Utilizando Teoria das Filas e Simulação de Alocação de Recursos em Ambientes de Desenvolvimento de Software

Francisco V. Pinheiro[1,2], **Maria Silva**[2], **Carla Bezerra**[1,2], **Emanuel Coutinho**[1,2]

[1]Programa de Pós-Graduação em Computação (PCOMP)

[2]Universidade Federal do Ceará (UFC) – Quixadá – CE – Brasil

{victor.pinheiro.ce, mariaerilane12}@alu.ufc.br

{carlailane, emanuel.coutinho}@ufc.br

Abstract. *Project management is considered crucial for the success of companies in different segments. The distributed allocation of human resources is a task that can maximize resource efficiency and optimize business performance. Software development is an activity that involves different roles and activities, often simultaneously. Despite the availability of generic models for the software maintenance process, there is a need for specific modeling methods for current software processes. This work presents the use of queuing theory and stochastic processes in software development projects, comparing resource allocation metrics and effort estimation.*

Resumo. *A gestão de projetos é considerada crucial para o sucesso de empresas de diversos segmentos. A alocação distribuída de recursos humanos é uma tarefa que pode maximizar a eficiência dos recursos e otimizar o desempenho dos negócios. O desenvolvimento de software é uma atividade que envolve diferentes papéis e atividades, muitas vezes simultaneamente. Apesar da disponibilidade de modelos genéricos para o processo de manutenção de software, há necessidade de métodos de modelagem específicos para os processos de software atuais. Este trabalho apresenta a utilização de teoria das filas e processos estocásticos em projetos de desenvolvimento de software, comparando métricas de alocação de recursos e estimativa de esforço.*

1. Introdução

O gerenciamento de projetos é considerado crucial para o sucesso de empresas em diversos segmentos. A utilização desta prática tornou-se imprescindível para organizações que buscam eficiência e controle em projetos, capacidade de mensuração de tempo, riscos, tarefas e recursos [Gomes et al. 2019]. O Instituto de Gestão de Projetos (PMI) em sua edição do ano de 2014, elencou os problemas mais eminentes na gerência de projetos, que são: má comunicação, falta de cumprimento de prazos, definição inadequada do escopo, mudanças constantes no escopo, recursos humanos e riscos insuficientes não avaliados adequadamente [Alencar et al. 2018].

A alocação distribuída 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 e esforço, auxiliando a organização na tomada de decisão na contratação de recursos ou projetos [Chiang and Lin 2020].

Apesar da disponibilidade de modelos genéricos para o processo de manutenção de software, há uma necessidade de métodos específicos de modelagem para os atuais processos de software. 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] [Coutinho and Bezerra 2021].

A teoria das filas é um campo bastante consolidado, com estudos datando desde 1900 [Shortle et al. 1998][Antoniol et al. 2004]. Esta teoria foi aplicada com sucesso a uma grande variedade de problemas, e no contexto de desenvolvimento de software, a teoria das filas pode auxiliar na distribuição da equipe do projeto, na avaliação de nível de serviço e na diminuição de erros em estimativas de esforço [Bouajaja and Dridi 2017].

Neste contexto, o trabalho tem por objetivo apresentar a utilização de teoria das filas e simulação para o auxílio em alocação de tarefas e recursos em ambientes de desenvolvimento de software. Como resultados se obteve alguns dados sobre riscos em estimativa de esforço, ociosidade de recursos e grande tempo de espera no atendimento de tarefas. Como resultados se obteve alguns dados sobre riscos em estimativa de esforço e também ociosidade em recursos e grande tempo de espera no atendimento de tarefas. O foco deste trabalho é dar uma visão geral da utilização de Teoria das Filas em utilização na Engenharia de Software.

Este trabalho está dividido nas seguintes seções além desta introdução: na Seção 2 alguns trabalhos relacionados são discutidos; a Seção 3 apresenta a metodologia adotada no trabalho; na Seção 4 os resultados e discussões são descritos; e por fim, as conclusões e trabalhos futuros são apresentados na Seção 5.

2. Trabalhos Relacionados

Antoniol et al. (2004) apresentaram uma abordagem baseada em teoria das 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 utilizados para simular e estudar diferentes configurações do centro de serviço para um projeto de manutenção de software distribuído geograficamente.

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 de custo é viável e auxiliar 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 tal, os autores utilizaram 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.

Coutinho e Bezerra (2021) utilizaram a técnica de teoria das filas para auxiliar na distribuição de tarefas no contexto de desenvolvimento e manutenção de software. Foram realizadas simulações previamente projetadas com o intuito de simular cenários de ambientes de desenvolvimento de software por meio das seguintes etapas: (i) definição de três

cenários; (ii) projeto dos cenários na ferramenta ARENA; (iii) execução das simulações e coleta de dados; e (v) análise e interpretação dos dados.

Os trabalhos relacionados utilizaram a ideia de analisar requisições de manutenção e melhoria de alocação de recursos, realizaram análises da manutenção de sistemas e alocação das pessoas em projetos, e analisaram a alocação de recursos aos projetos. O trabalho proposto traz a simulação de alocação de recursos, porém com cenários mais complexos em relação aos trabalhos relacionados. A Tabela 1 compara os trabalhos relacionados com o trabalho proposto por meio de alguns parâmetros.

Tabela 1. Comparação dos trabalhos relacionados com o trabalho proposto

Trabalho	Teoria das Filas	Simulação	Dados Históricos	Desenvolvimento de Software	Experimento mais Abrangente
Antoniol et al. (2004)	Sim	Sim	Não	Não	Não
Chiang e Lin (2020)	Não	Sim	Não	Não	Não
Coutinho e Bezerra (2021)	Sim	Sim	Não	Sim	Não
Trabalho proposto	Sim	Sim	Não	Sim	Sim

3. Materiais e Métodos

A metodologia empregada neste trabalho é simplificada e composta por cinco fases: (i) Definição de métricas para identificação de estimativas de esforço; (ii) Definição de três cenários de simulação; (iii) Modelagem das simulações na ferramenta ARENA; (iv) Execução das simulações e coleta de dados; e, (v) Análise e interpretação dos resultados.

3.1. Métricas de Estimativa de Esforço

A Tabela 2 apresenta algumas métricas de estimativa de esforço utilizadas neste trabalho, com o intuito de avaliar os cenários de simulação de projetos de desenvolvimento de software com base nas métricas de estimativa de esforço do trabalho de [Gomes et al. 2019] e nas métricas propostas pela própria ferramenta de simulação, utilizando a nomenclatura dada no trabalho de [Coutinho and Bezerra 2021].

Tabela 2. Métricas de Estimativa de Esforço

Métrica	Descrição
T_I	Tempo destinado ao levantamento inicial de requisitos
T_C	Tempo de criação do *mockup*
T_A	Tempo para aprovação dos clientes
P_I	Previsão inicial de esforço
T_{CC}	Tempo para construção de casos de uso
T_E	Tempo para especificações
T_{CD}	Tempo para codificação
T_H	Tempo para homologação
T_Q	Quantidade de tarefas
T_R	Tempo de retrabalho

3.2. Cenários e Contextualização

O cenário 1 é o cenário menos complexo, possui cliente e programador, basicamente tarefas chegam ao programador, são avaliadas pelo cliente e pelos testes e posteriormente homologadas. O cenário 2 é uma evolução do cenário anterior, com a adição de tarefas antes e posterior a codificação, deixando claro a correção de testes. Em caso de não aprovação do cliente ou do gerente de projetos, há um programador que corrige o pedido e só assim a tarefa é concluída. Por fim, o cenário 3, possui a adição de ações do cliente e da equipe de desenvolvimento no fluxo do desenvolvimento.

3.3. Simulações

A modelagem e a simulação propostas neste trabalho foram realizadas com o apoio do software ARENA *Simulation*[1]. Optou-se por esta ferramenta por sua ampla utilização no mercado, grande penetração no meio acadêmico e dispor de uma versão para o uso estudantil. Para cada um dos cenários propostos neste trabalho, foram configurados alguns parâmetros referentes às funções estocásticas de chegada das tarefas, produtividade dos programadores, da quantidade de tarefas atendidas e do tempo de duração de toda a simulação. Na ferramenta, quatro componentes serão utilizados para a descrição dos cenários: ***Create:*** ponto inicial para as entidades em um modelo de simulação, ou seja de onde inicia a simulação; ***Process:*** principal método de processamento na simulação; ***Decide:*** permite processos de tomada de decisão no sistema; ***Dispose:*** ponto final para as entidades de um modelo de simulação. A Figura 1 exibe os componentes do ARENA na notação definida na ferramenta.

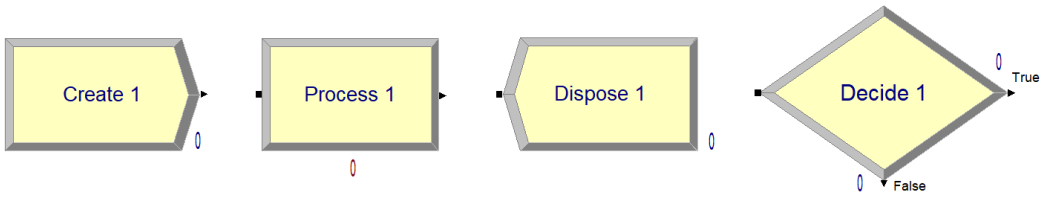

Figura 1. Componentes da ferramenta ARENA.

No cenário 1, as tarefas chegam conforme a distribuição exponencial de parâmetro 1. O número máximo de tarefas foi limitada a 150. Nas filas de implementação, considerou-se a produtividade representada por uma distribuição triangular com os parâmetros: mínimo = 2, mais provável = 4, máximo = 6, e nas filas de correção: mínimo = 1, mais provável = 3, máximo = 5. O *designer* que corrige os erros do *mockup* e o programador corrige os erros dos *bugs*, os mesmos possuem uma produtividade diferente do programador e do *designer* iniciais do projeto, regida por uma distribuição triangular com parâmetros: mínimo = 1, mais provável = 2, máximo = 4. A taxa de erros é de 20%, a unidade de tempo é horas e o tempo de trabalho são 40 dias para todos os cenários.

No cenário 2, as tarefas chegam conforme uma distribuição exponencial com parâmetro 1, e o máximo de tarefas que chegam são 150. Para as filas, também considerou a produtividade representada por uma distribuição triangular com parâmetros: mínimo = 3, mais provável = 6, máximo = 8. O programador que corrige os *bugs* possui uma produtividade diferente do programador inicial, regida por uma distribuição triangular com parâmetros: mínimo = 1, mais provável = 2, máximo = 4.

No cenário 3, as tarefas chegam em uma distribuição exponencial com parâmetro 1 e no máximo de 100 chegadas, sendo tarefas advindas do cliente e também da própria equipe de desenvolvimento conforme uma distribuição exponencial com parâmetro 2 e com limite máximo de 50 tarefas. A programação, utiliza uma distribuição triangular com os parâmetros: mínimo = 3, mais provável = 6, máximo = 8. As demais filas utilizam uma distribuição triangular com os parâmetros: mínimo = 3, mais provável = 6, máximo = 8.

[1] `https://www.rockwellautomation.com/pt-br/products/software/arena-simulation.html`

O *designer* que corrige os erros do *mockup* e o programador que corrige os *bugs* e realiza os testes, possuem uma produtividade diferente do programador e do *designer* iniciais, regida por uma distribuição triangular com parâmetros: mínimo = 1, mais provável = 2, máximo = 4.

A ferramenta disponibiliza métricas de desempenho das filas, essas métricas são apresentadas na Tabela 3, a serem utilizadas neste experimento, já adaptadas para as entidades de cada um dos cenários. A quantidade de programadores e *designers* será variada em 2, 3 e 4. A quantidade de programadores e *designers* de correção variam de acordo com a capacidade de cada cenário.

Tabela 3. Métricas de filas dos cenários

Métrica	Descrição
T_E	Tarefa que foi originada por meio de demandas internas, manutenção ou planejamento da equipe.
T_C	Tarefa que foi originada por meio de demandas do cliente
E_E	Quantidade de entradas da equipe - quantidade total de tarefas $_E$
E_C	Quantidade de entradas do cliente - quantidade total de tarefas TC
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 TE - Tempo médio em horas que a tarefa permaneceu no sistema
$TEFT_E$	Tempo de Espera na Fila da Tarefa TE - Tempo médio em horas que a tarefa permaneceu na fila
TAT_E	Tempo de Atendimento da Tarefa TE - Tempo médio em horas que a tarefa permaneceu em atendimento
TFT_E	Tamanho da Fila TE - Quantidade de tarefas na fila
TST_C	Tempo no Sistema da Tarefa TC - Tempo médio em horas que a tarefa permaneceu no sistema
$TEFT_C$	Tempo de Espera na Fila da Tarefa TC - Tempo médio em horas que a tarefa permaneceu na fila
TAT_C	Tempo de Atendimento da Tarefa TC - Tempo médio em horas que a tarefa permaneceu em atendimento
TFT_C	Tamanho da Fila TC - Quantidade de tarefas na fila
$TEFMT_C$	Tempo de Espera na Fila da Tarefa Mockup - Tempo médio em horas que a tarefa permaneceu na fila de mockup
$TEFCT_C$	Tempo de Espera na Fila da Tarefa Correção - Tempo médio em horas que a tarefa permaneceu na fila de correção
$TEFCDT_C$	Tempo de Espera na Fila da Tarefa de Codificação - Tempo médio em horas que a tarefa permaneceu na fila de codificação
$TEFET_C$	Tempo de Espera na Fila da Tarefa Especificação - Tempo médio em horas que a tarefa permaneceu na fila de especificação
$TEFCBT_C$	Tempo de Espera na Fila da Tarefa Correção de Bug - Tempo médio em horas que a tarefa permaneceu na fila de correção de bug
$TEFCUT_C$	Tempo de Espera na Fila de Detalhar Caso de Uso - Tempo médio em horas que a tarefa permaneceu na fila de detalhar caso de uso
U_P	Utilização do Programador - Porcentual de ocupação do programador na tarefa de codificação
U_B	Utilização do Programador na Correção - Porcentual de ocupação do programador na tarefa de correção
U_D	Utilização do Designer - Porcentual de ocupação do designer na tarefa inicial de desenvolvimento
U_E	Utilização do Programador na especificação - Porcentual de ocupação do programador na tarefa de especificação
U_C	Utilização do Programador na descrição dos casos de uso - Porcentual de ocupação do programador na tarefa descrição de caso de uso
U_T	Utilização do Programador nos testes - Porcentual de ocupação do programador na tarefa de teste
U_{DC}	Utilização do Designer na correção - Porcentual de ocupação do Designer na tarefa de correção

4. Resultados e Análises

Esta seção apresenta os resultados e as análises feitas a partir dos dados coletados nas simulações, de acordo com as métricas de estimativa de esforço e as métricas propostas pela ferramenta.

4.1. Resultados

A Figura 2 ilustra os cenários propostos neste trabalho e já projetados na ferramenta. Eles mostram o momento final da simulação, apresentam os recursos utilizados, indicando que nem todas as tarefas foram atendidas. Os números representam a movimentação das tarefas nas entidades no momento de simulação.

Os resultados referentes ao cenário 1, estão ilustrados na Tabela 4. Por ser um cenário menos complexo, nem todas as métricas citadas estão presentes nos resultados. Neste cenário foram utilizados 2 programadores. Percebe-se que com apenas 1 programador e 1 *designer*, nem todas as tarefas foram atendidas no prazo. Quando se ampliou para 2 programadores, todas as tarefas foram atendidas. A utilização do recurso U_P foi de 100% quando havia apenas 1 programador. Com 3 e 4 programadores e 2 *designers*, o impacto foi apenas na redução dos tempos e ocupação dos recursos. A Figura 3 apresenta os gráficos para cada um dos cenários.

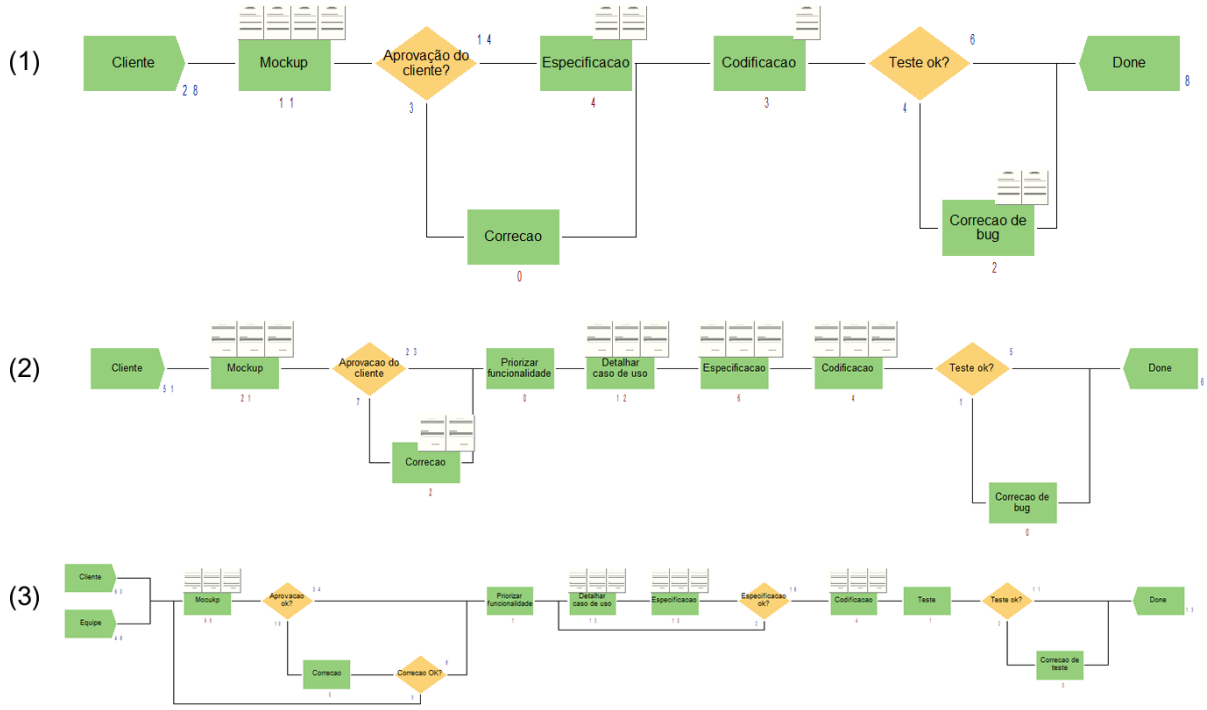

Figura 2. Representação das filas para os três cenários.

Tabela 4. Métricas e resultados para o cenário 1.

Métrica	[P = 1, D = 1]	[P = 2, D = 1]	[P = 3, D = 2]	[P = 4, D = 2]
E_C	150	150	150	150
S	101	150	150	150
TST_C	484,37	291,52	138,19	87,97
$TEFT_C$	472,02	279,19	125,98	75,66
$TEFMT_C$	240,27	228,25	84,05	27,86
$TEFCT_C$	269,83	242,12	94,67	23,80
$TEFCDT_C$	152,94	55,03	35,37	22,81
$TEFET_C$	125,55	32,95	22,72	19,40
$TEFCBT_C$	137,18	69,71	12,90	22,50
TAT_C	93,67	43,6	21,59	13,74
TFT_C	89,84	45,55	19,68	11,82
U_P	100	62	40	31
U_D	72	69	36	23

Os resultados referentes ao cenário 2 estão apresentados na Tabela 5. O cenário 2 é um cenário intermediário. Percebe-se que com apenas 1 programador e 1 *designer* nem todas as tarefas foram atendidas no prazo, e ao expandir a capacidade para 2, aumentou-se a quantidade de tarefas atendidas em 70% em relação a capacidade anterior e aumentando a capacidade para 3 e 4, todas as tarefas foram atendidas. A utilização dos recursos programador e *designer*, foi de 100% e 80% na primeira capacidade para 61% e 27%. Com 4 programadores, o impacto foi apenas na redução dos tempos e ocupação dos recursos.

Os resultados referentes ao cenário 3, estão ilustrados na Tabela 6. Este é um cenário mais complexo, com a inclusão da correção de testes, com a adição de um programador de teste e um *designer* de correção de *mockup* e de demandas da equipe de desenvolvimento. Métricas relacionadas as tarefas da equipe foram adicionadas (T_E). Também

Tabela 5. Métricas e resultados para o cenário 2.

Métrica	[P = 1, D = 1]	[P = 2, D = 1]	[P = 3, D = 2]	[P = 4, D = 3]
E_C	150	150	150	150
S	30	105	150	150
TST_C	488,33	418,33	403,22	325,19
$TEFT_C$	467,03	397,14	358,33	303,41
$TEFMT_C$	267,25	284,13	109,57	47,29
$TEFCT_C$	280,67	301,74	98,92	49,18
$TEFCDT_C$	201,97	55,36	100,26	86,21
$TEFET_C$	193,66	55,02	97,65	82,85
$TEFCBT_C$	157,66	52,93	87,94	75,51
$TEFCUT_C$	192,33	59,02	78,02	64,49
TAT_C	122,31	85,48	68,78	50,81
TFT_C	129,36	82,53	65,36	47,42
U_P	100	98	82	61
U_D	80	84	41	27

percebe-se que com apenas 1 programador e 1 *designer*, nem todas as tarefas foram totalmente processadas, neste caso um número bem inferior ao número total de tarefas, e ao ampliar a capacidade, todas as tarefas foram atendidas. Entretanto, a quantidade total de tarefas é maior devido as demandas do cliente e da equipe de desenvolvimento. A utilização dos recursos na capacidade 1 e 2 se assemelham com o cenário 2, porém nas capacidade 3 e 4 o impacto foi maior na redução dos tempos e ocupação dos recursos.

Tabela 6. Métricas e resultados para o cenário 3.

Métrica	[P = 1, D = 1 DC = 1, PC = 1 T = 1]	[P = 2, D = 1 DC = 1, PC = 2 T = 1]	[P = 3, D = 2 DC = 2, PC = 3 T = 2]	[P = 4, D = 3 DC = 2, PC = 4 T = 3]
E_C	100	100	100	100
E_E	50	50	50	50
S	31	91	150	150
TST_C	456,31	402,13	365,23	309,34
TST_E	474,44	478,01	486,16	357,13
$TEFT_C$	378,06	422,58	435,22	368,37
$TEFT_E$	444,24	447,49	455,89	326,50
$TEFMT_C$	383,56	377,11	140,46	87,70
$TEFCT_C$	55,90	43,90	23,56	12,88
$TEFCDT_C$	159,33	67,09	91,05	81,55
$TEFET_C$	180,63	65,58	90,58	77,67
$TEFCBT_C$	90,09	78,63	53,88	32,66
$TEFCUT_C$	179,15	65,35	78,90	64,58
$TEFTT_C$	0,06	0,34	1,99	1,40
TAT_C	85,26	65,90	48,46	41,59
TAT_E	40,01	31,48	25,32	18,60
TFT_C	122,92	93,68	69,08	55,16
U_P	100	98	87	68
U_D	93	95	45	31
U_B	1	2	4	2
U_E	100	95	87	68
U_C	100	95	87	68
U_{DC}	9	10	5	7
U_T	18	50	44	29

4.2. Discussões e Análises

Observou-se nos experimentos que à medida em que se adiciona mais recursos nos cenários de simulação, mais tarefas são atendidas por completo, trazendo consigo uma questão de tempo ocioso dos muitos recursos. Quando se adiciona um recurso a mais, já havia a percepção da ociosidade dos recursos, as tarefas passam a ser atendidas mais

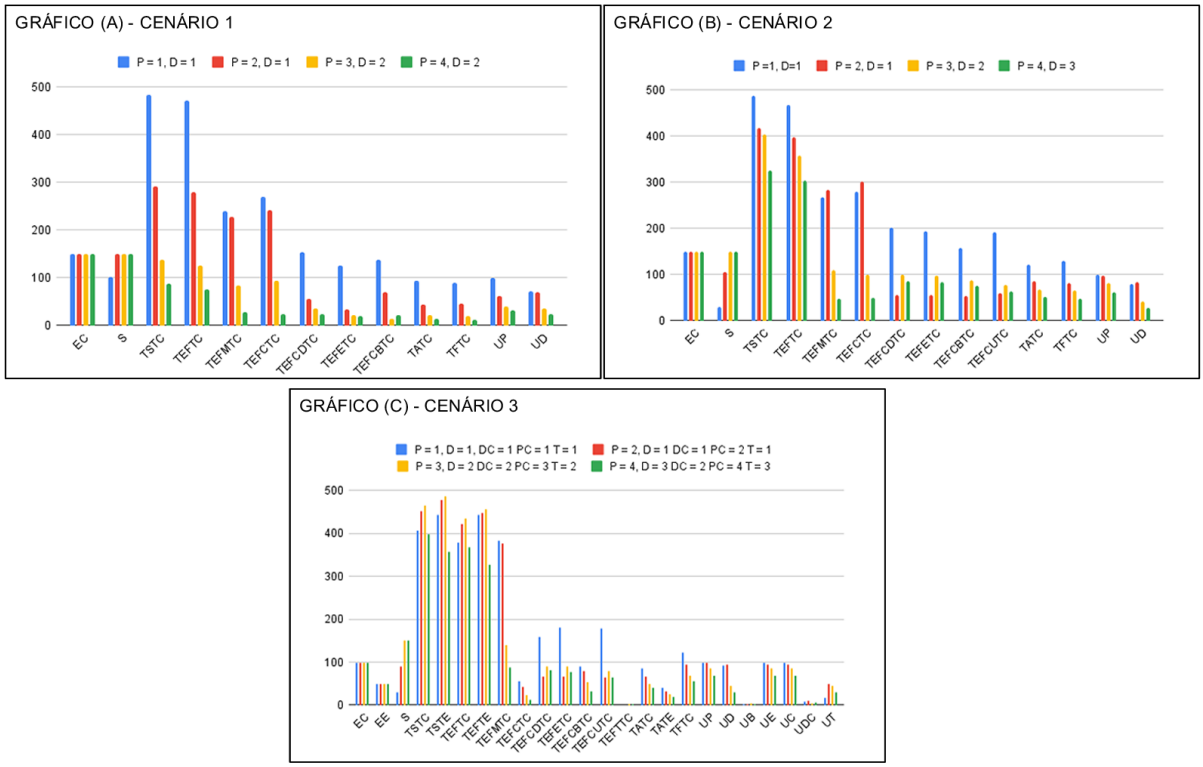

Figura 3. Resultados das métricas para o cenários 1, 2 e 3.

rápido, porém o recurso trabalha um pouco menos. Em média os tempos de atendimento das tarefas são muito próximos, mesmo com a adição de mais recursos. Neste caso, isso se deve à produtividade definida seguir uma mesma distribuição estocástica, o que influencia no projeto é a quantidade de tarefas atendidas, que aumentou, e os tempos de espera para o atendimento que reduziram [Coutinho and Bezerra 2021]. Isso pode ser inspecionado na métrica T_E Tamanho da Fila, que ao aumentar o número de recursos, esse tamanho foi reduzindo, indicando a execução de tarefas concomitantemente ao longo da simulação.

Analisando os gráficos resultantes, pode-se observar que em todos os cenários de execução há redução dos tempos no sistema TST_C TST_E e de espera na fila $TEFT_C$ $TEFT_E$ para o programador no atendimento das demandas iniciais. O mesmo comportamento pode ser observado na diminuição das filas em TFT_C e TFT_E. Com isso, a consequência é a aparente ociosidade dos recursos U_P, U_B, U_D, U_E U_{DC}, U_T e U_C. Em alguns recursos como por exemplo para o programador corretor ou *designer* corretor o tempo foi quase constante. Isso se deve ao fato da probabilidades de erros.

Em relação a estimativa de esforço, alguns dados obtidos das métricas da própria simulação e através das métricas de estimativa de esforço, como em relação ao tempo de levantamento inicial de requisitos T_I e tempo de *mockup* T_C, os dados foram animadores no que diz repeito ao tempo das filas na simulação, como mostram as Tabelas 4, 5 e 6. Em relação as demais métricas de estimativa, os dados foram condizentes ao tamanho das filas, o tempo de processamento das tarefas e principalmente a taxa de utilização dos recursos em relação a taxa de ociosidade.

Em relação a custos, pode-se afirmar que houve momentos de ociosidade ex-

trema e aparente, contudo, também pode-se observar que diversas atividades além da programação, da criação de *mockup*, da especificação do projeto, em si são existentes e reais em ambientes de desenvolvimento de software, objetivando uma melhor distribuição dos recursos, pois programadores e *designers* possuem custos elevados, seja valor hora ou valor fixo, logo quanto mais recursos humanos, mais custos financeiros.

As simulações de modo geral seguiram valores de parâmetros ajustáveis que por sua vez, influenciaram nos resultados obtidos neste trabalho, e devem ser de modo geral cuidadosamente empregados. A quantidade de 40 dias de trabalho, pode ser justificada como a quantidade de dias de 2 *sprints*, ou pode ser o tamanho de uma fase do projeto de um porte maior. O quantitativo de horas de trabalho, por sua vez influencia bastante no atendimento e processamento das tarefas, assim como nos valores de produtividade e taxas de chegada e solicitação de serviço.

4.3. Limitações de Pesquisa

Esta implementação da simulação de eventos contínuos, aplicada a ambientes de desenvolvimento de software é um estudo inicial da literatura, com a intenção de motivar e aplicar de forma considerável a Teoria das Filas com foco no desenvolvimento de software. Os valores utilizados para a execução das simulações dos cenários foram baseados parcialmente em estudos anteriores, porém esses estudos não são tão conclusivos para a literatura e possuem várias ameaças à validade, de forma que é uma limitação desta pesquisa.

Outra limitação é a dificuldade em generalizar os resultados obtidos para outros ambientes de desenvolvimento de software, servindo apenas para os ambientes projetados neste estudo, podendo ser utilizado parcialmente para outros projetos. Em relação ao quantitativo de experimentos realizados, eles seguiram uma sequência de tarefas pré-definidas na ferramenta, este processo foi executado uma única vez. Por outro lado, este estudo não considerou a estocacidade do modelo, logo modelos dessa finalidade podem ser executados várias vezes e com inúmeros elementos aleatórios.

5. Conclusão e Trabalhos Futuros

Este trabalho consistiu na realização de simulação contínua de ambientes de desenvolvimento de software em funcionamento, levando em consideração a utilização de recursos humanos e estimativas de esforço. Com isso, conclui-se que o planejamento, alocação e análise dos recursos ao longo dos tempos, é uma atividade bastante importante em projetos de software, pois possibilitam que as equipes se auto-organizem melhor, e na dinâmica das frequências de tarefas ao longo do projeto sejam mais auto-ajustáveis e melhor atendidas.

As contribuições científicas deste trabalho estão voltadas a alocação de recursos e estimativa de esforço em projetos de desenvolvimento de software, assim como em relação a custos, projeto, predição de atendimento de tarefas conforme as demandas do projeto e a quantidade de recursos existentes.

Este trabalho enfoca o estudo de cenários com dados fictícios em relação a ambientes de desenvolvimento de software, que por sua vez, a utilização de dados históricos de projetos reais em empresas, possibilitariam uma melhor simulação, ou seja simulações mais reais, trazendo maiores benefícios a tomada de decisão dos projetos. Para o ensino

de Engenharia de Software, a simulação pode auxiliar em dar uma visão da dinâmica dos sistemas, não apenas em processos de desenvolvimento, mas também em gerenciamento de projetos, permitindo aos alunos um visão mais real dos efeitos de mudanças [Coutinho and Bezerra 2021].

Como trabalhos futuros, pode-se expandir os modelos tornando-os mais complexos e principalmente ampliar os cenários de forma a serem mais robustos e com mais capacidade tanto em relação a recursos como em vezes de replicação das simulações. Outro trabalho a ser conduzido é em relação a estudos com dados históricos, ou seja dados reais de projetos de software que por sua vez podem dar resultados mais completos quanto a estimativas de esforço e risco.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 - Número do Processo 88887-617302/2021-00.

Referências

Alencar, T., Cortés, M., Veras, N., and Magno, L. (2018). An agent-oriented approach for assisting risk management in software projects. *Anais do Computer on the Beach*, pages 791–800.

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

Bouajaja, S. and Dridi, N. (2017). A survey on human resource allocation problem and its applications. *Operational Research*, 17(2):339–369.

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

Coutinho, E. and Bezerra, C. (2021). Simulação de alocação de recursos em projetos de desenvolvimento de software utilizando teoria das filas. In *Anais do III Workshop em Modelagem e Simulação de Sistemas Intensivos em Software*, pages 30–39, Porto Alegre, RS, Brasil. SBC.

Gomes, J. Z., Montenegro, J. L., Santos, J. C. d., Barbosa, J. L. V., and Costa, C. (2019). A strategy using continuous simulation to mitigate effort estimation risks in software projects. *IEEE Latin America Transactions*, 17(8):1390–1398.

Shortle, J. F., Thompson, J. M., Gross, D., and Harris, C. M. (1998). *Fundamentals of queueing theory*, volume 399. John Wiley & Sons.

Zhao, W., Pu, S., and Jiang, D. (2020). A human resource allocation method for business processes using team faultlines. *Applied Intelligence*, 50(9):2887–2900.