



UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UNINDO O COMPORTAMENTO DINÂMICO COM OTIMIZAÇÃO NO
PLANEJAMENTO DA ALOCAÇÃO DE HORAS EXTRAS EM PROJETOS DE
SOFTWARE

Luiz Antonio Oliveira de Araujo Junior

Orientador

Márcio de Oliveira Barros


RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2015

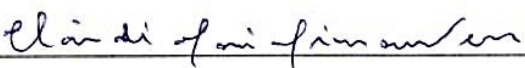
UNINDO O COMPORTAMENTO DINÂMICO COM OTIMIZAÇÃO NO
PLANEJAMENTO DA ALOCAÇÃO DE HORAS EXTRAS EM PROJETOS DE
SOFTWARE

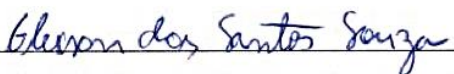
Luiz Antonio Oliveira de Araujo Junior

DISSERTAÇÃO APRESENTADA COMO REQUISITO PARCIAL PARA
OBTENÇÃO DO TÍTULO DE MESTRE PELO PROGRAMA DE PÓS-GRADUAÇÃO
EM INFORMÁTICA DA UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE
JANEIRO (UNIRIO). APROVADA PELA COMISSÃO EXAMINADORA ABAIXO
ASSINADA.

Aprovada por


Prof. Márcio de Oliveira Barros, D.Sc. - UNIRIO


Prof. Cláudia Maria Lima Werner, D.Sc. - UFRJ


Prof. Gleison dos Santos Souza, D.Sc. - UNIRIO

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2015

Araujo Junior, Luiz Antonio Oliveira de.

A663 Unindo o comportamento dinâmico com otimização no
planejamento da alocação de horas extras em projetos de software /
Luiz Antonio Oliveira de Araujo Junior, 2015.

xii, 83 f. ; 30 cm

Orientador: Márcio de Oliveira Barros.

Dissertação (Mestrado em Informática) - Universidade Federal do
Estado do Rio de Janeiro, Rio de Janeiro, 2015.

*A minha mãe que ficaria bastante feliz se pudesse estar aqui para me ver alcançar
mais esta conquista.*

Agradecimentos

A minha esposa pela colaboração e compreensão durante a dedicação empregada para a conclusão deste trabalho.

Ao meu orientador Márcio Barros, que acreditou em minha capacidade e sempre apresentou suas críticas e sugestões sobre os meus textos. Também, por entender as minhas constantes viagens pelo trabalho que impossibilitavam muitas reuniões presenciais, que sorte que temos voip.

Ao meu ex-chefe Marcos Vinicius Pereira Pessoa, pelo incentivo e apoio no início do curso de mestrado.

A todos os meus familiares, amigos e colegas de trabalho, que compreenderam as minhas ausências nos momentos de descontração devido ao aproveitamento do tempo de folga para estudar.

RESUMO

Os profissionais da área de TI são frequentemente submetidos a escalas de trabalho superiores as classificadas como normais (8 horas diárias). Isto ocorre devido às dificuldades que são enfrentadas durante a execução das atividades do processo de desenvolvimento de software. Entretanto, o trabalho em horas extras causa impactos na vida profissional e pessoal dos profissionais de TI. Estes impactos ressaltam a necessidade de uma estratégia de alocação de horas extras que auxilie o gerente do projeto a promover o melhor aproveitamento da menor quantidade de horas extras possível. Pensando nesta estratégia, esta dissertação apresenta uma proposta para o problema de planejamento de horas extras (PPH) para projetos de desenvolvimento de software. A proposta considera os efeitos positivos do uso de horas extras na produtividade, assim como seus efeitos negativos na qualidade do produto. Para capturar estes efeitos, foi desenvolvida uma ferramenta de simulação capaz de emular o comportamento de um modelo que descreve a dinâmica de projetos de desenvolvimento de software, no qual foi aplicado um cenário que descreve a dinâmica do trabalho em horas extras. Aliado à modelagem e simulação dinâmica, a proposta utiliza da Engenharia de Software baseada em Buscas (SBSE) com o algoritmo genético (NSGA-II) para otimizar os resultados (custo, duração e quantidade de horas extras) na busca de soluções próximas ao ótimo para o PPH. Para a avaliação da proposta, um estudo experimental foi realizado com instâncias de projetos reais, no qual comparamos a proposta com práticas utilizadas pela indústria e com uma formulação similar sem os efeitos negativos das horas extras. Os resultados mostraram que desconsiderar os efeitos negativos do trabalho em horas extras pode levar a tomada de decisões equivocadas. Por exemplo, a alocação excessiva de horas extras no cronograma pode fazer com que o gerente subestime o custo e a duração do projeto em 5,9% e 9,2%, respectivamente. Isto ocorre devido às longas atividades de testes que se tornam necessárias para a correção de erros adicionais que foram introduzidos por profissionais cansados. Por fim, as evidências também confirmaram que a estratégia empregada pela indústria que concentra as horas extras na segunda metade do cronograma para compensar os atrasos incorridos em atividades anteriores, produz bons resultados.

Palavras-chave: horas extras, simulação, dinâmica, SBSE, NSGA-II

ABSTRACT

IT professionals are frequently subjected to workload higher than the one classified as normal (8 hours). This occurs due to the difficulties that are faced in carrying out activities of the software development process. However the overtime work impacts on professional and personal lives of IT professionals. These impacts highlight the need for an overtime allocation strategy to assist the project manager in taking the most from as few overtime hours as possible. Thinking about this strategy, this work presents a proposal for the overtime-planning problem (OPP) for software development projects. The proposal considers the positive effects on productivity and the negative effects on product quality, both derived from this excess of working hours. To capture these effects, a simulation tool was developed with continuous time management, which is capable to emulate the behavior, designed to describe the software development projects dynamics, in which we applied the scenario that describes the overtime work dynamics. Combined with dynamic modeling, the proposal uses search-based software engineering (SBSE) with genetic algorithm (NSGA-II) for optimal results (cost, duration and amount of overtime) in solutions for OPP. For the evaluation of the proposal, an experimental study was conducted with real-world project instances, in which we compare the proposal with practices used by the industry and with a similar formulation without the negative effects of overtime. The results showed that the non consideration of the negative effects of overtime work could lead the manager to make wrong decisions. For example, excessive overtime allocation can cause the manager to underestimate the cost and the duration of the project at 5.86% and 9.21%, respectively. This occurs due to the long testing activities that have become necessary to correct the additional errors that were introduced by tired workers. Finally, the evidence also confirmed that the overtime allocation strategy employed by the industry that concentrates overtime in the second half of the schedule to compensate for delays produces good results.

Keywords: overtime, simulation, dynamics, SBSE, NSGA-II

Capítulo 1 - Introdução	1
1.1. Objetivos	2
1.2. Justificativa	3
1.3. Organização	4
Capítulo 2 - A Modelagem de Horas extras para Projetos e a Engenharia de Software Baseada em Buscas.....	5
2.1. O Trabalho em Horas extras e Seus Efeitos.....	5
2.2. A Modelagem Dinâmica e os Modelos de Horas extras em Projetos de Desenvolvimento de Software	9
2.2.1. A Dinâmica de Sistemas no Contexto de Projetos de Software	9
2.2.2. A Modelagem segundo a Dinâmica de Sistemas	11
2.2.3. Efeitos do Trabalho em Horas extras sobre o Projeto	16
2.3. A Otimização da Gerência de Tempo em Projetos	18
2.3.1. A Engenharia de Software baseada em Buscas	18
2.3.2. Aplicação da SBSE no Gerenciamento de Projetos de Software	20
2.4. Considerações Finais	24
Capítulo 3 - Simulação do Modelagem de Dinâmica com Otimização Heurística	25
3.1. Formalização do Problema	26
3.1.1. A Dinâmica de um Projeto de Desenvolvimento de Software	26
3.1.2. A Dinâmica do Trabalho em Horas extras	30
3.1.3. A Necessidade de um Mecanismo de Simulação	31
3.1.4. A Necessidade de Otimização Heurística.....	32
3.1.5. Os Objetivos da Otimização Heurística	33
3.2. Proposta de Solução	34
3.2.1. O Simulador.....	34
3.2.2. Objetos de Simulação e Recursos para Projetos.....	36
3.2.3. Cenário de Horas extras.....	39
3.2.4. A Otimização.....	40
3.3. Considerações Finais	43
Capítulo 4 - A Avaliação da Proposta de Solução.....	45
4.1. Questões de Pesquisa	45
4.2. Objetos de Estudo	47
4.3. Execução dos Experimentos e Análise dos Resultados	49

4.3.1.	Definição e Cálculo dos Indicadores de Qualidade.....	49
4.3.2.	Parametrização do Algoritmo.....	51
4.3.2.1.	Número Máximo de Avaliações.....	52
4.3.2.2.	Tamanho de População.....	54
4.3.3.	A Análise dos Resultados.....	56
4.3.3.1.	Resposta ao Teste de Sanidade (QP1).....	57
4.3.3.2.	Resposta ao Teste de Competitividade (QP2).....	58
4.3.3.3.	Resposta ao Teste de Consistência (QP3).....	61
4.3.3.4.	Consolidando Resultados.....	64
4.4.	Ameaças à Validade.....	67
4.4.1.	Ameaças à Validade de Conclusão.....	68
4.4.2.	Ameaças à Validade Interna.....	69
4.4.3.	Ameaças à Validade de Construção.....	70
4.4.4.	Ameaças à Validade Externa.....	72
4.5.	Considerações Finais.....	73
Capítulo 5 -	Conclusões.....	75
5.1.	Contribuições.....	76
5.2.	Limitações e Perspectivas de Pesquisas Futuras.....	76
Referências Bibliográficas.....		78

Índice de Figuras

Figura 1 - Representação das estruturas básicas de Dinâmica de Sistemas com a ferramenta Stella.....	13
Figura 2 - Representação das estruturas básicas de Dinâmica de Sistemas com a ferramenta Vensim.	13
Figura 3 - Modelo do comportamento da população em função das taxas de nascimento e morte.....	14
Figura 4 - Gráfico comparativo dos comportamentos da população pelo tempo com as combinações dos parâmetros do modelo não linear	14
Figura 5 - Modelo simplificado de um projeto de desenvolvimento de software	15
Figura 6 - Diagrama de Blocos de uma estrutura genérica de SBSE (baseada na Figura 2 de Harman; Mansouri e Zhang (2012))	20
Figura 7 - Um exemplo de dependências entre grupos de funcionalidades	26
Figura 8 - Ordem de precedência dos tipos de atividades	27
Figura 9 - Dependência entre Grupos de Funcionalidades (GF1 e GF2)	27
Figura 10 - Gráfico do multiplicador da taxa de geração de erros por carga horária diária	31
Figura 11 - Modelo de dinâmica de projeto de desenvolvimento de software.....	38
Figura 12 - Modelo de dinâmica de projeto de desenvolvimento de software considerando a geração de erros adicionais pelo trabalho em horas extras.	40
Figura 13 - Gráfico das soluções da otimização bi-objetivo com a Curva de Pareto.....	42
Figura 14 - Representação do <i>Contributions</i>	50
Figura 15 - Representação do <i>Hypervolume</i>	50
Figura 16 - Representação do <i>Generational Distance</i>	51
Figura 17 - Representação do <i>Spread</i>	51
Figura 18 - Diagrama com todos os resultados de todas as configurações, instâncias e indicadores de qualidade. NSGA-II por <i>boxplot</i> , Margarine por X, Second Half por * e CPM por +.	61
Figura 19 - Projeções 2D para as combinações da quantidade de horas extras, duração e custo para todas as instâncias. É possível observar as relações lineares resultantes do NSGA-II _{SE} e o formato convexo do NSGA-II, assim como as poucas soluções identificadas a partir das estratégias utilizadas pela indústria. MAR (○), SH (△) e CPM (+) são representados com os símbolos na cor preta, NSGA-II (°) com o símbolo em cinza claro e NSGA-II _{SE} (°) com o símbolo em cinza escuro.	65

Índice de Tabelas

Tabela 1 - Taxas de erros introduzidos por ponto de função e percentual de erros regenerados por fase do projeto.....	28
Tabela 2 - Taxas de alocação de esforço pelas fases do projeto de desenvolvimento de software (Jones, 2007, pg 97).....	29
Tabela 3 - Resumo das Instâncias	48
Tabela 4 - Valores de média e desvio padrão do I_{GD} para as configurações elencadas para o número máximo de avaliações em todas as instâncias, marcando em negrito os melhores valores.....	53
Tabela 5 - Valores de média e desvio padrão do I_{SP} para as configurações elencadas para o número máximo de avaliações em todas as instâncias, marcando em negrito os melhores valores.....	53
Tabela 6 - Indicação de semelhança nos resultados entre as configurações de número máximo de avaliações para todas as instâncias considerando o I_{GD}	53
Tabela 7 - Indicação de semelhança nos resultados entre as configurações de número máximo de avaliações para todas as instâncias considerando o I_{SP}	54
Tabela 8 - Valores de média e desvio padrão do I_{GD} para as configurações elencadas para o parâmetro de tamanho da população em todas as instâncias, marcando em negrito os melhores valores.	55
Tabela 9 - Valores de média e desvio padrão do I_{SP} para as configurações elencadas para o parâmetro de tamanho da população em todas as instâncias, marcando em negrito os melhores valores.	55
Tabela 10 - Indicação de semelhança nos resultados entre as configurações de tamanho da população e todas as instâncias considerando o I_{GD}	56
Tabela 11 - Indicação de semelhança nos resultados entre as configurações de tamanho da população e todas as instâncias considerando o I_{SP}	56
Tabela 12 - Resultados do p -value (PV) do teste de Wilcoxon-Mann-Whitney e o tamanho de efeito (ES) de \hat{A}_{12} Vargha-Delaney comparando os resultados da solução proposta e do algoritmo aleatório para todas as instâncias e indicadores de qualidade.....	58
Tabela 13 - Valores de médias e desvios padrão para todos os indicadores de qualidade em todas as instâncias para o NSGA-II e valores observados para as estratégias de gerenciamento de horas extras utilizadas pela indústria (MAR, SH e CPM), marcando em negrito os melhores valores.	59
Tabela 14 - O tamanho de efeito \hat{A}_{12} de Vargha-Delaney considerando o algoritmo NSGA-II e cada uma das estratégias de alocação de horas extras praticadas pela indústria para todas as instâncias e indicadores de qualidade. O tamanho de efeito mostra as chances do NSGA-II obter um maior valor para o I_C e o I_{HV} e um menor valor para o I_{SP} e o I_{GD} , mostrando a superioridade do NSGA-II.	60
Tabela 15 - Médias e desvios padrão comparando o NSGA-II e o NSGA-II _{SE} para todas as instâncias e indicadores de qualidade, marcando em negrito os melhores valores....	62
Tabela 16 - Resultados do p -value (PV) do teste de Wilcoxon-Mann-Whitney e o tamanho de efeito (ES) \hat{A}_{12} de Vargha-Delaney comparando o NSGA-II e o NSGA-II _{SE} para todas as instâncias e indicadores de qualidade. O tamanho de efeito mostra as chances do NSGA-II obter um maior valor para o I_C e o I_{HV} e um menor valor para o I_{SP} e o I_{GD} , mostrando a superioridade do NSGA-II _{SE}	62

Tabela 17 - Comparativo do número de soluções geradas pelo NSGA-II e pelo NSGA-II _{SE} para todas as instâncias.	63
Tabela 18 - Comparativo da diferença média da duração e do custo para as soluções envolvendo excessiva alocação de horas extras para todas as instâncias, considerando os algoritmos NSGA-II e NSGA-II _{SE}	64
Tabela 19 - Comparativo do custo de redução de duração de cronograma considerando dois períodos de 10 dias, chegando a duração mínima encontrada.	66

Capítulo 1 - Introdução

Um projeto é definido como um esforço com duração definida para criar um produto, serviço ou resultado, conforme seus requisitos. Para cumprir estes requisitos, é necessário executar atividades aplicando técnicas, ferramentas, habilidades e conhecimento, ou seja, aplicando o gerenciamento de projetos (HYMAN *et al.*, 2003), (PMI, 2013).

Estão incluídos nas atividades da área de conhecimento de planejamento, os processos que envolvem o cronograma, a carga horária de trabalho, alocação de recursos humanos e as atividades que serão utilizadas no projeto (PMI, 2013). Porém, quando os gerentes não realizam o planejamento de forma adequada ou quando imprevistos são encontrados durante o projeto, os desenvolvedores precisam executar o trabalho em um prazo mais curto, tipicamente realizando o trabalho em horas extras. Esta situação para os profissionais de TI, por ser frequente, torna a vida pessoal e profissional bastante estressante (HOUDMONT; ZHOU; HASSARD, 2011).

Devido a frequência com que os profissionais de TI são submetidos ao trabalho em horas extras, a qualidade do trabalho produzido é diminuída por causa do aumento da taxa de geração de erros (DEMARCO, 1982)(AKULA; CUSICK, 2008). Estes erros adicionais, decorrentes do estado de atenção reduzida provocado pelo cansaço imposto aos desenvolvedores, farão com que o projeto tenha um acréscimo de custo para que as correções possam ser efetuadas e testadas (CARUSO, 2006). Os efeitos negativos do trabalho excessivo sobre o estado de atenção de trabalhadores são amplamente documentados. Por exemplo, testes simulados de direção mostraram que profissionais que trabalharam 80 horas por semana apresentaram um comportamento comparado ao de motoristas com nível de concentração de álcool no sangue equivalente a 0,04% e 0,05% (por 100 ml de sangue) (ARNEDT *et al.*, 2005).

Os efeitos positivos e os negativos da realização do trabalho em horas extras vêm sendo modelados para projetos de TI, retratando o aumento de produtividade, de fadiga, de geração de erros e de retrabalho pela geração de produtos defeituosos (JIA; FAN; LU,

2007) (MEILONG; LI; CHEN, 2008). Mostrando não ser particularidade de projetos de TI, os modelos de projetos de construção civil modelam os mesmos comportamentos nocivos do trabalho em horas extras (ZHAO; LV; YOU, 2007).

A busca por uma alocação minimizada de horas extras para os projetos de TI se caracteriza como um problema de otimização. Este problema envolve o balanceamento de objetivos de gerenciamento de projetos (prazo e custo), combinado com o objetivo de minimizar a quantidade de horas extras. As possíveis combinações de quantidades de horas extras que podem ser atribuídas a cada atividade do cronograma formam um enorme espaço de busca, que não pode ser completamente examinado em tempo aceitável para encontrar soluções ótimas para este problema. Para este tipo de problema, a utilização de técnicas de otimização heurística é necessária para encontrar boas soluções (HARMAN; MANSOURI; ZHANG, 2012). A indústria também vem utilizando estas técnicas devido aos seus bons resultados, além de sua ampla utilização no meio acadêmico (HARMAN; MANSOURI; ZHANG, 2012).

Sendo assim, buscamos as quantidades de horas extras a serem alocadas em um projeto de TI que possam ser consideradas no planejamento do projeto. Neste planejamento estará detalhada a quantidade de horas extras para cada uma das atividades do projeto, de forma que possibilite que os profissionais se preparem para as jornadas de trabalho que irão desempenhar e o projeto tenha uma estimativa que considere os efeitos positivos e negativos causados pelos trabalhos em horas extras.

1.1. Objetivos

Como objetivo geral da Dissertação temos a formulação de uma proposta para o problema de planejamento de horas extras para projetos de desenvolvimento de software que considere os efeitos do cansaço dos desenvolvedores trabalhando em horas extras e realize o balanceamento entre o número de horas extras trabalhadas em um projeto, sua duração e seu custo. Para atingir o objetivo geral, elencamos três objetivos específicos:

1. Buscar na literatura atual pelo estado da arte relacionado aos assuntos considerados na Dissertação: Dinâmica de Sistemas, modelos dos efeitos do trabalho em horas extras em projetos de desenvolvimento de software e Engenharia de Software baseada em Buscas (SBSE);

2. Apresentar a proposta de solução para o problema do planejamento de horas extras para projetos de desenvolvimento de software, mostrando sua evolução em relação ao estado da arte da literatura;
3. Avaliar a proposta de solução por meio de um estudo experimental. Esta avaliação envolve uma comparação da proposta com práticas utilizadas na indústria para o planejamento da alocação de horas extras.

1.2. Justificativa

O gerenciamento de projetos é o conjunto de técnicas que buscam alcançar o sucesso dos projetos por meio da execução de atividades pelos recursos durante todo o ciclo de vida de um projeto. Na fase inicial dos projetos, durante o planejamento, os tomadores de decisões normalmente consideram a carga horária de trabalho diária normal, ou seja, oito horas diárias de trabalho, deixando a opção de utilização de horas extras para uma eventual necessidade de ajuste de cronograma do projeto. Entretanto, durante a execução de projetos, imprevistos podem ocorrer, resultando em retrabalho, atraso e necessidade de maior tempo para conclusão de atividades, consequentemente gerando impactos no projeto.

Quando o projeto é afetado e sua duração não pode ser alterada, a primeira solução do gerente é aumentar a carga de trabalho diária para suprir a necessidade de horas de trabalho para concluir as atividades necessárias. Entretanto, estes acréscimos também geram impactos na vida do profissional. Mahadik (2014) ressaltou um grande diferencial para os projetos de TI, uma vez que existe uma dependência elevada nas pessoas que desempenham as atividades. Desta forma, percebemos que é necessário considerar a alocação dos profissionais para modelar e otimizar o planejamento e o cronograma de um projeto de software, ou seja, a carga de trabalho.

Buscando contabilizar os efeitos do trabalho em horas extras e encontrar uma alocação otimizada, Ferrucci *et al.* (2013) buscaram um balanceamento entre o aumento da carga horária de trabalho e os riscos dos projetos. Os autores utilizaram técnicas de otimização heurística e comparações com práticas classificadas como utilizadas atualmente pela indústria. Entretanto, neste modelo os autores não consideraram os efeitos nocivos do trabalho em horas extras, nem a dinâmica da geração de erros, se restringindo ao ganho de produtividade pelo aumento de horas de trabalho.

Para apresentar os efeitos negativos do trabalho em horas extras, evoluiremos o trabalho de Ferrucci *et al.* (2013) considerando uma abordagem de otimização e a modelagem da dinâmica do trabalho em horas extras, assim como a geração de erros, ambas com base no modelo proposto por Abdel-Hamid e Madnick (1991). A Dissertação visa cobrir esta limitação do trabalho anterior que não combinou a modelagem dinâmica do projeto de software com a utilização de otimização heurística. Desta forma, conforme os objetivos definidos na seção anterior, a Dissertação apresenta uma evolução do estado da arte em relação aos trabalhos relacionados.

1.3. Organização

A Dissertação está organizada em cinco capítulos, dos quais o primeiro compreende esta introdução, que contém os objetivos, a justificativa e a organização.

O segundo capítulo contém a revisão da literatura, apresentando o trabalho em horas extras e seus efeitos positivos e negativos, a modelagem dinâmica de projetos de desenvolvimento de software e os modelos do trabalho em horas extras neste contexto, assim como as aplicações de otimização heurística na gerência de tempo em projetos.

O terceiro capítulo contém a proposta de solução para o problema de planejamento de horas extras para projetos de desenvolvimento de software que considera os efeitos do cansaço dos desenvolvedores trabalhando em horas extras, realizando o balanceamento entre o número de horas extras trabalhadas em um projeto, a sua duração e o seu custo.

O quarto capítulo apresenta as questões de pesquisa, os objetos de estudo, a execução dos experimentos e a análise dos resultados, assim como as ameaças à validade deste estudo experimental que visa a avaliação da proposta de solução do terceiro capítulo.

O quinto capítulo apresenta as considerações finais do trabalho, ressaltando suas contribuições, limitações e as perspectivas para pesquisas futuras.

Capítulo 2 - A Modelagem de Horas extras para Projetos e a Engenharia de Software Baseada em Buscas

Com o objetivo de identificar as pesquisas realizadas em áreas relacionadas com o tema desta Dissertação foi realizada uma revisão bibliográfica em artigos, Teses, Dissertações e livros focando nas áreas de trabalho em horas extras, modelagem de projetos de software e Engenharia de Software baseada em Buscas.

Este capítulo está organizado em quatro seções, iniciando pelo levantamento dos trabalhos relacionados com a realização de trabalho excedendo as horas normais e os efeitos positivos e negativos causados por este excesso de trabalho. Na segunda seção temos exemplos de modelagem dinâmica para a carga horária de trabalho em projetos de desenvolvimento de software, considerando horas extras. Na terceira seção foram reunidas pesquisas que envolvem otimização e projetos de desenvolvimento de software. A quarta seção apresenta as considerações finais do capítulo.

2.1. O Trabalho em Horas extras e Seus Efeitos

Nishikitani *et al.* (2005) definem horas extras como a realização de qualquer trabalho após a jornada de trabalho normal, prevista no acordo ou contrato de trabalho entre o profissional e o empregador. A realização de trabalhos excedendo a carga horária normal, ou seja, em horas extras, pode refletir na qualidade do trabalho desempenhado e na vida do profissional.

Com a evolução da tecnologia e informática, houve um aumento na quantidade de empresas de software, assim como no número de profissionais desta área e, simultaneamente, o aumento de ocorrência de stress ocupacional (FUJIGAKI; ASAKURA; HARATANI, 1994). Selye Hans, em 1936, definiu o stress como "a resposta não específica do organismo a qualquer exigência colocada contra ele". Como observado por Robbins *et al.* (1998), stress tem sido reconhecido como um dos problemas cruciais

dos locais de trabalho em diferentes países ao redor do globo. Diversos estudos mostraram que o *stress* ocupacional pode levar a várias consequências negativas para o indivíduo e para o local de trabalho (OGIŃSKA-BULIK, 2006). Conforme observado por Fujigaki, Asakura e Haratani (1994), este *stress* é frequentemente atribuído à carga de trabalho e fatores organizacionais relacionados com o gerenciamento de projetos.

Considerando os efeitos na vida do profissional de Tecnologia da Informação (TI), a realização de trabalho em horas extras nem sempre se reflete homogeneamente por toda a equipe, uma vez que cada indivíduo tem seus compromissos intransferíveis ou necessidades em relação a família, como por exemplo um filho pequeno. A realização heterogênea de horas extras pela equipe por um curto espaço de tempo é geralmente passível de ser tolerada. Porém, quando uma equipe mantém determinados membros realizando continuamente uma jornada de trabalho agressiva do ponto de vista da carga horária, enquanto que outros continuam em sua vida normal ou levemente afetados, a tolerância usualmente dá lugar à rivalidade. Este tipo de ocorrência é classificado por DeMarco e Lister (1999) como um efeito “equipecida”, ou seja, resulta na dissolução da equipe pela quantidade de conflitos gerados.

Profissionais da China da área de TI foram identificados como praticantes de excessivas quantidades de horas extras, o que culminou na ocorrência de níveis de bem-estar psicológico significativamente mais baixos em relação àqueles que executaram poucas horas extras (HOUDMONT; ZHOU; HASSARD, 2011). Karita *et al.* (2006) acrescentam que uma longa jornada de trabalho combinada com a diminuição do período de sono provoca exaustão, mudança de humor, depressão e doenças do coração. Em um estudo realizado com profissionais japoneses das áreas de consultoria, integração de sistemas e gerenciamento de dados, foi observado que estas exigências de extrema carga de trabalho são motivadas pela competição entre empresas. Em um estudo realizado com profissionais da área de TI de Singapura, Lim e Teo (1999) mostraram que a necessidade de realizar trabalhos em casa, fora do horário definido no contrato de trabalho, representa a maior causa de *stress*. Devido a esta carga de trabalho, muitos trabalhadores se sentem culpados pela negligência com a família. Conforme Woo *et al.* (1989), funcionários de TI algumas vezes encontram problemas que requerem solução que ultrapassa as horas normais de sua jornada de trabalho.

A intensidade de trabalho afeta tanto os profissionais de TI quanto das demais áreas. Prathibha, Ravichandran e Johnson (2013) compararam os níveis de *stress* entre os profissionais de TI e os professores, mostrando que ambos têm o nível de *stress*

aumentado com a intensidade de trabalho e pressão dos colegas. Apesar de áreas distintas, estes profissionais apresentam problemas semelhantes, ou seja, áreas distintas da TI também sofrem reflexos do ritmo do trabalho.

Considerando os trabalhadores dos Estados Unidos da América, sem restrição de área, Caruso (2006) ressalta que longas jornadas de trabalho são comuns e que cerca de 26% dos homens e 11% das mulheres trabalharam, aproximadamente, 50 horas ou mais por semana no ano 2000. Kuhn e Lozano (2005) vão além e constatarem que estes trabalhadores, atualmente, são mais propensos a trabalhar 50 ou mais horas por semana do que no quarto de século anterior.

Fischer *et al.* (2000) relataram que, para os trabalhadores brasileiros de usinas petroquímicas, as longas jornadas de trabalho refletiram na diminuição significativa do estado de alerta após a décima hora de trabalho consecutivo, tanto para turnos diurnos quanto noturnos. Mitchell e Williamson (2000) também reportaram que, para os trabalhadores de usinas hidroelétricas da Austrália, houve um aumento nos erros das tarefas de vigilância ao final dos turnos de doze horas, quando comparados com atividades similares desempenhadas no início dos turnos. Estes efeitos não eram reportados para os turnos de oito horas diárias, tanto diurnos quanto noturnos.

Nishikitani *et al.* (2005) identificaram que, mostrando preocupação com o aumento da carga horária para trabalhadores do Japão, foram realizados estudos que evidenciaram que este aumento ocorre com maior frequência entre os profissionais jovens e de meia-idade, principalmente em grandes empresas. Foi observada uma forte correlação positiva entre o aumento na quantidade de horas extras e o risco de doenças cerebrais e cardíacas, além de medidas de depressão e agressividade. Desta forma, um limite de segurança de 45 horas extras por mês vem sendo considerado em muitas empresas japonesas.

Apesar dos diversos estudos relatados acima, percebe-se ainda uma grande dificuldade em classificar os problemas de perda de qualidade no trabalho e o aparecimento de doenças como efeitos causados pelo trabalho em horas extras, pois normalmente são atribuídos ao estilo de vida do profissional (NISHIKITANI *et al.*, 2005). De acordo com Liu *et al.* (2002), foi identificada uma associação entre infarto agudo no miocárdio e a realização de trabalho frequente em regime de horas extras: muitos trabalhadores que sofreram infarto haviam realizado horas extras no mês anterior à data da ocorrência do infarto.

Arnedt *et al.* (2005) mostraram que médicos residentes que trabalhavam em longas jornadas de trabalho, chegando a 80 horas por semana, apresentavam grandes dificuldades em manter atenção e vigilância em suas atividades. Em testes simulados de direção, os resultados foram comparados aos de motoristas com nível de concentração de álcool no sangue equivalente a 0,04% e 0,05% (para cada 100 ml de sangue). Outro resultado observado foi que, no ápice do trabalho, os profissionais já não reconheciam a extensão dos danos provocados em suas funções.

Considerando o desenvolvimento de software, quando profissionais trabalham frequentemente em horas extras, observa-se um aumento na taxa de geração de erros (DEMARCO, 1982)(AKULA; CUSICK, 2008). DeMarco e Lister (1999) também relacionam as horas extras com o aumento de atrito entre os componentes da equipe. Estas condições podem reduzir a troca de conhecimento entre os desenvolvedores, levando alguns profissionais a necessitar de mais tempo para conclusão de suas atividades. Ao analisar os aspectos de produtividade, ambiente de trabalho e os demais causadores de *stress*, Donald *et al.* (2005) encontraram correlação negativa entre *stress* e produtividade, ou seja, o *stress* causado pelo excesso de trabalho em horas extras diminui a produtividade do profissional. Caruso (2006) ressaltou ainda que, em virtude da geração adicional de erros devido à perda de qualidade do trabalho desempenhado pelos profissionais submetidos a excesso de trabalho, o projeto poderá ter um acréscimo de custo para que estas correções possam ser efetuadas e testadas.

Após observar as pesquisas citadas acima, constatou-se a frequente ocorrência de trabalho em horas extras por pessoas de diferentes nacionalidades e áreas de atuação. Independentemente de serem brasileiros, norte-americanos, australianos, singapurianos chineses, japoneses, trabalharem na área de saúde, de tecnologia ou de petróleo, estes profissionais estão sujeitos aos efeitos nocivos à saúde do trabalho excessivo em horas extras, assim como aos efeitos prejudiciais à vida profissional. Dentre os efeitos nocivos ligados diretamente à saúde do profissional, podemos destacar os problemas para dormir, o cansaço excessivo, as doenças cerebrais e cardíacas. Em relação aos efeitos nocivos ligados ao trabalho, tem-se a perda de qualidade dos produtos, a dificuldade em resolver problemas simples e a rivalidade com consequente dissolução das equipes.

Nos casos estudados, a necessidade de realização de horas extras se manifestava de forma súbita ou com pouco planejamento prévio, gerando assim impactos para o trabalhador e para o projeto. Desta forma, percebemos que a existência de algum planejamento para a realização de horas extras poderia mitigar estes impactos, permitido

que o trabalhador se organizasse para a realização das atividades em regime de trabalho excepcional e o projeto se mantivesse dentro do previsto em termos de custo e prazo. Visando facilitar este planejamento, precisamos modelar os efeitos do trabalho em horas extras sobre um projeto e permitir que o gerente de projetos analise diversos cenários de trabalho neste regime. Para tanto, lançamos mão de modelagem em projetos de desenvolvimento de software, que será o assunto discutido na próxima seção.

2.2. A Modelagem Dinâmica e os Modelos de Horas extras em Projetos de Desenvolvimento de Software

Nos projetos de desenvolvimento de software, os trabalhadores são frequentemente submetidos a jornadas de trabalho que excedem o horário normal, conforme citado na seção anterior. Estes efeitos, somados aos demais fatores dos projetos, contribuem para o comportamento diferenciado em cada projeto. Para descrever formalmente o efeito destes diferentes fatores, inclusive, mas não limitado ao trabalho em horas extras, foram criados modelos capazes de capturar seus efeitos sobre o comportamento do projeto.

Considerando que um sistema é um conjunto de elementos interconectados de modo a formar um todo organizado, podemos depreender que um projeto de desenvolvimento de software pode ser extrapolado na forma de um sistema. Existem diversas correntes para a modelagem de sistemas. Dentre elas, temos a Dinâmica de Sistemas, que foi criada na década de 1960 e, inicialmente, batizada como Dinâmica Industrial por Jay Forrester. A Dinâmica Industrial foi baseada nos conceitos da dinâmica não linear e do relacionamento entre a matemática, física e engenharia. Ao evoluir para Dinâmica de Sistemas, se tornou um método para ajudar as pessoas a aprender a complexidade do comportamento dos sistemas ao longo do tempo, fornecendo aos gerentes de projetos uma forma clara de entender os eventos do presente e do futuro (FORRESTER, 1961).

2.2.1. A Dinâmica de Sistemas no Contexto de Projetos de Software

As práticas tradicionais de gerenciamento de projetos, conforme Cao, Ramesh e Abdel-Hamid (2010), consideram um projeto como uma série de atividades, geralmente classificadas em grandes grupos, como planejamento, implementação e controle. Temos como exemplo destas práticas tradicionais, a técnica PERT (*Program Evaluation and Review Technique*) e o método CPM (*Critical-path Method*). A técnica PERT é baseada

na disposição das atividades de forma ordenada no tempo, sendo utilizada para calcular o tempo total do projeto com base no tempo necessário para a execução destas atividades. Este tempo é estimado por técnicos, os quais espera-se que tenham conhecimento das atividades que precisam ser realizadas. Como esta estimativa é realizada com base principalmente na experiência do profissional, uma parcela de incerteza é acrescentada, gerando uma distribuição de probabilidade para representar as estimativas de conclusão e não um valor único para a estimativa. Como o projeto possui diversas atividades com interdependências, estas distribuições de probabilidade propagam as incertezas no cronograma do projeto. Além do controle tradicional, com estimativas e prazos de forma independente para todas as atividades, o método CPM determina o caminho crítico do projeto, mapeando as atividades que se encontram em uma condição na qual, caso ocorra algum atraso o prazo final do projeto, será necessariamente alterado (FAZAR, 1959).

A combinação do PERT com o CPM foi realizada para viabilizar o controle, focando nas atividades e no tempo gasto para a sua execução pelos desenvolvedores na rotina diária dos projetos. Mesmo assim, o gestor, buscando por antecipação ou retomada dos prazos, pode arbitrariamente alocar tempo superior a rotina diária normal de trabalho, incorrendo na realização de horas extras. Sendo assim, vemos que o gestor busca com estas ferramentas somente a redução de prazos com a alocação de horas extras, não capturando os impactos negativos da dinâmica da produtividade e geração de erros, devido ao cansaço dos profissionais envolvidos no projeto em decorrência do trabalho em horas extras.

De acordo com Barros (2001), determinadas características de um projeto somente podem ser avaliadas por meio da integração de um modelo que represente o projeto e suas características. As características inerentes aos projetos de desenvolvimento de software são evidenciadas durante a execução do projeto, independente do comportamento definido pelo gerente do projeto, e seus impactos necessitam ser mensurados para que decisões possam ser tomadas de forma mais adequada. A recorrência de situações práticas que diferem do esperado pelos gerentes evidencia que os projetos de desenvolvimento de software apresentam um comportamento dinâmico complexo, difícil de prever, durante a sua execução.

Para a realização do mapeamento da dinâmica citada nos parágrafos anteriores, temos uma técnica de modelagem, a Modelagem de Dinâmica de Projetos de Software. Esta técnica não é fácil, inibindo assim a sua utilização prática para o gerenciamento de projetos. Dentre estas dificuldades destacam-se a dificuldade de construção dos modelos,

a incapacidade de representar detalhes e as limitações na representação de conhecimento (BARROS, 2001).

Visando mitigar estas limitações, associa-se a simulação com a Modelagem Dinâmica, possibilitando assim o entendimento do comportamento dinâmico de projeto de desenvolvimento de software. Dentre a variedade de aspectos do desenvolvimento de software, a simulação foi usada para investigar a confiabilidade, conforme Rus, Collofello e Lakey (1999), o gerenciamento de requisitos, conforme Höst *et al.* (2001), e os testes, conforme Caruso (2006), entre outros. Com a simulação, criamos um laboratório para a realização de estudos da variabilidade e complexidade associada a estes projetos e suas características.

O comportamento dinâmico de um projeto de desenvolvimento de software foi modelado por Abdel-Hamid e Madnick (1991) utilizando a técnica de Dinâmica de Sistemas. Eles realizaram um estudo por meio de entrevistas com 27 profissionais de 3 empresas diferentes, focando na área de tecnologia de informação, mais especificamente em profissionais de projetos de desenvolvimento de software. Este estudo, junto com uma análise da literatura, permitiu a identificação de diversos relacionamentos entre decisões e ações gerenciais ocorridas no decorrer de um projeto de software. Concluído o estudo, os modelos foram desenvolvidos a partir dos relacionamentos identificados, de forma que permeassem as fases de planejamento, controle, desenvolvimento, testes, avaliação de qualidade e auditoria de um projeto. Focando a fase de desenvolvimento, podemos destacar os modelos que envolvem o gasto de tempo com a comunicação entre os profissionais, proporcionalmente, ao tamanho da equipe e o efeito na produtividade do cansaço resultante de uma excessiva carga horária de trabalho.

2.2.2. A Modelagem segundo a Dinâmica de Sistemas

A modelagem dinâmica utiliza os princípios e técnicas dos sistemas de controle de retroalimentação para representar os comportamentos dinâmicos de sistemas, cobrindo tanto sistemas simples quanto complexos, envolvendo desde aspectos técnicos até socioeconômicos. A utilização da modelagem vem sendo amplamente estudada em áreas como planejamento urbano, políticas públicas, economia, tomada de decisões, saúde e educação (CAO; RAMESH; ABDEL-HAMID, 2010).

Na modelagem dinâmica, existem dois tipos de modelos, os modelos de tempo discreto e os modelos de tempo contínuo, conforme Barros (2001), e suas diferenças são relacionadas com a forma com que as mudanças no estado dos componentes do modelo

ocorrem. Nos modelos discretos, as mudanças ocorrem por meio de eventos que modificam os valores das variáveis do modelo, podendo gerar novos eventos em intervalos de tempo não previamente determinados e provocando mudanças de estado sem um período constante. Nos modelos contínuos, os valores das variáveis e os estados dos componentes são alterados em intervalos de tempo constantes, previamente definidos e, geralmente, infinitesimais.

A Dinâmica de Sistemas é uma técnica de modelagem que permite a construção de modelos de tempo contínuo. Estes modelos são considerados úteis para capturar o comportamento dinâmico de um projeto de desenvolvimento de software, pois viabilizam a simulação da rotina diária de um projeto utilizando intervalos de tempo bastante reduzidos. Segundo o guia de aprendizado de Dinâmica de Sistemas (MIT, 2005), os modelos contínuos podem ser construídos em diversas ferramentas, como, por exemplo, Stella¹, Vensim², Powersim³ e Dynamo⁴. Nestas ferramentas, é possível, além da construção, a execução de rodadas de simulação.

Em qualquer ferramenta de modelagem de Dinâmica de Sistemas teremos parâmetros de controle, como o tempo inicial, o tempo final, o passo e o tempo atual. O tempo inicial representa o valor inicial para iniciar a execução do modelo, como por exemplo, o primeiro dia. O tempo final representa o valor final para encerrar a execução do modelo, como por exemplo, o centésimo dia. O passo representa o acréscimo ou o decréscimo realizado em cada instante da execução do modelo, como por exemplo, a cada instante de execução do modelo ocorrerá o incremento de um dia. O tempo atual representa o instante de tempo da execução do modelo, como por exemplo após a execução de três instantes de um modelo com tempo inicial igual ao primeiro dia e o passo igual a um dia, teremos o tempo atual no terceiro dia.

Nos modelos desenvolvidos segundo a Dinâmica de Sistemas, existem quatro tipos de componentes: acumuladores (*stock*), taxas (*flow*), auxiliares (*converter*) e constantes. O acumulador representa qualquer coisa que possa ser acrescida ou diminuída de quantidade, como, por exemplo, a população de uma cidade. A taxa representa uma alteração realizada sobre um acumulador em cada instante de tempo da execução do

¹ <http://www.iseesystems.com/>

² <http://www.vensim.com/>

³ <http://www.powersim.com/>

⁴ Sistema fora de comercialização

modelo, seja positivamente ou negativamente, como, por exemplo, as taxas de nascimentos e de mortes. Auxiliares são responsáveis por armazenar valores que serão alterados durante a simulação com base em operações com outros componentes, como a quantidade de nascimento de pessoas em relação a população atual de uma cidade. Constantes armazenam valores que não serão alterados durante a simulação e que podem ser utilizados pelos demais componentes, como o número de nascimentos por dia. Finalmente, para que dois componentes possuam um relacionamento é necessária a utilização de conectores, pois assim é possível mencionar o primeiro componente (origem do conector) na equação que descreve o segundo componente (destino do conector).

Estes componentes são representados visualmente por blocos de forma diferente nas diversas ferramentas existentes, conforme pode ser percebido nos modelos iguais apresentados na Figura 1 e na Figura 2, construídos nas ferramentas Stella e Vensim, respectivamente.

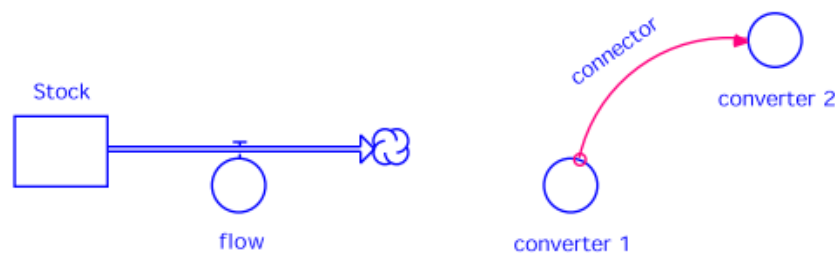


Figura 1 - Representação das estruturas básicas de Dinâmica de Sistemas com a ferramenta Stella.

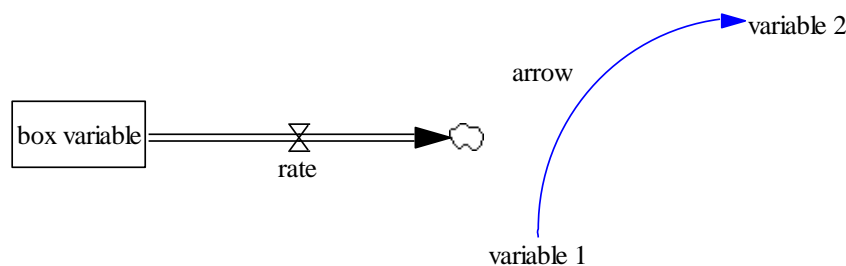


Figura 2 - Representação das estruturas básicas de Dinâmica de Sistemas com a ferramenta Vensim.

Na Figura 3, apresenta-se um modelo simples, construído a partir dos componentes acumulador e taxa, que representa o comportamento de uma população em relação às taxas de nascimentos e de mortes. Como o número de nascimentos e de mortes tanto dependem como influenciam no tamanho da população, isto representa uma retroalimentação. A retroalimentação que um sistema pode sofrer durante sua simulação reflete em seus acumuladores. Tomando como exemplo o modelo de população da Figura 3, à medida que a população aumenta, a taxa de nascimentos aumenta proporcionalmente.

Este exemplo de retroalimentação é chamado de *feedback* positivo, pois um aumento no acumulador reflete em um aumento na taxa que o alimenta, provocando assim um aumento ainda maior. Existe também o decréscimo proporcional, o *feedback* negativo, no qual temos a associação da taxa de mortes ao tamanho da população. Neste caso, o aumento na taxa provoca uma redução no acumulador, que provocará uma redução na taxa em um momento posterior.

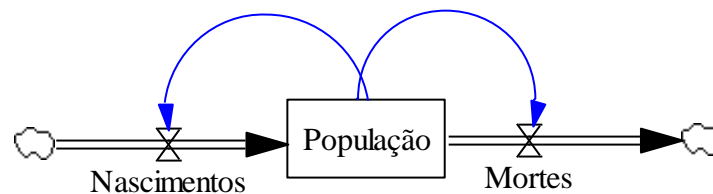


Figura 3 - Modelo do comportamento da população em função das taxas de nascimento e morte.

Para simular o comportamento de um sistema tal como o anterior foram assumidos os valores de população inicial de 100 pessoas, a taxa de nascimentos de 5% da população por mês, a taxa de mortes de 1% da população por mês, tempo inicial zero, tempo final de 100 meses e passo de 1 mês. A população resultante em cada passo de simulação é obtida pela soma do número de nascimentos ao valor atual da população e a subtração do número de mortes ao valor atual da população. Como pode ser observado na Figura 4, o comportamento da população varia quando se considera somente a taxa de nascimentos, somente a taxa de mortes, ambas as taxas ou nenhuma das taxas. Desta forma, pode-se perceber a variedade de comportamentos não lineares que podem ser assumidos mesmo em um modelo simples como o apresentado acima.

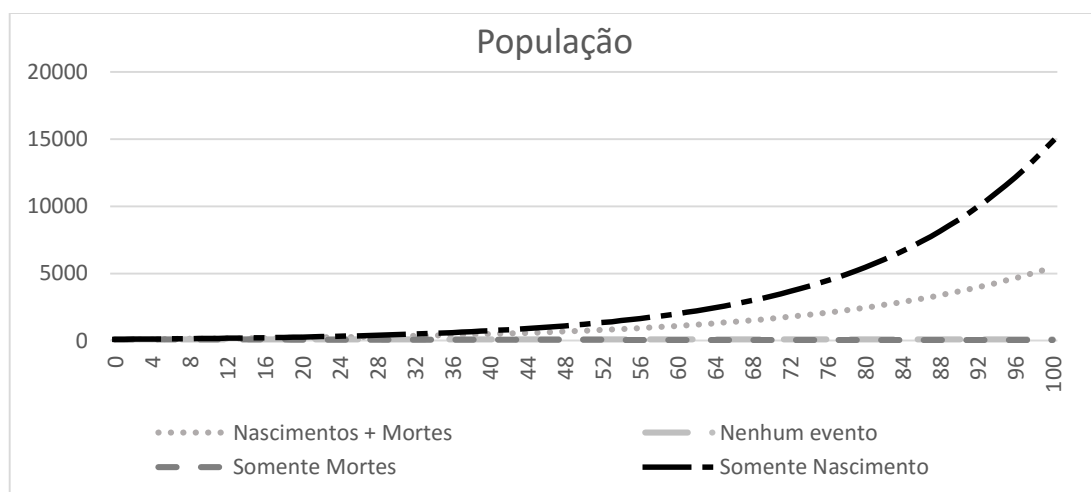


Figura 4 - Gráfico comparativo dos comportamentos da população pelo tempo com as combinações dos parâmetros do modelo não linear

Focando em um projeto de desenvolvimento de software, temos na Figura 5 um modelo de comportamento que envolve requisitos para serem desenvolvidos pela equipe durante uma carga horária definida. Simplificando o modelo, assume-se que todos os integrantes da equipe possuem uma mesma produtividade média, definida em requisitos por hora. O modelo é iniciado com os seguintes parâmetros: uma determinada quantidade de requisitos a ser implementada, uma quantidade de integrantes da equipe, uma quantidade de horas como carga horária diária, uma taxa de produtividade média e uma taxa de geração de erros média. Durante a execução do modelo, a equipe implementa os requisitos conforme a carga horária e a taxa de produtividade média. Devido à taxa de geração de erros, uma quantidade de requisitos é implementada com erros. Estes requisitos são retornados para o montante de requisitos a ser implementado, de modo que seus erros sejam corrigidos. O modelo se encerra quando a quantidade de requisitos a ser implementado estiver zerada ou o tempo estipulado para a simulação do modelo se encerrar.

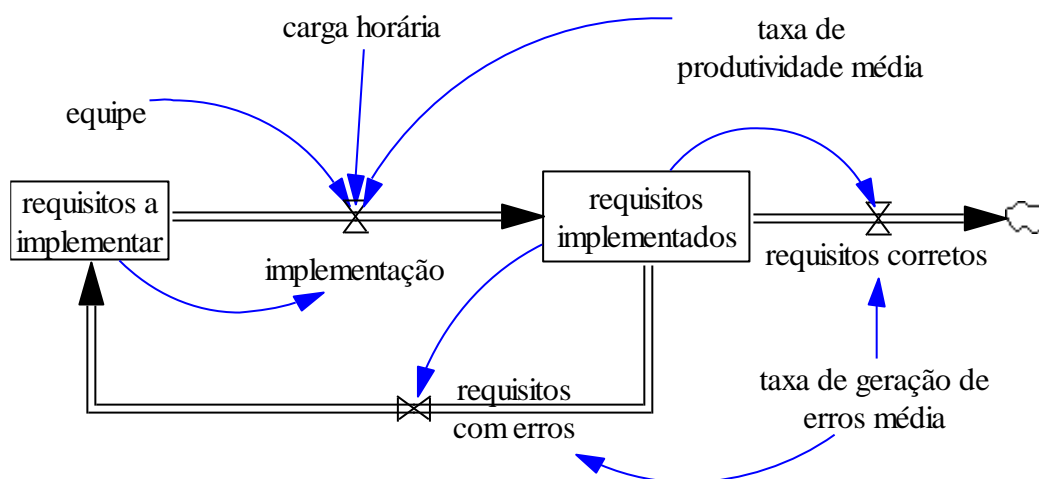


Figura 5 - Modelo simplificado de um projeto de desenvolvimento de software

O modelo apresentado na Figura 5, por ser um modelo simplificado, não considera todos os aspectos que podem afetar um projeto de desenvolvimento de software. Dentre estes aspectos, podemos destacar a fadiga, aumento de carga de trabalho, o tempo de aprendizado da equipe, o tempo gasto com comunicação na equipe e o efeito na qualidade pelo trabalho em horas extras. No entanto, o modelo serve como estrutura geral para o modelo que desenvolveremos, a fim de integrar alguns destes aspectos. O modelo desenvolvido no contexto deste trabalho será apresentado no Capítulo 3.

2.2.3. Efeitos do Trabalho em Horas extras sobre o Projeto

Os projetos se comportam de maneira variada conforme as atividades a serem desempenhadas, os recursos humanos que estarão envolvidos, as decisões tomadas pelos gestores e a jornada de trabalho imposto.

No tocante à jornada de trabalho, no Brasil temos a Consolidação das Leis Trabalhistas (CLT) que é a principal norma legislativa referente ao Direito do Trabalho. Nesta norma estão as definições das regras consideradas normais para as cargas horárias diárias dos diversos tipos de profissionais. Levando em consideração um profissional de qualquer atividade em uma empresa privada, a carga horária deve ser de 8 horas diárias⁵ (BRASIL, 1943). Entretanto, como mencionado na Seção 2.1, os profissionais da área de TI, frequentemente são requisitados a realizar trabalhos em uma carga horária superior à carga horária definida como normal. Além das limitações e definições das quantidades de horas extras permitidas e da formalística da solicitação até a realização, existe também regulamentações referentes aos custos associados a estas horas, que chegam a 200% do valor da hora normal, naturalmente considerando também adicional noturno, caso o trabalho seja realizado após as 22:00.

Jia, Fan e Lu (2007), focando na estratégia de alocação de horas extras, construíram um modelo que mapeia seus efeitos positivos, aumentando a produtividade e diminuindo a quantidade de trabalho restante a ser realizado, assim como os efeitos negativos, aumentando a fadiga, a geração de erros e o retrabalho pela geração de produtos defeituosos. Os autores utilizaram um estudo de caso para validar seu modelo. Neste estudo, o modelo foi calibrado com informações do projeto ISAM3.1 da empresa Alcatel Shanghai Bell e foram executadas oito rodadas de simulação. O projeto tinha um prazo real de 10 meses, sendo que para verificar os efeitos de compressão e folga dos prazos foram executadas rodadas de simulação com prazos que variaram de 7 a 12 meses. Os resultados obtidos evidenciaram que a alocação de horas extras nem sempre é uma solução efetiva para a redução da pressão do cronograma. Quando a equipe já alcançou o limite de realização de horas extras, a persistência na alocação de horas extras incorrerá em mais atrasos no cronograma. A definição da melhor política de horas extras inicia com

⁵ As horas diárias consideradas normais podem variar de acordo com a função exercida pelo profissional, podendo ser uma carga horária parcial de no máximo 25 horas semanais ou até mesmo um profissional que trabalhe em regime de plantão com duração de 12 horas diárias e com período de descanso após este período de trabalho. Na medida em que uma função possui características diferenciadas, poderá ser fixada uma carga horária diferente da normal, conforme previsto no artigo 58 da CLT.

a definição do cronograma com base nas informações do projeto e com a busca pelo menor tempo para finalização do projeto por meio da modelagem da dinâmica de sistemas.

Meilong, Li e Chen (2008) construíram modelos para o comportamento de projetos de desenvolvimento de software e os efeitos da inclusão de novos desenvolvedores. Nestes modelos, os novos desenvolvedores consumiam tempo para treinamento e alinhamento com a equipe experiente. Os autores também retrataram os efeitos das práticas de horas extras, aumentando a produtividade, porém afetando a qualidade do que é produzido e resultando em retrabalho. Este estudo mostrou a existência de efeitos positivos e negativos relacionados com a alocação de horas extras aos desenvolvedores.

Os efeitos colaterais citados não são particularidades de projetos de software, uma vez que Zhao, Lv e You (2007) utilizaram os mesmos conceitos para a modelagem de projetos de construção civil. Os autores construíram um grande modelo para o mapeamento do comportamento de um projeto que contém toda a parte comum de execução de atividades, assim como a disponibilidade de recursos humanos e alteração de qualidade do trabalho pela fadiga pela horas extras. Para a validação e teste do modelo, foram utilizadas informações de um caso real. Após analisar os resultados do modelo, percebemos que assim como em projetos de software, à medida que os profissionais aumentam o número de horas diárias de trabalho, a qualidade do que produzem é diminuída, gerando assim retrabalho que, por sua vez, gera impacto no custo total e no prazo do projeto.

Conforme citado na subseção 2.2.1, a Modelagem Dinâmica de Projetos de Software proporciona o mapeamento e o entendimento do comportamento de projetos. Nesta modelagem, é possível incluir os efeitos observados quando os profissionais de desenvolvimento de software são submetidos a uma carga horária superior à normal. Ao pesquisar autores que discutem esta temática, verificou-se que Meilong, Li e Chen (2008) e Jia, Fan e Lu (2007) modelaram o efeito das horas extras sobre o prazo do projeto. Neste sentido, quando são adicionadas horas extras, a produtividade aumenta pelo acréscimo de horas no período de trabalho realizado e o prazo, em dias, diminui em função da concentração de esforço. Entretanto, com o excesso de trabalho, o profissional tende a ficar cansado e acaba por produzir um trabalho com qualidade inferior ao produzido durante a carga horária normal. Sendo assim, os autores modelaram efeitos positivos e os efeitos negativos de aumentar a carga horária diária de trabalho, faltando considerar a

heterogeneidade entre as atividades de projeto, uma vez que cada tipo de atividades de um projeto de desenvolvimento de software possui características próprias e um comportamento diferenciado no modelo.

Visando avaliar a qualidade das inúmeras possibilidades de alocações de horas extras para todas as atividades de um projeto de desenvolvimento de software, necessitaremos de recursos para a busca pela melhor estratégia para a definição das horas extras para o projeto. Para tanto, lançamos mão de Engenharia de Software Baseada em Buscas, que será o assunto discutido na próxima seção.

2.3. A Otimização da Gerência de Tempo em Projetos

Capturando os aspectos do comportamento de um projeto de desenvolvimento de software, não facilmente evidenciados sem as técnicas de Dinâmica de Sistemas e simulação, encontramos uma grande quantidade de possibilidades para o planejamento de horas extras das suas atividades.

Conforme Artigues, Demasse e Neronw (2008), os problemas que envolvem balanceamento de objetivos (*prazo e custo*) em gerenciamento de projetos são considerados problemas NP completos que possuem um grande conjunto de possíveis soluções em seu espaço de busca. Considerando a definição de Harman e Clark (2004), a enumeração destas possíveis soluções é chamada de espaço de busca e para este tipo de problema o espaço de busca é enorme, tornando-se impossível a enumeração de todas as soluções. Desta forma, a Engenharia de Software convencional não é suficiente para encontrar uma solução ótima com relação aos objetivos selecionados.

Para buscar pela melhor estratégia de alocação de horas extras para as atividades de um projeto de desenvolvimento de software, modelamos o problema como um problema de otimização e utilizamos técnicas de otimização heurística, conforme Harman, Mansouri e Zhang (2012) modelam este tipo de questão de Engenharia de Software.

2.3.1. A Engenharia de Software baseada em Buscas

A Engenharia de Software Baseada em Buscas, tradução da nomenclatura em inglês “*Search Based Software Engineering*” (SBSE), foi considerada uma área de pesquisa emergente em 2001. A SBSE é uma área que reformulou os clássicos problemas de Engenharia de Software como problemas de busca, utilizando técnicas baseadas em

buscas que fornecem soluções para problemas difíceis de balanceamento de restrições conflitantes. Desta forma, é possível encontrar soluções aceitáveis em situações onde soluções ótimas são impossíveis ou praticamente inviáveis (HARMAN; JONES, 2001).

Ferrucci, Harman e Sarro (2014) comparam a SBSE com as outras engenharias, pois todas têm como principal objetivo otimizar e melhorar alguma coisa, sintetizando com a seguinte frase: “...buscamos construir sistemas que são melhores, rápidos, baratos, confiáveis, flexíveis, escaláveis, responsivos, adaptáveis, manuteníveis e testáveis”⁶ (FERRUCCI; HARMAN; SARRO, 2014, p 2).

Na SBSE é necessário definir como o problema será representado e a forma com que a solução para este problema será avaliada. A avaliação da solução é realizada segundo informações calculadas a partir de características do problema e da solução ou extraídas diretamente destas características. Estas informações são chamadas objetivos. Quando temos somente uma informação a ser avaliada chamamos de otimização mono-objetivo e quando possuímos mais de uma informação a ser avaliada chamamos de otimização multi-objetivo. A função capaz de mensurar a qualidade de cada uma das soluções encontradas, por meio de um conjunto de medidas, é chamada de função de *fitness* (HARMAN; CLARK, 2004).

Em um contexto multi-objetivo, a avaliação da qualidade das possíveis soluções necessita encontrar um balanceamento adequado entre os objetivos conflitantes para um projeto, ou seja, a busca pela menor duração, o menor custo e a maior qualidade. Este balanceamento representa, essencialmente, uma questão de otimização dos objetivos do projeto, uma vez que compreende uma grande quantidade de possibilidades de combinações, podendo ser bastante difícil encontrar boas soluções (FERRUCCI; HARMAN; SARRO, 2014).

Devido a obtenção de bons resultados com a utilização de SBSE, além do ambiente acadêmico, Harman; Mansouri e Zhang (2012) ressaltaram que algumas empresas de software também vêm apresentando interesse em SBSE, como a Daimler ((BÜHLER; WEGENER, 2008); (WEGENER; BARESEL; STHAMER, 2001); (WINDISCH; WAPPLER; WEGENER, 2007)), Ericsson (ZHANG *et al.*, 2010), IBM ((YOO; HARMAN; UR, 2009), (YOO; HARMAN; UR, 2011)), Microsoft ((LAKHOTIA *et al.*, 2010); (XIE *et al.*, 2008)), Motorola (BAKER *et al.*, 2006), Nokia (DEL ROSSO, 2006) e NASA (FEATHER; KIPER; KALAFAT, 2004).

⁶ Tradução livre

2.3.2. Aplicação da SBSE no Gerenciamento de Projetos de Software

O Gerenciamento de Projetos de Software possui diversas atividades críticas para o sucesso do projeto, que envolvem um balanceamento entre objetivos muitas vezes conflitantes. Este balanceamento é um problema de otimização que pode ser resolvido com a aplicação de SBSE.

Montando uma estrutura de SBSE para gerenciamento de projetos de desenvolvimento de software, temos na Figura 6 uma estrutura genérica que é composta pelas informações de pacotes de trabalho ou atividades com suas respectivas informações de duração, dependências e custos, assim como as capacidades e habilidades da equipe prevista para o projeto. Com estas informações de entrada, utilizamos os algoritmos de otimização para buscar pela melhor solução em relação aos objetivos desejados. O andamento do projeto é guiado pelo simulador que proporciona a execução ou desenvolvimento dos pacotes de trabalho pelas equipes, conforme as informações dos envolvidos (pacote de trabalho, equipe e simulador). Como o simulador é responsável por guiar o andamento do projeto, a modelagem do problema nem sempre pode ser considerada tão realista em relação ao projeto de desenvolvimento software real, sendo considerado uma limitação nesta abordagem (FERRUCCI *et al.*, 2014; HARMAN *et al.*, 2012).

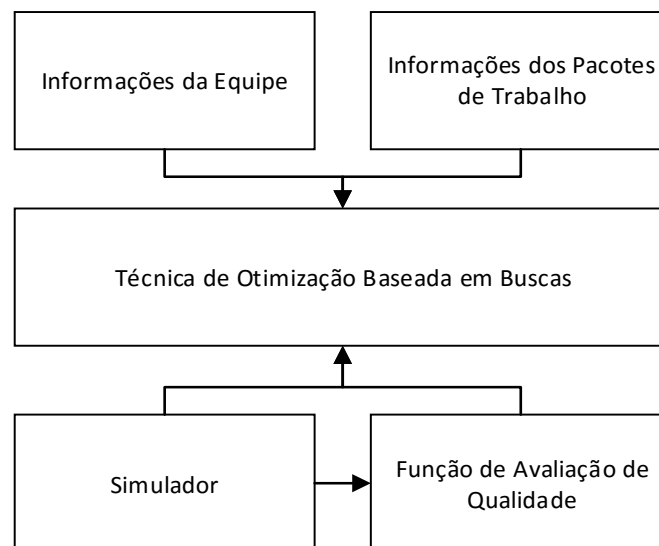


Figura 6 - Diagrama de Blocos de uma estrutura genérica de SBSE (baseada na Figura 2 de Harman; Mansouri e Zhang (2012))

Visando a redução da duração de um projeto, Antoniol, Di Penta e Harman (2005) utilizaram *Simulated Annealing*, Algoritmos Genéticos⁷ e *Hill Climbing* na otimização da alocação de pacotes de trabalho aos recursos em um projeto de manutenção de software. O algoritmo *Simulated Annealing* (Arrefecimento Simulado) é uma metáfora de um processo térmico, utilizado em metalurgia para obtenção de estados de baixa energia num sólido. O processo consiste de duas etapas: na primeira, a temperatura do sólido é aumentada e, na segunda, o resfriamento deve ser realizado lentamente até que o material se solidifique, sendo acompanhado e controlado esse arrefecimento. Na prática, o algoritmo substitui a solução atual por uma solução próxima, de sua vizinhança no espaço de busca, escolhida de acordo com uma função objetivo e com uma variável T (dita *Temperatura*, por analogia). Quanto maior for T, maior a componente aleatória que será incluída na próxima solução escolhida. A medida que o algoritmo progride, o valor de T é decrementado, começando o algoritmo a convergir para uma solução ótima, necessariamente local. O algoritmo *Hill Climbing* (Subida da montanha) é uma metáfora para uma escalada, na qual o topo é o melhor lugar onde se quer chegar, mas à medida que se chega no topo de uma montanha, outras montanhas maiores são encontradas e a escalada continua. Na prática, o algoritmo inicia de um ponto aleatório no espaço de busca, e esta solução tem partes modificadas com base em uma função aleatória. Este processo é repetido variando a qualidade das soluções até que não encontre mais soluções melhores. No projeto modelado pelos autores, era possível acrescentar mais recursos para ajudar a reduzir o tempo do projeto, contrariando diretamente a lei de Brooks⁸ (Brooks 1975). Esta prática somente se mostrou vantajosa para grandes equipes, compostas por exemplo, por 40 pessoas.

Di Penta, Harman e Antoniol (2011) utilizaram novamente *Simulated Annealing* (SA), Algoritmos Genéticos (GA) e *Hill Climbing* (HC) para avaliar o agendamento e contratação de pessoal no gerenciamento de projetos de software considerando, desta vez, instâncias do mundo real e uma otimização mono e multi-objetivo com objetivos conflitantes (duração, nível de capacitação dos recursos e quantidade de alocação dos recursos). Os experimentos executados evidenciaram que a contratação de pessoal para

⁷ Algoritmos Genéticos serão explicados em detalhes na seção 3.2.4.

⁸ O acréscimo de mais recursos a um projeto pode gerar um atraso maior ainda pelo gasto adicional com comunicação entre a equipe e o novo integrante. Em resumo, não devemos assumir que uma atividade que necessitaria de um tempo de X meses para uma equipe Y, reduziria para X/2 meses para uma equipe 2*Y.

equipes a partir de certos tamanhos não representa redução nos prazos de projetos que envolvam grande número de dependências entre as atividades. Os autores recomendam evitar ao máximo a fragmentação das atividades dos profissionais dos projetos, naturalmente existem situações de projetos simultâneos e treinamento de aperfeiçoamento. Por fim, os autores reforçam que algoritmos fornecem soluções para os gerentes, mas não conseguem substituir as atividades e a capacidade de tomada de decisão do gerente do projeto.

Chicano *et al.* (2011) utilizaram vários Algoritmos Genéticos para uma otimização multi-objetivo (custo e duração) para o problema de alocação de atividades aos recursos de um projeto de desenvolvimento de software, além de envolver a especialização e a quantidade de desenvolvedores alocados. O experimento mostrou correlação inversa entre o custo e a duração do projeto, conforme esperado. Em relação ao custo e a quantidade de desenvolvedores alocados, observou-se correlação positiva somente para um grupo de desenvolvedores, entretanto, isto não significa que o custo diminua para o outro grupo, isto ocorreu devido aos desenvolvedores com salário inferiores.

Ponnam e Geethanjali (2014) aplicaram o algoritmo de Colônia de Formigas para a otimização de cronograma de projetos. Este algoritmo é baseado em uma analogia com a forma com que as formigas escolhem o menor caminho entre o formigueiro e a comida. Ele inicia em um ponto aleatório para a realização do percurso. Após a conclusão de um percurso, este é avaliado e conforme a qualidade desta solução um reforço positivo (feromônio) é aplicado, motivando novas jornadas a passarem por trechos deste caminho. Com as execuções do percurso, a solução final acumula os melhores trechos, com o melhor caminho. Os experimentos mostraram efetividade para atividades com pequena duração. Entretanto, como não consideraram o controle de alocação de recursos, os resultados não foram totalmente satisfatórios para o problema do planejamento de atividades para um projeto, uma vez que a alocação da mesma tarefa para diferentes recursos representa um problema enfrentado.

Durante a análise das pesquisas de diversos autores na área de gerenciamento de projetos baseado em buscas, como por exemplo, Aguilar-Ruiz *et al.* (2001), Chicano *et al.* (2011), Chang, Christensen e Zhang (2001) e Minku e Yao (2013), verificamos que os experimentos são frequentemente realizados com dados gerados por geradores de instâncias, ao invés de dados reais de projetos de desenvolvimento de software. Nas pesquisas, a utilização de instâncias reais foi constatada para poucos autores, como por

exemplo Rahman *et al.* (2010), Di Penta, Harman e Antoniol (2011) e Kang, Jung e Bae (2011). Em relação aos algoritmos, percebemos que a maior parte dos estudos utilizam Busca Local, *Simulated Annealing*, Algoritmo Genético, Programação Genética e o *Hill Climbing* ou variações destes.

Buscando por pesquisas focadas na gestão de tempo para projetos de desenvolvimento de software, encontramos o trabalho de Ferrucci *et al.* (2013) que buscou otimizar a alocação de horas extras com um balanceamento dos riscos do projeto. Para isto, foi utilizado o algoritmo genético NSGA-II com um operador de cruzamento modificado que obteve resultados melhores durante os experimentos utilizados para avaliar a proposta. Estes resultados, que são compostos pelas alocações de horas extras ao longo das atividades do projeto foram comparados com as decisões técnicas praticadas atualmente pelos gerentes de projetos na indústria. Os autores classificaram estas práticas (*Margarine*, *Second Half* e *Critical-path Method*) como estratégias de gerenciamento de horas extras, que serão apresentadas no Capítulo 3.

Mahadik (2014) buscou otimizar o planejamento e o cronograma de projetos de desenvolvimento de software, ressaltando um grande diferencial entre os projetos de software e os demais tipos de projetos. Para os projetos de software existe uma dependência elevada nas pessoas que desempenham as atividades, ou seja, os recursos utilizados no projeto, haja visto que os recursos utilizados neste tipo de projeto são basicamente compostos de pessoas. Esta característica evidenciou que, para modelar e otimizar o planejamento e o cronograma de um projeto de software, será necessário considerar a alocação das pessoas, ou seja, a carga de trabalho além do cronograma das tarefas. Para esta otimização foi utilizado o algoritmo de Colônia de Formigas, que por sua vez, obteve resultados com diminuição de custos e carga de trabalho.

Apesar dos trabalhos acima considerarem os efeitos positivos da realização de horas extras e a preocupação com o profissional, conforme Abdel-Hamid e Madnick (1991), existem na dinâmica de um projeto de desenvolvimento de software os efeitos nocivos de perda de qualidade do trabalho realizado devido ao aumento da carga de trabalho. Outra particularidade também não considerada nos trabalhos pesquisados, foi o custo excedente para as horas extras trabalhadas. No Brasil, existem legislações que afetam os custos dos projetos que contemplam a realização de horas extras, planejadas ou não. Esta característica, evidencia que para os trabalhos de otimização de cronogramas e planejamentos de projetos de software, o objetivo de minimização de custo deve considerar os acréscimos causados pelas horas extras.

2.4. Considerações Finais

Por meio da busca bibliográfica relacionada com a prática de horas extras, percebemos a grande frequência com que a realização de horas extras é solicitada aos profissionais alocados em projetos de desenvolvimento de software. Devido a esta frequência elevada e não planejada, os projetos acabam tendo seus prazos e orçamentos afetados.

Para que os planejamentos possam ser mais facilmente realizados, temos as técnicas de modelagem dinâmica e simulação que permitem o mapeamento do comportamento do projeto ao longo do tempo. A medida que os parâmetros de duração da carga horária de trabalho são alterados, os efeitos destas alterações podem ser verificados por meio de simulação.

Em complemento ao planejamento, verificamos a necessidade de utilização de otimização, uma vez que a quantidade de combinações de cargas horárias diárias de trabalho para cada uma das atividades de um projeto de desenvolvimento de software são inúmeras, impossibilitando a enumeração simples. Desta forma, o processamento viabilizará a busca pela melhor solução para o problema de verificação da necessidade de alocação de horas extras às atividades de um projeto de desenvolvimento de software.

No próximo capítulo será apresentada a formalização do problema de planejamento de horas extras de cronogramas de projetos de desenvolvimento de software, batizado de Problema do Planejamento de Horas extras (PPH), assim como a proposta de solução.

Capítulo 3 - Simulação do Modelagem de Dinâmica com Otimização Heurística

Os estudos apresentados no Capítulo 2 tratam dos efeitos positivos e negativos da alocação de horas extras no planejamento de cronograma para projetos de desenvolvimento de software. Entretanto, os estudos apresentados consideravam todas as atividades do projeto de forma homogênea, não levando em consideração as peculiaridades de cada uma delas e a propagação dos erros gerados nas atividades das fases iniciais para as atividades seguintes do processo de desenvolvimento.

Este capítulo apresenta uma proposta de solução para o problema de planejamento de horas extras de cronogramas de projetos de desenvolvimento de software, batizado de Problema do Planejamento de Horas extras (PPH). Esta solução se dará por meio da simulação do comportamento de um projeto utilizando técnicas de otimização para buscar por um balanceamento adequado entre os efeitos positivos e negativos decorrentes da realização de trabalho em horas extras. Os efeitos positivos são representados pelo aumento de produtividade dos desenvolvedores em função do maior número de horas trabalhadas em um mesmo período. Os efeitos negativos são representados por um aumento na taxa de geração de erros em função do cansaço. A abordagem proposta tem como objetivo identificar a melhor quantidade de horas extras para cada uma das atividades do projeto, de modo a balancear os dois tipos de efeito e suas consequências indiretas no projeto.

Este capítulo está organizado em três seções, iniciando com a formalização do problema, seguindo com a proposta de solução e encerrando com as considerações finais do capítulo.

3.1. Formalização do Problema

A formalização do Problema do Planejamento de Horas extras (PPH) recebe como entrada um projeto de desenvolvimento de software cuja solução produz como saída um número de horas extras a ser consumido por cada atividade do projeto.

Esta seção está dividida em cinco subseções que iniciam com a dinâmica de um projeto de desenvolvimento de software, apresentam a dinâmica do trabalho em horas extras, a necessidade de simulação, a necessidade de otimização para busca da solução, e, por fim, os objetivos da otimização heurística.

3.1.1. A Dinâmica de um Projeto de Desenvolvimento de Software

Um projeto é representado por um grafo direcionado acíclico $\langle GF, GFDEP \rangle$, cujo conjunto de nós é formado por uma lista $GF = \{gf_1, gf_2, \dots, gf_n\}$ de grupos de funcionalidades e cujo conjunto de arestas é formado por uma lista de pares ordenados $GFDEP = \{(gf_i, gf_j): i \neq j, 1 \leq i \leq n, 1 \leq j \leq n\}$ de dependências entre os grupos de funcionalidades. Neste contexto, um grupo de funcionalidades é um grupo de requisitos fortemente relacionados e que serão implementados em conjunto. Cada $gf_i \in GF$ é descrito por uma quantidade de pontos de função, que representa a complexidade dos requisitos reunidos neste grupo e que será utilizada como base para calcular o esforço necessário para o seu desenvolvimento. Cada entrada na lista $GFDEP$ representa um grupo de funcionalidades (gf_j) cujo desenvolvimento somente pode ser iniciado após a conclusão da atividade de levantamento de requisitos de um segundo grupo de funcionalidades (gf_i). A Figura 7 apresenta um exemplo de dependência entre grupos de funcionalidades, considerando que $(GF_1, GF_2, GF_3, GF_4, GF_5) \in GF$ e as setas apontam para os grupos de funcionalidades que apresentam dependência de outros.

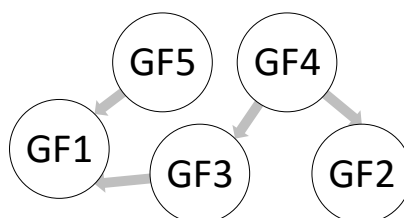


Figura 7 - Um exemplo de dependências entre grupos de funcionalidades

O cronograma do projeto é gerado a partir do fluxo de trabalho necessário para transformar os requisitos reunidos nos grupos de funcionalidades em um software executável e correto. Neste sentido, o cronograma de cada grupo de funcionalidades é composto por quatro tipos de atividades: especificação, projeto, codificação e testes. Estas

atividades são dispostas na ordem de precedência ilustrada pela Figura 8. Uma atividade de testes é sempre precedida por uma atividade de codificação, que é sempre precedida por uma atividade de projeto. A atividade de especificação é a primeira da sequência de atividades, sendo necessária a sua conclusão para o prosseguimento para a atividade de projeto. Estas dependências entre atividades representam o fluxo de trabalho necessário para a modelagem do problema, o projeto da solução, a implementação e teste do seu código-fonte. Naturalmente, o responsável por um projeto de desenvolvimento de software pode decidir organizar suas atividades de outra forma, mas para abordar o PPH consideraremos esta estrutura de cronograma.

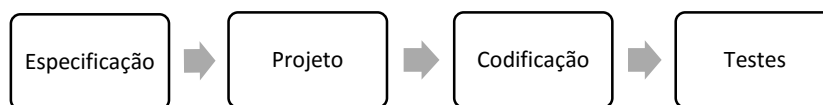


Figura 8 - Ordem de precedência dos tipos de atividades

O cronograma do projeto também prevê as dependências entre os grupos de funcionalidades, expressas no conjunto de arestas GFDEP. Para toda entrada do conjunto GFDEP, o cronograma considera uma dependência entre as atividades de especificação dos dois grupos de funcionalidades envolvidos na entrada, conforme apresentado na Figura 9, considerando que $(GF_1, GF_2) \in GFDEP$.

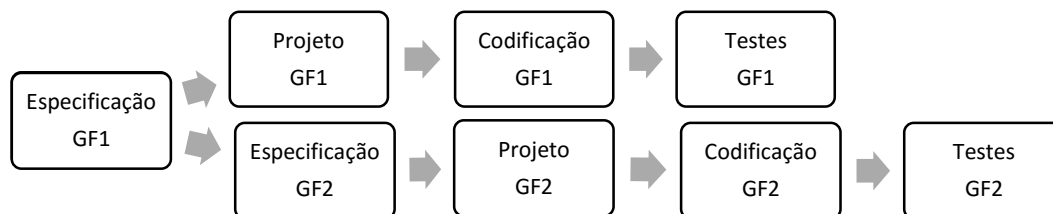


Figura 9 - Dependência entre Grupos de Funcionalidades (GF1 e GF2)

Com base no processo de construção descrito acima, o cronograma de um projeto é representado por um grafo direcionado acíclico $\langle AT, ATDEP \rangle$, cujo conjunto de nós é formado por uma lista $AT = \{at_1, at_2, \dots, at_m\}$ de atividades e cujo conjunto de arestas é formado por uma lista de pares ordenados $ATDEP = \{(at_i, at_j): i \neq j, 1 \leq i \leq m, 1 \leq j \leq m\}$ de dependências término-início⁹ entre as atividades. Cada atividade é classificada como especificação, projeto, codificação ou testes. Cada atividade de especificação, projeto e codificação é descrita por uma quantidade de trabalho a ser executado (medida em horas

⁹ Esta relação de dependência indica que para que uma atividade se inicie, a atividade anterior deve obrigatoriamente ter terminado.

de desenvolvedor) e por uma quantidade de erros que se espera gerar durante a execução desta atividade, de acordo com as taxas de introdução de erros por atividade (Tabela 1) apresentadas por Jones (2007, pg. 114). Esta tabela foi modificada para considerar o efeito de regeneração de erros apresentado por Abdel-Hamid e Madnick (1991, pg.113). Ela indica que um desenvolvedor típico, trabalhando em um ambiente típico, introduzirá, em média, um erro por ponto de função em cada atividade executada. A regeneração de erros captura a dinâmica através da qual um erro introduzido em uma fase inicial do desenvolvimento de software, além de se perpetuar caso não seja corrigido, provoca erros de interpretação nas fases seguintes, fazendo com que novos erros sejam introduzidos durante o desenvolvimento das fases seguintes.

Conforme a ordem de precedência apresentada na Figura 8, os projetos se iniciam na fase de especificação, não ocorrendo regeneração de erros nesta fase devido à inexistência de fase anterior. Entretanto, durante esta fase é introduzido 1 erro por ponto de função. Durante a fase de projeto são introduzidos 1,25 erros por pontos de função, devido a inclusão de 1 novo erro por ponto de função durante esta fase, acrescido de 25% decorrente da regeneração dos erros gerados até a conclusão da fase de especificação. Durante a fase de codificação são introduzidos 1,75 erros por pontos de função, devido a inclusão de 1 novo erro por ponto de função desta fase, acrescidos de 33% dos erros acumulados na fase de projeto. Somando as quantidades de erros gerados e regenerados, temos um total de 4 erros por ponto de função, que correspondem à quantidade média de erros que atingem a fase de testes.

Tabela 1 - Taxas de erros introduzidos por ponto de função e percentual de erros regenerados por fase do projeto

Fase	Quantidade de erros introduzidos (Erros/pontos de função)	Quantidade de erros regenerados (% de erros da fase anterior)
Especificação	1	N/A
Projeto	1	25%
Codificação	1	33%

Além da quantidade de erros que se espera gerar, as atividades de especificação, projeto e codificação também são descritas pelo esforço médio necessário para produzir seus resultados. As atividades de teste são descritas pelo esforço médio necessário para identificar e corrigir cada erro gerado nas atividades anteriores. Para calcular o esforço necessário para as atividades de especificação, projeto e codificação consideramos o percentual de esforço dedicado a cada uma destas atividades em um projeto de

desenvolvimento de software, conforme apresentado por (Jones, 2007, pg 97) e sintetizado na Tabela 2. Neste sentido, as atividades de especificação contemplaram as atividades de *Requirements Definition*, indicadas por Jones (2007), variando entre 5% e 9% do esforço total do projeto, de acordo com o tamanho do projeto em pontos por função. As atividades de projeto contemplaram as atividades de *Design*, indicadas por Jones (2007), variando de 5% a 13% do esforço total do projeto. Finalmente, as atividades de codificação contemplaram as atividades de *Coding*, indicadas por Jones (2007), ocupando entre 15% e 50% do esforço total do projeto.

Tabela 2 - Taxas de alocação de esforço pelas fases do projeto de desenvolvimento de software (Jones, 2007, pg 97)

Tamanho do projeto (PF)	Percentual do esforço total do projeto para cada fase			
	Especificação	Projeto	Codificação	Testes
10	5%	5%	50%	27%
100	5%	6%	40%	30%
1000	7%	10%	30%	30%
10k	8%	12%	20%	33%
100k	9%	13%	15%	34%

Para o cálculo do esforço total do projeto, somamos o número de pontos de função atribuídos a cada grupo de funcionalidades e utilizamos uma taxa de produtividade média de 27,8 pontos de função desenvolvidos e testados por um profissional por mês, conforme apresentado em Jones (2000). Desta forma, encontramos o esforço em homens x mês necessário para realizar o desenvolvimento e teste de todos os grupos de funcionalidades. Após termos o esforço total, calculamos o esforço de cada atividade de especificação, projeto, codificação e testes, com base na Tabela 2.

Para o cálculo do esforço para a atividade de testes, consideramos a produtividade média (27,8 PF/mês), o esforço esperado para a fase de testes e a quantidade média de erros que atingem a fase de testes, calculada com base na Tabela 1, na qual mostramos que as taxas de geração e regeneração de erros influenciam diretamente neste valor de esforço. Dividindo o esforço dedicado à atividade de testes pela quantidade média de erros que atingem a fase de testes, teremos o esforço médio necessário para a correção de cada erro gerado.

Para o cálculo da duração do projeto, utilizamos o esforço calculado (homem x mês) e a quantidade de profissionais alocados para as atividades do projeto. Considerando

uma jornada de trabalho normal (8 horas diárias), fazemos uma divisão simples do esforço pela quantidade de profissionais alocados para obtermos a duração.

Com a definição do esforço necessário para execução de cada uma das atividades do projeto e a quantidade de profissionais alocados por atividade, podemos realizar os cálculos dos custos pela multiplicação da quantidade de horas de trabalho pelo custo da hora do profissional que está alocado na atividade.

Nesta subseção definimos as formas de cálculo do esforço, da duração e do custo para um projeto. Entretanto, conforme apresentado no Capítulo 2, os profissionais de TI são frequentemente submetidos a trabalhos em horas extras. Os impactos que os trabalhos em horas extras geram para os projetos de desenvolvimento de software serão discutidos na próxima seção.

3.1.2. A Dinâmica do Trabalho em Horas extras

O gerente do projeto de desenvolvimento de software pode definir a melhor política de horas extras para o projeto sob sua gestão, podendo defini-la de forma homogênea para todo o projeto, modificando por fase do projeto ou até mesmo por um período de tempo no qual o projeto está passando por uma necessidade de urgência. Por este motivo, para o PPH, consideramos a possibilidade de alocação de carga horária diária diferenciada para cada atividade, permitindo maior flexibilidade de gestão do projeto.

A representação matemática da alocação de horas extras para as atividades se dará por uma sequência de m números inteiros n , onde m representa a quantidade de atividades previstas para um projeto e cada n representa a carga horária empregada em cada uma das m atividades, sendo escrito conforme o conjunto $\{n \in \mathbb{Z} / 1 \leq n \leq 9\}$. Desta forma, temos nove possibilidades que traduzem a variação, com escala de 30 minutos, das 8 até 12 horas diárias de trabalho. Esta escala foi utilizada devido à necessidade de encontrar um balanceamento entre o tempo de processamento que será necessário para encontrar boas soluções para o PPH e a precisão. Como o parágrafo 1º do artigo 58 da CLT (BRASIL, 1943) informa que a realização de somente 10 minutos adicionais à carga horária diária não é caracterizado como hora extra e como estamos fazendo um planejamento de alocação de horas extras, arredondamos para variações de 30 minutos. Por exemplo, uma atividade A associada a um valor 6, terá uma carga horária diária de 10,5 horas de trabalho, ou seja, as 8 horas normais somadas às 2,5 horas extras.

Conforme apresentado na subseção anterior, o cálculo do custo para execução de uma atividade utiliza o custo da hora de trabalho do profissional responsável pela

atividade. Entretanto, o custo da hora do profissional varia quando o mesmo está trabalhando além das horas consideradas normais. Conforme a legislação brasileira (BRASIL, 1943), temos as horas classificadas como normais no intervalo de 0 até 8 horas diárias. Estas horas normais possuem um custo base, calculado a partir do salário do desenvolvedor. Para os casos em que a carga horária de alguma atividade exceder as 8 horas diárias, teremos um acréscimo de 20% para as duas primeiras horas extras e de 50% para as próximas duas horas extras, limitando em 12 horas de trabalho por dia.

Em contrapartida ao aumento de custos, a alocação de horas extras para as atividades do projeto proporciona mais horas de trabalho e, conseqüentemente, mais tempo de produção. Entretanto, conforme a lei de DeMarco e Lister (1999), um desenvolvedor que trabalha mais do que a carga horária normal pode produzir software com baixa qualidade, ou seja, com maior quantidade de erros. Quanto mais horas um desenvolvedor trabalha, mais cansado ele fica, se tornando menos cuidadoso com o trabalho que realiza e, conseqüentemente, tende a introduzir mais erros no software.

Abdel-Hamid e Madnick (1991) consideram que um desenvolvedor pode introduzir até 50 % mais erros para o caso de aumentar em 50 % a carga horária diária, ou seja, 12 horas diárias, conforme pode ser observado no gráfico da Figura 10. Por este motivo, no cálculo de esforço para a fase de testes, apresentado na subseção anterior, consideramos as taxas de geração e regeneração de erros que, devido à alocação de horas extras para as atividades do projeto, afetam as quantidades de erros introduzidos no projeto, já que caso as atividades possuam uma carga horária mais elevada que a normal, a quantidade de erros introduzidos também será aumentada, podendo chegar a 6 erros/PF.

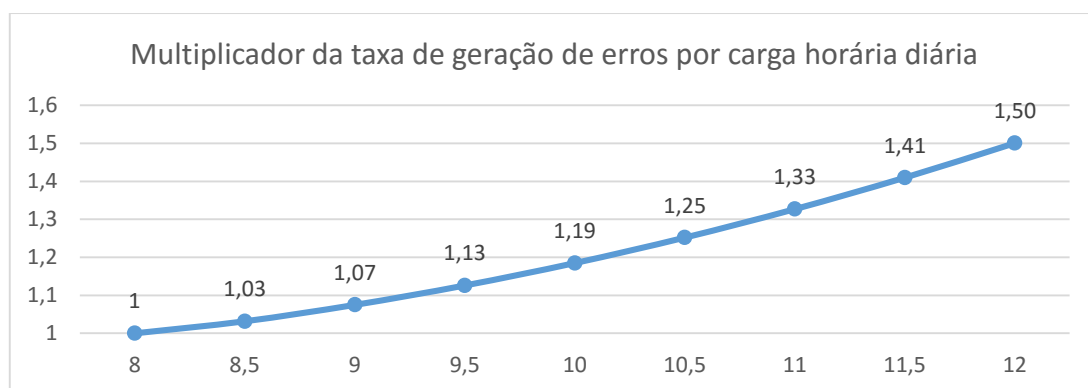


Figura 10 - Gráfico do multiplicador da taxa de geração de erros por carga horária diária

3.1.3. A Necessidade de um Mecanismo de Simulação

Conforme visto nas subseções anteriores, um projeto de desenvolvimento de software é composto por diversas atividades a serem executadas conforme a carga horária definida

para cada uma delas. Este projeto é influenciado pela dinâmica do trabalho em horas extras, devido a alocação individualizada de horas extras para cada atividade do projeto que resulta em impactos na produtividade e na geração de erros, modificando o tempo necessário para a conclusão do projeto.

Estes impactos são representados pelos efeitos positivos (aumento de produtividade) e pelos efeitos negativos (aumento da geração de erros) devido ao trabalho em horas extras. Diferentes perfis de alocação de horas extras (ou seja, diferentes números de horas extras atribuídos a cada atividade) levarão a variações no número de erros que chegam nas atividades de testes, proporcionando modificações no esforço necessário para concluir estas atividades. Como o conjunto destes efeitos não é linear e não pode ser calculado por uma fórmula fechada, utilizaremos um mecanismo de simulação para simular os efeitos da alocação de horas extras sobre um projeto e otimização heurística para buscar a alocação de horas extras que encontre um balanceamento entre os ganhos de produtividade e as perdas decorrentes do aumento na taxa de geração de erros.

O mecanismo de simulação considera a quantidade de desenvolvedores disponíveis para a execução das atividades, assim como a ordem de precedência das atividades do projeto, mencionada na subseção 3.1.1. O simulador funciona por meio de um relógio com intervalos infinitesimais e as atividades vão sendo executadas conforme os intervalos vão passando. Em cada intervalo, parte do esforço necessário para executar uma atividade é consumido. Desta forma, as atividades que não são concluídas ao exato término de um dia de trabalho (considerando a carga horária definida) são continuadas no dia seguinte até que tenham sua conclusão. Para considerar os impactos citados no parágrafo anterior, aplica-se o cenário da dinâmica do trabalho em horas extras à simulação, alterando a duração do dia de trabalho, a quantidade de erros gerados, os custos e a duração do projeto. Somente após a conclusão de todas as atividades que a simulação é encerrada.

3.1.4. A Necessidade de Otimização Heurística

A busca por uma solução que apresente um balanceamento entre a quantidade de erros gerados e o aumento da quantidade de horas trabalhadas devido a horas extras resulta em um espaço de busca bastante grande. Se considerarmos um projeto fictício com 5 grupos de funcionalidades, ou seja, 20 atividades com 9 possibilidades de alocação de carga horária diária de trabalho (8 a 12 horas variando em 30 minutos), teremos 9^{20} possibilidades. Caso o mecanismo de busca gastasse 0,0001 segundos para cada uma das

possibilidades de busca, a busca completa precisaria de 30 milhões de anos para percorrer todo o espaço de busca. Por este motivo, precisamos de uma busca que não analise todas as alternativas, como a busca heurística.

Uma solução candidata para o PPH é representada por uma sequência de números inteiros $X = \{x_1, x_2, x_3, \dots, x_m\}$, onde $1 \leq x_i \leq 9$, para $1 \leq i \leq m$, que representa a quantidade de horas de horas extras para cada uma das atividades do cronograma. Nossa formulação é baseada em uma semana de trabalho normal de 40 horas, uma carga horária diária de trabalho mínima de 8 horas, uma variação de carga horária em múltiplos de 30 minutos até chegar na carga horária diária de trabalho máxima de 12 horas, conforme mencionado na seção anterior.

3.1.5. Os Objetivos da Otimização Heurística

Para avaliar as alternativas encontradas pela busca heurística, enunciamos três objetivos para o PPH: a quantidade de horas extras consumidas (QHE), a duração total do projeto (DTP) e o custo total do projeto (CTP). O PPH é formulado como um problema de decisão multi-objetivo, no qual o QHE, DTP e o CTP são objetivos potencialmente conflitantes a serem minimizados.

Para o cálculo destes objetivos vamos considerar C como o conjunto de caminhos formados pela sequência de atividades representadas pelo grafo $\langle AT, ATDEP \rangle$, considerando as relações de dependência entre as atividades que precisam ser executadas para a conclusão do projeto.

O QHE é obtido pela quantidade de horas extras consumidas na execução das atividades durante o projeto e é calculado pela equação (1). A quantidade de horas extras (qhe) consumida em cada atividade é calculada pelas características da atividade e pelo número máximo de horas extras diárias associadas a ela. Durante a execução de cada atividade, o simulador emula o consumo das horas normais e, em seguida, horas extras pelos recursos. Com isto, o simulador calcula somente a quantidade de horas extras consumidas até que a atividade tenha sido concluída.

$$QHE = \sum_{a \in AT} (qhe_a) \quad (1)$$

O segundo objetivo, o DTP, é obtido pelo caminho mais extenso do conjunto C que é o caminho crítico de um projeto, conforme discutido na subseção 2.2.1 do Capítulo 2, e é calculado pela equação (2). A duração de cada atividade é calculada pelas características da atividade e suas configurações de execução, incluindo o número de

horas extras diárias associadas a ela. Durante a execução de cada atividade, o simulador calcula a quantidade de erros produzidos por cada atividade de desenvolvimento e propaga estes erros para as atividades subsequentes, até alcançar as atividades de testes relacionadas. Para as atividades de testes, o simulador calcula o esforço necessário para identificar e corrigir todos os erros recebidos das atividades anteriores.

$$DTP = MAX_{p \in C} \left(\sum_{a \in p} duração(a) \right) \quad (2)$$

Para o terceiro objetivo, o CTP, somaremos os custos para execução das atividades do projeto. Estes custos consideram a diferenciação entre os custos de horas normais e horas extras previstos na legislação brasileira. Se um trabalho regular custa por hora o valor de X, cada uma das duas primeiras horas extras custa o valor de X acrescido de 20% e cada uma das próximas duas horas extras custa o valor de X acrescido de 50%. Os custos são obtidos pela multiplicação do custo da hora pela duração calculada pelo simulador, considerando cada uma das três possibilidades de variação de custo na jornada de trabalho: hora normal, hora com acréscimo de 20% e hora com acréscimo de 50%.

3.2. Proposta de Solução

Para o PPH formalizado na seção anterior, propomos utilizar a simulação do comportamento dinâmico de um modelo de projeto de desenvolvimento de software, considerando a geração de erros em cada fase do projeto e sua regeneração nas fases subsequentes. Adicionalmente, vamos considerar os efeitos da realização de trabalhos em horas extras, ou seja, o aumento de produtividade e aumento na geração de erros. Associado a este simulador, utilizaremos a Engenharia de Software Baseada em Buscas para buscar pela melhor definição da quantidade de horas extras para cada uma das atividades do projeto segundo os três objetivos definidos na subseção 3.1.5.

A proposta de solução foi dividida em quatro subseções: o simulador, os objetos de simulação e os recursos para o projeto, o cenário de horas extras e a otimização.

3.2.1. O Simulador

A simulação tem a capacidade de evidenciar um comportamento que nossa visão e raciocínio analítico não conseguem facilmente identificar. Um simulador é responsável por copiar o comportamento apresentado por um processo ou sistema do mundo real ao longo do tempo (BANKS *et al.*, 2000). Para isto, o simulador pode manipular diversos

componentes interligados, viabilizando o entendimento do comportamento de sistemas complexos formados por estes componentes.

Conforme Barros, Werner e Travassos (2002), simuladores de tempo contínuo e simuladores baseados em eventos discretos são os mais comumente utilizados para simular projetos de software. Entretanto, até os simuladores mais utilizados possuem limitações, como por exemplo: a incapacidade de separar um comportamento de uma situação específica do comportamento geral, a dificuldade em isolar partes de um modelo relativo a um dado componente e a incapacidade de modelar o consumo simultâneo de um recurso limitado por dois ou mais clientes.

Para superar alguns destes problemas, projetamos um simulador de tempo contínuo utilizando orientação a objetos e baseado em três elementos básicos: objetos de simulação, recursos e cenários. Entendemos este mecanismo de simulação como uma extensão da proposta apresentada em Barros (2001), especialmente focando na incapacidade de modelar o consumo simultâneo de um recurso limitado. Ao invés de criar uma sintaxe específica para representar estes elementos, a linguagem de programação Java foi escolhida para a codificação do simulador, uma vez que facilita a integração com as bibliotecas de buscas heurísticas, necessárias para a otimização.

Objetos de simulação são elementos cujo comportamento queremos simular. Eles representam os nós de um grafo direcionado e acíclico cujas arestas significam dependências entre estes objetos. Eles são inicializados antes da simulação iniciar e ficam aguardando até que todas as suas dependências sejam processadas. Quando as dependências estiverem processadas, os objetos se tornam ativos e iniciam sua participação na simulação, até que sua conclusão seja sinalizada para o simulador. A simulação é encerrada quando o simulador recebe a sinalização de conclusão de todos os objetos de simulação.

Recursos são elementos que representam uma quantidade limitada de materiais ou capacidades que podem ser consumidas pelos objetos de simulação durante um passo de simulação, podendo ser reabastecidos entre os passos de acordo com o comportamento do sistema simulado. Um objeto de simulação pode utilizar parte da quantidade disponível de um recurso, atualizando o seu estado de acordo com a quantidade de recursos disponível. Desta forma, podem existir diversos objetos de simulação sendo executados simultaneamente e consumindo um mesmo recurso. O simulador controla o uso destes recursos limitados em paralelo.

Cenários são elementos que representam comportamentos complementares, que podem ser aplicados a determinados objetos de simulação. Um cenário é escrito para influenciar um tipo de objeto de simulação e só participa na simulação se for ativado sobre objetos deste tipo. Um cenário pode consultar e atualizar o estado dos objetos de simulação a que estiver associado, como por exemplo, para representar as decisões administrativas impostas a um subconjunto das atividades de um projeto de software.

Todos estes elementos são guiados pelo tempo, que é gerenciado de forma contínua pelo simulador. A simulação se inicia no tempo zero e avança por intervalos infinitesimais contabilizados no relógio do simulador, ou seja, os passos consecutivos de simulação. A cada passo, todos os objetos de simulação ativos podem atualizar o seu estado interno. O estado interno dos objetos é definido por um conjunto de informações de acordo com as características do mundo real que estão sendo representadas na simulação. O comportamento de um objeto de simulação é representado pelas alterações observadas em seu estado interno durante a simulação. Os objetos de simulação são objetos da linguagem Java que implementam uma interface de comunicação com o simulador para viabilizar o controle de seus estados internos pelo simulador.

Para controlar a simulação e, conseqüentemente, alterar o estado interno dos objetos de simulação, o simulador consulta os estados internos dos objetos de simulação e utiliza quatro métodos: *init()*, *start()*, *step()* e *finish()*. O método *init()* tem o objetivo de inicializar o objeto de simulação no tempo zero. O método *start()* é chamado para iniciar o ciclo de vida do objeto de simulação quando todas as dependências já estiverem concluídas e o objeto puder participar na simulação. O método *step()* é chamado para que o objeto de simulação execute um novo passo da simulação. O método *finish()* é chamado para que o objeto de simulação seja terminado, já que concluiu suas funções na simulação.

O simulador foi construído com uma arquitetura reusável e flexível de forma que seja possível adaptá-lo para diversas situações de simulação, uma vez que os seus componentes podem ser especializados e customizados de acordo com o sistema que se deseja representar. O sistema que vamos representar na proposta de solução para o PPH descreve o comportamento do projeto de desenvolvimento de software, suas atividades e seus desenvolvedores.

3.2.2. Objetos de Simulação e Recursos para Projetos

Considerando o simulador apresentado na seção anterior, criamos objetos de simulação e recursos para o PPH no formato de um projeto de desenvolvimento de software. Com

isto, podemos verificar o comportamento da execução de um projeto sendo influenciado pelas características de suas atividades e de seus desenvolvedores, assim como os efeitos de um cenário de alocação de horas extras nas suas atividades.

Os desenvolvedores, responsáveis pela execução das atividades, foram modelados como recursos do simulador. Uma das propriedades que definem os desenvolvedores é a limitação da capacidade de realização de trabalho. Esta limitação foi definida em um número de horas de trabalho que se renova em uma base diária, ou seja, se a carga horária diária de trabalho configurada for de 8 horas diárias, o simulador disponibilizará para cada desenvolvedor uma quantidade de 8 horas de trabalho para ser consumida. Caso um desenvolvedor seja alocado em duas ou mais atividades que estejam em execução paralela, as tarefas serão desempenhadas uma de cada vez, ou seja, o mesmo desenvolvedor não fará várias atividades ao mesmo tempo. Nesta situação, o simulador alocará o desenvolvedor para executar as atividades conforme a ordenação topológica imposta pelas dependências.

As atividades foram modeladas como objetos de simulação. Devido à semelhança de comportamento entre os diferentes tipos de atividades, foram modelados dois grandes tipos: desenvolvimento e testes. Conforme previsto em nosso projeto de desenvolvimento de software e devido à calibração dos parâmetros do modelo, as atividades de desenvolvimento foram instanciadas em especificação, projeto e codificação. A atividade de teste teve somente uma instância de mesmo nome, totalizando os quatro tipos de atividades. Esta subdivisão em quatro tipos de atividades foi necessária para a calibração do modelo com parâmetros diferenciados para cada um dos quatro tipos de atividades, conforme discutido na subseção 3.1.1.

O modelo é iniciado com as informações do sistema a ser desenvolvido, assim como do ambiente de desenvolvimento (carga horária e o número de recursos), as características dos desenvolvedores (produtividade média e a taxa de geração média de erros), as características das atividades do projeto de desenvolvimento de software (duração e a taxa de regeneração de erros).

Como o processo de desenvolvimento proposto para a criação do cronograma foi dividido em quatro atividades, construímos o modelo com uma atividade para cada grupo de funcionalidade do sistema a ser desenvolvido. Durante a execução do modelo, as atividades alteram o estado do sistema sendo desenvolvido, passando para especificado após a conclusão da atividade de especificação dos grupos de funcionalidades, depois para projetado e codificado. Após estas atividades, a atividade de testes inicia retirando

os erros introduzidos ao longo do processo e mudando o estado para desenvolvido ao final. Durante a execução de cada atividade, a produtividade média, a taxa de geração média de erros e a taxa de regeneração de erros influenciam diretamente na geração de erros. Estas taxas foram modeladas de forma individualizada por atividade, conforme previsto na subseção 3.1.1.

Para ilustrar a dinâmica da geração de erros, a Figura 11 apresenta a dinâmica da geração, a propagação e a regeneração dos erros de cada uma das atividades do projeto.

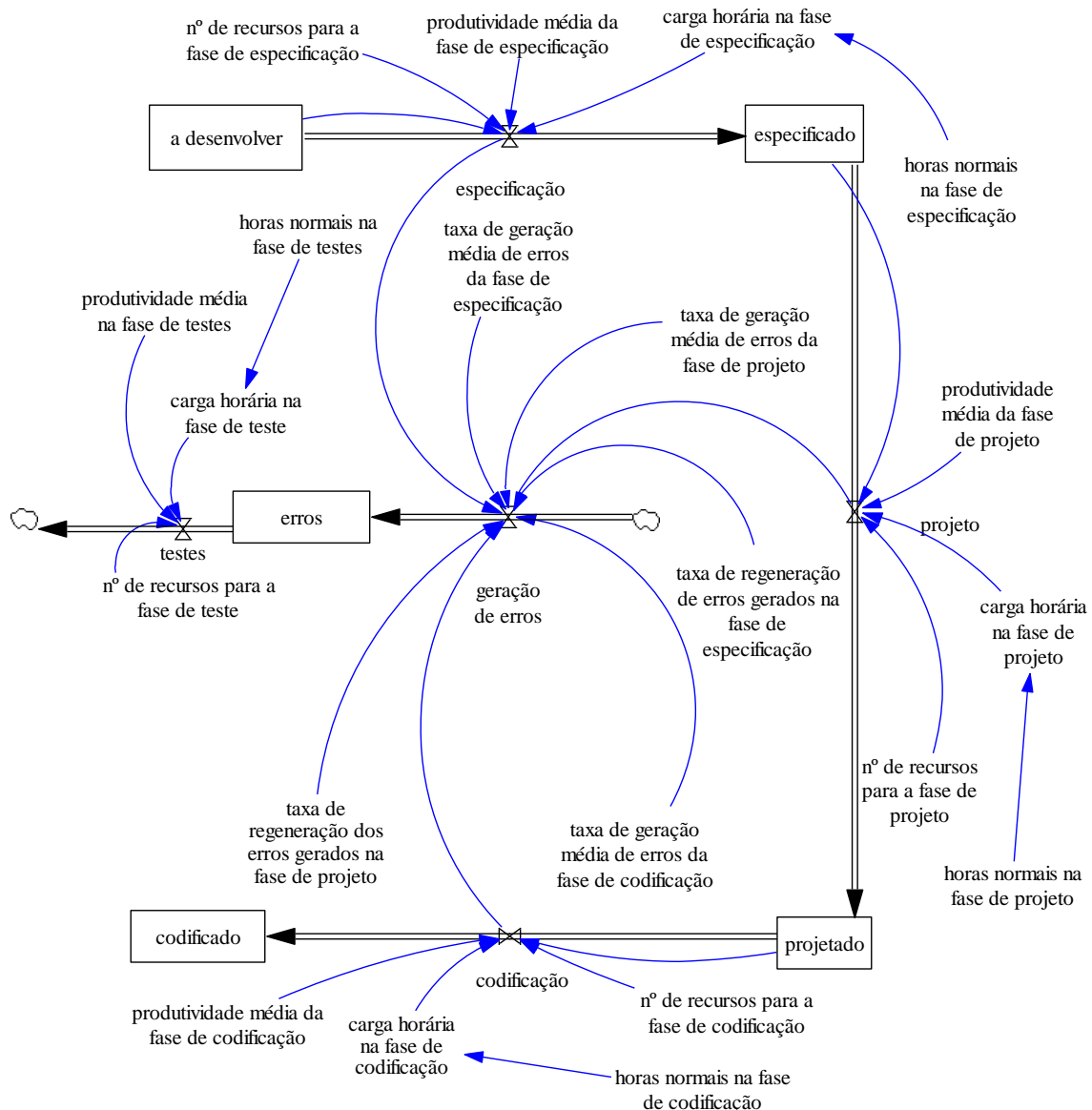


Figura 11 - Modelo de dinâmica de projeto de desenvolvimento de software.

Para a contabilização dos erros introduzidos que necessitam ser corrigidos pelas atividades de testes utilizamos duas equações que calculam a quantidade de pontos de função que foi desenvolvida em cada atividade (QPF, equação 3) e quantos erros foram introduzidos após a conclusão de cada atividade (ERROS, equação 4). A equação 3 é

executada após a conclusão de cada atividade, servindo de entrada para a equação 4, que tem suas taxas preenchidas de acordo com a atividade que está terminando.

$$QPF = \text{produtividade média} \times \text{quantidade de recursos} \times \text{carga horária} \quad (3)$$

$$ERROS = QPF \times \text{taxa de geração média de erros} \times \text{taxa de regeneração de erros} \quad (4)$$

O simulador evidencia o comportamento do projeto com as influências da geração e regeneração dos erros, que se complementam com os efeitos causados pelas decisões tomadas em relação às alocações de horas extras nas atividades do projeto, que serão apresentados na próxima seção.

3.2.3. Cenário de Horas extras

As tomadas de decisões sobre as alocações de horas extras nas atividades dos projetos de desenvolvimento de software podem resultar em impactos nos objetivos do projeto. Por este motivo, o cenário de horas extras foi criado para representar a visão desta teoria, observada por DeMarco e Lister (1999) e retratada nas fórmulas de Abdel-Hamid e Madnick (1991). Este cenário incluiu a dinâmica do trabalho em horas extras e suas influências na produtividade do desenvolvedor e na taxa de geração de erros, conforme apresentado na subseção 3.1.2.

Quando um desenvolvedor é submetido a uma carga de trabalho diária superior a sua carga horária normal, ou seja, 8 horas/dia, sua produtividade poderá ser aumentada conforme o número adicional de horas de trabalho a qual for submetido, limitando-se a 12 horas diárias, conforme permitido pela legislação brasileira (BRASIL, 1943).

Para aplicar este cenário, a Figura 12 apresenta o modelo ilustrado na Figura 11, no qual é acrescentada a taxa de geração de erros adicional devido as horas extras de cada uma das fases, assim como a parcela referente às horas extras é somada na carga horária de cada uma das fases para se capturar o efeito positivo na produtividade.

No início da simulação, o cenário é conectado aos objetos de simulação, ou seja, às atividades que possuem horas extras alocadas e que terão suas taxas de geração de erros modificadas conforme o gráfico da Figura 10, que apresenta as taxas que são consideradas para definir o acréscimo de erros em relação a carga horária alocada. A equação 3, apresentada na seção anterior, que não considerava o aumento de produtividade proporcionado pelo aumento de horas de trabalho decorrente das horas extras foi atualizada pela equação 5. A equação 4, apresentada na seção anterior, com a aplicação deste cenário foi alterada para considerar tanto a taxa de geração adicional de

erros quanto a carga horária de cada uma das atividades, conforme apresentado na equação 6.

$$QPF = \text{produtividade média} \times \text{quantidade de recursos} \times (\text{horas normais} + \text{horas extras}) \quad (5)$$

$$\text{ERROS} = QPF \times \text{taxa de geração média de erros} \times \text{taxa de regeneração de erros} \times \text{taxa de geração adicional de erros devido a horas extras} \quad (6)$$

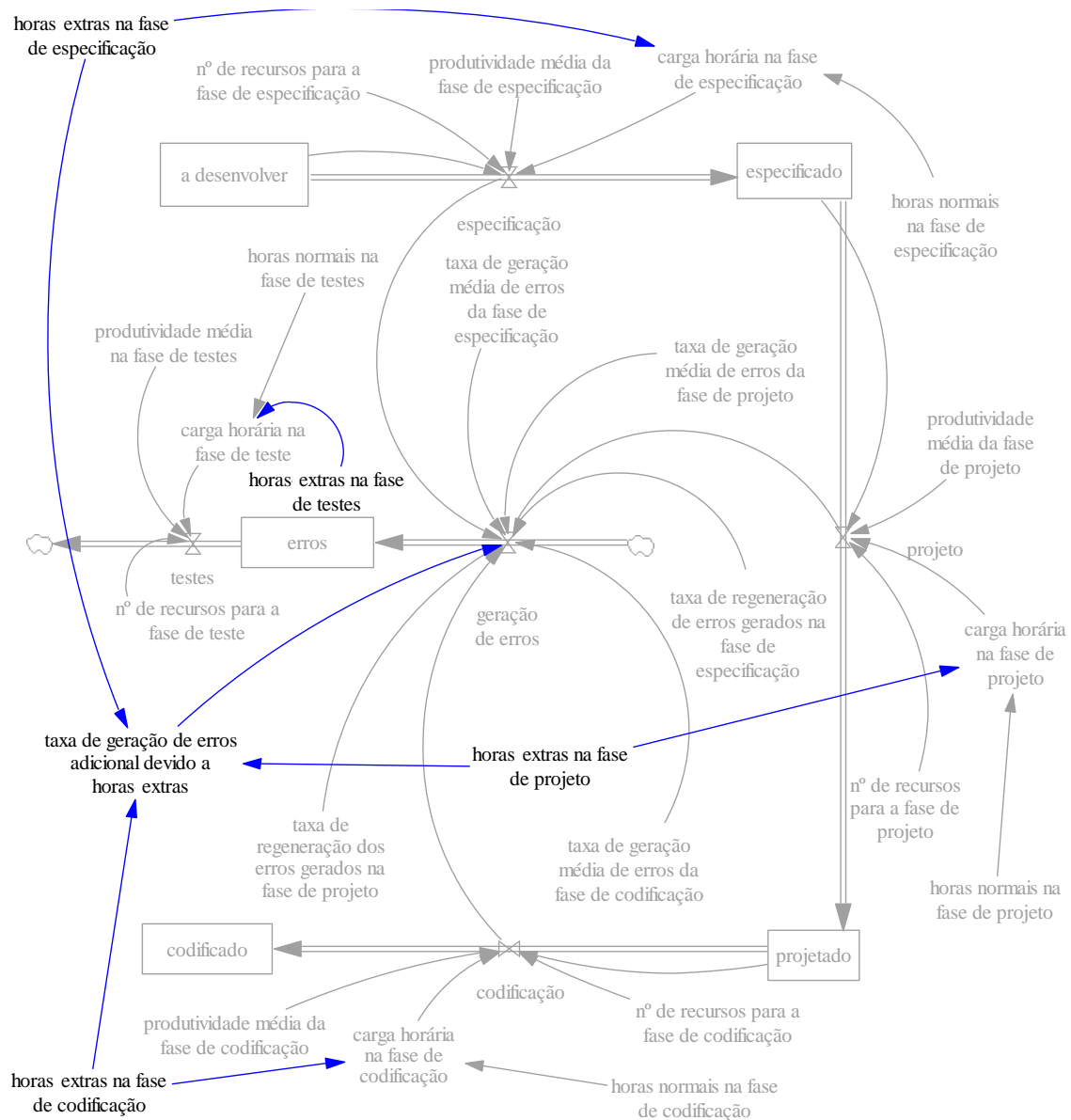


Figura 12 - Modelo de dinâmica de projeto de desenvolvimento de software considerando a geração de erros adicionais pelo trabalho em horas extras.

3.2.4. A Otimização

Em uma otimização assumimos que temos que encontrar o mínimo ou máximo de uma função. Existem poucas classes de problemas para as quais existem técnicas que sempre

encontram o ótimo global. Para casos gerais, é comum utilizar métodos heurísticos, como os algoritmos genéticos. Os algoritmos genéticos (*Genetic Algorithms* - GA) foram desenvolvidos por Holland e sua equipe na universidade de Michigan nas décadas de 60 e 70 (REEVES, 1997).

A motivação para a abordagem do GA foi a analogia biológica com o processo de reprodução de animais e plantas, de forma que os filhos são formados por algumas características que foram determinadas em um nível genético pela combinação dos genes dos cromossomos dos pais. No GA, o conjunto de cromossomos é classificado como uma população. A recombinação dos genes dos cromossomos é realizada utilizando analogias da genética como cruzamento e mutação, sendo a busca guiada pelos resultados das avaliações da função objetivo para cada cromossomo da população. Baseado nas avaliações, os cromossomos que obtiverem melhores resultados são identificados e com isto possuem maior oportunidade de reprodução (REEVES, 1997).

Em relação aos operadores, Reeves (1997) acrescentou que o operador de cruzamento funciona como uma simples troca de alguns genes dos cromossomos pais, gerando dois cromossomos filhos. Este operador pode ser com um único ponto de corte (*single point crossover*), no qual o primeiro filho recebe a primeira parte de seus genes do primeiro pai e a segunda parte de seus genes do segundo pai, intervertendo a origem dos genes dos pais para o segundo filho. O outro operador comum é o operador de mutação que altera aleatoriamente um gene ou um conjunto de genes de um cromossomo, conforme uma probabilidade estabelecida.

As avaliações que são realizadas com a função objetivo podem possuir somente um objetivo, sendo assim classificadas como mono-objetivo e viabilizando a escolha da melhor solução dentre as encontradas nas rodadas da otimização. Entretanto, os problemas de Engenharia de Software tipicamente são multi-objetivo, ou seja, possuem mais de um objetivo a ser otimizado, fazendo com que a escolha da melhor solução por comparação direta não possa ser executada. Por este motivo, para a escolha das melhores soluções utilizamos a Frente de Pareto, que representa um conjunto de soluções não-dominadas, que são consideradas as melhores soluções encontradas durante as rodadas da otimização. Neste conjunto podemos dizer que cada solução não é pior que qualquer outra deste mesmo conjunto, mas também não podemos dizer que é melhor. A Figura 13 exemplifica um gráfico das soluções de uma otimização de minimização de dois objetivos, na qual a área cinza representa o espaço onde as soluções foram posicionadas e a Frente de Pareto é representada pela curva de cor preta que está posicionada na parte

inferior do gráfico, cortando os pontos S1, S2 e S3. Com isto, os pontos S4 e S5 não estão no conjunto das melhores soluções, por não estarem posicionados na Frente de Pareto (HARMAN; MANSOURI; ZHANG, 2012).

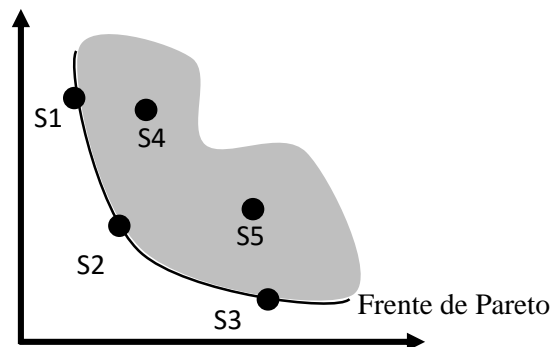


Figura 13 - Gráfico das soluções da otimização bi-objetivo com a Curva de Pareto

Problemas que envolvem balanceamento de objetivos em gerenciamento de projetos (como o PPH), geralmente, não podem ser resolvidos por buscas exaustivas, que examinem todas as possíveis soluções para selecionar a melhor delas, em tempo aceitável devido ao enorme número de possíveis combinações dos componentes de solução. Portanto, utilizamos a otimização heurística baseada no algoritmo genético NSGA-II para encontrar boas soluções (mesmo que não ótimas) para o PPH.

O NSGA-II é um algoritmo baseado no procedimento de *ranking*, que classifica as soluções candidatas de acordo com sua dominância. Uma solução candidata A domina uma solução B se os valores de todos os objetivos da solução A forem melhores ou iguais aos respectivos objetivos da solução B e pelo menos um objetivo na solução A for estritamente melhor do que na solução B. No procedimento de *ranking*, as soluções da Frente de Pareto calculada recebem *ranking* 1, caso seja calculada uma nova Frente de Pareto, as soluções que forem retiradas criam uma nova Frente de Pareto com *ranking* 2. Seguindo este processo, o valor do *ranking* é modificado e novas Frentes de Pareto são criadas na medida que as soluções são retiradas de suas Frentes de Pareto originais (DEB *et al.*, 2002).

O NSGA-II evolui a população aplicando operadores de cruzamento e de mutação, que alteram as soluções para, posteriormente, realizar a seleção dentre as soluções candidatas. O processo de seleção prioriza soluções com menor *ranking* e quando um grupo de soluções possui o mesmo *ranking*, o algoritmo utiliza a medida de densidade. Esta medida evita a concentração de soluções em uma pequena divisão do espaço de soluções, valorizando soluções que estejam mais dispersas/afastadas de outras soluções,

permitindo escolher a solução candidata e cobrindo o espaço de busca da forma mais uniforme possível.

O NSGA-II possui configurações e operadores que precisam ser definidos de forma que sejam customizados para o problema de otimização em questão. A forma como as soluções para o PPH será representada foi apresentada na subseção 3.1.2. O número máximo de avaliações da função objetivo foi estabelecido em 150 mil vezes e o tamanho da população de cromossomos foi estabelecido em 2 vezes o número de atividades do projeto, conforme os resultados apresentados nos experimentos das seções 4.3.2.1 e 4.3.2.2, respectivamente. A necessidade de experimentação para a definição destes parâmetros ocorreu devido à dificuldade em escolher os valores para obtenção dos melhores resultados, assim de forma arbitrária e em virtude da ampla gama de possibilidades.

Para os operadores, definimos o valor 30% de probabilidade de mutação, conforme utilizado por Ferrucci *et al.* (2013), para todas as gerações. Com este operador, cada gene de uma geração é trocado com 30% de probabilidade para um novo valor n compreendido em $\{n \in \mathbb{Z} / 1 \leq n \leq 9\}$. Para o operador de cruzamento escolhemos a adaptação proposta por Ferrucci *et al.* (2013), que apresentou melhor performance em relação ao mais comumente utilizado no NSGA-II. Neste operador, ao invés da troca do material genético entre os dois pais P_1 e P_2 ser realizada após o ponto de corte $C \in \mathbb{Z} / 1 \leq C \leq m$, os resultados O_1 e O_2 são produzidos pela Equação 7, apresentada a seguir. Adicionalmente aos pontos de corte, existe a probabilidade de ocorrência que foi configurada em 50% de probabilidade. Caso contrário, os pais são copiados para as próximas gerações. A seleção dos pais se dá por torneio binário, no qual duas soluções são selecionadas e a melhor solução é mantida.

$$\begin{aligned}
 O_1(g) &= \begin{cases} P_1(g) & 1 \leq g < C \\ \max(P_1(g), P_2(g)) & C \leq g \leq m, p < 0.5 \\ \min(P_1(g), P_2(g)) & C \leq g \leq m, p \geq 0.5 \end{cases} \\
 O_2(g) &= \begin{cases} P_2(g) & 1 \leq g < C \\ (P_1(g) + P_2(g)) / 2 & C \leq g \leq m \end{cases}
 \end{aligned} \tag{7}$$

3.3. Considerações Finais

Para o Problema do Planejamento de Horas extras (PPH) enunciamos três objetivos a serem minimizados: a quantidade de horas extras (QHE), a duração total do projeto (DTP)

e o custo total do projeto (CTP). Para isto, dividimos a proposta de solução em quatro segmentos: o simulador, os objetos de simulação e recursos para o projeto, o cenário de horas extras e a otimização.

Para o PPH foi construído um simulador de tempo contínuo para emular o comportamento de um projeto de desenvolvimento de software que contém atividades e desenvolvedores, envolvendo características de produtividade, geração e regeneração de erros. Associado a esta simulação introduzimos os impactos positivos e negativos do uso de horas extras nas taxas de geração de erros e produtividade dos desenvolvedores. Por fim, utilizamos a otimização, por meio do algoritmo NSGA-II, para buscar o conjunto das melhores soluções para o problema, considerando os três objetivos enunciados.

No próximo capítulo, será apresentado o estudo experimental para avaliar a proposta de solução apresentada neste capítulo.

Capítulo 4 - A Avaliação da Proposta de Solução

Para avaliar a proposta de solução apresentada no Capítulo 3, projetamos um estudo experimental que utilizou instâncias reais de projetos para verificar a competitividade das soluções encontradas pela proposta com práticas utilizadas pelos gerentes de projetos na indústria, assim como a consistência do planejamento de projetos de software quando não consideramos os efeitos nocivos do trabalho em horas extras.

Nas próximas seções enumeramos questões de pesquisa, selecionamos projetos para utilizar como instâncias nas análises, projetamos o experimento, desenvolvemos as integrações com o simulador, executamos o estudo experimental, colhemos e, por fim, analisamos os resultados.

4.1. Questões de Pesquisa

Três questões de pesquisa foram elencadas para nortear o estudo experimental. A primeira questão de pesquisa se dá pela necessidade de verificar se a otimização está sendo conduzida corretamente pelo algoritmo genético. Para isto, faremos a comparação da solução proposta com uma alternativa de otimização não sistemática: o algoritmo de busca aleatória. Esta comparação pode ser considerada como um “testador de sanidade”, conforme previsto em Harman, Mansouri e Zhang (2012). Este teste permite avaliar a solução proposta, pois em nenhum caso o algoritmo aleatório deve produzir melhores soluções que um algoritmo heurístico mais sofisticado, como o utilizado na solução proposta. Caso isto aconteça, haveria três possibilidades: (1) as métricas foram selecionadas de maneira equivocada, não representando melhorias efetivas; (2) houve um erro de implementação do algoritmo, que passou a produzir resultados piores que o algoritmo de busca aleatória; ou (3) o algoritmo não é adequado ao problema. Por estes motivos, temos a seguinte questão de pesquisa:

QP1 (Validação): Existem diferenças vantajosas entre os resultados obtidos pelo algoritmo NSGA-II quando comparado com um algoritmo de busca aleatória?

Hipóteses	
Nula ($H_{0,QP1}$)	Alternativa ($H_{1,QP1}$)
Não há diferenças significativas entre os resultados obtidos pelo algoritmo de busca aleatória e o algoritmo NSGA-II.	Há diferenças significativas entre os resultados obtidos pelos diferentes algoritmos. Inclusive, o NSGA-II apresenta melhores resultados.

Em seguida, comparamos a solução proposta com três estratégias de gerenciamento de horas extras utilizadas na indústria identificadas por Ferrucci *et al.* (2013). A primeira estratégia, batizada de “*Margarine*” (MAR), estabelece uma quantidade de horas extras para todo o projeto e distribui esta quantidade de horas extras uniformemente para todas as atividades do cronograma. O nome desta estratégia foi criado em uma analogia ao ato de espalhar manteiga homogeneamente em um pão. A segunda estratégia, batizada de “*Second Half*” (SH), aplica horas extras somente às atividades da segunda metade do cronograma, de modo a recuperar supostos atrasos imputados nas atividades do início do cronograma. A terceira estratégia, batizada de “*Critical-path Method*” (CPM), atribui as horas extras para as atividades posicionadas no caminho crítico do cronograma, de modo a mitigar o risco de atraso do prazo final. Visando mostrar que a solução proposta é competitiva com as práticas utilizadas na indústria, temos a seguinte questão de pesquisa:

QP2 (Competitividade): As estratégias de gerenciamento de horas extras empregadas na indústria para projetos de software apresentam resultados mais vantajosos quando comparados com a solução proposta?

Hipóteses	
Nula ($H_{0,QP2}$)	Alternativa ($H_{1,QP2}$)
As práticas atualmente empregadas na indústria produzem melhores resultados que a solução proposta.	A solução proposta produz melhores resultados que as práticas atualmente empregadas na indústria.

Em seguida, passamos a avaliar o efeito da redução de qualidade nas atividades executadas a medida que o número de horas extras alocadas aumenta, gerando cansaço e falta de atenção nos desenvolvedores (modelo apresentado no Capítulo 3). Para tanto, a solução proposta será comparada com uma versão que desconsidera os efeitos nocivos do trabalho em horas extras, considerando apenas o ganho de produtividade gerado pelas mesmas. Com isto, esperamos demonstrar que desconsiderar os problemas associados com a alocação de horas extras pode gerar um planejamento inconsistente com

comportamentos postulados na literatura de Engenharia de Software, devido a decisões incorretas tomadas com base em informações incompletas sobre a dinâmica do projeto. Neste sentido, temos a seguinte questão de pesquisa:

QP3 (Consistência): O planejamento de um projeto de desenvolvimento de software é completo mesmo desconsiderando a perda de qualidade causada pelo cansaço do trabalho sob o regime de horas extras?

Hipóteses	
Nula ($H_{0,QP3}$)	Alternativa ($H_{1,QP3}$)
O planejamento do projeto não apresenta diferenças significativas entre os resultados considerando e não considerando a perda de qualidade causada pelo cansaço do trabalho sob o regime de horas extras.	O planejamento do projeto apresenta diferenças significativas entre os resultados considerando e não considerando perda de qualidade causada pelo cansaço do trabalho sob o regime de horas extras.

4.2. Objetos de Estudo

Para aumentar a confiança nos resultados obtidos por meio do experimento escolhemos instâncias de projetos do mundo real. O experimento contempla a utilização de dados de projetos já concluídos. Portanto, não será executado durante a condução do projeto no mundo real e pode ser considerado como *off-line* (WOHLIN *et al.*, 2000).

A seleção de projetos para serem utilizados como instâncias considerou projetos nos quais os integrantes grupo de pesquisa estiveram envolvidos ou aos quais tiveram acesso. Encontramos dificuldade em conseguir acesso a grande quantidade de projetos com a contagem de pontos de função e a quantidade de funções de dados e de funções de transações corretas, uma vez que representantes de empresas/organizações apresentaram receio em ceder informações mesmo que as informações com algum grau de sigilo estivessem descaracterizadas.

Para que as informações dos projetos (definição das funcionalidades e seus respectivos tamanhos funcionais) estivessem registradas de forma padronizada, decidimos pela utilização de informações de projetos coletadas conforme a técnica de Análise de Pontos de Função (APF). Estas informações foram armazenadas em arquivos no formato XML e detalhadas em funções de dados, grupos de funcionalidades, funções transacionais com suas respectivas dependências com outras funções e as quantidades de pontos de função. O conceito de grupo de funcionalidades não é previsto na técnica de Análise de Pontos de Função (APF), porém existem funcionalidades que são relacionadas

com uma mesma informação, como por exemplo, usuário e suas funções transacionais incluir, alterar, excluir e bloquear. Este conceito foi utilizado para agrupar as funções transacionais e as funções de dados ligadas às informações por elas tratadas.

Seguem abaixo as seis instâncias selecionadas para o experimento, com os seus respectivos identificadores e descritivos:

- Acadêmico (ACAD) é um sistema acadêmico para controle de cadastro de turmas, alunos e professores de uma instituição de ensino, bem como matrículas dos alunos;
- Gestão de Pessoas (PSOA) é um sistema para gestão de pessoas que é utilizado por uma empresa de tecnologia de grande porte;
- OPMET (OMET) é um sistema que gerencia, armazena e fornece informações meteorológicas, incluindo mensagens e estatísticas relacionadas ao tempo;
- Parâmetros (PARM) é um sistema que centraliza parâmetros utilizados por diversos sistemas, que precisam de informações compartilhadas por várias aplicações;
- WEBAMHS (WAMS) é um sistema de roteamento de mensagens responsável pelo recebimento e encaminhamento de mensagens de controle de tráfego aéreo;
- WEBMET (WMET) é um sistema de cadastro de informações de observações meteorológicas em banco de dados e fornecimento de relatórios de meteorologia.

ACAD foi desenvolvido para dar suporte a um programa de pós-graduação em uma universidade brasileira e está em uso há mais de 10 anos. WMET, WAMS e OMET estão em uso há mais de 5 anos, oferecendo suporte ao controle de tráfego aéreo no Brasil. PARM e PSOA são utilizados por uma empresa de TI responsável pelo gerenciamento de fundos de aposentadoria no Brasil.

O resumo das instâncias citadas é apresentado na Tabela 3. Esta tabela apresenta a quantidade de funções transacionais, funções de dados, grupos de funcionalidades e a quantidade de pontos de função das instâncias.

Tabela 3 - Resumo das Instâncias

Projeto	Funções Transacionais	Funções de Dados	Grupos de Funcionalidades	Pontos de Função
ACAD	39	7	10	185
OMET	129	22	21	635
PARM	98	21	27	451
PSOA	65	9	18	390
WAMS	67	8	15	381
WMET	48	7	11	225

Para a execução do experimento foi construído um software utilizando a linguagem Java com o JDK (*Java Development Kit*) na versão 7. Esta escolha de

linguagem de programação foi realizada pela familiaridade do grupo de pesquisa, preferências pessoais e a existência de bibliotecas disponíveis já utilizadas em outros experimentos, como por exemplo, em (FERRUCCI *et al.*, 2013).

4.3. Execução dos Experimentos e Análise dos Resultados

O experimento está dividido em três subseções: definição e cálculo dos indicadores de qualidade, geração das soluções e a análise dos resultados.

4.3.1. Definição e Cálculo dos Indicadores de Qualidade

Conforme apresentamos na subseção 3.2.4, a comparação simples e direta das soluções até que a melhor, dentro do espaço de busca, seja encontrada, pode ser realizada quando a otimização envolver somente um objetivo. Entretanto, quando ela envolve dois ou mais objetivos, a comparação simples das soluções não será suficiente para garantir a escolha da melhor solução. Desta forma, devido a quantidade de objetivos que temos na otimização deste experimento utilizamos nos resultados as Frentes de Pareto (F).

Para comparar duas frentes F_A e F_B , nós precisamos definir as medidas que irão avaliar a qualidade destas frentes. Estas medidas são chamadas de indicadores de qualidade. Um indicador de qualidade recebe um conjunto de pontos definidos no espaço dos objetivos e produz um número real. Desta forma, o resultado de uma frente F_A pode ser comparado ao resultado de uma outra frente F_B , levando a conclusões sobre a qualidade destas frentes e, conseqüentemente, às configurações dos algoritmos que as produziram. Os indicadores de qualidade normalmente comparam cada uma das frentes com uma frente de referência chamada de F_R . Esta última é formada pelas soluções não-dominadas de todas as frentes usadas na comparação, ou seja, esta frente será a melhor aproximação do conjunto ótimo de soluções não-dominadas.

Para os nossos experimentos, escolhemos fazer uma avaliação com indicadores de convergência e de diversidade. Os indicadores de qualidade relacionados com a convergência medem a proximidade de uma frente F com a frente de referência F_R , enquanto os indicadores de diversidade medem a quantidade do espaço de busca que foi coberto pela frente F . Considerando estas propriedades, escolhemos quatro indicadores: *Contributions* (I_C), *Hypervolume* (I_{HV}), *Generational Distance* (I_{GD}) e *Spread* (I_{SP}).

Contributions (I_C) é um indicador de convergência que representa a quantidade de soluções de uma frente F que também pertencem a frente F_R . Quanto maior for o valor de

I_C para uma frente F , melhor será considerada esta frente, pois ela contribuirá com um maior número de pontos para a frente de referência (F_R). Na Figura 14, F_R é formada pelas soluções das duas frentes (F_A , F_B), F_A apresenta uma quantidade menor de soluções que F_B na F_R , fazendo com que, neste exemplo, F_B tenha um I_C melhor que F_A .

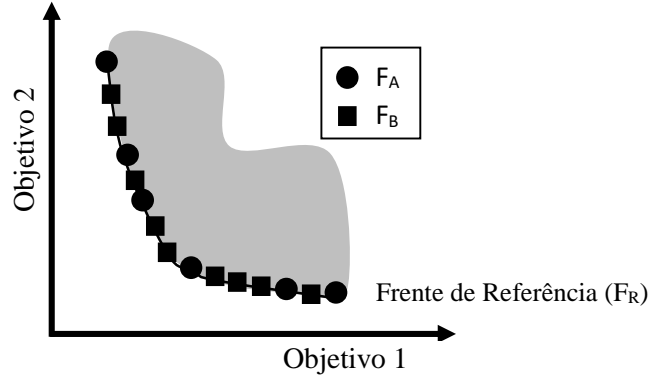


Figura 14 - Representação do *Contributions*

Hypervolume (I_{HV}) é um indicador de convergência e diversidade apresentado na Figura 15 e é obtido pelo cálculo do volume da região coberta entre o conjunto não-dominado de soluções da frente F_R e um ponto de referência W . Este ponto de referência W é formado pelas piores soluções. Para cada solução pertencente a F_R constroi-se um hipercubo com referência no ponto W . O cálculo do volume de todos os hipercubos encontrados resulta nesta métrica. Quanto maior for o valor de I_{HV} para uma frente F , melhor será considerada esta frente, pois ela apresentará uma menor distância entre os valores e maior espalhamento na frente de referência, ou seja, cobrindo uma grande extensão do espaço de busca (DURILLO *et al.*, 2009).

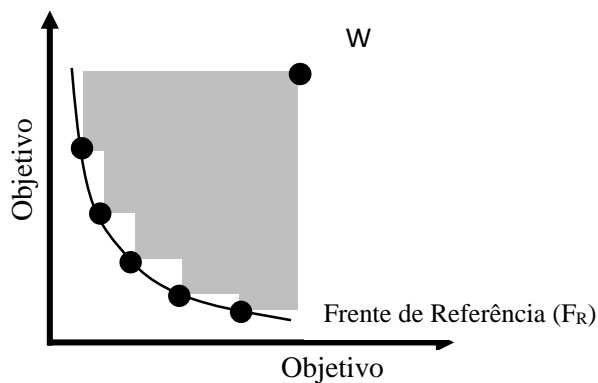


Figura 15 - Representação do *Hypervolume*

Generational Distance (I_{GD}) é um indicador de convergência apresentado na Figura 16, que calcula a distância média dos pontos entre a solução obtida e a frente de referência (F_R) (DEB *et al.*, 2002). Quanto menor for o valor de I_{GD} para uma frente F ,

melhor será considerada esta frente, pois ela apresentará uma menor distância aos valores da frente de referência.

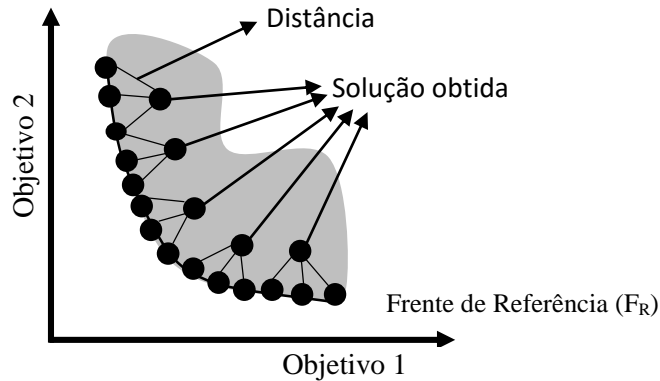


Figura 16 - Representação do *Generational Distance*

Spread (I_{SP}) é um indicador de diversidade apresentado na Figura 17. Ele calcula o espalhamento das soluções considerando a distância euclidiana entre as soluções (DEB *et al.*, 2002). Quanto menor for o valor de I_{SP} para uma frente F , melhor será considerada esta frente, pois ela apresentará uma menor distância entre os valores da frente de referência, indicando menor dispersão das soluções na frente de referência.

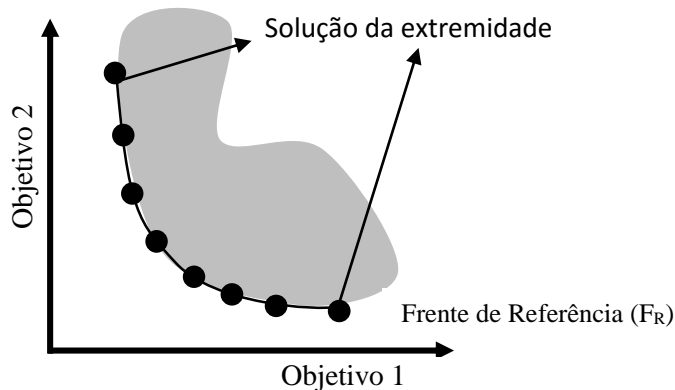


Figura 17 - Representação do *Spread*

Os quatro indicadores de qualidade acima foram implementados com base no *framework* jMetal (DURILLO *et al.*, 2009), uma biblioteca Java que implementa diversos algoritmos heurísticos mono- e multi-objetivo e os mecanismos para permitir a comparação entre estes algoritmos (como os indicadores de qualidade).

4.3.2. Parametrização do Algoritmo

Na solução proposta, conforme citado na subseção 3.2.4, temos parâmetros que não puderam ser arbitrariamente definidos conforme explicado anteriormente, necessitando de experimentação para a calibração correta do algoritmo com os melhores resultados.

Por este motivo, executamos experimentos com o algoritmo implementado para variações do número máximo de avaliações, apresentado na subseção 4.3.2.1. Para terminar a calibração do algoritmo proposto, executamos a experimentação para o tamanho da população, apresentado na subseção 4.3.2.2.

4.3.2.1. Número Máximo de Avaliações

A definição do parâmetro de número máximo de avaliações foi realizada por experimentação e análise inferencial estatística. Este parâmetro define a quantidade de vezes que são calculados os objetivos do problema durante uma execução do algoritmo. Pela dificuldade de definir arbitrariamente este parâmetro, o algoritmo foi executado para todas as instâncias com os valores máximos de avaliações definidos para 5.000 (5K), 10.000 (10K), 20.000 (20K), 50.000 (50K), 100.000 (100K) e 150.000 (150K). Para isto, utilizamos os parâmetros configurados conforme a subseção 3.2.4. Em relação ao tamanho da população, definimos como 2 vezes a quantidade de atividades.

Devido a existência de componentes aleatórias nos algoritmos de busca, é necessário examinar a média e a dispersão de diversos ciclos de execução ao algoritmo para garantir a representatividade dos resultados. Para evitar considerar apenas resultados muito bons ou muito ruins de uma configuração do algoritmo, executamos cada configuração 50 vezes. Desta forma, após a conclusão das execuções, cada instância terá 50 conjuntos de soluções candidatas não-dominadas do algoritmo.

A avaliação dos resultados se deu pela análise das médias e desvios padrão dos valores calculados para os indicadores de qualidade escolhidos e pelo teste de inferência estatística não-paramétrico de Kruskal-Wallis com 95% de confiança, que determina se existe diferença significativa entre as médias de um par de configurações.

Considerando os indicadores de qualidade apresentados na subseção 4.3.1, escolhemos analisar um indicador de convergência e um indicador de diversidade. Portanto, descartamos o *Hypervolume* (I_{HV}) por ser um indicador misto e o *Contributions* (I_C) por ter apresentado grande proximidade nos resultados colhidos para as diversas configurações, com todas as médias iguais até a quarta casa decimal e diferenças apenas na segunda casa decimal do desvio padrão. Sendo assim, analisamos o *Generational Distance* (I_{GD}) e o *Spread* (I_{SP}), que são apresentados na Tabela 4 e na Tabela 5, respectivamente. Os valores em negrito representam, respectivamente, o menor I_{GD} e o menor I_{SP} .

Tabela 4 - Valores de média e desvio padrão do I_{GD} para as configurações elencadas para o número máximo de avaliações em todas as instâncias, marcando em negrito os melhores valores

INST	5K	10K	20K	50K	100K	150K
ACAD	0,0093±0,0024	0,0089±0,0019	0,0083±0,0017	0,004 ±0,0012	0,0035±0,0016	0,0033 ±0,001
OMET	0,0064±0,0008	0,0051±0,0006	0,0041±0,0006	0,0027±0,0003	0,0024±0,0002	0,002 ±0,0002
PARM	0,0133±0,0019	0,0086±0,0008	0,0091±0,0009	0,0069±0,0013	0,0057±0,0008	0,0046±0,0005
PSOA	0,0173±0,0065	0,0157±0,0048	0,0118±0,0056	0,0055±0,0029	0,0042±0,0016	0,0046±0,0025
WAMS	0,0046±0,0005	0,004 ±0,0005	0,0032±0,0003	0,0019±0,0003	0,0018±0,0002	0,0018±0,0004
WMET	0,0063±0,0008	0,0061 ±0,001	0,0037±0,0003	0,0023±0,0004	0,0023±0,0004	0,002 ±0,0004

Tabela 5 - Valores de média e desvio padrão do I_{SP} para as configurações elencadas para o número máximo de avaliações em todas as instâncias, marcando em negrito os melhores valores

INST	5K	10K	20K	50K	100K	150K
ACAD	0,6517±0,0568	0,6509±0,0562	0,6441 ±0,052	0,5268±0,0622	0,4853±0,0574	0,4575±0,0489
OMET	0,6559±0,0519	0,6447±0,0367	0,6325±0,0335	0,6237 ±0,036	0,6416±0,0404	0,6227±0,0392
PARM	0,6619±0,0689	0,6711±0,0548	0,6565±0,0468	0,6761±0,0519	0,6636±0,0478	0,6701±0,0431
PSOA	0,6716±0,084	0,6595±0,0713	0,6807±0,0871	0,6946 ±0,088	0,6414±0,0619	0,63 ±0,0739
WAMS	0,629±0,0478	0,6066±0,0334	0,5977±0,0354	0,4951±0,0249	0,4616±0,0328	0,4436±0,0354
WMET	0,6186±0,0521	0,5987±0,0424	0,5705±0,0399	0,4821 ±0,036	0,4344 ±0,044	0,4301±0,0335

O I_{GD} e o I_{SP} obtiveram valores convergentes em uma das configurações elencadas para o parâmetro de número máximo de avaliações. A Tabela 6 e a Tabela 7 apresentam os resultados do teste de Kruskal-Wallis com 95% de nível de confiança, mostrando as instâncias que apresentaram semelhança em seus resultados comparando duas configurações de número máximo de avaliações, segundo a combinação dos valores apresentados nos títulos dos eixos vertical e horizontal. Por exemplo, na Tabela 6 verificamos que as instâncias ACAD, PSOA e WAMS não demonstraram diferenças significativas entre as configurações de 100K e 150K.

Tabela 6 - Indicação de semelhança nos resultados entre as configurações de número máximo de avaliações para todas as instâncias considerando o I_{GD}

	100K	10K	150K	20K	50K
10K					
150K	ACAD PSOA WAMS				
20K	ACAD PARM				
50K	WAMS WMET				
5K	ACAD PSOA WMET				
	ACAD				

Tabela 7 - Indicação de semelhança nos resultados entre as configurações de número máximo de avaliações para todas as instâncias considerando o I_{SP}

	100K	10K	150K	20K	50K
10K	OMET PARM PSOA				
150K	OMET PARM PSOA WAMS WMET	OMET PARM PSOA			
20K	OMET PARM PSOA	ACAD OMET PARM PSOA WAMS	OMET PARM PSOA		
50K	OMET PARM	PARM OMET PSOA	OMET PARM	OMET PARM	
5K	OMET PARM PSOA	ACAD OMET PARM PSOA WAMS WMET	PARM PSOA	ACAD OMET PARM PSOA	PARM PSOA

Considerando o IGD, os resultados mostraram que a única instância que não obteve melhores resultados para 150k, conforme Tabela 4, foi a instância PSOA. Entretanto, conforme observado na Tabela 6, não foi identificada diferença significativa entre os resultados obtidos nas configurações com 100K (que apresentou a menor média observada para a instância) e 150K para esta instância. Considerando o ISP, os resultados mostraram que a única instância que não obteve melhores resultados para a configuração 150K, conforme Tabela 5, foi a instância PARM. Entretanto, conforme observado na Tabela 7, também não foi identificada diferença significativa entre os resultados obtidos nas configurações com 20K (que apresentou a menor média observada para a instância) e 150K para esta instância. Sendo assim, definimos o parâmetro de número máximo de avaliações em 150.000 para todas as instâncias.

4.3.2.2. Tamanho de População

A definição do parâmetro de tamanho de população foi realizada por experimentação e análise inferencial estatística. Tendo fixado o número máximo de avaliações, este parâmetro define a quantidade de gerações que serão permitidas durante a execução do algoritmo e o número de indivíduos na população em cada geração.

Pela dificuldade de definir arbitrariamente este parâmetro, o algoritmo foi executado para todas as instâncias com os valores de tamanho de população definidos

pela multiplicação da quantidade de atividades no projeto pelos valores 2, 4 e 8. Para isto, utilizamos os mesmos parâmetros configurados para o experimento de definição do número máximo de avaliações, conforme apresentado na subseção 4.3.2.1. Em relação ao parâmetro de número máximo de avaliações, utilizamos o valor de 150.000, que foi o valor definido no experimento da subseção anterior.

A avaliação dos resultados se deu pela análise das médias e desvios padrão dos valores calculados para os indicadores de qualidade escolhidos e pelo teste de inferência estatística não-paramétrico de Kruskal-Wallis com 95% de confiança. Assim como no experimento anterior, analisamos o *Generational Distance* (I_{GD}) e o *Spread* (I_{SP}), que são apresentados na Tabela 8 e na Tabela 9, respectivamente. Os valores em negrito representam, respectivamente, o menor I_{GD} e o menor I_{SP} .

Tabela 8 - Valores de média e desvio padrão do I_{GD} para as configurações elencadas para o parâmetro de tamanho da população em todas as instâncias, marcando em negrito os melhores valores.

INSTÂNCIA	2X	4X	8X
ACAD	0,0033±0,001	0,0034±0,0007	0,0047±0,0012
OMET	0,002±0,0002	0,0024±0,0002	0,0028±0,0002
PARM	0,0046±0,0005	0,0054±0,0005	0,0057±0,0005
PSOA	0,0046±0,0025	0,0052±0,0021	0,0075±0,0044
WAMS	0,0018±0,0004	0,0017±0,0002	0,0018±0,0002
WMET	0,002±0,0004	0,002±0,0002	0,0019±0,0002

Tabela 9 - Valores de média e desvio padrão do I_{SP} para as configurações elencadas para o parâmetro de tamanho da população em todas as instâncias, marcando em negrito os melhores valores.

INSTÂNCIA	2X	4X	8X
ACAD	0,4575±0,0489	0,6392±0,0417	0,6543±0,05
OMET	0,6227±0,0392	0,6256±0,0379	0,629±0,0429
PARM	0,6701±0,0431	0,6621±0,0518	0,6704±0,0438
PSOA	0,63±0,0739	0,668±0,0678	0,7249±0,1105
WAMS	0,4436±0,0354	0,6062±0,0296	0,6114±0,0328
WMET	0,4301±0,0335	0,5864±0,0301	0,6042±0,0275

O I_{GD} e o I_{SP} obtiveram valores convergentes em uma das opções elencadas para o parâmetro de tamanho da população. A Tabela 10 e a Tabela 11 apresentam os resultados do teste de Kruskal-Wallis, mostrando as instâncias que não apresentaram diferença significativa em seus resultados comparando duas configurações de tamanho da população, segundo a combinação dos valores apresentados nos títulos dos eixos vertical e horizontal.

Tabela 10 - Indicação de semelhança nos resultados entre as configurações de tamanho da população e todas as instâncias considerando o I_{GD} .

	2X	4X
4X	ACAD WMET	
8X	WAMS WMET	WMET

Tabela 11 - Indicação de semelhança nos resultados entre as configurações de tamanho da população e todas as instâncias considerando o I_{SP} .

	2X	4X
4X	OPMET PARM	
8X	OPMET PARM	ACAD OPMET PARM WAMS

Considerando o I_{GD} , a Tabela 8 mostra que as instâncias que não obtiveram melhores resultados para a configuração 2x foram WAMS e WMET. Entretanto, conforme observado na Tabela 10, não foi identificada diferença significativa entre os resultados obtidos nas configurações com 2x e 8x para a instância WMET. Para a instância WAMS foi identificada diferença significativa entre os resultados obtidos nas configurações com 2x e 4x, porém foi obtido um *p-value* de 0.04446599, bastante próximo de 0,05 (indicador da semelhança).

Considerando o I_{SP} , a Tabela 9 mostra que a única instância que não obteve melhores resultados para 2x foi a instância PARM. Entretanto, conforme observado na Tabela 11, não foi identificada diferença significativa entre os resultados obtidos nas configurações com 2x e 4x para esta instância.

Com base nos resultados acima, definimos o parâmetro de tamanho de população em 2x para todas as instâncias. Após a finalização da calibração do algoritmo NSGA-II, este foi executado para que seus resultados pudessem ser colhidos e armazenados para utilização nas respostas às questões de pesquisa.

4.3.3. A Análise dos Resultados

A análise dos resultados foi dividida conforme as questões de pesquisa apresentadas na seção 4.1, apresentado um consolidado ao final. Ela consiste na comparação direta dos indicadores de qualidade apresentados na subseção 4.3.1 para a execução de testes inferenciais estatísticos. Como os valores analisados são formados por $v \in \mathbb{Q}_+ / 0 \leq v \leq 1$, não foi necessária a realização de testes de verificação de normalidade dos dados, ou

seja, excluiu-se a possibilidade de que os dados tivessem distribuição próxima à curva normal e, consequentemente, a possibilidade de realização de testes paramétricos. Sendo assim, todas as análises discutidas a seguir se baseiam em testes inferenciais e medidas de tamanho de efeito não-paramétricos, por meio da utilização da ferramenta R¹⁰ na versão 3.1.1.

4.3.3.1. Resposta ao Teste de Sanidade (QP1)

Visando a resposta para a QP1, utilizamos o algoritmo aleatório (*RandomSearch*¹¹), também proveniente do *framework* jMetal (DURILLO *et al.*, 2009). Este algoritmo, como o nome já informa, gera soluções de forma aleatória. Estas soluções foram associadas às atividades e posteriormente avaliadas, resultando em um conjunto de soluções não-dominadas em uma frente de Pareto. Como o algoritmo de busca aleatória também possui os parâmetros de número de ciclos e o número máximo de avaliações, utilizamos os mesmos valores associados ao algoritmo NSGA-II da solução proposta, ou seja, executamos 50 ciclos do algoritmo e estabelecemos o número máximo de avaliações em 150 mil cálculos das funções de *fitness*.

Após a execução do algoritmo de busca aleatória, foi realizada a comparação entre os resultados de todos os indicadores de qualidade da solução proposta e do algoritmo de busca aleatória. O teste estatístico não-paramétrico de Wilcoxon-Mann-Whitney com 95% de confiança foi executado para encontrar a probabilidade dos dois conjuntos de dados apresentarem médias iguais, ou seja, $p\text{-value} < 0,05$ (BARROS, 2012; DE SOUZA *et al.*, 2011). Em complemento foi utilizada a medida de tamanho de efeito não-paramétrica \hat{A}_{12} de Vargha e Delaney (2000). Este cálculo permite identificar o percentual de vezes em que um conjunto obtém resultados melhores que o outro nas execuções observadas (BARROS, 2012). Os resultados dos testes estatísticos para os indicadores de qualidade das execuções com os dois algoritmos foram analisados e consolidados na Tabela 12.

Conforme pode ser observado na Tabela 12, o tamanho de efeito (ES) próximo ou igual a 100%, assim como o $p\text{-value}$ sempre assumindo valores menores que 0,01, mostram a superioridade do algoritmo NSGA-II em encontrar boas soluções para o PPH. Os valores do $p\text{-value}$ apresentaram diferença significativa para todas as instâncias e

¹⁰ <http://www.R-project.org>

¹¹ `jmetal.metaheuristics.randomSearch.RandomSearch`

indicadores de qualidade avaliados com 95% de confiança. Analisando o tamanho de efeito (ES), pode-se observar que em todos os indicadores de qualidade e instâncias, o algoritmo NSGA-II apresentou melhores resultados em pelo menos 94,4% das execuções pelo I_{SP} , 96% das execuções pelo I_C e 100% das execuções pelo I_{HV} e I_{GD} .

Tabela 12 - Resultados do p -value (PV) do teste de Wilcoxon-Mann-Whitney e o tamanho de efeito (ES) de \hat{A}_{12} Vargha-Delaney comparando os resultados da solução proposta e do algoritmo aleatório para todas as instâncias e indicadores de qualidade.

INSTÂNCIA	I_C		I_{HV}		I_{GD}		I_{SP}	
	PV	ES	PV	ES	PV	ES	PV	ES
ACAD	<0,01	100,0	<0,01	100,0	<0,01	100,0	<0,01	100,0
OMET	<0,01	100,0	<0,01	100,0	<0,01	100,0	<0,01	94,4
PARM	<0,01	96,0	<0,01	100,0	<0,01	100,0	<0,01	98,7
PSOA	<0,01	99,0	<0,01	100,0	<0,01	100,0	<0,01	95,8
WAMS	<0,01	100,0	<0,01	100,0	<0,01	100,0	<0,01	100,0
WMET	<0,01	100,0	<0,01	100,0	<0,01	100,0	<0,01	100,0

Este resultado somente confirmou as expectativas de que o algoritmo NSGA-II apresentaria superioridade em relação ao algoritmo aleatório, servindo como um teste de sanidade para a proposta de solução. Desta forma, podemos rejeitar a hipótese nula e assumir a hipótese alternativa que postula a superioridade do algoritmo NSGA-II em relação a uma busca não sistemática para o PPH.

4.3.3.2. Resposta ao Teste de Competitividade (QP2)

Para responder a QP2, implementamos as três estratégias de gerenciamento de horas extras identificadas por Ferrucci *et al.* (2013): *Margarine* (MAR), *Second Half* (SH) e o *Critical-path Method* (CPM). Para definir o conjunto de soluções do MAR, geramos as soluções que contemplem cada uma das variações previstas de quantidade de horas extras igualmente para todas as atividades previstas nos projetos. Para construir o conjunto de soluções do SH, geramos as soluções que contemplam cada uma das variações previstas de quantidade de horas extras alocadas na segunda metade das atividades previstas nos projetos. Para criar o conjunto de soluções do CPM, geramos as soluções que contemplam cada uma das variações previstas de quantidade de horas extras para as atividades previstas nos projetos que se encontram no caminho crítico. Para cada uma das três estratégias de gerenciamento de horas extras foram geradas nove soluções (número de horas extras compreendido entre 0 a 4 com variações de 30 minutos em cada atividade compatível com a estratégia), que foram avaliadas através da simulação, resultando

somente as soluções não-dominadas em uma frente de Pareto para cada uma das estratégias.

Em virtude de as estratégias praticadas pela indústria não envolverem componentes aleatórios (como o NSGA-II), elas não foram executadas múltiplas vezes. Foram comparadas as frentes de Pareto resultantes de cada uma das configurações (50 ciclos do NSGA-II, MAR, SH e CPM). Seguem, na Tabela 13, os resultados dos indicadores de qualidade, visando uma comparação de todas as configurações.

Tabela 13 - Valores de médias e desvios padrão para todos os indicadores de qualidade em todas as instâncias para o NSGA-II e valores observados para as estratégias de gerenciamento de horas extras utilizadas pela indústria (MAR, SH e CPM), marcando em negrito os melhores valores.

INSTÂNCIA	CONFIG	I _c	I _{HV}	I _{GD}	I _{SP}
ACAD	NSGA-II	0,0198±0,009	0,5160±0,0024	0,0024±0,0007	0,5063±0,0304
	MAR	0,0024	0,2303	0,1783	0,7194
	SH	0,0072	0,3589	0,0488	0,1194
	CPM	0,0048	0,0663	n/a	0,9485
OMET	NSGA-II	0,0196±0,0104	0,4561±0,0032	0,0017±0,0002	0,6633±0,0336
	MAR	0,0076	0,2545	0,2346	0,6774
	SH	0,0115	0,3544	0,0300	0,1195
	CPM	0,0076	0,2089	0,1356	0,5164
PARM	NSGA-II	0,0190±0,0121	0,4211±0,0031	0,0034±0,0003	0,7247±0,0312
	MAR	0,0137	0,2308	0,1518	0,7151
	SH	0,0342	0,3579	0,0215	0,1182
	CPM	0,0137	0,0498	n/a	0,9613
PSOA	NSGA-II	0,0197±0,0088	0,3327±0,0033	0,0056±0,0031	0,6379±0,0775
	MAR	0,0063	0,1496	0,2309	0,6891
	SH	0,0063	0,2069	0,2725	0,5156
	CPM	0,0063	0,0633	n/a	0,9353
WAMS	NSGA-II	0,0197±0,0100	0,4922±0,0027	0,0016±0,0003	0,4767±0,0262
	MAR	0,0053	0,2547	0,2414	0,6772
	SH	0,0079	0,3547	0,0418	0,1198
	CPM	0,0053	0,1718	0,1510	0,7761
WMET	NSGA-II	0,0198±0,0110	0,4980±0,0027	0,0019±0,0004	0,4565±0,0308
	MAR	0,0055	0,2544	0,2420	0,6776
	SH	0,0055	0,3538	0,0418	0,1200
	CPM	0,0055	0,1913	0,1530	0,5646

Conforme pode ser observado na Tabela 13, o algoritmo NSGA-II apresenta maior contribuição (I_c) em todas as instâncias, com exceção de PARM, que obteve melhor resultado para a estratégia SH. Em relação ao I_{HV} e ao I_{GD}, para todas as instâncias os resultados obtidos pelo NSGA-II foram melhores. Na estratégia CPM para ACAD, PARM e PSOA, não foi calculado o I_{GD} devido à quantidade de soluções ser menor que

três, inviabilizando o cálculo deste indicador. Em relação ao I_{SP} , os resultados foram melhores para o SH em todas as instâncias, uma vez que este indicador premia soluções nas extremidades da frente resultante e as frentes geradas pela estratégia SH possuem poucas soluções, porém bem localizadas em relação aos extremos do uso de horas extras nas atividades. Buscando identificar o tamanho de efeito entre estas configurações, calculamos a medida não-paramétrica \hat{A}_{12} de Vargha e Delaney (2000), conforme apresentado na Tabela 14.

Tabela 14 - O tamanho de efeito \hat{A}_{12} de Vargha-Delaney considerando o algoritmo NSGA-II e cada uma das estratégias de alocação de horas extras praticadas pela indústria para todas as instâncias e indicadores de qualidade. O tamanho de efeito mostra as chances do NSGA-II obter um maior valor para o I_C e o I_{HV} e um menor valor para o I_{SP} e o I_{GD} , mostrando a superioridade do NSGA-II.

INSTÂNCIA	CONFIG	I_C	I_{HV}	I_{GD}	I_{SP}
ACAD	MAR	99%	100%	100%	100%
	SH	94%	100%	100%	0%
	CPM	97%	100%	n/a	100%
OMET	MAR	93%	100%	100%	64%
	SH	81%	100%	100%	0%
	CPM	93%	100%	100%	0%
PARM	MAR	64%	100%	100%	38%
	SH	14%	100%	100%	0%
	CPM	64%	100%	n/a	100%
PSOA	MAR	92%	100%	100%	82%
	SH	92%	100%	100%	0%
	CPM	92%	100%	n/a	100%
WAMS	MAR	97%	100%	100%	100%
	SH	92%	100%	100%	0%
	CPM	97%	100%	100%	100%
WMET	MAR	90%	100%	100%	100%
	SH	90%	100%	100%	0%
	CPM	90%	100%	100%	100%

Conforme apresentado na Tabela 14, o tamanho de efeito do I_{GD} na estratégia CPM para ACAD, PARM e PSOA não foi calculado devido à quantidade de soluções ser menor que três, inviabilizando o cálculo deste indicador. A Tabela 14 apresenta evidências que o NSGA-II é superior em relação às estratégias de gerenciamento de horas extras para todos os indicadores de qualidade, perdendo somente no I_{SP} que premia soluções nas extremidades da frente resultante, situação que ocorre na frente do SH.

Visando proporcionar uma comparação visual entre a distribuição das 50 execuções do NSGA-II e cada estratégia de gerenciamento de horas extras foram gerados diagramas *boxplots* para as configurações e os indicadores de qualidade, apresentados na Figura 18.

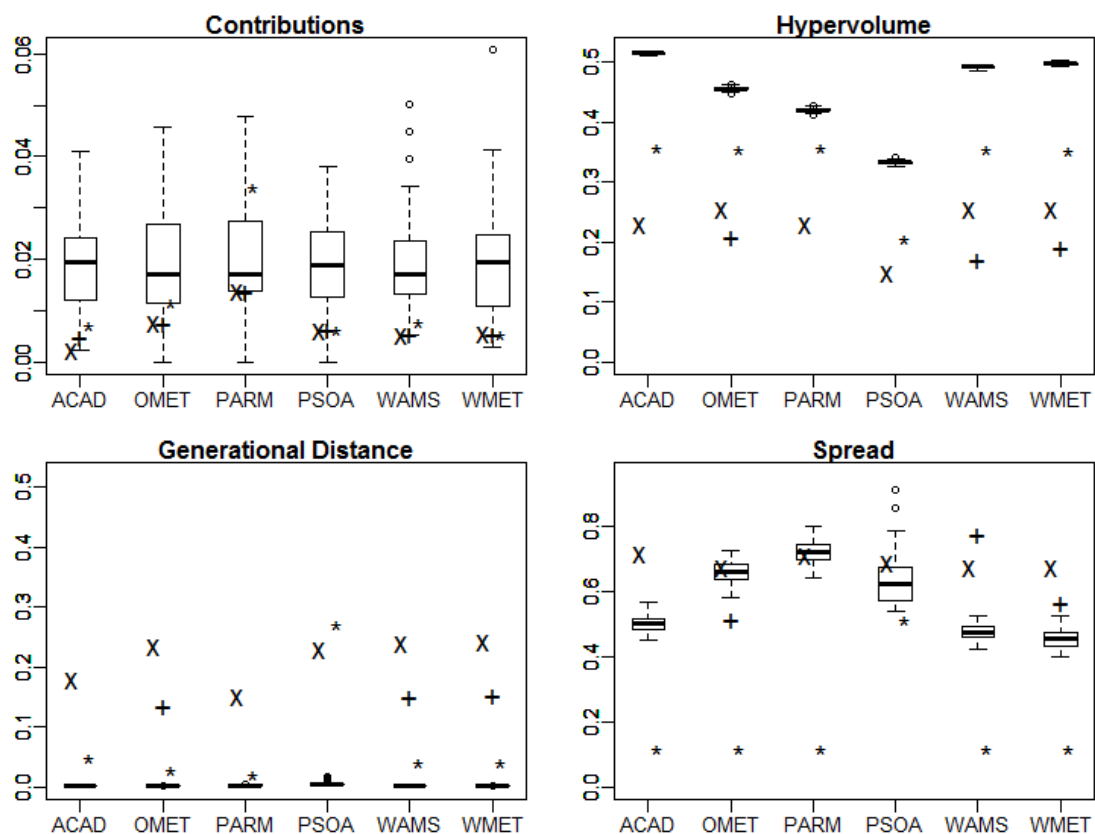


Figura 18 - Diagrama com todos os resultados de todas as configurações, instâncias e indicadores de qualidade. NSGA-II por *boxplot*, Margarine por X, Second Half por * e CPM por +.

Conforme foi apresentado na Tabela 13, na Tabela 14 e na Figura 18, identificamos indícios que o NSGA-II gera melhores resultados que as estratégias utilizadas na indústria, ficando a estratégia “*Second Half*” em segundo lugar. Desta forma, podemos rejeitar a hipótese nula e assumir a hipótese alternativa com a apresentação de melhores resultados para a solução proposta com o algoritmo NSGA-II.

4.3.3.3. Resposta ao Teste de Consistência (QP3)

Para responder a QP3, batizamos de NSGA-II_{SE} a utilização do algoritmo genético (NSGA-II) sem o cenário proposto na subseção 3.2.3, ou seja, não considerando a geração adicional de erros durante a execução das atividades sob o regime de horas extras. Desta forma, as soluções serão avaliadas pelo simulador, resultando somente as soluções não-dominadas em uma frente de Pareto.

Em seguida, comparamos os resultados do NSGA-II considerando a dinâmica de geração e propagação de erros devido ao trabalho em horas extras e os resultados do NSGA-II_{SE}. A Tabela 15 apresenta as médias e desvios padrão para todos os indicadores

de qualidade apresentados na subseção 4.3.1. Os valores de I_C e I_{GD} foram multiplicados por 100 para facilitar a apresentação e comparação.

Tabela 15 - Médias e desvios padrão comparando o NSGA-II e o NSGA-II_{SE} para todas as instâncias e indicadores de qualidade, marcando em negrito os melhores valores.

INSTÂNCIA	CONFIG	I_C	I_{HV}	I_{GD}	I_{SP}
ACAD	NSGA-II	0,2636±0,1741	0,3075±0,0024	1,5962±0,3456	0,6309±0,0513
	NSGA-II _{SE}	1,7368±0,406	0,3327±0,0011	0,0306±0,0051	0,4593±0,038
OMET	NSGA-II	0,0006±0,0042	0,2729±0,0034	2,036±0,2711	0,7662±0,0382
	NSGA-II _{SE}	1,9994±0,2324	0,3528±0,0014	0,0106±0,0024	0,467±0,0235
PARM	NSGA-II	0,0006±0,0042	0,232±0,0047	2,813±0,2549	0,7887±0,0445
	NSGA-II _{SE}	1,9996±0,2092	0,3628±0,0018	0,0102±0,0014	0,4733±0,0172
PSOA	NSGA-II	0,0024±0,0082	0,3439±0,0043	0,8928±0,2048	0,659±0,0688
	NSGA-II _{SE}	1,998±0,2019	0,3992±0,0016	0,0108±0,0027	0,5349±0,0219
WAMS	NSGA-II	0,006±0,0297	0,308±0,0023	2,0528±0,3486	0,6474±0,04
	NSGA-II _{SE}	1,996±0,2978	0,3531±0,0008	0,0204±0,002	0,4597±0,0237
WMET	NSGA-II	0,0848±0,1002	0,3136±0,0025	2,3436±0,449	0,6323±0,0384
	NSGA-II _{SE}	1,9164±0,3881	0,3482±0,0007	0,0302±0,0038	0,4445±0,0291

Conforme pode ser observado na Tabela 15, o NSGA-II_{SE} é superior em todos os indicadores. Realizamos a comparação dos valores com o teste estatístico Wilcoxon-Mann-Whitney para encontrar a probabilidade dos dois conjuntos de dados apresentarem médias iguais com 95% de certeza (BARROS, 2012; DE SOUZA *et al.*, 2011). Em complemento, foi utilizada a medida de tamanho de efeito não-paramétrica \hat{A}_{12} de Vargha e Delaney (2000), conforme observado na Tabela 16.

Tabela 16 - Resultados do p -value (PV) do teste de Wilcoxon-Mann-Whitney e o tamanho de efeito (ES) \hat{A}_{12} de Vargha-Delaney comparando o NSGA-II e o NSGA-II_{SE} para todas as instâncias e indicadores de qualidade. O tamanho de efeito mostra as chances do NSGA-II obter um maior valor para o I_C e o I_{HV} e um menor valor para o I_{SP} e o I_{GD} , mostrando a superioridade do NSGA-II_{SE}.

INSTÂNCIA	I_C		I_{HV}		I_{GD}		I_{SP}	
	PV	ES	PV	ES	PV	ES	PV	ES
ACAD	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	40%
OMET	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	0%
PARM	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	0%
PSOA	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	20%
WAMS	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	0%
WMET	<0,01	0%	<0,01	0%	<0,01	0%	<0,01	0%

Os resultados do tamanho de efeito (ES) na Tabela 16 mostram que o NSGA-II_{SE} é superior ao NSGA-II, uma vez que o último apresenta 0% de tamanho de efeito para o I_{HV} , o I_{GD} e o I_C em todas as instâncias. Em relação ao I_{SP} , o comportamento foi muito parecido com os demais indicadores, porém superior a zero para as instâncias ACAD e

PSOA. Estes resultados são confirmados pelo teste de Wilcoxon-Mann-Whitney com *p-value* (PV) inferior a 0,01 para todas as instâncias, ratificando que os resultados dos algoritmos não apresentam medianas semelhantes.

Todos os resultados encontrados foram piores para o NSGA-II em relação ao NSGA-II_{SE} devido à quantidade, bastante superior, de soluções produzidas pelo NSGA-II_{SE}. Este número de soluções permite que o NSGA-II_{SE} cubra partes da frente de referência que não são cobertas pelo NSGA-II, diminuindo o I_{GD} e aumentando o I_C do NSGA-II_{SE}. O NSGA-II contribui com mais pontos que o NSGA-II_{SE} na parte da frente de referência que ele cobre, porém não é o suficiente para compensar a distância entre os pontos das extremidades de sua frente de Pareto e as partes da frente de referência definidas somente pelo NSGA-II_{SE}.

Tabela 17 - Comparativo do número de soluções geradas pelo NSGA-II e pelo NSGA-II_{SE} para todas as instâncias.

INSTÂNCIA	GRUPOS DE FUNCIONALIDADES	NSGA-II _{SE}	NSGA-II	NSGA-II _{SE} /NSGA-II
ACAD	10	1235	411	3,00
OMET	21	3016	269	11,21
PARM	27	2936	156	18,82
PSOA	18	3528	323	10,92
WAMS	15	1987	384	5,17
WMET	11	1243	365	3,41
MÉDIA				8,76

Conforme apresentado na Tabela 17, o NSGA-II_{SE} produz, na média, 8,76 vezes mais soluções que o NSGA-II, chegando a 18,82 vezes mais na maior instância (PARM). A razão entre a quantidade de soluções do NSGA-II e o NSGA-II_{SE} aumenta a medida que o número de grupos de funcionalidades aumenta e independe das demais características das instâncias utilizadas.

Além das análises estatísticas dos indicadores de qualidades apresentadas na Tabela 15 e na Tabela 16, é interessante considerar as diferenças observadas nos valores dos objetivos influenciados pela dinâmica do trabalho em horas extras e a consequente geração e propagação de erros. A Tabela 18 apresenta as diferenças médias em duração e custo calculadas para as soluções envolvendo as maiores alocações de horas extras diárias que, na média, variam de 2,37 até 3,51 horas, considerando todas as instâncias. A tabela mostra que o algoritmo que não utilizou a dinâmica de geração adicional de erros devido ao trabalho em horas extras apresentou duração e custo menores em até 9,21% e 5,86%,

respectivamente. Estas diferenças ilustram que o NSGA-II_{SE} subestima o custo e a duração do projeto quando um grande número de horas extras é alocado à equipe de desenvolvimento. Essa diferença deve-se a longas atividades de testes que são necessárias para identificar e corrigir os erros introduzidos no software pelos desenvolvedores cansados.

Tabela 18 - Comparativo da diferença média da duração e do custo para as soluções envolvendo excessiva alocação de horas extras para todas as instâncias, considerando os algoritmos NSGA-II e NSGA-II_{SE}.

INSTÂNCIA	HORAS EXTRAS		NSGA-II _{SE} < NSGA-II	
	Diária	Total	Duração	Custo
ACAD	2,53 - 2,99	337,6 - 378,8	5,13 %	3,51 %
OMET	2,72 - 3,30	1031,9 - 1138,0	8,99 %	5,86 %
PARM	2,37 - 2,84	812,9 - 902,9	7,27 %	5,50 %
PSOA	2,67 - 3,06	1296,1 - 1451,9	2,23 %	0,99 %
WAMS	2,88 - 3,51	644,9 - 713,9	9,21 %	5,24 %
WMET	2,88 - 3,49	379,7 - 420,5	8,56 %	5,83 %

Baseado nas análises acima, encontramos evidências de que a otimização do planejamento de alocação de horas extras para cronograma de projetos de desenvolvimento de software, sem levar em conta a dinâmica do trabalho em horas extras e o aumento na taxa de geração de erros produz resultados diferentes do que um processo de otimização que considere tal efeito. Desta forma, podemos rejeitar a hipótese nula e assumir a hipótese alternativa com a apresentação de resultados mais consistentes pelo algoritmo NSGA-II.

4.3.3.4. Consolidando Resultados

Visado facilitar a percepção dos resultados obtidos, a Figura 19 apresenta em gráficos de duas dimensões, os resultados das combinações dos três objetivos enunciados na seção 3.1.5 (a quantidade de horas extras, duração e custo). Cada gráfico contém a melhor frente de Pareto de todos os ciclos executados dos experimentos, considerando cinco configurações: as três estratégias identificadas por Ferrucci *et al.* (2013) (MAR, SH e CPM), a proposta de solução (NSGA-II) e a proposta de solução desconsiderando a dinâmica de geração erros adicionais devido ao trabalho sob o regime de horas extras (NSGA-II_{SE}). Estas configurações foram representadas da seguinte forma: MAR (×), SH (*) e CPM (+) com os símbolos na cor preta; NSGA-II (°) com o símbolo em cinza claro e NSGA-II_{SE} (°) com o símbolo em cinza escuro.

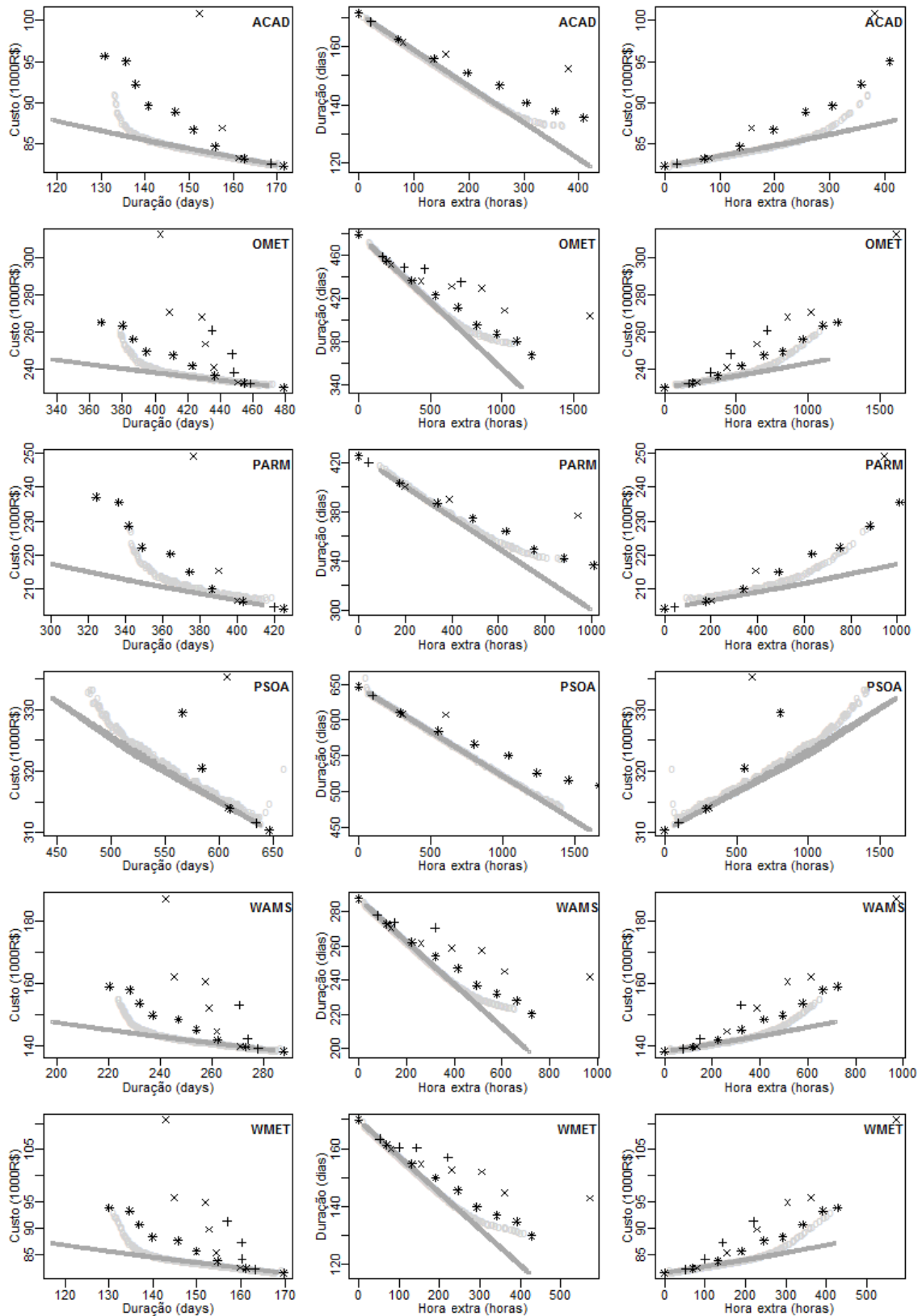


Figura 19 - Projeções 2D para as combinações da quantidade de horas extras, duração e custo para todas as instâncias. É possível observar as relações lineares resultantes do NSGA-II_{SE} e o formato convexo do NSGA-II, assim como as poucas soluções identificadas a partir das estratégias utilizadas pela indústria. MAR (×), SH (*) e CPM (+) são representados com os símbolos na cor preta, NSGA-II (°) com o símbolo em cinza claro e NSGA-II_{SE} (°) com o símbolo em cinza escuro.

O primeiro resultado que pode ser observado nos gráficos é o formato linear que a frente de Pareto que o NSGA-II_{SE} obteve, preenchendo a diagonal principal do hipercubo que representa o espaço dos objetivos. O formato desta frente de Pareto mostra a existência de relações lineares entre o número de horas extras utilizadas no projeto, sua duração e seu custo quando a dinâmica de geração de erros não é considerada. Este relacionamento linear é incompatível com os achados das pesquisas na Engenharia de Software, que prevêem um relacionamento exponencial entre o custo (esforço) e a duração para projetos de software não triviais (BOEHM, 1981). As relações lineares também fazem com que a otimização fique mais fácil para o algoritmo genético, resultando em uma grande quantidade de soluções não-dominadas que cobrem grande parte do espaço de busca não explorado por uma busca mais complexa, quando consideramos a dinâmica de geração de erros devido ao trabalho em horas extras.

Em relação à proposta de solução (NSGA-II), podemos observar a forma convexa obtida pelas frentes de Pareto. Estas frentes também preenchem a diagonal principal do hipercubo que representa o espaço dos objetivos. Estas curvas podem ser divididas em regiões que apresentam variação (aumento ou diminuição) rápida dos valores dos objetivos e regiões em que estas variações são mais lentas.

Como pode ser observado na Figura 19, o custo para redução da duração do cronograma do projeto aumenta de forma não linear. Para comparação deste custo, elencamos na Tabela 19, reduções de prazo de 10 dias em cada uma das instâncias em dois períodos, o primeiro começando no momento um pouco antes do ponto de convecção da curva apresentada na Figura 19 e o segundo terminando na menor duração encontrada no experimento.

Tabela 19 - Comparativo do custo de redução de duração de cronograma considerando dois períodos de 10 dias, chegando a duração mínima encontrada.

INSTÂNCIA	VARIAÇÃO TOTAL DO EXPERIMENTO		PERÍODOS DE REDUÇÃO DE 10 DIAS					
	Custo (%)	Duração (dias)	1°			2°		
			De	Para	Custo (%)	De	Para	Custo (%)
ACAD	10,42	37,2	153,2	143,2	1,50	143,2	133,2	6,87
OMET	12,20	94,4	398,7	388,7	1,97	388,7	378,7	6,90
PARM	10,16	76,2	362,3	352,3	2,05	352,3	342,3	5,64
PSOA	3,88	181,3	499,0	489,0	0,83	489,0	478,8	0,72
WAMS	11,36	62,7	243,6	233,6	1,85	233,6	223,6	6,93
WMET	13,15	37	151,0	141,0	1,75	141,0	131,0	10,02

Como pode ser observado na Tabela 19, o aumento de custo do projeto necessário para reduzir os últimos 10 dias, para alcançar a duração mínima, é de 10,02% (WMET). Enquanto que para a segunda redução de 10 dias, o acréscimo de custo passa a ser de 1,75% (WMET), ou seja, muito inferior a primeira redução. Isto mostra que os custos de redução entre os períodos podem chegar a 5,72 vezes menos que o custo da redução para a duração mínima. Este comportamento mostra a oportunidade que pode ser utilizada pelos gerentes na tomada de decisões sobre reduções de cronograma.

Finalmente, apesar dos gráficos apresentarem diversos formatos, o NSGA-II_{SE} e o NSGA-II apresentaram correlação inversa extremamente alta entre o número de horas extras e a duração do projeto (muito perto de -1,0 pelo coeficiente de *Spearman*), assim como entre a duração e o custo do projeto. De forma oposta, o número de horas extras e o custo do projeto apresentaram correlação direta extremamente alta (muito perto de 1,0 pelo coeficiente de *Spearman*), conforme esperado. Isto nos mostra que não precisamos considerar todos os três objetivos durante a otimização, mas apenas dois deles, uma vez que a duração mínima é sempre observada em soluções com o máximo de horas extras. Em termos práticos, descartando um objetivo reduzimos o esforço computacional consumido pela otimização, permitindo que mais gerações sejam analisadas no mesmo intervalo de tempo e que um maior número de soluções seja encontrado nas frentes de Pareto produzidas pelos algoritmos.

4.4. Ameaças à Validade

Wohlin *et al.* (2000) ressalta que no planejamento de um projeto experimental é importante se preocupar com a validade do experimento para que os resultados obtidos sejam válidos no contexto assumido. Por este motivo, avaliamos quatro tipos de ameaças a validade: a validade interna, a validade de construção, a validade de conclusão e a validade externa.

Considerando experimentos com algoritmos de busca no contexto da Engenharia de Software, Barros e Neto (2011) enumeram as ameaças que podem comprometer a validade do experimento. Neste trabalho, analisamos os tratamentos dados a cada ameaça, assim como os riscos que podem afetar a validade do experimento. Para isto, esta seção foi dividida em quatro subseções que listam e discutem estas ameaças no contexto dos experimentos deste capítulo.

4.4.1. Ameaças à Validade de Conclusão

Para ameaças à validade de conclusão, Barros e Neto (2011) enumeram quatro ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

- **Não consideração da variação aleatória.** A inicialização de um algoritmo heurístico geralmente é feita de forma aleatória. Por exemplo, a população inicial de um algoritmo genético pode ser gerada aleatoriamente. Por isto, os resultados são diferentes em cada execução, podendo uma execução isolada ser muito positiva ou muito negativa. Visando a mitigação deste aspecto, executamos o algoritmo diversas vezes de modo a eliminar esta ameaça. Como descrito na subseção 4.3.2.1, definimos o número de ciclos que os algoritmos fossem executados em 50 vezes;
- **Falta de boa estatística descritiva.** A execução repetida dos algoritmos produz uma amostra de dados relativamente grande. Portanto, estatísticas descritivas e análises gráficas devem ser recursos explorados na análise do experimento. A análise dos resultados do experimento incluiu a utilização de tabelas e/ou gráficos (de linha e *boxplot*) que apresentavam médias, mínimos, máximos e desvios padrão, abrangendo assim um amplo espectro de formas de sumarizar os dados;
- **Falta de formalização de hipóteses e testes estatísticos.** Os estudos baseados em comparação devem ser baseados em hipóteses formais, que devem ser avaliadas por testes estatísticos apropriados. Testes estatísticos devem ser conduzidos através de procedimentos de inferência estatística paramétricas ou não-paramétricas, de acordo com a distribuição observada nos dados. O projeto do experimento foi planejado para testar as hipóteses apresentadas nas questões de pesquisa apresentadas na Seção 4.1. Os valores observados para as medidas de qualidade utilizadas na comparação das soluções não apresentaram distribuição normal, conforme descrito na subseção 4.3.3. Portanto, utilizamos testes não-paramétricos para a avaliação das hipóteses;
- **Falta de base comparativa significativa.** Estudos que envolvem comparação de algoritmos usualmente analisam os resultados da execução destes algoritmos para as mesmas instâncias. Neste caso, alguns trabalhos utilizam de busca aleatória para mostrar que um determinado problema não pode ser resolvido encontrando soluções randômicas. No entanto, para tirar conclusões confiáveis baseadas em comparações, a base para esta comparação deve ser representativa de uma solução de qualidade. No experimento, utilizamos o algoritmo genético NSGA-II, que é amplamente utilizado

no contexto de otimização na área da Engenharia de Software, inclusive para o problema de planejamento de horas extras (FERRUCCI *et al.*, 2013). Este algoritmo foi comparado com a busca aleatória, mas também com estratégias utilizadas na indústria e um algoritmo similar com diferenças no modelo submetido ao simulador. Portanto, acreditamos que a base para as diversas comparações proposta foi justa e compatível com as questões de pesquisa.

4.4.2. Ameaças à Validade Interna

Para ameaças à validade interna, Barros e Neto (2011) enumeram quatro ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

- **Parametrização não sistemática dos algoritmos utilizados.** Os valores assumidos pelos parâmetros dos algoritmos utilizados em um estudo experimental não são explicitamente apresentados no projeto do estudo. Ao esconder os valores dos parâmetros, pode-se favorecer uma ou outra técnica, limitando a capacidade de generalizar as conclusões observadas. Além disso, uma descrição completa de valores dos parâmetros é necessária para o experimento ser passível de reprodução. A avaliação da solução proposta apresentou os experimentos que foram necessários para calibrar os parâmetros do algoritmo proposto, conforme as subseções 4.3.2.1 e 4.3.2.2. Os demais parâmetros foram devidamente apresentados na subseção 3.2.4;
- **Falta de discussão sobre a implementação dos algoritmos.** O código-fonte usado em um experimento pode esconder ajustes ou instrumentações específicas para favorecer certos casos ou algoritmos, influenciando assim os resultados observados. O código-fonte deve ser tornado público, permitindo que outros pesquisadores reproduzam e fiscalizem o experimento. O experimento utilizou o *JMetal*, que disponibiliza seu código-fonte para o público, sendo que devido às necessidades de experimentos anteriores, o grupo de pesquisa em Engenharia de Software baseada em Busca da UNIRIO realizou customizações do código-fonte do *JMetal*¹² voltado para análise dos resultados dos experimentos. O código-fonte da implementação¹³ para a proposta de solução está disponível na Internet;

¹² https://github.com/unirio/JMetal_UNIRIO

¹³ <https://github.com/unirio/Hector>

- **Falta de instâncias de problemas reais.** A Engenharia de Software é uma ciência prática e, como tal, deve lidar com problemas reais. Embora casos gerados aleatoriamente possam ser úteis para tratar o comportamento de uma nova técnica, em certas situações estes casos podem não ter as propriedades encontradas em casos reais. Com isto, podem influenciar as conclusões tiradas a partir do experimento. O experimento executado para avaliação da proposta de solução considerou instâncias reais de projetos de software aos quais o grupo de pesquisa teve acesso;
- **Falta de clareza na descrição dos procedimentos e ferramentas de coleta de dados.** Os procedimentos e ferramentas utilizados para coleta de informações de instâncias do mundo real ou aleatórias devem ser descritos com precisão para se certificar de que aspectos descritos não influenciem os resultados do estudo, beneficiando casos favoráveis ao pesquisador. Na Seção 4.2, descrevemos as características das instâncias utilizadas nos estudos e todos os tratamentos aplicados sobre elas.

4.4.3. Ameaças à Validade de Construção

Para o cenário de validade de construção, Barros e Neto (2011) enumeram três ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

- **Falta de medidas para avaliar custo de execução.** Ao medir o custo de execução de um algoritmo, o pesquisador deve justificar a métrica selecionada. Segundo Barros e Neto (2011), o número de avaliações da função de avaliação é a medida mais aceita e usada para medir custo de execução de algoritmos. O experimento utilizou a mesma métrica para controlar o tempo de execução dos algoritmos: o número máximo de avaliações das funções de avaliação, previamente determinado;
- **Falta de métrica que avaliem com qualidade dos resultados.** As medidas escolhidas devem ser relevantes para o problema: uma melhora observada em seus valores deve estar relacionada com a melhoria da qualidade da solução. Portanto, a seleção de medidas adequadas é relevante para se certificar de que os resultados observados aceitam ou rejeitam a teoria proposta. Para avaliação da qualidade das soluções, utilizamos no experimento os quatro indicadores de qualidade apresentados na subseção 4.3.1, como as métricas que avaliam a qualidade das soluções. Estas métricas já foram utilizadas por trabalhos anteriores, como por exemplo o de Ferrucci, Harman e Sarro (2014);

- **Não discutir se o modelo proposto reflete o mundo real.** O ambiente no qual o software é desenvolvido é geralmente complexo, podendo envolver questões técnicas e sociais. Portanto, qualquer modelo que descreva um problema relacionado ao software é uma simplificação do mundo real. Ao formalizar esses modelos, é preciso discutir suas limitações e como as simplificações adotadas por estes modelos podem influenciar na aplicação prática dos resultados observados.

A solução proposta para o PPH é baseada em um conjunto de parâmetros que foram utilizados para descrever a dinâmica de projetos de desenvolvimento de software e a dinâmica do trabalho em horas extras. Estes parâmetros foram coletados da literatura (ABDEL-HAMID; MADNICK, 1991; JONES, 2000, 2007) e podem não representar a realidade para todas as empresas de desenvolvimento de software. Eles deveriam ser coletados de uma empresa específica para aumentar a precisão dos resultados. Empresas que contratam desenvolvedores mais experientes, utilizam métodos de inspeção formais ou outras estratégias de verificação e validação de software podem introduzir menos erros ou identificar e corrigi-los em fases anteriores aos testes, tornando o projeto mais resiliente aos efeitos negativos do trabalho em horas extras.

As premissas e simplificações adotadas no modelo também podem fazer com que os resultados sejam afetados, sendo necessária a explicitação destas características de forma que possibilite o correto entendimento da abrangência do modelo. O modelo proposto não considerou a capacitação e as habilidades dos desenvolvedores, assumindo a mesma produtividade para os desenvolvedores e para todas as atividades. Isto permitiu investigar o relacionamento entre os lados positivo e negativo do trabalho em horas extras sem tratar um grande número de parâmetros, que seriam necessários para capturar as diferenças entre os desenvolvedores. Entretanto, desenvolvedores experientes podem introduzir menos erros e ser mais resistentes a longas jornadas de trabalho, possivelmente afetando nossos resultados. A ausência de sólidas informações sobre isto na literatura impediu a utilização destas características no modelo proposto.

Em relação à quantidade de desenvolvedores, o modelo proposto considerou somente um desenvolvedor para cada atividade, desconsiderando a dinâmica de trabalho em equipe e efeitos como a curva de aprendizado, o tempo gasto em reuniões e com comunicação em geral e a Lei de Brooks. A dinâmica de trabalho em equipe, normalmente, contribui para a redução individual da produtividade na medida que os desenvolvedores investem parte do seu tempo em outras atividades para o grupo.

Estamos assumindo o máximo de produtividade para mostrar que o lado negativo das horas extras pode agir contra a intuição. Se considerássemos os efeitos do trabalho em equipe, os ganhos de produtividade tenderiam a ser menores e o custo para remover os erros extras seria essencialmente o mesmo.

Finalmente, assumimos um custo linear para a identificação e correção de erros. Por outro lado, pode-se considerar que o custo depende da quantidade de erros no software: encontrar um erro em um software com uma pequena quantidade de erros latentes é mais difícil do que em um sistema repleto de erros. Ao utilizar médias e lidar com grande número de erros, acreditamos que o custo para encontrar erros mais fáceis ou mais difíceis é suavizado. Também assumimos que todos os erros são corrigidos antes da implantação e que os testes não introduzirão novos erros. Quebrar a primeira premissa poderia reduzir o esforço total para os testes e compensar o esforço para correção do grande número de erros decorrentes das horas extras. Quebrar a segunda premissa, por outro lado, aumentaria o número de erros ainda mais, reforçando os resultados encontrados pela execução do estudo.

4.4.4. Ameaças à Validade Externa

Para o cenário de validade externa, Barros e Neto (2011) enumeram três ameaças que devem ser consideradas no planejamento do experimento e análise dos resultados.

- **Falta de definição clara das instâncias.** Nenhuma generalização é possível se os pesquisadores não conhecem as instâncias utilizadas em um estudo experimental. Portanto, qualquer projeto experimental deve fornecer uma definição clara das instâncias selecionadas. Conforme apresentado na Seção 4.2, as instâncias foram detalhadas de forma que informações como origem, áreas de utilização, tamanho, idade e objetivo foram apresentadas sobre todas as instâncias utilizadas no experimento;
- **Falta de uma estratégia objetiva de seleção de objetos.** O experimento deve mostrar claramente como as instâncias utilizadas no experimento foram selecionadas, projetadas, geradas aleatoriamente ou capturadas de problemas do mundo real. O pesquisador deve justificar como estas instâncias são usadas no experimento. Na Seção 4.2, discutimos a razão pela qual as instâncias utilizadas no experimento foram selecionadas;

- **Falta de diversidade de tamanho e complexidade nas instâncias.** Técnicas de Engenharia de Software são projetadas para lidar com sistemas e equipes que podem variar em tamanho e complexidade. Portanto, um experimento deve avaliar uma técnica em uma amplitude de instâncias de problemas, variando em tamanho e complexidade, para fornecer uma avaliação sobre os limites da nova proposta. Conforme apresentado na Seção 4.2, as instâncias utilizadas no experimento possuem características bem distintas, principalmente em relação ao tamanho funcional e a área de utilização. O número de transações, por exemplo, varia entre 39 a 129 funções de transação, a quantidade de pontos de função, por exemplo, varia entre 185 a 635, e a área de atuação, por exemplo, varia de sistemas acadêmicos a sistemas de meteorologia.

4.5. Considerações Finais

O experimento tinha o objetivo de avaliar a solução proposta para o planejamento da alocação de horas extras considerando a dinâmica de um projeto de desenvolvimento de software e a dinâmica do trabalho em horas extras. Esta proposta considera a utilização de algoritmo genético (NSGA-II) para a busca heurística e um simulador para emular o andamento do projeto. Para isto, primeiro avaliamos, por meio de indicadores de qualidade e testes de inferência estatística, que a proposta é minimamente boa ao comparar com um algoritmo aleatório, obtendo êxito para a solução proposta. Na segunda parte do experimento, avaliamos a competitividade em relação às três estratégias de alocação de horas extras utilizadas pela indústria de desenvolvimento de software, conforme identificado por Ferrucci *et al.* (2013). Esta avaliação mostrou superioridade para a solução proposta em todas as comparações. Entretanto, para a estratégia SH, a qualidade das soluções apresentou proximidade. Na terceira parte do experimento, avaliamos a consistência de um planejamento que não considerasse dinâmica de geração erros devido a dinâmica do trabalho em horas extras. Esta avaliação mostrou que o/a gerente pode ter um planejamento inconsistente, o que pode fazer com que ele/ela enfrente imprevistos durante o projeto, no tocante aos objetivos do projeto, ou seja, a duração e o custo do projeto.

Os relacionamentos entre os objetivos apresentados pela análise das frentes de Pareto podem ser explorados pelos gerentes como alternativas de alocação de horas extras. Desta forma, as grandes mudanças em um dos objetivos podem ser tão vantajosas

que pequenas mudanças nos demais objetivos, uma vez que estas pequenas mudanças podem ser admissíveis. Desta forma, isto permite que os responsáveis pelo projeto possam responder perguntas como: “O que preciso fazer para reduzir a duração do projeto?”, “Quanto custa a redução da duração do projeto em X dias?” ou “Quais atividades proporcionam a melhor vantagem sob uma certa quantidade de horas extras?”.

Podem perguntar se precisamos de uma busca heurística para responder estas perguntas, ou seja, não conseguimos encontrar estas respostas confiando somente em simulação? Nós consideramos que isto não é possível porque estamos interessados nos relacionamentos entre os diferentes resultados apresentados pela simulação. Se nós estivéssemos interessados nas relações entre parâmetros e resultados, poderíamos coletar amostras dos parâmetros selecionados e calcular a distribuição dos resultados. Entretanto, para calcular a distribuição conjunta de dois ou mais resultados, nós precisamos coletar amostras de todos os parâmetros, que equivalerem a uma completa busca (todas as combinações de valores possíveis são associadas aos parâmetros) ou uma busca aleatória (uma quantidade limitada de combinações é aleatoriamente selecionada e associada aos parâmetros). A busca completa é inviável para problemas complexos, devido ao tamanho do espaço de busca ($p \cdot m$, onde “p” significa a quantidade de possibilidades de representação das horas extras, ou seja, 9 e “m” significa a quantidade de atividades do projeto). Por outro lado, os resultados de uma busca aleatória foram superados pela solução proposta, conforme mostrado na subseção 4.3.3.1. Por estes motivos, utilizamos as buscas heurísticas, já que complementam a simulação por buscarem pelas combinações mais próximas das ótimas, dentro do espaço de busca.

Em relação aos resultados da heurística utilizada no experimento, podemos observar forte concentração de alocação de horas extras nas atividades de codificação e testes, conforme apresentado nas frentes de Pareto calculados para NSGA-II. Esta concentração é consistente com a dinâmica de geração de erros, uma vez que, os erros introduzidos nas fases de especificação ou projeto vão ser propagados para as atividades subsequentes, aumentando o custo da atividade de testes no final do cronograma.

Capítulo 5 - Conclusões

Neste capítulo apresentamos as considerações finais com base nos resultados obtidos e nos objetivos firmados na Seção 1.1.

Conforme mencionado no Capítulo 2, os profissionais da área de desenvolvimento de software são frequentemente submetidos a escalas de trabalho superiores às 8 horas diárias. Por este motivo, um planejamento da alocação de horas extras para os profissionais do projeto facilitará a organização da vida profissional e da vida pessoal, fazendo com que a qualidade do produto não seja prejudicada pelo cansaço dos desenvolvedores. Para facilitar o entendimento dos efeitos das horas extras sobre os projetos, encontramos estudos que envolvem a criação de modelos para o trabalho executado em horas extras em projetos de software. Aliado à modelagem, encontramos trabalhos relacionados que utilizam a Engenharia de Software baseada em Buscas (SBSE) com o objetivo de realizar a otimização dos resultados.

Após a análise dos estudos já realizados, apresentamos uma proposta de solução para o problema do planejamento de alocação de horas extras (PPH) para projetos de desenvolvimento de software. Esta proposta utiliza otimização multi-objetivo (três objetivos) e considera tanto os efeitos positivos e negativos do trabalho em horas extras nos projetos de software. Para emular o comportamento dos projetos, utilizamos um mecanismo de simulação contínua, visando refletir sobre o projeto de software os diferentes efeitos das alocações de horas extras.

Para avaliar a solução proposta, utilizamos indicadores de qualidade, testes de inferência estatística e medidas de tamanho de efeito. A solução proposta foi comparada com estratégias de alocação de horas extras utilizadas pela indústria, conforme identificado por Ferrucci *et al.* (2013). Nesta avaliação, observamos que a proposta obteve resultados melhores na maioria dos indicadores e instâncias, mantendo em segundo lugar a estratégia *Second Half*, que determina a alocação de horas extras na segunda parte do cronograma dos projetos. Os resultados evidenciaram também, que o “feeling” dos

gerentes de projetos que alocam horas extras na segunda metade do cronograma se confirmaram como uma boa estratégia. Também comparamos a solução proposta com uma formulação similar que não considera os efeitos negativos de geração de erros devido ao trabalho em horas extras. Nesta avaliação, observamos que caso o tomador de decisão faça o planejamento sem considerar os efeitos negativos das horas extras, a duração e o custo do projeto poderão sofrer acréscimos não esperados. No experimento, estes acréscimos chegaram a 9,21% para a duração e 5,86% para o custo do projeto.

5.1. Contribuições

Podemos destacar como contribuições deste trabalho:

- Apresentar uma nova formulação multi-objetivo para o problema de planejamento de horas extras, que considera o aumento nas taxas de geração de erros devido ao cansaço do desenvolvedor por estar trabalhando em horas extras. Nossa formulação realiza o balanceamento entre o número de horas extras trabalhadas em um projeto, a duração e o custo;
- Projetar e executar um estudo experimental baseado em 6 projetos reais com até 635 pontos de função de tamanho. Os resultados mostraram que a concentração de horas extras na segunda parte de cronogramas dos projetos (uma prática utilizada pela indústria) é uma boa estratégia de alocação de horas extras;
- Coletar evidências que mostram que o impacto negativo sobre a qualidade do produto causado pelo trabalho em horas extras precisa ser considerado ao definir a estratégia de alocação de horas extras para um projeto de software. Uma estratégia que pode parecer ótima se esse impacto for desconsiderado, pode não ser a melhor escolha se ele for levado em conta.

5.2. Limitações e Perspectivas de Pesquisas Futuras

Para trabalhos futuros, pretendemos criar novos cenários para a modelagem de outros aspectos de projetos de desenvolvimento de software, tais como a exaustão do desenvolvedor devido a excesso de horas extras (e, conseqüente, propensão a não trabalhar mais em horas extras), a curva de aprendizagem, o gasto de tempo com comunicação devido ao tamanho da equipe e as inspeções. Também pretendemos integrar

a abordagem com uma ferramenta de gerenciamento de projetos, permitindo uma análise experimental com o envolvimento de seres humanos, para podermos observar a reação dos gerentes de projeto nas situações de atribuição de horas extras sugeridas pela ferramenta. Acreditamos também, que a criação de meios para ajudar na visualização das diferenças entre as alocações de horas extras propostas pelo otimizador é um aspecto importante para a sua adoção na prática. Pretendemos também enriquecer o estudo com a avaliação de outros algoritmos multi-objetivo, uma vez que somente um algoritmo multi-objetivo (NSGA-II) foi utilizado.

Referências Bibliográficas

ABDEL-HAMID, T. K.; MADNICK, S. E. **Software Project Dynamics: an integrated approach**. [s.l.] Prentice-Hall, USA , 1991.

AGUILAR-RUIZ, J. S. *et al.* An evolutionary approach to estimating software development projects. **Information and Software Technology**, v. 43, n. 14, p. 875–882, 2001.

AKULA, B.; CUSICK, J. **Impact of Overtime and Stress on Software Quality**Proceedings of the 4th International Symposium on Management, Engineering, and Informatics. **Anais...**2008

ANTONIOL, G.; PENTA, M. DI; HARMAN, M. **A Robust Search-Based Approach to Project Management in the Presence of Abandonment, Rework, Error and Uncertainty**.IEEE METRICS. **Anais...**IEEE Computer Society, 2005

ARNEDT, J. T. *et al.* Neurobehavioral Performance of Residents After Heavy Night Call vs After Alcohol Ingestion. **Jama-journal of The American Medical Association**, v. 294 , p. 1025–1033, 2005.

ARTIGUES, C.; DEMASSEY, S.; NERON, E. **Resource-constrained project scheduling**. [s.l.] Wiley Online Library, 2008.

BAKER, P. *et al.* **Search Based Approaches to Component Selection and Prioritization for the Next Release Problem**Software Maintenance, 2006. ICSM '06. 22nd IEEE International Conference on. **Anais...**2006

BANKS, J. *et al.* **Discrete-Event System Simulation (3rd Edition)**. [s.l.] Prentice Hall, 2000.

BARROS, M. D. O.; NETO, A. C. D. Threats to Validity in Search-based Software Engineering Empirical Studies. **Relatórios Técnicos do DIA/UNIRIO**, n. 6, p. 12, 2011.

BARROS, M. DE O. **Gerenciamento de Projetos Baseado em Cenários: Uma Abordagem de Modelagem Dinâmica e Simulação**. [s.l.] Universidade Federal do Rio de Janeiro, 2001.

BARROS, M. DE O. **An Analysis of the Effects of Composite Objectives in Multiobjective Software Module Clustering**Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation. **Anais...**New York, NY, USA: 2012

BARROS, M. O.; WERNER, C. M. L.; TRAVASSOS, G. H. A system dynamics metamodel for software process modeling. **Software Process: Improvement and Practice**, v. 7, n. 3-4, p. 161–172, 2002.

- BOEHM, B. W. **Software Engineering Economics**. [s.l.] Prentice Hall , 1981.
- BRASIL. **Decreto-Lei N.º 5.452, de 1º de maio de 1943**. Disponível em: <http://www.planalto.gov.br/ccivil_03/decreto-lei/del5452.htm>. Acesso em: 11 dez. 2014.
- BÜHLER, O.; WEGENER, J. Evolutionary Functional Testing. **Comput. Oper. Res.**, v. 35, n. 10, p. 3144–3160, 2008.
- CAO, L.; RAMESH, B.; ABDEL-HAMID, T. K. Modeling dynamics in agile software development. **ACM Trans. Management Inf. Syst.**, v. 1, n. 1, p. 5, 2010.
- CARUSO, C. C. Possible Broad Impacts of Long Work Hours. **Industrial Health**, v. 44, n. 4, p. 531–536, 2006.
- CHANG, C.; CHRISTENSEN, M.; ZHANG, T. Genetic Algorithms for Project Management. **Annals of Software Engineering**, v. 11, n. 1, p. 107–139, 2001.
- CHICANO, J. F. *et al.* **Using multi-objective metaheuristics to solve the software project scheduling problem**. (N. Krasnogor, P. L. Lanzi, Eds.)GECCO. **Anais...ACM**, 2011
- DE SOUZA, J. T. *et al.* **An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements**Proceedings of the Third International Conference on Search Based Software Engineering. **Anais...Berlin, Heidelberg: Springer-Verlag**, 2011
- DEB, K. *et al.* A fast and elitist multiobjective genetic algorithm: NSGA-II. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.
- DEL ROSSO, C. **Reducing Internal Fragmentation in Segregated Free Lists Using Genetic Algorithms**Proceedings of the 2006 International Workshop on Workshop on Interdisciplinary Software Engineering Research. **Anais...: WISER '06**.New York, NY, USA: ACM, 2006
- DEMARCO, T. **Controlling software projects : management, measurement & estimation**. New York, NY: Yourdon Press, 1982.
- DEMARCO, T.; LISTER, T. **Peopleware-productive projects and teams**. [s.l.] Dorset House Publishing co. , 1999.
- DI PENTA, M.; HARMAN, M.; ANTONIOL, G. The use of search-based optimization techniques to schedule and staff software projects: an approach and an empirical study. **Software: Practice and Experience**, v. 41, n. 5, p. 495–519, 2011.
- DONALD, I. *et al.* Work Environments, Stress, and Productivity: An Examination Using ASSET. **International Journal of Stress Management**, v. 12, n. 4, p. 409–423, 2005.

DURILLO, J. J. *et al.* **A Study of the Multi-objective Next Release Problem** Proceedings of the 2009 1st International Symposium on Search Based Software Engineering. **Anais...** Washington, DC, USA: IEEE Computer Society, 2009

FAZAR, W. **Program evaluation and review technique**. [s.l.] The American Statistician, 1959.

FEATHER, M. S.; KIPER, J. D.; KALAFAT, S. Combining Heuristic Search, Visualization and Data Mining for Exploration of System Design Spaces. **INCOSE International Symposium**, v. 14, n. 1, p. 824–836, 2004.

FERRUCCI, F. *et al.* **Not Going to Take This Anymore: Multi-objective Overtime Planning for Software Engineering Projects** Proceedings of the 2013 International Conference on Software Engineering. **Anais...** NJ, USA: IEEE Press, 2013

FERRUCCI, F.; HARMAN, M.; SARRO, F. Search-Based Software Project Management. In: **Software Project Management in a Changing World**. [s.l.] Springer , 2014. v. 19 p. 373–399.

FISCHER, F. M. *et al.* Implementation of 12-hour shifts in a Brazilian petrochemical plant: Impact on Sleep and Alertness. **Chronobiology International**, v. 17, n. 4, p. 521–537, jan. 2000.

FORRESTER, J. **Industrial Dynamics**. [s.l.] MIT Press , 1961.

FUJIGAKI, Y.; ASAKURA, T.; HARATANI, T. Work stress and depressive symptoms among Japanese Information Systems managers. **Industrial Health**, p. 231–238, 1994.

HARMAN, M.; CLARK, J. **Metrics are fitness functions too** Software Metrics, 2004. Proceedings. 10th International Symposium on. **Anais...** 2004

HARMAN, M.; JONES, B. F. Search-Based Software Engineering. **Information and Software Technology**, v. 43, p. 833–839, 2001.

HARMAN, M.; MANSOURI, S. A.; ZHANG, Y. Search-based Software Engineering: Trends, Techniques and Applications. **ACM Comput. Surv.**, v. 45, n. 1, p. 11:1–11:61, 2012.

HÖST, M. *et al.* Exploring bottlenecks in market-driven requirements management processes with discrete event simulation . **Journal of Systems and Software** , v. 59, n. 3, p. 323–332, 2001.

HOUDMONT, J.; ZHOU, J.; HASSARD, J. Overtime and psychological well-being among Chinese office workers. **Occupational Medicine**, v. 61, n. 4, p. 270–273, 2011.

HYMAN, J. *et al.* Work-Life Imbalance in Call Centres and Software Development. **British Journal of Industrial Relations**, v. 41, n. 2, p. 215–239, 2003.

JIA, J.; FAN, X.; LU, Y. System Dynamics Modeling for Overtime Management Strategy of Software Project. 2007.

JONES, C. **Software Assessments, Benchmarks, and Best Practices**. Boston, MA, USA: Addison-Wesley Longman Pub. Co., Inc., 2000.

JONES, C. **Estimating Software Costs: Bringing Realism to Estimating**. 2. ed. [s.l.] McGraw-Hill Osborne Media, 2007.

KANG, D.; JUNG, J.; BAE, D.-H. Constraint-based human resource allocation in software projects. **Software: Practice and Experience**, v. 41, n. 5, p. 551–577, 2011.

KARITA, K. *et al.* Effect of Overtime Work and Insufficient Sleep on Postural Sway in Information-Technology Workers. **Journal of occupational health**, v. 48, n. 1, p. 65–68, jan. 2006.

KUHN, P.; LOZANO, F. **The Expanding Workweek? Understanding Trends in Long Work Hours Among U.S. Men, 1979-2004**: Working Paper Series. [s.l.: s.n.]. Disponível em: <<http://www.nber.org/papers/w11895>>.

LAKHOTIA, K. *et al.* **FloPSy: Search-based Floating Point Constraint Solving for Symbolic Execution** Proceedings of the 22Nd IFIP WG 6.1 International Conference on Testing Software and Systems. **Anais...**: ICTSS'10. Berlin, Heidelberg: Springer-Verlag, 2010 Disponível em: <<http://dl.acm.org/citation.cfm?id=1928028.1928039>>

LIM, V. K. G.; TEO, T. S. H. Occupational Stress and IT Personnel in Singapore: Factorial Dimensions and Differential Effects. **Int. J. Inf. Manag.**, v. 19, n. 4, p. 277–291, ago. 1999.

LIU, Y.; TANAKA, H. Overtime work, insufficient sleep, and risk of non-fatal acute myocardial infarction in Japanese men. **Occupational and Environmental Medicine**, v. 59, n. 7, p. 447–451, jul. 2002.

MAHADIK, A. An Improved Ant Colony Optimization Algorithm for Software Project Planning and Scheduling. **International Journal of Advanced Engineering and Global Technology**, v. 2, n. 1, p. 6, 2014.

MEILONG, X.; LI, C.; CHEN, J. **System Dynamics Simulation to Support Decision Making in Software Development Project** Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on. **Anais...**2008

MINKU, L. L.; YAO, X. Software Effort Estimation As a Multiobjective Learning Problem. **ACM Trans. Softw. Eng. Methodol.**, v. 22, n. 4, p. 35:1–35:32, 2013.

MIT. **A Guide to Learning System Dynamics** . [s.l.: s.n.].

MITCHELL, R. J.; WILLIAMSON, A. M. Evaluation of an 8 hour versus a 12 hour shift roster on employees at a power station. **Applied Ergonomics**, v. 31, n. 1, p. 83–93, 2000.

NISHIKITANI, M. *et al.* Influence of Overtime Work, Sleep Duration, and Perceived Job Characteristics on the Physical and Mental Status of Software Engineers. **Industrial Health**, v. 43, n. 4, p. 623–629, 2005.

OGIŃSKA-BULIK, N. Occupational Stress and Its Consequences in Healthcare Professionals: The Role of Type D Personality. **International Journal of Occupational Medicine and Environmental Health**, v. 19, n. 2, p. 113–122, 2006.

PMI. **PMBOK - A Guide to the Project Management Body of Knowledge (PMBOK® Guide) — Fifth Edition** . [s.l.: s.n.].

PONNAM, V. S.; GEETHANJALI, D. N. Event Based Project Scheduling Using Optimized Ant Colony Algorithm. 2014.

PRATHIBHA, K. M.; RAVICHANDRAN, M.; JOHNSON, P. Comparison of occupational stress in teachers and software professionals: A questionnaire study. **Journal of Clinical and Biomedical Sciences**, v. 3, n. 4, p. 167–170, 2013.

RAHMAN, M. M. *et al.* **Evaluation of Optimized Staffing for Feature Development and Bug Fixing** Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. **Anais...: ESEM '10**. New York, NY, USA: ACM, 2010

REEVES, C. R. Genetic Algorithms for the Operations Researcher. **Journal on Computing**, v. 9, n. 3, p. 231–250, 1997.

ROBBINS, S. *et al.* **Organizational Behavior**. Sydney: Prentice Hall, 1998.

RUS, I.; COLLOFELLO, J.; LAKEY, P. Software process simulation for reliability management. **Journal of Systems and Software**, v. 46, n. 2-3, p. 173–182, 1999.

VARGHA, A.; DELANEY, H. D. A Critique and Improvement of the “CL” Common Language Effect Size Statistics of McGraw and Wong. **Journal of Educational and Behavioral Statistics**, v. 25, n. 2, p. 101–132, 2000.

WEGENER, J.; BARESEL, A.; STHAMER, H. Evolutionary test environment for automatic structural testing. **Information and Software Technology**, v. 43, n. 14, p. 841–854, 2001.

WINDISCH, A.; WAPPLER, S.; WEGENER, J. **Applying Particle Swarm Optimization to Software Testing** Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. **Anais...** New York, NY, USA: 2007

WOHLIN, C. *et al.* **Experimentation in Software Engineering: An Introduction**. Norwell, MA, USA: Kluwer Academic Publishers, 2000.

WOO, L. Y. *et al.* **Occupational Stress Among Computer Managers in Singapore**. [s.l.] National University of Singapore, 1989.

XIE, T. *et al.* **Fitness-Guided Path Exploration in Dynamic Symbolic Execution**. [s.l.] Microsoft Research, 2008.

YOO, S.; HARMAN, M.; UR, S. **Measuring and Improving Latency to Avoid Test Suite Wear Out**. ICST Workshops. **Anais...** IEEE Computer Society, 2009

YOO, S.; HARMAN, M.; UR, S. **Highly Scalable Multi Objective Test Suite Minimisation Using Graphics Cards**Proceedings of the Third International Conference on Search Based Software Engineering. **Anais...**Berlin, Heidelberg: Springer-Verlag, 2011

ZHANG, Y. *et al.* **Today/Future Importance Analysis**Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. **Anais...**Portland, Oregon, USA: 2010

ZHAO, Z.-Y.; LV, Q.-L.; YOU, W.-Y. **System Dynamic Model Applying on Analysis of Impact of Schedule Pressure on Project**Fourth International Conference on Fuzzy Systems and Knowledge Discovery. **Anais...**ago. 2007