, conforme a função (i).

- Taxa do evento = (X*i*)/Z (i)
- Taxa do evento = ((X*i*)*(k%))/Z (ii)

Para os casos em que os elementos não são alocados a 100% da carga horária diária inclui-se a percentagem de tempo alocada (k%) na formula que fica conforme a função (ii). Na forma geral (i) está implicitamente multiplicado por 100%/1. A Tabela 5.3 apresenta a instanciação das taxas dos eventos locais com base na formula apresentada neste paragrafo, onde AS = analista de sistemas, Dev = desenvolvedor, AT = analista de testes e Tst = Tester. De salientar que foram instanciados apenas os eventos locais das fases com carga horária diária de 9 horas a titulo de exemplo do uso da aplicação das funções (i) e (ii).

A Tabela 5.3 ilustra as taxas dos eventos locais de cada elemento em função do seu nível de experiência. A carga horária diária das fases do estudo de caso foi de 9 horas de carga horária por cada dia de trabalho, com exceção da fase de desenvolvimento que gastava 12 horas a cada dia de trabalho e a fase de homologação que foi alocado apenas 25% das 9 horas onde se aplica a função (ii) para a determinação tas taxas. De salientar que a carga horária da fase desenvolvimento de 12 horas foi apenas usada porque corresponde aos dados quantitativos do estudo de caso identificado para este estudo. Não houve um questionamento desta carga horária. Contudo, de acordo com conversas formais com peritos em gestão de projetos da software, a carga horária de 12 horas não é aplicável. Podendo ter sido considerado no projeto XYZ por alguma razão que não foi questionada neste trabalho.

Tabela 5.3 – Taxas dos eventos locais do estudo de caso

Evento	Expertise	AS	Dev	AT	Tst
qj	Sênior	6/9	6/9	6/9	6/9
	Pleno	5.7/9	5.7/9	5.7/9	5.7/9
	Júnior	2/9	2/9	2/9	2/9
aj	Sênior	1/9	1/9	1/9	1/9
	Pleno	3/9	3/9	3/9	3/9
	Júnior	6/9	6/9	6/9	6/9
rj	Sênior	1/9	1/9	1/9	1/9
	Pleno	1/9	1/9	1/9	1/9
	Júnior	1/9	1/9	1/9	1/9

Para determinar as taxas dos eventos sincronizantes recorre-se as seguintes funções:

- sji = sj +(1 − si) (iii)
- sij = si +(1 − sj) (iv)
- dji = dj +(1 − di) (v)

Nestas funções define-se de forma aleatória que o mais provável é o elemento mais experiente dar ajuda ao menos experiente e não o contrário. Este parâmetro carece de um estudo mais aprofundado de modo a definir de forma mais acertada como ocorre a colaboração entre os elementos das equipes. Na verdade as funções (iii), (iv) e (v) podem se resumir em apenas uma, mas se mantem 3 de modo a deixar mais claro e de acordo com a designação dos eventos. Dada

a definição aleatória deste parâmetro, ele fica flexível para uma definição mais exata, em estudos futuros para definição do critério de colaboração. O nível de experiência foi definido em função da taxa de produtividade (Tr) onde o elemento mais experiente leva mais tempo trabalhando comparado com o elementos com níveis de experiência mais baixos. O valor 1 presente na função serve para impedir que ocorram números negativos dado que o 1 representa o valor máximo das taxas que podem variar de 0 a 1.

Para as fases de especificação de requisitos, prototipagem, especificação de programas e documentação as taxas de produtividade são: sênior ($6/9 = 0.667$), pleno ($4/9 = 0.444$) e júnior ($2/9 = 0.222$). De acordo com estas taxas de produtividade de cada elemento define-se a matriz de taxas de eventos sincronizantes ilustrada na Tabela 5.4.

Tabela 5.4 – Matriz de taxas dos eventos sincronizantes das 3 primeiras fases

	Júnior	Pleno	Sênior
Júnior	1	0.778	0.555
Pleno	1.222	1	0.777
Sênior	1.445	1.223	1

Para o caso da fase de desenvolvimento que possui uma carga horária diária de 12 horas as taxas de produtividade são: sênior ($8/12 = 0.667$), pleno ($5/12 = 0.417$) e júnior ($3/12 = 0.25$). De acordo com estas taxas de produtividade define-se a matriz de taxas para a fase de desenvolvimento que se seguem na Tabela 5.5.

Tabela 5.5 – Matriz de taxas para os eventos sincronizantes para o desenvolvimento

	Júnior	Pleno	Sênior
Júnior	1	0.833	0.583
Pleno	1.167	1	0.75
Sênior	1.417	1.25	1

Na fase de homologação, em que os participantes foram alocados a 25% das 9 horas de carga horária diária, as taxas de produtividade são: sênior ($(6*0.25)/9 = 0.167$), pleno ($(4*0.25)/9 = 0.111$). Nesta fase não participa algum elemento júnior. De acordo com estas taxas define-se a matriz de taxas para a fase de homologação ilustrada na Tabela 5.6.

Tabela 5.6 – Matriz de taxas para os eventos sincronizantes na homologação

	Pleno	Sênior
Pleno	1	0.963
Sênior	1.056	1

As taxas determinadas nas 3 matrizes representam a disponibilidade para sair do estado Parado (Pa) para Colaborando (Co) de acordo com a respetiva fase do projeto. Representa também

as taxas de saída do estado colaborando (Co) para Retrabalhando (Re) no caso de quem esteve a receber ajuda e o regresso para o estado Parado (Pa) para o caso de quem estava ajudando.

5.4 Medição de Produtividade

De acordo com Czekster *et al* [11] uma forma eficaz de medir a produtividade é de se concentrar em tarefas de projeto bem sucedidos. Existem diversos fatores que afetam a produtividade, tais como o cronograma de tarefas previsto contra cronograma real, aumento das interações devido a marcos e equívocos nas tarefas ou mudanças nos requisitos do projeto. A medida da produtividade da equipe também é influenciado pela moral dos desenvolvedores, como eles devem se sentir constantemente que o projeto está progredindo sem problemas [11]. No modelo desenvolvido a produtividade é determinada com base no tempo em que os elementos da equipe estão, de fato, trabalhando na produção de algum output do projeto através do estado designado por working (Wk). No presente estudo pretende-se determinar a produtividade da equipe com base no mesmo evento wk (no modelo proposto neste estudo designado de (Tr)) mas aplicado aos dados de um caso de estudo XYZ e efetuando algumas variações do nível de experiência e o número de elementos de modo a constatar a sua influência na produtividade da equipe do projeto em cada fase específica.

6. EXECUÇÃO DO MODELO SAN

Czekster *et al* [11] apresentou em pesquisas recentes, modelos estocásticos para o projeto de desenvolvimento de software, a fim de avaliar a comunicação e disponibilidade de equipes de desenvolvimento de software globais (GSD). Ele considerou dois contextos (único local e multi local). Na sequência desta abordagem, foi criado um modelo estocástico de base para a avaliação de projetos FTS [16, 31]. De acordo com as descrições no paper [31] FTS se enquadra no contexto multi local. Segundo Santos [31], no mesmo contexto das pesquisas do Czekster *et al* [11], afirma que ainda existem oportunidades de capturar outros aspetos do GSD usando configurações FTS, tais como a eficiência de handoff, controle das fases do projeto, capacidade das equipes, dentre outros aspetos inerentes ao GSD. Na sequência dessas oportunidades, Santos [31] identificou e catalogou aspetos que podem influenciar o processo de tomada de decisão nos projetos Follw-The-Sun. Na sua pesquisa a questão do trabalho do Santos [31] era de compreender como qualificar os processos de tomada de decisão nos projetos FTS, avaliando fatores como, custo, tempo e qualidade.

Na mesma onda de oportunidades o presente trabalho concebe um modelo SAN de contexto único local, no mesmo contexto do modelo proposto por Czekster *et al* [11], para avaliar o desempenho de uma equipe que usa o modelo de processos Waterfall para desenvolvimento do projeto de software. É muito comum em grandes projetos de software, apesar da tendência atual de se optar por modelos ágeis, adotar-se o modelo de processos Waterfall para o desenvolvimento do projeto. O modelo de processos Waterfall permite que o projeto seja feito em fases bem distintas e isso permite que o projeto seja contratado em fases, podendo até serem diferentes empresas a implementar as distintas fases do projeto. Na contratação de projetos de software em fases cada fase carece de um "Ok" do cliente, só depois da aceitação do cliente se prossegue com as fases subsequentes. A modelagem estocástica proposta pelo Czekster *et al* [11] mostrou, através de dados empíricos, até que ponto pode contribuir na mitigação de riscos, principalmente no que se concerne nos prazos de entrega, custos e qualidade. Contribuindo para o reforço do nível acurácia da previsão que é uma das principais características do modelo de processo Waterfall e que constitui o grande diferencial com os modelos ágeis que optam por processos adaptativos que são mais propensos a riscos.

Apesar de existiram várias características do projeto que podem ser considerados, dentre as quais, custo e qualidade que juntamente com o tempo foram bem abordadas no esforço de modelagem do trabalho [31]. O esforço de modelagem deste trabalho está concentrado numa das caraterísticas mais importantes do projeto que é o tempo. O mapeamento conceptual foca sua atenção para o fator tempo.

A avaliação de performance considera o número de elementos da equipe ou tamanho da equipe e a carga horária e ainda o nível de experiência de cada Entidade (sênior, pleno e júnior). O estudo de caso deste trabalho, como foi referenciado em seções anteriores, consiste de um projeto de um Portal do Concessionário de um banco que por razões de confidencialidade será designado por

XYZ. O modelo SAN pode ser usado como uma ferramenta para melhorar o controle de performance de equipes que usam o modelo de processos Waterfall para desenvolvimento de projetos de software.

Métricas de software podem ajudar no planejamento de gerenciamento de projetos, nesta fase é possível identificar a quantidade de esforço necessário e um custo estimado para completar um determinado projeto de software [31]. Apesar de no projeto de software haver custos de materiais, servidores e tantos outros, o recurso humano é, sem duvidas, um dos fatores principais na gestão de custos. Aliado ao fator recurso humano está a grande necessidade de gestão acertada do tempo de modo a garantir que o projeto atinja os objetivos pre definidos dentro dos prazos estabelecidos.

6.1 Análise Numérica

Os resultados quantitativos reais sobre Projeto XYZ foram montados a partir de compilação de arquivos com relatórios do projeto, entrevistas informais com os gestores, e uma pesquisa para coletar informações quantitativas sobre as interações entre os participantes durante as fases do projeto. Os resultados da análise do modelo foram obtidas usando dados do Projeto XYZ para instanciar as taxas de eventos (Seção 4.2).

6.1.1 Dados Quantitativos do Projeto XYZ

A Tabela 6.1 mostra o esforço estimado e horas atuais para cada fase do Projeto XYZ. É importante ressaltar que a duração de cada fase foi expressa em horas sendo para todas fase 9 horas de trabalho por dia para cada elemento com exceção da fase de desenvolvimento que foram 12 horas de trabalho por dia para cada elemento que constituiu a fase. Em todas as fases os recursos foram alocados em tempo inteiro (100%), exceptuando a fase de homologação que os recursos participantes foram alocados a tempo parcial (25%).

Tabela 6.1 – Duração estimada e duração observada das fases do Projeto XYZ

Fases do projeto	T. estimado (horas)	T. observado (horas)
Especificação dos requisitos	189	192.8
Prototipagem	63	61.5
Especificação de programas	144	138
Desenvolvimento	912	1069
Documentação	171	213
Homologação	63	51

6.1.2 Resultados Quantitativos do Modelo

Esta seção apresenta os resultados quantitativos obtidos a partir da solução numérica do modelo proposto, de acordo com os parâmetros definidos na seção 5.3 que foram fruto de entrevistas formais com profissionais de Engenharia de Software que trabalharam no Projeto XYZ e vários outros estudos, dentre eles, [1, 11, 31].

A solução numérica é expressa primariamente pela probabilidade de estado estacionário do modelo SAN. A partir destas probabilidades e das cargas horárias individuais atribuídas aos participantes do Projeto XYZ, pode-se determinar a média do horário de trabalho por 9 horas por dia de trabalho para as fase com essa carga horária e 12 horas por dia de trabalho para a fase de desenvolvimento. Os leitores interessados em obter mais informações sobre a ferramenta de software para a solução numérica de modelos SAN consulte PEPS (Performance Evaluation of Parallel Systems) [15].

Nos resultados quantitativos apenas são tratadas as probabilidades estacionárias do estado (Tr) de cada elemento que é o parâmetro definido para avaliação da performance dos elementos da equipe em cada fase do projeto.

Para melhor entendimento, neste trabalho, designa-se de tempo estimado ao tempo pre definido pelo perito de acordo com sua experiencia antes do inicio da execução de cada fase. Tempo observado é o tempo que realmente cada fase do projeto durou. E por fim o tempo calculado é o que resulta das probabilidades estacionárias do modelo SAN.

Nas fases em que temos apenas elementos com o mesmo papel e diferença de níveis de experiência considera-se de forma individual a produtividade de cada elemento em função do seu nível de experiência. A produtividade total será dada pela soma das individuais Nos casos em que, numa determinada fase, temos elementos com diferentes papeis e temos mais de um elemento com o mesmo a produtividade sera em função dos papeis, onde para cada papel faz-se a média aritmética das probabilidades estacionárias dos diferentes níveis de experiência. Por exemplo, na fase de especificação de programas temos 2 Analistas de sistema (1 júnior e 1 sênior) a probabilidade média estacionária dos analistas será dada por $(7.0 + 2.0)/2 = 4.5$. Vide a Tabela 6.7. A probabilidade estacionária total da fase será a soma das probabilidades médias calculadas em função dos papeis, para a fase tida como exemplo temos $4.5 + 40 = 8.5$.

6.1.2.1 Fase de Especificação de Requisitos

A fase de especificação de requisitos teve a duração de 21 dias úteis e uma carga horária diária de 9 horas. Observando os resultados apresentados na Tabela 6.2, correspondentes a fase de especificação de requisitos, é claramente possível notar que o analista de sistemas sênior, dada a sua experiência, realmente tem maior percentagem de produtividade comparado com o analista de sistemas júnior.

Tabela 6.2 – Probabilidades da especificação de requisitos

Entidade	Estado	Probabilidade (%)
Analista de sistemas sênior	Tr	77.9
	Pa	11.8
	Co	7.5
	Re	2.8
Analista de sistemas júnior	Tr	24.3
	Pa	65.6
	Co	7.5
	Re	2.6

O tempo calculado com base nas probabilidades estacionárias do modelo SAN para uma analista de sistemas sênior, ou seja, o tempo que realmente o analista de sistemas sênior gasta produzindo algo nesta fase do projeto (77.9% correspondentes a 7 horas por cada dia de trabalho) é maior que as 6 horas estimadas no projeto.

Para o caso do Analista de sistemas júnior obteve-se que o tempo que ele realmente gasta produzindo um output no projeto é de 24.3%, o equivalente a aproximadamente 2 horas (2.19 horas exatamente) em cada 9 horas de trabalho diário. Neste caso o valor está muito próximo da taxa de entrada do estado (Tr) que foi de exatamente 2 horas de permanência do elemento no estado.

Tabela 6.3 – Horas de trabalho da fase de especificação de requisitos

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Analista de sistemas sênior	100	77.9	7.0
Analista de Sistemas júnior	100	24.3	2.19
Soma das horas dos analistas de sistemas			9.19

A Tabela 6.3 mostra a carga horária diária de trabalho de cada elemento da fase de especificação de requisitos do projeto obtida como resultado da execução do modelo SAN. Em conformidade com a Tabela 6.1 o Projeto XYZ possui 192 horas de tempo observado, contrariamente as 189 horas de tempo estimado. Essa diferença corresponde a exatamente 2.48% de erro relativo. O gráfico da Figura 6.5 ajuda a fazer uma melhor comparação entre os 3 tempos, podendo-se notar que realmente, o tempo calculado está mais próximo do temo observado.

A Tabela 6.3 mostra que o modelo analítico instanciado para os parâmetros da fase de especificação de requisitos do Projeto XYZ fornece uma média 9.2 horas diárias de trabalho. Usando a base de 21 dias, que é o número de dias úteis para especificação de requisitos, obtém-se 192,99 horas de tempo calculado. Com base nestes resultados nota-se que o modelo SAN obteve um número de horas de trabalho muito próximo as horas de trabalho atuais gastas nesta fase.

O modelo SAN obteve um tempo calculado com um erro de 0.4% em relação ao tempo observado. O que significa que o tempo calculado está próximo do tempo observado comparado com o tempo estimado situação que torna o modelo proposto válido.

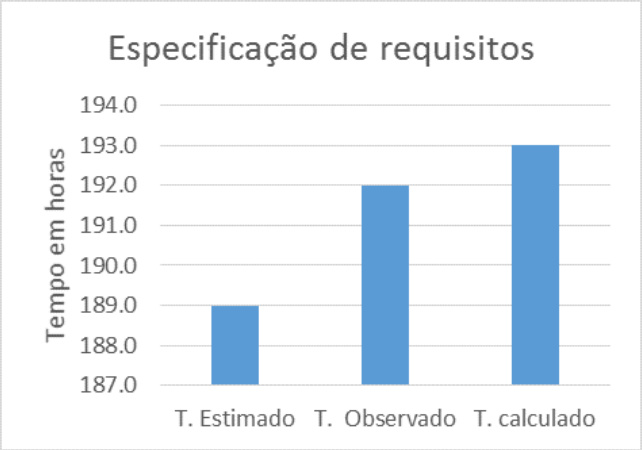


Figura 6.1 – gráfico dos tempos da especificação de requisitos

6.1.2.2 Fase de Prototipagem

A fase da prototipagem teve a duração de 7 dias úteis e uma carga horária diária de 9 horas. A Tabela 6.4 ilustra os resultados da execução do modelo proposto para as entidades da fase de prototipagem que por sinal as tarefas são realizados por elementos com os mesmos papeis e mesmo nível de experiência com os da fase de especificação de requisitos podendo não ser, necessariamente, os mesmos elementos.

Tabela 6.4 – Probabilidades da prototipagem

Entidade	Estado	Probabilidade (%)
Analista de sistemas sênior	Tr	77.8
	Pa	11.8
	Co	7.5
	Re	3.0
Analista de sistemas júnior	Tr	24.3
	Pa	65.6
	Co	7.5
	Re	2.6

O analista de sistemas sênior gasta 77.8% da carga horária diária de trabalho o que corresponde a 7 horas que está acima das 6 horas estimadas inicialmente. Contudo, o resultado da execução do modelo SAN realça o dado inicial que ilustra que o analista de sistemas sênior gasta a maior parte da carga horária de trabalho diária produzindo e não parado ou colaborando ou ainda retrabalhando.

O analista de sistemas júnior gasta pouco tempo trabalhando/produzindo dado o seu limitado nível de experiência, gastando muito tempo (65.6%) parado em busca de soluções com base em pesquisas individuais ou esperando para colaborar com o Analista sênior ou parado por qualquer outra limitação ou necessidade.

Tabela 6.5 – Horas de trabalho da fase de prototipagem

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Analista de sistemas sênior	100	77.8	7.0
Analista de sistemas júnior	100	24.3	2.2
Soma das horas dos analistas de sistemas			9.2

Observando os dados da Tabela 6.5 é possível afirmar que o analista de sistemas sênior possui maior capacidade de realizar suas tarefas sem muitos constrangimentos, gastando quase 80% do tempo alocado e contrariamente ao analista júnior que gasta pouco menos de 25%. Este fato pode ser observado em todas as fases do projeto independentemente do papel do elemento em análise.

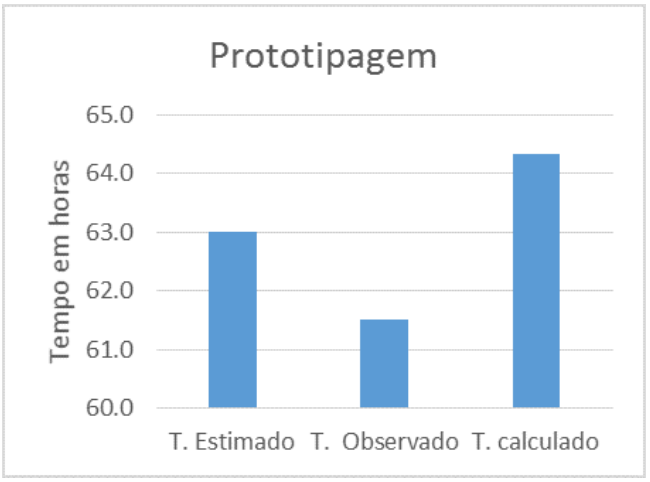


Figura 6.2 – gráfico dos tempos da prototipagem

A carga horária média diária da fase de prototipagem está ilustrada na Tabela 6.5 que é de 9.2 horas por cada dia . Nesta fase, conforme os dados da Tabela 6.5, o tempo calculado é de 64.4 horas que resulta do produto entre a produtividade total diária e número de dias que dura a fase de prototipagem. Fazendo uma comparação com os dados da Tabela 6.5 observa-se que o tempo calculado esta mais distante do tempo observado relativamente ao tempo estimado. Contudo, dada a distancia reduzida que se observa entre os 3 tempos torna o modelo fiável para previsão de prazos. O gráfico da Figura 6.2 ajuda a fazer uma rápida comparação entre o tempo estimado, tempo observado e tempo calculado.

6.1.2.3 Fase de Especificação de Programas

A fase de especificação de programas teve a duração de 16 dias úteis e uma carga horária diária de 9 horas. Os resultados obtidos na fase de especificação de programas são ilustrados na Tabela 6.6 e 5.7. Esta fase, diferentemente das duas analisadas anteriormente, foi realizada por 3 elementos nomeadamente: 1 analista de sistemas sênior, 1 analista de sistemas júnior e por fim 1 analista de testes pleno.

Tabela 6.6 – Probabilidades da especificação de programas

Entidade	Estado	Probabilidade (%)
Analista de sistemas sênior	Tr	76.9
	Pa	11.5
	Co	8.8
	Re	2.8
Analista de sistemas júnior	Tr	21.8
	Pa	51.6
	Co	24.2
	Re	2.4
Analista de testes pleno	Tr	44.8
	Pa	31.3
	Co	22.0
	Re	2.0

O analista de sistemas sênior, em semelhança das fases anteriores gasta a maior parte do tempo trabalhando que são exatamente 76.9% de acordo com as probabilidades do modelo SAN o que corresponde a 7.0 horas de trabalho contra 6 horas de taxa inicial estimada de acordo com a tabela da taxa dos eventos locais 4.3.

O Analista de testes pleno, dado o seu nível de experiência mediano (acima do nível de experiência do júnior), consome 44.8% da carga horária diária de trabalho o que corresponde a exatamente 4 horas de tempo de trabalho que são as mesmas horas definidas nas taxas dos eventos locais da Tabela 5.3.

O analista de sistemas sênior, dada a sua larga experiência, possui uma elevada autonomia de lidar com sua próprias tarefas consumindo maior parte do tempo da carga horária diária (76.9% correspondente a aproximadamente 7 horas de um total de 9 horas da carga horária diária de trabalho) a produzir algum output para a fase de especificação de programas gastando, deste modo, pouco tempo noutros 3 estados.

Tabela 6.7 – Horas de trabalho da fase de especificação de programas

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Analista de sistemas sênior	100	76.9	7.0
Analista de sistemas júnior	100	21.8	2.0
Média das horas dos analistas de sistemas			4.5
Analista de testes pleno	100	44.8	4.0

Na fase da especificação de programas o tempo estimado para o analista de sistemas sênior foi de 6 horas por cada 9 horas de carga horária diária de trabalho que é inferior ao tempo que se obtém com base na probabilidade do modelo SAN que é de 6.9 horas de trabalho.

O tempo calculado com base nas probabilidades do modelo SAN para o analista de sistemas júnior é 21.8%, equivalente a 1.96 horas (aproximadamente 2 horas) que esta muito próximo da taxa estimada para a produtividade (Tr) de um analista de sistemas júnior que foi de 2 horas para

cada dia de trabalho conforme ilustra a Tabela 5.3 do Capitulo 5 que apresenta a instanciação das taxas dos eventos locais . Para realçar as observações vide o gráfico da Figura 6.2.

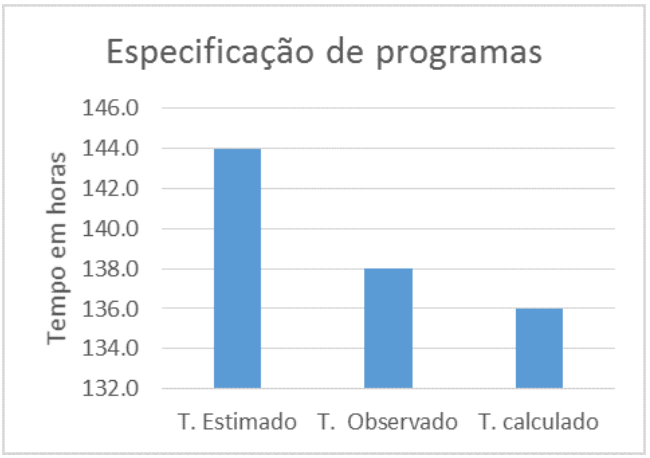


Figura 6.3 – gráfico dos tempos da especificação de programas

Observando os resultados representados na Tabela 6.7 pode-se determinar que o tempo calculado da fase de especificação de programas é 136 horas que se aproxima bastante do tempo observado de 138 horas comparado com o tempo estimado de 144 horas. O erro do tempo calculado em relação ao tempo observado é 1.44% que é menor que o erro do tempo observado em relação ao tempo estimado que é de 4.17%. Isso significa que, em conformidade com os resultados desta fase o modelo proposto oferece um nível de previsão aceitável.

6.1.2.4 Fase de Desenvolvimento

A fase de desenvolvimento teve a duração de 21 dias úteis e uma carga horária diária de 12 horas. Geralmente, de acordo com peritos na área de gestão de projeto, é pouco provável que a carga horária diária da fase de um projeto de software dure 12 horas. Não foi possível ter pormenores sobre as razões desta duração de 12 horas nesta fase especifica do Projeto XYZ. De modo a ter uma análise que vai ao encontro dos dados fornecidos este tempo não foi censurado neste trabalho mas, fica evidente que carece de mais pormenores para melhor tratamento. Esta carga horária foge do que realmente tem se verificado na prática.

Esta fase é caraterizada por possuir maior número de elementos e é a fase com maior duração de todas fases referentes ao Projeto XYZ. As tabelas 5.8 e 5.9 ilustram os resultados da execução do modelo proposto para as entidades da fase de desenvolvimento. Esta fase, conforme podemos ver na respetiva tabela, foi realizada por 5 elementos: 1 desenvolvedor sênior, 1 desenvolvedor pleno, 1 desenvolvedor júnior, 1 tester sênior e 1 tester júnior. Diferentemente das outras fases, esta fase tinha uma carga horária diária de 12 horas conforme já havia sido mencionado anteriormente.

Observando os dados na Tabela 6.8 pode-se determinar que o desenvolvedor sênior, conforme os resultados da execução do modelo SAN, gasta aproximadamente 9 horas (74.4% das 12 horas da carga horária diária) produzindo algo referente as tarefas desta fase, o que perfaz uma

Tabela 6.8 – Probabilidades de desenvolvimento

Entidade	Estado	Probabilidade (%)
Desenvolvedor sênior	Tr	74.4
	Pa	12.0
	Co	11.5
	Re	2.1
Desenvolvedor pleno	Tr	46.5
	Pa	25.5
	Co	23.9
	Re	4.1
Desenvolvedor júnior	Tr	28.6
	Pa	32.9
	Co	30.8
	Re	7.7
Tester sênior	Tr	73.5
	Pa	12.4
	Co	12.0
	Re	2.1
Tester júnior	Tr	24.2
	Pa	35.3
	Co	32.4
	Re	8.1

diferença 1 hora em relação a taxa prevista inicialmente que foi de 8 horas de tempo de trabalho para o desenvolvedor sênior conforme a Tabela 5.3.

Tabela 6.9 – Horas de trabalho da fase de desenvolvimento

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Desenvolvedor sênior	100	74.4	8.9
Desenvolvedor pleno	100	46.5	5.6
Desenvolvedor pleno	100	28.6	3.4
Média das horas dos desenvolvedores			5.97
Tester sênior	100	73.5	8.8
Tester júnior	100	24.2	2.9
Média das horas dos testers			5.85

Ainda na 6.8 pode-se notar que o desenvolvedor pleno gasta 5.7 horas de acordo com a probabilidade estaciona do modelo SAN contra 5 horas de taxa de ilustradas na tabela 5.3 da Seção 5.3 do Capitulo 5. Na mesma ordem, o desenvolvedor júnior, de acordo com a probabilidade da SAN, consome 3.4 horas diferentes da taxa estimada de 3 horas.

Apesar dessas diferenças mencionadas nos itens anteriores, observando a Tabela 6.9, o tempo calculado é de 898 horas e fazendo uma comparação, é um número aproximado ao tempo estimado de 912 horas. O tempo observado desta fase é de 1069 que esta muito distante do tempo calculado e do tempo estimado. O erro relativo entre o tempo calculado e o tempo observado é

de 14.6% que é uma diferença digna de muito questionamento que coloca em causa a validade do modelo. Mas, dado que esta carga horária desta fase levanta questionamento que carecem de aprofundamento, isso influencia nos resultados do modelo pelo fato de ser uma carga horária que foge do padrão dos projetos de software. Estes resultados deixam claro que realmente a carga horária de 12 horas afeta o processo de modelagem. Melhor tratamento desta fase carece de mais detalhes das tempos realmente gastos em cada um dos estados considerados e principalmente para o o estado (Tr) que serve de parâmetro para medição de produtividade. Os resultados para comparação estão patentes na Figura 6.4 que ilustra o gráfico de barras referente aos tempos em comparação. O tempo calculado se aproxima do tempo estimado pelo fato de as taxas terem sido inferidas com base nas taxas da carga horaria de 9 horas que é uma carga que não foge muito do padrão de 8 horas diárias de trabalho.

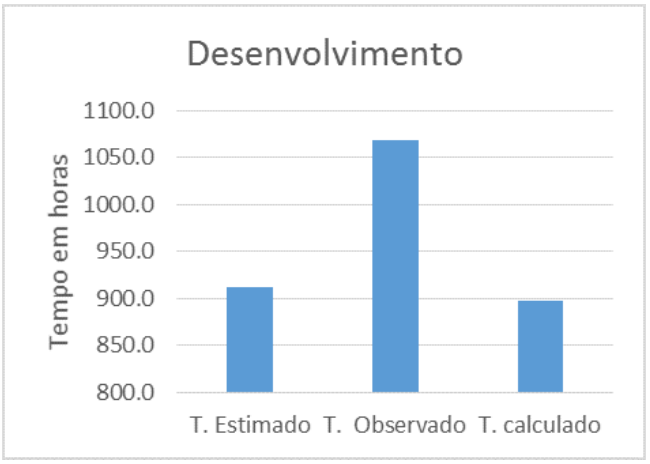


Figura 6.4 – gráfico dos tempos do desenvolvimento

Nesta fase, o analista de sistemas sênior e o tester sênior gastam acima de 70% da carga horária diária de trabalho o que mostra uma vez mais que os elementos seniores, independentemente do papel que desempenham, gastam mais tempo no estado Trabalhando (Tr) em detrimento dos 3 restantes estados.

6.1.2.5 Fase de Documentação

Com base nos dados apresentados na Tabela 6.10 verifica-se que o analista de sistemas sênior consome 6.96 horas (77.3% de 9 horas) trabalhando no sentido de produzir algum output referente a esta fase. Esse tempo, determinado com base nas probabilidades do modelo, está acima da taxa estimada inicialmente que foi de 6 horas de trabalho na fase de documentação para um elemento sênior.

Apesar dessa diferença, esse dado realça que o analista de sistemas sênior gasta mais tempo realizando as tarefas atribuídas, o que mostra a sua elevada capacidade de lidar com a suas próprias tarefas sem necessidade de colaboração dado o seu elevado nível de experiência.

A fase de documentação teve a duração de 19 dias uteis com uma carga horária diária de 9 horas. O analista de sistemas júnior consome 16.7% (1.5 horas) das 9 horas de carga horária a

Tabela 6.10 – Probabilidades da documentação

Entidade	Estado	Probabilidade (%)
Analista de sistemas sênior	Tr	77.3
	Pa	11.6
	Co	8.3
	Re	2.9
Analista de sistemas júnior	Tr	16.7
	Pa	60.5
	Co	15.8
	Re	7.0
Tester sênior	Tr	74.6
	Pa	13.9
	Co	10.5
	Re	1.0

cada dia de trabalho, ilustrando desse modo, que realmente, o analista de sistemas júnior possui limitadas capacidades de lidar com suas próprias tarefas.

Tabela 6.11 – Horas de trabalho da fase de documentação

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Analista de sistemas sênior	100	77.3	6.96
Analista de sistemas júnior	100	16.7	1.5
Média das horas dos analistas de sistemas			4.23
Analista de testes pleno	100	74.6	6.71

O tester sênior, devido ao seu elevado nível de experiência, também consome mais tempo (74.6%) produzindo algum output referente a fase e, deste modo, consumindo menos tempo nos estados colaborando (Co), parado (Pa) e retrabalho (Re).

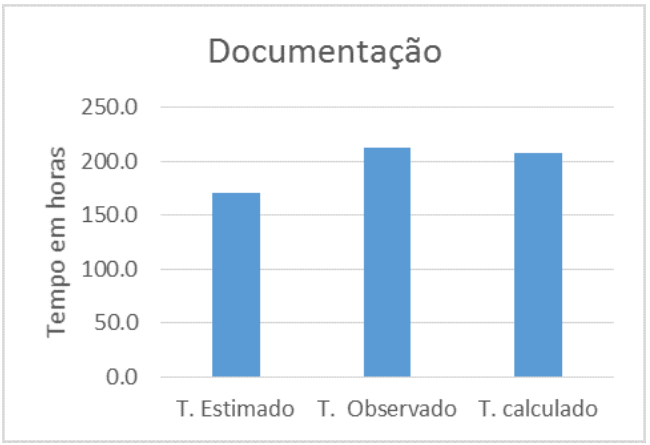


Figura 6.5 – gráfico dos tempos da documentação

Apesar das diferenças observadas nas taxas acima e observando a Tabela 6.11 constata-se que o tempo estimada de 171 horas esta muito abaixo do tempo calculado que é 207.9. O tempo

calculado, por sua vez, possui um valor muito próximo do tempo observado de 213 horas o que torna o modelo proposto neste estudo digno de atenção para geração de índices de desempenho das equipes de projetos de software baseados no modelo Waterfall. Vide o gráfico da Figura 6.4 que apresenta o gráfico que permite fazer uma comparação visual do cenário concluído com base no número de horas.

6.1.2.6 Fase de Homologação

Tabela 6.12 – Probabilidades da Homologação

Entidade	Estado	Probabilidade (%)
Analista de sistemas sênior	Tr	24.0
	Pa	34.4
	Co	2.9
	Re	40.0
Analista de testes pleno	Tr	36.0
	Pa	18.8
	Co	2.4
	Re	42.8
Desenvolvedor sênior	Tr	23.2
	Pa	27.7
	Co	2.8
	Re	46.3

Em todas as fases anteriores do projeto os recursos usados foram alocados a 100% com exceção da fase de homologação em que os elementos que participaram foram alocados a 25% das 9 horas correspondentes a carga horária diária de trabalho, o que corresponde a apenas 2.25 horas (2 horas e 15 minutos) a cada 9 horas diárias de trabalho.

Nesta fase, o analista de sistemas sênior, com base na probabilidade do modelo, gasta 0.54 horas que equivale a 24.0% das 2.25 horas que é a carga horária diária para a presente fase. Para o desenvolvedor sênior temos, com base na probabilidade estacionária do modelo, 23.2% das 2.25 horas que resultam em 0.52 horas. O analista de testes gasta 36.0% equivalente a 0.81 horas. De recordar que para esta fase não se tem uma estimativa inicial das taxas, usa-se a percentagem de tempo alocado conforme descrito na secção de definição de parâmetros.

Tabela 6.13 – Horas de trabalho da fase de homologação

Entidade	Alocação do tempo (%)	Taxas de Tr(%)	Horas/dia
Analista de sistemas sênior	25	24.0	0.54
Analista de testes pleno	25	36.0	0.81
Desenvolvedor pleno	25	23.2	0.52

Com base nas probabilidades do modelo estima-se que a fase dure 1.87 horas fazendo o somatório dos tempos de trabalho individuais. Significa que, com base nos resultados do modelo

SAN, o tempo calculado é de 52.4 horas que está próximo do tempo observado de 51.0 horas que resulta num erro relativo de 2.7%. Uma vez que o tempo calculado encontra-se mais próximo do tempo observado do que o tempo estimado de 63 horas pode-se concluir que o modelo proposto neste trabalho, com base nos temos desta fase, oferece um nível de previsão da duração das fase projetos aceitável. Observe-se os resultados da no gráfico da Figura 6.6.

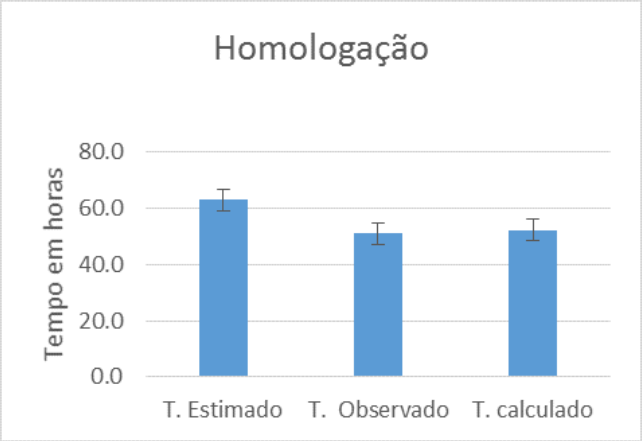


Figura 6.6 – gráfico dos tempos da homologação

6.1.2.7 Resultados Quantitativos das 5 Fases do Projeto XYZ

De acordo com os dados da Tabela 6.14 que possui uma ilustração dos tempos de todas as fases do Projeto XYZ nota-se de uma forma mais clara que na maioria dos casos temos os tempos calculados mais próximos dos tempos observados do que os tempos estimados. A Tabela 6.14 complementa a Tabela 6.1 que apresentou apenas os tempos estimado e calculado. Com base nos resultados da Tabela 6.15 nota-se que apenas a fase de desenvolvimento apresenta um tempo estimado mais próximo do real do que o tempo calculado. Contudo, a distância a favor do tempo estimado é bastante insignificante que não tira credibilidade e fiabilidade do tempo calculado com base no modelo SAN.

Tabela 6.14 – Tempo estimado, tempo observado e tempo calculado do Projeto XYZ

Fases do projeto	T. estimado (horas)	T. observado (horas)	T. calculado (horas)
Especificação dos requisitos	189.0	192.8	193.0
Prototipagem	63.0	61.5	64.4
Especificação de programas	144.0	138.0	136
Desenvolvimento	912.0	1069.0	898.0
Documentação	171.0	213.0	207.9
Homologação	63.0	51.0	52.4
TOTAL	1542.0	1724.5	1551.6

Ainda de acordo com a Tabela 6.14, observando concretamente a linha dos tempos totais, constatamos resultados que realçam o fato de que o modelo proposto oferecer um nível aceitável

de previsão dos prazos de um determinado projeto. O tempo total observado é de 1724.5 horas que tem uma diferença menor que o tempo total calculado (1551.6.82 horas) comparado com o tempo total estimado, apesar de ser uma diferença bem pequena. O modelo SAN garante maior qualidade de previsão nos casos em que os tempos calculados se aproximam dos tempos observados. As distancias entre os tempos estimado, observado e calculo são melhor visualizados e comparados na Tabela 6.15 onde são padronizados com base no tempo observado. A Figura 6.7 representa o gráfio dos resultados padronizados com base no tempo observado onde, também pode-se observar claramente a maior proximidade do tempo calculado em relação ao estimado na maior parte das fases, tomado como base o tempo observado. Olhando para o total nota-se que, praticamente, os 3 tempos são quase iguais. O valor 0.99 do total dos tempos estimados resulta da média aritmética dos valores padronizados das fases descritas na Tabela 6.15. O mesmo procedimento se aplica para os tempos observados e calculados.

Tabela 6.15 – Tempo estimado, tempo observado e tempo calculado padronizados

Fases do projeto	T. estimado	T. observado	T. calculado
Especificação dos requisitos	0.98	1.00	1.01
Prototipagem	1.02	1.00	1.05
Especificação de programas	1.04	1.00	0.99
Desenvolvimento	0.85	1.00	0.84
Documentação	0.80	1.00	0.98
Homologação	1.24	1.00	1.03
Média	0.99	1.00	0.98

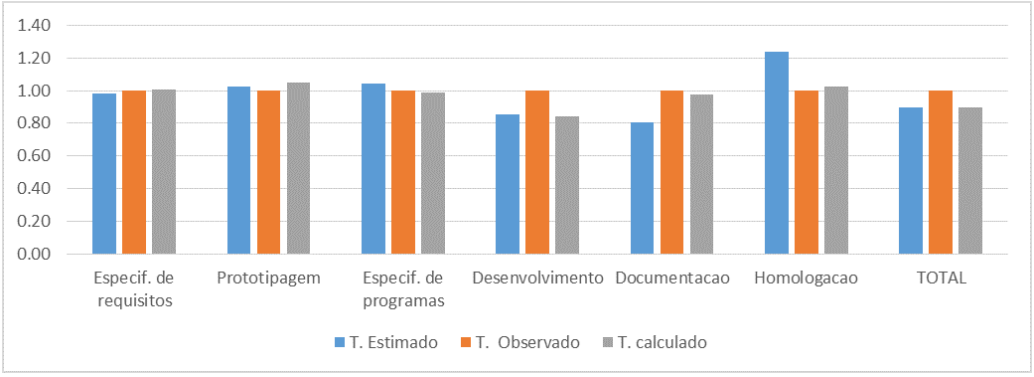


Figura 6.7 – gráfico dos tempos padronizados

6.1.3 Análise de Diferentes Cenários

Na secção anterior foi considerado uma instanciação de um estudo de caso para mostrar até que ponto o modelo proposto é aceitável e fiável para efeitos de previsão de prazos. Os resultados obtidos com o modelo foram bem sucedidos a um nível muito significativo. De modo a dar seguimento e maior robustez aos resultados do estudo, a presente secção faz uma modelagem

de diferentes cenários a fim de analisar o impacto da variação do nível de experiência e número de elementos nas equipes.

Para a realização da simulação de outros cenários não se levou em consideração os papeis dos elementos nem uma fase especifica do projeto. É considerado apenas número de participantes por equipe e o seu nível de experiência (sênior, pleno e júnior). O primeiro cenário que é designado por A e consiste de uma equipe constituída por elementos seniores e que varia o número de elementos de 2 a 6 de modo a observar o impacto da variação do número de elementos numa equipe de seniores.

O cenário B e C obedecem a mesma lógica do cenário A apenas com a diferença de que o cenário B é constituído por apenas elementos com nível de experiência pleno e o C por apenas elementos com o nível de experiência júnior. O cenário D introduz um elemento sênior nas equipes formadas por juniores (cenário C) e varia o número de juniores de 1 a 5. O cenário E mantem o sênior e varia o os elementos plenos de 1 a 5.

As taxas do parâmetros usadas são as apresentadas na Secção 5.3 que foram obtidas, basicamente, por meio de conversas formais com profissionais experientes na área de Engenharia de Software associados a alguns estudos relacionados [1, 11, 31] e também com base em conversas formais com os participantes do Projeto XYZ que ajudaram a recolher os tempos de permanência em cada estado. Para a construção dos diferentes cenários usa-se a carga horária diária de 9 horas que foi a carga predominante nas fases do estudo de caso usado na secção anterior para execução do modelo SAN.

A produtividade nos cenários estabelecidos é capturado com base nas probabilidades estacionárias do estado Trabalhando (Tr). A duração deste estado reflete a quantidade de trabalho que tem sido produzido e como essa duração influencia os outros estados para cada Entidade. As Figuras e tabelas que se seguem demonstram os principais resultados obtidos a partir do modelo proposto neste estudo, variando o tamanho da equipe de dois a seis em função do nível de experiência.

De acordo com alguns profissionais da área de Engenharia de Softwares sugere-se que o tamanho de uma equipe não seja superior a 10 elementos por isso, de uma forma aleatória, nesta Secção considera-se 6 como o número máximo de elementos de cada cenário pra não fugir muito do número máximo de elementos do estudo de caso deste trabalho que pela sua duração total se enquadra dentro de projetos de pequeno porte (no máximo 5000 horas). As justificações dos cenários baseam-se no senso comum podendo ser objeto de estudo para comprovar a sua veracidade.

6.1.3.1 Cenário A

Conforme descrito na Secção 6.1.3, o Cenário A consiste de equipes compostas apenas por elementos seniores variando apenas o numero de elementos de 2 a 6 em cada execução de modo a observar o impacto do aumento do número de elementos na equipe.

De acordo com os resultados observados na Tabela 6.15 pode-se notar que se pegarmos 82.1% que corresponde a probabilidade de (Tr) quando N = 2 e 71% quando N = 6 pode se constatar um decréscimo de 5.1% e dando a sensação de que a medida que o N aumenta a probabilidade de (Tr)

Tabela 6.16 – Resultados do cenário A

	n2	n3	n4	n5	n6
Tr	82.1	80.9	73.9	73	71
Pa	14.5	13.1	11.9	11.5	10.9
Co	1.7	3.2	11.7	12.62	18.1
Re	2.8	2.7	2.5	2.5	2.3

diminui. Se observamos o que acontece em cada valor de N e com ajuda do gráfico apresentado na Figura 6.8 nota-e claramente que realmente, a medida que aumenta o valor de N, as probabilidades de (Tr) diminuem apesar de não ser num ritmo muito acentuado. Com base nestes dados pode-se concluir nitidamente que o aumento de número de elementos não implica, necessariamente, o aumento da produtividade, podendo ser o resultado de o aumento de elementos aumentar mais tempo gasto em colaborações e consequentemente baixando a produtividade da equipe. Neste caso concreto pode-se notar que a maior produtividade se verifica para o menor número de elementos (N = 2).

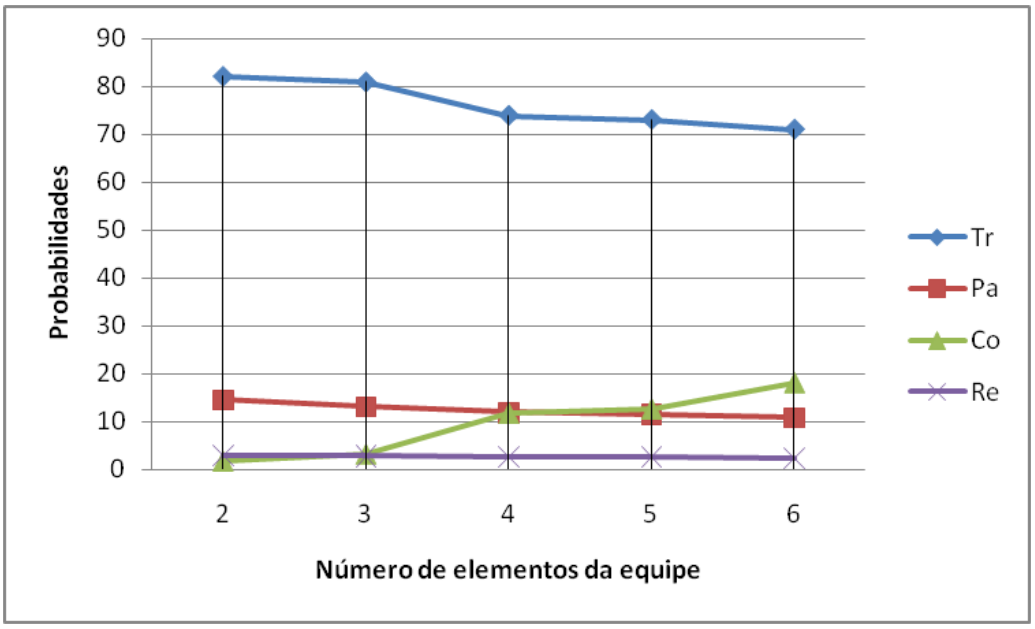


Figura 6.8 – gráfico do cenário A

Ainda no cenário A, os dados da Tabela 6.16 e do gráfico ilustrado na Figura 6.8 mostram que para todos os valores de N a probabilidade de (Tr) está sempre acima de 70% e a os restantes estados (Parado (Pa), Colaborando (Co) e Retrabalhando (Re)) gastam um pouco menos de 30% da carga horária diária. Com o aumento de número de elementos pode-se notar que a probabilidade de (Pa) diminui de 14.5% a 10.9% que representa uma variação de 3.6%.

Observando os dados de cada valor de N na Tabela 6.16 e o comportamento das probabilidades de (Pa) no gráfico da Figura 6.8 pode-se concluir que a medida que o N aumenta a probabilidade de (Pa) diminui para todos valores de N considerados neste cenário. Isso significa que com o aumento de número de elementos vai se gastando menos tempo parado.

As probabilidade de se estar no estado Colaborando (Co) vai aumentando com o aumento de N. Uma observação interessante é que para N = 6 o valor da probabilidade de (Co) dispara para acima de 18.1% o que significa que com apenas 2 elementos existe uma fraca colaboração se gastando apenas 1.7% do tempo diário de trabalho em colaboração e para N = 6 o nível de colaboração aumenta significativamente. Para o estado (Re), a medida que o N aumenta, a probabilidade de retrabalho diminui, apesar de ser uma diminuição muito ligeira que corresponde a uma variação de apenas 0.5% de N = 2 para N = 6. Dado o nível de experiencia sênior dos elementos deste cenário consome-se muito tempo em trabalho (Tr).

6.1.3.2 Cenário B

Conforme descrito na Secção 6.1.3, o Cenário B consiste de equipes compostas apenas por elementos do nível de experiência pleno variando apenas o número de elementos de 2 a 6 em cada execução de modo a observar o impacto do aumento do número de elementos na equipe.

Tabela 6.17 – Resultados do cenário B

	n2	n3	n4	n5	n6
Tr	51	47.8	46.1	45	44
Pa	34.3	30.0	28.3	25.8	22
Co	11.9	17.5	19.4	24	25
Re	2.7	4.7	6.2	6.3	9.0

O cenário B é apresentado na Tabela 6.17 e na Figura 6.9 que ilustra o respetivo gráfico. Pegando a probabilidade 51% de N = 2 e 44% de N = 6, calculando a variação entre os 2 extremos temos como resultado uma diferença negativa de 7% o que claramente nos induz a uma sensação de decréscimo das probabilidades de (Tr) a medida que o N aumenta. Na realidade, olhando mais atentamente nos dados da tabela e no comportamento do respetivo gráfico pode-se notar que as probabilidades vão diminuindo para cada valor de N. Contudo, pode-se notar, com base nos dados da tabela, que os valores das probabilidades de (Tr) variam entre 51% e 44% que consiste num nível de produtividade considerável para um elemento pleno. Do mesmo modo que acontece no cenário A, o melhor desempenho no cenário B se verifica para N = 2.

De igual modo que se verifica com as probabilidades estacionárias de (Pa) no cenário A, as probabilidade de (Pa) no cenário B diminuem a medida que o número de elementos da equipe aumenta . Essa diminuição das probabilidades de (Pa), por outras palavras, significa que com o aumento de N os elementos vão gastando menos tempo parados devido a disponibilidade de mais opções para colaboração.

Um fato interessante de sublinhar neste cenário, é o fato de que a medida que se aumenta o número de elementos, vai-se gastando menos tempo parado e mais tempo colaborando devido ao incremento de possibilidades de colaboração mas não há ganhos na produtividade.

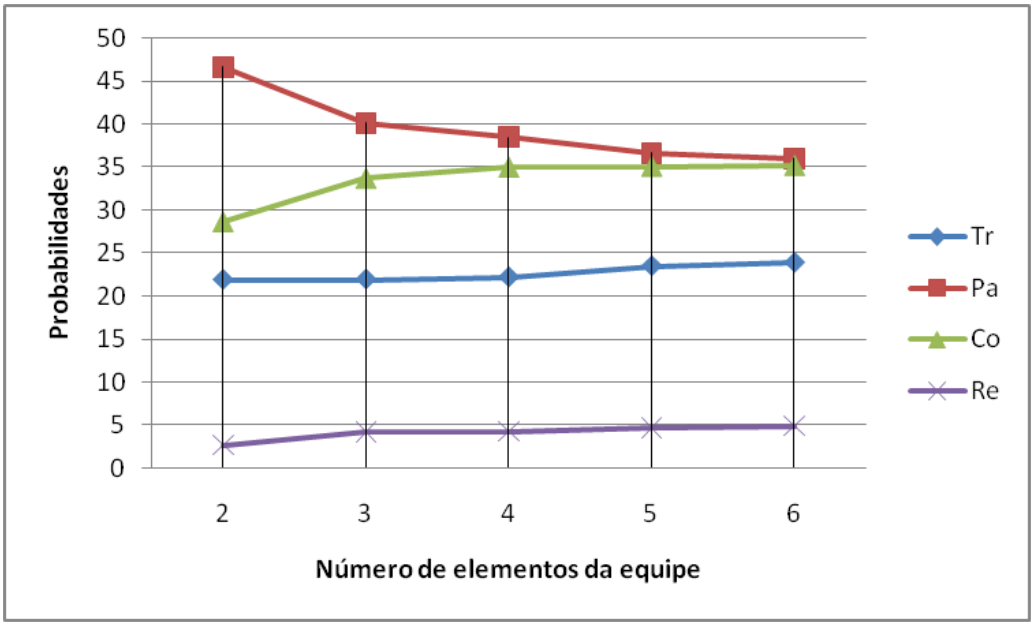


Figura 6.9 – gráfico do cenário B

6.1.3.3 Cenário C

Conforme descrito na Secção 6.1.3, o Cenário B consiste de equipes compostas apenas por elementos do nível de experiência júnior variando apenas o número de elementos de 2 a 6 em cada execução de modo a observar o impacto do aumento do número de elementos na equipe.

Os resultados do cenário C são apresentados na Tabela 6.18 e o respetivo gráfico ilustrado na Figura 6.10. Observando as probabilidades de (Tr) pode-se notar que este cenário apresenta uma situação de aumento de produtividade a medida que se aumenta o número de elementos, apesar de ser um nível de crescimento muito reduzido. Outro aspeto interessante de sublinhar, é o fato de a produtividade estar sempre abaixo dos 25% e se gastar mais tempo parado e/ou colaborando pelo fato de serem apenas elementos juniores e necessitando mais tempo para pesquisa individual ou colaboração para a realização das tarefas atribuídas.

Tabela 6.18 – Resultados do cenário C

	n2	n3	n4	n5	n6
Tr	21.9	21.95	22.2	23.5	23.9
Pa	46.6	40.15	38.5	36.7	36
Co	28.7	33.7	35	35.1	35.2
Re	2.7	4.2	4.3	4.7	4.9

No cenário C, igualmente com os cenários A e B, observando os dados da tabela e a linha do gráfico que representa as probabilidade do estado (Pa) nota-se que a medida que o N aumenta a probabilidade dos elementos da equipe estarem parados diminui.

Como pode-se observar na tabela temos para $N = 2$ uma probabilidade de 46.6% que vai descendo até que no $N = 6$ tenha uma probabilidade de 36% o que, por outas palavras, significa que

a medida que o N aumenta os elementos vão gastando menos tempos parados devido ao aumento das possibilidades de colaboração que influenciam na produtividade, principalmente quando se trata de elementos com um baixo nível de experiência como é o caso concreto de elementos juniores, apesar do aumento da produtividade neste cenário ser num nível muito reduzido.

Olhando para os 3 cenários anteriores podemos concluir que nem sempre o aumento de número de elementos numa determinada equipe significa o aumento de produtividade, podendo existir outros fatores que contribuem para o aumento da produtividade e não necessariamente o incremento do tamanho da equipe.

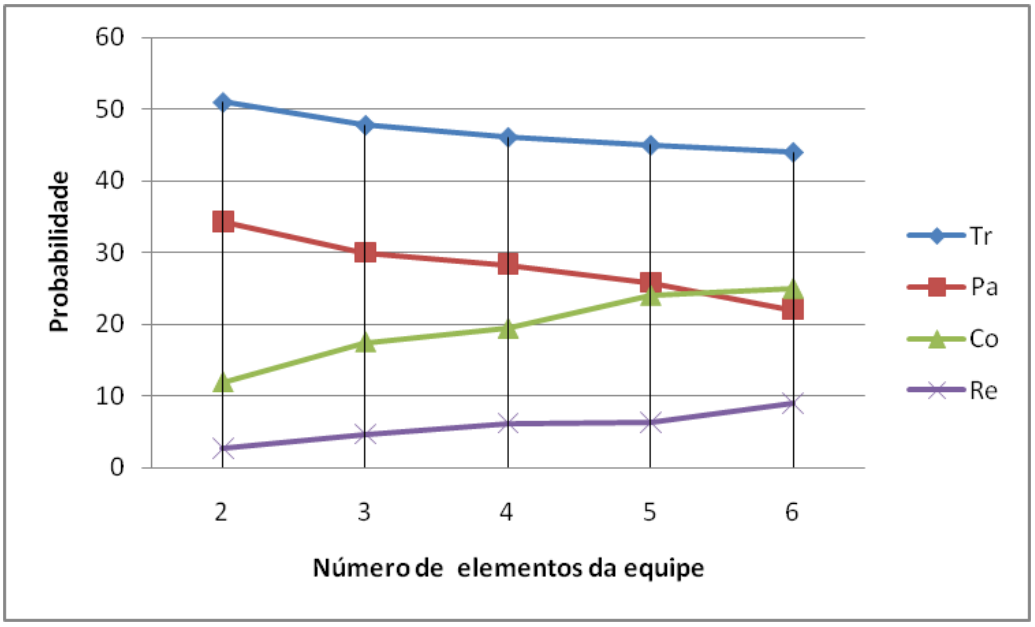


Figura 6.10 – gráfico do cenário C

6.1.3.4 Cenário D

O cenário D, como mencionamos no início desta subsecção, consiste de equipes que são constituídos por um elemento sênior e os restantes elementos juniores. Os resultados deste cenário são ilustrados na Tabela 6.19 e no respetivo gráfico na Figura 6.11. Neste cenário o foco é apenas para as probabilidades estacionárias do estado (Tr) de modo a fazer uma análise do impacto que a inclusão do elemento sênior tem na produtividade de uma equipe de juniores que foi analisada no cenário C. O cenário varia os elementos juniores de 1 para 5 e se mantendo sempre o elemento sênior.

Tabela 6.19 – Resultados do cenário D

Estado	%	n2	n3	n4	n5	n6
Tr	sênior	77.9	76.1	76	76.55	77.1
	Juniores	24.3	21	21.9	22.3	24.2
	Prod Med	51.1	48.6	49.0	49.4	50.7

De acordo com a Tabela 6.19 observa-se que para todos os valores de N o elemento sênior introduzido na equipe mantem a sua produtividade em torno de 76.5%. A produtividade dos elementos juniores mantem-se em torno dos 22%. De um modo geral, a produtividade média do cenário D aumenta consideravelmente com a introdução de um elemento sênior no meio dos juniores comparado com o Cenário C que consiste de apenas elementos juniores.

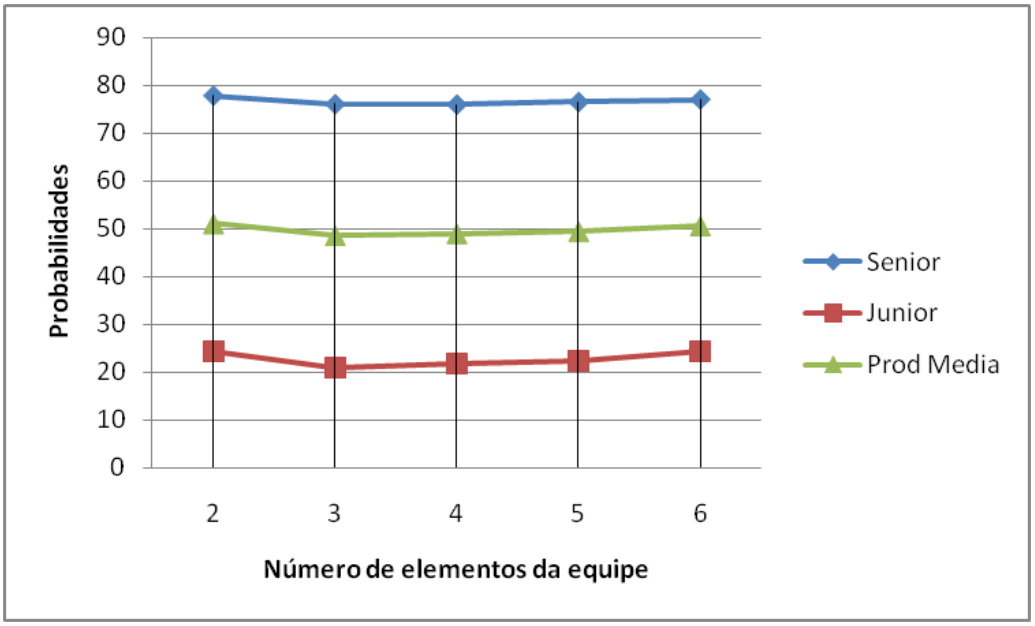


Figura 6.11 – gráfico do cenário D

6.1.3.5 Cenário E

O cenário E consiste de equipes que são constituídos por um elemento sênior e os restantes elementos plenos. Neste cenário o foco é para as probabilidades do estado (Tr) de modo a fazer a análise do impacto que a inclusão do elemento sênior tem na produtividade de uma equipe que tinha apenas elementos de capacidades plenas que foi analisada no cenário B.

Tabela 6.20 – Resultados do cenário E

Estado	%	n2	n3	n4	n5	n6
Tr	sênior	80	78	75.6	75.4	75.7
	plenos	54	49.2	44.4	45.2	45.7
	Prod Med	67	63.6	60	60.3	60.7

Em conformidade com a Tabela 6.20 e o gráfico da Figura 6.12 nota-se claramente que existe um ganho significativo na produtividade da equipe com o pico de produtividade para $N = 2$. A produtividade vai decrescendo tanto para o sênior incluso, bem como para a média de produtividades dos elementos de experiência plena. A partir de $N = 4$ quase que se mantem a produtividade o que realça a ideia de que nem sempre o aumento de número de elementos implica aumento de

produtividade. A introdução do sênior simplesmente aumentou a média de produtividade da equipe para todos valores de N.

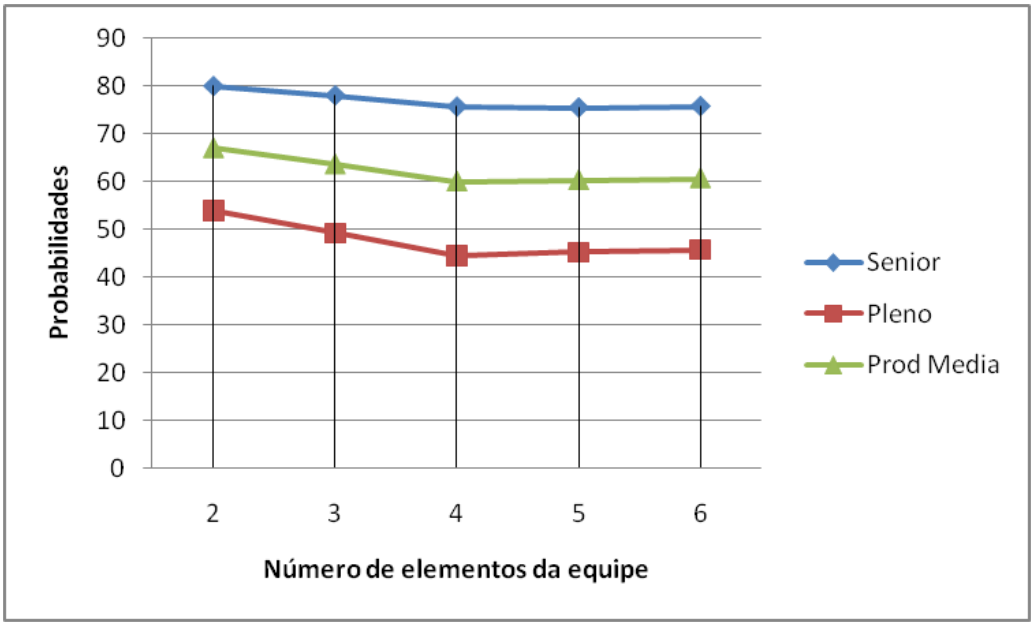


Figura 6.12 – gráfico do cenário E

Todas as observações consideradas nos 5 cenários definidos carecem duma sustentação baseada em experimentos e/ou simulações que comprovem as afirmações estabelecidas. As afirmações que justificam as prováveis causas do comportamento que se verifica em cada cenário são com base no censo comum como afirmado no inicio da Secção.

7. CONCLUSÕES

Com base neste estudo é possível substanciar que a avaliação de desempenho das equipes de desenvolvimento de software utilizando modelos analíticos é uma ferramenta importante que pode dar uma grande contribuição no sucesso dos processos na área de gestão de projetos de software por permitiram que os gestores de projetos, com base nesses modelos matemáticos, façam a análise e predição de comportamentos (esperados ou não esperados) dos elementos das equipes. Elementos com diferentes níveis de experiência (Sênior, Pleno e Júnior). Essa análise de comportamentos pode ser feita para situações bastante complexas.

Este trabalho tem uma contribuição importante de trazer mais um cenário de estudo de caso que demostra, com base em dados concretos, a aplicação do modelo SAN para avaliação de desempenho de equipes de desenvolvimento de software. Este estudo foi feito com base na suposição assumida com base em outros estudos que a modelagem analítica recorrendo ao formalismo SAN pode ser uma opção digna de credito para construir equipes de desenvolvimento de software, antecipando aquilo que sera o desempenho global da equipe e de cada elemento em particular.

O software PEPS é uma ferramenta eficiente para a resolução de sistemas de equações lineares derivadas de modelos SAN. Para qualquer responsável em tomar decisões em projetos de softwares, o mais importante do que a própria ferramenta é o grau de confiança dos resultados de saída gerados na execução dos modelos SAN. Este trabalho teve seu foco centrado na medição do nível de credibilidade dos resultados gerados pelos modelos executados no PEPS ao invés de estudar a qualidade da própria ferramenta PEPS.

De todo modo, a principal contribuição do nosso trabalho é apresentar as técnicas de análise de modelagem, especificamente o formalismo SAN, a fim de fornecer previsões confiáveis em projetos de software baseados na metodologia waterfall. Este estudo apenas apresentou uma abstração com base num estudo de caso concreto. No entanto, a metodologia waterfall pode ser aplicada em diferentes abordagens (algumas delas descritas neste trabalho no capítulo dos trabalhos relacionados). A abordagem do modelo waterfall considerada neste estudo pode ser considerada como um ponto de partida que visa mostrar até que ponto o modelo SAN pode ser útil em projetos de softwares baseados no modelo waterfall. Muito sobre a aplicabilidade do formalismo SAN em projetos da Engenharia de Software precisa ser feito e aprofundado de modo a realçar a importância do formalismo SAN na construção de equipes e contribuir em grande medida na antecipação no desempenho da equipe no geral e particularmente em cada constituinte/participante da equipe.

Na etapa de modelagem várias captações foram realizadas mas, com base nas taxas do estudo de caso foi possível obter resultados numéricos que ilustram de uma forma muito impressionante o quão os modelos SAN podem ser precisos na previsão de duração das fases de um projeto de desenvolvimento de software que se baseia no modelo de processos Waterfall. Na fase de desenvolvimento que ocupou a maior porção de tempo neste estudo obteve-se um tempo calculado de 898 horas que em termos numéricos tem uma diferença não muito significativa comparado com a diferença entre o tempo observado e o estimado, apesar de o tempo estimado estar mais próximo do

observado do que o calculado. Se pegar os resultados da fase de especificação de requisitos, temos tempo estimado 144 horas, tempo observado 138 horas e tempo calculado 136 horas conclui-se que o tempo calculado aproxima-se do tempo observado do que o estimado o que sustenta que o modelo proposto possui um nível aceitável de qualidade de previsão de tempo. De forma resumida, com base nos tempos totais estimados, observados e calculados conclui-se que o tempo total calculado aproxima-se do observado, comparado com o estimado. Essa é uma evidencia mais generalizada que, de acordo com os dados do estudo de caso, substancia a qualidade do modelo proposto.

Estes fatos descritos no parágrafo anterior, por si só justificam a hipótese inicial que a modelagem analítica, no nosso caso particular modelado utilizando o formalismo SAN, pode ser uma opção digna para construir equipes em projetos de desenvolvimento de software antecipando o seu desempenho.

Uma possível trabalho futuro é estender o modelo teórico atual para abranger outros aspectos diferentes que podem ser relevantes na atividades dos elementos de uma equipe de desenvolvimento de software de modo a projetar outros resultados que podem ser muito importantes para gestores de projetos de software. Por exemplo, nesta abstração não foi considerada a iteração das equipes de cada fase com o gerente do projeto que dentre outros suporte, fornece um suporte administrativo as equipes e seria interessante avaliar o nível de desempenho nessa iteração.

Outro trabalho futuro interessante é analisar cuidadosamente as representações modelos ágeis, variando o número de participantes com diferentes níveis de experiência (como foi realizado neste estudo) e analisar o impacto dessa variação de número de participantes e diferentes níveis experiência. Esta análise numérica pode fornecer conclusões interessantes não só para casos práticos específicos, mas também para fazer comparações formais entre diferentes modelos de processos de desenvolvimento de software.

De modo a enriquecer este estudo, pode ser um trabalho futuro interessante aumentar o número de estudos de caso de modo a aumentar os inputs para análise de resultados de execução dos modelos SAN. O modelo proposto neste trabalho pode ser mais abstraído para considerar vários aspetos que podem ser relevante para avaliação de desempenho. O modelo analítico proposto neste trabalho pode ser aplicado para um número elevado de casos práticos para além do projeto XYZ e obter boas previsões para equipes de desenvolvimento de software. Estudos de caso bem documentados e detalhados podem ser uma fonte rica de parâmetros e cenários de abstração que podem levar a outros modelos de equipes mais interessantes, mais complexas e mais abrangentes.

Este trabalho é mais um esforço com o objetivo de introduzir técnicas modelagem analítica na área de Engenharia de Software. E podemos usar estas técnicas tanto projetos no mesmo espaço geográfico assim como para projetos de software em espaços geograficamente separados .

De salientar que houve várias limitações para a realização deste trabalho dentre elas o difícil acesso a dados de projetos de desenvolvimento de software com base em modelos Waterfall.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] P. Fernandes, A. Sales, A. Santos, and T. Webber. Performance Evaluation of Software Development Teams: a Practical Case Study, Electronic Notes in Theoretical Computer Science (ENTCS), vol. 2755:73–92, 2011.

[2] I. Sommerville. Engenharia de Software, 9. edição, São Paulo, Pearson Prentice Hall, 2011.

[3] J Cummings, J. Espinosa, and C. Pickering. Crossing Spatial and Temporal Boundaries in Globally Distributed Projects: A relational model of coordination delay. Information Systems Research, 20:420–439, 2009.

[4] K. Swigger, F. Serce, F. Alpaslan, R. Brazile, G. Dafoulas, and V. Lopez. A Comparison of Team Performance Measures for Global Software Development Student Teams, In Proceedings of the 4th International Conference on Global Software Engineering (ICGSE'09), 27(1):490 – 503, 2009.

[5] M. Bass, J. Herbsleb, and C. Lescher. Collaboration in Global Software Projects at Siemens: An Experience Report, In Proceedings of the 2nd International Conference on Global Software Engineering (ICGSE'07), pp. 33–39, 2007.

[6] R. Sangwan, N. Mullick, M. Bass, D. Paulish, and J. Kazmeier. Global Software Development Handbook, CRC Press, 2006.

[7] A. Mayrhauser. The role of simulation in software engineering experimentation, Lecture Notes in Computer Science 706:177–179, 1993.

[8] S. Ferreira, J. Collofello, D. Shunk, and G. Mackulak. Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation, Journal of Systems and Software 82 (10) (2009) 1568–1577.

[9] S. Setamanit, W. Wakeland, and D. Raffo. Using simulation to evaluate global software development task allocation strategies: Research Sections, Software Process: Improvement and Practice 12 (5) (2007) 491–503.

[10] M. Kellner, R. Madachy, and D. Raffo. Software process simulation modeling: why? what? how?, The Journal of Systems Software, pp 1–7, 1999.

[11] R. Czekster, P. Fernandes, A. Sales, and T. Webber. Analytical Modeling of Software Development Teams in Globally Distributed Projects. In International Conference on Global Software Engineering (ICGSE'10), Princeton, NJ, USA, IEEE Computer Society, pp. 287–296, 2010.

[12] B. Plateau. On the Stochastic Structure of Parallelism and Synchronization Models for Distributed Algorithms, ACM SIGMETRICS Performance Evaluation Review, vol. 13, no. 2, pp. 147–154, 1985.

[13] L. Brenner, P. Fernandes, and A. Sales. The Need for and the Advantages of Generalized Tensor Algebra for Kronecker Structured Representations. International Journal of Simulation: Systems, Science Technology, 6(3-4):52–60, 2005.

[14] W. Stewart, Introduction to the numerical solution of Markov chains, Princeton University Press, 1994.

[15] L. Brenner, P. Fernandes, B. Plateau, and I. Sbeity. PEPS 2007 - Stochastic Automata Networks Software Tool. In Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST 2007), pages 163–164. IEEE Press, 2007.

[16] M. Tribastone, A. Duguid, and S. Gilmore. The PEPA Eclipse Plug-in, Performance Evaluation Review 36 (4):28–33, (2009).

[17] R. M. Czekster, P. Fernandes, and T. Webber. GTA express - A Software Package to Handle Kronecker Descriptors, in: Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST 2009), IEEE Computer Society, Budapest, Hungary, pp. 281–282, 2009.

[18] R. Chanin, F.Dotti, P. Fernandes, and A. Sales. Avaliação Quantitativa de Sistemas, 2005.

[19] R. Pressman. Engenharia de Software.McGraw-Hill, 2001.

[20] S. Pfleeger. Engenharia de Software: Teoria e Prática, Prentice Hall do Brasil, 2004.

[21] R. Wazlawick. Engenharia de Software: Conceitos e Práticas; Rio de Janeiro - Brasil, Elsevier Editora Ltda, 2013.

[22] S. Pühl, D. GmbH, and R. Fahney. How to assign cost to "Avoidable Requirements Creep" A step towards the waterfall's agilization, IEEE 19th International Requirements Engineering Conference, pages 94–102, 2011.

[23] R. Czekster. Solução Numérica de Descritores Markovianos a partir De Re-Estruturações de Termos Tensoriais, tese de doutorado, PUCRS, Porto Alegre, 2010.

[24] R.M. Czekster, P. Fernandes, R. Prikladnicki, A. Sales, A.R. Santos, and T. Webber. Follow-The-Sun Methodology in a Stochastic Modeling Perspective. In 6th IEEE International Conference on Global Software Engineering (ICGSE): Methods and Tools for Project/Architecture/Risk Management in Globally Distributed Software Development Projects (PARIS), pages 54–59, Helsinki, Finland, August 2011.

[25] E. Carmel, Y. Dubinsky, and J.A. Espinosa. Follow the sun software development: New perspectives, conceptual foundation, and exploratory field study. In System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on, pp 1–9, 2009.

[26] E. Carmel. Global Software Teams: collaborating across borders and time zones, 1999. Published by Prentice Hall-PTR.

[27] J. Espinosa, and E. Carmel. Modeling Coordination Costs Due to Time Separation in Global Software Teams, International Workshop on Global Software Development, part of the International Conference on Software Engineering, Portland, Oregon, USA, May, pp 9–7, 2003.

[28] J. Herbsleb. An Empirical Study of Global Software Development: Distance and Speed, in 23rd. International Conference on Software Engineering (ICSE). 2001. Toronto, Canada: IEEE Computer Society Press.

[29] P. Jalote, G. Jain. Assigning tasks in a 24-h software development model, Journal of Systems and Software 79(7), 904-911 (2006).

[30] P. Sooraj, and P. Mohapatra. Modeling the 24-hour software development process. Strategic Outsourcing: An International Journal, 1(2):122–141, 2008.

[31] A. Santos. Stochastic Modeling of Global Software Development Teams, Tese de Mestrado, PUCRS, Porto Alegre, 2012.

[32] L. Brenner, P. Fernandes, and A. Sales. MQNA - Markovian Queueing Networks Analyser. In 11th IEEE/ACM International Symposium on Modelling, Analysis and Simulation on Computer and Telecommunication Systems (MASCOTS'03), pages 194–199, Orlando, FL, USA, IEEE Computer Society, 2003.

[33] J. Raj. The Art of Computer Systems Perfomance Analysis – Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley Sons Inc, 1991.

[34] G. Dafoulas, K. Swigger, R. Brazile, F. Alpaslan, V. Cabrera, and F. C. Serce. Global Teams: Futuristic models of collaborative work for today's software development industry. Hawaii International Conference on System Sciences, 0:1–10, 2009.

[35] M. Pidd. Computer Simulation in Management Science. John Wiley Sons, Inc, NY, USA 3rd Edition, 1992.

[36] R. Rocha, C. Costa, R. Prikladnicki, R. De Azevedo, I. Junior, and S. Meira. Modelos de Colaboração no Desenvolvimento Distribuído de Software: uma Revisão Sistemática da Literatura, Centro de Informática – Universidade Federal de Pernambuco, 2002.

[37] A. Gil. Como elaborar projetos de pesquisa. 4ª ed. São Paulo: Atlas 2008.

[38] D. Miller. Slaying The Dragons: An Agile Aprroach To Software Development – A Management Overview. 2000.

[39] K. Beck. Programação Extrema Explicada. Bookman, pp 263, 1999.

[40] K. Beck. Programação extrema (XP) explicada: acolha as mudanças. 2ª ed. Porto Alegre, Bookman, pp 182, 2004.

[41] V. Teles. Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming. 2005. Dissertação (Mestrado em Informática), UFRJ.

[42] J. Hartmann. Estudo Sobre a Aplicação de Métodos Ágeis no Desenvolvimento e Gerenciamento de Projetos de Software, Springer-Verlag, 2003.

[43] K. Schwaber, and M. Beedle. Agile Software Development with Scrum, NJ, Prentice-Hall, 2002.

[44] C. Reis. Caracterização de um Modelo de Processo para Projetos de Software Livre. 2001.

[45] J. Highsmith, and K. Cockburn. Extreme Programming. E-Business Application Delivery, Feb., pp 4-17, 2000.

[46] E Garcia, R. Penteado, and J. Coutinho. Padrões e Métodos Ágeis: agilidade no processo de desenvolvimento de software, 2003.

[47] R. Sangwan, M. Bass, N. Mullick, D. Paulish, and J. Kazmeier. Global Software Development Handbook (Auerbach Series on Applied Software Engineering Series). Auerbach Publications, Boston, MA, USA, 2006.

[48] F. Padberg. A discrete simulation model for assessing software project scheduling policies. Journal Software Process: Improvement and Practice (SPIP), 7:127–139, 2002.

[49] J. Hopcroft, and J. Ullman. Introduction To Automata Theory, Languages, And Computation. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1990.

Pontifícia Universidade Católica do Rio Grande do Sul
Pró-Reitoria Acadêmica
Av. Ipiranga, 6681 - Prédio 1 - 3º. andar
Porto Alegre - RS - Brasil
Fone: (51) 3320-3500 - Fax: (51) 3339-1564
E-mail: proacad@pucrs.br
Site: www.pucrs.br/proacad