Simulação de Alocação de Recursos em Projetos de Desenvolvimento de Software Utilizando Teoria das Filas

Emanuel F. Coutinho[1], **Carla I. M. Bezerra**[1]

[1]Programa de Pós-Graduação em Computação (PCOMP)
Universidade Federal do Ceará (UFC) – Quixadá – CE – Brasil

emanuel.coutinho@ufc.br, carlailane@ufc.br

Abstract. *The variety of activities and roles in application development and maintenance entails replanning, mainly of effort and cost. The project cost is often associated with the project team. Queuing theory can be described as customers who arrive at a service, wait for the service if it is not immediate, and leave the system after being served. In this context, queuing theory can help in the project team and tasks distribution, and in the service level assessment. The objective of this work is to use queuing theory to aid the distribution of tasks in the context of application development and maintenance. As a result, idleness in the use of resources and a reduction in waiting time for tasks were detected.*

Resumo. *A variedade de atividades e papéis no desenvolvimento e manutenção de aplicações acarreta replanejamentos, principalmente de esforço e custo. O custo do projeto muitas vezes está associado à equipe do projeto. Teoria das filas pode ser descrita como clientes que chegam para o atendimento de um serviço, aguardam caso não seja imediato, e saem do sistema após serem atendidos. Nesse contexto, a teoria das filas pode auxiliar na distribuição da equipe do projeto, de tarefas e na avaliação do nível de serviço. O objetivo deste trabalho é utilizar teoria das filas no auxílio da distribuição de tarefas no contexto de desenvolvimento e manutenção de aplicações. Como resultados, detectou-se ociosidade na utilização dos recursos e redução no tempo de espera de atendimento das tarefas.*

1. Introdução

A alocação equitativa de recursos humanos é uma tarefa que pode maximizar a eficiência dos recursos e otimizar o desempenho dos negócios [Zhao et al. 2020]. No gerenciamento de projetos, especialmente no desenvolvimento de software, a alocação de recursos humanos é fundamental não apenas para o sucesso do projeto, mas também para uma boa estimativa de custos, auxiliando a empresa na tomada de decisão em contratar ou não projetos ou recursos humanos [Chiang e Lin 2020]. Para resolver o problema de alocação de recursos humanos, diferentes métodos da pesquisa operacional e da área de computação foram propostos na literatura, tais como métodos exatos, algoritmos heurísticos, metaheurísticas e métodos híbridos [Bouajaja e Dridi 2017]. Diversas abordagens visam minimizar a duração e o custo do projeto ou maximizar a receita e a eficiência.

O desenvolvimento de aplicações é uma atividade que envolve diversos papéis e atividades, muitas vezes de forma concomitante. A variedade de atividades necessita de

recursos para seu pleno funcionamento, podendo esses recursos serem humanos, equipamentos, software ou financeiros. A forma na qual os recursos são utilizados nos projetos de software também varia, pois dada a dinâmica dos projetos, muitas vezes se troca o papel das partes envolvidas, conforme a necessidade e disponibilidade, e isso pode acarretar atrasos ou retrabalhos no projeto. E todo esse esforço também se aplica às manutenções e evoluções dos softwares.

Apesar da disponibilidade de modelos genéricos para o processo de manutenção de software, há uma necessidade por métodos específicos analíticos de modelagem para os atuais processos de software que possuem certo nível de maturidade [Car e Mikac 2002]. A experiência mostra que a implementação de um processo de alto desempenho é essencial para atender às metas de negócios. Estimativas de custo e esforço são um aspecto importante do gerenciamento de projetos de software [Antoniol et al. 2004]. O desenvolvimento e a manutenção de software exigem muito esforço, e o custo do projeto está estritamente vinculado ao esforço necessário, ou seja, à equipe do projeto.

A teoria das filas é um campo bem consolidado, com estudos datando de 1900 [Gross e Harris 1998][Antoniol et al. 2004]. Esta teoria foi aplicada com sucesso a uma grande variedade de problemas: comutação telefônica, projeto de redes, manutenção de centros de funcionários, reparo de peças, distribuição de mercadorias e gerenciamento de centros de serviços. No contexto do desenvolvimento e manutenção de software, a teoria das filas pode auxiliar na distribuição da equipe do projeto, e na avaliação do nível de serviço. Um sistema de filas pode ser descrito como clientes que chegam para o serviço, aguardam pelo serviço caso o atendimento não seja imediato, e saem do sistema após serem atendidos [Antoniol et al. 2004]. O termo cliente é usado em um sentido geral e não implica necessariamente um cliente humano (por exemplo, solicitações de manutenção podem ser consideradas clientes). Um sistema de filas modela o estado estacionário do processo, enquanto situações transitórias não são levadas em consideração. Os parâmetros do sistema de enfileiramento observável são: (i) Taxa de tráfego de chegada: a taxa média de clientes que entram no sistema de filas e, se disponível, sua distribuição estatística; (ii) Tempo de serviço: o tempo que um servidor precisa para processar uma solicitação e, se disponível, sua distribuição estatística; (iii) Capacidade da fila: finita ou infinita; e (iv) Disciplina de fila: FIFO, LIFO, aleatória, com prioridade, etc.

No contexto da Engenharia de Software, simulações podem auxiliar as empresas em suas estimativas da alocação dos recursos humanos, e consequentemente nos custos associados. Pode informar situações de sobrecarga e de ociosidade de recursos, possibilitando um replanejamento mais efetivo. Outro aspecto importante da simulação é no ensino de Engenharia de Software. Por meio de simulações, situações variadas podem ser projetadas e analisadas. A facilidade da alteração dos recursos possibilita experimentar diversos cenários no contexto educacional, permitindo que os alunos vejam os efeitos das alterações em recursos pertinentes a projetos de software. Neste cenário, existe uma lacuna na literatura por trabalhos de simulação em gerenciamento e processos de software.

O objetivo deste trabalho é apresentar o uso da Teoria das Filas e simulação para auxiliar na distribuição de tarefas de manutenção no contexto de desenvolvimento e manutenção de aplicações. Como resultados, por meio das simulações detectou-se ociosidade na utilização dos recursos e o tempo de espera das tarefas para o atendimento pelos programadores reduziu. Não é o foco deste trabalho estudar maturidade de proces-

sos, e sim motivar o uso de Teoria das Filas. Este artigo está dividido nas seguintes seções: a Seção 2 descreve alguns trabalhos relacionados; a Seção 3 apresenta a metodologia do trabalho, com cenários e projeto das simulações; na Seção 4 os resultados e análises são descritos; e por fim, a Seção 5 exibe as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Alguns trabalhos na literatura estudaram a alocação de recursos em projetos de desenvolvimento do software. Cada um tratou de aspectos específicos, mas em geral o foco foi a melhoria dos processos.

Aversano et al. (2001) pesquisaram a introdução do gerenciamento de fluxo de trabalho em uma organização de manutenção de software com um projeto piloto envolvendo uma grande empresa internacional de software. O trabalho apresentou a implementação de protótipo baseada em fluxo de trabalho que automatiza o fluxo das atividades do processo de manutenção comum. Os líderes de projeto e de equipe tiveram a sensação de que a introdução da tecnologia de fluxo de trabalho ajuda a reduzir o tempo de gerenciamento de processos, principalmente na: (i) padronização dos canais pelos quais as solicitações de manutenção são recebidas; (ii) melhoria na capacidade de monitorar e controlar solicitações de manutenção; e (iii) melhoria no suporte à alocação de recursos.

Antoniol et al. (2004) apresentaram uma abordagem baseada na teoria de filas e simulação estocástica para ajudar a planejar, gerenciar e controlar a equipe do projeto e o nível de serviço resultante em processos de manutenção multifásicos distribuídos. Dados de uma intervenção massiva de manutenção em um grande sistema de software financeiro foram usados para simular e estudar diferentes configurações do centro de serviço para um projeto de manutenção de software distribuído geograficamente. A teoria das filas e a simulação estocástica forneceram um meio de avaliar a equipe, avaliar o nível de serviço e avaliar a probabilidade de cumprir o prazo do projeto durante sua execução.

Jai Asundi e Sarkar (2005) pesquisaram o problema de encontrar o requisito de pessoal mínimo modelando o ambiente de suporte como uma fila, incluindo um requisito do tipo SLA. Mostrou-se que existe um limite inferior para a taxa de serviço e como ela deve ser calculada. Certas regras práticas que os gerentes de projeto de software podem empregar diante de mudanças e suas restrições foram validadas. Utilizando o modelo, um processo foi descrito que pode ser usado pelos gerentes de projeto para calcular os níveis de pessoal para projetos de manutenção e suporte, conforme requisitos de SLA.

Chiang e Lin (2020) propuseram uma estrutura para ajudar uma empresa de software a avaliar os recursos existentes para tomar decisões sobre se a estimativa do custo é viável e ajudar a fazer a alocação de recursos humanos para a formação de equipes em duração fixa do projeto, com habilidade de trabalho e restrições orçamentárias. Para isso, utilizou-se programação inteira e um estudo de simulação para demonstrar a aplicabilidade do modelo. Os resultados de eficiência máxima de habilidade e custo mínimo de contratação são estudados na alocação dos recursos do projeto.

Os trabalhos descritos possuem algumas semelhanças com o trabalho proposto. Aversano et al. (2001) também utiliza a ideia de analisar requisições de manutenção e melhoria na alocação dos recursos. Antoniol et al. (2004) se a assemelha na análise da manutenção de sistemas e alocação das pessoas do projeto. Jai Asundi e Sarkar (2005)

analisa o desempenho, mas seu foco foi no SLA. Chiang e Lin (2020) analisaram a alocação de recursos aos projetos, porém utilizaram programação inteira nas simulações.

3. Metodologia

O objetivo do experimento é simular cenários de ambientes de desenvolvimento de software por meio de simulações projetadas. Para o atendimento deste objetivo, as seguintes etapas foram planejadas: (i) Definição de três cenários de simulação; (ii) Projeto dos cenários na ferramenta; (iii) Execução das simulações e coleta dos dados; (iv) Consolidação dos dados; e (v) Análise e interpretação dos resultados.

3.1. Contextualização e Cenários

A ideia deste trabalho é simular um ambiente de desenvolvimento e manutenção de aplicações. Em um ambiente deste tipo é comum diversos papéis estarem envolvidos no processo de desenvolvimento de software, tais como programadores, testadores, gerentes de projeto, arquitetos de software, além de partes interessadas externas à equipe, como clientes. Para fim de simplificação, considerou-se para o trabalho apenas os seguintes papéis: programador, testador e cliente. O programador foi dividido em dois tipos: o que implementa a funcionalidade e o que corrige a funcionalidade.

Demandas de desenvolvimento chegam à equipe, podendo vir de diversas fontes, como novas funcionalidades solicitadas pelo cliente, demandas já planejadas previamente para o projeto, e solicitações de correção ou melhoria na aplicação. Todos os cenários se iniciam sempre com alguma fonte de tarefas a serem desenvolvidas.

O Cenário 1 é o mais simples dos três, possuindo apenas o programador. Basicamente as tarefas chegam para o programador, que deve implementá-las, e após o término a tarefa é considerada atendida. Neste cenário, testes e correções que não estejam dentro da própria tarefa do programador não estão sendo considerados, ou seja, não há nenhum outro papel envolvido. O Cenário 2 é uma evolução do primeiro, pois explicitamente há a tarefa de testes. Em caso de necessidade de correção, há o papel de um programador específico para as devidas correções. Só após o evento de testes e as possíveis correções finalizadas, é que a tarefa é encerrada e considerada atendida. Por fim, o Cenário 3 inclui ações do cliente no fluxo do desenvolvimento. As demandas chegam ou por planejamento prévio ou por solicitações eventuais do cliente. Ambas são implementadas pelo programador, testadas, corrigidas caso necessário, e finalizadas.

3.2. Simulações

Para as simulações, utilizou-se o software Arena[1] com o modelo FIFO (*first-in first-out*). Na ferramenta, quatro componentes serão utilizados para a descrição dos cenários: *Create*: ponto de partida para entidades em um modelo de simulação. As entidades são criadas usando um agendamento ou programação ou com base no tempo entre as chegadas, e saem deste módulo para iniciar o processamento pelo sistema; *Dispose*: ponto final para entidades em um modelo de simulação. As estatísticas da entidade podem ser registradas antes de seu descarte; *Process*: principal método de processamento na simulação. Opções para capturar e liberar restrições de recursos estão disponíveis; e *Decide*: permite

[1] Arena - https://www.arenasimulation.com

processos de tomada de decisão no sistema. Inclui opções para tomar decisões com base em uma ou mais condições ou com base em probabilidades.

Para cada um dos três cenários, alguns parâmetros foram configurados, referentes às funções estatísticas de chegada das tarefas, de produtividade dos programadores, da quantidade de tarefas a serem atendidas, e do tempo de duração da simulação.

No Cenário 1, as tarefas chegavam conforme uma distribuição exponencial com parâmetro 1. A distribuição exponencial é uma distribuição muito utilizada na prática para modelar tempo de falha de objetos, e também as variáveis aleatórias de tempos entre chegadas e atendimento. Além disso, o máximo de tarefas que chegavam era limitada a 100. Para a programação, considerou-se a produtividade representada por uma distribuição triangular com os seguintes parâmetros: mínimo = 3, mais provável = 6 e máximo = 8. A unidade de tempo utilizada foi horas. O tempo de trabalho a ser simulado é de 20 dias.

O Cenário 2 também foi projetado para receber tarefas conforme uma distribuição exponencial com parâmetro 1 e o máximo de tarefas que chegavam limitado a 100. Para a programação, também se considerou a produtividade representada por uma distribuição triangular com parâmetros mínimo = 3, mais provável = 6 e máximo = 8. O programador que corrige os erros possui uma produtividade diferente do programador inicial, regida por uma distribuição triangular com parâmetros mínimo = 1, mais provável = 2 e máximo = 4. A taxa de erros foi de 30%. A unidade de tempo utilizada foi horas. O tempo de trabalho a ser simulado é de 20 dias.

O Cenário 3 também foi projetado para receber tarefas conforme uma distribuição exponencial com parâmetro 1 e o máximo de tarefas que chegavam limitado a 100. Porém, há também demandas do cliente que chegam conforme uma distribuição exponencial com parâmetro 16 e sem limite de tarefas que chegam. A programação também utilizou uma produtividade representada por uma distribuição triangular com parâmetros mínimo = 3, mais provável = 6 e máximo = 8, e o programador que corrige os erros utilizou uma produtividade representada por uma distribuição triangular com parâmetros mínimo = 1, mais provável = 2 e máximo = 4. A taxa de erros foi de 30%. A unidade de tempo utilizada foi horas. O tempo de trabalho a ser simulado é de 20 dias.

A ferramenta disponibiliza as métricas apresentadas na Tabela 1, a serem utilizadas neste trabalho, já personalizadas para as entidades equipe, cliente e programadores. Além disso, a capacidade para cada cenário a quantidade de programadores será variada em 1, 2 e 3. A quantidade de programadores que corrige erros sempre será fixa em 1.

4. Resultados e Análises

4.1. Resultados

A Figura 1 apresenta o modelo para os cenários 1, 2 e 3 projetados na ferramenta. Elas retratam o momento final da simulação e apenas para a utilização de um recurso de programação, indicando que em nenhum caso todas as demandas foram atendidas no prazo. Os números representam a movimentação das tarefas nas entidades. Como foi a imagem final, algumas tarefas não puderam ser atendidas com a configuração do modelo, permanecendo na fila, ainda em atendimento.

A Tabela 2 apresenta os resultados da simulação para o Cenário 1. Como este cenário é o mais simples, nem todas as métricas citadas na Tabela 1 estão presentes, ape-

Tabela 1. Métricas para os modelos de filas dos cenários

Métrica	Descrição
T_E	Tarefa que foi originada por meio de demandas internas, manutenções ou planejamento interno à equipe
T_C	Tarefa que foi originada por meio de demandas do cliente
E_E	Quantidade de entradas da equipe - quantidade total de tarefas T_E
E_C	Quantidade de entradas do cliente - quantidade total de tarefas T_C
S	Saída - Quantidade de tarefas que são finalizadas (atendidas) totalmente na simulação, independente de serem originadas internamente ou pelo cliente
TST_E	Tempo no Sistema da Tarefa T_E - Tempo médio em horas que a tarefa permaneceu no sistema
$TEFT_E$	Tempo de Espera na Fila da Tarefa T_E - Tempo médio em horas que a tarefa permaneceu na fila
TAT_E	Tempo de Atendimento da Tarefa T_E - Tempo médio em horas que a tarefa permaneceu em atendimento
TFT_E	Tamanho da Fila T_E - Quantidade de tarefas na fila
TST_C	Tempo no Sistema da Tarefa T_C - Tempo médio em horas que a tarefa permaneceu no sistema
$TEFT_C$	Tempo de Espera na Fila da Tarefa T_C - Tempo médio em horas que a tarefa permaneceu na fila
TAT_C	Tempo de Atendimento da Tarefa T_C - Tempo médio em horas que a tarefa permaneceu em atendimento
TFT_C	Tamanho da Fila T_C - Quantidade de tarefas na fila
U_P	Utilização do Programador - Porcentual de ocupação do programador na tarefa inicial de desenvolvimento
U_C	Utilização do Programador na Correção - Porcentual de ocupação do programador na tarefa de correção

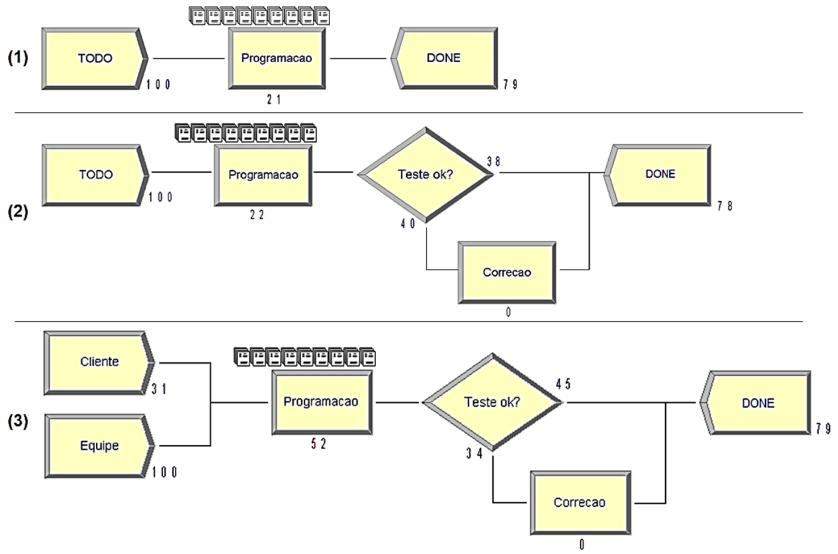

Figura 1. Representação da fila para os três cenários

nas as relacionadas ao programador no atendimento de T_E. Percebe-se que com apenas 1 programador, nem todas as tarefas foram atendidas no prazo (20 dias). Quando se ampliou para 2 programadores, todas as tarefas foram atendidas (variável S). A utilização do recurso U_P foi de 100% apenas quando havia um programador. Com 3 programadores o impacto foi apenas na redução dos tempos e ocupação dos recursos. A Figura 2 apresenta os gráficos para o Cenário 1 com todas as métricas coletadas.

Tabela 2. Métricas para o Cenário 1

Métrica	Capacidade = 1	Capacidade = 2	Capacidade = 3
E_E	100	100	100
S	79	100	100
TST_E	243,96	155,55	104,7
$TEFT_E$	237,89	149,44	98,58
TAT_E	6,06	6,11	6,11
TFT_E	60,15	31,13	20,53
U_P	100	64	42

A Tabela 3 apresenta os resultados da simulação para o Cenário 2. Este é um

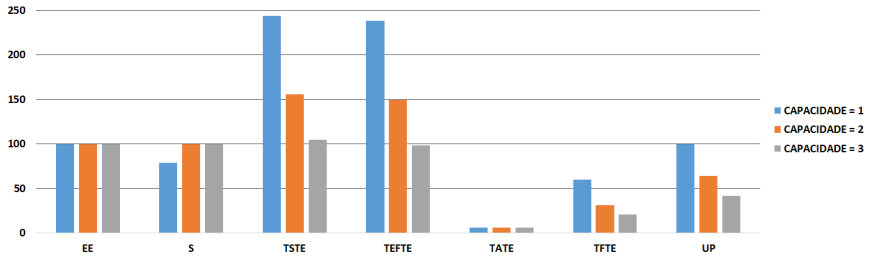

Figura 2. Métricas para o Cenário 1 - sem testes e sem cliente

cenário intermediário, com a inclusão de testes e um programador para correção. Apenas a métrica de utilização do recurso U_C de correção foi adicionada. Também percebe-se que com apenas 1 programador nem todas as tarefas foram atendidas no prazo (20 dias), e ao ampliar a capacidade todas as tarefas foram atendidas (variável S). A utilização do recurso de programação U_P foi de 100% apenas quando havia um programador, igual ao Cenário 1. Porém, a utilização do recurso de correção U_C, apesar de bem menor, aumentou. Com 3 programadores o impacto foi apenas na redução dos tempos e ocupação dos recursos. A Figura 3 apresenta os gráficos para o Cenário 2 com todas as métricas coletadas.

Tabela 3. Métricas para o Cenário 2

Métrica	Capacidade = 1	Capacidade = 2	Capacidade = 3
E_E	100	100	100
S	78	100	100
TST_E	239,91	156,57	102,62
$TEFT_E$	232,93	149,71	95,82
TAT_E	6,98	6,85	6,79
TFT_E	59,84	31,17	19,92
U_P	100	64	42
U_C	14	15	17

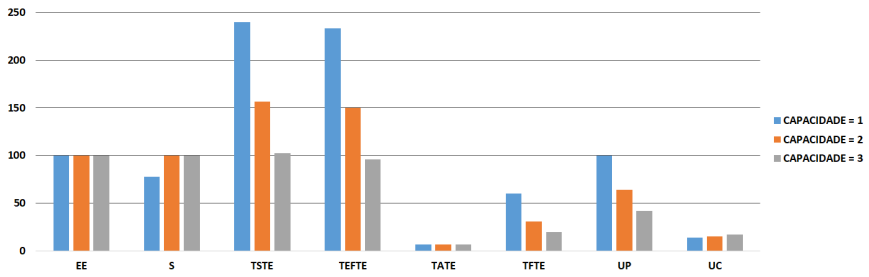

Figura 3. Métricas para o Cenário 2 - com testes e sem cliente

A Tabela 4 apresenta os resultados da simulação para o Cenário 3. Este é um cenário mais complexo, com a inclusão de testes, um programador para correção e demandas do cliente. Métricas relacionadas às tarefas do cliente foram adicionadas (T_C). Também percebe-se que com apenas 1 programador nem todas as tarefas foram atendidas no prazo (20 dias), e ao ampliar a capacidade todas as tarefas foram atendidas (variável S). Entretanto, a quantidade total de tarefas é maior devido às demandas do cliente (E_E e E_C). O comportamento da utilização dos recurso de programação U_P e correção U_C se assemelharam ao Cenário 2. Com 3 programadores o impacto foi apenas na redução dos tempos e ocupação dos recursos. A Figura 4 apresenta os gráficos para o Cenário 3 com todas as métricas coletadas.

Tabela 4. Métricas para o Cenário 3

Métrica	Capacidade = 1	Capacidade = 2	Capacidade = 3
E_E	100	100	100
E_C	31	31	30
S	79	130	130
TST_E	246,82	155,2	106,23
$TEFT_E$	239,92	148,41	99,52
TAT_E	6,9	6,78	6,7
TFT_E	75,94	37,54	23,11
TST_C	5,69	109,62	45,75
$TEFT_C$	0	102,9	38,74
TAT_C	5,69	6,71	7,01
TFT_C	0	0,01	0,03
U_P	100	81	55
U_C	14	23	20

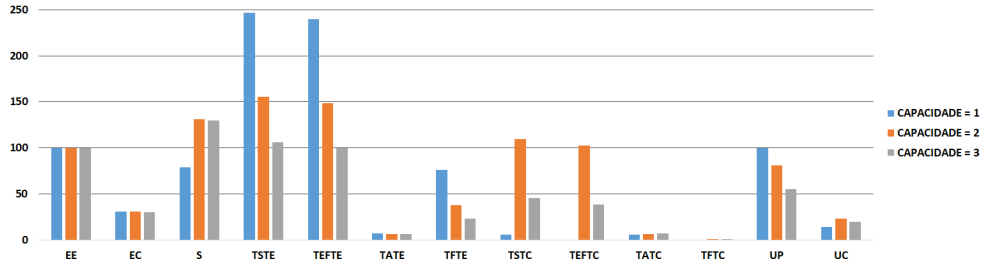

Figura 4. Métricas para Cenário 3 - com testes e cliente

4.2. Discussões e Análises

Como é de se esperar, à medida em que se adicionam recursos de programação, as tarefas passam a ser atendidas. Porém, também há a questão da ociosidade. Em todos os cenários, quando se adicionava o segundo programador, já se percebia uma ociosidade. Em média, os tempos de atendimento das tarefas são bastante próximos, mesmo com a adição de mais recursos. Isso se deve à produtividade definida seguir a mesma distribuição estatística. Nesse aspecto, o que influencia no projeto em si é a quantidade de tarefas atendidas, que aumentou, e os tempos de espera para o atendimento, que reduziram. A vazão de tarefas também aumentou conforme o aumento dos recursos. Isso pode ser verificado pelo Tamanho da Fila T_E, que foi reduzindo ao aumentar recursos, indicando que existem mais tarefas sendo executadas em paralelo, logo com uma maior vazão de atendimento.

Analisando os gráficos das Figuras 2, 3 e 4, visualizamos para todos os cenários a redução considerável dos tempos no sistema TST_E e de espera na fila $TEFT_E$ para o programador que atende as demandas iniciais. O mesmo comportamento para a diminuição das filas em TFT_E. Consequentemente, há uma aparente ociosidade nos recursos (U_P). Para o programador encarregado de correção, as imagens revelam que a utilização foi quase constante (U_C). Isto se deve ao fato que havia uma probabilidade de erros, e nem todas as tarefas seguiam o processo de correção. Logo havia mais tempo para desenvolvimento e menos tarefas de correção.

Em relação a custos, percebe-se que houve momentos de aparente ociosidade dos programadores. Também se sabe que existem diversas outras atividades além da programação em si em um ambiente de desenvolvimento de software, mas esta observação pode direcionar a uma distribuição melhor dos recursos, pois programadores possuem um custo, seja homem hora ou valor fixo, e assim predizer gastos financeiros.

Os valores dos parâmetros da simulação também influenciam nos resultados, e devem ser cuidadosamente empregados. A quantidade de 20 dias definida para o trabalho pode ser justificada como a quantidade de dias de uma *sprint*, por exemplo, mas pode ser o tamanho de uma fase de um projeto. de maior porte. A quantidade de horas de trabalho também influencia no atendimento das atividades, assim como os valores de produtividade e taxa de chegada de solicitações de serviço. Os papéis envolvidos nas simulações, assim como a quantidade, variam muito de projeto para projeto, de estrutura organizacional, de empresa. Utilizou-se neste trabalho apenas programadores para o desenvolvimento inicial e correções, mas existem diversos outros papeis que estão envolvidos em um projeto de software, não necessariamente recursos fixos do projeto, como testadores, arquitetos, equipe de qualidade, etc.

4.3. Limitações da Pesquisa

Destaca-se que este estudo é inicial, com a intenção de motivar a utilização de Teoria das Filas para o desenvolvimento de software. Os valores utilizados como parâmetros não foram baseados em algum estudo nem em uma base real, como uma base histórica de uma empresa. Isso consiste em uma limitação da pesquisa.

A quantidade de experimentos executada seguiu uma sequência de tarefas definidas na ferramenta de simulação. Todo esse processo foi executado apenas uma única vez. A análise não considerou a estocacidade do modelo de simulação. Modelos dessa natureza devem ser executados múltiplas vezes com diferentes elementos aleatórios e o intervalo de confiança verificado para que os resultados tenham alguma validade.

Os cenários projetados foram relativamente simples, com o intuito de motivar a utilização de Teoria das Filas. Os valores para os parâmetros definido para as simulações foram definidos por pessoas sem muita experiência em Teoria das Filas, mais com conhecimento básico e na ferramenta. Isso pode prejudicar o projeto das simulações, a qualidade dos resultados e as análises. Distribuições estatísticas foram utilizadas para representar a produtividade dos recursos e a taxa das chegadas das demandas, consistindo em uma limitação do trabalho, pois há a necessidade de valores reais, e cada equipe possui sua própria produtividade. Além disso, cada recurso possui uma produtividade particular. Uma estratégia seria aplicar a produtividade histórica de equipes neste trabalho.

Outro aspecto é que ao se incluir um objeto exato, como uma máquina ou qualquer elemento que não se tenha o fator humano ou subjetivo, é de se esperar que funcione bem. Porém, ao se incluir mais um programador no projeto implica em ter um tempo para se adaptar a equipe, ao projeto e processos. Essa adaptação inclui uma curva de aprendizagem que o modelo proposto pelos autores não foi considerado. Mesmo em uma simulação, outras variáveis poderiam ter sido consideradas, como tempo de adaptação do novo recurso, capacidade de produtividade diferente entre os recursos e etc.

5. Conclusão

Como conclusão geral deste trabalho, considera-se que a atividade de planejamento da alocação dos recursos e sua análise ao longo do tempo como uma atividade importante para o gerenciamento de projetos, principalmente devido à natureza dinâmica da composição das equipes e das frequências na qual demandas chegam para serem atendidas. Ferramentas de simulação podem colaborar para a melhoria do processo. Suas

contribuições científicas estão na pesquisa da alocação de recursos a projetos de desenvolvimento, estudo de viabilidade da aplicação de processos de desenvolvimento e manutenção às equipes, e na predição do atendimento de entregas conforme demandas e quantidade de recursos disponíveis ao projeto.

A utilização dos dados históricos de empresas em simulação possibilitaria que os resultados fossem ainda mais reais, beneficiando o gerenciamento dos projetos e tomada de decisão. Para o ensino de Engenharia de Software, a simulação pode auxiliar possibilitando uma visão da dinâmica dos sistemas, não apenas processos de desenvolvimento de software e gerenciamento de projetos, permitindo que alunos tenham uma visão mais real dos efeitos de mudanças.

Como trabalhos futuros, pretende-se projetar modelos mais complexos, com mais tarefas, de forma a modelar um processo de desenvolvimento de software e manutenções mais adequadamente. E utilizar valores de produtividade mais adequados para que os resultados das simulações sejam mais reais. Outro trabalho a ser conduzido é quanto à granularidade do que está sendo simulado. Uma distribuição do experimento poderia prever diferentes configurações para dois problemas de simulação conhecidos neste contexto: simulação da alocação em tarefas e simulação de processo. A primeira é mais granular e independente do processo, já a segunda monitora instâncias do processo.

Referências

Antoniol, G., Cimitile, A., Di Lucca, G. A., e Di Penta, M. (2004). Assessing staffing needs for a software maintenance project through queuing simulation. *IEEE Transactions on Software Engineering*, 30(1):43–58.

Aversano, L., Betti, S., De Lucia, A., e Stefanucci, S. (2001). Introducing workflow management in software maintenance processes. In *Proceedings IEEE International Conference on Software Maintenance. ICSM 2001*, pages 441–450.

Bouajaja, S. e Dridi, N. (2017). A survey on human resource allocation problem and its applications. *Operational Research*, 17(2).

Car, Z. e Mikac, B. (2002). A method for modeling and evaluating software maintenance process performances. In *Proceedings of the Sixth European Conference on Software Maintenance and Reengineering*, pages 15–23.

Chiang, H. Y. e Lin, B. M. T. (2020). A decision model for human resource allocation in project management of software development. *IEEE Access*, 8:38073–38081.

Gross, D. e Harris, C. (1998). *Foundamentals of Queuing Theory*. John Wiley Sons.

Jai Asundi e Sarkar, S. (2005). Staffing software maintenance and support projects. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*.

Zhao, W., Pu, S., e Jiang, D. (2020). A human resource allocation method for business processes using team faultlines. *Applied Intelligence*, 50(9).