



**Universidade Federal da Paraíba
Centro de Informática
Departamento de Informática
Programa de Pós-Graduação em Informática**

**Uma Abordagem para Alocação de Equipes em
Projetos de Desenvolvimento Distribuído de Software com base
em Atributos Não-Técnicos**

Bruno Luna Ribeiro

João Pessoa – PB

Abril de 2013



Universidade Federal da Paraíba
Centro de Informática
Departamento de Informática
Programa de Pós-Graduação em Informática

**Uma Abordagem para Alocação de Equipes em
Projetos de Desenvolvimento Distribuído de Software com base
em Atributos Não-Técnicos**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Glêdson Elias da Silveira

Orientando: Bruno Luna Ribeiro

João Pessoa – PB

Abril de 2013

RESUMO

Com a crescente adoção de projetos e processos de Desenvolvimento Distribuído de Software (DDS), muitas organizações que praticavam o desenvolvimento tradicional, co-localizado, passaram a atuar no mercado global, dividindo suas unidades colaborativas em vários pontos geográficos do globo. Além da aproximação com o mercado internacional, a utilização de DDS possibilita uma redução do *time-to-market*, melhor qualidade dos produtos gerados, redução de custos e maior facilidade para encontrar mão-de-obra qualificada. Entretanto, a distância entre as equipes envolvidas no DDS gera uma série de problemas relacionados a comunicação não enfrentados no desenvolvimento co-localizado. Segundo a NBR ISO 10006, o fator humano é chave para o sucesso dos projetos de software, já que é uma atividade genuinamente humana. No desenvolvimento tradicional, empresas utilizam processos de alocação de recursos humanos para encontrar os profissionais adequados para realizar uma determinada atividade. Porém, a distância geográfica, como também os fatores culturais e temporais existentes no DDS não são enquadrados nesses processos. Neste contexto, a fim de reduzir os problemas de comunicação durante a fase de implementação, o presente trabalho apresenta uma abordagem para alocação de equipes em projetos de DDS, tendo como objetivo mitigar os efeitos problemáticos gerados pela necessidade de comunicação entre as equipes geograficamente dispersas. Considerando que o acoplamento existente entre os componentes de software interfere na comunicação requerida para desenvolvê-los, a abordagem proposta leva em consideração características não-técnicas das equipes, como atributos temporais, culturais e de afinidade, para descobrir o grau de proximidade das equipes, comparando-as com o nível de acoplamento entre os componentes, de modo a identificar quais equipes possuem melhores condições de implementar os componentes de software com grau de comunicação satisfatório com as demais equipes. Deve-se destacar que as equipes candidatas à alocação com base em atributos não-técnicos são equipes tecnicamente habilitadas para a implementação dos componentes. Pela complexidade do problema de alocação ser alta, é utilizado um algoritmo genético, com função de avaliação baseada nos atributos não-técnicos para a alocação das equipes, tornando o problema tratável computacionalmente em tempo hábil.

ABSTRACT

Taking into account the increasing adoption of projects and processes of Global Software Development (GSD), many organizations that practiced traditional software development, co-located, began to operate in the global market, dividing their collaborative units in many points of the globe. Besides the proximity to the international market, the use of GSD allows a reduction in time-to-market, improved quality of generated products, cost reduction and greater ease to find skilled labor. However, the distance between the teams involved in the GSD creates a variety of communication problems not faced in co-located development. According to NBR ISO 10006, human factor is the key to the success of software projects, since it is a genuinely human activity. In traditional development, companies use resources allocation processes to find the appropriate professional to perform a particular activity. However, geographical distance, and also the cultural and temporal factors existing in GSD are not classified in these processes. In this context, in order to reduce communication problems during the implementation phase, this work presents an approach for allocating teams in GSD projects, aiming to mitigate the problematic effects generated by the need for communication between geographically dispersed teams. Considering that the coupling between software components interferes to the communication required to develop them, the proposed approach takes into account non-technical characteristics of the teams such as temporal, cultural and affinity attributes, to discover the degree of proximity of the teams, comparing them with the level of coupling between components in order to identify which teams have better able to implement the software components with suitable quality of communication with other teams. It should be noted that the teams that are candidates for allocation based on non-technical attributes are technically qualified teams for the implementation of components. Due to the high complexity of the allocation problem, it is used a genetic algorithm with fitness evaluation based on the non-technical attributes for the allocation of the teams, making the problem computationally treatable in time.

SUMÁRIO

RESUMO	III
ABSTRACT	IV
SUMÁRIO	V
LISTA DE FIGURAS	VII
LISTA DE TABELAS	VIII
LISTA DE EQUAÇÕES	IX
LISTA DE ABREVIATURAS E SIGLAS	X
1. INTRODUÇÃO	11
1.1 Motivação	14
1.2 Objetivo do trabalho	15
1.3 Organização da proposta	16
2. FUNDAMENTAÇÃO TEÓRICA	17
2.1 Engenharia de Software	17
2.2 Desenvolvimento Distribuído de Software	19
2.3 Engenharia de Software Baseada em Buscas	21
2.3.1 Algoritmos Genéticos	23
2.4 Considerações finais	26
3. TRABALHOS RELACIONADOS	27
3.1 Modelo para o desenvolvimento 24h	27
3.2 Modelo de simulação Global Software Development	28
3.3 DIMANAGER	30
3.4 TAMRI: Task Allocation based on Multi cRiteria	30
3.5 Comparativo da abordagem proposta com os trabalhos relacionados	31
4. FRAMEWORK DE RECOMENDAÇÕES	34
4.1 Recomendação de módulos de software	35
4.2 Recomendação técnica de equipes	36
4.3 Recomendação de alocação de equipes	38
4.4 Avaliação de desempenho das equipes	39
5. ABORDAGEM PARA RECOMENDAÇÃO DE ALOCAÇÃO DE EQUIPES	40
5.1 Descrição não-técnica das equipes	41
5.1.1 Atributos temporais	43
5.1.2 Atributos culturais	44

5.1.2.1	Questionário de idiomas	44
5.1.2.2	Questionário de perfil organizacional	46
5.1.3	Atributo de afinidade	54
5.2	Alocação de equipes	55
5.2.1	Definição de pesos.....	57
5.2.2	Análise não-técnica.....	58
5.2.2.1	Representação das Soluções e População Inicial	59
5.2.2.2	Função objetivo	59
5.2.2.3	Operadores genéticos	69
5.2.3	Seleção de recomendação.....	71
5.3	Considerações finais	72
6.	PROVA DE CONCEITO	73
6.1	Problemas reais e fictício	73
6.1.1	MultiMOM	74
6.1.2	GingaCC.....	74
6.1.3	SPL e-commerce	75
6.1.4	BigExample	76
6.2	Instanciação de equipes	77
6.3	Parâmetros e características do algoritmo genético.....	81
6.3.1	Mecanismo de seleção de cromossomos pais	82
6.3.2	Operadores genéticos	82
6.3.3	Pesos utilizados	83
6.4	Resultados.....	84
6.4.1	MultiMOM	85
6.4.2	Ginga CC.....	87
6.4.3	SPL e-commerce	90
6.5	Comparativo da abordagem proposta com a busca aleatória	92
6.5.1	Multi-MOM.....	93
6.5.2	SPL e-commerce	94
6.5.3	Ginga CC.....	96
6.5.4	BigExample	99
6.6	Comentários finais	102
7.	CONSIDERAÇÕES FINAIS.....	104
7.1	Contribuições.....	105
7.2	Limitações	106
7.3	Trabalhos futuros.....	107
	REFERÊNCIAS	108
	ASSINATURAS	127

LISTA DE FIGURAS

Figura 1 – Sequência de passos de um algoritmo genético	25
Figura 2 – Grafo Acíclico Direcionado para o modelo de desenvolvimento 24h	28
Figura 3 – <i>Framework</i> de recomendações.....	34
Figura 4 – Artefatos gerados durante as recomendações	35
Figura 5 – Module Based Matrix (MBM)	36
Figura 6 – Team x Module Mapping (TMM)	38
Figura 7 – Divisão de etapas da recomendação de alocação de equipes	41
Figura 8 – Atributos não-técnicos	42
Figura 9 – Exemplo para formulário de atributos temporais.....	44
Figura 10 – Perfil organizacional – Processo de software.....	48
Figura 11 – Perfil organizacional – Modelo de maturidade	49
Figura 12 – Perfil organizacional - CSCW.....	51
Figura 13 – Perfil organizacional – Perguntas diferenciais	52
Figura 14 – Exemplo de conversão de relacionamentos externos em dependência.....	57
Figura 15 – Team x Module Mapping (TMM)	57
Figura 16 – Representação e geração de um cromossomo para população inicial.....	59
Figura 17 – Exemplo de cromossomo com dependências entre módulos.....	62
Figura 18 – Exemplo de mutação	69
Figura 19 – Exemplo de cruzamento uniforme	70
Figura 20 – Condicionais do questionário de perfil organizacional.....	79
Figura 21 – Probabilidade de respostas do questionário	80

LISTA DE TABELAS

Tabela 1 – Comparativo entre os trabalhos relacionados e o <i>framework</i>	33
Tabela 2 – Semelhança entre respostas da Questão 3	66
Tabela 3 – Tabela de condicionais da Questão 3.....	66
Tabela 4 – Tabela de condicionais da Questão 8.....	67
Tabela 5 – MBM de relacionamentos do MultiMOM.....	74
Tabela 6 – MBM de dependências do MultiMOM	74
Tabela 7 – MBM de relacionamentos do GingaCC	75
Tabela 8 – MBM de dependências do GingaCC	75
Tabela 9 – MBM de relacionamentos da SPL e-commerce	75
Tabela 10 – MBM de dependências da SPL e-commerce.....	76
Tabela 11 – MBM de relacionamentos do BigExample.....	76
Tabela 12 – MBM de dependências do BigExample	77
Tabela 13 – Pesos utilizados na prova de conceito	84
Tabela 14 – Solução ótima para problema MultiMOM e seus atributos.....	85
Tabela 15 – Respostas do Perfil Organizacional para MultiMOM	86
Tabela 16 – Solução ótima para problema Ginga CC e seus atributos.....	88
Tabela 17 – Respostas do Perfil Organizacional para Ginga CC	89
Tabela 18 – Solução ótima para problema SPL e-commerce e seus atributos.....	90
Tabela 19 – Respostas do Perfil Organizacional para SPL e-commerce	91
Tabela 20 – Soluções da abordagem proposta e da busca aleatória para Multi-MOM.....	94
Tabela 21 – Soluções da abordagem proposta e da busca aleatória para SPL e-commerce.....	95
Tabela 22 – Soluções da abordagem proposta e da busca aleatória para Ginga CC	97
Tabela 23 – Soluções da abordagem proposta e da busca aleatória para BigExample	100
Tabela 24 – Soluções dos algoritmos com 1000 iterações para BigExample	101
Tabela 25 – Configurações da abordagem proposta para BigExample.....	102

LISTA DE EQUAÇÕES

Equação 1 - Avaliação de idioma	46
Equação 2 - Avaliação de perfil organizacional	53
Equação 3 - Avaliação de afinidade	55
Equação 4 - Avaliação de dependência	56
Equação 5 - Função objetivo	60
Equação 6 - Avaliação para um critério não-técnico.....	61
Equação 7 - Penalidade baseada em excesso de dependência interna.....	61
Equação 8 - Avaliação temporal.....	62
Equação 9 - Avaliação cultural.....	63
Equação 10 - Avaliação de idioma entre duas equipes	63
Equação 11 - Ponderação de múltiplos idiomas.....	64
Equação 12 - Avaliação de perfil organizacional entre duas equipes	65
Equação 13 - Avaliação do questionário	68
Equação 14 - Avaliação de afinidade entre duas equipes.....	68
Equação 15 - Afinidade inicial	68
Equação 16 - Função linear de cruzamento e mutação	71
Equação 17 - Complemento da função linear.....	83

LISTA DE ABREVIATURAS E SIGLAS

ADDS	Ambiente de Desenvolvimento Distribuído de Software
CSCW	<i>Computed-Supported Cooperative Work</i>
DBC	Desenvolvimento Baseado em Componentes
DDS	Desenvolvimento Distribuído de Software
DES	<i>Discrete-event</i>
DiSEN-PPM	<i>Distributed Software Environment – Project Management Model</i>
ED	Engenharia de Domínio
EA	Engenharia de Aplicação
ESBB	Engenharia de Software Baseada em Buscas
FOP	<i>Feature-Oriented Programming</i>
GAD	Grafo Acíclico Direcionado
GSD	<i>Global Software Development</i>
IE	<i>Interaction Effect</i>
LPS	Linhas de Produto de Software
MBM	<i>Module Based Matrix</i>
RH	Recursos Humanos
SD	<i>System Dynamics</i>
TMM	<i>Team x Module Mapping</i>

Capítulo 1

Introdução

Com a globalização, a indústria, em vários segmentos, busca novos mecanismos de obter vantagens de mercado. Com o desenvolvimento de software não é diferente. Atualmente, o Desenvolvimento Distribuído de Software (DDS) vem crescendo significativamente, fazendo com que o desenvolvimento tradicional de software, centralizado, passe a acontecer dividido em vários pontos geográficos do planeta. A utilização de DDS, além de possibilitar a aproximação com o mercado global, possibilita a redução do *time-to-market*, melhoria da qualidade do software e redução de custos (AUDY e PRIKLADNICKI, 2007) (HERBSLEB e GRINTER, 1999). Esses benefícios são desencadeados pela maior facilidade em se aproximar dos envolvidos (*stakeholders*) no projeto, uma vez que eles podem estar separados geograficamente uns dos outros e a multiplicação das equipes no mundo ajuda no processo; e de se encontrar mão de obra qualificada ou barata, já que cada país possui uma situação financeira diferente. Alguns deles podem ser financeiramente mais vantajosos que outros, como acontece na Índia (KOBITZSCH, ROMBACH e FELDMANN, 2001).

Entretanto, a distância existente entre todos os envolvidos também desencadeia problemas relacionados à comunicação, um dos principais desafios do DDS (SOUZA, COSTA, *et al.*, 2008). Esses problemas são decorrentes da falta de comunicação e/ou comunicação de baixa qualidade entre os envolvidos, podendo aumentar as falhas ou inconsistências existentes no software. Com isso, um aumento no tempo para o desenvolvimento do projeto acontece, e, em casos mais extremos, pode se chegar a conclusão de que o melhor caminho é o cancelamento do mesmo (AUDY e PRIKLADNICKI, 2007).

Para minimizar alguns problemas de DDS, o Desenvolvimento Baseado em Componentes (DBC) é bastante usado como abordagem de desenvolvimento com equipes distribuídas (SZYPERSKI, 2000), alocando os componentes de software entre as diferentes equipes e depois realizando a integração dos mesmos. Essa abordagem facilita o uso de desenvolvimento distribuído pela delimitação de trabalho existente entre os componentes de software. Em contrapartida, os componentes possuem dependências entre si, geralmente na

forma de interfaces providas e requeridas, o que torna difícil e complexo a alocação dos componentes entre as equipes distribuídas.

Além disso, como uma forma de entender a relação de uma arquitetura de software e a comunicação entre os envolvidos em sua implementação, vários estudos (GHEZZI, JAZAYERI e MANDRIOLI, 2002) (OLSON e OLSON, 2000) (SAKTHIVEL, 2005) (HERBSLEB e MOCKUS, 2003) mostram que a dependência existente entre os componentes arquiteturais tem reflexo no grau de comunicação necessário entre as equipes que irão implementá-los.

Além do DBC, as Linhas de Produto de Software (LPS) estão cada vez mais sendo utilizadas como uma forma de organizar a arquitetura de software, adicionando reuso de uma forma sistemática ao desenvolvimento de software (POHL, BOCKLE e VAN DER LINDEN, 2005). As abordagens de LPS utilizam o conceito de reuso na forma de dois processos, denominados: Engenharia de Domínio (ED) e Engenharia de Aplicação (EA). A ED é o processo onde um conjunto de artefatos reusáveis por todas as aplicações da linha são implementados, enquanto que a EA é o processo de instanciação de uma aplicação propriamente dita, usando como base os artefatos gerados na ED e a implementação de artefatos específicos da EA (VAN DER LINDEN, SCHMID e ROMMES, 2007). A LPS forma uma plataforma de artefatos que podem ser reutilizados ou modificados para a criação de aplicações que compartilham o mesmo domínio. Sendo as linhas de produto também baseadas em componentes de software, evidentemente, o reflexo entre o acoplamento dos componentes e a comunicação entre as equipes também existe.

Além do aspecto técnico, no desenvolvimento de software, um dos fatores mais importantes para o sucesso de um projeto são as pessoas que compõem a equipe de desenvolvimento (BARRETO, 2005). Essas pessoas são os recursos humanos do desenvolvimento de software, que segundo a norma NBR ISO 10006 (ABNT, 2000), é o fator chave para determinar a qualidade de um projeto. Essa necessidade de valorizar o recurso humano acontece pelo desenvolvimento de software ser uma atividade genuinamente humana (PRESSMAN, 2011), que depende da interação de pessoas com um mesmo propósito. Com isso, o gerenciamento adequado dos recursos humanos faz com que equipes trabalhem de forma mais eficiente, auxiliando na qualidade dos produtos gerados e na escolha certa de quem deve desempenhar um determinado papel, seja por qualidade técnica ou outro fator.

Como uma forma de melhor utilizar as pessoas disponíveis, processos de alocação de recursos humanos são utilizados para selecionar as melhores pessoas para desempenhar uma atividade, ou o conjunto de pessoas, que juntos formarão uma equipe com o nível desejado.

Com a maior adoção de DDS, os processos de alocação devem evoluir para levar em consideração outros fatores além dos fatores técnicos, como horário de trabalho, disponibilidade, país, idiomas, entre outros. Além disso, a grande maioria dos processos de alocação assume que as pessoas envolvidas fazem parte da mesma organização, o que nem sempre acontece, já que o DDS pode ser feito com equipes subsidiárias da própria organização (*insourcing*) ou equipes terceirizadas (*outsourcing*).

Com o aumento dos projetos de software no âmbito global, cada vez mais organizações tentam distribuir suas equipes pelo mundo. Com isso, os fatores de localização, tempo, distância e também a estrutura organizacional, a qual o DDS está inserido, interferem na forma como devem ser alocadas as equipes de desenvolvimento de software.

A maioria dos processos de alocação leva em consideração apenas fatores técnicos no momento da análise, porém, quando tratamos de DDS, os fatores não-técnicos podem ser responsáveis pelo sucesso ou fracasso de um projeto (AUDY e PRIKLADNICKI, 2007) (GUMM, 2006) (CARMEL e PRIKLADNICKI, 2010).

Para favorecer a alocação com o uso de DDS, um *framework* de recomendações para alocação de equipes distribuídas foi desenvolvido pelo laboratório COMPOSE (PEREIRA, SANTOS, *et al.*, 2010), como parte do projeto INES (Instituto Nacional de Ciência e Tecnologia para Engenharia de Software). O *framework* se baseia em uma arquitetura de LPS e sugere três fases de recomendações: (i) recomendação de módulos de software; (ii) recomendação técnica de equipes e (iii) recomendação de alocação de equipes.

A **recomendação de módulos de software** explora o acoplamento existente entre os componentes para avaliar se a necessidade de comunicação entre as equipes que vão implementá-los será alta. A recomendação tem como objetivo agrupar componentes de software muito acoplados, gerando módulos de software menos acoplados entre si do que os componentes de software, reduzindo a dependência externa aos módulos, o que diminui de forma significativa os requisitos de comunicação entre as equipes distribuídas responsáveis pela implementação destes módulos. Ao mesmo tempo, as dependências internas aos módulos tendem a aumentar com a adição de componentes aos módulos. Porém, o impacto das dependências internas não é refletida na comunicação entre as equipes distribuídas, mas sim

na comunicação interna entre os membros de uma mesma equipe, onde a comunicação é facilitada, ou, pelo menos, melhor gerenciada.

A **recomendação técnica de equipes** utiliza os conhecimentos técnicos e de domínio das equipes e os compara com o que é requisitado tecnicamente pelos módulos de software, gerando uma lista de equipes, por módulo, que são consideradas habilitadas para a sua implementação, fazendo com que apenas equipes tecnicamente hábeis sejam consideradas no momento da alocação dos módulos para a sua implementação.

A **recomendação de alocação de equipes** utiliza as dependências entre módulos e os compara com uma medida obtida com base nas características não-técnicas das equipes, analisando apenas as equipes tecnicamente habilitadas de forma a prever as melhores combinações de alocação que geram uma boa comunicação entre as equipes. Essa medida de características não-técnicas, denominada de relacionamento entre as equipes, é verificada com base nos atributos como horário de trabalho, fuso horário, idiomas, entre outros.

É importante destacar que esse trabalho tem como foco a instanciação da fase de **recomendação de alocação de equipes** do *framework* de recomendações.

1.1 Motivação

Como a dependência entre componentes de software gera impacto na comunicação entre os responsáveis por implementá-los (MOCKUS e WEISS, 2001), a alocação de equipes de software com o uso de DDS deve possuir equipes que possam se comunicar eficientemente, principalmente nos casos em que as dependências entre os módulos são mais altas.

Para tratar tal problema, é importante que as equipes sejam caracterizadas por atributos não-técnicos, que remetem a fatores que afetam a comunicação. Um dos maiores problemas relatados com o uso de DDS é a comunicação entre as unidades distribuídas (AUDY e PRIKLADNICKI, 2007), sendo assim, se faz necessário uma abordagem que utilize os aspectos não-técnicos como fator determinante para a alocação.

Para isso, uma série de atributos não-técnicos devem ser descritos, como fuso-horário, horário de trabalho, idiomas conhecidos, entre outros. A partir desses atributos é que será possível comparar pares de equipes e relacionar uma avaliação não-técnica entre as equipes e a dependência entre os módulos, fazendo com que seja possível ponderar a alocação das equipes com a dependência existente entre os módulos.

Além disso, a **recomendação de alocação de equipes** é muito complexa para ser realizada de forma *ad hoc* ou baseada apenas na experiência de um gerente de projetos. A quantidade de possibilidades geradas é o exponencial do número de equipes ao número de artefatos de software a serem implementados, fazendo com que os fatores existentes nas equipes e nos artefatos sejam impossíveis de serem analisados sem nenhum auxílio computacional. Adicionalmente, a alocação de equipes é considerada um problema NP-Completo, uma vez que, mesmo computacionalmente, é impossível calcular todas as possibilidades existentes em tempo hábil.

1.2 Objetivo do trabalho

Este trabalho tem como objetivo a criação de uma abordagem para alocação de equipes distribuídas para a implementação de módulos de software baseados em componentes, que é referente à **recomendação de alocação de equipes** do *framework* de recomendações.

A recomendação tem como objetivo auxiliar o gerente de projeto a alocar as equipes tecnicamente habilitadas para trabalhar de forma distribuída, sabendo que existirá necessidade de comunicação entre as equipes que forem alocadas a módulos dependentes.

Para se alcançar o objetivo geral, temos alguns objetivos específicos que fazem parte do escopo desse trabalho: i) definir os atributos não-técnicos que afetam a comunicação com o uso de DDS; ii) caracterizar as equipes a cerca dos atributos não-técnicos; iii) desenvolver um algoritmo baseado em meta-heurística que verifique os atributos não-técnicos das equipes tecnicamente habilitadas e as dependências entre os módulos de software, recomendando alocações de equipes que facilitem a comunicação entre as equipes distribuídas, com base em aspectos não-técnicos; e iv) realizar uma prova de conceito a cerca do algoritmo criado.

A definição dos atributos não-técnicos leva em consideração os atributos que não sejam os que avaliam tecnicamente uma equipe, mas que avaliem a viabilidade e a qualidade de comunicação que uma equipe pode ter se relacionada com outra. Dessa forma, é proposta uma análise que compara as equipes quanto a esses atributos, a fim de encontrar um conjunto de recomendações de alocação de equipes que auxiliem a decisão do gerente de projeto, verificando fatores que dificilmente seriam analisados sem uma ferramenta computacional.

Em relação à recomendação da alocação, se destaca a utilização de um algoritmo baseado em meta-heurística que reduza o universo de comparações possíveis, a fim de deixar o problema a ser resolvido tratável computacionalmente, uma vez que o universo de soluções possíveis pode ser calculado como o número de equipes elevado ao número de módulos.

Um ponto importante que também deve ser destacado é que os problemas de comunicação são enfrentados na maioria das empresas, entretanto, a forma mais comum é a utilização de ferramentas como mensageiros eletrônicos, ferramentas de vídeo-conferência ou até mesmo redes sociais internas. A atual proposta não diminui nem vai contra a utilização dessas ferramentas. Pelo contrário, é subentendido que algumas dessas ferramentas são utilizadas para manter a comunicação a distância. O objetivo desse trabalho é distribuir as atividades de implementação de software (módulos de software) de forma a facilitar a comunicação durante a fase de implementação, inclusive levando em consideração o uso das ferramentas mencionadas.

1.3 Organização da proposta

O trabalho está organizado da seguinte forma: o Capítulo 2 apresenta uma fundamentação teórica, introduzindo o leitor sobre assuntos como Engenharia de Software, Desenvolvimento Distribuído de Software e Engenharia de Software Baseada em Buscas. O Capítulo 3 apresenta os trabalhos relacionados e os compara entre si, apontando os pontos fortes, fracos e semelhanças. O Capítulo 4 apresenta o *framework* de recomendações ao qual esse projeto está inserido, detalhando as fases de **recomendação de módulos de software**, **recomendação técnica de equipes** e **recomendação de alocação de equipes**. O Capítulo 5 apresenta a abordagem proposta, mostrando a identificação dos atributos não-técnicos e caracterização de equipes, bem como a fundamentação das funções utilizadas para o algoritmo de alocação. O Capítulo 6 apresenta as provas de conceito realizadas com a abordagem, mostrando os resultados e analisando o que eles significam. Além disso, é realizado um comparativo entre o algoritmo utilizado e um algoritmo aleatório. E, finalmente, no Capítulo 7 são apresentadas as considerações finais, apresentando um comparativo entre os trabalhos relacionados e a atual abordagem, as contribuições, as limitações, e as possibilidades de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo tem o objetivo de fundamentar o leitor a cerca dos assuntos que são debatidos para o entendimento da atual abordagem. Nesse sentido, os temas de Engenharia de Software, Desenvolvimento Distribuído de Software e Engenharia de Software Baseada em Buscas são explicados de forma a introduzir conceitos básicos sobre os mesmos. Além disso, a seção de Engenharia de Software Baseada em Buscas possui uma subseção que detalha um tipo específico de algoritmo de otimização, os algoritmos genéticos, utilizado no trabalho.

Este trabalho se fundamenta em conhecimentos da Engenharia de Software para tratar problemas de comunicação existentes no Desenvolvimento Distribuído de Software, utilizando a Engenharia de Software Baseada em Buscas de forma a tratar um problema considerado NP-Completo.

2.1 Engenharia de Software

Na década de 1970, a indústria da computação ficou marcada pela grande evolução do *hardware*, enquanto que a indústria do software sofria de uma escassez de conhecimento, o qual foi denominada Crise do Software (SOMMERVILLE, 2007). Com isso, a busca pelo conhecimento de forma a melhorar o desenvolvimento de software deu origem a uma nova área conhecida como Engenharia de Software.

Com o passar do tempo, o desenvolvimento de software foi se organizando e se tornando sistemático. Uma formulação de métodos de trabalho foi iniciada com o surgimento de paradigmas de desenvolvimento. O primeiro deles foi denominado **paradigma de cascata**, também conhecido como sequencial, justamente pela sua natureza ser a sequencialidade dos passos do desenvolvimento de software, que são: requisito, análise, projeto, codificação, testes e manutenção. Segundo (PRESSMAN, 2011), o modelo cascata deve ser usado para que cada fase seja concluída em sua totalidade, passando então para a fase seguinte.

Com o tempo, o modelo cascata foi aprimorado para modelos que suportassem mais de um ciclo de desenvolvimento de software, passando pelas fases de desenvolvimento mais de uma vez. Os **modelos em espiral**, por exemplo, trazem aspectos gerenciais ao ambiente de

desenvolvimento, eliminando a visão sequencial e fornecendo a ideia de ciclos com a criação de protótipos e o seu aperfeiçoamento com os diversos ciclos.

Nos anos 90, a evolução do modelo espiral consolidou um novo paradigma conhecido como **Iterativo e Incremental** (SOMMERVILLE, 2007), que além dos aspectos gerenciais, os ciclos não mais geram protótipos, mas sim uma versão parcial do produto a ser entregue. Assim, o *time-to-market* é reduzido e o cliente pode utilizar parcialmente o software, mesmo que ainda carente de todas as funcionalidade requisitadas. Nesse modelo, o desenvolvimento de software é feito de forma a aceitar com maior naturalidade as mudanças que virão a acontecer nas várias iterações, onde cada iteração será análoga a um projeto, passando por uma análise de riscos e todas as fases do ciclo desenvolvimento.

Com esse novo paradigma de desenvolvimento, o tempo entre as iterações passou a ser utilizado de forma a otimizar o desenvolvimento de software de acordo com a necessidade de *time-to-market* e o amadurecimento dos casos de uso requisitados pelos clientes, fazendo com que duas vertentes de modelos iterativos e incrementais fossem se distinguindo. O modelo inicial, no qual se enquadra o **Processo Unificado**, se baseia na criação de uma vasta quantidade de artefatos de diversas necessidades, como diagramas de classe, diagramas de sequência, diagrama de atividades, entre outros. Essa vertente, conhecida como modelo **iterativo e incremental baseado em documentação**, tem como objetivo o amadurecimento completo dos requisitos para que a implementação seja iniciada. Por outro lado, a vertente de **desenvolvimento ágil** tem como objetivo a redução do tempo das iterações, para que as entregas sejam feitas com maior frequência, porém com menor conteúdo. A escolha das vertentes depende muito da organização e dos produtos a serem desenvolvidos, bem como o amadurecimento dos requisitos dos clientes.

Com o passar do tempo, baseado nos paradigmas de desenvolvimento, diferentes processos que mostram como deve ser praticado o desenvolvimento de software foram criados, como Rational Unified Process (RUP), eXtreme Programming (XP) e o Scrum, que são alguns dos mais comuns processos de software.

Com o amadurecimento do desenvolvimento de software, o uso de **Desenvolvimento Baseado em Componentes** (DBC) passou a ser utilizado de forma a reaproveitar componentes já desenvolvidos e que possuem similaridade de requisitos, mesmo que em diferentes projetos de software. Com isso, o reuso passou a ser visto como um novo caminho para o desenvolvimento de software, o que desencadeou posteriormente a técnica de **Linhas**

de Produtos de Software (LPS), que propõe sistematizar a produção de famílias de produtos com base nos conceitos de *plataforma* e *customização*, de forma análoga as linhas de montagens de carros, criados por Ford (POHL, BOCKLE e VAN DER LINDEN, 2005). A grande diferença entre LPS e DBC é que o reuso nas LPS é previsto com a adoção de dois processos distintos, a Engenharia de Domínio (ED) e a Engenharia de Aplicação (EA). A ED é o processo realizado para todos os produtos de uma mesma família de produtos, alimentando uma plataforma com componentes reutilizáveis. É o desenvolvimento para reuso. A EA é o processo que instancia um produto especificamente, se usando da plataforma existente para a criação de produtos para clientes específicos. É o desenvolvimento com reuso.

A engenharia de software evoluiu e continua a evoluir bastante. Como pode ser percebido, existe uma linha sequencial evolucionária entre os processos de software, definindo um conjunto de processos que nasceram no modelo cascata até a definição de linhas de produto. Entretanto, isso não faz com que todas as empresas de software utilizem os modelos mais modernos, uma vez que é uma escolha pessoal dos gerentes de projetos. Isso faz com que exista um conjunto de equipes que podem utilizar diferentes modelos ou processos de software, o que causa impacto direto no DDS, uma vez que diferentes equipes provavelmente terão processos diferentes.

2.2 Desenvolvimento Distribuído de Software

A globalização e os altos investimentos de empresas ao redor do mundo vêm fazendo com que novas formas de colaboração e competição na área da Engenharia de Software venham sendo amadurecidas (HERBSLEB e MOITRA, 2001). O principal fator é a transformação dos mercados locais para mercados globais. Neste período de amadurecimento, o mercado vivenciou grandes crises, como problemas de altos índices de falhas em projetos e o aumento considerável da demanda de trabalho (PRIKLADNICKI e AUDY, 2006).

Nos últimos anos, a busca por melhores oportunidades de negócio, seja para reduzir custos de desenvolvimento ou alavancar vendas, bem como a escassez de profissionais qualificados, têm motivado a indústria de software a adotar abordagens de desenvolvimento distribuído de software (GUMM, 2006). Com a globalização, o uso de Desenvolvimento Distribuído de Software (DDS) passou a ser possível.

As principais vantagens do uso de DDS são a aproximação do mercado global, a redução do *time-to-market*, o aumento da qualidade do software e a redução dos custos de

produção do mesmo (HERBSLEB e GRINTER, 1999). Adicionalmente, o DDS oferece a possibilidade de utilizar terceirização (*outsourcing*) em qualquer lugar do mundo, facilitando a busca por mão-de-obra qualificada ou barata, como acontece em polos como a Índia (KOBITZSCH, ROMBACH e FELDMANN, 2001).

Uma outra vantagem da adoção do DDS é o uso do *follow-the-sun*, forma de desenvolvimento de software em que sempre vai existir ao menos uma equipe de desenvolvimento trabalhando no software em algum lugar do mundo (AUDY e PRIKLADNICKI, 2007). Outra forma de DDS, diferente do *follow-the-sun*, é o uso de desenvolvimento em paralelo, onde as equipes trabalham simultaneamente, mas não se preocupam em manter o desenvolvimento por 24h, uma vez que o ganho está no paralelismo de várias equipes de desenvolvimento.

Os maiores problemas com o uso de DDS são derivados da distância entre os envolvidos, que desencadeia a falta de comunicação efetiva e dificuldade para compartilhamento de conhecimentos e experiências entre os membros das equipes. Esses problemas podem vir a trazer um aumento no tempo para o desenvolvimento do projeto e, em casos mais extremos, até o cancelamento do projeto como um todo (SOUZA, COSTA, *et al.*, 2008) (AUDY e PRIKLADNICKI, 2007). A comunicação entre equipes geograficamente distribuídas pode não ser fácil, eficiente e efetiva, devido às inúmeras diferenças que podem existir entre elas, em termos de disponibilidade, recursos, conhecimento técnico e valores sócio-culturais (HERBSLEB, 2007), como por exemplo, a diferença de fuso-horário, idioma, processos de software utilizados, ferramentas de comunicação, entre outros.

O DDS pode ser classificado das seguintes formas (AUDY e PRIKLADNICKI, 2007):

- **Onshore insourcing:** uma organização possui mais de um departamento de desenvolvimento no mesmo país;
- **Onshore outsourcing:** uma organização terceiriza parte do desenvolvimento de software com outra organização do mesmo país de origem;
- **Offshore insourcing:** uma organização possui mais de um departamento de desenvolvimento de software em diferentes países;
- **Offshore outsourcing:** uma organização terceiriza parte do desenvolvimento de software com outra organização de país diferente do país de origem;

Essa classificação, apesar de ser vista como quatro formas diferentes, podem possuir pontos semelhantes e, ao mesmo tempo, podem possuir classificações iguais com

características diferentes. Até porque, os problemas existentes no desenvolvimento de software convencional (centralizado) passam a ser mais críticos com o uso de DDS, como, por exemplo, a heterogeneidade de processos de software nas diferentes equipes. Portanto, a proposta dessa dissertação é encontrar pontos que caracterizem as equipes pelo ponto de vista não-técnico para que seja possível encontrar pontos semelhantes e conflitantes entre as equipes que desempenham o DDS.

2.3 Engenharia de Software Baseada em Buscas

A Engenharia de Software Baseada em Buscas (ESBB), como seu nome sugere, lida com problemas da engenharia de software como problemas de busca, utilizando algoritmos de otimização para automatizar a construção e busca de soluções para tais problemas (HARMAN e JONES, 2001). É uma área de pesquisa que cresceu bastante nos últimos anos, devido a sua aplicabilidade, principalmente em problemas NP-completos, ou seja, intratáveis computacionalmente por algoritmos de busca extensiva.

Alguns dos problemas tratados pela ESBB são problemas de alocação de recursos, estimativas de tamanho, estimativas de custo, seleção de requisitos, priorização de requisitos, melhoria de qualidade da arquitetura, otimização de código fonte, seleção de casos de teste, geração de casos de testes (HARMAN e JONES, 2001), entre outros.

Em alguns casos, as buscas exaustivas podem ser usadas, quando o universo de soluções possíveis não é demasiadamente grande. Entretanto, alguns desses problemas possuem complexidade polinomial ou exponencial, o que os torna impossíveis de serem tratados em um tempo computacionalmente hábil (ULLMAN, 1975). Para isso, técnicas baseadas em meta-heurística são usadas nos algoritmos de otimização com o propósito de encontrar uma solução ótima, ou um conjunto de soluções satisfatórias próximas da ótima em tempo hábil.

Uma meta-heurística é um conjunto de métodos genéricos que podem ser utilizados em uma vasta quantidade de problemas. Cada um desses problemas possui características unitárias que são mapeadas em funções e operações. Alguns dos algoritmos baseados em meta-heurísticas utilizadas são: Subida da Montanha (*Hill Climbing*), Busca Tabu (*Tabu Search*), Arrefecimento Simulado (*Simulated Annealing*) e Algoritmos Genéticos (*Genetic Algorithms - GA*), sendo o último, utilizado neste trabalho.

O método de Subida da Montanha utiliza a metáfora dos escaladores para buscar por melhores soluções. Uma solução é usada para derivar uma nova solução, chamada de solução

vizinha por ser ligeiramente diferente da solução inicial. Caso essa solução seja considerada melhor, ela passa a ser usada como novo ponto de referência, caso não seja melhor, ela é descartada. O método continua até o momento em que melhores soluções deixam de ser encontradas. De forma bastante semelhante, funciona a Busca Tabu, que vasculha as soluções vizinhas, verificando uma lista tabu que armazena as soluções já encontradas, até que um critério de parada seja atingido. O Arrefecimento Simulado simula a técnica de recozimento utilizada na metalurgia, na qual o metal é aquecido até seu ponto de fusão e resfriado lentamente, tornando-se sólido novamente. O resfriamento e aquecimento são utilizados para armazenar soluções boas e ruins de forma a vasculhar melhor o espaço de busca, identificando soluções melhores com o passar do tempo (KIRKPATRICK, GELATT e VECCHI, 1983). O algoritmo genético utiliza a metáfora do processo de evolução natural, onde os mais fortes sobrevivem diante dos mais fracos, sendo os mais fortes utilizados para derivar novas soluções com base nos operadores genéticos de cruzamento e mutação (LINDEN, 2008). Os algoritmos baseados em meta-heurísticas já são bastante utilizados, com aplicabilidade em áreas como engenharia mecânica, engenharia química, engenharia médica e biomédica, engenharia civil e engenharia eletrônica, além da engenharia de software (HARMAN e JONES, 2001).

Para a utilização da ESBB, uma série de passos deve ser tomada para reformular um problema da engenharia de software em um problema de otimização. Os três passos identificados são (HARMAN e JONES, 2001): (i) criar a representação do problema; (ii) definir uma função objetivo e (iii) definir os operadores de manipulação.

A abordagem aqui proposta utiliza um algoritmo baseado na meta-heurística de algoritmos genéticos. Essa escolha foi feita pela representação do problema ser facilmente traduzida para a representação de um algoritmo genético. Até mesmo por já existirem outras abordagens de alocação, sem utilização de DDS, que utilizam esse tipo de algoritmo (BARRETO, 2005).

Como mencionado acima, a abordagem proposta nesta dissertação define um algoritmo baseado na meta-heurística de algoritmos genéticos. Seu propósito é a alocação de equipes de desenvolvimento de software para a implementação de módulos de software. Essa alocação, por sua vez, deve ser baseada na necessidade de comunicação entre as equipes que implementarão módulos dependentes, relacionando os atributos não-técnicos das equipes e a dependência entre os módulos de software.

2.3.1 Algoritmos Genéticos

Os algoritmos genéticos (GA – *Genetic Algorithm*) são um tipo de algoritmo evolucionário, dessa forma, é utilizando um modelo semelhante aos processos naturais de evolução, que relatam a sobrevivência dos mais fortes diante dos mais fracos (FORREST, 1996).

Nos GAs, populações de indivíduos são criadas e submetidas a operadores genéticos. Esses operadores utilizam uma caracterização da qualidade de cada indivíduo, chamada de avaliação ou *fitness*, e através de um processo de evolução natural destes indivíduos, geram-se novos indivíduos. Estes novos indivíduos podem ou não ser mais fortes que os anteriores, o que, no mínimo, resultará em mais diversidade para a população (LINDEN, 2008).

A terminologia utilizada por um GA também se baseia na metáfora da evolução natural. Um indivíduo da população é chamado de cromossomo, que representa uma solução para um problema de otimização. Cada uma das características desse cromossomo é conhecida como gene e os valores associados a cada um dos genes são os alelos.

Os GAs são técnicas heurísticas de otimização global, ou seja, se opõem aos métodos como gradiente (Subida da Montanha), que seguem a derivada de uma função de forma a encontrar o ponto máximo próximo a ela, o que pode acarretar em um máximo local. Um GA utiliza a variedade genética de todo um universo de busca que amadurece na população com o decorrer de sua execução. Para que exista um ponto de equilíbrio entre a variedade genética e a convergência genética, existem os operadores genéticos de cruzamento e mutação.

Esses operadores de cruzamento e mutação geram novos cromossomos, respectivamente, com uma operação sexuada e uma operação assexuada. Os operadores genéticos consistem em passos estruturados que se aplicam aos cromossomos da população a fim de melhorar a qualidade da mesma (RUSSEL e NORVIG, 1996).

O operador de cruzamento necessita da seleção de dois cromossomos presentes na população (cromossomos pais) para que uma recombinação seja realizada e novos cromossomos sejam criados. Os novos cromossomos (cromossomos filhos) possuem parte do material genético (genes) de cada um dos pais. O operador de mutação utiliza a seleção de um cromossomo pai e faz a alteração de um gene escolhido aleatoriamente por um dos alelos disponíveis, definindo um novo indivíduo para a população.

Existem diversas formas de seleção de pais, como seleção por roleta viciada, por torneios, por ranqueamento. (HAUPT, 2009) (LINDEN, 2008) (LOUREIRO, MARGOTO, *et*

al., 2005). O método de roleta viciada faz com que a probabilidade de um cromossomo ser sorteado seja proporcional a sua avaliação. O método por ranqueamento utiliza a posição do cromossomo, após o ranqueamento pela avaliação, para definir o percentual de chance de ser sorteado para ser um cromossomo pai, necessitando de uma função que converta a posição em percentual. O método de torneio, por sua vez, é o mais simples, sorteando um número n de cromossomos e selecionando o mais forte dentre os que foram sorteados.

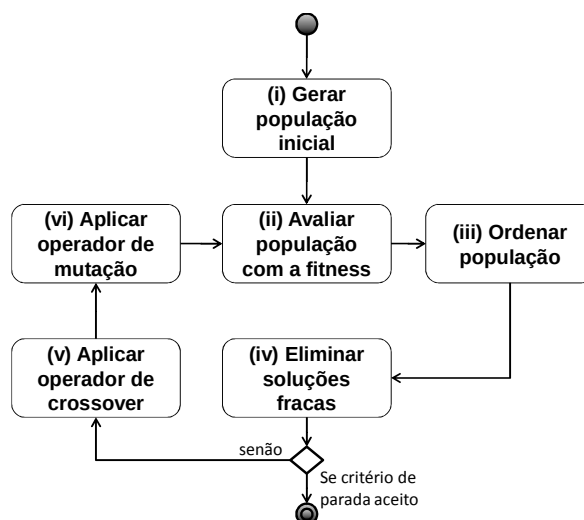
O operador genético de cruzamento utiliza a seleção de dois pais. Ambos terão partes de seu material genético (genes) usadas para criar outro cromossomo para a população, parte do primeiro pai, parte do segundo. Sendo assim, os filhos gerados depois de uma sequência de cruzamentos serão soluções que tendem a ser mais semelhantes aos mais fortes cromossomos da população. Com o tempo, essa forma de reprodução fará com que a população seja convergida a um padrão genético, fazendo com que não se consiga mais evoluir para soluções diferentes. Para que a convergência se torne mais lenta, a mutação também deve ser utilizada, mantendo variedade genética para a população. Na mutação, um gene do cromossomo possui seu material genético modificado por outro valor, que deve ser selecionado aleatoriamente dentre os alelos, justamente para diversificar as novas soluções.

Existem diversas formas de cruzamento, como cruzamento de um ponto, de n pontos, cruzamento uniforme, entre outros. O cruzamento de um ponto realiza um corte em um ponto aleatório do cromossomo, dividindo os genes em genes *anteriores* ao corte e genes *posteriores* o corte. Com a seleção de dois pais, um dos pais oferecerá seus genes da posição anterior ao corte e o outro oferecerá os genes da posição posterior, gerando o novo cromossomo. Com o caso do cruzamento de n pontos, são realizados n cortes, a partir daí são utilizados dois pais (ou $n+1$ pais) para distribuir seus genes alternadamente. No caso de $n = 2$ (dois cortes) o que é feito é alternar o primeiro pai, o segundo pai e o primeiro pai novamente. Para o cruzamento uniforme, são escolhidos dois pais. Para cada gene é sorteado um valor binário, caso o valor seja zero, ele recebe o gene do primeiro pai, caso seja 1, recebe do segundo pai, gerando o cromossomo.

Para que um GA possa ser usado para um determinado problema, o problema precisa ser transformado em um problema de busca, o que significa dizer a tradução de uma solução em forma de cromossomo, a caracterização dos operadores genéticos para o problema em questão e a definição de uma função objetivo (avaliação) que define o quão forte é um cromossomo.

A sequência de passos da execução de um algoritmo genético, como pode ser visto na Figura 1, é: (i) gerar população inicial; (ii) avaliar população com função objetivo; (iii) ordenar soluções com base na função objetivo; (iv) eliminar soluções fracas; (v) aplicar operador de cruzamento e (vi) aplicar operador de mutação. A segunda iteração do algoritmo não necessita gerar uma nova população inicial, logo, se inicia no passo (ii).

Figura 1 – Sequência de passos de um algoritmo genético



O passo (iv) só deve ser utilizado caso o algoritmo genético em questão seja um algoritmo elitista, fazendo com que a nova população possa ter cromossomos pais e filhos. No caso de um GA não elitista, a nova população só possui cromossomos filhos da geração anterior (HAUPT, 2009).

Além disso, alguns parâmetros são configuráveis em todos os GAs e variam de acordo com o problema. São eles: (i) tamanho da população e (ii) número de iterações (ou gerações). O aumento do número da população vai fazer com que maior material genético seja mantido durante a execução do algoritmo, o que é vantajoso por um lado, mas tem como lado negativo o maior tempo de execução e o maior consumo de memória. O aumento do número de iterações também pode ser benéfico para o GA, mas também aumenta o tempo de execução e nem sempre apresenta melhores resultados (LINDEN, 2008).

2.4 Considerações finais

Esse capítulo apresentou as principais áreas de estudo na qual o trabalho está envolvido, embasando o leitor nos conhecimentos que se mostram importantes para o bom entendimento. Foram abordados temas como a engenharia de software e sua evolução histórica, bem como o desenvolvimento distribuído de software e a engenharia de software baseada em buscas. O conteúdo de cada um dos temas foi baseado na relevância exigida para os próximos capítulos.

Capítulo 3

Trabalhos Relacionados

Este capítulo tem como objetivo apresentar alguns trabalhos que possuem relação direta com a presente proposta, evidenciando os pontos positivos e negativos de cada uma deles, além de fazer um comparativo entre as abordagens relacionadas. Para um melhor entendimento, o comparativo entre os trabalhos relacionados e a abordagem aqui proposta está presente no capítulo de considerações finais.

Foram escolhidos trabalhos de alocação que lidam diretamente com alocação em desenvolvimento de software e que possuem alguma relação com o DDS.

3.1 Modelo para o desenvolvimento 24h

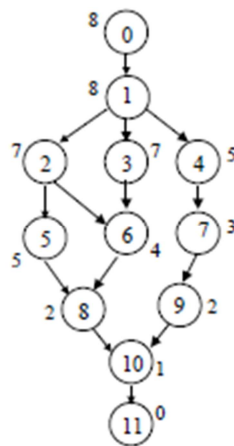
O modelo para desenvolvimento 24h (JALOTE e JAIN, 2006), tem como principal objetivo a alocação de membros de uma equipe com o propósito de potencializar os casos em que o desenvolvimento distribuído pode ser usado com o *follow the sun*, ou desenvolvimento 24h.

Para essa alocação, as tarefas que terão as equipes alocadas possuem para sua avaliação o esforço para execução da atividade, bem como uma verificação da dependência entre as atividades envolvidas, por conta do *follow the sun*, já que segundo o autor, caso não exista nenhuma dependência, não há vantagem em se usar o *follow the sun*. Além disso, são analisadas as habilidades requeridas e os recursos requeridos para realizar a atividade.

O algoritmo se baseia no método de caminho crítico, método este que é designado para problemas em que o conjunto de tarefas ou atividades não possui margem de atraso. É utilizado um Grafo Acíclico Direcionado (GAD) no qual suas arestas representam as dependências existentes, como por exemplo, tarefas com restrições. E também três conjuntos de recursos, representando os membros alocados a cada local remoto. Esses três recursos representam a fração de 8 horas no desenvolvimento. O método de caminho crítico busca realizar as alocações de modo a reduzir a duração do projeto.

Um exemplo de GAD para o modelo de desenvolvimento 24h pode ser visto na Figura 2. Nota-se que cada ponto do grafo possui valores internos, que representam os números das atividades, e um número externo, que representa o tempo restante, valor este que é baseado em uma variável de habilidade, que é relacionada à alocação.

Figura 2 – Grafo Acíclico Direcionado para o modelo de desenvolvimento 24h



Para o cálculo da função que define o valor da alocação há uma equação que se baseia na soma de todos os custos de tempo: $weight(\tau) = \max_k \sum_{i \in \phi_k} time(i)$.

Para análise do algoritmo, foi realizada uma avaliação experimental. Para tal, foram gerados randomicamente 100 grafos de atividades, com a quantidade de atividades variando entre 5 e 100. O tempo atribuído para cada atividade é uniformemente distribuído pela margem de 1 a 20 unidades de tempo. O número de atividades que foram designadas como habilidade necessária varia de 10% a 70% do total presente no grafo gerado aleatoriamente. O conjunto de grafos de atividade foram testados com três diferentes tamanhos de conjuntos de recursos, que foram 3, 5 e mais de 5 recursos. Como critério de comparação, o algoritmo também foi executado supondo a remoção do DDS, utilizando apenas equipes em uma mesma localização. Os resultados mostraram que quanto maior é a quantidade de recursos, maior é o ganho obtido pelo algoritmo com o DDS.

3.2 Modelo de simulação Global Software Development

O modelo de simulação GSD (*Global Software Development*) (SETAMANIT, WAKELAND e RAFFO, 2007) se baseia na utilização de dois paradigmas, o primeiro, voltado para a habilidade de se representar a estrutura do processo e os mecanismos usados para transferência de trabalho, enquanto que o segundo é coordenar atividades.

Os autores identificaram três níveis de fatores que afetam o desempenho e produtividade em projetos distribuídos, são eles: (i) fatores fundamentais; (ii) fatores estratégicos; e (iii) fatores organizacionais. Os fatores fundamentais incluem problemas de comunicação, diferenças culturais e diferenças de fuso-horário. Os fatores estratégicos estão ligados ao amadurecimento da arquitetura, estratégia de alocação de recursos e distribuição dos recursos desperdiçados. Por fim, os fatores organizacionais, uma vez que se trata de uma simulação, existem os impactos causados pelas equipes virtuais.

A estrutura do modelo é dividida em três componentes: submodelo de evento discreto (DES – *Discrete-event sub-model*), submodelo de sistema dinâmico (SD – *System Dynamics sub-model*) e submodelo de efeito de interação (IE – *Interaction Effect sub-model*).

O DES é responsável por possibilitar a captura do impacto das diferenças dos fatores fundamentais, como o fuso-horário. O SD captura o acompanhamento total do projeto, incluindo planejamento e controle de atividades. O SD é dividido em três partes: Recurso humano, Planejamento e Controle. O recurso humano age como uma interface entre os recursos humanos e os outros módulos do SD. O planejamento monitora e identifica a força de trabalho necessária para vencer o planejamento do projeto. E o controle recebe a informação do progresso do projeto e avalia os ajustes que precisam ser feitos no planejamento. Cada ponto de desenvolvimento remoto tem seu próprio submodelo SD, que representa as características que cada recurso humano tem independente de outro, como produtividade, geração de defeitos, taxa de detecção, entre outros.

O submodelo IE representa os efeitos de interação quando recursos de diferentes pontos remotos precisam colaborar ou trabalhar conjuntamente. O IE representa a produtividade colaborativa como diferente da atividade realizada independente, como uma forma de avaliar os custos, uma vez que o submodelo IE calcula a eficiência de coordenação de uma equipe distribuída e então aplica o efeito de coordenação à produtividade da mesma, para só então enviar os valores para o submodelo DES. Se a eficiência de coordenação da equipe é baixa, a produtividade será baixa. O algoritmo utiliza dois fatores para determinar a eficiência de coordenação: frequência de comunicação e confiança.

O trabalho não apresenta documentação de como suas funções funcionam, porém, sua alocação é feita com base nos locais remotos e membros, considerando fatores como etapas do processo de desenvolvimento (requisitos, projeto, implementação, teste e retrabalho),

atividades desenvolvidas, estratégia de alocação, produtos de trabalho, recursos humanos, fuso-horário e meios de comunicação.

3.3 DIMANAGER

O DIMANAGER faz parte de um Ambiente de Desenvolvimento Distribuído de Software (ADDs) chamado DiSEN-PPM (*Distributed Software Environment – Project Management Model*) (HUZITA, SILVA, *et al.*, 2008).

O DiSEN tem como características a formação de equipes virtuais e uso necessário de ferramentas de apoio para o desenvolvimento de software. A especificação das atividades e participantes distribuídos por locais remotos exige um controle mais rigoroso por parte do gerente de projetos, que é apoiada no ambiente DiSEN pela ferramenta DIMANAGER. Essa ferramenta faz uso de um mecanismo que foi implementado com base em agentes de software e que considera o conhecimento, habilidade e disponibilidade das pessoas em diferentes locais para a alocação de recursos humanos.

A ferramenta utiliza Lógica *Fuzzy* para quantificar os valores atribuídos aos critérios considerados, alocando membros de uma equipe e se baseando em conhecimentos, disponibilidade e habilidades desses membros. Além disso, é considerada a afinidade entre os envolvidos.

A modelagem mostra apenas uma caixa preta no que diz respeito ao algoritmo utilizado, só deixando claro que é utilizada Lógica *Fuzzy* e o uso de agentes inteligentes. Além disso, o ambiente DiSEN também conta com uma ontologia que é capaz de responder a perguntas relacionadas ao projeto com base nas informações das equipes, informações estas que são cadastradas em uma ontologia (OntoDiSEN) e utilizada para as respostas.

3.4 TAMRI: Task Allocation based on Multi cRiteria

O TAMRI (LAMERSDORF e MÜNCH, 2010) é uma ferramenta que pode auxiliar o gerente do projeto em identificar alocações de tarefas que auxiliem no andamento do projeto. Seu algoritmo utiliza: (i) um conjunto de objetivos; (ii) características das atividades do projeto e os membros remotos disponíveis; e (iii) relacionamentos causais entre características e objetivos.

O TAMRI conta com um conjunto de atributos não-técnicos, como idiomas, diferenças culturais, fuso-horário, dependência entre atividades, problemas de comunicação e

tempo de desenvolvimento. Os atributos são colocados diretamente em uma ferramenta que obtém os dados e os armazena, sem nenhum processamento inicial das informações.

O TAMRI apresenta um método baseada em redes Bayesianas para alocar tarefas a membros distribuídos. Neste método, aspectos técnicos e não-técnicos das equipes são considerados. Por exemplo, o custo para realizar as atividades e o custo de comunicação entre as localidades são os fatores considerados no momento da alocação.

O algoritmo inicialmente instancia uma rede bayesiana para os custos de comunicação e cria randomicamente os caminhos para se definir uma alocação, sempre observando o custo de comunicação. Os resultados são mostrados ao gerente do projeto em ordem de melhor resultado encontrado.

3.5 Comparativo da abordagem proposta com os trabalhos relacionados

Existem diversos trabalhos relacionados a alocação de equipes, seja para formação de equipes ou para alocação de atividades para membros de uma equipe. Entretanto, pelo trabalho ter foco principal no DDS, os trabalhos relacionados selecionados foram apenas os trabalhos de alocação voltados para o DDS.

Apesar de ser uma nova área de pesquisa para o DDS, já é possível perceber uma série de trabalhos capazes de lidar com as novas características presentes em um ambiente de desenvolvimento de software realizado de forma distribuída. O que se pode perceber dentre os trabalhos relacionados é uma clara distinção dos objetivos de cada um dos projetos.

A ferramenta TAMRI se mostra como a mais próxima da abordagem aqui proposta, uma vez que ela considera mais atributos não-técnicos que os demais trabalhos relacionados. Entretanto, sua estratégia de alocação não considera equipes, mas sim membros individuais que estão distribuídos, fazendo com que se tenha a noção de que todos os membros distribuídos fazem parte de uma mesma organização.

Dessa forma, alguns dos aspectos que deveriam ser levados em conta no DDS com *outsourcing* são deixados de lado, como os modelos de processo e de maturidade. Além disso, não é detalhado como a obtenção dos atributos não-técnicos é feita. O foco do trabalho está na definição de um algoritmo que lida com as informações, mas sem tratar a obtenção das mesmas.

Além disso, o algoritmo não é detalhado. Embora considere diretamente aspectos não-técnicos das equipes, esta ferramenta não define mecanismos para coletar informações sobre

tais aspectos não-técnicos. Ao contrário da abordagem aqui proposta (juntamente com o *framework* de recomendações), cuja coleta de informações é baseada em questionários submetidos aos gerentes de equipes e seus membros.

O modelo de simulação GSD oferece como diferencial em relação a abordagem proposta a coordenação das equipes distribuídas, verificando a manutenção das atividades e se elas estão acontecendo no tempo esperado. Esse modelo é mais próximo dos níveis de processo, uma vez que, dos atributos não-técnicos adotados pelo *framework* de recomendações, o modelo GSD utiliza apenas os atributos temporais e a manutenção dos atributos de processo e modelo de maturidade.

O DiMANAGER, juntamente com o *framework* de recomendações, são os únicos que consideram a afinidade para realizar as alocações. Como diferencial, a afinidade no DiMANAGER é avaliada por uma ontologia interna do ambiente DiSEN, enquanto na abordagem proposta é sinalizada pelo próprio gerente de equipe. Além disso, o DiMANAGER avalia critérios técnicos e apenas o critério de ferramentas de comunicação para os critérios não-técnicos.

O Modelo de desenvolvimento 24h tem como diferencial em relação a abordagem proposta a utilização do *Follow-the-sun* como foco principal, evitando o paralelismo e utilizando equipes que complementam o tempo de desenvolvimento para 24h por dia. Ambos os trabalhos analisam as dependências entre as atividades, mas o foco da dependência do modelo 24h é no *follow-the-sun*, ou seja, para a continuidade de atividades, enquanto que na abordagem proposta o foco é alocar equipes com facilidades de comunicação. No modelo de desenvolvimento 24h, as equipes são tecnicamente avaliadas, porém, apenas os atributos temporais fazem parte dos critérios não-técnicos.

Na Tabela 1 é possível identificar os pontos presentes em cada abordagem.

Tabela 1 – Comparativo entre os trabalhos relacionados e o *framework*

	Follow-the-sun	Dependência	Coordenação	Requisitos Técnicos	Volume de trabalho	Temporal	Idioma	Processo	Modelo de Maturidade	Ferramentas de comunicação	Awareness	Religião	Tomada de decisão	Formalidade na documentação	Afinidade
Framework Proposto		Sim		Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
TAMRI		Sim		Sim	Sim	Sim	Sim			Sim		Sim			
Sim. GSD			Sim		Sim	Sim		Sim	Sim						
DiManager				Sim	Sim	Sim				Sim					Sim
Modelo 24h	Sim	Sim		Sim	Sim	Sim									

Outro ponto importante a se destacar é que os trabalhos relacionados identificados aqui não fazem a separação entre a análise dos atributos técnicos e dos atributos não-técnicos. Com essa separação, a abordagem proposta tem a liberdade de tratar os atributos com critérios e algoritmos diferentes. No caso da análise técnica, realizada na segunda fase do *framework* de recomendações do qual a abordagem aqui proposta faz parte, possibilita a criação de diversas políticas de seleção de equipes, sem afetar a análise não-técnica.

Portanto, a abordagem aqui proposta tem como foco principal os atributos não-técnicos, mas os atributos técnicos são considerados e avaliados na segunda fase do *framework* de recomendações, o que faz com que a abordagem aqui proposta tenha foco nos atributos não-técnicos, sem relevar os atributos técnicos.

Capítulo 4

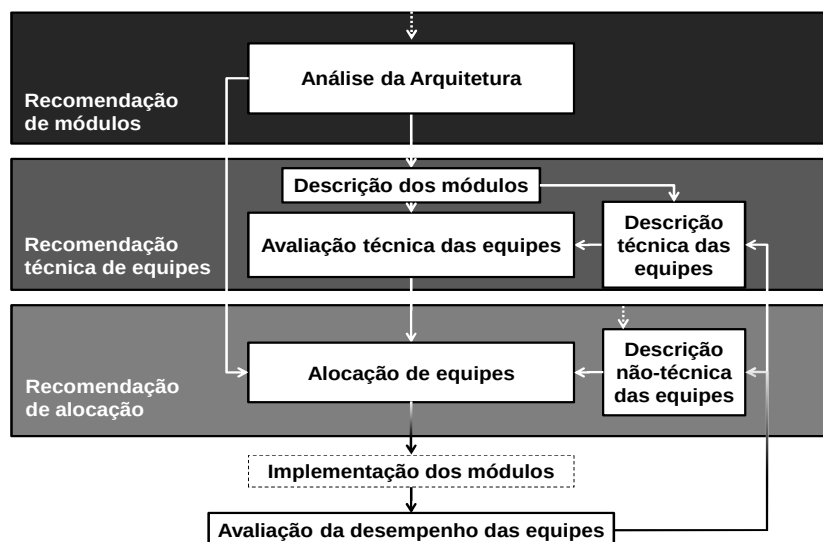
Framework de Recomendações

O presente trabalho está contido em um *framework* de recomendação para alocação de equipes globais para projetos de linhas de produto de software (PEREIRA, SANTOS, *et al.*, 2010). Neste capítulo, o *framework* é introduzido, deixando bem clara a separação em três recomendações distintas, onde a terceira delas compõe o presente trabalho.

O *framework* tem a proposta de mitigar os problemas de comunicação enfrentados durante a fase de implementação de uma LPS, onde as equipes candidatas para desempenhar essa atividade são equipes distribuídas globalmente.

Como pode ser observado na Figura 3, o *framework* é dividido em três níveis de recomendações: i) recomendação de módulos de software ii) recomendação técnica de equipes e iii) recomendação de alocação de equipes.

Figura 3 – Framework de recomendações

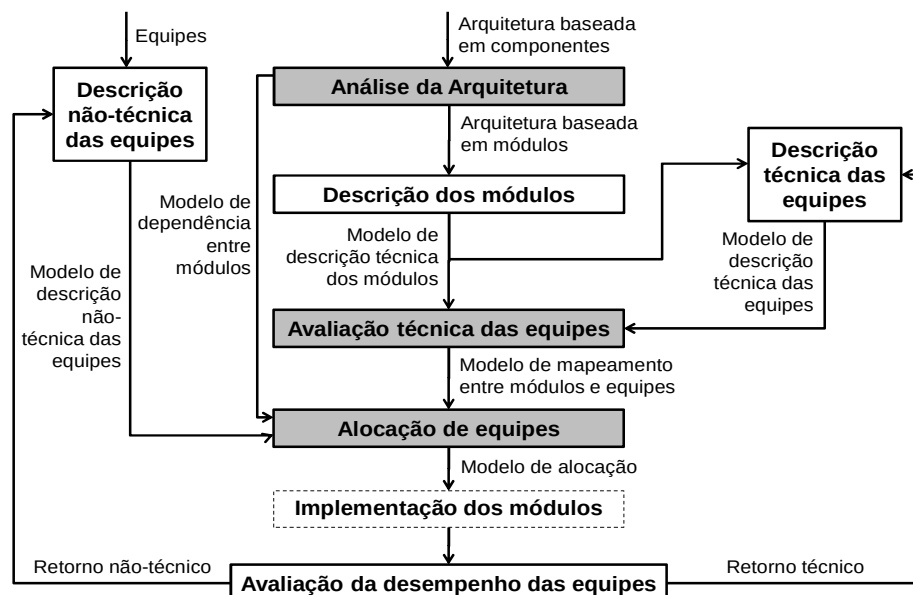


A maior parte das abordagens existentes para reduzir problemas relacionados a DDS visam resolvê-los após o seu acontecimento (GUMM, 2006), ou seja, funcionam de forma reativa ao acontecimento do problema, com o objetivo de remediá-lo. Com o *framework* de recomendações, o objetivo é reduzir a possibilidade dos problemas acontecerem, funcionando de uma forma proativa, prevenindo que problemas de comunicação aconteçam, reduzindo a

necessidade de comunicação nos casos onde os componentes seriam demasiadamente acoplados, além de possibilitar a alocação apenas de equipes tecnicamente qualificadas e com melhores condições de comunicação com base em atributos não-técnicos.

As próximas seções apresentam o detalhamento de cada uma das recomendações, além de uma etapa final, que acontece depois da implementação dos módulos, com o intuito de retroalimentar as características fornecidas pelas equipes. A Figura 4 mostra as fases sem a identificação das recomendações, mas deixando presente os artefatos que são criados e onde eles são utilizados como entrada.

Figura 4 – Artefatos gerados durante as recomendações



4.1 Recomendação de módulos de software

Na primeira fase do *framework* de recomendações é recebido como entrada uma arquitetura baseada em componentes da LPS. O propósito dessa fase é identificar componentes muito acoplados entre si e agrupá-los, gerando módulos de software com menor dependência entre si que quando separados em componentes (PEREIRA, 2011).

Sendo assim, um módulo de software representa a unidade de trabalho que deve ser implementada por uma equipe de desenvolvimento de software, ou seja, o conjunto de módulos $M = \{m_1, m_2, \dots, m_n\}$ que serão implementados por algumas das equipes presentes em um conjunto de equipes $E = \{e_1, e_2, \dots, e_m\}$.

A fim de alcançar o objetivo da recomendação de módulos, a fase de *análise da arquitetura* utiliza métricas arquiteturais com o objetivo de identificar os componentes mais

acoplados. A partir disso, o problema é transformado em um problema de otimização para que se possa buscar soluções que avaliem o agrupamento de componentes em módulos. O algoritmo baseado em meta-heurística utilizado é o de arrefecimento simulado.

Além disso, as métricas são identificadas entre componentes e entre módulos de software, fazendo com que se tenha a noção entre relacionamento externo e interno. Como são levados em consideração os módulos, os relacionamentos externos são métricas que identificam a dependência entre módulos. Já o relacionamento interno é uma métrica entre os componentes que fazem parte de um módulo, sendo uma característica interna ao módulo de software, que pode ser utilizada como volume de trabalho, uma vez que quanto mais complexo é um módulo, mais componentes com alta dependência existem nele.

O resultado da fase de recomendação de módulos é uma matriz que identifica os módulos e os relacionamentos internos e externos entre os módulos. Esse artefato é chamado *Module Based Matrix* (MBM) e é utilizado pela recomendação de alocação de equipes. Como pode ser visto na Figura 5, os valores presentes na diagonal principal da matriz representam os relacionamentos internos, já que denotam o acoplamento dos componentes que fazem parte deste módulo. As demais posições representam as dependências entre módulos, ou seja, o relacionamento externo aos módulos.

Figura 5 – Module Based Matrix (MBM)

	M1	M2	M3	M4
M1	1	0,209	0,215	0,210
M2	0,181	0,118	0,039	0,039
M3	0,200	0,039	0,130	0,039
M4	0,200	0,039	0,039	0,127

4.2 Recomendação técnica de equipes

Uma vez que os módulos são identificados na recomendação de módulos, a recomendação técnica de equipes tem o objetivo de oferecer um conjunto de equipes por módulo, consideradas como tecnicamente qualificada para a implementação do mesmo (SANTOS e ELIAS, 2011).

Para tal, os módulos devem ser caracterizados em termos de requisitos técnicos necessários para a sua implementação. Além disso, o conjunto de equipes $E = \{e_1, e_2, \dots, e_m\}$ também devem ser caracterizadas em termos de seus conhecimentos técnicos, que são

baseados em níveis de graduação e certificações. Cada uma dessas características podem ser relacionadas entre si, como por exemplo uma certificação de programador Java está associada a linguagem Java.

As descrições dos módulos e descrições técnicas das equipes devem ser usadas em conjunto com regras e políticas que definem o interesse do gerente de projeto sob os critérios de avaliação técnica. Assim, é possível recomendar um mapeamento $EM = \{(m, ET) | m \in M \wedge ET \subseteq E\}$, associando cada módulo de software m ao subconjunto de equipes ET que são tecnicamente qualificadas para implementar o módulo. Utilizando *lógica fuzzy*, termos “baixo”, “mediano” e “alto” são mapeados para os níveis de conhecimentos requeridos, fazendo com que o gerente do projeto possa criar políticas ou modificar as políticas existentes para atender necessidades específicas de cada projeto.

Por exemplo, se a política da organização é poupar equipes mais experientes para trabalhos mais difíceis, o processo de avaliação definirá como adequadas as equipes que possuírem conhecimentos mais próximos aos demandados pelos módulos. Por outro lado, se o gerente deseja finalizar o projeto o mais rápido possível, o conjunto de regras é alterado de forma que equipes com qualificações mais altas sejam sempre mais adequadas.

O resultado dessa recomendação é o artefato *Team x Module Mapping* (TMM) que identifica o conjunto EM mencionado para cada módulo a ser implementado. Como pode ser visto na Figura 6, cada módulo possui um conjunto independente de equipes consideradas qualificadas. Isso acontece justamente pela qualificação das equipes serem feitas módulo a módulo, tornando possível que cada um dos módulos tenha seus requisitos técnicos específicos para a sua implementação.

Figura 6 – Team x Module Mapping (TMM)

m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀
e ₁₄	e ₁₂	e ₁	e ₄	e ₉	e ₂	e ₁	e ₉	e ₁₃	e ₁₂
e ₁₁	e ₁	e ₁₉	e ₈	e ₅	e ₃	e ₉	e ₁	e ₁₀	e ₁₄
e ₂	e ₅	e ₃	e ₁₀	e ₁₂	e ₁	e ₅	e ₈	e ₅	e ₁
e ₅	e ₉	e ₆	e ₂	e ₁₀	e ₉	e ₄	e ₁₀	e ₂	e ₇
e ₁₉	e ₁₀	e ₇	e ₁₉	e ₁₁	e ₂₀	e ₁₁	e ₁₃	e ₂₀	e ₁₁
e ₃	e ₁₉	e ₂	e ₆	e ₄	e ₁₈	e ₁₂	e ₁₉	e ₃	e ₁₉
e ₁	e ₁₁	e ₁₃	e ₃	e ₅	e ₈	e ₁₇	e ₂₀	e ₉	e ₂₀
e ₄	e ₂	e ₄	e ₁₃	e ₇	e ₇	e ₁₃	e ₄	e ₂₁	e ₃
e ₇		e ₉	e ₁₅	e ₁		e ₇	e ₇	e ₁₁	e ₁₈
e ₁₂		e ₈	e ₁₁			e ₁₉		e ₁₄	e ₅
e ₉			e ₅					e ₁	e ₉

Equipes capacitadas para módulo M1

Equipes qualificadas por módulo

4.3 Recomendação de alocação de equipes

A recomendação de alocação de equipes utiliza o artefato MBM recebido pela recomendação de módulos e o artefato TMM da recomendação de equipes habilitadas para avaliar critérios não-técnicos. Sabendo-se que existe relação direta entre a dependência de dois componentes e o grau de comunicação necessário para a implementação destes componentes (GHEZZI, JAZAYERI e MANDRIOLI, 2002), essa recomendação tem o objetivo de oferecer ao gerente de projeto um conjunto de possibilidades de alocação de equipes, considerando apenas equipes tecnicamente qualificadas, avaliando as equipes a cerca de atributos não-técnicos e relacionando essa avaliação com as dependências entre os módulos (RIBEIRO e ELIAS, 2011), (RIBEIRO e ELIAS, 2011b), (RIBEIRO e ELIAS, 2010).

Para que isso seja possível, inicialmente as equipes são caracterizadas segundo características não-técnicas, tais como localização, idiomas conhecidos, disponibilidade, e características culturais e afinidade, ou seja, atributos relevantes no contexto de DDS. Com base nesses atributos é que é viável a avaliação das equipes quanto a possibilidade de boa comunicação. Além da caracterização, uma forma de se obter os atributos das equipes é necessária, baseando-se em questionários de perguntas qualitativas e diferenciais. Todas as perguntas do questionário são utilizadas para verificar semelhanças entre as equipes, porém, apenas as perguntas qualitativas também denotam qualidade na resposta obtida. Note que as equipes já são pré-estabelecidas, ou seja, são equipes prontas que se candidatam à alocação de equipes. Esse trabalho não tem a intensão de formar as equipes e sim utilizá-las de forma a possibilitar o uso do DDS com uma comunicação efetiva entre as equipes.

O problema é também transformado em um problema de otimização e é utilizado a meta-heurística de algoritmos genéticos para buscar resultados com base em avaliações não-técnicas.

Este trabalho tem como objetivo a instanciação da recomendação de alocação de equipes. O Capítulo 5 apresenta a abordagem proposta, apresentando a definição dos atributos não-técnicos e o detalhamento do algoritmo genético a ser utilizado. O Capítulo 6 detalha a prova de conceito realizada com a abordagem e apresenta seus resultados.

4.4 Avaliação de desempenho das equipes

Após a implementação dos módulos e consequentemente a sua integração, como pode ser visto na Figura 4, as equipes devem ser avaliadas em relação ao seu desempenho técnico e não-técnico.

Essa etapa é chamada de *Avaliação de desempenho das equipes* e ela considera os dois tipos de atributos citados (técnicos e não-técnicos). Seu propósito é atualizar os atributos de reputação técnica e de afinidade das equipes.

A reputação é um atributo técnico que é formado com base em uma série de avaliações de desempenho realizadas pelo gerente do projeto como um todo. A reputação é utilizada pela recomendação técnica de equipes para valorizar as equipes que possuem boa reputação e desvalorizar as equipes que não possuem bons trabalhos realizados anteriormente.

A afinidade, por sua vez, é um atributo bidirecional entre equipes. Nesse caso, cada gerente de equipe é responsável por avaliar as demais equipes que necessitaram se comunicar durante a implementação do software. Essa avaliação afetará a afinidade entre as equipes, que é um atributo não-técnico utilizado na recomendação de alocação de equipes. O detalhamento de como a afinidade é definida e modificada também faz parte do escopo desse trabalho e se encontra no Capítulo 5.

Capítulo 5

Abordagem para Recomendação de Alocação de Equipes

A presente abordagem tem como objetivo a instanciación de uma abordagem de recomendação de alocação de equipes para desenvolvimento distribuído de software baseado em atributos não-técnicos (RIBEIRO e ELIAS, 2011) (RIBEIRO e ELIAS, 2011b) (RIBEIRO e ELIAS, 2010). Esta abordagem faz parte do *framework* de recomendações para alocação de equipes distribuídas para projetos de Linhas de Produto de Software (PEREIRA, SANTOS, *et al.*, 2010) (SANTOS, PEREIRA, *et al.*, 2010).

Como foi visto na Figura 4, a **recomendação de alocação de equipes** utiliza como entrada informações de ambas as recomendações anteriores. A primeira delas, que corresponde à saída da **recomendação de módulos de software**, é a informação que define os módulos e os relacionamentos internos e externos existentes. A segunda é o mapeamento dos módulos e as respectivas equipes tecnicamente qualificadas à sua implementação, referente à **recomendação técnica de equipes**. Os artefatos são denominados MBM (*Module Based Matrix*) e TMM (*Team x Module Mapping*) para as respectivas recomendações.

Com base no relacionamento existente nos módulos de software, uma análise apenas com as equipes tecnicamente qualificadas deve ser feita de forma a encontrar um conjunto de alocações que favoreça a comunicação das equipes durante a implementação dos módulos, servindo de sugestão para o gerente de projeto.

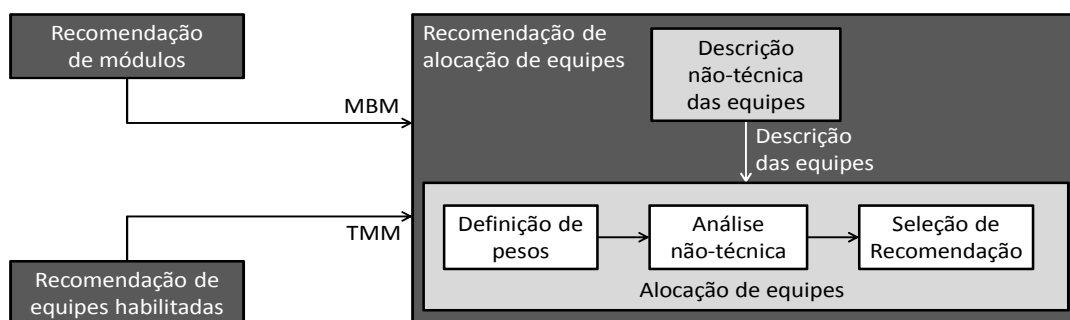
Para que a análise seja possível, um conjunto de atributos que ajudem nessa avaliação deve ser definido. Esses atributos devem ser fornecidos pela equipe ou gerados a partir de informações fornecidas por elas.

A **recomendação de alocação de equipes** é dividida em duas etapas, *descrição não-técnica das equipes* e *alocação de equipes*, como pode ser visto na Figura 7. Essa figura

ilustra o escopo desse trabalho, mostrando as sub-atividades referentes às duas etapas da recomendação que geram os artefatos de entrada (MBM e TMM).

As seções seguintes detalham a proposta de cada uma das etapas da recomendação de alocação de equipes: *Descrição não-técnica das equipes* e *Alocação de equipes*.

Figura 7 – Divisão de etapas da recomendação de alocação de equipes



5.1 Descrição não-técnica das equipes

A descrição não-técnica das equipes tem como objetivo caracterizar informações das equipes, que, a partir das mesmas, seja possível prever a viabilidade e a qualidade de comunicação entre as equipes geograficamente distribuídas. Neste trabalho, as entidades a serem alocadas, e logicamente a serem analisadas, são equipes de implementação de software. Contudo, não é parte deste trabalho a formação de equipes, mas sim a alocação das equipes existentes para a implementação dos módulos de software, daí a importância dos atributos das equipes.

As informações são caracterizadas na forma de atributos, que devem refletir não apenas a distância geográfica, mas diferentes tipos de características que causem impacto na comunicação requerida durante a fase de implementação. Para não causar problemas na distinção entre a descrição não-técnica e a descrição técnica (segunda parte do *framework* de recomendações), assume-se que as características não-técnicas são aquelas que são usadas para analisar os atributos de uma equipe com outra sob o ponto de vista de comunicação, enquanto que as características técnicas são aquelas que confrontam atributos de uma equipe com os requisitos técnicos necessários para a implementação dos módulos de software (SANTOS, PEREIRA, *et al.*, 2010).

Usando como base para a especificação dos atributos estudos relacionados ao DDS, (GUMM, 2006) e (AUDY e PRIKLADNICKI, 2007) descrevem 4 dimensões de fatores que afetam a comunicação: física (geográfica), organizacional, temporal e distância entre os envolvidos (*stakeholders*). Dentre elas, é excluída do escopo desta proposta a distância de

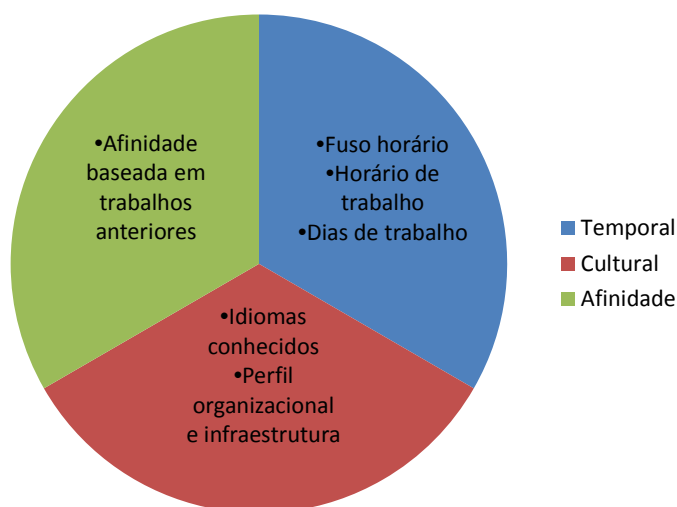
stakeholders, já que a alocação diz respeito à atividade de implementação dos módulos, ou seja, a atividade de elicitação de requisitos e modelagem já foram realizadas.

Além das já mencionadas dimensões física e temporal, (WARREN MCFARLAN, 1996) comenta que a dimensão cultural é mais uma diferença entre as equipes que pode afetar o desenvolvimento de software, principalmente no cenário global, com *outsourcing*. Já (LAMERSDORF e MÜNCH, 2010) acrescentam aos demais os fatores organizacionais e de infra-estrutura como pontos que afetam a comunicação. Além disso, (CHAVES, HUZITA, *et al.*, 2009) e (SILVA, 2007) utilizam a afinidade como uma forma de prever a proximidade entre as equipes.

No atual trabalho, para a descrição não-técnica das equipes, foram definidas três classes de atributos: **temporal**, **cultural**, e **afinidade**. Nesta classificação, consideramos que os atributos organizacionais e de infra-estrutura fazem parte dos atributos culturais. Cada uma dessas classes é composta por um conjunto de atributos que formam a descrição não-técnica da equipe.

O artefato gerado com a informação dos atributos não-técnicos é chamado *modelo de descrição não-técnica das equipes*, contendo os atributos **temporais** (*fuso-horário, horário de trabalho, dias semanais de trabalho*), **culturais** (*idiomas conhecidos e perfil organizacional e de infra-estrutura*) e de **afinidade** (*qualidade de comunicação com base em trabalhos anteriores*), como pode ser visto na Figura 8.

Figura 8 – Atributos não-técnicos



A forma de obtenção dos atributos é feita por meio de questionários *on-line* ou manuais, que devem ser preenchidos, parte pelos gerentes das equipes (ou líderes de equipes),

parte pelos membros das equipes. Alguns dos atributos podem ser obtidos com perguntas diretas, como é o caso dos atributos temporais. Já outros, como é o caso dos atributos culturais, devem ser obtidos através de perguntas que detectam algumas características das equipes que podem ser usadas para diferenciar culturas e características, uma vez que equipes com culturas semelhantes possuem maior possibilidade de se comunicar com mais facilidade (AUDY e PRIKLADNICKI, 2007). Por sua vez, os atributos de afinidade são obtidos na fase de avaliação de desempenho existente no *framework* de recomendações, apesar de possuir valores iniciais que são criados no momento da primeira alocação em que as equipes trabalham juntas.

Nas seções seguintes, cada classe de atributo é explicada separadamente, mostrando como os respectivos atributos são gerados a partir dos questionários.

5.1.1 Atributos temporais

Todo tipo de trabalho colaborativo necessita de comunicação. O fato de ser um trabalho colaborativo em que equipes estão geograficamente distribuídas faz com que essa atividade seja ainda mais crítica. Levando em consideração que equipes que possuem o mesmo horário de trabalho terão maior possibilidade de se comunicar (AUDY e PRIKLADNICKI, 2007), os atributos temporais podem ser vistos como atributos que verificam a disponibilidade. Esses atributos devem definir temporalmente o momento em que as equipes estarão em horário de trabalho. Assim, tornando possível identificar as equipes que possuem horário de trabalho semelhante, mesmo considerando as diferenças de fuso-horário.

Os atributos utilizados são o *fuso horário*, *horário de trabalho diário* e *dias de trabalho semanais*. Esses atributos são referentes à equipe como um todo, ou seja, subentende-se que todos os membros da uma equipe estão co-localizados e trabalhando simultaneamente.

O fuso horário deve estar no formato baseado no horário do meridiano de *Greenwich* (GMT). No caso de uma equipe na cidade de João Pessoa, Brasil, o fuso horário seria GMT-3, enquanto que uma equipe de Londres, Inglaterra, tem fuso horário GMT+0, ou seja, o horário padrão no meridiano. O comparativo do tempo de trabalho é realizado com base no mesmo fuso horário, o que quer dizer que, todos os horários levam em consideração o horário de trabalho em GMT (*Greenwich Mean Time*).

O preenchimento dos atributos é feito diretamente pelo gerente da equipe, respondendo perguntas diretas sobre os atributos temporais, como pode ser visto no formulário presente na Figura 9.

Figura 9 – Exemplo para formulário de atributos temporais

• Qual o fuso horário em que você se encontra? (Baseado em GMT)
 Por exemplo, no Brasil, a resposta seria -3.

☐ Segunda-feira
☐ Terça-feira
☐ Quarta-feira
☐ Quinta-feira
☐ Sexta-feira
☐ Sábado
☐ Domingo

• Qual o horário de início das atividades? (Hora e minutos)
 Exemplo: 8 : 00
 :

• Qual o horário final das atividades? (Hora e minutos)
 Exemplo: 12 : 00
 :

Adicionar

☐ Intervalo 1
☐ Intervalo 2

Remover

Nessa figura, o questionário é referente a um intervalo de tempo de trabalho e cada equipe pode cadastrar quantos intervalos de tempo forem necessários, sendo possível a consideração de qualquer pausa no tempo de trabalho diário. Por exemplo, caso uma equipe trabalhe das 8 as 12 pela manhã e das 14 as 18 pela tarde, são dois intervalos de tempo. Essa forma de preenchimento fornece a possibilidade de intervalos de tempo diferentes para cada dia da semana, como por exemplo, um turno reduzido no sábado ou domingo, ou até mesmo horários diferentes para cada dia da semana.

5.1.2 Atributos culturais

No conjunto de atributos culturais, as características que diferenciam culturas devem ficar evidentes no conjunto de respostas de dois questionários. Um questionário referente aos idiomas conhecidos pela equipe e um questionário referente às características que formarão um perfil organizacional e de infraestrutura da equipe. Esse perfil é formado com base em características sobre processos de software, modelos de maturidade, CSCW e alguns aspectos de filosofia organizacional.

As próximas seções detalham os dois questionários e como as suas respostas são quantificadas em valores matemáticos.

5.1.2.1 Questionário de idiomas

O questionário de idiomas deve ser aplicado a todos os membros da equipe, identificando os idiomas que são conhecidos pela equipe como um todo. Mesmo sabendo que apenas um

membro da equipe tem conhecimento sobre um dado idioma, essa informação é importante e deve ser incluída como característica da equipe. Por fim, a equipe possuirá um valor entre 0 e 1 para cada idioma, indicando o conhecimento da equipe no respectivo idioma.

O iTEP (*International Test of English Proficiency*), amplamente utilizado na Índia, que é um dos maiores centros de *outsourcing* do mundo (KOBITZSCH, ROMBACH e FELDMANN, 2001), utiliza em seu teste de avaliação de idioma uma divisão em cinco pontos: audição, leitura, gramática, escrita e fala (BridgeEnglish, 2012). Com base no iTEP, adota-se a verificação do idioma neste trabalho sob quatro pontos: audição, leitura, escrita e fala, já que foi assumido que a distinção da gramática já se faz presente nos demais pontos de avaliação. Sob esses quatro pontos, o nível de qualidade do idioma pode ser definido usando os níveis a seguir:

- | |
|--|
| <ol style="list-style-type: none">0. Nenhum1. Baixo2. Mediano3. Ótimo4. Nativo |
|--|

Note que os diferentes níveis são equidistantes, ou seja, a distância entre o nível baixo para o mediano é a mesma do nível ótimo para o nativo. Perceba que cada nível de idioma pode parecer fazer parte de uma escala ordinal, como se cada nível definisse uma classe de conhecimentos para idioma, não sendo clara a distância entre os níveis. Entretanto, isso não acontece por ser considerada a distância entre eles numericamente a mesma. Sendo assim, estamos tratando de uma escala métrica. Além disso, como possuímos um nível numericamente nulo na sua caracterização, que é o nível de conhecimento nenhum, a escala pode ser classificada como escala métrica de razão.

Sendo assim, toda equipe possuirá um conjunto $K = \{k_1; k_2; \dots; k_k\}$ de idiomas que incluem todos os idiomas que possuem ao menos um membro que tenha conhecimento no mesmo. A avaliação da equipe em cada idioma é feita com base nos quatro pontos de todos os n membros da equipe.

A avaliação de um idioma é definida pela média da avaliação dos pontos de todos os membros. Para que as avaliações das equipes se mantenham entre o intervalo $[0, 1]$, a

avaliação é racionalizada com base no número de pontos avaliados (4) e número de níveis possíveis (4, uma vez que o nível “Nenhum” assume o valor zero).

A avaliação de um idioma k para uma equipe x pode ser vista na Equação 1, levando em consideração os quatro pontos de avaliação. Cada avaliação $a_{aud_k}(m_i)$, $a_{lei_k}(m_i)$, $a_{esc_k}(m_i)$ e $a_{fal_k}(m_i)$ equivale a uma avaliação de um membro m_i em um idioma k , sabendo que existem n membros.

$$a_{id_k}(e_x) = \frac{\sum_{i=1}^n [a_{aud_k}(m_i) + a_{lei_k}(m_i) + a_{esc_k}(m_i) + a_{fal_k}(m_i)]}{n \times 4 \times 4} \quad (1)$$

Por exemplo, caso uma equipe possua conhecimento em três idiomas, o conjunto K irá conter três valores referentes à avaliação dos três idiomas, segundo a Equação 1.

5.1.2.2 Questionário de perfil organizacional

Diferentemente do questionário de idioma, o questionário que define o perfil organizacional é preenchido apenas pelo gerente da equipe. O perfil organizacional deve ser capaz de captar equipes com características organizacionais semelhantes, bem como favorecer equipes que possuam maior nível de qualidade nos fatores analisados.

O modelo de questionário foi elaborado baseado em estudos que apontam problemas causados pela diferença entre a estrutura organizacional e infraestrutura entre equipes distribuídas que trabalham colaborativamente (LAMERSDORF e MÜNCH, 2010). O questionário é composto por 10 perguntas, sendo três sobre processos de software, uma sobre modelo de maturidade, três sobre trabalho colaborativo apoiado por computador e três sobre filosofia organizacional.

As perguntas são classificadas como qualitativas e diferenciais. As perguntas 1, 2, 3, 5, 6 e 7 são consideradas como qualitativas, uma vez que é analisado não apenas a semelhança entre as equipes, mas também o nível de qualidade na qual a equipe se encontra, denotando qualidade às equipes que possuem os níveis mais altos. As perguntas 3, 8, 9 e 10 são consideradas diferenciais, pois, as diferentes opções de respostas não causam impacto na qualidade do software gerado, mas a semelhança das equipes nessas características facilita a comunicação durante o trabalho colaborativo.

As perguntas sobre processos de software incorporam três pontos importantes dos processos de software: os modelos de processo, a padronização dos processos e o ciclo de vida e integração.

Um processo de software é o conjunto de atividades que leva à produção de um produto de software (PRESSMAN, 2011). Dessa forma, cada processo possui seu conjunto de passos e atividades que são diferentes entre si. Entretanto, todo processo pode ser visto de forma abstrata e simplificada, mostrando apenas passos mais significativos, omitindo atividades de menor granularidade. Essa abstração é chamada de modelo de processo ou paradigma de processo (SOMMERVILLE, 2007).

Desde a criação da Engenharia de Software, quando iniciou o cuidado em se organizar o processo de desenvolvimento de software, modelos são criados e passam por evoluções que tornaram o desenvolvimento de software o que é hoje. O primeiro modelo de processo utilizado foi o **modelo sequencial ou cascata**, que, como o próprio nome sugere, passos são realizados sequencialmente de forma a só iniciar o próximo passo quando o anterior for finalizado. Os passos são até hoje utilizados, mesmo que em outros modelos de processo. São eles: Definição de requisitos, Projeto de software, Implementação e teste de unidade, Integração e teste de sistema e por fim, Operação e manutenção.

A evolução dos modelos de processo passou pela percepção de que poderiam ser necessários vários ciclos sequenciais para que um software estivesse em um nível aceitável, utilizando protótipos que são refinados ou melhorados até que o mesmo esteja finalizado. Com a necessidade cada vez mais urgente de que os softwares não levem anos para que fiquem prontos, os protótipos foram substituídos por versões simplificadas do sistema e já utilizados pelo cliente, produto este que passava por atualizações até a sua versão final. A este modelo se deu o nome de **modelo iterativo e incremental**.

Como o Desenvolvimento Baseado em Componentes (DBC), em que o software é modularizado em componentes de software reutilizáveis, a evolução do modelo de processo passou pela metáfora de **linhas de produto de software**, em que os componentes são feitos com o propósito de serem reutilizados, ou seja, é feito um desenvolvimento para reuso e depois um desenvolvimento que usufrui do reuso (VAN DER LINDEN, SCHMID e ROMMES, 2007).

Como o modelo de processo é uma abstração de um processo de software, mesmo identificando todos os processos utilizados pelas equipes, cada equipe possui uma forma

diferente de instanciar o processo e de realizar as atividades, ou seja, cada processo, por mais semelhante que seja, possui variações. Dessa forma, a flexibilidade da equipe para se adequar a processos de outras equipes será de fundamental importância para que elas consigam trabalhar de forma colaborativa (SANTOS, PEREIRA, *et al.*, 2010).

Outra característica dos processos de software a ser considerado nesse trabalho é o ciclo de vida e integração do processo. Mesmo com processos semelhantes, a forma de trabalho utilizada para integrar as partes implementadas pode ser diferente. A semelhança nessa forma seria de fundamental importância para que elas tenham interesse de integrar os módulos no mesmo momento.

As três primeiras perguntas do questionário podem ser vistas na Figura 10.

Figura 10 – Perfil organizacional – Processo de software

1 – Qual o modelo de processo de software adotado pela equipe? ☐ 0 - Não adota processos de software ☐ 1 - Processos de software com modelo sequencial (cascata) ☐ 2 - Processos de software com modelo iterativo e incremental ☐ 3 - Processos de linhas de produto de software 2 – Qual a flexibilidade da equipe para adaptação a diferentes processos de software? ☐ 0 - Não adota processo de software ☐ 1 - Apenas os processos padronizados pela equipe ☐ 2 - Aberto a adaptação de fases do processo padronizado pela equipe ☐ 3 - Aberto a adaptação a diferentes processos do mesmo modelo padronizado ☐ 4 - Aberto a adoção de diferentes processos de diferentes modelos 3 – Qual a flexibilidade da equipe para adaptação a mudanças do projeto (ciclo de vida de desenvolvimento e integração)? ☐ 1 - Desenvolvimento e integração sem adaptação e em iteração única (cascata) ☐ 2 - Desenvolvimento e integração com adaptação em iterações longas (métodos orientados a documentação) ☐ 3 - Desenvolvimento e integração com adaptação em iterações curtas (métodos ágeis)
--

Ao contrário das duas primeiras perguntas (perguntas qualitativas), a terceira pergunta é uma pergunta diferencial, uma vez que não existe diferença qualitativa na escolha da resposta, apenas uma diferenciação na forma de trabalho.

A quarta pergunta do questionário é referente a modelos de maturidade. Os modelos de maturidade foram criados com o objetivo de verificar o quanto uma equipe é capaz de executar os processos e repetir os passos que foram responsáveis por fazê-lo obter êxito. Equipes com modelos semelhantes apresentarão menores problemas para se comunicar acerca de artefatos, documentos e passos que remetem a qualidade do processo.

Foram usados como base os modelos de maturidade CMMI (SEI, 2010) e MPS.BR (SOFTEX, 2011). O CMMI possui 5 níveis de maturidade enquanto que o MPS.BR possui 7 níveis, sendo os 3 primeiros níveis do MPS.BR equivalentes ao primeiro nível do CMMI. Dessa forma, o MPS.BR possui uma evolução mais atenuada nos três primeiros níveis que o CMMI, porém, com as mesmas nuances na sua evolução. Essas características foram utilizadas de forma a caracterizar as equipes, mesmo que sem a adoção formal dos níveis dos modelos pelas equipes.

As primeiras atividades utilizadas nos modelos para se garantir a qualidade de um processo é o controle de cronogramas e das funcionalidades implementadas. Então, é verificada se existe a padronização dos documentos criados. Em seguida, o gerenciamento quantitativo do processo e dos produtos gerados deve ser feita, obtendo dados de onde estão os ganhos e perdas do processo de software. Nos últimos níveis, esses dados são utilizados para realizar melhorias no processo de forma contínua e sistemática, fazendo com que ele esteja em constante melhoria com base nas informações coletadas (SOFTEX, 2011) (SEI, 2010).

Na Figura 11 pode ser vista a quarta pergunta do questionário. Cada resposta ascendente assume que a atividade da resposta anterior está englobada.

Figura 11 – Perfil organizacional – Modelo de maturidade

- 4 – Qual o modelo de maturidade que a equipe se enquadra em processos de software?**
- ☐ 0 - Nenhum modelo de maturidade é adotado
 - ☐ 1 - Controle de cronograma e de funcionalidades
 - ☐ 2 - Padronização da documentação, validação e verificação
 - ☐ 3 - Gerenciamento quantitativo do processo e dos produtos
 - ☐ 4 - Melhoria contínua do processo com base em informações quantitativas

As três perguntas seguintes são sobre Trabalho Cooperativo Apoiado por Computador, em inglês *Computer-Supported Cooperative Work* (CSCW). CSCW é uma ciência multidisciplinar que estuda métodos e técnicas de trabalho em grupo auxiliadas por tecnologia e comunicação (MOECKEL, 2003).

Ciência esta que já foi conhecida como Sistema de Automação de Escritório, nela *Groupware* é definido como todo tipo de ferramenta que ajude a comunicação e a interação virtual, aumentando o desempenho e sendo adaptável às necessidades dos membros (MOECKEL, 2003).

Como na atual abordagem as equipes distribuídas realizarão trabalhos colaborativos, a utilização de ferramentas *groupware* será de fundamental importância para o êxito da comunicação e consequentemente do desenvolvimento distribuído de software.

Classificando as ferramentas para DDS, podemos dividi-las entre ferramentas para comunicação síncrona e ferramentas para comunicação assíncrona. Ambas são importantes para o DDS, sendo a primeira utilizada em tempo real enquanto que a segunda para armazenar informações que podem ser pesquisadas sem a necessidade de comunicação direta com algum envolvido no trabalho. Como exemplo de ferramentas síncronas, temos o *chat*, audioconferência e videoconferência. Para ferramentas assíncronas,aremos citar murais digitais, *blogs* e *wikis* (TRINDADE, 2008). A diferenciação entre cada um deles está ligada a quem modifica a informação, onde os murais digitais são atualizados por apenas uma equipe, um *blog* é atualizado também apenas por uma equipe, porém, oferece a possibilidade de comentários. Por fim, os *wikis* podem ser modificados por ambas as equipes de forma colaborativa. As ferramentas de comunicação assíncrona são importantes por fornecerem uma forma das equipes se comunicarem sem a necessidade de sincronização entre as mesmas.

Além disso, existem as formas de percepção do ambiente de trabalho (*awareness*) e as simulações de ambientes reais utilizando câmeras ao vivo entre as equipes (PRIKLADNICKI, 2010). Essas formas de trabalho também são utilizadas para melhorar a comunicação e o entrosamento entre equipes distantes fisicamente.

Ambientes com suporte a percepção podem utilizar ferramentas assíncronas, ou seja, que percebem o que foi realizado, mas não é possível a publicação em tempo real. Esse tipo de percepção auxilia na coordenação das atividades. Com suporte a percepção em tempo real, cada atividade que está sendo feita é notificada em tempo real e todos (de todas as equipes envolvidas) tomam conhecimento dessa informação (PINHEIRO, DE LIMA e BORGES, 2001). Na Figura 12, as perguntas 5, 6 e 7, que são perguntas qualitativas podem ser observadas, sobre trabalho cooperativo apoiado por computador.

Figura 12 – Perfil organizacional - CSCW

5 – Qual o tipo de ferramenta colaborativa mais avançada que é utilizada? ● 1 - Apenas e-mail e mensageiro de chat ● 2 - Audioconferência ● 3 - Videoconferência par-a-par ● 4 - Videoconferência em grupo ● 5 - Simulação do mesmo ambiente com vídeo 6 – Quais os tipos de suporte a percepção (awareness) são utilizadas? ● 0 - Nenhuma ferramenta de suporte a percepção ● 1 - Ferramenta de percepção assíncrona ● 2 - Ferramenta de percepção síncrona ● 3 - Ferramenta completa para ambas percepções assíncrona e síncrona 7 – Qual o tipo de ferramenta de compartilhamento de informação é utilizado? ● 0 - Nenhuma ● 1 - Mural ou portal de notícias ● 2 - Blog ● 3 - Wiki

Por fim, as três últimas perguntas são perguntas diferenciais. No caso de processos de software, a qualidade é levada em consideração pelo fato do modelo de linha de produtos ser uma evolução do iterativo, que por sua vez é uma evolução do sequencial. No caso das perguntas diferenciais, por exemplo, em uma pergunta sobre religião, a existência de crença não afeta a qualidade do software, mas pode causar impacto positivo ou negativo na comunicação entre os envolvidos em caso de respostas similares.

Além da já comentada religião, as perguntas englobam os assuntos de formalidade da equipe na comunicação e centralização da tomada de decisão da equipe. A formalidade é definida pelo ponto de vista de documentação, ou seja, se a equipe quer manter formalizados todos os pedidos recebidos e oferecidos de documentação ou se prioriza a agilidade, fornecendo a informação mais rapidamente. Perceba que essa formalidade não se trata de especificação formal de documentação e sim da formalidade das equipes para se requisitar e fornecer artefatos criados por elas.

Já a tomada de decisão atribui às equipes os extremos de serem equipes centralizadoras, ou seja, que o gerente é o responsável pelas tomadas de decisão, ou a descentralização, que opta pela tomada de decisão feita pelos membros do grupo.

As últimas perguntas do questionário podem ser vistas na Figura 13.

Figura 13 – Perfil organizacional – Perguntas diferenciais

- 8 – Crenças e religião podem dificultar a comunicação com outras equipes?**

 - 1 - Sim
 - 2 - Não

9 – Qual o grau de formalismo adotado na comunicação com outras equipes?

 - 1 - Deve ser formalizado e documentado
 - 2 - Pode ser informal para agilizar a troca de informação

10 – Qual o nível de centralização na tomada de decisões na equipe?

 - 1 - Centralizado (gerente da equipe é o responsável pela tomada de decisões)
 - 2 - Descentralizado (membros da equipe com autonomia na tomada de decisões)

O questionário é respondido pelo gerente da equipe. As respostas das perguntas qualitativas são convertidas em um valor numérico que se baseia na resposta obtida r e no valor máximo possível alt , utilizando a teoria dos conjuntos da lógica *fuzzy* (COX, 1995) e mantendo o valor numérico no intervalo $[0, 1]$. A proximidade do 0 mostra um perfil com menor padrão de qualidade, enquanto que mais próximo do 1 define uma métrica com mais qualidade.

Cada resposta das perguntas qualitativas têm um índice numéricos que variam de 0 a alt e algumas perguntas podem não conter o índice 0, o que faz com que os índices variem de 1 a alt . O valor de r para a resposta fornecida para a pergunta utilizará o índice numérico da resposta como valor r . Como já foi comentado a cerca dos níveis de idioma, os níveis aqui descritos também são em escala métrica, uma vez que a distância entre os diferentes níveis existe e é igual a 1. No caso das questões que existe o ponto nulo, que é a avaliação zero, trata-se de uma escala métrica de razão, no caso das demais que não possuem ponto nulo, trata-se de uma escala numérica intervalar. Lidar com os níveis dos questionários de forma numérica é uma forma de se gerar valores aproximados para as equipes, sendo possível a comparação numérica entre as equipes.

No caso da primeira pergunta, a primeira opção, “*Não adota processos de software*” assume matematicamente o índice zero, uma vez que essa resposta exprime a ausência completa do conhecimento ou uso de uma determinada característica. A diferença entre essas perguntas e as outras que não possuem zero como opção é que as demais não existe ausência de aptidão e sim um nível muito baixo da mesma. Como por exemplo na questão 5, onde a resposta de índice mais baixo é “Apenas e-mail e mensageiro de chat”. Essa resposta tem como índice o valor 1, já que não existe valor nulo como possibilidade de resposta. Já a variável alt sempre vai ter como valor o número do índice da opção mais alta dentre as

respostas possíveis. Na questão 1, temos $alt = 3$, mesmo com a existência de quatro respostas possíveis, porém, a primeira delas possui índice zero. No caso da questão 5, temos $alt = 5$.

Na Equação 2, a função matemática da avaliação do perfil organizacional para uma pergunta qualitativa q para uma equipe x pode ser vista.

$$a_{po_q}(e_x) = \frac{r}{alt} \quad (2)$$

Exemplificando com a pergunta 5, supondo que a equipe e_x obteve a resposta $r = 3$, “*Videoconferência par-a-par*”, teremos uma avaliação da pergunta qualitativa 5 igual a: $a_{po_5}(e_x) = 3/5 = 0,6$. Esse valor numérico representa um ponto de qualidade em que a equipe se encontra com base em sua resposta. Posteriormente, resultados de outras equipes a cerca dessa pergunta são utilizados e comparados de forma a verificar a semelhança entre as equipes para o processo de alocação. Equipes com valores semelhantes possuem respostas semelhantes e com nível de qualidade semelhante, o que é ideal para o DDS.

As perguntas diferenciais não denotam qualidade, então apenas a relação entre as equipes será analisada. Para tal, uma tabela que mostra as possibilidades de respostas deve ser feita com os valores numéricos associados a cada combinação. No caso da pergunta 8, por exemplo, as respostas possíveis são “sim” ou “não”. Logo, a tabela deve conter os valores numéricos que representem todas as combinações possíveis dessas respostas, que seriam “sim/sim”, “sim/não” (igual a “não/sim”) e “não/não”. Nesse caso, a combinação “não/não” retrata uma situação em que existe a menor possibilidade de ocorrência de problemas, enquanto que a combinação “sim/sim” apresenta o maior risco. Dessa forma, numericamente, a combinação “sim/sim” deve possuir um valor numérico pior que a combinação “não/não”. A combinação “sim/não”, por sua vez, apresenta uma situação de menor risco que a combinação “sim/sim”, mas não tão confortável como a combinação “não/não”, já que uma das equipes retrata que pode haver problemas com crenças e religião, o que significa dizer que ela deve ter um valor numérico intermediário entre o “sim/sim” e o “não/não”.

Para a definição dos valores numéricos para o questionário, os valores numéricos também se baseiam na teoria dos conjuntos de *logica fuzzy*, e a apresentação das tabelas é feita na seção 5.2.2.2, onde acontece o detalhamento da função objetivo do algoritmo genético.

Deve ser percebido que o questionário foi criado com base em problemas retratados na engenharia de software (PRESSMAN, 2011) (SOMMERVILLE, 2007) e no desenvolvimento distribuído de software (LAMERSDORF e MÜNCH, 2010) (CARMEL e PRIKLADNICKI, 2010) (PRIKLADNICKI e AUDY, 2006) (AUDY e PRIKLADNICKI, 2007). O modelo com perguntas qualitativas e perguntas diferenciais pode ser adaptado ou modificado para atender algum problema distinto dos que foram tratados nesse trabalho. Para tal, devem ser respeitados os requisitos das perguntas qualitativas definirem em cada índice uma resposta que demonstra maior aptidão ou conhecimento, e nas perguntas diferenciais seja criada uma tabela condicional das combinações de respostas entre pares de equipes.

5.1.3 Atributo de afinidade

Esse atributo representa a afinidade que uma equipe possui com outra em particular, de acordo com trabalhos passados. Esse atributo existe para cada par de equipes, sendo bidirecional, sabendo que a avaliação é realizada tanto de uma equipe e_x para uma equipe e_y como de uma equipe e_y para a equipe e_x . Dessa forma, a avaliação de afinidade $af_{e_x \rightarrow e_y}$ não necessariamente será igual à avaliação de $af_{e_y \rightarrow e_x}$.

O valor numérico para a afinidade também varia no intervalo $[0, 1]$. Como a afinidade é uma característica que é construída através do tempo, um valor inicial de afinidade deve ser adotado. Nesse trabalho, o ponto médio 0,5 será utilizado para equipes que não possuem trabalhos anteriores. Para a manutenção dos atributos de afinidade, após cada projeto realizado, os pares de equipes devem avaliar a comunicação com as outras equipes, avaliação esta realizada pelo gerente da equipe. Essa avaliação faz parte da etapa descrita no *framework* de recomendações como Avaliação de Desempenho das Equipes, que acontece após a implementação dos módulos. Valores que se aproximem do 1 mostram uma afinidade alta entre o par de equipes, enquanto que valores próximos do 0 mostram pouca afinidade.

Esse atributo é incrementado ou decrementado de acordo com a avaliação realizada por outra equipe. A avaliação de desempenho das equipes pode ser realizada de duas formas, utilizando uma avaliação *incremental* ou uma avaliação *absoluta*.

Ambos os métodos de avaliação de afinidade são baseados na escolha de um nível dentre cinco níveis de satisfação, sobre como foi o desempenho da outra equipe em relação à comunicação, representado pelo *feedback*. Os **níveis de satisfação** são *muito insatisfeito, insatisfeito, indiferente, satisfeito e muito satisfeito*.

Na avaliação absoluta, o nível selecionado é atribuído diretamente à afinidade, ou seja, o valor da afinidade torna-se igual ao valor definido como *feedback*. Também seguindo como base a teoria dos conjuntos de *fuzzy*, os valores numéricos identificados pelos cinco **níveis de satisfação** são respectivamente $\{0,0\}$, $\{0,333\}$, $\{0,5\}$, $\{0,667\}$ e $\{1\}$.

Já na avaliação incremental, o nível de satisfação selecionado é utilizado de forma a alterar gradativamente a afinidade entre as duas equipes de acordo com o *feedback* do gerente da equipe. O novo valor da afinidade entre uma equipe e_x para uma equipe e_y é definido com base na afinidade atual $af_{e_x \rightarrow e_y}$ e o *feedback* $fb_{e_x \rightarrow e_y}$ do gerente da equipe e_x , como pode ser visto na Equação 3. Para manipular a intensidade da modificação incremental, foram definidos dois pesos, w_{af} que define o peso da avaliação atual e w_{fb} que define o peso do *feedback* do gerente. Pesos maiores de w_{af} em relação a w_{fb} fazem com que a afinidade sofra uma mudança mais atenuada, enquanto que valores maiores de w_{fb} em relação a w_{af} fazem com que a mudança seja mais acentuada. Os valores utilizados como padrão por esse trabalho foram $\{w_{af} = 2\}$ e $\{w_{fb} = 1\}$. Essa escolha foi feita de forma a manter a afinidade anterior com maior peso que o *feedback*, mas ainda manter a mudança da afinidade acentuada.

$$af_{e_x \rightarrow e_y} = \frac{w_{af} \times af_{e_x \rightarrow e_y} + w_{fb} \times fb_{e_x \rightarrow e_y}}{w_{af} + w_{fb}} \quad (3)$$

Na avaliação absoluta, o novo valor da afinidade será um dos **níveis de satisfação** mencionados, não levando em consideração o valor anterior da afinidade entre as equipes, o que significa dizer que o peso w_{af} é considerado como zero, fazendo com que a função seja vista de forma simplificada como: $af_{e_x \rightarrow e_y} = fb_{e_x \rightarrow e_y}$

5.2 Alocação de equipes

A alocação de equipes é o momento em que as informações obtidas e processadas anteriormente serão utilizadas para gerar recomendações de alocação que auxiliarão o gerente de projeto a escolher a mais adequada ao propósito do projeto.

Essa fase utiliza os dois artefatos recebidos das recomendações anteriores, o MBM (*Matrix Based Matrix*) e o TMM (*Team x Module Mapping*). O MBM fornece a informação correspondente ao agrupamento de componentes em módulos, identificando relacionamentos

internos e relacionamentos externos em relação aos módulos de software. Esse artefato é representado por uma matriz quadrada no qual seu tamanho é igual ao número de módulos a serem alocados. Com exceção da diagonal principal, que define os relacionamentos internos aos módulos, cada posição da matriz define um relacionamento externo $re(m_i, m_j)$ entre um determinado módulo m_i e outro módulo m_j . Os valores se encontram no intervalo $[0, 1]$ e a ausência de relacionamento é representada pelo valor zero.

Um ponto importante a ser destacado é que um relacionamento externo $re(m_i, m_j)$ entre dois módulos não é igual ao relacionamento externo entre $re(m_j, m_i)$. Esse comportamento é caracterizado pelo acoplamento entre um componente c_i e um componente c_j não ser o mesmo que do componente c_j com um componente c_i . Entretanto, na análise de alocação, o uso dos relacionamentos externos é utilizado como proporção para perceber o quanto de comunicação será necessária para a implementação dos módulos. Sendo assim, a direção de quem requer a comunicação não é um fator importante, sabendo que independentemente disso, a comunicação será necessária.

Dessa forma, a partir desse ponto, sempre que for mencionado o relacionamento externo entre dois módulos m_i e m_j , a dependência entre esses módulos é definida pela média entre os dois relacionamentos externos $re(m_i, m_j)$ e $re(m_j, m_i)$, conforme definido na Equação 4.

$$d(m_i, m_j) = \frac{re(m_i, m_j) + re(m_j, m_i)}{2} \quad (4)$$

Na Figura 14 é possível ver um exemplo dos relacionamentos externos sendo convertidos em dependência entre os módulos. Os campos deixados em branco podem ser preenchidos com os valores espelhados, uma vez que a nova matriz é uma matriz simétrica, pois $d(m_i, m_j) = d(m_j, m_i)$.

Figura 14 – Exemplo de conversão de relacionamentos externos em dependência

	M1	M2	M3	M4
M1	1	0,209	0,215	0,210
M2	0,181	0,118	0,039	0,039
M3	0,200	0,039	0,130	0,039
M4	0,200	0,039	0,039	0,127



	M1	M2	M3	M4
M1	1	0,195	0,208	0,205
M2		0,118	0,039	0,039
M3			0,130	0,039
M4				0,127

O segundo artefato de entrada, denominado TMM, fornece a informação de quais são as equipes tecnicamente qualificadas para cada módulo, listando na ordem de adequabilidade técnica. Na **recomendação de alocação de equipes**, não é utilizado o valor numérico da adequabilidade técnica e sim a identificação de quais são as equipes que fazem parte do conjunto de equipes qualificadas para cada módulo. Por exemplo, segundo a Figura 15, todos os módulos que são listados na TMM devem possuir adequabilidade técnica suficiente.

Figura 15 – Team x Module Mapping (TMM)

	m ₁	m ₂	m ₃	m ₄	m ₅	m ₆	m ₇	m ₈	m ₉	m ₁₀
	e ₁₄	e ₁₂	e ₁	e ₄	e ₉	e ₂	e ₁	e ₉	e ₁₃	e ₁₂
	e ₁₁	e ₁	e ₁₉	e ₈	e ₅	e ₃	e ₉	e ₁	e ₁₀	e ₁₄
	e ₂	e ₅	e ₃	e ₁₀	e ₁₂	e ₁	e ₅	e ₈	e ₅	e ₁
	e ₅	e ₉	e ₆	e ₂	e ₁₀	e ₉	e ₄	e ₁₀	e ₂	e ₇
	e ₁₉	e ₁₀	e ₇	e ₁₉	e ₁₁	e ₂₀	e ₁₁	e ₁₃	e ₂₀	e ₁₁
	e ₃	e ₁₉	e ₂	e ₆	e ₄	e ₁₈	e ₁₂	e ₁₉	e ₃	e ₁₉
	e ₁	e ₁₁	e ₁₃	e ₃	e ₅	e ₈	e ₁₇	e ₂₀	e ₉	e ₂₀
	e ₄	e ₂	e ₄	e ₁₃	e ₇	e ₇	e ₁₃	e ₄	e ₂₁	e ₃
	e ₇		e ₉	e ₁₅	e ₁		e ₇	e ₇	e ₁₁	e ₁₈
	e ₁₂		e ₈	e ₁₁			e ₁₉		e ₁₄	e ₅
	e ₉			e ₅					e ₁	e ₉

A **alocação de equipes** é dividida em três etapas, a *definição de pesos*, *análise não-técnica* e *seleção de recomendação*. As seções seguintes detalham cada uma dessas etapas.

5.2.1 Definição de pesos

A definição de pesos é a etapa utilizada pelo gerente do projeto para adequar as suas prioridades para a análise não-técnica. Como mostrado na seção 5.1, os atributos não-técnicos são enquadrados em três tipos: temporais, culturais e de afinidade. Analogamente, a resolução do problema levará em consideração esses atributos de forma a avaliá-lo segundo três dimensões: avaliação temporal, avaliação cultural e avaliação de afinidade.

Nessa etapa de definição de pesos, uma ponderação para as avaliações temporal, cultural e de afinidade (respectivamente, w_t , w_c e w_a) devem ser definidos. A ausência da definição dos pesos manterá os pesos com valores iguais a um.

Nessa etapa também, os pesos usados para a afinidade na avaliação de desempenho das equipes são definidos (w_{af} e w_{fb}), conforme detalhado na seção 5.1.3. Além disso, a avaliação cultural é formada por uma avaliação de idioma e uma avaliação de perfil organizacional (w_{id} e w_{po}), sendo assim, pesos para definir o grau de importância de ambas as avaliações também devem ser definidos nessa fase do projeto.

5.2.2 Análise não-técnica

Com base nos artefatos de entrada MBM e TMM, a análise não-técnica tem o objetivo de encontrar soluções satisfatórias sobre o problema de alocação de equipes distribuídas. Em um modelo simplificado do problema, temos que a quantidade de soluções possíveis é igual ao número de equipes elevado ao número de módulos. Em um exemplo com 10 módulos e 15 equipes hábeis cada, existem 576.650.390.625 soluções possíveis. Para evitar a busca exponencial por todo o espaço de soluções, essa proposta adota a meta-heurística de algoritmos genéticos, considerada uma técnica de busca probabilística, não determinística e evolucionária, baseada na metáfora do processo biológico de evolução natural (LINDEN, 2008).

Para definir quais as soluções mais fortes, uma função objetivo é utilizada para avaliar matematicamente uma solução. Nesse caso, como o propósito do *framework* de recomendações é reduzir os problemas de comunicação durante a implementação do software, o algoritmo genético possui uma função objetivo que analisa os atributos não-técnicos a cada par de equipes que necessitam se comunicar, considerando que a necessidade de comunicação é diretamente relacionada com a dependência entre os módulos. Assim, a função objetivo se baseia nas avaliações temporal, cultural e de afinidade.

As próximas seções detalham o funcionamento do algoritmo genético na abordagem de recomendação de alocações. Para um melhor entendimento, o conteúdo será apresentado na seguinte sequência: representação das soluções e população inicial, função objetivo e operações genéticas.

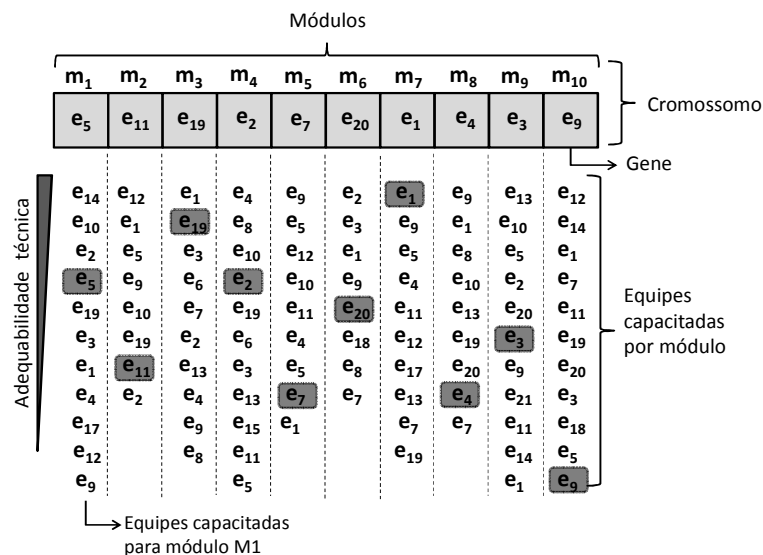
5.2.2.1 Representação das Soluções e População Inicial

Um cromossomo é formado por um conjunto de genes que o caracterizam. No caso proposto, os genes simbolizam o conjunto de módulos de software e seus alelos representam as equipes qualificadas a implementá-los, provenientes do artefato TMM. Dessa forma, a solução corresponde a seleção dos alelos para os genes.

Como mencionado anteriormente, o algoritmo genético deve possuir uma população para passar por operações genéticas e serem criadas novas soluções. Dessa forma, o primeiro passo é identificar uma população inicial baseada no artefato TMM, sendo também, realizada com base na aleatoriedade para selecionar as equipes qualificadas de forma imparcial. A aleatoriedade é utilizada com o intuito de manter a variedade de soluções da população (variedade genética) a maior possível, principalmente por se tratar da população inicial.

Na Figura 16 é possível visualizar um exemplo de cromossomo ao qual a caracterização dos genes são 10 módulos ($m_1, m_2, \dots, m_{10}$) e os alelos podem identificar um conjunto de equipes diferente por módulo. No caso de m_1 os alelos fazem parte do conjunto $\{e_{14}, e_{10}, e_2, e_5, e_{19}, e_3, e_1, e_4, e_{17}, e_{12}, e_9\}$. Como pode ser visto, as escolhas aleatórias de cada gene identifica a formação de um cromossomo, sendo assim, adicionado à população inicial.

Figura 16 – Representação e geração de um cromossomo para população inicial



5.2.2.2 Função objetivo

Nesta seção, será detalhada a função objetivo, que é a função responsável por avaliar a qualidade de uma solução ou cromossomo (DEB, 1999).

Diferentemente do que acontece na avaliação técnica da **recomendação técnica de equipes** e em outros trabalhos tradicionais de alocação, nesse trabalho, o que é observado e avaliado para a alocação são as dependências dos módulos e não seus requisitos técnicos. Isso acontece devido ao requisito de comunicação ser proporcional a dependência entre os módulos (GHEZZI, JAZAYERI e MANDRIOLI, 2002), ou seja, quanto maior for a dependência, maior será a necessidade de conversação durante o desenvolvimento do software, principalmente por serem equipes distribuídas (TREINEN e MILLER-FROST, 2006).

Como mencionado anteriormente, a função objetivo se baseia na utilização das três avaliações não-técnicas, que são as avaliações temporal, cultural e de afinidade. Cada uma dessas dimensões avalia a alocação sobre os atributos não-técnicos.

Dada uma solução de alocação s , as avaliações temporal, cultural e de afinidade resultam em valores no intervalo $[0, 1]$, representadas pelos termos $A_t(s)$, $A_c(s)$ e $A_a(s)$, respectivamente. Com o propósito de comparar a qualidade das soluções de alocação candidatas, estes termos são então aplicados a função objetivo $f(s)$, definida na Equação 5, onde w_t , w_c e w_a representam os pesos definidos anteriormente pelo gerente de projeto para as dimensões temporal, cultural e de afinidade, na etapa de *definição dos pesos*.

$$f(s) = \frac{w_t \times A_t(s) + w_c \times A_c(s) + w_a \times A_a(s)}{w_t + w_c + w_a} \quad (5)$$

Para uma dada solução de alocação s , a avaliação em uma dada dimensão x é calculada como uma média ponderada do nível de dependência entre cada par de módulos (m_i, m_j) e o valor da avaliação naquela dimensão entre as equipes (e_x, e_y) alocadas ao par de módulos (m_i, m_j). Formalmente, as avaliações temporal, cultural e de afinidade são calculadas usando uma expressão similar à indicada na Equação 6, porém substituindo os termos genéricos $A_x(s)$ e $a_x(e_x, e_y)$ por $A_t(s)$ e $a_t(e_x, e_y)$, $A_c(s)$ e $a_c(e_x, e_y)$, e $A_a(s)$ e $a_a(e_x, e_y)$, respectivamente. Note que, nesta equação, o termo $d(m_i, m_j)$ representa o nível de dependência entre os módulos m_i e m_j . Já o termo $a_x(e_x, e_y)$ representa o valor da avaliação da respectiva dimensão x (temporal, cultural ou afinidade) entre o par de equipes alocadas aos módulos m_i e m_j , representado pelas equipes e_x e e_y . Por fim, $p(e_i, e_j)$ representa uma

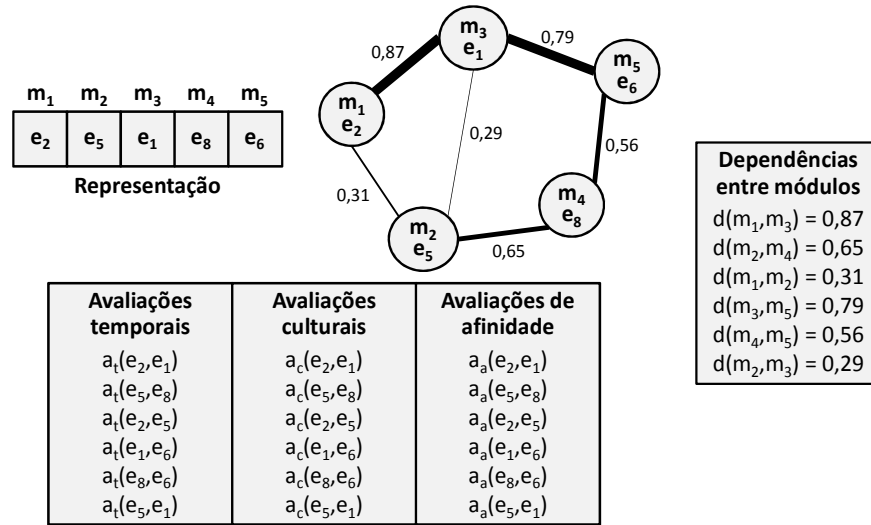
penalidade proporcional ao esforço de comunicação requerido das equipes e_x e e_y para implementar todos os módulos aos quais estão alocados.

$$A_x(s) = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n [d(m_i, m_j) \times a_x(e_i, e_j) \times p(e_i, e_j)]}{\sum_{i=1}^{n-1} \sum_{j=i+1}^n [d(m_i, m_j)]} \quad (6)$$

Para o cálculo dessa penalidade, que pode ser vista na Equação(7, o termo $d(m, m)$ representa a dependência interna do módulo m , pertencente ao conjunto M_{e_x} que simboliza o conjunto de módulos alocado à equipe e_x . Note que, quanto maior o número de módulos alocados às equipes, maior o esforço requerido, e, assim, menor a disponibilidade das equipes para comunicação com outras equipes. Sendo assim, maior será a penalidade das equipes em todos os aspectos analisados. Observe que, quanto maior a penalidade, menor o valor de $p(e_i, e_j)$. Essa função retrata esse comportamento uma vez que a alocação a muitos módulos fará com que o acúmulo de dependências internas deixe a penalidade alta.

$$p(e_x, e_y) = \left(1 - \frac{\sum_{i=1}^{m \in M_{e_x}} [d(m_i, m_i)] + \sum_{i=1}^{m \in M_{e_y}} [d(m_i, m_i)]}{2 \times \sum_{i=1}^{m \in M_U} [d(m_i, m_i)]} \right) \quad (7)$$

Na Figura 17, o exemplo de um cromossomo é mostrado, apresentando as dependências $d(m_i, m_j)$ e as avaliações $a_t(e_x, e_y)$, $a_c(e_x, e_y)$ e $a_a(e_x, e_y)$ que devem ser consideradas. Observe que, no caso do módulos m_1 e m_3 , alocados às equipes e_2 e e_1 , deve-se considerar as avaliações $a_t(e_2, e_1)$, $a_c(e_2, e_1)$ e $a_a(e_2, e_1)$.

Figura 17 – Exemplo de cromossomo com dependências entre módulos

Os valores das avaliações temporal $a_t(e_x, e_y)$, cultural $a_c(e_x, e_y)$ e de afinidade $a_a(e_x, e_y)$ entre o par de equipes alocado aos módulos m_i e m_j são mensuradas levando-se em consideração os atributos temporais, culturais e de afinidade destas equipes.

Considerando o par de equipes e_x e e_y , a avaliação temporal é calculada conforme a Equação 8. Observe que, a equação representa o percentual de horas de trabalho semanal que as equipes e_x e e_y compartilham simultaneamente em relação ao conjunto total de horas de trabalho semanal de todas as equipes candidatas, representado pelo termo $a_t(e_x, e_y)$, baseado no tempo do meridiano de Greenwich (GMT) para ajustar os fusos-horários. Quanto mais tempo simultâneo as equipes dispõem para se comunicar, maior a chance da comunicação ser efetiva (LAMERSDORF e MÜNCH, 2010).

$$a_t(e_x, e_y) = \left(\frac{T_{e_x} \cap T_{e_y}}{T_U} \right) \quad (8)$$

Já a avaliação cultural para um par de equipes e_x e e_y é calculada como uma relação entre a avaliação de todos os idiomas em comum às equipes $a_{id}(e_x, e_y)$ e da avaliação do perfil organizacional $a_{po}(e_x, e_y)$ entre elas.

A relação entre o idioma e o perfil organizacional, como foi mencionado anteriormente, possui pesos associados à análise de idioma e à análise de perfil organizacional. Como ambos têm efeito na relação cultural entre as equipes, o produto dos fatores retrata a relação direta que ambos possuem. Os pesos w_{id} e w_{po} funcionam de forma a ponderar a importância para o projeto de $a_{id}(e_x, e_y)$ e $a_{po}(e_x, e_y)$. Por se tratar de fatores que, quando baixos, causam um prejuízo muito alto ao projeto, quanto maior for o peso definido, maior é o prejuízo por uma avaliação ruim. A função pode ser vista na Equação 9.

$$a_c(e_x, e_y) = a_{id}(e_x, e_y)^{w_{id}} \times a_{po}(e_x, e_y)^{w_{po}} \quad (9)$$

Para calcular a avaliação de idioma $a_{id}(e_x, e_y)$, por sua vez, é necessário comparar os idiomas em comum entre as equipes e_x e e_y , verificando o nível de qualidade e também a discrepância entre as equipes, uma vez que equipes com níveis muito diferentes terão preconceitos e dificuldades maiores que equipes com níveis semelhantes (LAMERSDORF e MÜNCH, 2010).

Com base nisso, os idiomas em comum devem ser avaliados segundo a Equação 10. Essa fórmula leva em consideração a média para denotar qualidade e o complemento do desvio padrão para denotar a discrepância entre as equipes. Sabendo que as avaliações sempre variam no intervalo $[0, 1]$, o complemento do desvio padrão entre uma equipe e_x e uma equipe e_y é $[1 - \sigma_{id_k}(e_x, e_y)]$. Sendo assim, a avaliação de um idioma k entre duas equipes e_x e e_y é o produto entre a média μ_{id_k} e o complemento do desvio padrão σ_{id_k} .

$$a_{id_k}(e_x, e_y) = \mu_{id_k}(e_x, e_y) \times (1 - \sigma_{id_k}(e_x, e_y)) \quad (10)$$

Como podem existir vários idiomas em comum entre as equipes, temos que existe um conjunto $K_{e_x, e_y} = \{a_{id_{k1}}(e_x, e_y), a_{id_{k2}}(e_x, e_y), \dots, a_{id_{kn}}(e_x, e_y)\}$, onde $a_{id_{k1}}(e_x, e_y) \geq a_{id_{k2}}(e_x, e_y) \geq \dots \geq a_{id_{kn}}(e_x, e_y)$. O que significa um conjunto da avaliação de todos os idiomas em comum as duas equipes.

O idioma que obtiver a melhor avaliação, ou seja, o idioma $k1$, é o idioma principal para a comunicação entre as equipes e_x e e_y , sendo os outros idiomas utilizados para

melhorar a avaliação, uma vez que, quanto mais idiomas conhecidos maior chance de todos os membros se comunicarem corretamente. Para isso é utilizada uma função recursiva para os idiomas do conjunto K , de forma a, mantendo no intervalo $[0, 1]$, melhorar a avaliação a cada idioma. Essa melhoria é proporcional à pontuação dos idiomas seguintes. Outro ponto importante a se destacar é que com valores menores da avaliação do idioma principal, os próximos idiomas possuem maior importância na bonificação. Isso ocorre pelo fato do idioma principal não ser dominante, aumentando a importância de outros idiomas. Essa função pode ser vista na Equação 11. A função recursiva é detalhada, mostrando a função inicial $\beta_{id_{k_1}}$ e a função $\beta_{id_{k_n}}$. Essa última, por sua vez, é multiplicada por 0,25, limitando o aumento na avaliação de outros idiomas a esse valor, que é a distância matemática entre um nível de idioma para outro (baixo para mediano, ou mediano para avançado).

$$\begin{cases} a_{id}(e_x, e_y) = \beta_{id_{k_n}}(K_{e_x, e_y}) \\ \beta_{id_{k_n}}(K_{e_x, e_y}) = \beta_{id_{k_{n-1}}}(K_{e_x, e_y}) + \left[(1 - \beta_{id_{k_{n-1}}}(K_{e_x, e_y})) \times a_{id_{k_n}}(e_x, e_y) \times 0,25 \right] \\ \beta_{id_{k_1}}(K_{e_x, e_y}) = a_{id_{k_1}}(e_x, e_y) \end{cases} \quad (11)$$

Para finalizar a avaliação cultural $a_c(e_x, e_y)$, ainda nos resta a avaliação de perfil organizacional. Como foi detalhado na seção 5.1.2.2, existem dois grupos de perguntas, as qualitativas e as diferenciais. Todas as perguntas são utilizadas para se verificar a diferença nas características de cada equipe, entretanto, as perguntas qualitativas, além disso, definem qualidade, onde cada resposta de índice maior define uma qualidade maior da equipe do que uma equipe que responde uma pergunta com uma resposta de índice menor.

Como as perguntas qualitativas já foram transformadas em um valor numérico, os valores entre as equipes é comparado utilizando uma função semelhante à avaliação de idioma, com a média e o complemento do desvio padrão dos valores das duas equipes, considerando a qualidade e a semelhança entre as equipes, como pode ser visto na Equação 12. Essa função é utilizada para cada pergunta p entre as equipes. A média é responsável por identificar a qualidade das equipes, enquanto que o complemento do desvio padrão vai prejudicar equipes com respostas diferentes.

$$a_{pq_p}(e_x, e_y) = \mu_{pq_p}(e_x, e_y) \times (1 - \sigma_{pq_p}(e_x, e_y)) \quad (12)$$

Para as perguntas diferenciais, não existe a definição de qualidade. Durante o preenchimento do perfil não-técnico da equipe, o gerente da equipe responde às perguntas diferenciais. Ao contrário das perguntas qualitativas, as diferenciais não são convertidas em um valor numérico, mantendo-se para a análise não-técnica as respostas obtidas no questionário. Nesse momento em que as equipes serão comparadas para a obtenção da função objetivo, as respostas obtidas pelas duas equipes nas perguntas diferenciais são comparadas entre si e, com base na tabela de condicionais, os valores numéricos são obtidos.

As tabelas de condicionais contêm todas as possibilidades de combinação das perguntas diferenciais. No caso do modelo instanciado nesse trabalho, as questões 3 e 8 possuem uma tabela cada, enquanto que as questões 9 e 10, possuem uma mesma tabela, pois suas perguntas possuem comportamentos semelhantes. Essa última, por sua vez, avalia apenas duas possibilidades, o fato das respostas serem iguais ou diferentes. O fato delas serem afirmativas ou negativas não é importante, já que não denotam qualidade, mas apenas se são iguais. Assim, no caso de respostas iguais, é assumido o valor 1 e em caso de respostas diferentes, o valor 0, o que respeita o intervalo [0, 1] das avaliações.

No caso da questão 3, as opções de respostas sobre processo são: **“sequencial”**, **“baseado em documentação”** e **“ágil”**. Três critérios de comparação entre as três respostas foram escolhidos para criar os valores numéricos para as associações, são eles: (i) se o processo é iterativo; (ii) se o processo é documentado; e (iii) se ele tem um bom *time-to-market*. Como pode ser visto na Tabela 2, cada uma das opções de resposta foi então marcado com um S ou um N para cada critério existente ou não. Logo, as equipes que possuem mais pontos semelhantes possuem maior avaliação. É importante destacar que, por não ser uma pergunta qualitativa, nenhuma resposta é tida como melhor que outra, apenas é analisada as suas semelhanças. Caso ambas as respostas possuam um S, teremos um ponto de semelhança, assim como se ambas também possuam um N, uma vez que a ausência de uma característica também é uma semelhança.

Tabela 2 – Semelhança entre respostas da Questão 3

	Iterativo	Documentado	Time-to-market
Sequencial	Não	Sim	Não
Documentação	Sim	Sim	Sim
Ágil	Sim	Não	Sim

Com base nessas semelhanças, temos a tabela de condicionais para a Questão 3, como pode ser visto na Tabela 3:

Tabela 3 – Tabela de condicionais da Questão 3

Resposta Equipe 1	Resposta Equipe 2	Pontos em comum	Pontuação
Sequencial	Sequencial	3/3	1,0
Documentação	Documentação	3/3	1,0
Ágil	Ágil	3/3	1,0
Documentação	Ágil	2/3	0,667
Sequencial	Documentação	2/3	0,333
Sequencial	Ágil	0	0,0

As três primeiras opções da tabela são claras, uma vez que as duas equipes possuem a mesma resposta. Detalhando a opção “Documentação x Ágil”, temos que “Documentação” possui todas as características citadas, enquanto que “Ágil” possui as características de ser “iterativo” e possuir bom “time-to-market”. Isso faz com que essa combinação possua dois pontos em comum, que é a iteratividade e o *time-to-market*. Dessa forma, dois dos três

critérios de comparação estão presentes, que é o que faz com que a combinação tenha valor igual a 0,667.

Para a questão 8, sobre crenças e religião causarem problemas no ambiente de trabalho, temos como possíveis respostas a opção “sim” e “não”. Fazendo com que existam três comparações possíveis, como pode ser visto na Tabela 4.

Tabela 4 – Tabela de condicionais da Questão 8

Resposta Equipe 1	Resposta Equipe 2	Pontos em comum	Pontuação
Não	Não	2/2	1,0
Não	Sim	1/2	0,5
Sim	Sim	0	0,0

Foi utilizado como critério para diferenciação das respostas o percentual de equipes que pode suscitar algum problema devido a religião. Dessa forma, o percentual de equipes que selecionar a opção “sim”.

Por fim, para as perguntas 9 e 10, como já foi mencionado, apenas a semelhança dos resultados é importante, o que faz com que tenhamos apenas duas situações possíveis:

- Igual = 1
- Diferente = 0

Para que a avaliação do perfil organizacional seja obtida, cada valor das perguntas qualitativas $a_{pq_p}(e_x, e_y)$ e das perguntas diferenciais $a_{pd_p}(e_x, e_y)$ é somado e mantido no intervalo $[0, 1]$, como mostra a Equação 13.

$$a_{po}(e_x, e_y) = \frac{\sum_{i=1}^{i=p} [a_{pq_i}(e_x, e_y)] + \sum_{j=1}^{j=q} [a_{pd_j}(e_x, e_y)]}{p + q} \quad (13)$$

A avaliação de idioma e de perfil organizacional é utilizada para formação da avaliação cultural. É importante destacar que a avaliação de idioma de duas equipes só será considerada zero se não existir nenhum idioma em comum entre elas. Já o perfil organizacional nunca será zero por existirem perguntas que não possuem zero no nível de aptidão.

A última avaliação da função objetivo é a avaliação de afinidade. Conforme foi detalhado na seção 5.1.3, as afinidades são bidirecionais. Para a definição da avaliação de afinidade entre duas equipes $a_a(e_x, e_y)$, temos que o nível e a discrepância também são os pontos importantes a serem destacados, como está representado na Equação 14.

$$a_a(e_x, e_y) = \mu_a(af_{e_x \rightarrow e_y}, af_{e_y \rightarrow e_x}) \times (1 - \sigma_a(af_{e_x \rightarrow e_y}, af_{e_y \rightarrow e_x})) \quad (14)$$

Além disso, em caso de não existir valor de afinidade entre as equipes e_x e e_y , o valor padrão 0,5 será assumido, como mostra a Equação 15.

$$af_{0_{e_x \rightarrow e_y}} = af_{0_{e_y \rightarrow e_x}} = 0,5 \quad (15)$$

A partir de então, os valores seguintes de afinidade serão obtidos já com base na avaliação de desempenho, utilizando os valores da afinidade atual e o *feedback*. Também é importante destacar que, no caso de uma mesma equipe estar alocada a dois módulos dependentes, a avaliação de afinidade para uma equipe e_x será $a_a(e_x, e_x) = 1$, uma vez que é eliminada qualquer comunicação a distância. Porém, a penalidade da função objetivo é responsável por penalizar a equipe por excesso de volume de trabalho.

5.2.2.3 Operadores genéticos

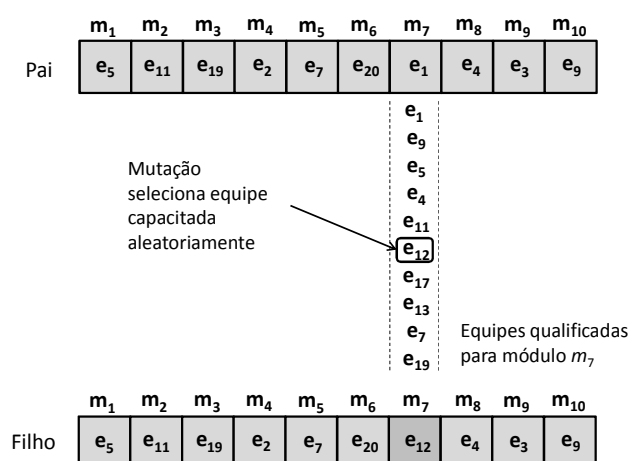
Os operadores se resumem ao cruzamento (*crossover*) e a mutação. Ambos ainda dependem, para o seu funcionamento, de uma forma de seleção de pais, já que o cruzamento depende de ao menos dois pais e a mutação de um cromossomo (SIVANANDAM e DEEPA, 2010).

Os métodos de seleção de pais utilizado é o método de torneio. O método de roleta viciada e de ranqueamento não foram utilizados devido às avaliações *fitness* não gerarem avaliações com valores muito discrepantes entre os cromossomos, fazendo com que a roleta funcione da mesma forma que o ranqueamento, com ponderações muito semelhantes. O uso do método de torneio também é significativamente mais rápido.

Segundo (LINDEN, 2008), o funcionamento do método de torneio sorteia um determinado número de cromossomos para participarem de um torneio. Nesse torneio, o que obtiver o maior valor de *fitness* será o cromossomo selecionado para a operação genética. Quando maior for o número de sorteios, mais favorecidos serão os cromossomos de maior avaliação.

Para o caso de alocação de equipes, como pode ser visto na Figura 18, a mutação representa a mudança da alocação de uma equipe por outra equipe que seja habilitada pelo TMM. Segundo o exemplo, o gene com módulo m_7 terá o alelo da equipe e_1 alterado para a equipe e_{12} .

Figura 18 – Exemplo de mutação



Para essa abordagem, o cruzamento utilizado é o de cruzamento uniforme, uma vez que os genes vizinhos não possuem uma relação direta, já que a relação é baseada no relacionamento entre os módulos, através da MBM. Como pode ser visto na Figura 19, o

cruzamento uniforme utiliza uma string binária que é definida aleatoriamente para cada reprodução sexuada.

Figura 19 – Exemplo de cruzamento uniforme

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
Primeiro pai	e_5	e_{11}	e_{19}	e_2	e_7	e_{20}	e_1	e_4	e_3	e_9
Segundo pai	e_3	e_{12}	e_1	e_7	e_6	e_2	e_{10}	e_{19}	e_4	e_{11}
Valores binários aleatório por gene										
	1	0	0	1	0	1	1	0	1	0
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9	m_{10}
Primeiro filho	e_3	e_{11}	e_{19}	e_7	e_7	e_2	e_{10}	e_4	e_4	e_9
Segundo filho	e_5	e_{12}	e_1	e_2	e_6	e_{20}	e_1	e_{19}	e_3	e_{11}

Os algoritmos genéticos têm a proposta de proliferar genes considerados como fortes, fazendo com que os filhos gerados em próximas gerações consigam obter melhores avaliações. Porém, a proliferação de um gene de uma solução na atual abordagem não é suficiente para levar para as próximas gerações as características que foram responsáveis pela boa avaliação do cromossomo, exatamente pela avaliação ser feita com base nos relacionamentos entre equipes. Por exemplo, considerando que um gene tem alocado uma equipe e_1 e outro gene tem uma equipe e_2 . Para que a retratação do relacionamento entre eles seja levada para próximas gerações da população, os dois genes devem ser levados juntos.

Caso fosse utilizado um cruzamento utilizado cortes (1, 2 ou n cortes), a proliferação dos genes continuaria ocorrendo normalmente, mas a proliferação das combinações de genes que formam um bom cromossomo seria prejudicada, principalmente no caso de genes que se situam distantes no cromossomo. Por exemplo, caso o cromossomo fosse considerado bom pelo motivo de uma combinação entre as equipes responsáveis pelo módulo m_1 e o módulo m_{10} , no caso de cruzamento com 1 corte, nunca existiria um corte que mantivessem os genes m_1 e m_{10} preservados para um filho.

Esse comportamento não acontece no caso de cruzamento uniforme, já que cada gene possui uma mesma probabilidade de ser obtido de um pai ou de outro.

Os operadores de cruzamento e mutação são utilizados em todas as iterações do algoritmo genético. A taxa de cruzamento utilizada nas execuções realizadas é de 60%,

fazendo com que esse percentual do total de cromossomos da população sofra cruzamento (LINDEN, 2008). Por outro lado, o operador de mutação adota taxa de 40%.

Para que a taxa de mutação cresça no fim da execução e a taxa de cruzamento se reduza na mesma proporção, é utilizada uma função linear de forma a inverter os valores gradativamente, resultando na taxa de cruzamento de 40% e de mutação de 60% na última iteração. Esse comportamento tem o objetivo de reduzir a convergência genética no momento em que a diversidade está reduzindo, aumentando a probabilidade de se encontrar uma solução ótima. A Equação 16 pode ser vista a seguir, identificando o comportamento da função linear, de acordo com o número de iterações do algoritmo.

$$T_{Cross} = [0,3 \times (it_n - it_i) / it_n] + 0,4 \quad (16)$$

A taxa de mutação é sempre o complemento da taxa de cruzamento, fazendo com que a soma de ambas seja sempre 1.

5.2.3 Seleção de recomendação

Uma vez que as soluções de alocação são identificadas e recomendadas pelo algoritmo genético, o passo denominado **alocação das equipes** é realizado pelo gerente de projetos com o objetivo de selecionar a solução de alocação que melhor se adéqua aos interesses do projeto.

Um ponto importante a ser destacado é que, no algoritmo proposto, a função objetivo $f(s)$ avalia a qualidade das soluções de forma integrada em termos dos atributos temporais, culturais e de afinidade das equipes. No entanto, como pode ser visto na Equação 5, as avaliações temporal, cultural e de afinidade são calculadas de forma independente. Consequentemente, caso seja de interesse do gerente de projeto, o algoritmo pode ser facilmente adaptado para identificar soluções *pareto-ótimas*, nas quais as melhores soluções em cada dimensão são selecionadas, sem, no entanto, impor a noção de melhor solução. Uma solução *pareto-ótima* é melhor que outra quando é melhor em pelo menos uma das dimensões e não é pior em relação as outras dimensões. Por exemplo, uma solução que possui baixa avaliação temporal e cultural, mas possui a melhor avaliação de afinidade dentre todas as soluções é considerada uma solução *pareto-ótima*. Note que soluções *pareto-ótimas* permitem ao gerente conhecer a influência das equipes em cada dimensão.

5.3 Considerações finais

Neste capítulo, a descrição formal da abordagem para alocação de equipes foi realizada, dividindo em dois pontos, a **Descrição não-técnica das equipes** e a **Alocação de equipes**.

A descrição não-técnica identifica os atributos não-técnicos, bem como o detalhamento de como as equipes se caracterizam a cerca dos mesmos. Os atributos não-técnicos foram distribuídos em três aspectos: temporal, cultural e de afinidade. A caracterização dos atributos para as equipes é realizada com base em questionários que manipulam as informações fornecidas de forma a comparar, em outro momento, equipes que necessitam se comunicar. Os atributos temporais correspondem ao fuso-horário e os intervalos de tempo de trabalho na qual a equipe se encontra. Os atributos culturais são caracterizados pelos idiomas e um perfil organizacional e infraestrutura, perfil este que contém características relativas a processos de software, modelos de maturidade, ferramentas colaborativas síncronas e assíncronas, *awareness* e filosofias organizacionais, como religião, tomada de decisão e formalidade na comunicação. Para os atributos de afinidade, existe um valor bidirecional para cada par de equipes que representa a afinidade entre aquele par, fazendo com que equipes que já trabalham bem anteriormente, sejam favorecidas em alocações futuras.

A alocação de equipes é dividida em três passos. A *definição de pesos* é responsável por ponderar os pesos das variáveis não-técnicas a serem consideradas, como os aspectos temporal, cultural e afinidade, além dos pesos relativos ao idioma e perfil organizacional e infraestrutura. Além disso, por conta da fase de retroalimentação do *framework* de recomendações, os pesos referentes a afinidade e o *feedback* fornecido pelos gerentes das equipes também devem ser definidos. O segundo passo da alocação de equipes, a análise não-técnica é onde as funções utilizadas no algoritmo genético são definidas, identificando as funções temporal, cultural e de afinidade, bem como as funções internas da função cultural, para idioma e perfil organizacional. Além disso, é realizado um detalhamento do algoritmo genético e suas operações genéticas. Por fim, o capítulo também apresenta o passo de *seleção de recomendação de equipes*, no qual o gerente do projeto é responsável por selecionar, dentre as alocações oferecidas pelo algoritmo genético, qual a que mais se adequa ao seu projeto. Com isso, seu conhecimento e experiência não é desperdiçado e ainda é melhor aproveitado, uma vez que as alocações oferecidas já foram avaliadas segundo os atributos não-técnicos e lidam apenas com equipes tecnicamente qualificadas.

Capítulo 6

Prova de Conceito

Este capítulo tem como objetivo avaliar a aplicação dos conceitos definidos nesse trabalho, aplicando a abordagem proposta em exemplos fictícios e reais.

Como o framework de recomendações já possui a fase de recomendação de módulos concluída e avaliada, os resultados obtidos dessa fase são utilizados para a atual abordagem. São casos reais de projetos pequenos, que torna viável a comparação de um algoritmo genético, com um algoritmo aleatório e um algoritmo de busca exaustiva. Além disso, uma prova de conceito com um projeto fictício será utilizada de forma a explorar um caso em que o universo de busca é maior que os casos validados, se aproximando mais de um caso global.

Nas próximas seções serão detalhados os problemas reais e fictícios a serem usados na avaliação, os parâmetros e configurações assumidas no algoritmo genético utilizado, a instanciamento de equipes fictícias baseadas em informações reais obtidas por meio de referências, os resultados da função objetivo e, por fim, os resultados dos algoritmos genético, aleatório e exaustivo.

6.1 Problemas reais e fictício

Nos estudos para avaliação do framework de recomendações, três arquiteturas foram analisadas. São elas: MultiMOM (BEZERRA, 2010), GingaCC (SOARES, MORENO e C., 2010) e SPL e-commerce. Os dois primeiros são projetos reais e implementados, enquanto que o SPL e-commerce é uma linha de produto utilizada academicamente no livro “Software Product Line in Action” (VAN DER LINDEN, SCHMID e ROMMES, 2007).

O quarto e último problema a ser utilizado para a prova de conceito é fictício, projetado com o objetivo de validar problemas com espaço de busca bem maior que os analisados em fases anteriores do framework. Para esse último, nomeamos de BigExample.

6.1.1 MultiMOM

O Multi-MOM consiste numa linha de produtos de *middleware* para computação móvel que está essencialmente focada na funcionalidade de comunicação.

Ela é composta por 11 componentes e após a recomendação de módulos, temos que sua nova visão da arquitetura possui 3 módulos e 3 relacionamentos externos aos módulos, como pode ser visto na Tabela 5.

Tabela 5 – MBM de relacionamentos do MultiMOM

	m1	m2	m3
m1	1	0,12	0,06
m2	0,6	0,32	0
m3	0	0	0,18

Como existem os relacionamentos externos $re(m_1, m_2)$ e $re(m_2, m_1)$, após a conversão da MBM original para uma MBM simétrica de dependências, temos duas dependências entre módulos, como mostra a Tabela 6.

Tabela 6 – MBM de dependências do MultiMOM

	m1	m2	m3
m1	1	0,36	0,03
m2		0,32	0
m3			0,18

A Tabela 6 forma o conjunto de dependências utilizado na análise não-técnica do MultiMOM.

6.1.2 GingaCC

O Ginga é a especificação de *middleware* para aplicações de TV digital do SBTVD (Sistema Brasileiro de TV Digital Terrestre) e sua arquitetura é composta por diversos ambientes, sendo o Ginga-CC (*Common Core*) quem oferece um conjunto de serviços de TV Digital comuns aos outros ambientes. A arquitetura do Ginga-CC passa por um processo de refatoração para uma arquitetura de LPS. Esse modelo refatorado, que possui 11 componentes, é que foi utilizado pela recomendação de módulos do *framework* de recomendações.

O GingaCC possui 4 módulos formados na recomendação de módulos e 6 relacionamentos externos, como mostra a Tabela 7.

Tabela 7 – MBM de relacionamentos do GingaCC

	m1	m2	m3	m4
m1	1	0	0,14	0,13
m2	0,07	0,41	0,07	0
m3	0,11	0	0,05	0
m4	0	0	0,05	0,27

Após a conversão para MBM simétrica de dependências, temos 5 dependências, como mostra a Tabela 8.

Tabela 8 – MBM de dependências do GingaCC

	m1	m2	m3	m4
m1	1	0,035	0,125	0,065
m2		0,41	0,035	0
m3			0,05	0,025
m4				0,27

6.1.3 SPL e-commerce

A linha de produto de comércio eletrônico consiste em uma LPS altamente distribuída baseada na *World Wide Web* que controla sistemas B2B (*business-to-business*) e B2C (*business-to-consumer*) (VAN DER LINDEN, SCHMID e ROMMES, 2007). Trata-se de uma linha de produtos desenvolvida segundo um processo de engenharia reversa, que conta com 11 componentes.

Após a recomendação de módulos do *framework* de recomendações, o SPL e-commerce possui 3 módulos e 6 relacionamentos externos, ou seja, possui relacionamentos entre todos os módulos, como mostra a Tabela 9.

Tabela 9 – MBM de relacionamentos da SPL e-commerce

	m1	m2	m3
m1	1	0,42007	0,21581
m2	0,38188	0,32416	0,07815
m3	0,20071	0,07815	0,13055

A seguir, na Tabela 10, temos a MBM simétrica de dependências, retratando as 3 dependências.

Tabela 10 – MBM de dependências da SPL e-commerce

	m1	m2	m3
m1	1	0,400975	0,20826
m2		0,32416	0,07815
m3			0,13055

6.1.4 BigExample

Para o BigExample, foi criado um problema com 15 módulos e 30 relacionamentos externos colocados aleatoriamente, além dos 15 relacionamentos internos. Para simular a abordagem adotada na fase de recomendação de módulos e consequentemente manter o problema semelhante a um problema real, um dos módulos deve possuir relacionamento 1, seja interno ou externo, resultado este que é uma consequência da recomendação de módulos, por conta da normalização adotada naquela fase do *framework* de recomendações.

Dos 30 relacionamentos externos, 9 deles formaram um relacionamento bidirecional, o que é bastante comum na recomendação de módulos. A MBM do BigExample pode ser vista na Tabela 11, mostrando os relacionamentos externos e internos do problema fictício.

Tabela 11 – MBM de relacionamentos do BigExample

	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15
m1	0,11	0,48	0,32												
m2	0,16	0,8		0,38	0,17										
m3	0,2		0,23	0,19											
m4				0,09		0,71						0,2			
m5		0,37			0,32	0,65	0,24								
m6				0,09	0,2	0,26		0,45			0,22				
m7							1	0,07							
m8						0,3	0,34	0,99							
m9						0,39			0,68			0,2		0,31	
m10						0,47				0,42					0,19
m11						0,37					0,05				
m12												0,49		0,21	
m13									0,8				0,22		
m14									0,3				0,59	0,37	
m15											0,2				0,27

Como resultado da conversão da MBM para uma MBM simétrica de dependências, que pode ser vista na Tabela 12, temos 21 dependências entre módulos.

Tabela 12 – MBM de dependências do BigExample

	m1	m2	m3	m4	m5	m6	m7	m8	m9	m10	m11	m12	m13	m14	m15
m1	0,11	0,32	0,26												
m2		0,8		0,19	0,27										
m3			0,23	0,1											
m4				0,09		0,4						0,1			
m5					0,32	0,42	0,12								
m6						0,26		0,37	0,19	0,24	0,29				
m7							1	0,2							
m8								0,99							
m9									0,68			0,2	0,4	0,31	
m10										0,42					0,09
m11											0,05				0,1
m12												0,49		0,1	
m13													0,22	0,29	
m14														0,37	
m15															0,27

6.2 Instanciação de equipes

Para que a prova de conceito seja realizada corretamente, é necessário um conjunto de equipes de vários locais do mundo e que possuam os questionários de idioma e de perfil organizacionais preenchidos, bem como os horários de trabalho da equipe e as afinidades entre eles.

Para tornar o cenário o mais próximo possível da realidade, foram criadas 20 equipes de 16 países diferentes, utilizando os 16 maiores países da indústria de software (VELOSO, BOTELHO, *et al.*, 2003). São eles: Estados Unidos, Japão, Alemanha, Inglaterra, Índia, Brasil, Coreia do Sul, Irlanda, China, Espanha, Taiwan, Israel, Finlândia, Singapura, Argentina e México. Países como Brasil, Índia, Estados Unidos e China possuem duas equipes cada, totalizando as 20 equipes. Essa escolha foi feita com base na quantidade de *outsourcing* realizada por esses países no mundo (VELOSO, BOTELHO, *et al.*, 2003) (KOBITZSCH, ROMBACH e FELDMANN, 2001) (HERBSLEB e MOITRA, 2001).

Assim, para definir os idiomas conhecidos pelas equipes, foi identificado o idioma nativo de cada equipe e os idiomas estrangeiros mais falados em cada país, com base em informações do censo ou pesquisas oficiais (COMMISSION, 2006) (VISTAWIDE, 2006) (M. S. THIRUMALAI, 2002) (US, 2003) (DIRECTORY, 2012) (TIMES, 2004) (NOI, 2004) (INDIAGOVERNMENT, 2011).

Em algumas dessas fontes, também está presente o percentual da população que fala cada idioma. Para os países que não possuem esse percentual (Japão, Índia, Singapura, Argentina, México e Brasil), foi utilizado um percentual padrão que se baseia nos percentuais dos países europeus, por ser a fonte mais completa e com a maior quantidade de países. Para os países que possuem o Inglês como idioma principal, foi percebido que os idiomas secundários são menos difundidos, assim o percentual foi baseado na Inglaterra, com valores 30%, 20% e 10% para os três idiomas estrangeiros mais falados no país. No caso de equipes de idioma nativo diferente de Inglês, todas possuíam o Inglês como o primeiro idioma estrangeiro, por ser um dos idiomas mais falados no mundo. Nesses casos, os idiomas estrangeiros foram definidos com os percentuais de 50% (inglês), 20% e 10% para os três melhores idiomas estrangeiros, respectivamente. Note que os percentuais foram arredondados para simplificar a criação das equipes.

Para cada equipe foram criados 10 membros, cada qual possui o idioma nativo com avaliação máxima (Nativo – 1,00). Para os idiomas estrangeiros, o percentual da população que fala cada idioma é utilizado para definir quantos membros falam o determinado idioma. Assim, os membros que falam o idioma assumem a avaliação como “Avançado” (0,75) para o percentual definido. Por exemplo, no caso da equipe do Japão, que possui idioma principal Japonês e o único idioma secundário como Inglês, 50% da população tem conhecimento em Inglês, dessa forma, cinco membros da equipe possuem avaliação “Avançado” para Inglês, enquanto que os outros membros desconhecem o idioma, considerando a avaliação como “Nenhum” (0). Assim, a avaliação do idioma Inglês dessa equipe é 0,375. O percentual da população do Japão que fala Inglês é de 50%, o que poderia, intuitivamente, indicar que a avaliação da equipe fosse de 0,5 e não 0,375. Porém, esse é um idioma estrangeiro, não sendo comum o nível de conhecimento como “Nativo”, sendo este o motivo da avaliação de idioma estrangeiro ser assumido como “Avançado”.

Cada equipe possui no máximo quatro idiomas, contando com o nativo, porém, alguns países possuem menos idiomas, como é o caso do Japão, que possui apenas Japonês e Inglês. As tabelas com as informações das equipes e dos seus idiomas, bem como a avaliação de cada membro, podem ser encontradas no Apêndice A desse trabalho.

Para os questionários de perfil organizacional, por sua vez, foram realizados sorteios para definir as respostas para as perguntas. Além disso, o sorteio também respeita as condicionais de impossibilidade, como por exemplo, dizer em uma questão que não utiliza

processo de software e em outra que o processo é ágil. A Figura 20 mostra as condicionais respeitadas.

Figura 20 – Condicionais do questionário de perfil organizacional

– Se (Q1 == 0)
• Q2 = 0
• Q4 = 0
– Se (Q1 == 1)
• Q2 != 0
• Q3 = 1
– Se (Q1 > 1)
• Q2 != 0
• Q3 != 1

Sendo assim, em caso de na questão 1 ser sorteada a opção 0 (“Não adota processos de software”), as questões 2 e 4 devem ser marcadas com a opção 0, que denotam respectivamente as respostas: “Não adota processos de software” e “Nenhum modelo de maturidade é adotado”. No caso da primeira questão ter sorteado a opção 1 (“Modelo cascata”), a segunda questão deve ser diferente de 0 e a terceira questão deve ser igual a 1 (“Iteração única – cascata”). Por fim, caso a questão 1 tivesse respostas diferentes de 0 e 1, as questões 2 e 3 deveriam ter respostas diferentes de 0 e 1, respectivamente.

Além disso, os perfis organizacionais dos questionários não podem ser sorteados de forma aleatória, uma vez que algumas características são mais comuns que outras. No caso das perguntas 1, 4 e 5, foram definidos percentuais de maior probabilidade de sorteio para algumas respostas. Esses percentuais foram definidos arbitrariamente de acordo com as tendências esperadas do mercado de software nos próximos anos. Na Figura 21 podem ser vistas as probabilidades das questões citadas.

Figura 21 – Probabilidade de respostas do questionário

1 Qual o modelo de processo de software adotado pela equipe?	
10%	0 Não adota processos de software
20%	1 Processos de software com modelo sequencial (cascata)
40%	2 Processos de software com modelo iterativo e incremental
30%	3 Processos de linhas de produto de software
4 Qual o modelo de maturidade que a equipe se enquadra em processos de software?	
10%	0 Nenhum modelo de maturidade é adotado
40%	1 Controle de cronograma e de funcionalidades
30%	2 Padronização da documentação, validação e verificação
20%	3 Gerenciamento quantitativa do processo e dos produtos
10%	4 Melhoria contínua do processo com base em informações quantitativas
5 Qual o tipo de ferramenta colaborativa mais avançada que é utilizada?	
11%	1 Apenas e-mail e mensageiro de chat
22%	2 Audioconferência
34%	3 Videoconferência par-a-par
22%	4 Videoconferência em grupo
11%	5 Simulação do mesmo ambiente com vídeo

Os valores foram colocados da seguinte forma: pontos foram divididos pontuando a possibilidade de ser sorteada cada resposta, então, o total de 100% foi dividido por essa pontuação. Na primeira pergunta, por exemplo, foram definidos quatro pontos para “iterativo e incremental”, três para “linhas de produto de software”, dois para “sequencial” e um para “nenhum processo”. Como “iterativo e incremental” tinha 4 pontos e a soma dos pontos distribuídos entre as respostas foram 10, temos 4/10, definindo o percentual como 40%.

Como foi mencionada na Fundamentação Teórica desse trabalho, a evolução de Engenharia de Software levou a existência das LPS, entretanto, sua ponderação foi colocada menor que o modelo iterativo e incremental, pois as LPS ainda estão em processo de amadurecimento e expansão. Os modelos sequenciais também devem ser considerados, porém em uma menor proporção que os iterativos, enquanto que a ausência de modelos é mais raramente encontrada, mas ainda assim deve ser considerada no modelo.

Na questão 4 (Modelo de Maturidade), apesar de se esperar que a maturidade de processo cresça com o passar do tempo, atualmente, os níveis mais elevados ainda não são tão facilmente encontrados, principalmente se vendo pelo ponto de vista do DDS. Dessa forma, o nível “nenhum” e o mais alto foram definidos com um ponto, enquanto que os outros foram ponderados de forma decrescente do nível um ao três, com valores respectivamente de quatro, três e dois.

Na questão 5 (Ferramenta colaborativa), o uso das ferramentas de comunicação síncrona também devem possuir valores diferentes dos completamente aleatórios. As ferramentas de videoconferência par-a-par são esperadas por serem as mais comuns, devido ao amadurecimento da Internet e a globalização das ferramentas. Abaixo dela, foram

colocadas a videoconferência em grupo e a audioconferência. E por fim, Simulação de ambiente *sametime* e e-mail e mensageiro de chat foram colocados com a menor probabilidade de sorteio. Lembrando que, por se tratar de uma pergunta qualitativa, as respostas mais bem qualificadas incluem as características das respostas de menor nível.

As demais perguntas não possuem características tão diferentes entre as respostas, ou seja, se espera que as respostas no mundo real tenham percentual semelhante entre todas as respostas possíveis.

Para a definição das afinidades, para que houvessem afinidades diferentes do valor inicial, as 15 primeiras equipes foram aleatoriamente sorteadas entre si, de forma a simular que tenha existido um trabalho anterior. Apenas equipes que possuem idiomas em comum foram sorteadas para a afinidade. Diante disso, os valores do *feedback* dado de uma equipe a outra foi sorteado aleatoriamente dentro dos 5 níveis possíveis, também limitando que a distância entre suas avaliações fossem de no máximo dois níveis (“Muito insatisfeito” e “Indiferente”, por exemplo). As afinidades entre as equipes podem ser vistas no Apêndice B.

Todas as equipes possuem o fuso horário de seu país. Os dias de trabalho foram colocados de segunda-feira a sexta-feira, turno integral, enquanto que no sábado apenas um turno matutino. Para os horários de trabalho foram definidos valores a serem distribuídos entre as equipes. De todas as equipes, 12 possuem horário de trabalho das 8h as 12h e das 14h as 18h. Cinco equipes trabalham das 9h as 13h e das 15h as 19h. As últimas três equipes possuem apenas um turno de trabalho, sendo duas delas das 8h as 14h, e a última das 10h as 18h. Perceba que todos esses horários mencionados são os horários locais de seus países, não convertidos com o fuso horário, conversão esta que é feita apenas durante a análise temporal das equipes. Uma vez definidos os horários, eles foram aleatoriamente distribuídos entre as equipes criadas.

Os perfis completos das equipes, com os atributos temporais e culturais podem ser encontrados no Apêndice A.

6.3 Parâmetros e características do algoritmo genético

Seguindo os passos de execução de um algoritmo genético, deve ser definido (i) mecanismo de seleção de cromossomos pais; (ii) operadores genéticos (cruzamento e mutação); e (iii) parâmetros e pesos do algoritmo genético e do problema de alocação.

6.3.1 Mecanismo de seleção de cromossomos pais

Como detalhado no Capítulo 2, existem várias formas de seleção de cromossomos pais, cada um com sua vantagem. Nesse trabalho é utilizado o mecanismo de seleção por método de torneio. Essa escolha foi baseada nos valores obtidos com a função objetivo serem muito próximos entre si. Apesar de retratar as diferenças existentes no problema, sempre serão valores no intervalo $[0, 1]$. Dessa forma, um método de roleta viciada comprometeria significativamente o poder dos mais fortes diante dos mais fracos, uma vez que o percentual de chance de sorteio do mais forte seria pouco maior que as soluções mais fracas.

Como o método de torneio se baseia em múltiplos sorteios, foi escolhido uma taxa de 3 sorteios para a seleção de um cromossomo pai. Lembrando que quanto maior o número de sorteios, maior a força dos cromossomos mais fortes (SIVANANDAM e DEEPA, 2010).

6.3.2 Operadores genéticos

Para o cruzamento, foi escolhido para essa prova de conceito o cruzamento uniforme, definindo a string binária aleatoriamente. Como os relacionamentos não estão em posições vizinhas no cromossomo, um cruzamento utilizando 1 ou n cortes não será tão eficaz no processo de passagem da genética de um cromossomo pai para um cromossomo filho.

Com o cruzamento uniforme, a string binária garante a possibilidade de qualquer informação genética presente no cromossomo seja levada às próximas gerações, exatamente pelo fato do relacionamento não estar claramente retratado na forma do cromossomo e sim em estruturas de dados a parte da alocação, que são as dependências.

Para a mutação, é adotada uma mutação simples em um gene do cromossomo, alterando a equipe alocada por outra aleatoriamente selecionada dentro do universo de equipes tecnicamente aptas para o gene.

A taxa de uso dos operadores genéticos é de 100% do tamanho da população, ou seja, a cada geração o número de cromossomos irá dobrar e apenas os melhores (tamanho da população) permanecem. Essa taxa de operadores genéticos é dividida de forma complementar entre o cruzamento e a mutação.

Após diversos testes empíricos, os valores de cruzamento e mutação foram estabelecidos com valor inicial de 60% e 40%, respectivamente. Com a execução do algoritmo, uma função linear é responsável por fazer a taxa de cruzamento cair e a de mutação subir, obtendo valores inversos ao inicial no fim da execução. Essa forma de uso dos

operadores genéticos recebe o nome de interpolação linear (LINDEN, 2008), e é utilizada para aumentar a diversidade genética quando a população começa a convergir, evitando ótimos locais. A função linear foi explicada na Equação (16).

O complemento dessa função é que define o valor da taxa de mutação, como mostra a Equação 17.

$$T_{mut} = 1 - T_{cruz} \quad (17)$$

6.3.3 Pesos utilizados

Com base em várias execuções teste, percebeu-se que os valores das avaliações temporal, cultural e de afinidade possuem valores distintos entre si. Assim, os pesos utilizados na prova de conceito foram definidos com o objetivo de nivelar os valores de forma a que seja perceptível os três fatores não-técnicos.

A análise temporal é a que possui os menores valores, devido ao valor máximo só poder ser atingido quando todas as equipes trabalham por 24h, 7 dias da semana, realizando *follow the sun*.

A análise cultural também possui valores reduzidos, devido à multiplicação da análise de idioma e análise de perfil organizacional. Porém os valores não são tão baixos quanto a análise temporal.

Já a análise de afinidade possui os valores mais altos, justamente por ser um atributo iniciado em 0,5 e que a média e seu desvio padrão são utilizados para formar a avaliação final, sem outros produtos.

Dessa forma, os pesos para a avaliação temporal w_t , cultural w_c e de afinidade w_a recebem respectivamente os pesos 3, 2 e 1. Assim, os resultados encontrados mostram as três análises não-técnicas fazendo efeito na busca.

Além dos pesos da função objetivo, a análise cultural também possui pesos, referentes a avaliação de idioma e a avaliação de perfil organizacional. Partindo da premissa de que sem idioma não adianta estrutura organizacional alguma, o peso do idioma deve ser maior que o do perfil organizacional. Dessa forma, os pesos w_{id} e w_{po} , respectivamente idioma e perfil organizacional recebem os valores $w_{id} = 2$ e $w_{po} = 1$.

Por fim, como detalhado na seção 6.2, a instanciação das equipes também possui pesos para a definição das afinidades, uma vez que 15 equipes foram escolhidas para simular um trabalho anterior. Os pesos utilizados para o *feedback* w_{fb} e o valor anterior de afinidade w_{af} foram 1 e 2, respectivamente. O comportamento desejado é uma mudança um pouco mais lenta com o uso de avaliações de afinidade.

Na Tabela 13 pode ser visto todos os pesos escolhidos para a prova de conceito.

Tabela 13 – Pesos utilizados na prova de conceito

w_t	3
w_c	2
w_a	1
w_{id}	2
w_{po}	1
w_{af}	2
w_{fb}	1

6.4 Resultados

Nas seções seguintes, os casos MultiMOM, GingaCC e SPL e-commerce terão suas soluções ótimas mostradas com o objetivo de analisar o resultado da função objetivo. Essa seção não inclui o problema BigExample devido a complexidade do mesmo tornar impossível, em tempo hábil, o cálculo da solução ótima.

Um algoritmo exaustivo é utilizado nos três casos, encontrando a solução ótima nos problemas em questão e com as equipes geradas anteriormente. O esperado como saída é uma solução que tenha características que representem o inverso da problemática existente no DDS. Ou seja, no aspecto temporal, se espera equipes que possuam um bom tempo disponível para se comunicar. No aspecto cultural, que as equipes possuam ao menos um idioma em comum para que elas possam se comunicar adequadamente, como também um perfil organizacional razoavelmente semelhante para que não exista uma diferença exagerada na filosofia de trabalho das mesmas. E, por fim, no aspecto de afinidade, que as equipes com boa afinidade possam trabalhar juntas novamente, ou que equipes sem bom relacionamento evitem trabalhar juntas, apesar de respeitar as exigências dos atributos temporais e culturais.

6.4.1 MultiMOM

O MultiMOM é um problema pequeno, com 20 equipes e 3 módulos, o que gera um espaço de 8000 soluções possíveis. Dessa forma, um algoritmo exaustivo pode ser utilizado sem problemas com a função objetivo do presente trabalho, de forma a encontrar a solução ótima com base nos aspectos não-técnicos.

Como mostra a Tabela 14, a solução ótima possui associada a seus módulos m_1 , m_2 e m_3 as equipes CHI1 (China), CHI2 (China) e TAI (Taiwan). Do ponto de vista temporal, como as equipes possuem fuso horário próximo (+8 para todas) e muita proximidade de seus horários de trabalho (apenas 1h a mais para a equipe do Taiwan), com os horários convertidos para GMT, temos que as equipes possuem bastante tempo trabalhando simultaneamente. As equipes da China trabalham de 16h as 20h e das 22h as 2h da manhã do dia seguinte. A equipe do Taiwan, por sua vez, tem o horário de trabalho das 17h até as 21h e das 23h as 3h da manhã (horários em GMT). Isso significa dizer que as equipes chinesas possuem o mesmo horário de trabalho (8h diárias compartilhadas), enquanto que a equipe da Singapura compartilha com as demais das 17h às 20h e das 23h as 2h da manhã, o que totaliza 6h por dia, exceto o sábado, que as equipes da China compartilham apenas o turno da manhã, totalizando 3h de trabalho. A alta quantidade de horas simultâneas faz com que a comunicação com DDS ocorra de forma mais adequada (CARMEL e ESPINOSA, 2011).

Tabela 14 – Solução ótima para problema MultiMOM e seus atributos

	M1	M2	M3	Fitness
	CHI1	CHI2	TAI	0.1768
Tempo	16h a 20h 22h a 2h	16h a 20h 22h a 2h	17h a 21h 23h a 3h	0.1446
Idiomas	Mandarim 1 Inglês 0,375 Português 0,15	Mandarim 1 Inglês 0,375 Português 0,15	Mandarim 1 Inglês 0,375	0.1727
P. Org	2 4 3 1 4 0 2 2 1 2	2 2 3 2 2 0 1 2 1 1	2 2 3 1 2 0 2 1 2 2	
Afinid.	CHI1, CHI2=0,5 CHI1, TAI=0,5	CHI2, CHI1=0,5 CHI2, TAI=0,5	TAI, CHI1=0,5 TAI, CHI2=0,5	0.2818

Pelo ponto de vista cultural, as três equipes possuem como idioma em comum o Mandarim, sendo o idioma nativo para as três equipes. Além disso, o idioma Inglês pode ser

utilizado entre todas as equipes, restando ainda o Português entre as equipes chinesas (em caso de moradores do Macau, que falam nativamente o português) (BROOKE, 2004).

Como pode ser melhor observado na Tabela 15, o perfil organizacional das equipes possui algumas boas semelhanças, apesar de algumas diferenças naturais, compensadas pelas vantagens do próprio perfil organizacional. Quanto aos processos, as três equipes utilizam modelos de processo iterativo e incremental, oferecendo qualidade e semelhança entre elas.

Tabela 15 – Respostas do Perfil Organizacional para MultiMOM

Pergunta	CHI1	CHI2	TAI	Legenda
1	2	2	2	2 - Interativo e Incremental
2	4	2	2	4 - Flexível para diferentes processos e modelos 2 - Flexível para fases do processo
3	3	3	3	3 - Métodos ágeis
4	1	2	1	1 - Controle de cronograma e funcionalidades 2 - Padronização da documentação, validação e verificação
5	4	2	2	2 - Audioconferência 4 - Videoconferência em grupo
6	0	0	0	0 - Nenhuma ferramenta de awareness
7	2	1	2	1 - Mural 2 - Blog
8	2	2	1	1 - Religião causa problema 2 - Religião não é problema
9	1	1	2	1 - Formal 2 - Informal
10	2	1	2	1 - Centralizado 2 - Descentralizado

A equipe CHI1 possui uma ótima flexibilidade de adaptação de processo, permitindo mudanças até mesmo no uso de modelos ou processos diferentes dos usuais da equipe. As equipes CHI2 e TAI, por outro lado, também possuem flexibilidade, mas em um nível menor, uma vez que ambas são aptas a adaptar suas fases de processo em detrimento de bom relacionamento de trabalho com outras equipes. Como último ponto em relação a processos, as três equipes preferem trabalhar com ciclos de vida mais curtos, como um processo ágil. Essa semelhança entre as equipes também facilita o DDS (LAMERSDORF e MUNCH, 2009).

Partindo para a pergunta sobre modelos de processo, todas utilizam algum nível de maturidade para qualidade do software. Sendo as equipes CHI1 e TAI aptas a realizar controle de cronograma e de funcionalidades (nível 1), enquanto que a equipe CHI2 realiza um nível a frente, com padronização da documentação, validação e verificação.

Partindo para a parte de CSCW, temos que as equipes CHI1 e TAI optem pelo uso de audioconferência, sem vídeo, enquanto que a equipe CHI2 permite uso até de videoconferência em grupo, porém, sem uso de ferramentas de percepção ou ambiente *same-time*. Sobre as ferramentas assíncronas de CSCW, as equipes CHI1 e TAI novamente mostraram semelhanças, optando pelo uso de Blogs. A equipe CHI2, por sua vez, utiliza um portal/mural de notícias.

Chegando às perguntas de filosofia organizacional, as equipes chinesas relatam problemas relacionados a religião no trabalho, o que não ocorre com a equipe do Taiwan. As equipes chinesas também possuem a semelhança de terem um certo nível de formalismo na passagem de informação entre as equipes. E, por fim, as equipes CHI1 e TAI possuem filosofia descentralizada, enquanto que a equipe CHI2 opta pela decisão ser tomada pela figura de um líder.

A afinidade das equipes selecionadas está com o valor inicial 0,5 entre todas, o que relata que não existem maiores problemas entre as equipes selecionadas. De certo, existem equipes com maior afinidade que as selecionadas, como é o caso da equipe IND1 e ARG. Porém, os outros atributos não técnicos não compensam essa vantagem de afinidade.

Diante dos resultados, percebe-se que os três aspectos não-técnicos foram levados em consideração para a alocação de equipe. Até mesmo a afinidade, que possui equipes com maiores valores de afinidade, também existem equipes com afinidade bem pior e com bons atributos não-técnicos, como é o fato de ING e EUA1, mas que ainda assim não compensam completamente, sofrendo perda temporal ou de aspectos organizacionais.

6.4.2 Ginga CC

O Ginga CC é um problema de maior complexidade que o MultiMOM, sendo 20 equipes candidatas e 4 módulos a serem implementados, gerando um número de 160.000 possibilidades de solução.

Como solução ótima, mostrada na Tabela 16, temos as equipes IRL (Irlanda), EUA2 (Estados Unidos), ING (Inglaterra) e ING (Inglaterra), alocadas aos módulos m_1 , m_2 e m_3 , respectivamente.

Tabela 16 – Solução ótima para problema Ginga CC e seus atributos

	M1	M2	M3	M4	Fitness
	IRL	EUA2	ING	ING	0.2090
Tempo	9h a 13h 15h a 19h	3h a 7h 9h a 13h	9h a 13h 15h a 19h	9h a 13h 15h a 19h	0.1469
Idiomas	Inglês 1 Irlandês 0,3 Francês 0,15 Alemão 0,075	Inglês 1 Espanhol 0,225 Francês 0,15 Alemão 0,075	Inglês 1 Francês 0,225 Alemão 0,075 Espanhol 0,05	Inglês 1 Francês 0,225 Alemão 0,075 Espanhol 0,05	0.2253
P. Org	2 2 2 2 4 0 3 2 2 2	2 3 3 0 2 0 3 2 2 1	2 1 3 0 5 1 1 2 2 2	2 1 3 0 5 1 1 2 2 2	
Afinid.	IRL,EUA2=0,5 IRL,ING=0,5	EUA2,IRL=0,5 EUA2,ING=0,5	ING,IRL=0,5 ING,EUA2=0,5	ING,ING=1	0.3625

Nesse caso, percebe-se que a equipe da Inglaterra foi alocada a dois módulos, m_3 e m_4 . Esse tipo de solução acarreta num maior volume de trabalho para a equipe inglesa, porém, ajuda consideravelmente a reduzir os problemas de DDS, visto que é reduzida a comunicação na equipe alocada a m_3 e a m_4 , além de reduzir o número de equipes diferentes que as outras equipes precisarão se comunicar.

Pelo aspecto temporal, as equipes ING e IRL possuem o mesmo horário de trabalho, contabilizando 8h diárias e 4h no sábado de tempo compartilhado de trabalho. A equipe dos Estados Unidos, por sua vez, tem seu fuso horário diferente, iniciando o dia mais cedo que as demais equipes. Mesmo assim, ainda possui o seu turno vespertino coincidindo com o turno matutino das demais, totalizando 4h diárias de tempo compartilhado, exceto o sábado. Também é importante destacar que a equipe inglesa não se comunicará com nenhuma outra para integrar os módulos m_3 e m_4 .

Em relação aos idiomas, todas as três equipes falam nativamente o idioma Inglês, possuindo também, alguns idiomas secundários em comum. Os perfis organizacionais, como pode ser observado na Tabela 17, também se mostraram bastante semelhantes, com todas as equipes utilizando modelos de processo iterativo e incremental, sendo a equipe da Irlanda mais propensa ao uso de ciclos de vida longos e com mais documentação, enquanto que as demais optam pelo uso de ciclos de vida mais curtos.

Tabela 17 – Respostas do Perfil Organizacional para Ginga CC

Pergunta	IRL	EUA	ING	Legenda
1	2	2	2	2 - Interativo e Incremental
2	2	3	1	1 - Apenas processos padronizados pela equipe 2 - Flexível para fases do processo 3 - Aberto a adaptação de processos de mesmo modelo
3	2	3	3	2 - Iterações longas 3 - Métodos ágeis
4	2	0	0	0 - Nenhum modelo de maturidade adotado 2 - Padronização da documentação, validação e verificação
5	4	2	5	2 - Audioconferência 4 - Videoconferência em grupo 5 - Simulação do mesmo ambiente com vídeo
6	0	0	1	0 - Nenhuma ferramenta de awareness 1 - Ferramenta de percepção assíncrona
7	3	3	1	1 - Mural 3 - Wiki
8	2	2	2	2 - Religião não é problema
9	2	2	2	2 - Informal
10	2	1	2	1 - Centralizado 2 - Descentralizado

Além disso, as equipes ING e EUA2 não utilizam modelos de maturidade, o que pode reduzir a qualidade dos produtos gerados, mas o fato de serem semelhantes faz com que tenham os problemas de filosofia de trabalho reduzidos.

Em relação a CSCW, a equipe inglesa opta pelo uso mais completo de ferramentas síncronas, com o uso de videoconferências em grupo e até mesmo o uso de ferramentas de suporte a percepção assíncronas. A equipe EUA2 utiliza ferramentas de audioconferência, enquanto que a equipe IRL utiliza até mesmo videoconferências em grupo. Nesse caso, a audioconferência é uma ferramenta comum a todas as equipes, podendo ainda o uso de videoconferência entre as equipes ING e IRL. Nas ferramentas assíncronas, EUA2 e IRL utiliza Wikis, enquanto que a equipe ING prefere o uso de murais de notícia.

A cerca das filosofias organizacionais, nenhuma das equipes possui problemas relacionados a religião, além de possuírem o mesmo formalismo na disseminação da

informação, que é a informal. Por fim, apenas a equipe dos Estados Unidos opta pela tomada de decisão centralizada no líder da equipe.

A afinidade, de forma semelhante ao estudo realizado com o MultiMOM, selecionou equipes que não tinham trabalhos anteriores. Entretanto, a escolha da mesma equipe para os módulos m_3 e m_4 também faz com que a avaliação de afinidade seja melhor, devido a afinidade interna na equipe inglesa ser maior que as avaliações entre outras equipes.

6.4.3 SPL e-commerce

Para o SPL e-commerce, temos as 20 equipes disputando 3 módulos de software, o que torna o problema com o universo do mesmo tamanho do MultiMOM, 8.000 possibilidades.

Como pode ser visto na Tabela 18, a solução ótima associa aos módulos m_1 , m_2 e m_3 as equipes IND1 (Índia), IND2 (Índia) e IND2 (Índia), respectivamente.

Tabela 18 – Solução ótima para problema SPL e-commerce e seus atributos

	M1	M2	M3	Função objetivo
	IND1	IND2	IND2	0.1899
Tempo	13h a 17h 19h a 23h	13h a 17h 19h a 23h	13h a 17h 19h a 23h	0.1366
Idiomas	Hindi 1 Inglês 0,375	Hindi 1 Inglês 0,375	Hindi 1 Inglês 0,375	0.2150
P. Org	1 4 1 2 1 0 2 1 2 2	3 4 2 1 3 0 3 2 2 2	3 4 2 1 3 0 3 2 2 2	
Afinid.	IND1, IND2=0,5	IND2, IND1=0,5		0.2997

Todas as equipes selecionadas para essa alocação foram equipes indianas, sabendo que IND2 foi alocada a dois módulos, m_2 e m_3 . Como pode ser percebido, o tempo disponível entre as equipes são as 8h diárias e 4h aos sábados, o que significa que sempre que uma equipe estiver trabalhando, a outra estará trabalhando rigorosamente no mesmo horário.

Pelo ponto de vista de idioma, todas possuem como idioma nativo o Hindi, possuindo ainda conhecimento como idioma secundário em inglês. A cerca do perfil organizacional, como pode ser observado na Tabela 19, a equipe IND1 utiliza processo sequencial, enquanto que IND2 utiliza linhas de produto de software.

Tabela 19 – Respostas do Perfil Organizacional para SPL e-commerce

Pergunta	IND1	IND2	Legenda
1	1	3	2 - Interativo e Incremental
2	4	4	4 - Flexível para diferentes processos e modelos
3	1	2	1 - Cascata 2 - Iterações Longas
4	2	1	1 - Controle de cronograma e funcionalidades 2 - Padronização da documentação, validação e verificação
5	1	3	1 - Apenas e-mail e mensageiro de chat 3 - Videoconferência par-a-par
6	0	0	0 - Nenhuma ferramenta de awareness
7	2	3	2 - Blog 3 - Wiki
8	1	2	1 - Religião causa problema 2 - Religião não é problema
9	2	2	2 - Informal
10	2	2	2 - Descentralizado

Apesar disso, as duas equipes se comprometem a ter flexibilidade total de seu modelo de processo e até mesmo utilizar processos diferentes dos adotados internamente à equipe. IND2 prefere métodos iterativos com iterações longas, métodos esses que ainda possuem características em comum, como a filosofia de gerar documentos e os passos utilizados sequencialmente aos processos (requisito, projeto, implementação e testes) (SOMMERVILLE, 2007). A cerca dos modelos de maturidade, ambas utilizam modelos baixos de maturidade. Em IND1, o segundo nível (padronização, validação e verificação) é utilizado, enquanto que IND2 possui um nível a menos (controle de cronograma e funcionalidades).

Em relação a CSCW, as ferramentas síncronas utilizadas pela equipe IND1 mostram preferência pelo uso de ferramentas de chat, enquanto que IND2 pode utilizar até mesmo videoconferência um a um. Entretanto, nenhuma das equipes adota ferramenta de suporte a percepção e ambiente *same-time*. Sobre as ferramentas assíncronas, IND1 opta pelo uso de Blogs, enquanto que IND2 direciona suas informações para Wikis.

Pelos critérios de filosofia organizacional, IND1 indica ter problemas em lidar com religião no trabalho, mas o mesmo não acontece com IND2, o que evita maiores problemas desse cunho. Diante do formalismo e centralização do trabalho, ambas concordam que a informalidade e descentralização são melhores para suas equipes.

6.5 Comparativo da abordagem proposta com a busca aleatória

Na seção anterior, os resultados obtidos mostram que a função objetivo funciona de acordo com o que foi proposto inicialmente, tendo a capacidade de identificar alocações de equipe cujos aspectos não-técnicos contribuem de forma satisfatória para viabilizar uma boa comunicação. Porém, em cenários maiores, com um universo de soluções gigantesco, passa a ser impossível encontrar a solução ótima com um algoritmo exaustivo, sendo então, necessário o uso de uma meta-heurística para encontrar um conjunto de recomendações de alocações.

Essa seção tem como objetivo aplicar a abordagem proposta com as 4 linhas de produto: MultiMOM, GingaCC, SPL e-commerce e BigExample. No caso específico da BigExample, temos 20 equipes e 15 módulos a serem alocados, gerando um número de possibilidades de solução de 32.768.000.000.000.000. Inicialmente, será mostrado um comparativo das soluções encontradas nos problemas Multi-MOM, SPL e-commerce e Ginga CC, comparando os resultados obtidos com a abordagem proposta e um algoritmo de busca aleatória, comparando as soluções encontradas por ambos com a solução ótima. Por fim, os resultados do problema BigExample são analisados entre a abordagem proposta e o algoritmo de busca aleatória, sob o ponto de vista da Engenharia de Software. No caso do BigExample, em função do tamanho do espaço de soluções, não temos a solução ótima para comparar com as soluções encontradas pela abordagem proposta.

Para a execução da abordagem proposta, existem ainda dois parâmetros que não tiveram seus valores definidos na seção 6.3.3, são eles: número de iterações e tamanho da população. Essa escolha foi deixada para o presente momento pois os algoritmos Multi-MOM, SPL e-commerce e Ginga CC possuem universo de busca incrivelmente menor que o BigExample. Sendo assim, teremos valores diferentes para as demais provas de conceito e o BigExample.

Segundo (PINHO, MONTEVECHI e MARINS, 2007), o valor inicial de um experimento com algoritmos genéticos deve variar entre 50 e 200 cromossomos. Como os três primeiros casos são pequenos, foi definido o valor de 50 cromossomos para o tamanho da

população. Para o número de iterações, também foi definido o número de 50 iterações. Essa escolha foi feita para não deixar o número de soluções máximo gerado pelo algoritmo tão alto, uma vez que o universo de busca é pequeno. O número de cromossomos gerados com essa configuração é de 2.500 cromossomos, o que representa 32% do universo de busca do Multi-MOM e SPL e-commerce (8.000 cromossomos) e 1,5625% no Ginga CC (160.000 cromossomos). Os 32% representam um valor consideravelmente alto, mas por ser um problema pequeno, a redução exagerada do número de cromossomos e/ou do número de iterações resultaria em uma queda significativa na eficiência do algoritmo genético (LINDEN, 2008). Entretanto, como o algoritmo aleatório também utilizará o mesmo número de soluções geradas (2.500), já é possível uma boa comparação dos resultados.

Para o BigExample, uma vez que apenas a abordagem proposta e o algoritmo de busca aleatória foram utilizados, um número maior de testes com os parâmetros de número de iterações e número de cromossomos foram realizados.

Para cada algoritmo, foram realizadas quatro execuções independentes, coletando quatro recomendações. Em cada algoritmo, cada recomendação conta com 50 cromossomos como resultado. Para um melhor comparativo, foram utilizados os resultados das posições: 1, 10 e 50. Isso nos fornece informação da melhor solução encontrada, de como está a qualidade dentro das 10 melhores posições e, por fim, nos fornece a qualidade da última solução armazenada.

Nas seções a seguir os resultados obtidos são mostrados e comparados.

6.5.1 Multi-MOM

Na Tabela 20, podem ser vistas as 12 soluções de cada algoritmo, com os valores obtidos nas quatro execuções e nas posições definidas. Ambos os algoritmos encontraram a solução ótima dentre as quatro execuções, sendo que a abordagem proposta encontrou a solução ótima três vezes e o algoritmo de busca aleatória encontrou duas vezes. As demais soluções encontradas no primeiro lugar de ambos os algoritmos possuem resultados bastante semelhantes, pelo ponto de vista da Engenharia de Software. Comparando as últimas soluções em primeiro lugar, temos como solução da abordagem proposta: $\{m_1:ING / m_2:IRL / m_3:EUA2\}$, enquanto que com a busca aleatória: $\{m_1:ING / m_2:IRL / m_3:SING\}$. Ambas soluções possuem uma proximidade muito grande, sendo a primeira solução melhor no quesito temporal, por conta do fuso horário da equipe EUA2 ser mais próxima das demais equipes do que a equipe SING. Em contrapartida, a segunda solução possui uma melhor relação cultural,

devido as avaliações dos idiomas e perfis organizacionais mais semelhantes. Em ambos os casos, a afinidade possui o mesmo resultado.

Tabela 20 – Soluções da abordagem proposta e da busca aleatória para Multi-MOM

	Abordagem Proposta	Busca Aleatória
1ªs	0.17680	0.17680
	0.17680	0.17680
	0.17680	0.17647
	0.17655	0.17501
10ªs	0.17432	0.17381
	0.17429	0.17371
	0.17429	0.17370
	0.17297	0.17316
50ªs	0.17075	0.14216
	0.16974	0.13955
	0.16974	0.13746
	0.13770	0.13588

Comparando os demais resultados, nas 10ªs posições, apenas uma solução da busca aleatória é matematicamente melhor que a pior solução da abordagem proposta, o que mostra uma melhor eficiência do algoritmo genético já nas dez primeiras posições. A melhor solução dentre as dez da busca aleatória pode, inclusive, ter sido um acaso difícil de ser repetido, já que aconteceu apenas uma vez em quatro execuções.

Olhando agora para as 50ªs posições das execuções, temos uma diferença bem maior que as posições anteriores, com três soluções da abordagem proposta melhor que as demais soluções do algoritmo aleatório. Entretanto, uma solução da abordagem proposta foi pior que duas soluções da busca aleatória, isso se dá devido o algoritmo ser probabilístico (LINDEN, 2008), e não determinístico. Como diante de todos os resultados, essa solução foi a única que destoou das demais, pode-se considerar que foi um acaso.

6.5.2 SPL e-commerce

Os resultados do SPL e-commerce foram bastante semelhantes ao Multi-MOM, pelo ponto de vista de comparação entre a abordagem proposta e a busca aleatória, até mesmo por possuir o mesmo tamanho de universo de soluções do Multi-MOM (8.000).

Como pode ser vista na Tabela 21, os dois algoritmos conseguiram encontrar o resultado ótimo, sendo que a abordagem proposta encontrou em duas oportunidades e a busca

aleatória em apenas uma oportunidade. Os demais resultados na primeira posição possuem valores semelhantes, mas a abordagem proposta ainda encontrou soluções matematicamente um pouco melhores.

Tabela 21 – Soluções da abordagem proposta e da busca aleatória para SPL e-commerce

	Abordagem Proposta	Busca Aleatória
1ªs	0.18510	0.18510
	0.18510	0.17773
	0.17773	0.17512
	0.17773	0.16894
10ªs	0.16316	0.15029
	0.16008	0.14845
	0.15768	0.14563
	0.15438	0.14391
50ªs	0.14073	0.13520
	0.14039	0.13346
	0.13975	0.13085
	0.13960	0.13002

Os piores resultados na primeira posição foram respectivamente: Abordagem proposta – $\{m_1:ING / m_2:IRL / m_3:IRL\}$ e Busca aleatória – $\{m_1:CHI1 / m_2:CHI2 / m_3:TAI\}$. Nas duas soluções existem 8h de tempo compartilhado entre as equipes, porém, a solução encontrada pela abordagem proposta possui uma mesma equipe (IRL) alocada aos módulos M2 e M3, o que gera um maior trabalho interno, devido a implementação de dois módulos. Esse comportamento acarreta em uma redução no tempo para comunicação com a outra equipe, que é a equipe ING. Com isso, a avaliação temporal da solução encontrada pela busca aleatória é melhor que a encontrada com a abordagem proposta. Por outro lado, os atributos culturais tornam melhor a solução encontrada pela abordagem proposta, por oferecer um perfil organizacional mais semelhante entre as equipes. Em ambos os casos existem vários idiomas em comum, até mesmo com idiomas secundários e terciários semelhantes, mas os processos utilizados na solução da abordagem proposta possuem diferenças mais significativas. Por exemplo, todas as equipes possuem processos iterativos e incrementais mas somente as equipes ING e IRL mostram flexibilidade semelhante de adaptação de seus processos. As equipes CHI1, CHI2 e TAI, da solução da busca aleatória, possuem características menos flexíveis, o que dificulta a adequação de suas atividades. Além disso, as equipes IRL e ING, da solução da abordagem proposta, possuem características idênticas de filosofias organizacionais, onde os aspectos de crença e religião não afetam as atividades, a

filosofia de comunicação informal é preferida e a tomada de decisão é descentralizada. No caso da solução da busca aleatória, a religião afeta a equipe TAI, não afetando as equipes CHI1 e CHI2, as equipes da China possuem formalidade na disseminação da informação, enquanto que a equipe do Taiwan prefere a filosofia informal, e, por fim, a equipe CHI2 prefere a tomada de decisão centralizada, enquanto que as demais equipes optam pela descentralização na tomada de decisões. Além da grande diferença nos atributos culturais, a solução encontrada pela abordagem proposta também possui uma afinidade melhor entre as equipes que a existente nas equipes da solução da busca aleatória, até mesmo por existir uma equipe a menos para se comunicar, o que facilita a comunicação como um todo. E, nesse caso, mesmo com o maior trabalho devido à equipe IRL está alocada a mais de um módulo, a afinidade é compensada pela redução da comunicação. Em conclusão, os aspectos não-técnicos tornam as soluções ótimas encontradas pela abordagem proposta melhores ou iguais às aquelas encontradas pela busca aleatória.

Passando para os resultados na décima posição, pode ser percebido que todas as soluções da abordagem proposta, todas possuem avaliação melhor que as soluções encontradas pela busca aleatória, o que mostra a melhor qualidade dos resultados obtidos com a abordagem proposta.

Vale ressaltar que, o mesmo pode-se dizer dos resultados da quinquagésima posição, onde nenhum resultado da busca aleatória é melhor que os resultados obtidos com a abordagem proposta.

6.5.3 Ginga CC

O Ginga CC é um problema de universo de busca maior, o que é melhor para um comparativo entre a abordagem proposta e a busca aleatória. Neste caso, em vez de 8.000 soluções existentes, são 160.000, o que reduz significativamente o percentual de soluções visitadas em relação ao espaço de busca total, uma vez que o número de iterações e de cromossomos utilizados é o mesmo. Sendo assim, o impacto da meta-heurística é maior, aumentando a distância entre os resultados de um algoritmo baseado em meta-heurística para um algoritmo aleatório.

Na Tabela 22, podem ser vistas as soluções de cada algoritmo. Como pode ser percebido, a abordagem proposta encontrou a solução ótima em três das quatro execuções, fato que não ocorreu dessa vez na busca aleatória.

Tabela 22 – Soluções da abordagem proposta e da busca aleatória para Ginga CC

	Abordagem Proposta	Busca Aleatória
1ªs	0.20635	0.20128
	0.20635	0.19525
	0.20635	0.19269
	0.20496	0.18371
10ªs	0.19578	0.16947
	0.19543	0.16831
	0.19525	0.16682
	0.19093	0.16529
50ªs	0.18477	0.15215
	0.18358	0.15084
	0.18356	0.15023
	0.17769	0.14986

Das quatro execuções da abordagem proposta, a pior delas já é melhor que a melhor solução encontrada pela busca aleatória, o que é um comportamento ainda melhor do que o que foi visto nos problemas anteriores, com menor universo de soluções.

Olhando agora para as soluções da décima posição, temos uma diferença muito grande, para melhor, entre as avaliações das soluções da abordagem proposta e da busca aleatória. Isso se deve ao algoritmo genético buscar as melhores soluções, enquanto que o algoritmo aleatório apenas depende da sorte em todas as suas execuções. No caso de um universo de busca ainda maior, essa discrepância poderia ser ainda mais significativa.

Ainda considerando as soluções da décima posição, podemos observar que a pior solução encontrada pela abordagem proposta, ou seja, a pior solução das quatro execuções realizadas, possui uma avaliação melhor que a melhor solução encontrada pela busca aleatória, o que mostra que mesmo nas piores execuções da abordagem proposta, o seu comportamento é melhor que a busca aleatória. Na verdade, exceto em raros casos, as soluções encontradas pela abordagem proposta na décima posição são equivalentes ou até melhores que as soluções encontradas pela busca aleatória na primeira posição.

Para fazer um comparativo entre os resultados pelo ponto de vista da Engenharia de Software também para o Ginga CC, optamos por comparar a melhor solução da quinquagésima posição da abordagem proposta com a melhor solução da décima posição da busca aleatória, os resultados são respectivamente: Abordagem proposta – $\{m_1:CHI2 /$

m_2 :EUA / m_3 :CHI1 / m_4 :CHI1}, e Busca aleatória – $\{m_1$:IRL / m_2 :COR / m_3 :ING / m_4 :ESP}.

Pelo ponto de vista temporal, a solução da busca aleatória possui uma ligeira vantagem, uma vez que a outra solução possui uma equipe com fuso horário consideravelmente diferente das demais (EUA). Porém, a diferença não é tão grande pois, na solução da abordagem proposta, os outros três módulos são alocados com duas equipes chinesas, que possuem 8h compartilhadas entre si. A equipe EUA possui 2h de tempo compartilhado por dia com as demais equipes, porém, de acordo com a tabela de dependências dos módulos, a equipe alocada ao módulo 2 (EUA) só terá relacionamento com a equipe do módulo 3 (CHI1). Dessa forma, a equipe EUA não precisará, em condições normais, se comunicar com outras equipes, apenas com CHI1. Na alocação da busca aleatória, o quesito temporal, por sua vez, terá quatro equipes de diferentes países alocados aos quatro módulos. A equipe IRL compartilhará 8h com a equipe ING, 2h com a equipe ESP e 1h com a equipe COR, mesmos tempos da equipe ING. E a equipe ESP compartilhará 3h com a equipe COR.

Pelo ponto de vista cultural, a solução encontrada pela abordagem proposta mostra resultados bem melhores. As equipes chinesas falam o mesmo idioma nativo, além de se comunicarem bem com a equipe americana com o idioma inglês. Na outra alocação da busca aleatória, as equipes IRL e ING falam nativamente o inglês e certamente se comunicam com a equipe coreana e a espanhola com o mesmo idioma. A comunicação entre as equipes espanhola e irlandesa ainda poderia acontecer em espanhol, mas com avaliação baixa. As equipes espanhola e coreana, por sinal, passariam a necessitar se comunicar basicamente sempre em inglês com as demais equipes e é importante destacar que as avaliações do idioma inglês pelas equipes ESP e COR são menores que as das equipes chinesas. Isso significa que as equipes CHI1 e CHI2 teriam menores problemas para se comunicar em inglês com a equipe americana. Além disso, entre si as equipes chinesas se comunicariam em Mandarim.

Para os perfis organizacionais, na solução da abordagem proposta, temos as equipes CHI1, CHI2 e EUA mais parecidas que as demais, por adotarem processos iterativos e incrementais, flexibilidade de modificação de processo, ferramentas de comunicação mais semelhantes, imparcialidade sob o aspecto de religião e informalidade na comunicação, sendo diferente apenas a equipe CHI1 sob o aspecto de tomada de decisão ser centralizada, diferente das outras duas equipes. No caso da solução da busca aleatória, a equipe coreana, por sua vez,

adota linhas de produto de software, enquanto que as demais preferem processos iterativos e incrementais. As equipes IRL e ESP, por exemplo, não são flexíveis para outros modelos de processo. Apesar disso, com exceção da equipe espanhola, que opta por audioconferência, as demais equipes admitem o uso de videoconferência de todas as formas, exceto ambiente *same-time*. Pelo ponto de vista de filosofias organizacionais, nenhuma tem a religião como um problema de trabalho, as equipes ESP e COR preferem formalismo na passagem de documentação enquanto que ING e IRL são mais informais. Por fim, apenas a equipe espanhola prefere a tomada de decisão mais centralizada.

Pelo ponto de vista da afinidade, existe uma leve superioridade da solução da busca aleatória, uma vez que a equipe CHI1 e EUA, alocadas na solução da abordagem proposta, participaram de projetos anteriores onde tiveram avaliações abaixo da média. Entretanto, as equipes encontradas no algoritmo aleatório não possuem afinidade acima da média, o que fez com que os aspectos culturais e temporais fossem mais significativos diante da afinidade, favorecendo a solução da abordagem proposta.

6.5.4 BigExample

O BigExample é um problema de alta complexidade devido ao número de módulos a serem implementados. Sendo assim, o valor da população foi alterado, inicialmente, para 500, enquanto que o número de iterações passou a ser de 100 iterações. Esses valores geram 50 mil soluções durante a execução do algoritmo, tanto para a abordagem proposta quanto para a busca aleatória. Dessa forma, o percentual do universo de busca explorado para esse problema é de apenas 0,000000087 %, consideravelmente menor que os casos anteriores.

Considerando o tempo de execução da busca exaustiva, que foi de 0,5s no Multi-MOM/SPLe-commerce e de 10s no Ginga CC, com 10 módulos, o tempo de execução passaria a ser de 20 anos, enquanto que, com os 15 módulos do BigExample, passaria a quase 65 milhões de anos. Portanto, não foi possível identificar qual a solução ótima para o problema BigExample.

De forma similar aos estudos de caso anteriores, para o comparativo entre a abordagem proposta e a busca aleatória, também foram realizadas quatro execuções, coletando quatro recomendações.

Como mostra a Tabela 23, a diferença matemática dos resultados obtidos entre a abordagem proposta e a busca aleatória é bastante significativa. Enquanto a abordagem

proposta encontrou como pior resultado das quatro execuções uma solução cuja avaliação tem o valor aproximado 0,3204, a melhor solução da busca aleatória possui avaliação de valor aproximado 0,2660.

Tabela 23 – Soluções da abordagem proposta e da busca aleatória para BigExample

	Abordagem Proposta	Busca Aleatória
1ªs	0.32907	0.26603
	0.32820	0.23551
	0.32857	0.23438
	0.32687	0.22977
10ªs	0.32684	0.21974
	0.32565	0.21771
	0.32466	0.21732
	0.32338	0.21518
100ªs	0.32456	0.19967
	0.32243	0.19817
	0.32161	0.19815
	0.32040	0.19811

Analisando o tempo de execução, a busca aleatória foi mais rápida, executando entre 18s e 19s, o que é esperado devido a diferença de complexidade de suas estruturas. A abordagem proposta executou entre 27s e 31s, o que representa um tempo excelente em função da qualidade da solução encontrada.

Posteriormente, para verificar os parâmetros utilizados, foi realizada uma execução com os mesmos 500 cromossomos de população e alterando o número de iterações para 1000 para a abordagem proposta. Para a busca aleatória a mesma proporção de soluções foi adicionada, ou seja dez vezes mais soluções. Como resultado, para o algoritmo aleatório, foi percebido uma melhora considerável da qualidade das soluções intermediárias em comparação com os resultados da busca com 100 iterações. Entretanto, os valores ainda são distantes dos obtidos com o algoritmo genético com 100 iterações.

Apresentando agora as soluções com 500 cromossomos e 1000 iterações, que pode ser vista na Tabela 24, a melhor solução da busca aleatória com 1000 iterações não conseguiu ser melhor que o melhor resultado obtido anteriormente com 100 iterações. Em contrapartida, a décima posição já foi consideravelmente melhor que todas as quatro execuções anteriores com 100 iterações da busca aleatória. Por fim, o maior avanço se mostra na centésima posição das recomendações, passando de aproximadamente 0,1996 para 0,2156. O tempo de execução

da busca aleatória passou de algo entre 18s e 19s com 100 iterações para 183s com 1000 iterações, aumentado de forma proporcional ao número de soluções visitadas que foi multiplicado por 10.

Tabela 24 – Soluções dos algoritmos com 1000 iterações para BigExample

	Abordagem Proposta	Busca Aleatória
1	0.33583	0.25376
10	0.33523	0.23293
100	0.33262	0.21560

Já a abordagem proposta teve um bom ganho de qualidade, apesar dos valores numéricos terem sido apenas um pouco superiores, devido às antigas soluções com 100 iterações já serem bem melhores que as soluções da busca aleatória. Consequentemente, essa distância dos valores numéricos das avaliações das soluções encontradas entre as execuções da abordagem proposta e todas as execuções da busca aleatória evidenciam como o algoritmo genético otimiza a busca no espaço de soluções. Observando a diferença entre a primeira e a centésima solução da abordagem proposta, percebe-se que o número de soluções de avaliação alta cresceu bastante, uma vez que a pior solução encontrada já foi maior que a melhor solução encontrada nas quatro execuções anteriores com 100 iterações. O tempo de execução da abordagem proposta passou de algo entre 27s e 31s com 100 iterações para 274s com 1000 iterações, aumentando também de forma proporcional ao número de soluções visitadas que foi multiplicado por 10.

Levando em consideração que foram 1000 iterações com 500 cromossomos para a população, foram gerados 500.000 soluções, o que ainda corresponde a um valor muito baixo do universo de busca. Porém, a passagem de 0,000000087% para 0,000000867% do espaço total da busca foi responsável por um ganho significativo na qualidade da abordagem proposta. Apesar do aumento do tempo de execução, a economia na análise dos atributos das equipes ainda é excepcionalmente valiosa para uma execução de pouco mais de 4 minutos, principalmente quando comparado com o esforço requerido ao gerente de projeto para realizar a alocação de forma intuitiva. E em comparativo de tempo, também é importante destacar que a abordagem proposta com 100 iterações operou em aproximadamente 30s e obteve resultados melhores que a busca aleatória com 1000 iterações que executou em 183s.

A abordagem proposta ainda foi executada com uma quantidade de 5.000 iterações. Porém, o resultado não mostrou melhora em relação à execução com 1000 iterações, e, inclusive, sendo bastante semelhante a execução com 100 iterações. Esse comportamento acontece devido à função de cruzamento e mutação serem afetadas pelo número de iterações. Com isso, a população converge muito cedo e no momento em que a taxa de mutação passa a crescer, a população dificilmente mudará, atingindo um ótimo local (LINDEN, 2008). O tempo de execução do algoritmo passou a ser de 1481s, sendo, portanto, proporcional ao incremento no número de soluções visitadas em relação às demais configurações.

Por fim, para verificar o parâmetro de tamanho da população, uma execução da abordagem proposta com 1000 iterações e tamanho de população de 1000 cromossomos foi efetuada. Os resultados foram semelhantes ao caso de 1000 iterações e tamanho de população 500, só que o tempo de execução passou de 274s com população 500 para 990s com população 1000, como mostra a Tabela 25. Neste último caso, observe que o tempo de execução aumentou de forma exponencial em relação ao número de soluções visitadas, pois uma maior população requer a aplicação de mais operações de cruzamento e mutação.

Tabela 25 – Configurações da abordagem proposta para BigExample

	(p=500 it=100)	(p=500 it=1000)	(p=500 it=5000)	(p=1000 it=1000)
1	0.32907	0.33583	0.32720	0.33502
10	0.32684	0.33523	0.32587	0.33382
50	0.32456	0.33262	0.32306	0.33165
Tempo	27s	274s	1481s	990s

6.6 Comentários finais

A prova de conceito tinha como objetivo observar dois pontos: (i) se a função objetivo, de fato, geraria recomendações favorecendo equipes semelhantes segundo os aspectos temporal, cultural e afinidade; e (ii) se a busca baseada em algoritmo genético proveria vantagem significativa em relação a busca aleatória, sem ser distante da solução ótima.

Como pode ser visto na análise do Multi-MOM, Gíngua CC e SPL e-commerce, os resultados obtidos com a busca exaustiva mostram que as equipes alocadas possuem tempo disponível para comunicação, idiomas em comum, infraestrutura e filosofias organizacionais que favorecem o uso de DDS, e respeitam as afinidades entre as equipes.

Além disso, o uso de algoritmo genético se mostrou eficiente pelo ponto de vista de qualidade e até mesmo de tempo de execução. Apesar de mais lento que o algoritmo aleatório,

o que já era esperado, a diferença não era suficiente para favorecer o algoritmo aleatório em nenhum ponto. Mesmo longas execuções do algoritmo aleatório não eram capazes de concorrer com a qualidade dos resultados obtidos por meio do algoritmo genético. Além disso, o comparativo realizado com a prova de conceito Multi-MOM, SPL e-commerce e Ginga CC mostram que a solução proposta com algoritmo genético muitas vezes encontra a solução ótima.

A cerca dos parâmetros de execução, como cada problema de otimização requer uma estrutura de dados e funções de avaliação únicas, o número de iterações e tamanho da população de um problema dificilmente poderá ser comparado com outro problema de otimização. No caso do problema de alocação em questão, para espaços de busca grandes, como é o caso do BigExample, o número de iterações que se mostrou mais adequado foi de 500 cromossomos de população e 1000 iterações. O comparativo mostrou que o aumento do número de população de 100 para 1000 resultou em melhores cromossomos em um aumento de tempo que não afeta o trabalho de um gerente de projeto. Em contrapartida, o aumento do número de iterações de 1000 para 5000 não foi favorável a melhores resultados. Outro ponto a se destacar é o aumento no tempo de execução no caso de aumento do tamanho da população de 500 para 1000, o que não foi traduzido na qualidade da população.

Outro fator importante a ser destacado é que os melhores resultados obtidos com o algoritmo genético possuem pequenas mudanças de uma solução para outra, o que é esperado com base na teoria dos esquemas, que diz que uma conversão genética será atingida durante a execução do algoritmo e a melhor solução será uma variação desse esquema (LINDEN, 2008). Esse modelo de esquemas pode ser visto nas soluções não-ótimas encontradas pelos algoritmos genético para o Multi-MOM, onde apenas um módulo possui uma equipe diferente alocada, o que mostra que um esquema foi encontrado.

O algoritmo aleatório, por sua vez, mostra um comportamento completamente inesperado e sem padrão de comportamento, exatamente pela sua natureza aleatória. Em problemas de universo de soluções pequeno, ele parece ter uma certa eficiência, mas isso é um comportamento completamente aleatório e não determinístico, o que significa dizer que nem sempre ele vai encontrar resultados satisfatórios. Com o aumento do universo de soluções nos problemas Ginga CC e BigExample, esse comportamento não adequado ficou mais claro.

Capítulo 7

Considerações Finais

Neste trabalho foi apresentado uma instanciação de uma abordagem para recomendação de alocação de equipes para desenvolvimento distribuído de software baseado em atributos não-técnicos. Conforme foi esclarecido, o processo de alocação de equipes é um trabalho muito complexo para ser resolvido apenas com o conhecimento do gerente de projeto, sendo necessário um auxílio computacional para obter mais sucesso nas escolhas das equipes que devem desempenhar determinada atividade. A abordagem faz parte do *framework* de recomendações para alocação de equipes em projetos distribuídos de LPS, sendo responsável por utilizar as dependências externas e internas de módulos de software e informações não-técnicas a cerca de equipes distribuídas e oferecer um conjunto de alocações dessas equipes aos módulos a serem implementados.

O desenvolvimento distribuído de software sofre com diversos problemas que são difíceis de serem eliminados, porém, só atualmente alguns projetos com o objetivo de reduzir esses problemas vêm sendo realizados. Além disso, os problemas comumente tratados na alocação de recursos humanos, na maior parte das propostas, leva em consideração apenas atributos técnicos. Com a globalização e a crise financeira mundial, a adoção de mão de obra terceirizada passou a ser usada corriqueiramente na indústria de software, sobretudo em países como Índia, Brasil e Estados Unidos. Devido a isso, as abordagens de alocação passaram a se mostrar ultrapassadas diante de tantas mudanças.

A abordagem aqui desenvolvida, por sua vez, lida com equipes tecnicamente qualificadas e módulos de software que, ao serem definidos, tiveram o propósito de reduzir a necessidade de comunicação durante a sua implementação, baseando-se na dependência interna e externa aos módulos. São levados em consideração como atributos não-técnicos aspectos temporais, culturais e de afinidade. Os aspectos temporais envolvem o horário de trabalho diário e o fuso horário das equipes. Os aspectos culturais envolvem conhecimento de idiomas, uso de processos de software, modelos de maturidade, ferramentas de trabalho colaborativo e filosofias organizacionais, como centralização de liderança e formalização da

documentação. Aos aspectos de afinidade, está a avaliação de trabalhos anteriores e relacionamento entre as equipes.

Dessa forma, a atual abordagem supera as limitações de outras abordagens a cerca de DDS, mostrando uma nova visão de alocação de equipes sem levar os atributos técnicos como fator determinante. Além disso, por se tratar de um problema de alta complexidade, o problema foi transformado em um problema de otimização, devido o aumento do número de módulos fazer com que exponencialmente cresçam o número de soluções possíveis para o problema. Para tal, é aplicado um algoritmo de meta-heurística, contribuindo também com a consolidação de pesquisa da Engenharia de Software Baseada em Buscas (*Search-based Software Engineering*).

A fim de avaliar os conceitos e técnicas adotadas nesse trabalho, foram apresentados quatro estudos de caso, sendo dois reais, um acadêmico e um fictício e de alta complexidade. Esses estudos mostram que os resultados obtidos são de excelente qualidade, reduzindo sobremaneira o esforço do gerente de projetos na tarefa de alocar as equipes distribuídas apenas usando seus conhecimentos, impressões e intuições sobre as equipes disponíveis, e, em sua grande maioria, sem o auxílio de qualquer ferramental computacional.

7.1 Contribuições

Como contribuição inicial, a abordagem proposta define os atributos não-técnicos que devem ser levados em consideração em problemas relacionados a DDS, dividindo-os em aspectos temporal, cultural e de afinidade. Além disso, a instanciação de um modelo de questionário de idioma e perfil organizacional foi criado com base em referências que identificam como as principais diferenças entre equipes os processos de software, modelos de maturidade, ferramentas colaborativas e filosofias organizacionais.

Ademais, a formalização e conversão de características reais em valores numéricos foi criada com o intuito de contabilizar fatores não-técnicos. Dessa forma, foi possível valorar atributos, seja para qualificar uma equipe, seja para analisar semelhanças entre equipes. Com isso, uma função para cada um dos aspectos não-técnicos foi criada de forma a verificar a eficiência de uma alocação.

A abordagem define uma estratégia de busca baseada em algoritmos genéticos para encontrar soluções para o problema de alocação de equipes distribuídas que sejam satisfatórias, ou seja, próxima da solução ótima, sem consumir um elevado custo computacional. Além do algoritmo genético, o trabalho também realiza um comparativo entre

a estratégia de busca baseada no algoritmo genético e em um algoritmo aleatório. Além disso, nos casos possíveis, um algoritmo exaustivo.

Por fim, o trabalho apresenta uma série de testes de parâmetros para o algoritmo genético em questão, utilizando diferentes valores de número de iterações e de tamanho máximo de população, permitindo uma melhor calibração desses atributos para o problema de alocação de equipes em questão.

A recomendação de alocação faz com que seja possível uma análise de atributos que dificilmente seriam analisados sem uma ferramenta computacional. Mesmo considerando os maiores tempos de execução, seria impossível realizar essa análise de forma *ad hoc*, sem o uso de uma ferramenta. Além disso, não é fornecida uma única solução, mas sim um conjunto de soluções que já possuem, matematicamente, um valor aceitável dentre as possibilidades de soluções existentes. Dessa forma, é possível para o gerente de projetos, verificar, diante das respostas oferecidas, qual a que mais se adequa ao estado atual de seu trabalho. Isso faz valer a experiência e deduções do gerente, até mesmo por se tratar de um modelo que, apesar de se aproximar do modelo real, contém aproximações.

7.2 Limitações

Apesar das relevantes contribuições, a abordagem desenvolvida apresenta algumas limitações. Por se tratar de um problema de engenharia de software, diversos problemas inerentes ao desenvolvimento de software, além dos desencadeados pelo DDS, podem ser definidos como limitações dessa abordagem. Por mais eficiente que sejam os processos de software, a engenharia de software co-localizada ainda enfrenta problemas inerentes ao desenvolvimento de software e esses problemas não serão resolvidos com uma alocação de equipes.

Um fator importante a ser destacado é que as documentações utilizadas para realizar recomendação de módulos assume que a arquitetura já está em um nível final de modelagem. Dessa forma, no caso de um projeto com múltiplas iterações, a alocação de equipes passará por vários passos, inclusive podendo atualizar módulos incompletos.

Outra limitação da abordagem diz respeito à sua validação. A avaliação feita foi uma prova de conceito a cerca das funções objetivo e do algoritmo de metaheurística. A validação prática da abordagem requer um cenário real e global para se aplicar a alocação e fazer as devidas análises dos resultados. Além disso, o uso de algoritmo genético se mostrou incrivelmente vantajoso devido a complexidade do problema e aos resultados, se comparados

a um algoritmo aleatório. Entretanto, o uso de outras técnicas de metaheurísticas também poderiam ter sido usadas na abordagem.

Por fim, o algoritmo para alocação de equipes foi implementado e possui uma interface gráfica minimamente usual, porém ainda carece de maior desenvolvimento para que possa ser utilizada na prática. Os resultados podem ser visualizados por meio de interface visual, entretanto, a instanciação do problema ainda necessita ser feita manualmente.

7.3 Trabalhos futuros

Para os trabalhos futuros, primeiramente é planejado o desenvolvimento completo da ferramenta, para que haja suporte ao uso da presente abordagem. A ferramenta deve permitir a instanciação do problema, instanciação das equipes, preenchimento dos parâmetros e pesos a serem utilizados e visualização dos resultados.

Além disso, como atualmente diversas redes sociais são utilizadas entre equipes de desenvolvimento, a obtenção de dados através de redes sociais também é uma trilha de trabalho futuro. Também seria possível definir um valor inicial de afinidade diferente do valor médio, seja pelo uso de redes sociais, ou pela utilização de mais perguntas sobre filosofias organizacionais ou até mesmo o uso de uma ontologia para comparar diferentes culturas.

Sob o ponto de vista técnico, os atributos técnicos utilizados em fases anteriores do *framework* de recomendações poderiam ser usados para verificar o volume de trabalho baseado nos relacionamentos internos dos módulos, favorecendo equipes que possuam maior qualidade técnica. Essa análise é realizada de forma superficial, com base nos relacionamentos internos, porém, os valores já atualmente calculados na recomendação de equipes hábeis não são ainda utilizados.

REFERÊNCIAS

ABNT, N. I. 1. Gestão da Qualidade – Diretrizes para a Qualidade no Gerenciamento de Projetos, Rio de Janeiro, RJ - Brasil, 2000.

AUDY, J.; PRIKLADNICKI, R. **Desenvolvimento distribuído de software**. Rio de Janeiro: Campos/Elsevier, 2007.

BARRETO, A. Apoio à alocação de recursos humanos em projetos de software: uma abordagem baseada em satisfação de restrições. **Thesis**, 2005.

BEZERRA, Y. **Multi-MOM: Um Middleware Multi-Paradigma Extensível e Orientado a Mensagens para Computação Móvel**. Universidade Federal da Paraíba. João Pessoa, p. 116. 2010.

BRIDGEENGLISH. **BridgeEnglish - iTEP**, 2012. Disponível em: <<http://www.bridgeenglish.com.br/itep/>>. Acesso em: 30 mar. 2012.

BROOKE, J. China Sees Advantages in Macao's Portuguese Past. **New York Times News**, 2004. Disponível em: <<http://query.nytimes.com/gst/fullpage.html?res=9F04E1D8103AF932A15753C1A9629C8B63&sec=&spon=&pagewanted=all>>. Acesso em: 2 nov. 2012.

CALLEGARI, D. A.; FOLIATTI, F. L.; BASTOS, R. M. **MRES - Ferramenta para seleção de recursos para tarefas de projetos de software via abordagem difusa e multicritérios**. SBES. Fortaleza: [s.n.]. 2009.

CARMEL, E.; ESPINOSA, J. A. **I'm working while they're sleeping: Time Zone Separation Challenges and Solutions**. Nedder Stream Press. [S.l.]: [s.n.]. 2011.

CARMEL, E.; PRIKLADNICKI, R. **Does Time Zone Proximity Matter for Brazil? A Study of the Brazilian I.T. Industry**. [S.l.]. 2010.

CHAVES, A. P. et al. **Context-Based Notification Supporting Distributed Software Teams Management and Coordination**. Simpósio Brasileiro de Sistemas Colaborativos. [S.l.]: [s.n.]. 2009.

COMMISSION, E. Europeans and their Languages. **Eurobarometer**, 2006. Disponível em: <http://ec.europa.eu/public_opinion/archives/ebs/ebs_243_en.pdf>. Acesso em: 18 dez. 2012.

COX, E. **Fuzzy logic for business and industry - Rockland**. [S.l.]: Charles River Media, 1995.

DEB, K. An introduction to genetic algorithms. **Sādhana**, vol. 24, Parts 4 & 5, August & October 1999. 293-315.

DIRECTORY, S. The Languages spoken in South Korea. **School Directory**, 2012. Disponível em: <<http://www.spainexchange.com/guide/KR-language.htm>>. Acesso em: 18 dez. 2012.

DUGGAN, J.; BYRNE, J.; LYONS, G. J. **A Task Allocation Optimizer for Software Construction**. IEEE Software. vol. 21, issue 3. [S.l.]: [s.n.]. 2004. p. 76-82.

FORREST, S. **Genetic algorithms**. SCUR. New York, USA: ACM. 1996. p. 77-80.

GHEZZI, C.; JAZAYERI, M.; MANDRIOLI, D. **Fundamentals of Software Engineering**. Upper Saddle River: Prentice Hall, 2002.

GRÜNINGER, M.; FOX, M. S. **Methodology for the Design and Evaluation of Ontologies**. Technical Report. University of Toronto. Toronto: [s.n.]. 1995.

GUILHERMINO, D.; TAIT, T.; HUZITA, E. **Um modelo para gerenciar a comunicação em um ambiente distribuído de desenvolvimento de software**. XIII Congreso Argentino de Ciencias de la Computación. [S.l.]: [s.n.]. 2007.

GUMM, D. C. Distribution Dimensions in Software Development Projects: A Taxonomy. **IEEE Software**, vol3, issue 5, September & October 2006. 45-51.

HARMAN, M.; JONES, B. Search-based Software Engineering. **Information & Software Technology**, v. 43, p. 833-839, 2001.

HAUPT, S. **Introduction to Genetic Algorithms**. [S.l.]: Springer, 2009.

HERBSLEB, J. D. **Global Software Engineering: The Future of Socio-technical Coordination**. International Conference on Software Engineering. [S.l.]: [s.n.]. 2007.

HERBSLEB, J. D.; GRINTER, R. E. **Splitting the Organization and Integrating the Code: Conway's Law Revisited**. ICSE'99. Los Angeles, CA: [s.n.]. 1999.

HERBSLEB, J. D.; MOCKUS, A. An empirical study of speed and communication in globally-distributed software development. **IEEE Trans. Softw. Eng.** vol. 29, no. 6, 2003. 481-494.

HERBSLEB, J. D.; MOITRA, D. **Global Software Development**. IEEE Software. EUA: [s.n.]. 2001. p. 16-20.

HUZITA, E. et al. **Um Conjunto de Soluções para Apoiar o Desenvolvimento Distribuído de Software**. II WDDS. Campinas: [s.n.]. 2008. p. 101-110.

INDIAGOVERNMENT. Census of India. **Census of India**, 2011. Disponível em: <<http://www.censusindia.gov.in/>>. Acesso em: 18 dez. 2012.

JALOTE, P.; JAIN, G. Assigning tasks in a 24-h software development model. **Journal of Systems and Software**, v. 79, p. 904-911, 2006.

KIRKPATRICK, S.; GELATT, C. D. J.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, v. Vol. 220, n. 4598, p. 671-680, 1983.

KOBITZSCH, W.; ROMBACH, D.; FELDMANN, R. Outsourcing in India. **IEEE SOFTWARE**, 2001.

LAMERSDORF, A.; MUNCH, J. **TAMRI: A Tool for Supporting Task Distribution in Global Software Development Projects**. 4th IEEE International Conference on Global Software Engineering. [S.l.]: [s.n.]. 2009. p. 322-327.

LAMERSDORF, A.; MÜNCH, J. **Studying the Impact of Global Software Development Characteristics on Project Goals: A Causal Model**. The Open Software Engineering Journal. [S.l.]: [s.n.]. 2010. p. pg 2-13.

LIMA, A. et al. **WebAPSEE: Um Ambiente Livre e Flexível para Gerência de Processos de Software**. VII Workshop de Soware Livre. Porto Alegre: [s.n.]. 2006.

LINDEN, R. **Algoritmos Genéticos**. Rio de Janeiro: Brasport, 2008.

LOUREIRO, S. et al. **Um mecanismo automático para busca de parâmetros de técnicas de classificação utilizando algoritmos genéticos**. Encontro Nacional de Inteligência Artificial (ENIA). São Leopoldo - RS: [s.n.]. 2005.

M. S. THIRUMALAI, B. M. Language in India. **Language in India**, 2002. Disponível em: <<http://www.languageinindia.com/april2002/officiallanguagesact.html>>. Acesso em: 18 dez. 2012.

MOCKUS, A.; WEISS, D. M. Globalization by chunking a quantitative approach. **IEEE Software**, v. 18, n. 2, p. 30-37, 2001. ISSN 10.1109/52.914737.

MOECKEL, A. **CSCW: conceitos e aplicação para cooperação**. CEFET-PR. Curitiba, p. 35p. 2003.

MONTANARI, R.; PILATTI, L. **Reuniões: um estudo de caso em equipes**. XIII SIMPEP. Bauru, SP: [s.n.]. 2006.

NOI, G. S. English Craze Hits Chinese Language Standards. **Yale Global Online**, 2004. Disponível em: <<http://yaleglobal.yale.edu/content/english-craze-hits-chinese-language-standards>>. Acesso em: 18 dez. 2012.

OLSON, G. M.; OLSON, J. S. Distance matters. **Human-Computer Interaction**, 2000. 139-178.

PAULISH, D. J. **Product Line Engineering for Global Development**. International Workshop on Product Line Engineering: The Early Steps. [S.l.]: [s.n.]. 2003.

PEREIRA, T. **Uma Abordagem para Recomendação de Módulos para Projetos de Desenvolvimento Distribuído de Linhas de Produto de Software**. Universidade Federal da Paraíba (UFPB). João Pessoa. 2011.

PEREIRA, T. et al. **A Recommendation Framework for Allocating Global Software Teams in Software Product Line Projects**. Second International Workshop on Recommendation Systems for Software Engineering (RSSE 2010). [S.l.]: [s.n.]. 2010.

PEREIRA, T.; ELIAS, G. **Uma Estratégia de Otimização para Agrupamento de Componentes de Software Baseada em DSM**. I Workshop Brasileiro de Otimização em Engenharia de Software. Salvador, Brasil: [s.n.]. 2010.

PINHEIRO, M.; DE LIMA, J. V.; BORGES, M. **Awareness em Sistemas de Groupware**. [S.l.]: [s.n.]. 2001.

PINHO, A. F.; MONTEVECHI, J. A.; MARINS, F. A. Análise da aplicação de projeto de experimentos nos parâmetros dos algoritmos genéticos. **Sistemas & Gestão**, v. II, n. 3, p. 314 -325, Dezembro 2007.

POHL, K.; BOCKLE, G.; VAN DER LINDEN, F. J. **Software Product Line Engineering: Foundations, Principles and Techniques**. New York: Springer, 2005.

PRESSMAN, R. S. **Software Engineering: A Practitioner's Approach**. 7ª. ed. [S.l.]: McGraw-Hill, 2011.

PRIKLADNICKI, R. **Trabalhamos Enquanto você Trabalha:** Utilizando Grounded Theory para identificar as vantagens do Brasil no mercado global de TI em função da sua posição geográfica. VII Experimental Software Engineering Latin america Workshop (ESELAW). Goiânia: [s.n.]. 2010.

PRIKLADNICKI, R.; AUDY, J. **MuNDDoS - Um Modelo de Referência para Desenvolvimento Distribuído de Software.** XVIII SBES. Brasília: [s.n.]. 2004.

PRIKLADNICKI, R.; AUDY, J. L. N. **Uma Análise Comparativa de práticas de Desenvolvimento Distribuído no Brasil e no exterior.** XX SBES. Fortaleza: [s.n.]. 2006. p. 255-270.

REIS, C. **Uma Abordagem Flexível para Execução de Processos de Software Evolutivos.** Porto Alegre. 2003.

RIBEIRO, B.; ELIAS, G. **Alocando Equipes Distribuídas com base em Aspectos Não-Técnicos.** IV Workshop de Desenvolvimento Distribuído de Software. Salvador: [s.n.]. 2010.

RIBEIRO, B.; ELIAS, G. **Uma Abordagem Baseada em Algoritmo Genético para Alocação de Equipes Distribuídas.** II Workshop de Engenharia de Software Baseada em Buscas (II WESB). São Paulo: [s.n.]. 2011.

RIBEIRO, B.; ELIAS, G. **Uso de Atributos Não-técnicos na Alocação de Equipes em Projetos de Desenvolvimento Distribuído de Software.** V Workshop de Desenvolvimento Distribuído de Software (V WDDS). São Paulo: [s.n.]. 2011b.

RUSSEL, S.; NORVIG, P. **Inteligência artificial.** [S.l.]: [s.n.], 1996.

SAKTHIVEL, S. Virtual workgroups in offshore systems development. **Information and Software Technology**, 2005. 305-318.

SANTOS, V. et al. **Um Framework de Recomendação para Alocação de Equipes de Desenvolvimento em Projetos Distribuídos de Linhas de Produto de Software.** IV Workshop de Desenvolvimento Distribuído de Software. Salvador: [s.n.]. 2010.

SANTOS, V.; ELIAS, G. **Uma Abordagem Baseada em Lógica Nebulosa para Seleção de Equipes Distribuídas Tecnicamente Qualificadas.** V Workshop de Desenvolvimento Distribuído de Software (WDDS). São Paulo: [s.n.]. 2011.

SEI, S. E. I. (. CMMI ® for Development, Version 1.3, 2010. Disponível em: <<http://www.sei.cmu.edu/reports/10tr033.pdf>>. Acesso em: 29 out. 2012.

SETAMANIT, S.; WAKELAND, W.; RAFFO, D. Improving global software development project performance using simulation. **Portland Int. Conf. on Management of Engineering and Technology**, Portland, p. 2458-2466, 2007.

SILVA, M. A. **WebAPSEE-Planner: Auxílio à Alocação de Pessoas em Projetos de Software através de políticas**. UFPA. [S.l.]. 2007.

SIVANANDAM, S. N.; DEEPA, S. N. **Introduction to Genetic Algorithms**. [S.l.]: Springer, 2010.

SOARES, L.; MORENO, M.; C., N. Ginga-NCL: Declarative Middleware for Multimedia IPTV Services. **IEEE. Communications Magazine**, v. 48, n. 6, p. 74-81, 2010.

SOFTEX. **MPS.BR: Melhoria de Processo do Software Brasileiro - Guia Geral**. [S.l.]. 2011.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: Addison-Wesley, 2007.

SOUZA, D. et al. **Um Processo de Gerência de Comunicação Baseada no PMBoK para o Desenvolvimento Distribuído de Software**. II Workshop de Desenvolvimento Distribuído de Software (WDDS). Campinas - SP: [s.n.]. 2008.

SZYPERSKI, C. **Component Software: Beyond Object-Oriented Programming**. [S.l.]: ACM Press / Addison-Wesley Publishing Co., 2000.

TIMES, N. Y. NY Times. **China Sees Advantages in Macao's Portuguese Past**, 2004. Disponível em: <<http://query.nytimes.com/gst/fullpage.html?res=9F04E1D8103AF932A15753C1A9629C8B63&sec=&spon=&pagewanted=all>>. Acesso em: 18 dez. 2012.

TREINEN, J.; MILLER-FROST, S. **Following the sun: Case Studies in Global Software Development**. IBM Corp. Riverton, NJ, USA: [s.n.]. 2006. p. 773-783.

TRINDADE, D. **Uma Ferramenta para Gerenciar a Comunicação em um Ambiente Distribuído de Desenvolvimento de Software**. Universidade Estadual de Maringá. Maringá. 2008. (Thesis).

ULLMAN, J. D. NP-complete scheduling problems. **Journal of Computer and System Sciences**, v. Vol. 10. Issue 3., p. 384-393, June 1975.

US, C. S. Country Studies US. **Country Studies US**, 2003. Disponível em: <<http://countrystudies.us/brazil/39.htm>>. Acesso em: 18 dez. 2012.

VAN DER LINDEN, F. J.; SCHMID, K.; ROMMES, E. **Software Product Lines in Action**. New York: Springer, 2007.

VELOSO, F. et al. **Slicing the Knowledge-based Economy in Brazil, China and India: A Tale of 3 Software Industries**. FINEP, Secretaria de Ciência e Tecnologia do Estado do Paraná, World Class. Paraná. 2003.

VISTAWIDE. World Languages & Cultures. **Vistawide**, 2006. Disponível em: <http://www.vistawide.com/languages/us_languages.htm>. Acesso em: 18 dez. 2012.

WARREN MCFARLAN, F. **Issues in Global Outsourcing**. Global Information Technology and Systems Management. Nashua: Ivy League Publishing. 1996.

APÊNDICE A

—

EQUIPES INSTANCIADAS

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
EUA 1	-8	Início	8h	14h	1	Inglês	1,0000	1	3
		Final	12h	18h	2	Espanhol	0,2250	2	3
					3	Francês	0,1500	3	3
					4	Alemão	0,0750	4	3
								5	3
								6	0
								7	1
								8	2
								9	1
								10	2

Espanhol	Francês	Alemão
0,75		
0,75	0,75	
0,75	0,75	0,75
0,225	0,15	0,075

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Japão	+9	Início	8h	14h	1	Japonês	1,0000	1	1
		Final	12h	18h	2	Inglês	0,3750	2	3
					3			3	1
					4			4	1
								5	2
								6	0
								7	2
								8	1
								9	1
								10	1

Inglês
0,75
0,75
0,75
0,75
0,75
0,75
0,375

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Alemanha	+1	Início	9h	15h	1	Alemão	1,0000	1	2
		Final	13h	19h	2	Inglês (51%)	0,3750	2	4
					3	Francês (15%)	0,2500	3	2
					4	Russo (5%)	0,0750	4	0
								5	5
								6	2
								7	0
								8	2
								9	1
								10	1

Inglês	Francês	Russo
0,75		
0,75		
0,75		
0,75	0,75	
0,75	0,75	0,75
0,375	0,25	0,075

[illegible]

Francês	Alemão	Espanhol
0,75		
0,75		
0,75	0,75	0,5
0,225	0,075	0,05

[illegible][illegible]

[illegible]

Inglês	Espanhol
0,75	
0,75	
0,75	
0,75	0,75
0,75	0,75
0,375	0,15

[illegible][illegible]

Nome	Fuso horário		Intervalo 1	Intervalo 2	Idiomas			Questionário
Irlanda	0	Início	9h	15h	1	Inglês	1,0000	1 2
		Final	13h	19h	2	Irlandês (36%)	0,3000	2 2
					3	Francês (20%)	0,1500	3 2
					4	Alemão (7%)	0,0750	4 2
								5 4
								6 0
								7 3
								8 2
								9 2
								10 2

Irlandês	Francês	Alemão
0,75		
0,75		
0,75	0,75	
0,75	0,75	0,75
0,3	0,15	0,075

Nome	Fuso horário		Intervalo 1	Intervalo 2	Idiomas			Questionário
China 1	+8	Início	8h	14h	1	Chinês Mandarim	1,0000	1 2
		Final	12h	18h	2	Inglês	0,3750	2 4
					3	Português	0,1500	3 3
					4			4 1
								5 4
								6 0
								7 2
								8 2
								9 1
								10 2

Inglês	Português
0,75	
0,75	
0,75	
0,75	0,75
0,75	0,75
0,375	0,15

[illegible]

Inglês	Francês	Alemão
0,75		
0,75	0,75	
0,75	0,75	0,75
0,225	0,15	0,075

[illegible][illegible]

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Israel	+2	Início	9h	15h	1	Hebreu	1,0000	1	1
		Final	13h	19h	2	Inglês	0,3750	2	1
					3			3	1
					4			4	1
								5	5
								6	3
								7	0
								8	1
								9	1
								10	1

Inglês
0,75
0,75
0,75
0,75
0,75
0,75
0,375

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Finlândia	+2	Início	8h		1	Finlandês	1,0000	1	2
		Final	14h		2	Sueco	0,6000	2	3
					3	Inglês (63%)	0,5250	3	3
					4	Alemão (18%)	0,1500	4	1
								5	5
								6	1
								7	1
								8	1
								9	2
								10	2

Sueco	Inglês	Alemão
0,75		
0,75	0,75	
0,75	0,75	
0,75	0,75	
0,75	0,75	
0,75	0,75	
0,75	0,75	0,75
0,75	0,75	0,75
0,6	0,525	0,15

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Singapura	+8	Início	10h		1	Inglês	1,0000	1	2
		Final	18h		2	Malaio	0,2250	2	4
					3	Chinês Mandarim	0,1500	3	3
					4	Tamil	0,0750	4	4
								5	5
								6	1
								7	1
								8	2
								9	2
								10	2

Malaio	Chinês Manda	Tamil
0,75		
0,75	0,75	
0,75	0,75	0,75
0,225	0,15	0,075

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
Argentina	-3	Início	8h		1	Espanhol	1,0000	1	2
		Final	14h		2	Inglês (sem ref)	0,3750	2	3
					3	Alemão		3	3
					4			4	0
								5	4
								6	0
								7	1
								8	2
								9	1
								10	2

Inglês	Alemão
0,75	
0,75	
0,75	
0,75	0,75
0,75	0,75
0,375	0,15

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
México	-6	Início	8h	14h	1	Espanhol	1,0000	1	1
		Final	12h	18h	2	Inglês	0,3750	2	4
					3	Alemão	0,1500	3	1
					4			4	2
								5	2
								6	0
								7	2
								8	1
								9	1
								10	2

Inglês	Alemão
0,75	
0,75	
0,75	
0,75	0,75
0,75	0,75
0,375	0,15

Nome	Fuso horário		Intervalo 1	Intervalo 2		Idiomas			Questionário
EUA 2	-5	Início	8h	14h	1	Inglês	1,0000	1	2
		Final	12h	18h	2	Espanhol	0,2250	2	3
					3	Francês	0,1500	3	3
					4	Alemão	0,0750	4	0
								5	2
								6	0
								7	3
								8	2
								9	2
								10	1

Espanhol	Francês	Alemão
0,75		
0,75	0,75	
0,75	0,75	0,75
0,225	0,15	0,075

Nome	Fuso horário		Intervalo 1	Intervalo 2	Idiomas			Questionário
Brasil 2	-3	Início	8h	14h	1	Português	1,0000	1 0
		Final	12h	18h	2	Inglês (sem ref)	0,3750	2 0
					3	Espanhol (sem ref)	0,1500	3 0
					4	Chinês Mandarim (sr)	0,0750	4 0
								5 3
								6 0
								7 0
								8 2
								9 1
								10 1

Inglês	Espanhol	Chinês Manda
0,75		
0,75		
0,75		
0,75	0,75	
0,75	0,75	0,75
0,375	0,15	0,075

Nome	Fuso horário		Intervalo 1	Intervalo 2	Idiomas			Questionário
China 2	+8	Início	8h	14h	1	Chinês Mandarim	1,0000	1 2
		Final	12h	18h	2	Inglês	0,3750	2 2
					3	Português	0,1500	3 3
					4			4 2
								5 2
								6 0
								7 1
								8 2
								9 1
								10 1

Inglês	Português
0,75	
0,75	
0,75	
0,75	0,75
0,75	0,75
0,375	0,15

APÊNDICE B

—

AFINIDADE ENTRE AS EQUIPES

Equipe x	Equipe y	fb(x, y)	fb(y, x)	af0	af(x, y)	af(y, x)
EUA1	CHI1	IN2	IN1	0,5	0,3333	0,4167
JAP	TAI	IND	IND	0,5	0,5	0,5
ALE	IND1	SA2	SA1	0,5	0,6667	0,5833
ING	EUA1	SA1	IN1	0,5	0,5833	0,4167
IND1	ARG	SA2	SA2	0,5	0,6667	0,6667
BRA1	ALE	IN1	IN2	0,5	0,4167	0,3333
COR	SIN	IND	IN2	0,5	0,5	0,3333
IRL	ISR	SA1	SA2	0,5	0,5833	0,6667
CHI1	ESP	SA1	SA1	0,5	0,5833	0,5833
ESP	JAP	IND	IND	0,5	0,5	0,5
TAI	FIN	IN1	IN1	0,5	0,4167	0,4167
ISR	ING	IN1	IN2	0,5	0,4167	0,3333
FIN	COR	SA1	SA1	0,5	0,5833	0,5833
SIN	IRL	IND	IN1	0,5	0,5	0,4167
ARG	BRA1	IND	IND	0,5	0,5	0,5

IN2	IN1	IND	SA1	SA2
0	0,25	0,5	0,75	1

Assinaturas

Prof. Dr. Alisson Vasconcelos de Brito - UFPB
Coordenador

Banca Examinadora:

Prof(a). Dr(a). Natasha Correia Queiroz Lino – UFPB
Membro Efetivo

Prof. Dr. Márcio de Oliveira Barros– UniRio
Membro Externo

Prof. Dr. Glêdson Elias da Silveira – UFPB
Orientador

João Pessoa, ____ de _____ de 2013.
