METHANIAS COLAÇO RODRIGUES JÚNIOR

IDENTIFICAÇÃO E VALIDAÇÃO DO PERFIL NEUROLINGUÍSTICO DE PROGRAMADORES ATRAVÉS DA MINERAÇÃO DE REPOSITÓRIOS DE ENGENHARIA DE SOFTWARE

Tese apresentada ao Programa Multiinstitucional de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, Universidade Estadual de Feira de Santana e Universidade Salvador, como requisito parcial à obtenção do título de Doutor em Ciência da Computação.

Orientador: Prof. Dr. Manoel Gomes de Mendonça Neto.

Salvador

2011

Colaço Júnior, Methanias.

Identificação e Validação do Perfil Neurolinguístico de Programadores Através da Mineração de Repositórios de Engenharia de Software / Methanias Colaço R. Júnior. – Salvador, 2011.

181f. : il.

Orientador: Prof. Dr. Manoel Gomes de Mendonça Neto.

Tese (doutorado) – Universidade Federal da Bahia, Instituto de Matemática, Doutorado Multiinstitucional em Ciência da Computação, 2011.

Referências bibliográficas.

1. Compreensão de Software. 2. Engenharia de Software Experimental.

I. Colaço Júnior, Methanias. II. Universidade Federal da Bahia, Instituto de Matemática. III. Título.

CDU – 004.41

METHANIAS COLAÇO RODRIGUES JÚNIOR

IDENTIFICAÇÃO E VALIDAÇÃO DO PERFIL NEUROLINGUÍSTICO DE PROGRAMADORES ATRAVÉS DA MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

Esta tese foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa Multiinstitucional de Pós-Graduação em Ciência da Computação da UFBA-UEFS-UNIFACS.

Salvador, 30 de junho de 2011.

Prof. Manoel Gomes de Mendonça Neto (orientador), Ph.D.
Universidade Federal da Bahia – UFBA

Prof. Asterio Kiyoshi Tanaka, Ph.D.
Universidade Federal do Estado do Rio de Janeiro – UNIRIO

Prof. Claudio Guimarães Cardoso, Dr.
Escola de Administração – Universidade Federal da Bahia – UFBA

Prof. Joberto Sérgio Barbosa Martins, Docteur
Universidade Salvador – UNIFACS

Prof. Christina von Flach Garcia Chavez, Dr.
Universidade Federal da Bahia – UFBA

RESUMO

O processo de desenvolvimento de software pode contar com a utilização de diversas ferramentas de apoio. Os sistemas de controle de versão, as listas de discussão entre as pessoas envolvidas no projeto e os sistemas de rastreamento de erros são usados freqüentemente para ajudar a controlar o andamento de projetos de software, produzindo repositórios de dados históricos.

Nos últimos anos, pesquisadores vêm realizando análises lingüísticas nas listas de discussão de projetos de software para compreender as complexidades e especificidades do seu desenvolvimento. Uma abordagem inovadora para isso é usar a Teoria da Neuro-Lingüística. A Neuro-Lingüística postula que indivíduos, em contextos específicos, utilizam um sistema representacional preferencial (SRP) para cognição. Isto significa que apesar de diferentes recursos e canais cognitivos serem usados pelos desenvolvedores para entender o software, existem sistemas representacionais preferidos pelos mesmos.

Nesta tese, apresentamos uma metodologia de análise psicométrica baseada na Neuro-Lingüística para classificar os Sistemas Representacionais Preferidos (SRP) de desenvolvedores de software. A avaliação experimental da abordagem foi realizada em três experimentos que visaram testar a classificação do SRP: (1) Um estudo realizado nas listas de discussão dos projetos do servidor Apache e do PostgreSQL; (2) Uma pesquisa de campo com engenheiros de software para estabelecer quais os tipos de sistemas representacionais são os preferidos pelos mesmos; e (3) Um experimento controlado feito na indústria para avaliação da efetividade da metodologia neste tipo de ambiente.

Os resultados indicaram que a nossa abordagem pode ser usada para classificar engenheiros de software com relação às suas preferências de representação para cognição. Esta classificação pode nortear a alocação de desenvolvedores em tarefas específicas e, possivelmente, melhorar a comunicação em organizações de desenvolvimento de software.

Palavras-chave: Compreensão de Software; Imagem Mental; Mineração de Textos; Neuro-Lingüística; Engenharia de Software Experimental.

ABSTRACT

Software development results in historic data. Version control systems, discussion lists of software projects, and defect tracking systems are used to help managing the progress of software projects.

In recent years, researchers have been conducting mailing lists linguistic analyses to understand the intricacies of software development. A new approach for that is to use NeuroLinguistic Theory (NT). NT postulates the use of a Preferred Representational cognitive System (PRS) in specific contexts. This means that, although different resources and cognitive channels are used by developers to understand software, there are representational systems that are preferred by them.

In this dissertation, we present a psychometrically-based neurolinguistic methodology to classify Preferred Representational Systems (PRS) of software developers. An experimental evaluation of the approach is carried out in three experiments to assess the Preferred Representational System of developers: (1) an study assesses PRS of top developers at the Apache server and the Postgresql mailing lists; (2) a survey of software engineers to establish what types of representational systems are the preferred by them; and (3) a controlled experiment that carefully evaluates the PRS of software engineers in an industrial setting.

The results indicate that our approach can indeed be used to classify software engineers with respect to a PRS. This type of information can support people profiling for task allocation, role assignment and, possibly, improve communication in software organizations.

Keywords: Software Comprehension; Mental Imagery; Text Mining; Neurolinguistic; Experimental Software Engineering.

Dedico este trabalho a Deus, a minha família, aos meus amigos e aos professores e alunos que fizeram parte da minha vida.

AGRADECIMENTOS

Agradeço ao Criador, pela misericórdia diária e constantes bênçãos derramadas. A minha ex-mulher Fátima, pelo apoio, carinho, amor e compreensão nos momentos que estive ausente. Aos professores, em especial ao meu orientador Manoel e à Professora Christina. Manoel, pelas luzes geniais e repentinas que alavancaram minhas idéias. Christina, pelo equilíbrio mágico entre conhecimento, serenidade e compreensão que eu jamais vi. E a ambos, por terem decifrado que poderiam vir bons frutos de um ´´desconhecido´´.

Ao professor Leite da Universidade da Flórida, pela apresentação da Teoria de Resposta ao Item, bem como aos professores Tao Xie, da Universidade da Carolina do Sul, e Hassan, da *Queen's University*, pelas conversas esclarecedoras sobre suas pesquisas.

Agradeço também aos amigos de turma, especialmente a Glauco, Manoel Neto, Terceiro e Antônio. Eles também ficaram com um pé atrás no primeiro momento, mas tornaram-se anfitriões maravilhosos, além de amigos e parceiros de trocas de ensinamentos.

Agradeço aos meus alunos e ex-alunos, os quais sempre contribuem para o meu crescimento. Em se tratando de participação na concretização das idéias, sou grato a Fernanda, Francisco, Wellyngton (colega e aluno, apresentou-me a Neuro-Lingüística), Mário, Alex, PH, Maria, Daniela, Igor e Geyzon. No caso de Salvador, especialmente a Dudu, Surf e Albert, pelo companheirismo. Não poderia esquecer André Vinícius, ex-aluno e agora também meu colega e Professor, segurando com raro profissionalismo nossos projetos. Sem ele, não haveria tempo suficiente para tese.

À Carmem do DMCC, pela prontidão e solicitude nas diversas vezes que precisei. À coordenação e aos professores do DSI da UFS, pela liberação de horas para meus estudos.

À Coca-Cola de Sergipe e Alagoas (CIAL), pela colaboração com dados resultantes de desenvolvimento de software realizado na indústria.

Por fim, agradeço a minha amada família, base da minha formação. Obrigado pai, mãe, Mahely, Rafa, Jonatas, Alex, Tia Tê, Tici, Sá, Felipe I e II, Lucas I e II, Júnior, Gardênia, ex-cunhados e ex-cunhadas, Valdeci, Manoel, Maria, Gabriela, Marina, Conceição, Licinho e Dida.

“Se as cores vão berrando, num sol ensurdecedor. Fecho os olhos, outro mundo, vou morar no interior”

Gessinger

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO DE TERMOS

Modelo Mental	é uma representação interna que o ser humano elabora a partir de uma determinada situação, contexto ou entidade a ele apresentado.
Sistema de Representação Preferencial	para apresentar ou aprender sobre uma situação, contexto ou entidade, os seres humanos usam 3 sistemas básicos de representação: (1) o auditivo, formado pelos sons; (2) o cinestésico, formado pela prática e pelo fazer; (3) e o visual, formado por desenhos, imagens e diagramas. Um Sistema de Representação Preferencial é o sistema básico de representação mais utilizado por uma pessoa em um determinado contexto.

LISTA DE SIGLAS E ABREVIATURAS

ADS	Ambiente de Desenvolvimento de Software
ANOVA	*Analysis of Variance*
API	*Application Program Interface*
CCI	Curva Característica do Item
CSA	*Cognitive Style Analysis*
DF	*Document Frequency*
DW	*Data Warehouse*
ES	Engenharia de Software
ETL	*Extraction, Transformation and Load*
GQM	*Goal Question Metric*
IDF	*Inverse Document Frequency*
LIWC	*Linguistic Inquiry and Word Count*
NP	*Noun Phrases*
OLAP	*On-Line Analytical Processing*
FOSS	*Free Open Source Software*
PNL	Programação Neuro-Lingüística
POS	*Part of Speech*
RD	Recuperação de Dados
RI	Recuperação de Informação
SMART	*Statistical Multilingual Analysis for Retrieval and Translation*
SPSS	*Statistical Package for the Social Sciences*
SRP	Sistema de Representação Preferencial
TCT	Teoria Clássica dos Testes
TRI	Teoria de Resposta ao Item
VAC	Visual – Auditivo - Cinestésico
VARK	Visual – Aural – Read/Writer - Kinesthetic
VI	*Verbalizer-Imagery*
VCS	*Version Control System*
WA	*Wholist-Analytic*

Capítulo 1

Este capítulo aborda a motivação principal para esta tese, bem como justificativa, seus objetivos e questões de pesquisa. Em seguida, é apresentada a visão geral e histórica da solução proposta.

1 INTRODUÇÃO

Em virtude do tamanho e complexidade que os softwares têm alcançado, diariamente, engenheiros de software se deparam com tarefas de manutenção que exigem compreensão de código não familiar. Essa situação suscita a compreensão desse código e de toda documentação que o envolve, exigindo do desenvolvedor a construção de um modelo mental para o entendimento do sistema sob análise (Klemola e Rilling, 2002).

Para alcançar esse entendimento, desenvolvedores utilizam diferentes recursos e sistemas de representação, dentre os quais é possível enumerar:

(1) exemplos, analogias e execução do código;

(2) descrições visuais, diagramas e modelos de gráficos do sistema; ou ainda

(3) descrições textuais e análise do código fonte.

Claramente estes recursos são complementares e podem ser combinados. Todavia, existe um Sistema de Representação Preferencial (SRP)? Ou ainda, existe uma ordem ou combinação preferida de sistemas representacionais no processo de compreensão?

Recursos visuais tais como diagramas (Moody, 2009) e metáforas não-convencionais de visualização estão sendo cada vez mais utilizados em engenharia de software (Diehl, 2007). Estudos mostram que a maneira pela qual os engenheiros de software processam esses recursos tem um impacto no processo de compreensão, tanto para textos (Maldonado et al., 2006), quanto para diagramas (Travassos et al., 1999). No entanto, não

existem evidências e estudos que avaliem quais tipos de sistemas de representação são os preferidos pelos engenheiros de software.

Esta é uma questão amplamente discutida no campo da psicologia: pessoas diferentes, em contextos diferentes, podem ter preferências de representação diferentes. De fato, a área de psicologia aceita bem a existência de diferentes formas de representação para a cognição (Dent, 1983)(Matthews, 1991)(Peters et al., 2008). Processos mentais internos, tais como a resolução de problemas, uso da memória e linguagem são formados por representações visuais (imagens e diagramas), auditivas (sons) e cinestésicas (experiências físicas e práticas). Essas representações são acionadas quando as pessoas pensam ou envolvem-se na realização de atividades e tarefas cotidianas. Quer seja em uma conversa, escrevendo sobre um tema específico ou lendo um livro, representações internas sensoriais são constantemente formadas e ativadas, impactando diretamente sobre o desempenho de uma pessoa na execução dessas atividades.

A afirmação de que existem sistemas de representação diferentes para cognição suscitou o surgimento de novas teorias, tais como a da Programação Neuro-Lingüística, a qual propõe que indivíduos, em contextos específicos, utilizam um Sistema de Representação Preferencial (SRP) (Bandler e Grinder, 1979).

Bandler e Grinder, criadores da Programação Neuro-Lingüística (PNL), fazem uma afirmação ainda mais controversa, a de que as pessoas dizem palavras e frases sensoriais, também chamadas de dicas verbais ou predicados sensoriais, as quais indicam um processamento contextual visual, cinestésico ou auditivo (Bandler e Grinder, 1979)(Dilts et al., 1980). Ou seja, quando o indíviduo fala ou escreve o que está pensando, é possível detectar, naquele momento e contexto, o sistema de representação utilizado. Além disso, para PNL, se o sistema representacional sendo usado por uma pessoa também for usado pelo seu interlocutor, a empatia e eficiência da comunicação aumentam.

As afirmações da PNL têm dividido os pesquisadores da área de psicologia cognitiva. Alguns não encontraram evidências para estas declarações (Elich et al., 1985), contudo, foram criticados pela falta de compreensão do conceito de SRP (Einspruch e Forman, 1985). Por outro lado, existem pesquisadores que têm apresentado evidências empirico-científicas sobre a PNL, enfatizando a necessidade de ampliação das pesquisas (Tosey e Mathison, 2009)(Turan e Stemberger, 2000).

Em se tratando de Engenharia de Software, esta tese se propõe a averiguar se a teoria PNL pode ser aplicada neste contexto, possivelmente permitindo melhorar o processo de comunicação em projetos, bem como o processo de alocação de recursos humanos em organizações de engenharia de software.

1.1 MOTIVAÇÃO

A principal ferramenta no desenvolvimeto de software são as pessoas. O sucesso ou fracasso de uma equipe depende, em alta proporção, das interações entre os seus membros. Uma interação efetiva entre os membros é influenciada pelas características da personalidade de cada um. Neste contexto, detectar as preferências representacionais dos desenvolvedores pode aumentar a empatia nas comunicações da equipe, ou seja, cada membro pode ser mais estimulado no seu Sistema de Representação Preferencial, aumentando a efetividade da comunicação, da compreensão do software e da resolução de atividades de desenvolvimento e manutenção.

Alocar uma pessoa em uma tarefa, não só por suas habilidades técnicas, mas também de acordo com sua personalidade, é fundamental para o sucesso de qualquer projeto de software. Howard (2001) afirma que conhecimento, qualificações, competências técnicas e experiência dos membros da equipe são essenciais, mas igualmente importante é compreender as diferentes personalidades de trabalho dos desenvolvedores de software. O segredo da produtividade é alinhar as exigências de um determinado projeto com as personalidades de seus membros.

Detectar, por exemplo, através da classificação neuro-linguística, que um analista de sistemas pouco usa o sistema de representação visual pode ajudar a solucionar suas dificuldades com diagramas de projeto ou motivar a realocação do mesmo para outra atividade. Muitas vezes perde-se um profissional por uma alocação mal feita de papéis. Um grande programador não se torna necessariamente um bom analista. Em outras situações, o sistema cognitivo preferencial de um indivíduo pode não se ajustar ao perfil de seus colegas

ou à forma de trabalho de uma organização. Young et al. apresentaram um caso de um bom programador com habilidades analíticas e boa capacidade auditiva que deixou uma empresa por não conseguir trabalhar com o resto da equipe, a qual tinha um foco diferente do seu (Young et al., 2005).

As pesquisas na área de PNL ainda são poucas. É possível encontrar alguns estudos nas áreas de administração e educação (Tosey e Mathison, 2003)(Peters et al., 2008), mas elas inexistem no âmbito de engenharia de software. Esta área é intensiva em comunicação e tecnologia, produzindo e utilizando muitos artefatos durante o desenvolvimento de software. Em outras palavras, diversos recursos de comunicação são utilizados por engenheiros de software na sua rotina diária de trabalho, sendo um dos principais o texto livre enviado através de listas de discussão e emails.

Comunidades virtuais para desenvolvimento de projetos de software possuem extensa comunicação entre seus participantes através de listas de discussão. De forma similar, times de projeto presenciais utilizam fortemente correio eletrônico e programas de conversação on-line durante projetos de software. Ambos os casos possibilitam a análise da comunicação textual desenvolvida por estes indivíduos.

Se comparadas com outros artefatos de desenvolvimento, tais como o próprio código e relatórios de erros, as comunicações com texto livre são notoriamente menos estruturadas, demandando uma análise diferenciada. Apesar desta dificuldade, este tipo de informação é muito atraente. Listas de discussão e correio eletrônico possuem um número maior de assuntos debatidos e uma liberdade de expressão que facilita a exteriorização de preferências e opiniões dos engenheiros de software.

A análise textual tem sido aplicada em vários tipos de artefatos da engenharia de software (Li et al., 2006)(Mockus e Votta, 2000). Li et al. (Li et al., 2006) combinou classificação manual, análise de texto e aprendizado de máquina para classificar bancos de dados de erros. Mockus et al (Mockus et al., 2002) e Bird et al (Bird et al., 2006) exploraram as listas de e-mails de projetos de software FOSS para extrair informação sobre o processo de desenvolvimento de software livre. Rigby & Hassan em (Rigby e Hassan, 2007) realizaram uma análise psicométrica, através da mineração das listas de discussões, para estudar os motivos de abandono de projetos de software livre por desenvolvedores. Apesar destes esforços, como abordado anteriormente, é desconhecida outra pesquisa que efetue algum tipo

de classificação neuro-linguística sobre informação textual não estruturada. Além deste aspecto, é possível enumerar mais 3 motivações para esta tese:

(1) deseja-se desenvolver uma nova abordagem para análise psicométrica baseada em classificação neuro-linguística e em análise automatizada de texto livre;

(2) deseja-se contribuir para a melhor compreensão da classificação neuro-linguística (e indiretamente da PNL), área com grande evidência de mercado, mas ainda com grande escassez de pesquisas científicas;

(3) e, principalmente, o estudo da classificação de sistemas representacionais preferenciais de desenvolvedores abrirá um conjunto significativo de oportunidades de melhorias no gerenciamento de pessoas, papéis, processos e comunicação em organizações de desenvolvimento de software.

1.2 CONTEXTO DO TRABALHO

Diante das lacunas apresentadas na seção anterior, esta pesquisa produziu uma metodologia e um questionário de análise psicométrica baseados na Neuro-Lingüística, além de ter desenvolvido estudos para avaliar a viabilidade de seus usos em engenharia de software.

A metodologia foi automatizada através de uma ferramenta, nomeada de NEUROMINER. O NEUROMINER usa análise lingüística e contagem de palavras (Linguistic Inquiry and Word Count - LIWC) para classificar os sistemas representacionais preferidos (SRP) de desenvolvedores, operando sobre texto livre produzido pelos mesmos. Há uma combinação de técnicas de mineração de texto e de análise estatística com palavras sensoriais da PNL.

A base do NEUROMINER é um dicionário neuro-linguístico formado pelos predicados sensoriais (palavras que detectam o sistema de representação usado) da PNL e por conceitos da Engenharia de Software. Além disto, foi feita uma adaptação de técnicas de

mineração de texto, a qual processa o conteúdo de e-mails e classifica programadores de acordo com esse dicionário.

Depois da ferramenta, foi desenvolvido um questionário para realização de uma pesquisa de campo (survey) baseada em Neuro-Lingüística. Os objetivos foram: (1) caracterizar os Sistemas Representacionais Preferenciais de engenheiros de software e (2) dispor de alternativa de avaliação do NEUROMINER. A Teoria de Resposta ao Item (TRI) foi utilizada para análise das questões e cálculo dos escores de cada participante.

A justificativa do trabalho, a metodologia adotada, os seus objetivos e questões de pesquisa são apresentados a seguir.

1.3 JUSTIFICATIVA

Em primeiro lugar, as limitações contextuais dos trabalhos investigados estimularam a exploração da associação de conceitos da psicologia ao contexto do desenvolvimento de software, uma vez que não foi encontrada essa associação nas áreas estudadas.

Em segundo plano, a inexistência de análises textuais Neuro-linguísticas de dados provenientes do processo de desenvolvimento de software justificaram o desenvolvimento do NEUROMINER. Além disso, desconhece-se outra ferramenta que faça algum tipo de análise Neuro-Lingüística automatizada de um texto.

Finalmente, diante da escassez de pesquisas científicas sobre a PNL, houve a oportunidade de apresentar resultados empírico-científicos da aplicação de um dos seus princípios (a classificação neuro-linguística) em Engenharia de Software.

1.4 METODOLOGIA E DELIMITAÇÃO DA PESQUISA

A caracterização de uma pesquisa é feita de diversas formas diferentes e o tipo de pesquisa a realizar depende dos objetivos do trabalho, da natureza do problema e das possibilidades do pesquisador (Richardson, 1999).

Köche (Koche, 1982) afirma que pode haver um número infinito de tipos de pesquisa e prefere classificá-las em relação ao procedimento geral que é utilizado na forma de investigar. Sendo assim, ele define três tipos de pesquisa: bibliográfica, experimental e descritiva. Para Ruiz (Ruiz, 1982), existem diversas espécies de pesquisa científica e diferentes metodologias. Para ele, as espécies de pesquisa científica são: exploratória, teórica e aplicada; e como metodologia define: pesquisa de campo, pesquisa de laboratório e pesquisa bibliográfica. Já Parra Filho (Parra Filho, 1998) classifica três tipos de pesquisa: pesquisa teórica, pesquisa aplicada e pesquisa de campo.

Richardson (Richardson, 1999) categoriza as pesquisas em conformidade com o método, da seguinte forma: pesquisa histórica, pesquisa-ação, exploratória, descritiva e explicativa.

Vergara (Vergara, 2006) propõe uma taxonomia para classificação da pesquisa, que a qualifica em relação a dois aspectos:

(1) quanto aos fins, na qual a pesquisa pode ser: exploratória, descritiva, explicativa, metodológica, aplicada e intervencionista;

(2) quanto aos meios de investigação, cuja classificação pode ser: pesquisa de campo, de laboratório, documental, bibliográfica, experimental, ex post facto, participante, ação e estudo de caso.

É possível observar que uma classificação ou categorização estanque para tipos de pesquisa não é defendida pelos autores, fato refletido em citações sobre as classificações expostas pelos mesmos. Por esse motivo, pois, que Köche (Koche, 1982) afirma:

> Não se pode a rigor querer estabelecer uma nítida separação entre um ou outro tipo de pesquisa. Muitas vezes se encontram esquemas mistos que utilizam tanto a constatação quanto a manipulação de variáveis.

Segundo Richardson (Richardson, 1999), “a categorização apresentada é arbitrária com categorias não excludentes. Portanto, o leitor não deve considerar dita classificação como algo definitivo”.

Partindo destas afirmações e definições encontradas sobre os diferentes tipos de pesquisas existentes, pode-se definir que a tese foi norteada com características inerentes aos tipos de pesquisa descritiva, metodológica e aplicada.

A pesquisa descritiva, por expor as características das atuais técnicas de análise textual em Engenharia de Software; a metodológica, na utilização de dados reais e proposição de um caminho para ajudar na combinação e representação de evidências experimentais; e a aplicada, na motivação pela finalidade prática, ou seja, ao invés de especulação, propôs-se e testou-se a utilização de técnicas específicas para o contexto em questão.

Quanto aos meios, a pesquisa foi bibliográfica, documental, de laboratório e experimental. Bibliográfica e documental, porque foram investigados materiais acessíveis ao público em geral, como livros, artefatos de teste, artigos, relatórios e listas de discussão públicas e privadas, bem como materiais não publicados pela indústria.

Classificou-se também como de laboratório e experimental, devido às simulações que foram feitas em computador e devido aos experimentos realizados com manipulação e controle de variáveis independentes, observando-se as variações produzidas em variáveis dependentes.

1.5 OBJETIVOS DO TRABALHO

Em se tratando dos objetivos deste trabalho, dividimos sua classificação em dois níveis. Um objetivo geral que define o que se pretende alcançar com a realização da pesquisa e um conjunto de objetivos específicos que definem as várias metas a serem cumpridas para o alcance do objetivo geral (Richardson, 1999).

1.5.1 Objetivo geral

Utilizar os princípios da Neuro-Lingüística para extrair o canal cognitivo mais usado pelos desenvolvedores envolvidos em projetos de software, validando se esses princípios possuem aplicabilidade na área de Engenharia de Software.

1.5.2 Objetivos específicos

1. Analisar as definições da Neuro-Lingüística sobre identificação de sistemas representacionais preferenciais, validando se as mesmas podem ser usadas para segmentar programadores.
2. Prover uma infra-estrutura de repositório central de dados para mineração de dados qualificados do processo de desenvolvimento de software. Este repositório deve suportar a coleta dos dados de interesse desta tese, e ser genérico o suficiente para coletar outros tipos de dados de engenharia de software.
3. Planejar e executar um experimento controlado para validar a infra-estrutura acima mencionada.
4. Identificar, adaptar, testar e comparar técnicas de mineração de dados para extrair informações das discussões entre programadores, segmentando os mesmos de acordo com seus sistemas preferenciais de representação.
5. Criar uma ferramenta de mineração de texto específica para classificação Neuro-Lingüística, utilizando as técnicas selecionadas e/ou adaptadas e alguma relação contextual com a área de Engenharia de Software.
6. Planejar e executar um experimento utilizando o NEUROMINER para classificar o SRP dos principais colaboradores de dois grandes projetos de software livre.
7. Desenvolver um questionário para identificar as preferências de aprendizado de uma pessoa, específico para área de Engenharia de Software.

8. Utilizar este questionário em um estudo de campo para analisar o SRP de uma população significativa de engenheiros de software.
9. Planejar e executar um experimento controlado na indústria com a triangulação do NEUROMINER, do questionário de preferências e da execução de atividades controladas de engenharia de software. O objetivo é averiguar se há efetividade no uso de cenários visuais por programadores considerados essencialmente não-visuais pela técnica e/ou pelo questionário desenvolvido.

1.6 QUESTÕES DE PESQUISA

Baseando-se na fixação dos objetivos específicos e dos resultados esperados, este trabalho de Doutorado ataca as seguintes questões de pesquisa:

1. É viável manter um repositório central para mineração de dados da Engenharia de Software?
2. Listas de discussão permitem uma classificação Neuro-Lingüística?
3. Existem evidências de que a classificação Neuro-Lingüística identifica os sistemas representacionais preferenciais de um desenvolvedor de software?
4. Quais os sistemas representacionais usados pelos desenvolvedores mais importantes de projetos FOSS? Quais os dos menos importantes?
5. Desenvolvedores FOSS que mais alteram código são menos dependentes dos sistemas de representação visual e auditivo?
6. Desenvolvedores mais cinestésicos da indústria estudada dependem mais do acesso ao código para realizar tarefas de compreensão?
7. Programadores visuais da indústria estudada alcançam um desempenho melhor ao usar uma ferramenta de visualização?

1.7 VISÃO HISTÓRICA DA PESQUISA

Este trabalho tem cinco principais contribuições, a saber. Projeto e construção de um *data warehouse* para engenharia de software. Desenvolvimento e avaliação de uma abordagem de mineração de modificações de software. Desenvolvimento e avaliação de uma abordagem de mineração Neuro-Lingüística de listas de discussão de software. Elaboração, aplicação e análise dos dados de um questionário para determinação do perfil neuro-lingüístico de engenheiros de software. E, execução de um experimento integrado utilizando as abordagens anteriores. Estas contribuições serão descritas brevemente nas seções a seguir e serão discutidas em detalhe no decorrer desta tese.

1.7.1 Fase 1: Software *data warehousing*

Os modelos de maturidade de software sugerem a criação de um repositório integrado para armazenamento de métricas (SEI, 2002). O objetivo é garantir a consistência e homogeneidade dos dados de diferentes projetos, característica comum em um *Data Warehouse* (DW). Um DW é um banco de dados histórico, separado lógica e fisicamente do ambiente de produção da organização, projetado para armazenar dados extraídos de diversos ambientes. Antes de serem armazenados no DW, os dados são selecionados, integrados e organizados para que possam ser acessados da forma mais eficiente, auxiliando assim o processo de tomada de decisão (Colaço Jr, 2004)(Kimball, 1998).

Neste contexto, antes da concepção do módulo principal do NEUROMINER, visando um alinhamento com os modelos de maturidade, avaliamos que, independente do tipo de análise que seria executada, havia a necessidade da criação de uma arquitetura com repositório central que possibilitasse qualidade e um bom desempenho para extração, carga e consulta de dados.

Existiam abordagens para a criação de repositórios centrais de dados para Engenharia de Software (Becker et al., 2006)(Palza et al., 2003), porém as mesmas não lidavam com os níveis mais baixos do processo de desenvolvimento de software, tais como o histórico de modificações do código e dados desestruturados oriundos de listas de discussão. Consequentemente, completando a abordagem sugerida em (Becker et al., 2006)(Palza et al., 2003), construímos um ambiente de *Data Warehousing* genérico para engenharia de software. Depois de validado (próxima seção), o ambiente foi explorado pelo NEUROMINER e pode ser explorado por outras ferramentas de mineração de dados de desenvolvimento de software, além de servir de base para produção de relatórios analíticos para a área.

1.7.2 Fase 2: Mineração de modificações de software

Para validar a arquitetura de DW construída, foi feito um estudo de caso e um experimento controlado, os quais testaram todas as camadas da arquitetura. Os dois testes exploraram dados do histórico de modificações de software e foram executados na indústria.

O estudo de caso utilizou o DW como fonte de dados para um *Dashboard* de acompanhamento de manutenções corretivas (Colaço Jr et al., 2009). No experimento, o DW foi minerado para descoberta de regras de associação durante as atividades de manutenção. Módulos de software que geralmente são alterados em conjunto foram identificados (Colaço Jr et al., 2009B), bem como um *plug-in* foi criado para aconselhar os programadores sobre os módulos que devem ser analisados em conjunto durante a manutenção de software (Santos, Colaço Jr. e Mendonça, 2009).

Os resultados dos testes iniciais não só indicaram a viabilidade do uso da arquitetura para o NEUROMINER, como também mostraram que é possível a adaptação de um processo formal de experimentação para a condução de experimentos que avaliem a mineração de repositórios de software (Wohlin et al., 2000). A partir deste trabalho, foi desenvolvido um módulo ETL específico para extração, transformação e carga de e-mails para o DW – *Data Mart* do NEUROMINER.

1.7.3 Fase 3: Criação do NEUROMINER para mineração Neuro-Lingüística de listas de discussão de software

Com a arquitetura de DW definida e validada, a análise de texto apresentada por Rigby & Hassan em (Rigby e Hassan, 2007) inpirou o desenvolvimento de uma ferramenta de análise psicométrica baseada na teoria da Neuro-Lingüística. Nomeada de NEUROMINER (www.neurominer.com), a ferramenta é um tipo de LIWC inovador para identificar o Sistema Representacional Preferencial (SRP) de desenvolvedores de software. O NEUROMINER combina técnicas de mineração de texto e estatística com os predicados sensoriais da PNL, objetivando classificar programadores.

As características básicas da NEUROMINER são:

1. Uso de um DW.
2. Utilização de um dicionário neuro-linguístico próprio.
3. Normalização de sinônimos com dicionários para o português do Brasil (http://www.nilc.icmc.usp.br/tep2/index.htm) (Maziero et al., 2008), e para língua inglesa (http://wordnet.princeton.edu/wordnet/download/) (Wordnet, 2006).
4. Processamento de locuções substantivas (*noun phrases*) contextuais formadas da combinação de uma taxonomia para Engenharia de Software com o dicionário neuro-linguístico.
5. Implementação e uso de um módulo de análise de variância (ANOVA) para classificação dos programadores.

Para sua avaliação, foi realizado um experimento em dois grandes projetos FOSS (Colaço Jr. et al., 2010), o Apache e o PostgreSQL. Os resultados apresentaram um alinhamento entre os perfis de trabalho dos desenvolvedores analisados e seus perfis neurolinguísticos levantados pela ferramenta.

1.7.4 Fase 4: Desenvolvimento de questionário para determinação do perfil neuro-lingüístico de Engenheiros de Software

O alinhamento dos desenvolvedores classificados com seus perfis neuro-lingüísticos, obtido no primeiro estudo com o NEUROMINER, não foi considerado suficiente para validação desta tese. Houve a necessidade do desenvolvimento de outros mecanismos que possibilitassem uma avaliação mais apurada das classificações geradas.

A alternativa levantada foi o preenchimento por parte dos desenvolvedores de formulários que avaliassem, através de *escores* para cada sistema, alguma preferência por um ou mais sistemas representacionais. Assim, seria possível comparar as classificações do NEUROMINER com as classificações do questionário.

Em ampla busca bibliográfica, encontramos apenas um questionário que trabalhava com SRP e apresentava uma base científica no seu desenvolvimento. Este questionário foi desenvolvido por Fleming para identificar as preferências de aprendizado de uma pessoa, dividindo-as em visual, cinestésica, auditiva e leitura/escrita (Fleming e Mills, 1992)(Fleming, 2008). Em (Leite e Svinicki, 2010), Leite et al. apresentam algumas evidências da validade dos *escores* utilizados por este questionário para medir as preferências de aprendizado de um indivíduo, contudo, eles consideram o sistema de *pontuação* utilizado no referido questionário arbitrário (Leite e Svinicki, 2010). O sistema é baseado em desvio padrão e sempre depende da existência de uma população para normalizar os e*scores*. Além disso, considera que todas as questões são equivalentes (possuem mesmo peso) com relação à quantificação das preferências (construto ou traço latente) de cada indivíduo.

Sendo assim, para esta pesquisa, procuramos resolver as limitações do sistema de pontuação e a fragilidade contextual do questionário de Fleming. Isto resultou no desenvolvimento de um questionário específico para área de Engenharia de Software e a aplicação de um sistema de *pontuação* não arbitrário, baseado na Teoria de Resposta ao Item (TRI) (Reise et al., 2005). A Teoria da Resposta ao Item é um conjunto de modelos matemáticos e estatísticos que são utilizados para (1) análise de itens e escalas, (2) criar e administrar medidas, e (3) medir indivíduos ou organizações em um traço latente de interesse, no nosso caso, preferências cognitivas (Reeve e Fayers, 2005).

Após a criação do questionário, foi realizado um *survey* (pesquisa de campo) com engenheiros de software, validando a eficácia do questionário e revelando uma grande diversidade de SRPs na população estudada.

1.7.5 Fase 5: Experimento integrado utilizando o NEUROMINER, o *SourceMiner* e o questionário

A liberdade fornecida no ambiente de desenvolvimento FOSS permite que desenvolvedores permaneçam anônimos. No experimento realizado com sistemas FOSS, não houve sucesso no pedido (via e-mail) de preenchimento do *survey* por parte dos principais desenvolvedores, inviabilizando comparações entre as classificações do NEUROMINER e as classificações do questionário. Por esta razão, optamos por realizar um experimento controlado em um ambiente corporativo.

O experimento foi planejado para classificar programadores da indústria e submetê-los ao uso de uma ferramenta de compreensão de software que explora profundamente um dos seus sistemas representacionais. Desta forma, foi possível averiguar a relação entre a preferência de um programador por um sistema representacional e o efetivo uso de uma ferramenta que explora esse sistema. Além disso, com o questionário pronto, foi viável solicitar o preenchimento do mesmo pelos participantes, constituindo uma validação integrada das abordagens desenvolvidas nesta tese.

Na seleção da ferramenta de compreensão a ser utilizada, optamos por uma ferramenta construída no próprio Laboratório de Engenharia de Software da UFBA. Carneiro et al. desenvolveram esta ferramenta como um *plug-in*, para a plataforma Eclipse, cujo objetivo principal é auxiliar programadores no processo de visualização e compreensão de software (Carneiro et al., 2009). Chamada de *SourceMiner*, a ferramenta fornece várias formas não convencionais de visualização do código e explora essencialmente o sistema representacional visual. Além disso, o *SourceMiner* produz um *log* estruturado de cada tarefa realizada pelo programador ao utilizar a plataforma. Essas características possibilitaram a

análise do *log* gerado por essa ferramenta, para avaliação da efetividade de uso da mesma por um programador menos dependente do sistema visual.

O NEUROMINER foi usado para minerar os e-mails dos programadores selecionados, gerando suas classificações neuro-lingüísticas. Em seguida, os programadores foram submetidos a tarefas controladas de compreensão de código com o *SourceMiner*, como definidas em (Carneiro et al., 2010). O *log* também foi analisado, permitindo a caracterização de uso do SourceMiner por cada programador.

Por fim, o questionário desenvolvido nesta tese foi preenchido por cada programador. Os resultados apresentaram uma precisão de 83 por cento, quando comparamos as classificações do NEUROMINER com os resultados do questionário.

1.8 ORGANIZAÇÃO DA TESE

Este documento está organizado da seguinte maneira. Este capítulo discutiu a motivação principal para o trabalho proposto, além de descrever de forma geral como ele foi desenvolvido. O Capítulo 2 apresenta os conceitos básicos e os termos que serão usados no resto do trabalho. O Capítulo 3 contém a descrição completa da ferramenta proposta e dos estudos experimentais realizados nos dois projetos de software livre. O Capítulo 4 detalha o processo de desenvolvimento do questionário e de avaliação estatística do *survey* realizado com Engenheiros de Software. O Capítulo 5 descreve o estudo experimental controlado realizado na indústria para cruzar classificações da ferramenta com classificações do *survey*, bem como para comparar os perfis encontrados com o uso de uma ferramenta de visualização de software por estes perfis. Finalmente, no Capítulo 6, são apresentadas as considerações finais e trabalhos futuros.

A Figura 1 apresenta a interdependência entre os capítulos.

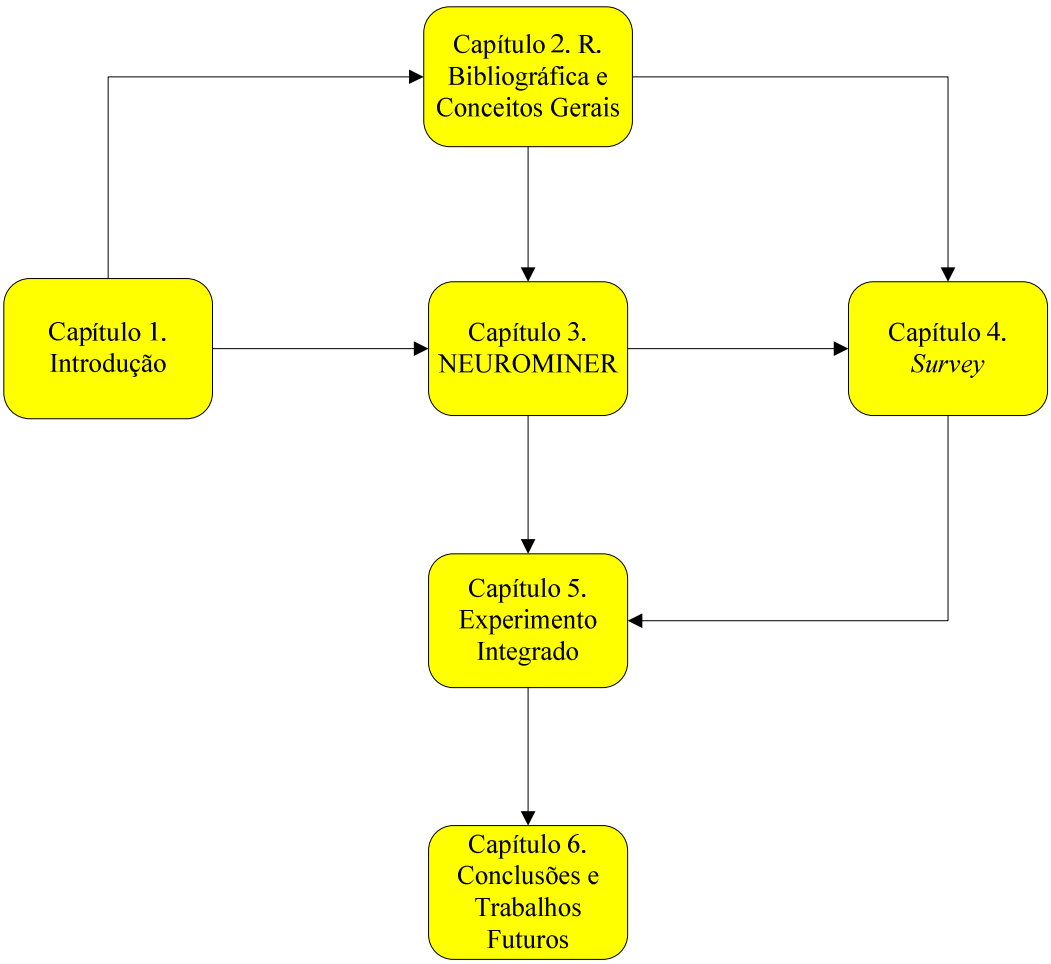

Figura 1. Interdependência entre os capítulos.

Capítulo 2

Neste capítulo, são apresentados os fundamentos computacionais e psicológicos básicos que nortearam a concepção do NEUROMINER. Além disso, o capítulo discute papéis de desenvolvedores em comunidades FOSS, conceitos utilizados em dois experimentos desta tese. Por fim, ele discute conceitos da Teoria de Reposta ao Item, utilizados para análise estatística do survey detalhado no capítulo 4.

2 REVISÃO BIBLIOGRÁFICA E CONCEITOS RELEVANTES AO TRABALHO

2.1 MINERAÇÃO DE TEXTOS E SUA APLICAÇÃO À ENGENHARIA DE SOFTWARE

Nos últimos dez anos, com a coleta intensiva de dados e conseqüente explosão de informações disponíveis em repositórios computacionais, a mineração de dados tem se apresentado como uma abordagem importante para descoberta de tendências e padrões previamente desconhecidos. No desenvolvimento de software, dados históricos dos projetos podem e devem servir de base para identificar, entre outros, padrões úteis de ocorrências de defeitos, associações, modificações, evolução e decomposição de sistemas. Por esta razão, pesquisadores da área têm desenvolvido abordagens para minerar repositórios a fim de melhor entender, predizer e planejar projetos de software (Mendonça e Sunderhaft, 1999).

Técnicas de mineração precisam ser desenvolvidas ou adaptadas para tarefas e domínios específicos. No desenvolvimento de software não é diferente, há a necessidade de novas abordagens para se minerar dados que são provenientes da construção de software. Com essas adaptações, é possível, por exemplo, extrair padrões do uso de uma API (Acharya

et al., 2007), predizer defeitos de software (Zimmermann et al., 2007) e estimar o esforço de desenvolvimento (Layman et al., 2008).

Entre as fontes de dados passíveis de análise podem ser citadas bases de código estático, histórico de versões do software, rastros de execução de programas, relatórios de erros, *logs* de implantação de sistemas e listas de discussão de projetos. Essa tese lidou particularmente com listas de discussão, as quais são uma rica fonte de informação sobre o processo de desenvolvimento e decisões dos projetos, bem como sobre as características de desenvolvedores que neles participaram.

Minerar listas de discussão é, essencialmente, minerar textos, cujos fundamentos remetem à área de recuperação de informação (RI), conceituada a seguir.

2.1.1 Recuperação de Informação

A área de Recuperação de Informação (RI) pesquisa formas de determinar a relevância de um objeto desestruturado, um documento textual, por exemplo. Um documento relevante é aquele que supre necessidade de informação do seu usuário (Manning et al., 2007). A Relevância é o que difere um Sistema de Recuperação de Informação de um Sistema de Recuperação de Dados (SRD) tradicional.

Uma consulta em um SRD baseia-se em termos e critérios booleanos simples, todavia, extrair informação sobre um tópico, baseado em termos, nem sempre trará somente bons resultados, ou seja, nem sempre trará resultados relevantes. Por exemplo, usar o termo *gramado* para recuperação de artigos sobre futebol pode surpreender o usuário com resultados irrelevantes sobre empresas de jardinagem.

O interesse maior é na recuperação de informação associada a documentos do que na recuperação dos termos presentes nos mesmos. Em (Rijsbergen, 1979), é apresentada uma Tabela com algumas diferenças entre a RI e a recuperação de dados (vide Tabela 1):

Tabela 1 – Recuperação de Dados x Recuperação de Informação

Características	Recuperação de Dados	Recuperação de Informação
Comparação	Exata	Aproximada
Dados	Fortemente estruturados	Fracamente estruturados
Inferência	Dedução	Indução
Modelo	Determinístico	Probabilístico
Linguagem de Consulta	Artificial	Natural
Especificação	Completa	Incompleta

2.1.1.1 Medidas de qualidade

Duas medidas são bastante utilizadas para avaliar a qualidade de um modelo usado para recuperação de informação: *Precision (P) e Recall (R)*, ou Precisão e Cobertura (Rijsbergen, 1979).

A Precisão descreve qual a fração de documentos retornados que são realmente relevantes. Atingir alta precisão é sempre muito importante, no entanto, muitos documentos importantes podem ser deixados de lado, mesmo quando se tem alta precisão para os documentos encontrados. Por esta razão, uma segunda medida é necessária, a Cobertura.

A Cobertura indica o percentual de documentos relevantes que foram encontrados. Em outras palavras, o modelo pode sempre acertar, ao recuperar um documento, mas pode deixar alguns documentos importantes para trás. A Cobertura mede exatamente isso, o percentual de documentos relevantes que foram cobertos pelo modelo.

Portanto, a qualidade de um modelo é somente assegurada quando se tem bons valores para as duas medidas. Considere, por exemplo, uma base de e-mails de onde queremos recuperar somente mensagens de interesse. Uma precisão de 100% com uma cobertura de 50% significa que recuperamos somente mensagens de interesse, mas deixamos metade das mensagens de interesse sem recuperar. Uma cobertura de 100% com uma precisão de 50%, significa que recuperamos todas as mensagens de interesse, mas junto com elas recuperamos um número igual de *spams*.

Essas medidas de qualidade e outros fundamentos básicos da área de RI são utilizados pela mineração de textos para obtenção de informação de qualidade (relevante) de um texto escrito em linguagem natural. Estes fundamentos são descritos a seguir.

2.1.2 Mineração de textos

A mineração de textos, tipo especial de mineração de dados, é uma tecnologia emergente para análise de grandes coleções de documentos não estruturados, visando a extração de padrões ou conhecimentos interessantes e não triviais dos mesmos (Visa, 2001).

Assim como a mineração de dados convencional, a mineração de texto possui etapas inerentes ao processo de descoberta de conhecimento (Fayyad et al., 1996). Neste trabalho, interessa-nos elucidar a fase de pré-processamento e a detecção de locuções substantivas, pois os fundamentos e conceitos destas atividades foram usados e adaptados para concepção do NEUROMINER, especialmente a representação de documentos através do modelo de espaço vetorial, descrita a seguir.

2.1.2.1 Pré-processamento

Uma vez montada a base com os textos a serem minerados, faz-se necessário converter cada documento para um formato compreensível por algoritmos computacionais. Essa tarefa é atribuída à etapa de pré-processamento.

A representação dos documentos em formato estruturado pode ser efetuada sob três óticas distintas: utilizando o modelo booleano, o modelo probabilístico ou o modelo vetorial.

O modelo booleano considera a consulta efetuada pelo usuário como uma expressão booleana convencional, a qual liga os termos através dos conectivos lógicos AND,

OR e NOT (Paice, 1984). De forma análoga, os documentos são representados por um conjunto de termos índices. Neste modelo, um documento é considerado relevante ou irrelevante para uma consulta através da aplicação de operadores lógicos.

No modelo booleano não existe resultado parcial e não há informações que permitam a ordenação do resultado da consulta, o que se caracteriza como uma desvantagem desse modelo. Atrelado a esse fato, convém destacar que o conhecimento sobre álgebra booleana não é comum a todos os usuários, constituindo-se também em um entrave no uso desse modelo.

O modelo probabilístico, por sua vez, baseia-se no Princípio da Ordenação Probabilística. Nesse modelo, busca-se saber a probabilidade de um documento D ser ou não relevante para uma consulta Q. Tal informação é obtida assumindo-se que a distribuição de termos na coleção é capaz de informar a relevância possível para um documento qualquer da coleção. Os termos extraídos dos documentos e das consultas não possuem pesos pré-definidos. A ordenação dos documentos é calculada pesando dinamicamente os termos da consulta relativamente aos documentos (Maron e Kuhns, 1960).

As principais desvantagens desse modelo são: (1) para várias aplicações, a distribuição dos termos entre documentos relevantes e irrelevantes não está disponível; e (2) o modelo define apenas uma ordenação parcial dos documentos.

O modelo vetorial vale-se da geometria para representação dos documentos. Introduzido por Salton et al., esse modelo foi desenvolvido para ser utilizado em um sistema de recuperação de informações chamado SMART. Segundo o modelo vetorial, cada documento é representado por um vetor de termos e cada termo possui um peso associado que indica seu grau de importância no documento (Salton et al., 1975). Em outras palavras, cada documento possui um vetor associado, o qual é constituído por elementos organizados por uma tupla de valores da forma: dj = {w1j ,.. , wij}, onde dj representa um documento e wij representa um peso associado a cada termo indexado de um conjunto de t termos do documento.

Cada elemento do vetor de termos é considerado uma coordenada dimensional. Desta forma, os documentos podem ser colocados em um espaço euclideano de n dimensões (onde n é o número de termos) e a posição do documento em cada dimensão é dada pelo peso

do termo associado e aquela dimensão. Na Figura 2, o DOC2, por exemplo, tem três termos de indexação e seus respectivos pesos.

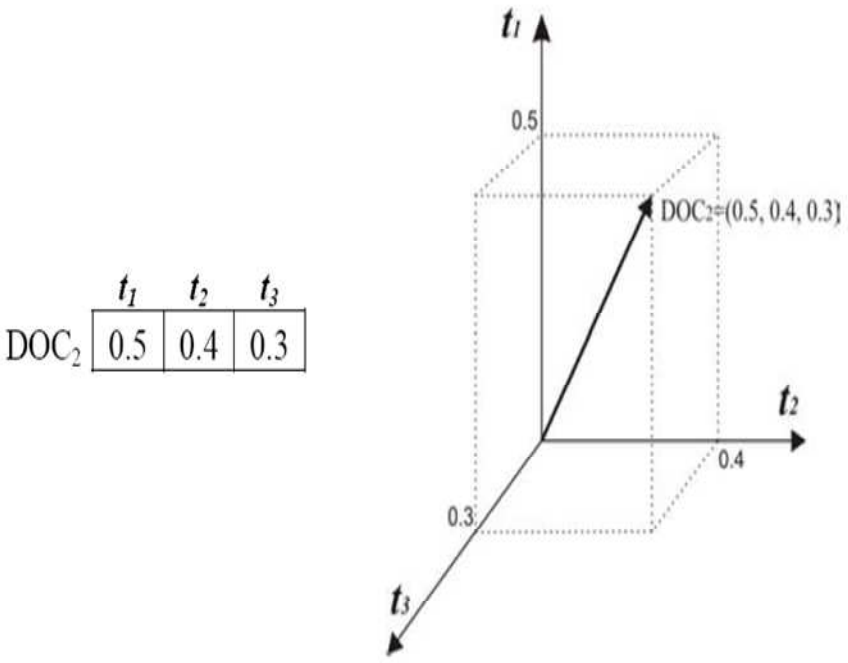

Figura 2. Modelo de espaço vetorial

No modelo do espaço vetorial, as consultas também são representadas por vetores. Assim, os vetores dos documentos podem ser comparados ao vetor da consulta e o grau de similaridade entre eles pode ser calculado. Os documentos mais similares (aqueles que apresentarem os vetores mais próximos ao vetor da consulta) são considerados relevantes, retornando como resposta para o usuário. Além disso, os documentos que apresentarem os vetores mais próximos são considerados similares entre si.

A montagem do vetor de termos segue as seguintes etapas:

Extração de termos

De uma forma geral, nem todos os termos que compõe um documento são relevantes quando se almeja extrair informações de alto nível. Assim, para compor o vetor de termos de um texto, é necessário identificar as palavras de forte conteúdo semântico,

selecionando apenas aquelas que realmente carregam em si um significado relevante para o propósito.

A atividade de extração de termos de um documento é composta de vários passos, contribuindo todos eles para o propósito final (Hiemstra e Jong, 2001). Enumeramo-los:

1. Análise Léxica: nem sempre o documento original se encontra em formato puramente textual. Em função disso, é necessário converter estes formatos, eliminando quaisquer atributos de formatação de apresentação para um formato padronizado.
2. Conversão de caracteres para maiúsculo ou minúsculo: este procedimento possibilita que palavras iguais, porém escritas com algum caractere em formato maiúsculo ou minúsculo diferente - a exemplo de neuro e Neuro possam ser interpretadas como o mesmo termo.
3. Uso de uma lista de palavras a serem desconsideradas: comumente chamadas de *stopwords*. Essa lista consiste em uma relação de palavras que não têm conteúdo semântico significativo (como por exemplo, preposições, conjunções, artigos, numerais e outros) e por conseqüência, não são relevantes na análise do texto.
4. Normalização morfológica: com o objetivo de agrupar termos com o mesmo significado conceitual, a exemplo das palavras computar e computação, pode ser aplicado um algoritmo de conversão de termos em radicais. No caso exemplificado, as palavras possuem o mesmo radical comput, e, por isto, podem ser resumidas a este termo.
5. Seleção de palavras simples ou compostas: em alguns casos, durante o pré-processamento do documento, várias palavras conjuntas (frases) podem ser tratadas como um termo único. Esta seleção pode ser feita a partir de listas pré-definidas de palavras ou técnicas estatísticas e sintáticas.
6. Normalização de sinônimos: palavras de mesmo significado podem ser reduzidas a um mesmo termo, a exemplo da sigla LES e a composição Laboratório de Engenharia de Software, as quais possuem o mesmo significado.
7. Análise estrutural: esta estratégia consiste em associar a cada termo informações relativas ao seu posicionamento na estrutura do documento, de forma a diferenciá-lo de um termo homônimo localizado em outra posição.

Atribuição de pesos

O processo de associar valores numéricos a cada termo previamente extraído é conhecido como atribuição de pesos. Em geral, a determinação do peso de um termo em um

documento pode ser efetuada mediante dois paradigmas (Sholom et al., 2005): (1) quanto mais vezes um termo aparece no documento, mais relevante ele o é para o tópico do documento; (2) quanto mais vezes um termo ocorre dentre todos os documentos de uma coleção, menos importante ele é para diferenciar os documentos.

Partindo deste princípio: duas são as abordagens passíveis de aplicação para cálculo de pesos:

1. Binário ou booleano - Os valores 0 e 1 são utilizados para representar, respectivamente, a ausência ou presença de um termo no documento.
2. Numérico – Baseia-se em técnicas estatísticas relacionadas à freqüência dos termos no documento.

Os pesos numéricos podem ser representados conforme as medidas a seguir:

Freqüência dos termos (*Term Frequency* - tf): Método simples, que consiste no número de vezes em que um termo wi ocorre em um documento d. Esse método está baseado na premissa de que a freqüência do termo no documento fornece informação útil sobre a importância desse termo para o documento. Para normalização dos dados, também é comum que o valor final da freqüência do termo seja dividido pelo número total de termos do documento. Isso equilibra o fato de existirem documentos extremamente maiores que outros (Pang-Ning et al., 2006).

Freqüência do documento (*Document Frequency* - DF): é o número de documentos no qual o termo wi ocorre pelo menos uma vez.

Freqüência Inversa do Documento (*Inverse Document Frequency* - idf): define a importância de um termo dentre um conjunto de documentos. Quanto maior for este índice, mais representativo é o termo para o documento ao qual o possui. A fórmula para cálculo da idf é:

$$\mathrm{idf_i} = \log \frac{|D|}{|\{d : t_i \in d\}|}$$

Onde |D| representa o total de documentos e $|\{d : t_i \in d\}|$ representa o número de documentos onde o termo ti aparece.

tf-idf: combina a freqüência de um termo com sua freqüência inversa de documento, a fim de obter um índice maior de sua representatividade. A fórmula para cálculo do peso tf-idf é:

$$(\text{tf-idf})_{i,j} = \text{tf}_{i,j} \times \text{idf}_i$$

Redução de dimensionalidade

Um problema central no processo de mineração de textos, independentemente da técnica a ser utilizada, consiste na grande dimensão do vetor de termos. Devido a esse fato, existem algumas técnicas que trabalham no intuito de reduzir a dimensão deste vetor, sendo estas classificadas da seguinte maneira (Sholom et al., 2005):

Seleção de Características – Visam remover termos não informativos dos documentos e construir um conjunto de termos ou radicais que facilite a identificação da categoria à qual o documento pertence. Tal ação almeja melhorar a eficácia da classificação destes através da redução da complexidade computacional por meio da redução do número de termos.

Extração de termos ou parametrização – Processo de construir novas características ou termos através de técnicas de combinação ou transformação dos termos originais.

2.1.2.2 Classes gramaticais e *noun phrases*

As palavras que possuem comportamento sintático e semântico similares podem ser agrupadas em uma mesma classe, originando as chamadas categorias sintáticas ou gramaticais, mais comumente denominadas *parts of speech* (POS) (Cunningham et al., 2002). As três principais são substantivo, verbo e adjetivo. Os substantivos referem-se a pessoas, animais, conceitos e coisas. O verbo é usado para expressar a ação numa sentença. Os adjetivos, por sua vez, expressam as propriedades dos substantivos.

Alguns softwares de domínio público identificam automaticamente a POS de cada palavra em uma sentença (http://gate.ac.uk/) (Cunningham et al., 2002). Esse tipo de software foi utilizado na realização deste trabalho, como será visto posteriormente.

A detecção de POS é importante, pois, em contextos específicos, duas ou mais palavras juntas, com categorias gramaticais diferentes, podem ter um único significado. A essa junção semântica de palavras, dá-se o nome de *Noun Prhases* (NPs) (Haythornthwaite e Gruzd, 2007), sintagma nominal ou locução substantiva.

Em NPs, o substantivo (*noun*) é o elemento central (*head*) que determina o caráter sintático da frase. Um adjetivo, por exemplo, pode ser um modificador (*mod*) desse substantivo.

Além de agrupar palavras dentro de um contexto, as NPs também podem melhorar a precisão de busca em um texto; para isto, faz-se necessário que um dicionário especifique que as palavras podem aparecer juntas. Em geral, não é necessário armazenar as palavras de forma composta, pois este processo exige tempo e não aumenta de forma considerável a eficiência do sistema. O que pode ser feito é o armazenamento da informação sobre a distância entre as palavras de um mesmo documento, deixando a técnica de consulta avaliar se as palavras são ou não adjacentes (Haythornthwaite e Gruzd, 2007).

A ferramenta elaborada nesta tese usou o conceito de NPs para relacionar e contextualizar a Neuro-Lingüística com a Engenharia de Software.

2.2 ESTILOS COGNITIVOS E ESTILOS DE APRENDIZADO

Na comunidade científica centrada na investigação cognitiva, é amplamente aceito que as estratégias escolhidas pelas pessoas para aprender têm um impacto sobre o desempenho da absorção de conhecimento (Cassidy, 2004). Existe uma vasta gama de definições, teorias, modelos, interpretações e medidas relacionadas com o processo de aprendizagem. Entre estas definições e interpretações, dois conceitos têm contribuído com informações valiosas para diversos campos de pesquisa: estilos de aprendizagem (Cassidy,

2004) e estilos cognitivos (Riding, 2000). Esta tese não pretende teorizar sobre esses conceitos ou até mesmo criar novas definições para os mesmos. As definições pré-existentes serão usadas para contribuir com a fundamentação teórica necessária para o trabalho aqui apresentado.

Os termos estilo cognitivo e estilo de aprendizagem têm sido definidos de diferentes maneiras por diferentes pesquisadores. Allport (Allport, 1937) descreveu estilo cognitivo como um hábito comum, ou uma maneira pessoal de pensar, perceber, lembrar e resolver problemas. Garity (Garity, 1985) pontuou que o termo estilo cognitivo tem sido usado para definir o processo cognitivo de pensar, perceber e recordar. Para Garity, estilo cognitivo é a forma como indivíduos processam a informação e como preferem aprender. Badenoch (Badenoch, 1986), em seu estudo sobre "o tipo de personalidade, preferências de estilo de aprendizagem e estratégias de ensino", fez uma correlação entre personalidade cognitiva e estilo de aprendizagem. Ele afirma que a teoria por trás dos estilos de aprendizagem investiga o processo de aprendizagem e seu produto. O objetivo é compreender as interações que ocorrem em um ambiente de aprendizagem e os resultados concretos dessas interações. Em sua opinião, o tipo de personalidade cognitiva, no entanto, é uma classificação da teoria do estilo de aprendizagem. Hartley (Hartley, 1998) segrega explicitamente os dois conceitos e define estilos cognitivos como as formas que as pessoas conduzem suas tarefas cognitivas e estilos de aprendizagem como as formas que as pessoas conduzem suas tarefas de aprendizagem.

Como pode ser notado, a leitura de artigos e referências diversas sobre os conceitos pode causar certa nebulosidade no entendimento da diferença dos mesmos. Esta tese adota a separação dos conceitos (Hartley, 1998) e considera que estilos de aprendizagem são as maneiras que cada indivíduo utiliza para entender alguns assuntos. Por outro lado, estilos cognitivos estão relacionados com as maneiras pelas quais indivíduos processam e transformam uma informação em conhecimento internalizado, recordando-a quando necessário. Cada indivíduo utiliza estilos de aprendizagem para entender e compreender um assunto, no entanto, a fim de processar e transformar as informações aprendidas em conhecimento facilmente recuperável, ou mesmo para melhor assimilar e aperfeiçoar o assunto aprendido, os estilos cognitivos são utilizados.

Em resumo, há uma diferença tênue entre aprendizagem e cognição, ou aprendizagem e apreensão, entre os estilos de "aquisição" de conhecimento e estilos de "uso, otimização e transformação" do conhecimento adquirido. Partindo desta separação de conceitos e fazendo uma analogia com a Engenharia de Software, nesta pesquisa, a princípio, estamos interessados em investigar como os Engenheiros de Software aprendem sobre um software, quais os seus estilos de aprendizagem preferidos.

As estratégias utilizadas para tentar capturar estilos de aprendizagem e estilos cognitivos possuem uma base científica comum, a qual apresenta algumas dimensões e formas de medir estilos cognitivos. Algumas dessas formas são discutidas a seguir.

2.2.1 Dimensões e medidas representativas de estilos cognitivos

Nos anos 70, os pesquisadores da Psicologia Cognitiva intensificaram as discussões sobre como medir as habilidades intelectuais de indivíduos. Hunt, Frost, e Lunneborg (Hunt et al., 1973) propuseram a utilização de exames laboratoriais para investigar a construção e o uso de tais capacidades humanas. Estes (Estes, 1974) propôs testes para medir as capacidades cognitivas como um meio de encontrar maneiras de melhorar o desempenho cognitivo. Underwood aplicou testes para detectar as diferenças entre indivíduos e utilizar essa informação como base para *Nomethetics*, uma teoria da psicologia que analisa o conhecimento do paciente sobre a doença e a influência disto no seu processo de auto-cura (Underwood, 1975).

Inicialmente, a medição de estilos cognitivos baseava-se em medir fortemente o *quanto*, dando pouca atenção em medir o *como* foi feito (Lohman e Bosma, 2002). No entanto, do ponto de vista de estilos cognitivos, é pouco relevante, por exemplo, medir apenas "quantas horas foram gastas em uma tarefa". A medição de estilos cognitivos deve ser fortemente baseada em *como* as tarefas propostas foram feitas. Para isso, Lohman e Bosma (Lohman e Bosma, 2002) propõem as seguintes premissas:

(1) Aplicar tarefas em que as diferenças individuais são claramente refletidas, principalmente nas medições do "como foi feito". Ou seja, as resoluções das tarefas propostas são avaliadas de acordo com as diferentes soluções estratégicas tomadas.

(2) Ter uma orientação que torne claras as inferências sobre as estratégias de respostas dadas para cada tarefa. Mesmo diante de maneiras diferentes de resolver algumas tarefas, as formas encontradas para resolvê-las devem ser correlacionadas com outras medidas dependentes de um indivíduo tais como a velocidade de raciocínio e o reflexo.

(3) O Modelo de Medição deve capturar o perfil do indivíduo, suas estratégias para resolução de problemas e as medidas correlacionadas (Lohman e Ippel, 1993). A fim de garantir a coerência entre diversos experimentos que podem ser realizados, o modelo deve ser validado e empacotado (padronizado) para reutilização.

(4) É necessário que o Modelo de Medição contemple a análise do relacionamento entre as diferentes estratégias.

Ao tentar medir um estilo cognitivo, as estratégias avaliadas não podem ser classificadas como pertencentes a uma categoria ou estilo específico. Por exemplo, o fato de um Engenheiro de Software preferir usar diagramas em uma situação em que o mesmo precisa compreender uma classe, não implica que há uma preferência visual em geral. Em outras palavras, nem todas as estratégias representam um estilo com a mesma clareza (Lohman e Bosma, 2002).

Riding e Cheema (Riding e Cheema, 1991), criadores da *Cognitive Style Analysis* (CSA), descrevem que os estilos cognitivos têm duas dimensões fundamentais e independentes: a *Wholist-Analytic* (WA) e a *Verbalizer-Imagery* (VI). A dimensão WA foca na forma habitual adotada por um indivíduo no processamento e organização das informações. Alguns indivíduos processam e organizam as informações em detalhes (*analytics*), enquanto outros têm uma visão global ou geral da informação (*wholists*).

A dimensão VI descreve as formas mais comuns de **representação da informação na memória**. Estas formas são acionadas no momento que o indivíduo pensa sobre alguma coisa. Para Riding (Riding, 1991), *verbalizers* convertem as informações lidas, vistas ou ouvidas, em palavras ou associações verbais. *Imagers*, por outro lado, convertem freqüentemente essas informações em imagens mentais espontâneas.

Segundo Riding (Riding, 1994), a validade do modelo cognitivo proposto na abordagem CSA é sustentada pela evidência de que as dimensões WA e VI são independentes uma da outra, bem como independentes da inteligência. Todavia, estas dimensões interagem com a personalidade e estão relacionadas a preferências e desempenho de aprendizagem, preferências gerais de cada pessoa e seu comportamento social.

A abordagem CSA foi base para o trabalho de Fleming (Fleming, 1995), principal influência para o *Survey* apresentado no Capítulo 4. O trabalho de Fleming propõe um questionário para classificar pessoas de acordo com os seus estilos de aprendizagem. Assim como esta pesquisa, além da CSA, Fleming (Fleming, 1995) também foi influenciado pela Neuro-Lingüística (Bandler e Grinder, 1979), correlacionando palavras ou expressões utilizadas por indivíduos com as suas formas preferidas de **representação da informação na memória (Sistema de Representacional Preferencial)**. São considerados outros sistemas de representação, estendendo a dimensão VI da CSA. Além de representações na forma de palavras (V) e imagens (I), são considerados os sistemas representacionais auditivo e cinestésico.

Esta tese propôs um questionário especificamente concebido para capturar o Sistema Representacional Preferencial (SRP) de Engenheiros de Software. Definir os SRPs, no contexto da Engenharia de Software, é definir os estilos que os Engenheiros de Software usam para se comunicar e aprender sobre um sistema. Em outras palavras, os estilos de aprendizagem indicam os sistemas de representação que são mais acionados na hora de entender um assunto relacionado a um software. A próxima seção introduz a Programação Neuro-Lingüística e detalha o conceito de SRP utilizado nesta pesquisa.

2.3 PROGRAMAÇÃO NEURO-LINGÜÍSTICA

A Programação Neuro-Lingüística (PNL), criada na década de 1970 (Bandler e Grinder, 1975)(Bandler e Grinder, 1979), alcançou considerável popularidade como uma abordagem para comunicação e desenvolvimento pessoal. O termo PNL é exaustivamente usado nas áreas de educação, gestão e treinamento. Contudo, apesar de terem sido publicadas evidências da PNL como modelo para compreensão e aprendizado (Tosey e Mathison, 2003), existem poucos trabalhos acadêmicos sobre o assunto.

A PNL foi desenvolvida por Richard Bandler e John Grinder. Bandler, cuja experiência era em matemática, programação de computadores e gestáltica, foi estudar na Universidade de Santa Cruz na década de 70, onde desenvolveu uma efetiva colaboração com John Grinder, um professor de lingüística. Bandler inferiu que os princípios de padrões e modelos que ele aplicava na programação de computadores poderiam ser usados para modelar comportamentos de pessoas que obtiveram sucesso. Com este tirocínio, ele percebeu que os modelos precisariam de base lingüística, a qual ele foi buscar com Grinder.

Para Grinder e Bandler, o nome Programação Neuro-Lingüística derivou-se dos estudos dos padrões ou programações criadas pela interação entre o cérebro (neuro), a linguagem (lingüística) e o corpo. Em outras palavras, a mente de um indivíduo é formada de conexões padronizadas entre processos neurológicos, linguagem e estratégias comportamentais de aprendizado (programação) (Dilts et al., 1980).

As definições para a PNL são as mais variadas: por exemplo, na literatura promocional, encontrada em centenas de sites na internet, a mesma é conceituada como a “arte da excelência na comunicação”. Por outro lado, seus criadores a conceituam como “o estudo da experiência subjetiva e o que pode ser calculado disso” (Dilts et al., 1980). O objetivo é analisar o impacto da comunicação verbal e não-verbal sobre a realidade intrapessoal e a qualidade da comunicação. Considerada como ciência aplicada pelos seus defensores, a PNL supostamente oferece procedimentos específicos e eficazes nos campos da educação, treinamento e administração. Em se tratando de relacionamentos, buscam-se os melhores tipos de interação e cooperação, baseados em como as pessoas são e não como se pensa que devem ser.

Entre as principais características da PNL está a orientação para o aprendizado como chave para mudança e desenvolvimento pessoal, pressupondo que as pessoas são intrinsecamente criativas e capazes, agindo de acordo com a forma que entendem e representam o mundo e não de acordo como o mundo é. A literatura constantemente cita a afirmação de Korzybski de que o "o mapa não é o território" (Korzybski, 2009), uma referência ao entendimento individual que cada pessoa tem – modelo mental – de acordo com a sua experiência, crenças, cultura, conhecimento e valores.

Em (Tosey e Mathison, 2009), Tosey & Mathison apresentam a PNL sob uma perspectiva epistemológica, com princípios científicos que muitas vezes não são apresentados em sua literatura promocional. Os primeiros trabalhos publicados por Bandler e Grinder (Bandler e Grinder, 1979)(Dilts et al., 1980) foram baseados nos modelos de Fritz Perls, fundador da Gestáltica, Virginia Satir, pesquisadora em terapia de família, e Milton Erickson, doutor em medicina, mestre em psicologia e hipnoterapeuta de reconhecimento mundial. Conseqüentemente, a visão epistemológica da PNL fortalece sua teoria sobre os processos através dos quais os indivíduos podem perceber, conhecer e aprender (Tosey e Mathison, 2003). Parte dessa teoria foi explorada nesta tese: a utilização de um sistema de representação preferencial para cognição.

De fato, a concepção de que existem diferentes formas de representação para cognição já é aceita há muito tempo na área de psicologia (Dent, 1983). Na seção anterior, vimos que a forma de representação é uma das dimensões fundamentais e independentes da análise de estilos cognitivos.

Para PNL, processos mentais internos tais como a resolução de problemas, uso da memória e linguagem consistem de representações visuais, auditivas e cinestésicas (VAC) que são formadas quando as pessoas pensam sobre tarefas e atividades diversas ou estão praticando as mesmas. Representações sensoriais internas estão constantemente sendo formadas e ativadas, possuindo um impacto sobre o desempenho de compreensão de uma pessoa, quer seja em uma conversa, escrevendo sobre um problema ou lendo um livro.

Esse sistema representacional ativado e utilizado é altamente contextualizado, ou seja, varia com a situação. O contexto influencia diretamente no sistema que está sendo usado (Einspruch e Forman, 1985). Por esta razão, algumas pessoas, em contextos específicos, podem preferir se comunicar e/ou aprender em um dos sistemas básicos ou através de uma

combinação dos mesmos (Dent, 1983)(Matthews, 1991). Os sistemas representacionais básicos são nomeados de acordo com o tipo de processamento efetuado. São eles:

1. Visual, que envolve a criação de imagens internas e a utilização de visões ou observações das coisas, incluindo fotos, diagramas, demonstrações, exposições, folhetos, etc.
2. Auditivo, o qual envolve lembranças de sons e transferência de informação através da escuta.
3. Cinestésico, que envolve as sensações internas de toque, emoções, experiências físicas, o realizar para entender e a prática. O termo *cinestésico* não deve ser confundido com o termo *sinestésico*, o qual significa a união de sensações. Algumas pessoas relatam já ter experimentado a *sinestesia*, como, por exemplo, olhar para uma cor e sentir um gosto específico.

Além do uso de um SRP (Sistema de Representação Preferencial) em contextos específicos, os criadores da PNL defendem que as pessoas dizem palavras e frases sensoriais, também chamadas de pistas verbais ou predicados, as quais indicam um processamento visual, auditivo ou cinestésico (Dilts et al., 1980). Em outras palavras, as frases que uma pessoa escolhe para descrever uma situação podem ser específicas para um sistema representacional (*sensory-based*) e indicarem como a consciência dessa pessoa está naquele momento. Os predicados indicam que porção – das representações internas – a pessoa trouxe da consciência (Dilts et al., 1980). Estas pistas podem nortear uma comunicação eficiente ou simplesmente passarem despercebidas.

Um dos grandes problemas da comunicação, seja ela utilizada em uma conversa informal ou para expor a funcionalidade de uma classe de software, é a falta de habilidade para apresentar um conteúdo para quem está lendo ou ouvindo. Muitas vezes, quem recebe a mensagem não sabe e não consegue absorver o que está sendo transmitido. Para PNL, uma das maneiras de otimizar a comunicação é identificar o sistema representacional que está sendo utilizado pelo indivíduo e, através de um processo chamado *matching*, utilizar o mesmo sistema para a construção de empatia. O *matching* consiste em identificar os predicados que indicam um sistema representacional e utilizar predicados do mesmo sistema para melhor comunicação com o seu interlocutor (Dilts et al., 1980).

Para exemplificar, num processo de *matching*, para uma frase como, “você viu a lógica dos algoritmos que lhe mostrei?”, uma resposta coerente seria: “ainda não, mas agora

que arrumei meus documentos e deixei minha mesa com uma visão limpa, vou examiná-los com toda a atenção". As palavras sensoriais "viu" e "mostrei" da primeira frase indicam um processamento visual, desta forma, a resposta usou o mesmo sistema através das palavras sensoriais visuais "visão limpa" e "examiná-los".

Esta pesquisa lida apenas com a identificação de palavras sensoriais e classificação de sistemas representacionais preferenciais de programadores, avaliando estes princípios da PNL, em atividades de engenharia de software. Vale ressaltar que o Sistema de Representação Preferencial está diretamente relacionado com o conceito de estilo de aprendizagem (vide seção 2.2), estudado e evidenciado em vários artigos da área de psicologia.

2.3.1 Críticas à Programação Neuro-Lingüística

A literatura sobre PNL em periódicos acadêmicos é mínima. Praticamente não existe pesquisa publicada de como a PNL é usada na prática (Thompson, Courtney e Dickson, 2002). A pesquisa experimental existente consiste de estudos laboratoriais executados nas décadas de 80 e 90, os quais investigaram duas noções particulares da PNL, o modelo de movimento dos olhos (Bandler e Grinder, 1979) e o conceito de SRP (Grinder e Bandler, 1976).

Heap (1988) e Sharpley (1987), em particular, têm argumentado que, com base nos estudos existentes, as alegações da PNL não podem ser aceitas. Heap, por exemplo, realizou uma pesquisa de campo baseada em várias dissertações de mestrado e monografias de pós-graduação sobre conceitos PNL, concluindo com uma forte crítica às suas afirmações. Todavia, o trabalho de meta-análise de Heap baseou-se apenas nos resultados destas dissertações, sem uma avaliação crítica de suas metodologias e validades.

Einspruch e Forman (1985) e Bostic St.Clair e Grinder (2001) também argumentaram que estudos como os de Heap são caracterizados por problemas que afetam a sua confiabilidade, incluindo a compreensão inexata dos conceitos da PNL e procedimentos

inválidos, uma vez que os estudos foram feitos por pesquisadores que não tinham treinamento e experiência nas técnicas PNL que foram testadas. Em verdade, Heap apresenta apenas uma conclusão provisória sobre a PNL, reconhecendo que testes da eficácia de terapias com PNL realizados com profissionais experientes e treinados ainda não foram devidamente realizados (Heap, 1988).

Tosey & Mathison afirmam que o atual corpo de pesquisa experimental não pode embasar conclusões definitivas sobre a PNL (Tosey e Mathison, 2009). Fica claro que não há apoio substancial para a PNL neste corpo de pesquisa experimental, mas o mesmo também é insuficiente para descartar a mesma. Neste contexto e em segundo plano, esta tese de doutorado ajuda a fortalecer o embasamento experimental para rejeição ou aceitação de uma das técnicas PNL.

2.3.2 Relação com a Área de Comunicação Colaborativa

As pesquisas na área de comunicação organizacional têm apresentando evidências de que, no processo de comunicação, os interesses individuais das pessoas envolvidas são mais importantes do que o meio utilizado para comunicação (Lewis, 2006).

Para Bogarosh (2008), os interesses individuais de cada colaborador tornam-se os interesses do grupo, além disso, uma crença de que todos os interesses podem ser realizados deve ser mantida, colocando todos do mesmo lado de uma solução. Neste processo, o gerente deve atuar como um líder e com um facilitador que coordena os interesses conflitantes, ao invés de controlá-los. Em outras palavras, um líder deve facilitar, ao invés de participar como parte interessada.

Esta tese não lida com a identificação, estimulação ou caracterização de interesses individuais de desenvolvedores de software. No nosso contexto, partimos da premissa de que há interesse em aprender ou compreender o software. A partir disso, hipotetizamos que há preferência de estratégia na hora de aprender ou compreender. Se uma pessoa tem interesse em aprender algo, independente das variáveis que a levaram a isso, a mesma pode ter

preferências contextuais para o processo de aprendizagem. Preferências estas, aceitas pela área de psicologia (Dent, 1983)(Matthews, 1991)(Peters et al., 2008). Por fim, consideramos a especificidade da Engenharia de Software de uso de artefatos visuais, textos, código-fonte e narrativas. Em outras palavras, além da relação entre os Engenheiros de Software, existe a relação entre os mesmos e os artefatos usados, os quais fazem uso de diferentes sistemas de representação.

Esta subseção, junto com as três seções anteriores, conclui a explanação dos conceitos necessários à compreensão do trabalho desenvolvido. As duas próximas seções apenas tratam de temas relevantes à compreensão da instrumentação dos estudos realizados. Elas podem ser desconsideradas sem prejuízo à compreensão holística da tese.

2.4 PAPÉIS DOS COLABORADORES DE PROJETOS FOSS

Em um de seus experimentos, este trabalho lidou com a classificação Neuro-Lingüística de engenheiros de software de duas comunidades de desenvolvimento de código aberto (FOSS – *Free Open Source Software*). Nessas comunidades, o estilo de desenvolvimento influencia as atribuições de cada colaborador, evidenciando a presença de desenvolvedores mais importantes que lideram o número de alterações e produção de código.

A maioria das comunidades FOSS adota o modelo Bazar de desenvolvimento. Para Raymond (Raymond, 1998), o estilo de desenvolvimento pode ser definido como Catedral ou Bazar. O estilo catedral é centralizado, organizado e utilizado por um grupo de desenvolvedores, sendo tipicamente adotado no desenvolvimento de softwares proprietários. No Bazar, os desenvolvedores estão dispersos geograficamente e o código é desenvolvido de forma totalmente aberta e pública, utilizando a Internet. A idéia central é garantir o sentido global e sinérgico postulado pela Teoria Geral dos Sistemas (Bertalanffy, 1969), exigindo a contribuição de todos para atender a um projeto com qualidade.

Mesmo sendo caracterizados pela descentralização e informalidade, projetos FOSS maduros possuem papéis bem definidos entre seus membros e, como todo projeto,

requerem coordenação. Para Taurion (Taurion, 2004), nesses projetos, as decisões são tomadas baseadas na meritocracia, ou seja, indivíduos com mérito, por agregarem maior valor ao projeto, destacam-se, tendo uma participação maior nas suas decisões.

Comumente, a partir da agregação de valor ao projeto, as comunidades em geral classificam os seus membros como Centrais (*Core*) e Periféricos (*Peripheral*) (Lave e Wenger, 1991). Em comunidades on-line, os membros *Core* são aqueles que se adaptam rapidamente ao ambiente virtual e executam diversas tarefas para o crescimento e qualidade do projeto (Zhang e Storck, 2001). Por outro lado, existem membros que possuem dificuldade de adaptação ou raramente são ouvidos pelos outros membros. Mesmo quando a interação é feita em ambientes relativamente fechados, tais como listas de distribuição de e-mail, apenas uma pequena parcela de membros *Core* participa regularmente das principais decisões de projeto (Finholt e Sproull, 1990).

Entre os que possuem pouca atividade no projeto, existem aqueles que são totalmente invisíveis para os outros e os que participam do projeto com uma freqüência menor, mas durante um longo período de tempo. Esses membros são reconhecidos pela comunidade e chamados de periféricos (Zhang e Storck, 2001).

Essa classificação em comunidades virtuais recebeu o nome de modelo cebola (*Onion Model*) (Crowston e Howison, 2005), indicando a existência de níveis concêntricos de contribuição (vide Figura 3). Um pequeno grupo de desenvolvedores, chamados de *Core developers*, localizados no núcleo, faz a maior parte do trabalho. Dois grandes grupos contribuem menos, porém, por um longo período de tempo: os *Co-developers*, atuando com correções de erros, documentação, etc; e os *Active Users*, apresentando relatórios de problemas a partir do uso do software. Há ainda o maior grupo, *Passive Users*, formado por pessoas que apenas usam o software e não dão nenhum tipo de *feedback*.

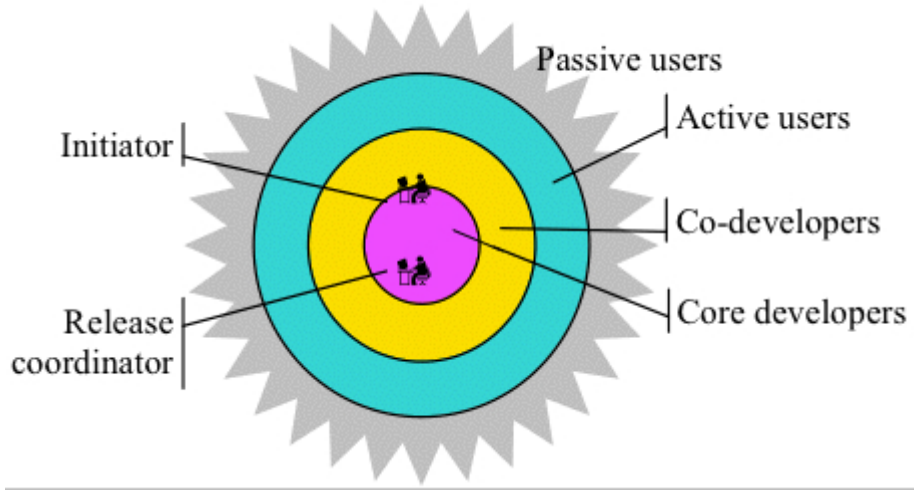

Figura 3. Estrutura da equipe de desenvolvimento de projetos FOSS (Crowston e Howison, 2005)

Como a maior parte do trabalho é feita por uma equipe central, é importante para os projetos manter esta equipe ativa e motivada. Alguns projetos são capazes de manter sua equipe principal com poucas ou nenhumas mudanças em toda a sua história, enquanto outros experimentam uma sucessão de diferentes gerações de desenvolvedores no seu núcleo (Robles et al., 2009)(Robles e Gonzalez, 2006).

Analisar os perfis psicológicos desses desenvolvedores, sob diversas perspectivas, pode ajudar a entender como manter a equipe motivada, enfatizando a empatia nas relações e a possibilidade de realocação de atividades entre os membros que mais contribuem, de acordo com esses perfis.

Os experimentos realizados com projetos FOSS nesta tese fizeram uma análise Neuro-Lingüística dos principais e maiores modificadores de código (top committers) desses projetos, membros centrais (Core), bem como buscou analisar o perfil da maioria da comunidade, essencialmente formada por membros periféricos.

2.5 TEORIA DE RESPOSTA AO ITEM

Esta seção introduz a Teoria de Resposta ao Item (TRI), base de nossa análise estatística aplicada no *survey* apresentado no Capítulo 4. Os conceitos aqui apresentados podem ser úteis para outros experimentalistas da área de Engenharia de Software que desejam criar e aplicar questionários para populações heterogêneas.

As teorias sobre testes estudam técnicas de modelagem estatística para medir o nível de propriedades subjetivas de indivíduos por meio de testes ou questionários. Neste contexto, a Teoria Clássica dos Testes (TCT), apesar de apresentar algumas fragilidades, é o modelo mais usual para medição de traços latentes de indivíduos em ciências sociais, psicologia, educação e áreas afins (Singh, 2004). A TCT é dependente dos indivíduos que participaram da pesquisa, além de focar suas análises nos escores individuais (total de pontos dos examinados nos testes). Isto torna o teste uma análise isolada não comparável, quando aplicado a uma população com características diferentes. Por esta razão, pesquisadores das áreas de Estatística, Psicologia e Educação conduziram pesquisas para resolver este problema (Karim, 2010).

A Teoria de Resposta ao Item (TRI) surgiu na área da psicometria como um instrumento para medir variáveis subjetivas e fornecer estatisticamente o nível de certas propriedades psicológicas do indivíduo. O enfoque está no nível da variável de interesse no individuo (Baker, 2001).

Na TRI, as estatísticas dos itens (questões) são independentes do grupo examinado e as pontuações são independentes da dificuldade do teste utilizado para descrever as habilidades dos indivíduos. O objetivo principal é exprimir, em primeiro lugar, os níveis dos itens (questões) ao invés do nível do teste. Por estes motivos, não há variação quando o modelo é aplicado em diferentes populações, seja em cultura, gênero, nível sócio-econômico ou grupo étnico (Baker, 2001). A TRI também tem vantagens em relação aos tamanhos de amostra, permitindo estimativas válidas para os parâmetros das questões mesmo em amostras relativamente pequenas.

Esta abordagem é amplamente utilizada em exames que são aplicados repetidamente para populações heterogêneas, como o *Graduate Record Examination* (GRE) e

o Teste de Inglês como Língua Estrangeira (TOEFL). Embora seja estudada há muitos anos, no Brasil, suas aplicações surgiram há pouco mais de uma década, como são os casos, por exemplo, do Sistema de Avaliação da Educação Básica (SAEB) e do Exame Nacional do Ensino Médio (ENEM) que utilizam a Teoria da Resposta ao Item em suas formas avaliativas.

Conceitos por trás da TRI

Modelos estatísticos têm suas aplicações e propriedades específicas para cada tipo de dados e análise para que se propõem. A Teoria de Resposta ao Item tem sua especificidade em medir variáveis subjetivas que não são aparentes no indivíduo, as quais não podem ser conhecidas em si, mas apenas por seus efeitos. Chamadas de variáveis latentes, estas variáveis são medidas por outras variáveis secundárias que são observáveis e que, de alguma forma, indicam o nível do traço latente avaliado (Rizopoulous, 2006).

Uma variável latente pode ser uma habilidade, capacidade, nível de satisfação, nível de proficiência em leitura, saúde mental, nível de stress, sistema de representação utilizado para compreensão de um software (objeto desta tese), entre outros. Para medir uma variável latente, geralmente é desenvolvido um teste com uma série de perguntas ou itens que expressam algum aspecto da variável subjetiva (Karim, 2010).

A primeira unidade de análise da TRI é o item (questão). O objetivo principal consiste em estimar os parâmetros dos itens e extrair as suas propriedades métricas, ao invés de ter um teste padronizado. Estes parâmetros medem quanto o item (questão) contribui, de fato, para avaliar a variável latente.

No modelo, as pontuações não são determinadas pela contagem simples das respostas que se referem à variável latente medida, mas pela comparação de diferentes funções de "conteúdo de informação" de cada item (questão), para diferentes grupos. Além disso, para estimar a variável latente, são utilizados apenas os itens respondidos. Todas essas características tornam possível a comparação de indivíduos em momentos diferentes.

Variáveis Latentes

A estimativa da variável latente é um valor representado pela letra grega *theta* (θ), calculado para cada um dos indivíduos examinados. A escala desta variável é representada em desvios-padrão (métrica do escore padrão). Baker afirma que, independentemente da habilidade, a mesma pode ser medida em uma escala com um ponto médio zero, uma unidade de medida e um intervalo de menos infinito a mais infinito (Baker, 2001). Em termos práticos, costuma-se utilizar média = 0 e desvio padrão = 1, limitando o intervalo em valores de -4 a + 4 ou -3 a +3 desvios padrões. Entre -3 e +3, por exemplo, estão 99,97% de todos os indivíduos de uma população. Nesta tese, será adotado o intervalo de -4 a +4, para garantir que, em termos práticos, nenhuma informação seja perdida.

Na TRI, o objetivo não é saber quantos itens o indivíduo acertou e, sim, por que ele acertou ou errou cada item individualmente. Desta forma, o interesse é descobrir qual é o tamanho de θ que o indivíduo deve ter para poder acertar o item (questão) (Reeve e Fayers, 2005). Um indivíduo que tem maior aptidão, isto é, que possui um nível mais elevado do traço latente que um dado item mede, terá uma probabilidade maior de acertar este item do que um sujeito com nível inferior de aptidão.

Se o traço latente é expresso como θ, então esta probabilidade de acerto é definida como $P_i(\theta)$, que se lê como: a probabilidade *(P)* de acertar o item (*i)* dado um tamanho de theta (θ). Assim, o sujeito com menor habilidade terá uma $P_i(\theta)$ pequena, enquanto um de aptidão superior terá tal probabilidade bem maior. Desta forma, a $P_i(\theta)$ de acertar um dado item vai de 0 a 1, onde a mesma será 0 para o sujeito que não tiver nenhuma aptidão que o item mede e 1 para o sujeito que tem uma aptidão teta ótima. Esta situação faz com que, à medida que cresce o tamanho do θ, vai crescendo também a $P_i(\theta)$, provocando visualmente uma curva de tipo S na escala de aptidão (Karim, 2010), como mostra a Figura 4. Esta curva é chamada de Curva Característica do Item (CCI) e é utilizada para estimação de parâmetros que avaliam a dificuldade, probabilidade de acerto ao acaso e o poder de discriminação dos itens do teste.

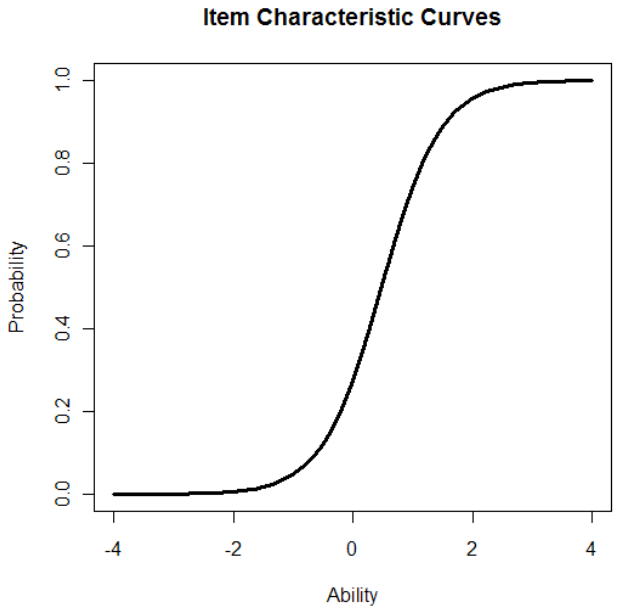

Figura 4. Curva Característica de um Item para modelos dicotômicos

Em se tratando de subdivisões, a TRI é dividida pelas dimensões consideradas, forma dos itens e pelo número de parâmetros dos itens a serem estimados. Quanto à dimensionalidade, o modelo TRI adotado pode ser uni ou multidimensional. Modelos multidimensionais ainda são objeto de pesquisa. A maioria das aplicações TRI pressupõe que há apenas uma aptidão dominante para executar uma atividade específica (unidimensionalidade), embora se saiba que as ações humanas são o resultado de uma série de motivações. O modelo unidimensional considera que a variável latente medida é a única responsável pelo desempenho do conjunto de itens do teste (Karim, 2010). Outro pressuposto intimamente ligado à unidimensionalidade é o da independência local, que afirma que todos os itens do teste são independentes uns dos outros.

Em relação à forma dos itens, existem dois tipos:

(1) Dicotômicos: quando as respostas são do tipo certo ou errado ou tem versus não tem. Ou seja, as respostas têm valores binários.

(2) Politômicos: quando as respostas são n-árias. Há dois principais grupos no presente caso. Modelos de respostas que assumem uma determinada ordem,

conhecidas como respostas graduadas, e os modelos de respostas nominais, os quais não impõem uma ordem para as respostas.

Estimativas de Parâmetros para as Variáveis Latentes

Em relação ao número de parâmetros a serem estimados, tem-se (Reise et al., 2005):

(1) Modelo logístico de um parâmetro ou modelo Rasch: um modelo que segue a premissa de que a probabilidade de sucesso de um item é influenciada apenas pela sua dificuldade, ou seja, este modelo tem apenas o parâmetro de dificuldade.
(2) Modelo logístico de dois parâmetros: neste modelo, além da dificuldade, a probabilidade de acerto de um item é influenciada pela sua discriminação.
(3) Modelo logístico de três parâmetros: assume que a probabilidade de acerto de um item é influenciada pela dificuldade, discriminação e acerto ao acaso.

Para cada um desses modelos existem diferentes funções logísticas da Curva Característica do Item (CCI). Na Figura 5, o gráfico ilustra a localização dos parâmetros na CCI.

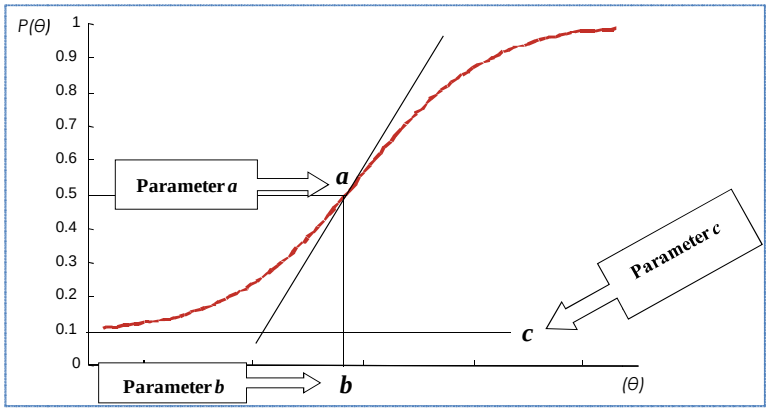

Figura 5. Localização dos parâmetros na CCI

As equações abaixo determinam $P_i(\theta)$ para cada um dos modelos (com um, dois e três parâmetros):

(1) Um parâmetro:

$$P(\theta) = \frac{e^{D(\theta - b_i)}}{1 + e^{D(\theta - b_i)}}$$

(2) Dois parâmetros:

$$P(\theta) = \frac{e^{Da_i(\theta - b_i)}}{1 + e^{Da_i(\theta - b_i)}}$$

(3) Três parâmetros:

$$P(\theta) = c_i + (1 - c_i)\frac{e^{Da_i(\theta - b_i)}}{1 + e^{Da_i(\theta - b_i)}}$$

Onde:

P(θ): a probabilidade de acertar o item i para um valor de θ específico;

θ: valor da variável medida;

b_i: índice de dificuldade do item i;

a_i: índice de discriminação do item i

c_i: índice de acerto ao acaso do item i;

e: número de Euler (~ 2,72);

D: constante de valor 1,7 (com esta constante os valores da função logística aproximam-se notavelmente dos da curva normal padronizada).

Apesar de o modelo parecer complexo, seus parâmetros têm uma relativa facilidade de interpretação. O parâmetro de dificuldade, chamado de b_i, é o ponto na escala de aptidão no qual a probabilidade de uma resposta correta é de 50% (ou seja, 0,5). A métrica teórica deste parâmetro vai de menos infinito a mais infinito, mas, na prática, transformando a escala da aptidão em escores padrões, com média = 0 e desvio padrão = 1, os valores de b_i

tipicamente se situam entre -4 (itens fáceis) e +4 (itens difíceis). Quanto maior for o b_i, maior deve ser o nível de aptidão exigido para que a pessoa examinada tenha a chance de 50% de acertar o item.

O parâmetro de discriminação, chamado de a_i, é expresso pela inclinação da CCI no momento da inflexão, isto é, quando a curva CCI corta a linha que corresponde à probabilidade de 0,5 de resposta correta, o que acontece sempre quando $\theta = b$. A métrica teórica deste parâmetro também vai de menos infinito a mais infinito, mas valores negativos geralmente não são esperados, pois os mesmos diriam que indivíduos de maior aptidão tendem a errar o item enquanto indivíduos de menor habilidade tendem a acertar. Na prática, a métrica deste parâmetro vai de 0 a 4, onde 0 significa nenhuma discriminação e 4, discriminação praticamente perfeita.

O ângulo de inclinação da curva característica determina a discriminação do item. Quanto mais íngreme a curva no momento da inflexão, mais discriminativo o item. Em outras palavras, ângulos de incidência no ponto de inflexão mais agudos indicam itens mais discriminativos. Assim, se um item apresenta uma discriminação perfeita, então o ângulo de incidência da curva seria de 90 graus, ou seja, uma perpendicular.

O parâmetro c_i afere o acerto casual, ou seja, a probabilidade de um indivíduo com baixa proficiência responder adequadamente ao estímulo. Ele é definido pela distância da origem até o ponto onde a curva corta a ordenada. Na Figura 5, por exemplo, a curva indica que um indivíduo tem 10% de chance de adivinhar a resposta correta.

Na avaliação do *survey* apresentado no Capítulo 4, este trabalho utilizou um modelo de dois parâmetros, justificado na seção resultados do mesmo capítulo. O pacote LTM (*Latent Trait Model*- modelo de variáveis latentes para dados dicotômicos) do Software R (http://www.R-project.org) foi utilizado para calcular os parâmetros dos itens, os quais são utilizados para obtenção dos escores θ dos indivíduos (Rizopoulous, 2006).

Capítulo

3

Este capítulo apresenta uma visão geral da metodologia adotada e detalha sua automação através do NEUROMINER, bem como apresenta os resultados de dois experimentos que auxiliaram na validação da abordagem.

3 UMA METODOLOGIA PARA CLASSIFICAÇÃO AUTOMATIZADA DE SISTEMAS DE REPRESENTAÇÃO PREFERENCIAIS DE DESENVOLVEDORES DE SOFTWARE

3.1 VISÃO GERAL DA METODOLOGIA

A metodologia adotada nesta tese pode ser subdividida nas seguintes etapas:

Fonte de Informações

Considerar todos os emails escritos pelo programador ao longo do projeto que o mesmo está alocado.

Dicionário Neurolinguístico

Cadastrar palavras sensoriais associadas aos sistemas de representação. A literatura sobre PNL traz diversos exemplos de predicados sensoriais (Bandler e Grinder, 1979)(Dilts et al., 1980) (Grinder e Bandler, 1976).

Conceitos da Engenharia de Software

Cadastrar uma taxonomia para Engenharia de Software (ES). O objetivo é relacionar conceitos da ES com predicados sensoriais. A taxonomia pode ser extraída de uma ontologia para ES, sem a preocupação com semântica e hierarquia dos conceitos.

Busca nos Emails por Predicados Sensoriais e suas Relações

Encontrar predicados sensoriais nos emails. Além disso, verificar se os mesmos atuam como adjetivos de conceitos da ES.

Pontuação para cada Sistema

Cada sistema representacional recebe uma pontuação de acordo com a freqüência dos seus predicados em todos os emails. Além disso, considerando que a repetição de uso de um sistema em dias diferentes indica uma preferência, o predicado que tem uma freqüência diária maior, pontua mais. Por fim, se o predicado atua como adjetivo de um conceito da ES, o mesmo recebe um bônus extra na sua pontuação final.

Classificação

Calcular a pontuação final de cada sistema mensalmente. Usar um método estatístico para averiguar se as médias são diferentes. Se as médias forem estatisticamente diferentes, classificar o sistema que obteve a maior média como o SRP.

Nas seções seguintes, abordaremos como esta metodologia foi automatizada.

3.2 ARQUITETURA GERAL PROPOSTA PARA AMBIENTES DE MINERAÇÃO DE REPOSITÓRIOS DE SOFTWARE

A qualidade dos dados é um dos requisitos para a mineração de dados eficiente. Por mais de 15 anos, pesquisadores das áreas de banco de dados e apoio à decisão têm estudado a construção de repositórios de dados para análises mais eficientes. Comumente chamados de *Data Warehouses* (DW), estes repositórios são bancos de dados históricos separados lógica e fisicamente do ambiente de produção da organização, bem como projetados para o apoio à decisão (Colaço Jr, 2004).

Um DW permite a utilização de técnicas de mineração de dados e ferramentas analíticas de forma mais eficaz (Kimball, 1998). Em engenharia de software, pode melhorar a análise dos dados e do processo de desenvolvimento, além de ser conceitualmente alinhado com o CMMI nível 5 e com o conceito de *fábrica de experiência* proposto por Basili e colegas (Basili et al., 2001), no sentido de que efetivamente é base para otimização de processos de software.

Seguindo este conceito, antes da concepção do NEUROMINER, foi proposta e construída uma arquitetura de DW em três camadas para mineração de repositórios de software (Colaço Jr et al., 2009):

(1) a camada ETL (Extração Transformação e Carga), formada por um conjunto de módulos responsáveis pela extração de dados armazenados pelas diversas ferramentas usadas no processo de desenvolvimento;

(2) a camada formada pelos recursos de banco de dados utilizados, para a qual os dados são transferidos pelos módulos ETL, inicialmente para a *Staging Area* e em seguida para o DW, depois de transformados; e

(3) a camada de análise.

Na camada de análise, estão ferramentas de suporte à decisão para apresentações, induções ou inferências feitas sobre o conteúdo do DW. A Figura 6 apresenta a infra-estrutura proposta baseada no estilo arquitetural em camadas definido pelo SEI (Clements et al., 2002).

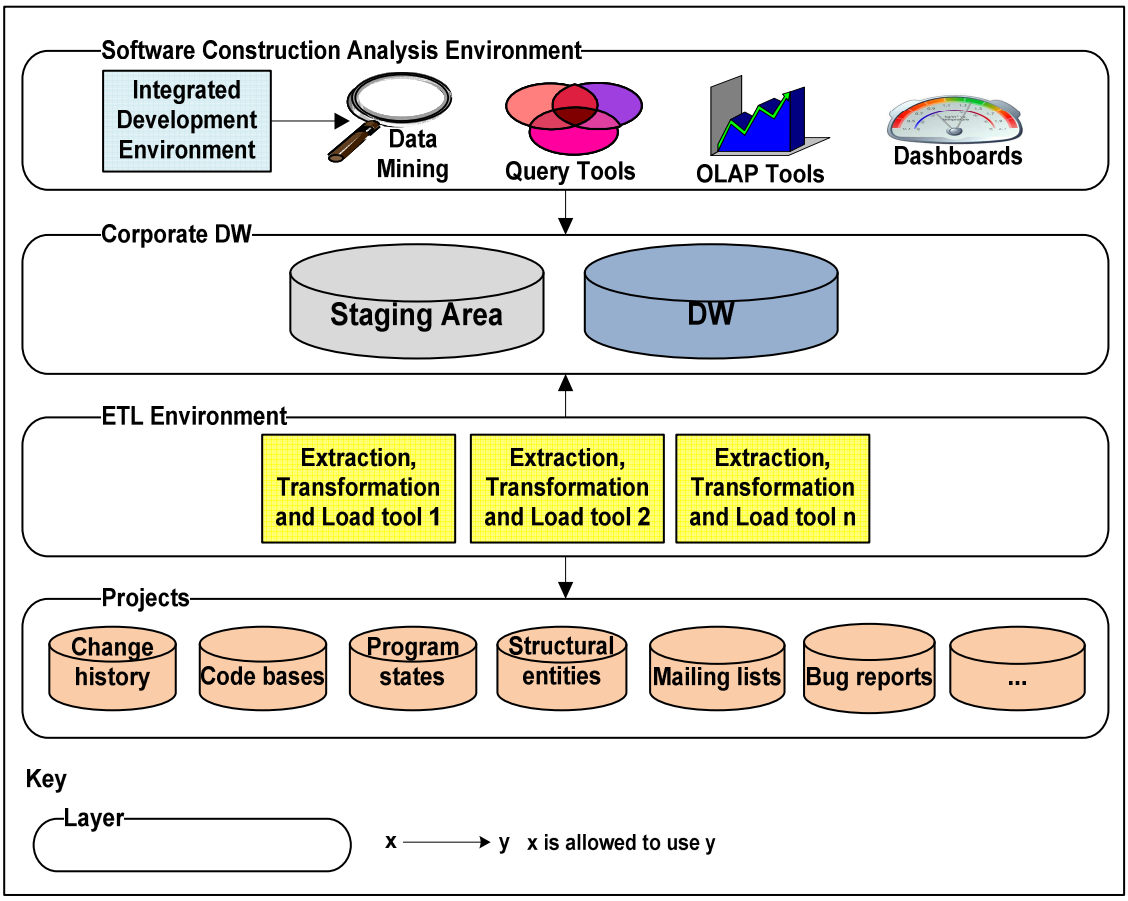

Figura 6. Arquitetura do DW (Colaço Jr et al., 2009)

Em um *Data Warehouse*, suas subdivisões lógicas são chamadas de *data marts* (Colaço Jr, 2004)(Kimball, 1998). Para validação da arquitetura, foi projetado e implementado um *data mart* sobre mudanças de software. Esse *data mart* serviu de base para realização de um estudo de caso e um experimento controlado, ambos na indústria. O parceiro escolhido foi a CIAL, uma empresa que produz e distribui produtos Coca-Cola para os estados de Sergipe e Alagoas. Essa cooperação foi possível devido ao relacionamento de consultoria que o autor deste trabalho mantém com a referida empresa. Todos os experimentos relatados nesta tese que tiveram execução em ambiente industrial foram executados na CIAL.

Os estudos descritos a seguir não se focaram em análise neurolinguística, mas objetivaram utilizar todas as camadas da arquitetura proposta, criando infra-estrutura para

implantação de um *Dashboard* de acompanhamento, bem como de um módulo para mineração de associações entre módulos de software.

3.2.1 Dashboard de Manutenções Corretivas

Desde 2001, a CIAL utiliza a plataforma Microsoft (MS) para implementar novos sistemas e reestruturar seus sistemas legados. Por esta razão, foi construído um ETL (camada 1) para extrair informações de mudanças de software armazenadas no sistema de controle de versões MS VSS (*Microsoft Version Control System*), carregando-as no DW corporativo (camada 2), armazenado em uma base MS SQL SERVER (Colaço Jr et al., 2009)(Colaço Jr et al., 2009B).

Para camada de análise, foi criado um *Dashboard* de manutenções corretivas, ilustrado na Figura 7. O exemplo nessa Figura usa os valores das métricas de acordo com indicadores de fatores de qualidade, mostrando os valores em cores diferentes (verde, amarelo e vermelho), como também numericamente (previsto versus valores reais). Em se tratando de manutenções corretivas, quanto menor o número de manutenções deste tipo (área verde), melhor.

Cada pequeno *dashboard* no painel monitora um sistema. As setas brancas indicam o estado atual de uma variável (número de manutenções corretivas), enquanto as setas cinza indicam seu valor anterior. No exemplo, dados de 2007 em comparação com dados de 2006.

O sistema foi disponibilizado em 2008 para ajudar os gerentes de projeto da CIAL em suas decisões. A principal vantagem do sistema é a sua capacidade de gerenciamento por exceção, priorizando ações de melhoria sobre os sistemas que não se encontram dentro das metas (fora da área verde).

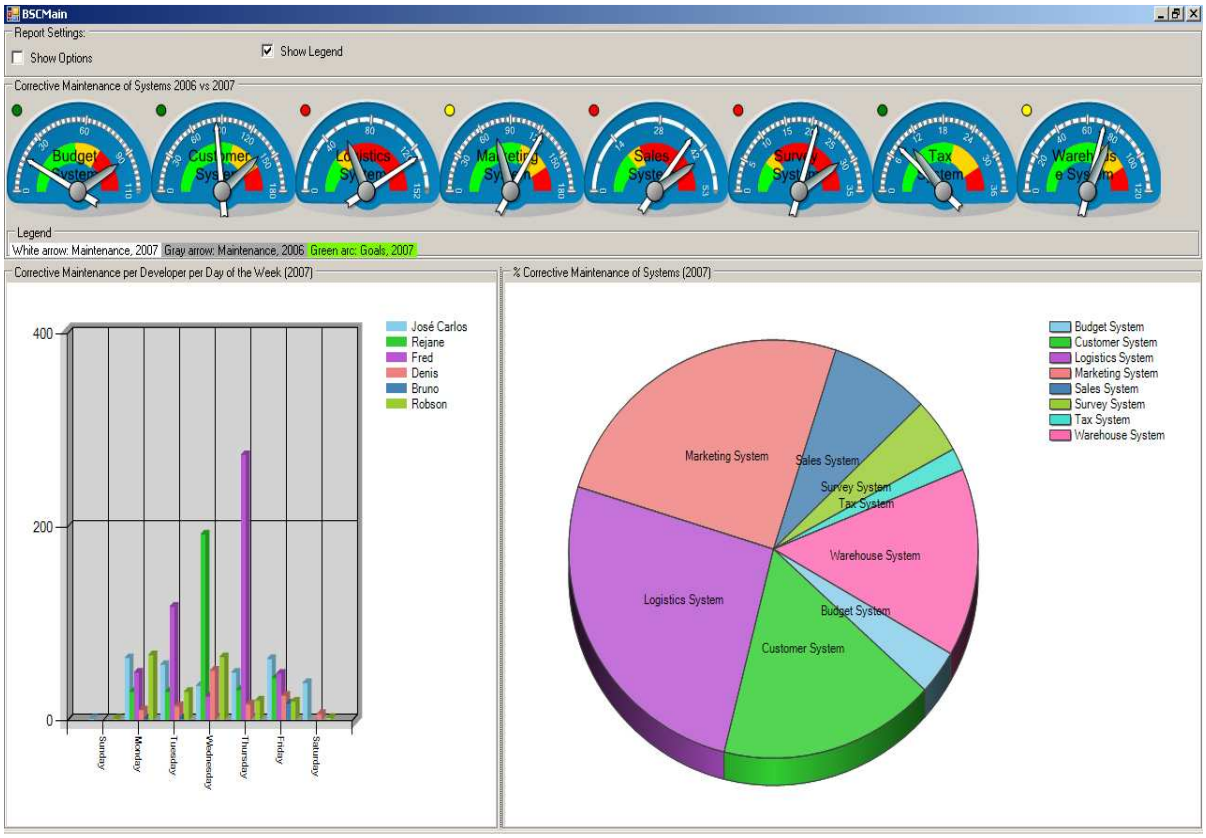

Figura 7. *Dashboard* de manutenções corretivas (Colaço Jr et al., 2009)

Além do *Dashboard*, foi implementado um módulo para mineração de associações de manutenção e realizado um experimento controlado para avaliá-lo. Este experimento é descrito a seguir.

3.2.2 Experimento com mineração de modificações de software

Utilizando a arquitetura descrita anteriormente, também foi testada a inferência de um modelo de predição e posterior acoplamento do mesmo a um ambiente de desenvolvimento de software (ADS), o ADS Microsoft Visual Studio. Baseado em um algoritmo de regras de associação, foi criado um *plug-in* para o ADS que minera o DW e sugere os próximos módulos a serem alterados após uma tarefa de manutenção.

Um novo ETL foi implementado para esta finalidade. Ele utiliza o algoritmo de extração proposto por Zimmermann e Weissgerber (Zimmermann e Weissgerber, 2004). São considerados associados os módulos modificados em uma mesma transação. A transação é definida por uma janela de tempo (Fogel e O'Neill, 2009), o nome do autor, o comentário e o caminho da alteração. A janela de tempo adotada foi de cinco minutos. Módulos alterados dentro dessa janela, pelo mesmo programador, são considerados modificados conjuntamente.

Uma vez carregados os módulos modificados e suas respectivas transações de associação no DW, aplicou-se outro algoritmo para descoberta de regras de associação. Para isto foi utilizado o pacote de *Business Intelligence* da Microsoft, o qual disponibiliza um algoritmo pertencente à família APRIORI (Agrawal e Srikant, 1994), eficiente para achar relações entre os itens de uma transação.

Como saída, este algoritmo produz regras com a associação entre os módulos de software, informando o nível de suporte e confiança para as mesmas. A regra é apresentada na forma de uma entidade (módulos ou classes) antecedente que ao ser modificada determina a alteração de uma entidade conseqüente. Neste contexto, o suporte é o número de transações em que o antecedente aparece e a confiança da regra é o percentual de transações do suporte nas quais as entidades antecedente e conseqüente aparecem.

Estes mecanismos de mineração do DW foram adicionados como plug-in do ADS. Um botão no ADS permite a descoberta *on-the-fly* de associações existentes entre a entidade sendo editada e as outras entidades do sistema. Ou seja, a busca no modelo de mineração de dados gerado é feita em tempo real.

Um exemplo de alerta de associação emitido pelo *plug-in* ao programador pode ser visto na Figura 8, traduzindo-se assim: *quem alterou ngVendasCC.cs, também alterou dbAcaoMkt.cs e fcAcaoMkt.cs, com confianças de 93,85 e 93,39, respectivamente.*

No resultado de consulta ao modelo (vide Figura 8), são apresentadas ao programador as dez primeiras entidades que alcançam uma confiança mínima de setenta e cinco pontos percentuais de associação com a entidade sendo alterada, listadas em ordem decrescente de suporte e confiança.

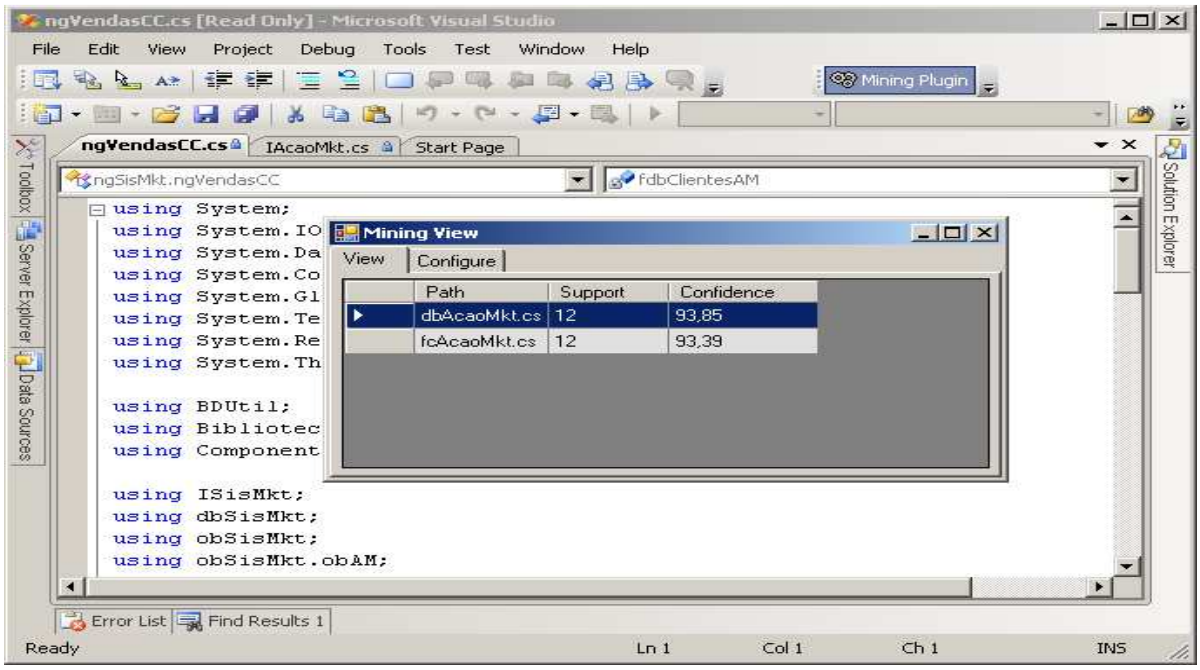

Figura 8. *Plug-in* para mineração de modificações de software (Santos, Colaço Jr. e Mendonça, 2009)

Após a instrumentação do ADS com o modelo de mineração de dados gerado, foi executado o experimento propriamente dito. Foram avaliadas as bases VSS de 18 sistemas da CIAL, obtendo-se resultados médios de cobertura e precisão (Rijsbergen, 1979) de 76% e 58%, respectivamente. Em outras palavras, a ferramenta sugere associações para 76% dos casos, acertando 58% das sugestões das próximas alterações. Para alguns sistemas, chegou a alcançar uma precisão de mais de 90%.

Os resultados detalhados do experimento foram publicados em (Colaço Jr et al., 2009B) e (Santos, Colaço Jr. e Mendonça, 2009), bem como a arquitetura do DW em (Colaço Jr et al., 2009). Como conclusão principal, o estudo de caso e o experimento mostraram a viabilidade de manutenção de uma base de dados histórica do processo de manutenção de código, bem como da exploração do acoplamento de técnicas de mineração de dados aos ambientes de desenvolvimento de software.

Uma vez definida e avaliada, a arquitetura foi utilizada para a construção e o suporte ao NEUROMINER. O próximo passo foi então a criação do NEUROMINER, junto com um ETL e um *data mart* específicos para esta ferramenta. O ETL objetivando a extração,

transformação e carga dos e-mails. O *data mart* objetivando o armazenamento histórico desses dados. Esses módulos são detalhados a seguir.

3.3 CONSTRUÇÃO DO NEUROMINER

Como descrito na seção 2.4, os criadores da PNL defendem que as pessoas dizem palavras e frases sensoriais, também chamadas de pistas verbais ou predicados sensoriais, as quais indicam um processamento visual, auditivo ou cinestésico (Dilts et al., 1980). Assim, em contextos específicos, algumas pessoas podem dar preferência para um dos sistemas de representação.

Em sinergia com a crítica feita em (Einspruch e Forman, 1985), as classificações do NEUROMINER não serão estanques, ou seja, o objetivo é identificar o sistema de representação mais usado por um programador e o percentual de uso dos demais. Para isso, foi adotada uma abordagem LIWC semelhante à apresentada em (Pennebaker et al., 2001). Ferramentas LIWC simplesmente contêm palavras pertencentes a um determinado dicionário, o qual é agrupado em dimensões psicométricas, com cada palavra inserida em uma ou mais dimensões (vide Tabela 2).

A partir do dicionário, palavras podem ser contadas, por exemplo, para determinar se um indivíduo é mentiroso, sociável, inibido ou otimista. Usualmente, as ferramentas LIWC apenas contam palavras, eximindo-se de análises estatísticas e preditivas.

Tabela 2 – Exemplo de dicionário LIWC

Dimensão	Exemplos de palavras	Explicação
Pronomes	‘our’, ‘I’	Sujeito ou objeto em uma sentença
Negação de outras palavras	‘no’, ‘not’	Palavras que negam outras palavras
Insight	‘think’, ‘know’	Palavras que indicam entendimento sobre um assunto
Inibição	‘block’, ‘contrain’	Palavras que mostram que o indivíduo é retraído
Tentativa	‘maybe’, ‘guess’	Palavras que indicam que foi falado algo incerto
Causalidade	‘because’, ‘effect’	Palavras que indicam por que alguma coisa aconteceu

O dicionário criado para o NEUROMINER possui três dimensões básicas (Visual, Cinestésica e Auditiva), bem como os predicados sensoriais (Bandler e Grinder, 1979)(Dilts et al., 1980) correspondentes a cada uma delas, nas línguas inglesa e portuguesa (vide Tabela 3).

Tabela 3 – Dicionário para Classificação Neuro-Linguística

Dimensão	Exemplos de palavras	Língua
Visual	‘claro’, ‘ilustrar’	Português
Cinestésico	‘concreto’, ‘áspero’	Português
Auditivo	‘dissonante’, ‘ressoar’	Português
Auditivo	‘dissonant’, ‘resonate’	Inglês

Além disso, foi criada uma dimensão Conceitos para aumentar o poder contextual da classificação. Esses conceitos, ou classes, foram extraídos de uma ontologia para Engenharia de Software utilizada em (Witte et al., 2008) e descrita em (Calero et al., 2006), a qual se baseia em vários domínios de programação, incluindo linguagens de programação, algoritmos, estruturas de dados e decisões-chave tais como padrões de projeto e arquiteturas de software. Apesar de terem sidos extraídos de uma ontologia, no nosso dicionário, não exploramos as características semânticas inerentes a este tipo de estrutura, ou seja, utilizamos os conceitos apenas como uma taxonomia para Engenharia de Software.

O objetivo é analisar o relacionamento direto entre predicados sensoriais e o contexto da Engenharia de Software. Para isto, a estratégia foi pré-cadastrar no dicionário as possibilidades de **locuções substantivas** ou *noun phrases* (vide seção 2.1.2.2) formadas pelo agrupamento contextual de palavras que representam conceitos da Engenharia de Software com palavras sensoriais. Esta é uma inovação do NEUROMINER.

Desta forma, os conceitos da Engenharia de Software foram marcados como substantivos, elementos centrais (*Head*), e os predicados sensoriais aptos para esse fim, como palavras adjacentes ou modificadores (*Mod*).

Na Tabela 4, a coluna *Tag* representa a parte de uma *noun phrase* (NP): *Mod* e *Head* referem-se a modificador e elemento central, respectivamente. Por exemplo, o fragmento de texto em inglês *"concrete algorithm"* será considerado um único termo (NP) pelo NEUROMINER. Além disso, esse termo será computado como processamento cinestésico, uma vez que seu modificador pertence à dimensão cinestésica. Em outras palavras, ao encontrar um conceito de Engenharia de Software no texto, o NEUROMINER buscará predicados sensoriais adjacentes.

Todos os modificadores próximos a um conceito são computados, independente do sistema utilizado. Por exemplo, na frase: *"i saw it and ran the algorithm"*, serão pontuados os sistemas visual e cinestésico, devido à presença dos modificadores *"see" e "run"*, pertencentes aos sistemas visual e cinestésico, respectivamente.

Por fim, na abordagem de classificação da ferramenta, as NPs sensoriais, formadas com conceitos da Engenharia de Software, têm um bônus multiplicado ao seu peso (vide próxima seção). Predicados sensoriais isolados serão computados, porém sem nenhum bônus.

Tabela 4 – Dicionário com conceitos da Engenharia de Software

Dimension	Example Word	Tag
Visual	'brilliant'	Mod
Auditive	'dissonant'	Mod
Kinaesthetic	'concrete'	Mod
Concepts	'algorithm'	Head

3.3.1 Modelo multidimensional

Para o projeto de banco de dados, adotamos a Modelagem Dimensional apresentada em [50,66]. A Modelagem Dimensional é uma técnica de modelagem lógica que apresenta os dados em um padrão intuitivo capaz de balancear desempenho e volume de dados. Em um esquema do modelo dimensional, também chamado de esquema estrela [50,66], podemos observar uma tabela dominante, chamada de Tabela de Fatos, cujos atributos representam, geralmente, medidas numéricas associadas ao contexto da análise. Relacionando-se com a Tabela de Fatos, encontramos as Tabelas de Dimensões, que caracterizam as medidas numéricas citadas anteriormente.

Conforme descrito na seção anterior, um dicionário LIWC é formado por dimensões psicométricas. Para não haver uma sobreposição de conceitos, chamaremos as tabelas de dimensões de facetas.

O modelo para o dicionário apenas beneficiou-se da base DW já adotada (Colaço Jr et al., 2009), ou seja, a possibilidade de extensão com o acréscimo de novos fatos e/ou novas facetas. Criou-se então um novo *data mart* para o NEUROMINER. A Figura 9, utilizando a notação de James Martin (Martin e Carma, 1985), apresenta parte do modelo utilizado.

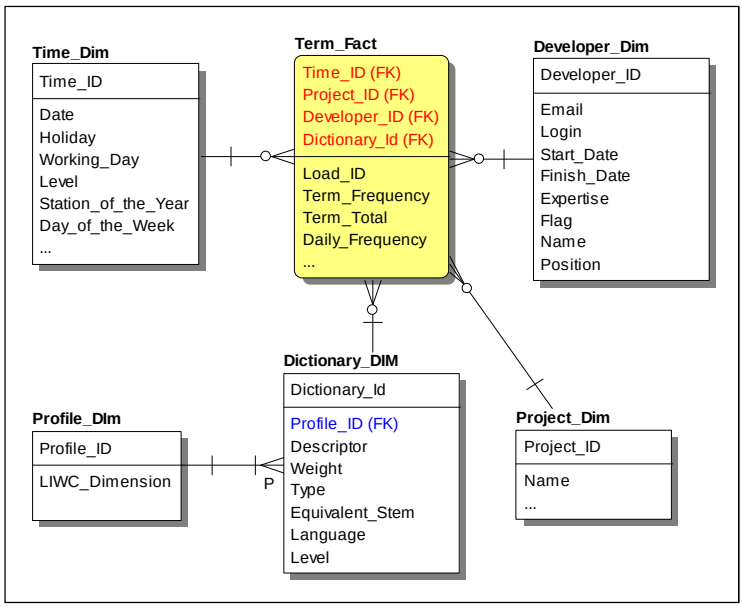

Figura 9. Modelo multidimensional

As dimensões LIWC que descrevem os perfis neuro-linguísticos e os conceitos de Engenharia de Software foram modeladas como uma única faceta (*Profile_Dim*) no esquema estrela. Essa faceta está associada a uma faceta dicionário (*Dictionary_Dim*) que armazena todos os termos correspondentes, seus radicais e sinônimos. No modelo, um termo pode ser: um predicado sensorial, um conceito da Engenharia de Software ou uma relação entre um predicado e um conceito.

A Tabela de fatos (*Term_Fact*), em um nível de granularidade mensal, armazena os *escores* calculados para os termos encontrados, bem como dados atômicos tal como a freqüência do termo, utilizados para calculo destes *escores*. As demais tabelas foram reaproveitadas ou conformadas [50,66] do *data mart* apresentado em (Colaço Jr et al., 2009).

3.3.2 Mineração de e-mails

Todas as etapas de pré-processamento apresentadas na seção 2.1.2.1 são executadas pelo módulo ETL desenvolvido. Isso ocorre tanto na extração de novos e-mails quanta na inclusão de novas palavras no dicionário. O NEUROMINER, por exemplo, remove automaticamente *stop words* e aplica radicalização a predicados sensoriais incluídos no dicionário e a todos os e-mails carregados, fato que reduz enormemente a dimensionalidade. Para detecção das locuções substantivas (*noun phrases*), foi usada a mesma abordagem do pacote FOSS MuNPEx (*Multi-Lingual Noun Phrase Extractor*, http://www.ipd.uka.de/~durm/tm/munpex/).

A Figura 10 resume os principais passos do processo ETL.

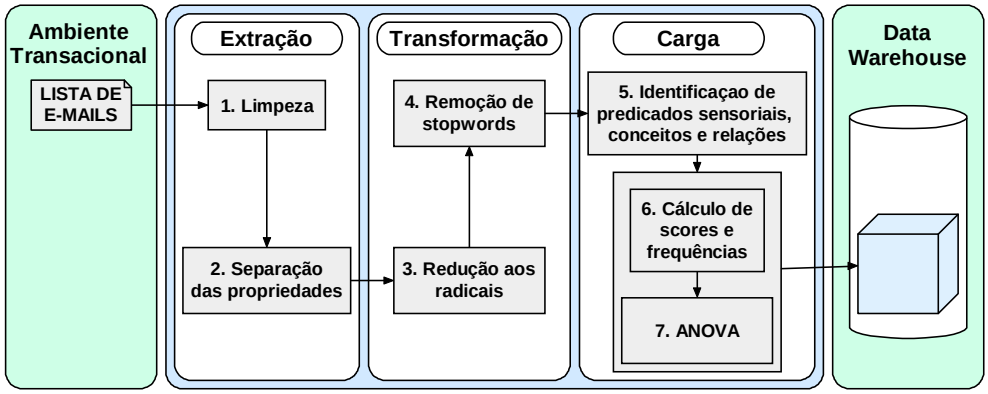

Figura 10. NEUROMINER ETL

Depois que é efetuado o download dos e-mails, são analisados os metadados dos mesmos, utilizando a mesma abordagem discutida em (Rigby e Hassan, 2007). São removidos anexos, respostas (por exemplo, linhas que começam'>'), assinaturas de e-mail e código HTML (passo 1 da Figura 10). No passo 2, são considerados apenas os metadados ou propriedades interessantes à nossa abordagem: remetente, texto escrito pelo remetente, assunto, data e hora do e-mail. Em outras palavras, apenas o texto do remetente é considerado, sendo removido o texto de outrem, geralmente anexado a uma resposta.

Na etapa de transformação (passos 3 e 4), os conteúdos das mensagens são refinados através da remoção de *stop words* e da redução das palavras aos seus radicais

(Porter, 1980). Na última etapa, os termos são detectados (passo 5), para isto, predicados sensoriais associados ou não a conceitos da engenharia de software são contados mensalmente, considerando as dimensões Neuro-linguísticas do dicionário. Vale ressaltar que o DW armazena os sinônimos diretos desses termos. Por exemplo, detecta-se o predicado sensorial "*see*", bem como sinônimos diretos tais como "*view*", "*look*" e "*visualize*".

Em seguida, no passo 6, as freqüências e *escores* dos termos encontrados, tanto mensais quanto acumulados, são calculados e armazenados na Tabela de fato *Term_Fact* (vide Figura 9). Nesse passo, o NEUROMINER inova com uma abordagem de mineração de textos para classificação Neuro-Lingüística de indivíduos, ao invés da tradicional classificação de documentos. É adotada uma abordagem cumulativa baseada em freqüência diária.

Desta forma, o conjunto de todos os e-mails escritos por um programador é considerado, analogicamente, um "grande texto" a ser classificado. Uma simples abordagem para isso é contar todas as palavras encontradas em todos os e-mails de um programador e verificar o percentual de cada sistema representacional. Contudo, vislumbrando análises de evolução posteriores e a base para evidências estatísticas, foi considerada a freqüência diária df(j) de cada termo j, a qual é atualizada e armazenada a cada mês. A freqüência diária é calculada pela razão entre o número de dias em que o predicado sensorial j apareceu e o número de dias carregados.

Conseqüentemente, outra inovação foi utilizar uma versão alternativa para fórmula tf-idf apresentada na seção 2.1.2.1. Na Equação (1), nota-se que o peso (*score)* tf-df atribuído ao predicado sensorial j é a freqüência do predicado (isto é, o número de ocorrências do predicado dividido pelo número de ocorrências de todas as palavras) modificado pelo fator de importância da palavra. Esse fator, para a abordagem NEUROMINER, é a freqüência diária df(j) abordada anteriormente. Assim, quanto mais um predicado sensorial aparece, mais importante ele é para classificar o indivíduo.

$$tf - df(j) = \left(tf(j) + df(j)\right) \times b \quad (1)$$

Além disso, um bônus b também é multiplicado à medida. O bônus pode ser 1 ou 2, onde b será igual a 2 se o predicado sensorial é uma NP contextual (vide seção 3. 2) ou frase, e igual a 1, se o termo é um predicado simples.

No final de cada mês, como as freqüências diárias mudam, os pesos dos predicados são recalculados. Além disso, é calculado e armazenado um total geral dos pesos (*final weight*) para cada sistema representacional. Em seguida, a média mensal de cada sistema representacional é computada.

Por fim, um agregado com dados estatísticos descritivos dos *escores* é carregado, para servir de base para um teste de hipótese ANOVA (passo 7). Um sistema representacional só é classificado como o mais usado por um desenvolvedor, se, e somente se, considerando-se um nível de significância de cinco por cento, houver evidências estatísticas que as médias mensais de *escores* são diferentes.

Independente da evidência estatística, o NEUROMINER apresenta os percentuais finais para cada sistema de representação, a distribuição de programadores no sistema, bem como as classificações feitas pela abordagem ANOVA. A Figura 11, mostrada mais a frente, mostra um exemplo de percentuais finais para o um desenvolvedor específico.

3.4 PRIMEIRO EXPERIMENTO

O restante deste capítulo descreve uma validação experimental da nossa abordagem. O processo experimental apresentado, neste e nos demais capítulos, segue as diretrizes de (Wohlin et al., 2000). A próxima seção concentrar-se-á na definição e planejamento do experimento. Nas seções seguintes, serão apresentados a execução e os resultados experimentais obtidos.

3.4.1 Definição de objetivo

O principal objetivo do nosso estudo é avaliar se *top committers* de projetos FOSS terão um SRP. Esse objetivo é formalizado utilizando o modelo GQM proposto por Basili (Basili e Weiss, 1984) e apresentado em (Solingen e Berghout, 1999):

> **Analisar** *top committers* de projetos FOSS
> **com a finalidade de** avaliação
> **em relação aos** Sistemas de Representação Preferenciais
> **do ponto de vista de** pesquisadores de engenharia de software
> **no contexto de** listas de discussão de projetos de FOSS.

3.4.2 Planejamento

3.4.2.1 Seleção de contexto

O experimento terá como alvo programadores de projetos FOSS.

3.4.2.1.1 Formulação da hipótese

Os assuntos que estamos tentando explorar são os seguintes:

(1) Estamos interessados em verificar se *top committers* de projetos FOSS terão um SRP.

(2) Além disso, acreditamos que *top committers* usam mais o sistema cinestésico do que o auditivo e o visual. Nossa crença é a de que os programadores do núcleo central de comunidades FOSS confiam fortemente em suas experiências, sendo menos dependentes de artefatos visuais e auditivos do que a população geral de engenheiros de software destas comunidades.

Considerando o árduo trabalho manual necessário para encontrar e validar todos os e-mails utilizados por cada *top committer*, a amostra será pequena. Vale ressaltar que alguns desenvolvedores possuem outra identidade na comunidade, utilizando e-mails que não possuem nenhuma associação com o nome da pessoa verdadeira. Conseqüentemente, devido ao baixo número de *top committers* totalmente identificados e validados, um teste estatístico formal não será executado para segunda questão. No entanto, considerando o grande número de e-mails que será extraído, a prova da existência de um SRP para cada selecionado terá grande poder. Também será feita uma análise qualitativa detalhada do perfil de cada desenvolvedor, a fim de verificar a acuracidade e validade dos *escores* encontrados para cada Sistema Representacional.

Formalmente, a hipótese que queremos confirmar é:

Hipótese nula H_0: *Top committers* de projetos FOSS têm a mesma média mensal para os 3 *escores* dos sistemas representacionais.

H_0^{SRP}: μ(*Visual Final Weight*) = μ(*Auditive Final Weight*) = μ(*Kinesthetic Final Weight*).

Hipótese alternativa H_1: no mínimo uma das médias é diferente das outras.

3.4.2.1.2 Seleção de participantes e objetos

Em primeiro lugar, selecionamos extrair e-mails das listas de discussão dos projetos FOSS Apache (Apache, 2010) e Postgresql (Postgresql, 2010). Para o Apache, foi possível selecionar os quatro desenvolvedores que efetuaram a maioria dos *commits* para o sistema. Não por acaso, esses são os mesmos desenvolvedores estudados em (Rigby e Hassan, 2007). Para o Postgresql, também elegemos os quatro desenvolvedores que efetuaram a maioria dos *commits*.

Em ambos os projetos, dois *top committers* ainda contribuem para o projeto e dois já deixaram o mesmo. Também optamos por considerar a população geral nos dois projetos. Sendo assim, foi planejado o cálculo de *escores* para um *cluster* com mensagens de todos os outros desenvolvedores. Durante este capítulo, a população em geral, para os dois projetos, será referida como *cluster*.

Por questões legais, não utilizaremos os nomes dos participantes neste trabalho. Serão utilizadas letras para identificar cada desenvolvedor. A execução será completamente não-intrusiva, pois os dados serão retirados diretamente das listas de discussão do projeto, sem nem mesmo o desenvolvedor saber que os e-mails postados seriam analisados um dia.

3.4.2.1.3 Instrumentação

O NEUROMINER será usado para calcular o peso final (*Final Weight*) de cada Sistema de Representação, bem como suas médias mensais e ANOVA, para o teste de hipótese (vide seção 3.3). Além disso, foi desenvolvido um módulo OLAP para auxiliar na validação dos dados.

3.4.2.1.3.1 Módulo OLAP

O módulo OLAP foi desenvolvido para prover geração de gráficos e navegação analítica nos dados que descrevem os perfis dos desenvolvedores. A seguir, a apresentação de algumas funcionalidades OLAP será feita com resultados do projeto Apache.

Analisando os *escores* acumulados na Figura 11 (gráfico à esquerda), observa-se que o perfil predominante para o *top committer* B é o visual. No gráfico à direita, é possível ver a evolução do perfil deste desenvolvedor no período de janeiro a dezembro de 1999.

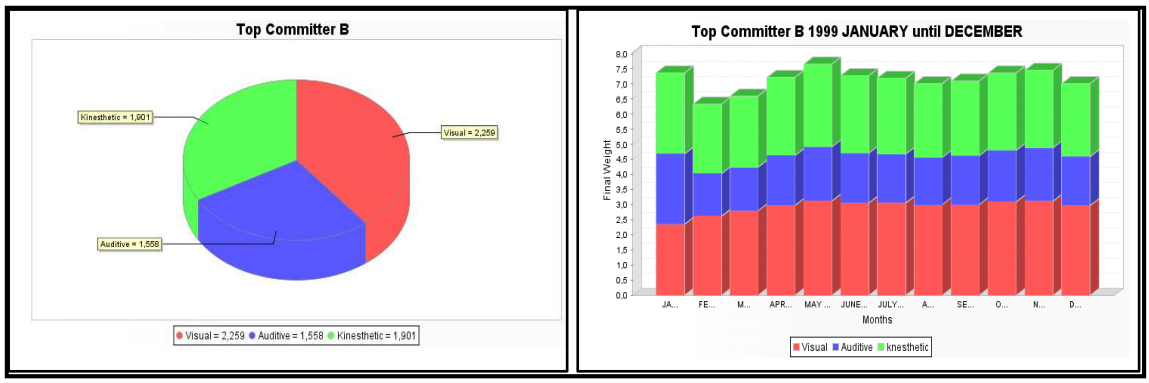

Figura 11. Perfil atual e evolução do *top committer* B.

Outro aspecto importante é a análise dos termos mencionados que mais pontuaram. Para isto, pode ser realizado um *drill-down*, em conjunto com um *ranking*, os quais dão acesso aos 10 termos ou às 10 frases que alcançaram os maiores *escores*. O resultado está ilustrado na Figura 12, na qual é interessante notar a presença de conceitos da Engenharia de Software, tais como *Server*, *Compiler e Module*, combinados com palavras sensoriais. Indo mais além, também é interessante notar o termo *pain* como um dos predominantes. Como o desenvolvedor B já deixou o projeto, isto pode ser um indício de insatisfação nas suas últimas mensagens, uma vez que *pain* indica dor ou desconforto.

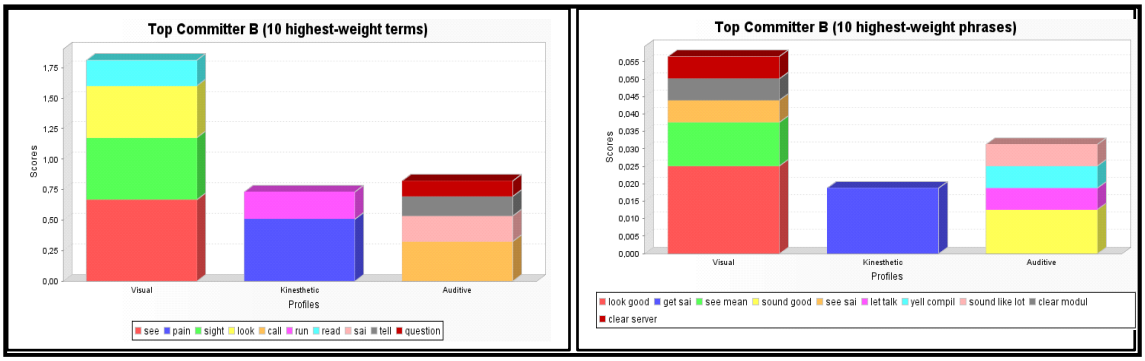

Figura 12. Termos e frases mais pontuadas pelo *top committer* B.

3.4.3 Operação

3.4.3.1 Execução

Para o PostgreSql, o NEUROMINER foi executado para analisar o corpo de todas as mensagens de e-mail da lista de discussão, entre 1997 e 2006 (57.159 mensagens). Para o Apache, também foram analisadas todas as mensagens de e-mail, porém entre 1996 e 2005 (35.483 mensagens).

Para cada colaborador e para os *clusters*, uma vez por mês, calculou-se o peso final (*Final Weight*) de cada Sistema de Representação (*escores*), utilizando a metodologia descrita na seção 3.3. Assim, geraram-se *data points* dos e-mails minerados por mês, para cada participante.

Os *clusters* foram minerados por um período de 3 anos (36 meses), perfazendo uma amostra dos 10 anos selecionados anteriormente. Os *top committers* foram minerados para os 10 anos selecionados, contudo, os *data points* foram gerados apenas para os meses em que eles postaram pelo menos um e-mail para a lista de discussão. Por fim, o NEUROMINER

também calculou a análise de variância (ANOVA) dos *data points* mensais, para cada participante.

3.4.3.2 Validação dos dados

Além das análise de metadados e seleção descritas na seção 3.3, este experimento executou as seguintes validações:

(1) com o módulo OLAP, para cada participante, os termos e frases mais ditas foram checados em e-mails de origem (até 10 e-mails), averiguando se realmente contextualizavam algo dito pelo indivíduo. A preocupação foi evitar que o termo ou frase fizesse parte de um texto não dito pelo remetente, tais como citações e frases de outras pessoas, comuns em assinaturas fixas;

(2) todos os cálculos ANOVA foram revalidados usando SPSS.

3.4.4 Resultados

As Tabelas 5 e 6 resumem os resultados. A coluna *Total* apresenta o número de meses (*data points* para cada participante), de dias e de e-mails explorados. Para cada sistema representacional, o peso final é mostrado com base em todos os predicados sensoriais encontrados, bem como as médias mensais desses pesos. A coluna ANOVA retrata os *p-values* calculados para o teste da hipótese nula.

3.4.4.1 Análise e interpretação

Para os testes estatísticos, foi estabelecido um nível de significância (α) de 0.05. As Tabelas 5 e 6 mostram que a primeira hipótese é rejeitada para os *clusters* e 7 desenvolvedores, os quais obtiveram um *p-value* 0.000. O único desenvolvedor não classificado foi o G, com *p-value* de 0.085, maior que 0.05.

Em resumo, os resultados para os *clusters* e os desenvolvedores A, B, C, D, E e H são significativamente menores do que 0.05, permitindo a forte rejeição da hipótese nula e averiguação de que os mesmos possuem um SRP (destacado em amarelo).

Tabela 5 – Resultados para os *top committers* do Apache (Colaço Jr. et al., 2010)

Participant	Left the Project?	Totals (1996 - 2005)			Visual		Auditive		Kinesthetic		ANOVA p-value
		Months	Days	Emails	Final Weight	Monthly Mean	Final Weight	Monthly Mean	Final Weight	Monthly Mean	
A	Yes	53	773	4357	2.458847752	2.7274	2.222439202	2.4645	2.579237886	2.9774	.000
B	Yes	33	320	1082	2.258647848	2.6680	1.557743226	1.6667	1.900853069	2.2514	.000
C	No	72	1213	4279	1.904784203	2.3577	1.684352265	2.0152	2.17853237	2.6210	.000
D	No	42	366	644	0.557085631	0.6013	0.526795847	0.6581	0.441884768	0.4684	.000
Cluster	-	36	1091	25121	6.906216384	7.7567	6.456228874	7.3756	8.849529521	9.5515	.000

Tabela 6 – Resultados para os *top committers* do PostgreSql

Participant	Left the Project ?	Totals (1997 - 2006)			Visual		Auditive		Kinesthetic		ANOVA p-value
		Months	Days	Emails	Final Weight	Monthly Mean	Final Weight	Monthly Mean	Final Weight	Monthly Mean	
E	Yes	62	899	2854	0.928310296	0.929891595	0.546308965	0.530733142	0.637687699	0.638343065	.000
F	Yes	53	478	1284	0.97883767	0.958569136	0.419031696	0.421918608	0.615530516	0.631261987	.000
G	No	55	536	1176	0.845112965	0.718432684	0.672360338	0.629946198	0.561439549	0.725690981	0.085
H	No	121	2728	17712	0.901184217	0.855757104	0.648274323	0.644466038	0.596461266	0.600263054	.000
Cluster	-	36	731	34133	0.745095331	0.717685413	0.623314426	0.617197033	0.727219871	0.752834067	.000

Em resposta à segunda hipótese, os resultados são completamente opostos ao que esperávamos. Observou-se que os desenvolvedores B, D, E, F e H não têm *score* maior para o sistema cinestésico. Isto contradiz a nossa hipótese inicial de que *top committers* são mais cinestésicos que visuais e auditivos. Além disso, o sistema cinestésico é o SRP da população em geral (vide linha *cluster* nas Tabelas 5 e 6).

Com relação aos *top committers*, descobrimos que há quatro visuais, dois cinestésicos e um auditivo. Averiguando seus perfis, percebemos que a maioria deles é muito focada na documentação dos sistemas, contrariando o nosso estereótipo inicial de um desenvolvedor apenas de baixo nível. Do ponto de vista da população em geral, o fato da mesma ser em média cinestésica, leva-nos a crer que a maioria das pessoas que postam na lista é realmente envolvida com as atividades práticas do projeto, contrapondo nossa convicção inicial de que muitas mensagens são postadas por iniciantes ou simplesmente curiosos – aqueles que querem apenas ouvir sobre o projeto.

Mesmo quando há predominância do sistema cinestésico, os resultados mostram que os desenvolvedores de software livre também têm significativo uso dos sistemas visual e auditivo. Isso pode indicar uma oportunidade de introdução de novas ferramentas de visualização, bem como de ferramentas de trabalho cooperativo que permitam uso de sons e da voz, aumentando, respectivamente, os estímulos visuais e a interação direta entre os desenvolvedores FOSS.

Analisando mais profundamente os perfis dos *top committers* disponíveis em (Apache Contributors, 2010) e (Postgresql Contributors, 2010), descobrimos que o desenvolvedor B teve um forte envolvimento com a arquitetura do projeto e com o trabalho de hibridizar o Apache. Isso parece dar suporte ao seu SRP Visual (vide Tabela 5 e Figura 13).

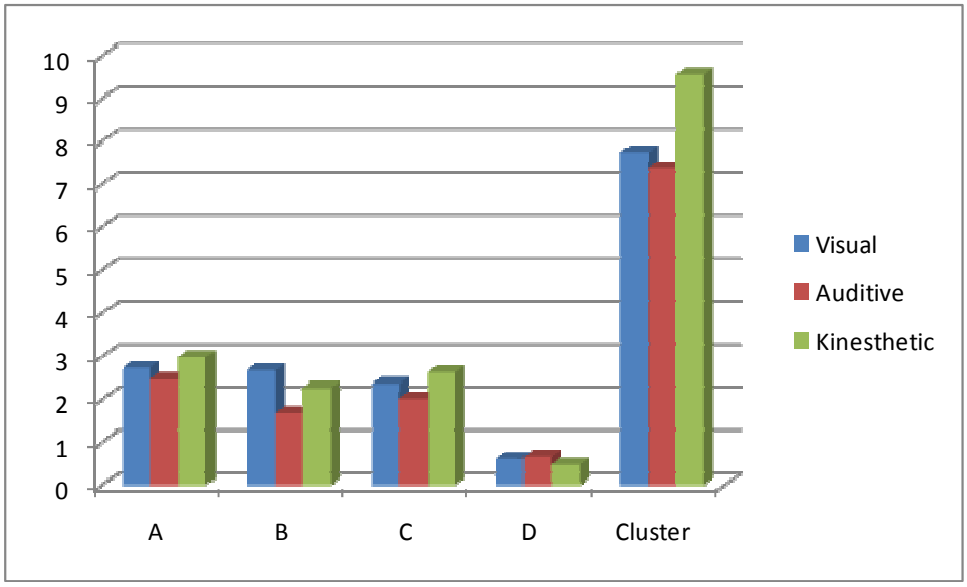

Figura 13. *Escores* para o Apache

O desenvolvedor D - o participante mais singular entre os *top committers* - tem o único SRP auditivo encontrado, mas com um forte *score* visual. Seu perfil indica que ele contribui muito com a documentação do projeto e, diferentemente dos outros programadores, que trabalham excessivamente com linguagem C, tem como linguagem predominante de programação a XML. Isso possivelmente se enquadra no perfil minerado, pois facilidade em ouvir, bem como uso de artefatos visuais são características pertinentes a pessoas envolvidas na documentação de softwares.

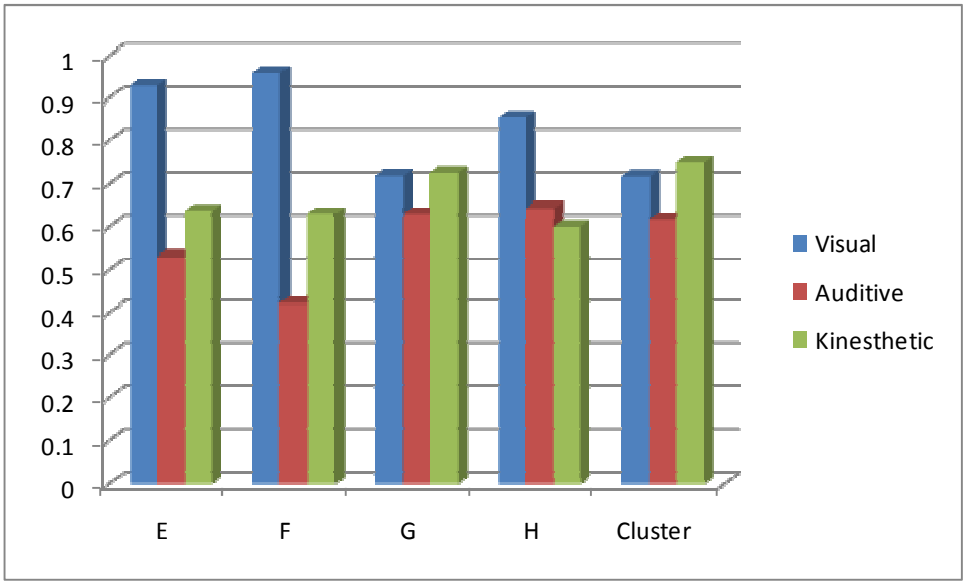

Figura 14. *Escores* para o PostgreSql

Essa percepções são extremamente alinhadas com os resultados apresentados em (Rigby e Hassan, 2007). Esse artigo relata que as métricas coletadas para o desenvolvedor D foram menos associadas com os outros participantes do estudo. Nosso estudo, entretanto, foi além e indicou uma classificação na qual verificamos o perfil do participante e sua função no projeto.

Quanto aos *top committers* do Postgresql, a primeira coisa que o gráfico apresentado na Figura 14 evidencia é que três deles são extremamente visuais. Além disso, o sistema visual tem um *score* alto mesmo para o desenvolvedor G (único não classificado) e a população em geral.

Os *top committers* E, F e G são altamente envolvidos com documentação. Além da documentação, o desenvolvedor G, único que não foi classificado em qualquer categoria, *p-value* 0.085, trabalha especificamente com testes de desempenho e de sintonia, que pode estar relacionado à sua pontuação alta para o sistema cinestésico. Ele também trabalha com grupos de usuários e na negociação para direcionamento geral do projeto, relacionando esse fato a sua pontuação auditiva também relativamente alta. Esse equilíbrio coloca-o realmente como um desenvolvedor multimodal, balanceando bem o uso dos 3 sistemas.

O *top committer H*, com grande vantagem, o mais ativo de todos eles, é visual. Todavia, também tem uma pontuação auditiva elevada, ainda maior do que a sua pontuação cinestésica. Isso pode ser explicado pelo fato dele também estar envolvido com a realização de treinamentos e com a manutenção das listas FAQ e TODO-*list*.

3.4.4.2 Ameaças à Validade

Apesar do fato do Apache e do PostgreSQL serem projetos reais, grandes e maduros, e de nossos resultados parecerem ser bastante consistentes com os perfis obtidos, os *escores* SRP ainda necessitam de mais pesquisas para garantia da validade externa. Um novo estudo foi executado em um ambiente industrial (vide capítulo 5), pois a possibilidade de execução do experimento em um ambiente mais controlado ajuda a aumentar o poder de generalização dos resultados.

Os perfis dos *top commmiters* foram obtidos através dos sites do projetos. Uma melhor análise seria possível com informações mais completas das características de cada um. Visando isso, foi desenvolvido um questionário para caracterizar e avaliar o SRP de engenheiros de software. Esse questionário está disponível publicamente em (Neurominer Survey, 2010) e é descrito no capítulo 4.

Entramos em contato com os *top committers* por e-mail, pedindo-lhes para preencher o questionário desenvolvido. Infelizmente, por questões de tempo e outros motivos que não podemos avaliar, não obtivemos sucesso nesse *feed-back*.

3.5 TRABALHOS RELACIONADOS

Não há trabalhos de utilização de conceitos de Classificação Neuro-Linguística em engenharia de software. Os trabalhos relacionados discutidos aqui serão então divididos

em dois grupos que não se intersectam: (1) estudos científicos de PNL; e (2) estudos de mineração de texto para engenharia de software.

Em se tratando diretamente da PNL, existem alguns artigos científicos apresentando evidências de algumas das suas afirmações. Além disso, existem publicações sobre preferências por determinados sistemas representacionais no processo de cognição e aprendizado, inclusive na área de computação (Sivilotti e Pike, 2007).

A base para os modelos e técnicas apresentadas pela PNL pode ser encontrada em estudos psicológicos que envolvem o chamado "efeito camaleão", o qual diz respeito a estímulos imitativos verbais e não-verbais (*matching*) para o aumento da empatia em uma comunicação. Em (Van Baaren et al., 2003), Van Baaren et al. executaram um experimento em um restaurante no sul da Holanda no qual metade das garçonetes estudadas usou o “efeito camaleão” para atendimento dos clientes. Os resultados mostraram que o valor médio das gorjetas praticamente dobrou para a metade que executou o *matching*. Bailenson e Yee analisaram indivíduos que interagiram com um software baseado em inteligência artificial – um agente para simulação de um indivíduo emitindo uma explicação. O agente que imitou movimentos dos participantes foi mais convincente, recebendo avaliações mais positivas. Foi o primeiro estudo de realidade virtual que mostrou os efeitos de um imitador não-verbal automático para ganho de empatia (Bailenson e Yee, 2005). Essas evidências não provam que as técnicas da PNL são válidas, mas estabelecem uma base empírica para uma investigação mais aprofundada de algumas de suas suposições.

Turam e Stemberger testaram hipótese da PNL de que o processo de *matching* aumenta a empatia na comunicação. A relação entre o *matching* e aumento da empatia foram significativas. A educação também foi relacionada ao aumento da empatia, contudo, mesmo quando houve um controle da mesma, a relação entre o *matching* e a empatia permaneceu significativa (Turan e Stemberger, 2000).

Paolo e Scott, pressupondo a preferência de alguns estudantes pelo processamento cinestésico em determinados contextos, desenvolveram e testaram um conjunto de atividades cinestésicas para um curso de sistemas distribuídos, com alunos de graduação e pós-graduação. O artigo apresenta descrições detalhadas dos exercícios e discute os fatores que contribuem para o sucesso ou insucesso dos mesmos (Paolo e Scott, 2007).

Considerando a mineração de textos em Engenharia de Software, independente da fonte de dados, análises lingüísticas têm sido conduzidas para a compreensão do desenvolvimento de softwares. Witte et al. considerou a importância semântica dos documentos escritos em linguagem natural no processo de manutenção e reengenharia. O resultado da pesquisa foi a criação de um sistema de mineração de texto capaz de preencher uma ontologia de software com informações extraídas desses documentos. Além disso, também é possível uma relação semântica entre os artefatos de softwares representados no código fonte e em linguagem natural (Witte et al., 2008).

Outros trabalhos já consideraram a análise especifica de e-mails. Pattison et al. estudou a relação entre as diversas entidades de um software que são mencionadas em e-mails e o número de vezes que essas entidades estão incluídas nas alterações realizadas (Pattison et al., 2008). Os resultados variaram em intervalos de tempos diferentes, apresentando uma forte relação entre a contagem acumulativa da menção de entidades e suas respectivas alterações.

Existem dois trabalhos que possuem uma proximidade maior com a pesquisa aqui apresentada. No primeiro, Scialdone et al. utilizou e-mails para avaliar a presença social de grupos de manutenção de projetos FOSS (Scialdone et al., 2009). Presença social é uma tática utilizada por membros de comunidades virtuais para projetar suas características pessoais, em outras palavras, o membro tenta transmitir a sensação de ser uma pessoa "real", pois o sentimento que um participante de uma comunidade virtual realmente existe tem sido associado ao sucesso e à coesão desses grupos (Stein et al., 2007)(Sallnas et al., 2000). Foram comparados membros *Core e Peripheral*, e os resultados mostraram que o comportamento de respeito para com a autonomia de outrem pode contribuir para a sobrevivência do grupo e continuidade da produção do software. O trabalho não aborda alternativas para o aumento da presença social e empatia, além disso, baseia-se apenas em aspectos puramente psicológicos e sociais, ou seja, não há uma relação desses aspectos com o contexto da Engenharia de Software.

O segundo trabalho é o de Rigby e Hassan, o qual analisou o conteúdo da lista de discussão do Apache para encontrar a personalidade dos desenvolvedores e conteúdo emocional geral (Rigby e Hassan, 2007). Esse trabalho utilizou uma ferramenta LIWC (*Linguistic Inquiry and Word Count*) para auxiliar as classificações. Essa ferramenta também é oriunda apenas da área de psicologia, não foi desenvolvida para explorar e-mails e não

possui técnicas de mineração de dados para classificações. Contudo, seu dicionário LIWC foi uma fonte inspiradora para a criação do nosso dicionário neuro-linguístico, uma vez que é formado por dimensões com seus significados específicos e por palavras que determinam estes significados (Pennebaker et al., 2001).

3.6 RESUMO

Este capítulo apresentou uma metodologia de classificação de Sistemas Representacionais Preferenciais (SRPs) de desenvolvedores de software e sua automatização através do NEUROMINER (www.neurominer.com), cuja característica principal é a psicometria baseada no uso de termos sensorias. A classificação é contextualizada pelos projetos em que os desenvolvedores estão envolvidos.

A estrutura básica da ferramenta é formada por um dicionário LIWC, similar ao apresentado em (Pennebaker et al., 2001), mas especialmente armazenado em um *Data Warehouse* e criado para identificar o Sistema Representacional Preferencial (SRP) de desenvolvedores de software. Há uma combinação de técnicas de mineração de texto e estatística com predicados sensoriais geralmente considerados em PNL.

O dicionário NEUROMINER possui três perfis básicos (Visual, Cinestésico e Auditivo), bem como os predicados sensoriais correspondentes a cada uma deles, nas línguas inglesa e portuguesa.

Além disso, foi criado um perfil Conceitos para aumentar o poder contextual da classificação. Esses conceitos, ou classes, foram extraídos de uma ontologia para Engenharia de Software. O objetivo é analisar o relacionamento direto entre predicados sensoriais e o contexto da Engenharia de Software. Frases formadas por conceitos da Engenharia de Software, caracterizados ou adjetivados por palavras classificadas como sensoriais no dicionário, são consideradas como perfil do predicado sensorial caracterizador e são pontuadas quando ocorrem no texto minerado. A contabilização de conceitos do domínio combinadas a palavras sensoriais é uma das inovações do NEUROMINER.

A avaliação experimental da abordagem foi realizada em um experimento para identificar o SRP de *top committers* dos sistemas FOSS Apache e PostgreSQL. Os resultados mostraram que os *top committers* são claramente diferentes da população em geral, além disso, uma análise qualitativa também indicou que os escores SRP obtidos estão alinhados com os perfis dos desenvolvedores. Uma evidência empírico-científica de que os perfis neurolinguísticos existem e conseqüentemente podem ser usados para ajudar na alocação de tarefas de Engenharia de Software, bem como podem servir de base para promoção de uma melhor comunicação.

Capítulo 4

Gráficos e descrições textuais têm sido destaque na lista de práticas recomendadas para melhorar a compreensão de software, evidenciando o uso de diferentes sistemas representacionais por engenheiros de software. Neste contexto, este capítulo apresenta os resultados de uma pesquisa de campo, baseada em Questionário e Neuro-Lingüística, para tentar caracterizar os Sistemas Representacionais Preferenciais de engenheiros de software. A Teoria de Resposta ao Item (TRI) foi utilizada para análise das questões e cálculo dos escores de cada participante.

4 QUESTIONÁRIO NEUROLINGUÍSTICO PARA ENGENHEIROS DE SOFTWARE

Em (Fleming, 1995), Fleming apresentou um questionário desenvolvido e usado na *Lincoln University* para identificar as preferências dos alunos por formas específicas de representação da informação (estilos de aprendizagem). Segundo o autor, o questionário pode ajudar os educadores a desenvolver estratégias para se comunicar com os alunos, superando a predisposição de muitos educadores em tratar todos os alunos de forma semelhante. Aplicado aos professores, o questionário também pode mostrar a falta de correspondência entre os seus sistemas preferidos de representação de informação e os sistemas dos alunos. Isto pode motivá-los (os professores) a estimular outros sistemas de representação, atingindo todos os alunos envolvidos.

Nomeado de VARK (*Visual, Aural, Reader/Writer, Kinesthetic)*, o questionário é atualmente a base de um serviço comercial de planejamento educacional (http://www.vark-learn.com/english/page.asp?p=questionnaire). A sigla provém da classificação feita pelo questionário para os estilos de aprendizagem. "V" é para alunos visuais, pessoas que têm preferência por ver (pensar em imagens e usar recursos visuais, como slides, diagramas e folhetos ilustrativos). "A" é para os alunos auditivos, pessoas que aprendem melhor através da

escuta (palestras, debates , fitas, etc). "R" é para os alunos do tipo escritor/leitor, pessoas que aprendem melhor através da visão de palavras impressas. E "K" é para os alunos cinestésicos, pessoas que preferem aprender através da experiência – deslocando-se, tocando e fazendo (por exemplo, a exploração ativa do ambiente, projetos de ciência, experiências, práticas, etc).

A classificação VARK difere da classificação PNL clássica, pois inclui a categoria leitor/escritor além das habituais categorias visual, cinestésica e auditiva. Segundo Fleming, os resultados mostram que, apesar de diagramas e palavras escritas serem visualmente percebidos, existe uma distinção entre a habilidade de alguns alunos para trabalhar com o material simbólico ou gráficos em comparação com aqueles que trabalham com material textual. Os alunos com preferências R e V usam seus olhos para "compreender o mundo", mas eles têm preferências dentro desse modo sensorial. Alguns gostam de texto e outros gostam de esquemas e material icônico - informação que é simbolicamente apresentada (Fleming, 1995).

Outro ponto levantado pelo VARK é que o mesmo indivíduo pode ter perfis diferentes em diferentes áreas (artes marciais, música, línguas, etc) e em diferentes períodos de tempo, ou seja, um indivíduo pode ser visual (V) para aprender artes marciais por um período e tornar-se cinestésico (K) em outro momento.

O questionário de Fleming tem evoluído ao longo dos anos, encontrando-se em sua sétima versão. É composto por 16 questões que têm, cada uma, uma opção relacionada a cada estilo de aprendizagem coberto. Segundo Fleming, a experiência sugere que, se existem muitas perguntas (mais de 25), algumas pessoas podem sentir cansaço ou tédio na hora de preencher o questionário (VARK-LEARN, 2010).

Cada pergunta do VARK tem quatro opções (uma para cada sistema representacional), sendo possível selecionar várias. Isso é desejável porque, apesar das preferências, os seres humanos tendem a ser multimodais, usando múltiplos canais cognitivos. Esta abordagem, contudo, torna mais difícil a análise da pontuação do questionário. Leite et al. (Leite e Svinicki, 2010) consideram que a pontuação utilizada para o VARK é arbitrária, pois se baseia em desvio padrão, criando uma dependência da existência de uma população de referência para normalizar os escores dos indivíduos. Além disso, o VARK considera que todas as perguntas são equivalentes no que diz respeito à medição do traço latente.

O questionário utilizado no *survey* apresentado neste capítulo segue uma abordagem semelhante à do VARK, mas é fortemente contextualizado para Engenharia de Software. Ele usa apenas os três estilos originais (VAK ou VAC – Visual, Aural e Cinestésico) propostos pela teoria da PNL, desconsiderando o estilo VARK leitor/escritor. No contexto do nosso *survey*, a "escrita" representa a escrita do código propriamente dito, sendo considerada uma atividade prática e cinestésica.

Contrariamente ao VARK, nosso questionário impõe preferência nas respostas de múltipla escolha, ou seja, quando uma pessoa é multimodal (utiliza mais de um sistema na situação proposta pela questão), exige-se uma ordenação das respostas. O objetivo é capturar a estratégia preferida do indivíduo.

Finalmente, nossa abordagem não tem uma pontuação arbitrária como a do VARK (Leite e Svinicki, 2010), pois consideramos que as questões são diferentes no que diz respeito à medição do traço latente. Usamos a Teoria de Resposta ao Item (TRI) (vide capítulo 2) para estimar a discriminação e dificuldade das questões. Como visto anteriormente, este método coloca a dificuldade dos itens na mesma escala de capacidade de uma pessoa. Desta forma, qualquer ponto da escala descreve tanto a dificuldade de um item como o escore de uma pessoa para o traço latente medido.

O traço latente é a habilidade/proficiência que se quer mensurar. Basicamente, se diz que quanto maior a proficiência do examinado, maior a probabilidade de ele responder corretamente ao item. Na nossa abordagem, temos 3 traços latentes para medir: as habilidades/ preferências visual, auditiva e cinestésica.

Com um número razoável de respostas, estimamos os parâmetros de cada item: dificuldade (b) e discriminação (a), através de um processo estatístico. A estimativa da habilidade do examinado é o valor da escala mais coerente com o conjunto de respostas dele, também obtido através de um processo estatístico. Assim, como nosso questionário foi analisado através de TRI, ele apresenta questões que representam diferentes níveis de dificuldade. Além disso, calcular a proficiência de cada um dos respondentes levou em consideração quais itens (questões) ele respondeu corretamente.

A concretização deste processo estatístico é apresentada a seguir.

4.1 SEGUNDO EXPERIMENTO

4.1.1 Definição de objetivo

O principal objetivo do nosso estudo é identificar quais os sistemas representacionais são os preferidos pelos Engenheiros de Software. Esse objetivo é formalizado utilizando o modelo GQM proposto por Basili (Basili e Weiss, 1984) e apresentado em (Solingen e Berghout, 1999):

Analisar os sistemas representacionais utilizados
com a finalidade de caracterização
em relação aos Sistemas de Representação Preferenciais
do ponto de vista de Engenheiros de Software
no contexto de tarefas de desenvolvimento e manutenção de software.

4.1.2 Planejamento

4.1.2.1 Seleção de contexto

Algumas das mais consolidadas metodologias de representação de software (ditas seguras e produtivas) são baseadas em uma exaustiva utilização de diagramas (Moody, 2009) (diagramas UML, por exemplo). Estas metodologias podem beneficiar pessoas com perfil visual (V), devido ao uso de símbolos visuais, inerentes a esta forma de representação. Como

esta não é a única maneira de representar e compreender o software, nós questionamos: "engenheiros de software têm preferências por outros sistemas representacionais?".

Para responder a essa pergunta, nós criamos um questionário baseado na teoria Neuro-Lingüística, levantando questões específicas para verificar quais os Sistemas Representacionais Preferidos pelos Engenheiros de Software. Os participantes responderam ao questionário e submeteram suas respostas *on-line*. Os dados coletados foram analisados usando TRI, a qual serviu de base para determinar o SRP de cada participante.

4.1.2.1.1 Formulação da hipótese

Engenheiros de Software exercitam a multimodalidade, ou seja, constantemente estimulam e utilizam os sistemas visual, auditivo e cinestésico. Contudo, em contextos específicos, podem ter preferências quanto à ordem de utilização dos mesmos, indicando a melhor estratégia de compreensão e de priorização de artefatos, na ausência da possibilidade de usar os 3 sistemas. Desta forma, a hipótese que queremos confirmar é a seguinte:

> **Hipótese nula H_0:** Engenheiros de Software têm os três sistemas balanceados, apresentando níveis próximos nos escores gerados pelo modelo TRI, ou seja, não existem diferenças significativas entre a primeira, segunda e terceira classificações dos sistemas.
>
> H_0^{SRP}: $\theta_1 = \theta_2 = \theta_3$.
>
> **Hipótese alternativa H_1**: Engenheiros de Software utilizam preferencialmente um sistema, o qual indica a melhor estratégia de compreensão e de priorização de artefatos, ou seja, existem diferenças significativas entre a primeira, segunda e terceira classificações dos sistemas.
>
> H_1^{SRP}: $\theta_1 > \theta_2 > \theta_3$,
>
> onde θ_1, θ_2 e θ_3 são os escores correspondentes às primeiras, segundas e terceiras classificações, respectivamente.

4.1.2.1.2 Seleção de participantes e objetos

Os primeiros participantes foram convidados por conveniência, utilizando os contatos do autor desta tese na indústria e na academia. Além disso, vários profissionais da comunidade FOSS foram convidados e preencheram o questionário. O questionário está disponível na Internet para que mais profissionais possam, voluntariamente, contribuir com esta pesquisa.

4.1.2.1.3 Instrumentação

A hipótese foi testada através de um questionário disponível na Internet. O questionário foi construído com base em uma adaptação do VARK, apresentando situações do cotidiano de engenheiros de software nas quais eles podem utilizar os três sistemas representacionais: visual, auditivo e cinestésico (VAK).

O questionário pode ser acessado em (www.neurominer.com/survey) e consiste de duas partes:

(1) A primeira parte é composta de 16 questões relacionadas às tarefas habituais de Engenheiros de Software (vide Tabela 7). Cada questão é constituída por três alternativas, cada uma referente a um sistema de representação que pode ser utilizado pelo participante na situação proposta. Um limiar de questões deve ser respondido, atribuindo-se números às alternativas, de acordo com a preferência do participante. Desta forma, o participante deverá responder no mínimo 12 questões (75%) para ter seu questionário submetido, podendo numerar preferencialmente as três alternativas presentes nas questões respondidas (sendo 1 a de maior preferência e 3 a de menor), ou numerar apenas a(s) alternativa(s) de maior preferência.

A fim de evitar efeitos indesejáveis de memória e as tendências para usar a mesma resposta dada nas perguntas imediatamente anteriores, Hill e Hill (Hill e Hill, 1998) sugerem a

utilização de uma Tabela de números aleatórios para escolher a posição das perguntas no questionário. Seguindo esse princípio, em cada nova entrada na página, o questionário cria uma ordem aleatória, não só para as alternativas, como também para as questões. Além disso, existem questões que remetem ao mesmo assunto, para que o participante seja avaliado em relação à coerência com que responde ao questionário.

(2) A segunda parte consiste na caracterização do participante: posição (analista, gerente ou programador), gênero (masculino, feminino), experiência em manutenção (1-3 anos, 3-5 anos, 5-10 anos, mais de 10 anos), a experiência expressa como o número de sistemas mantidos (1-5, 6-10, 11-20, mais de 20) e utilização prévia de uma ferramenta de visualização de software (sim ou não). O participante também pode fornecer informações tais como país e estado de origem, além de indicar seu endereço de e-mail, caso deseje receber os resultados do questionário.

Tabela 7 – Questionário em português

Questionário
1. Quando é alocado em um novo sistema, inicialmente você prefere…
a - ver um diagrama UML? - V
b - conversar consigo mesmo e com outras pessoas? - A
c - primeiro pegar o sistema pra usar e sentir como o mesmo pode evoluir? - K
2. Quando você tem uma tarefa de manutenção para executar, prefere…
a - expor suas ideias, independente de qualquer coisa? - K
b - imaginar diferentes perspectivas? - V
c - ouvir e falar sobre as opções de resolução da tarefa? - A
3. Quando implementa uma classe, a qual é testada com sucesso, prefere…
a - falar a novidade para todos da equipe?- A
b - projetar um diagrama claro, para que todos vejam? - V
c - cumprimentar ou dar um tapinha nas costas de cada um? - K
4. Quando entra em pauta a possibilidade de uma mudança de ferramenta ou metodologia, você prefere…
a - debater as opções? - A
b - imaginar as possibilidades? - imagining the possibilities? - V
c - ter uma atitude flexível? - K
5. Nos treinamentos, você prefere…
a - pegar a essência da mensagem? - K
b - ouvir a mensagem, palavra por palavra? - A
c - resumir o significado? - V
6. Quando precisa compreender uma nova classe, você inicialmente prefere…
a - ver uma imagem que apresente a classe em questão? - V
b - escutar alguém falar da mesma? - A
c - navegar e executar um teste? - K
7. No levantamento de requisitos, você prefere…
a - ver um desenho geral da visão dos usuários? - V
b - ouvir atentamente os comentários dos usuários? - A
c - sentir a pressão das necessidades? - K
8. Na definição da arquitetura de um software, você prefere…
a - ter um desenho abrangente da situação? - V
b - ouvir e discutir amplamente as ideias? - A
c - emitir várias sugestões? - K
9. Se precisar programar em local diferente do habitual, você prefere…
a - viver primeiro a experiência de como o dia passará? - K
b - pensar e tentar visualizar o dia atípico que terá? - V
c - conversar sobre sua programação (do dia)? - A
10. Quando precisa utilizar uma nova biblioteca, você prefere…
a - conversar com alguém que tem condições de discutir tal uso? - A
b - buscar a visão de um especialista? - V
c - utilizar a experiência de outro programador? - K
11. Para adquirir uma bibliografia sobre Engenharia de Software. Você se baseia:
a - num visual atraente e útil (dentro e fora do livro) - V
b - um profissional ter falado sobre a mesma e recomendado - A
c - presença de estudos de caso reais e exemplos - K
12. Quando entrevista um programador menos experiente para trabalhar contigo, você prefere…
a - examinar todos os aspectos de seu potencial? - V
b - fazer perguntas sobre os comentários em seu currículo? - A
c - ter domínio das ferramentas e possíveis conhecimentos experimentados pelo entrevistado? - K
13. Para aprender uma nova linguagem de programação, paradigma ou padrão de projeto. Você prefere:
a - conversar com profissionais que conhecem a técnica - A
b - começar a usar a técnica para aprender fazendo - K
c - observar os diagramas da documentação - V
14. Ao usar um site de uma ferramenta de engenharia de software, o que você gostaria de encontrar:
a - a oportunidade de perguntar e falar sobre a ferramenta e suas características. - A
b - screenshots mostrando o que cada módulo faz. - V
c - muitos exemplos de bons e maus usos da ferramenta - K
15. Você foi convocado repentinamente para apresentar sua mais nova implementação. Como você prepararia a apresentação ? Pense no que você gostaria de ver primeiro, se estivesse no público. Lembre também que há pouco tempo para preparar.
a - faria diagramas ou gráficos para apresentar. - V
b - escreveria os pontos-chave do código e praticaria o discurso. - A
c - reuniria exemplos reais e práticos para explicar - K
16. Lembre agora de um momento que você compreendeu totalmente um código novo. Como você compreendeu melhor?
a - executando e escrevendo código para algum tipo de manutenção. - K
b - escutando as explicações do programador e fazendo perguntas. - A
c - usando diagramas e gráficos. - V
(A -resposta auditiva; K - resposta cinestésica; V - resposta visual)

4.1.3 Resultados

Duzentos e nove (209) engenheiros de software responderam ao questionário, em um período de oito meses (de março a outubro de 2010). Todas as submissões foram feitas através da Internet.

País de Origem: a maioria dos participantes foi brasileira (89%), mas pessoas de diversos países participaram. Brasil = 186 (89%), United States = 6 (2,9%), Canada = 4 (1,9%), India = 3 (1,4%), China = 2 (1%), Afghanistan = 1 (0,5%), Andorra = 1 (0,5%), Argentina = 1 (0,5%), Finland = 1 (0,5%), German = 1 (0,5%), Ireland = 1 (0,5%), Romania = 1 (0,5%) and Minor Outlying Islands = 1 (0,5%).

Posição: gerentes = 47 (22,5%), analistas = 88 (42,1%) e programadores = 74 (35,4%). Vários analistas contatados para dar opiniões sobre a pesquisa afirmaram que também exercem atividades de programação.

Experiência (anos): 1-3 anos = 88 (42,1%), 3-5 anos = 48 (23%), 5-10 anos = 35 (16,7%), mais de 10 anos = 38 (18,2%). (Vide Figura15).

Experiência (número de sistemas mantidos): 1-5 = 108 (51,7%), 6-10 = 56 (26,8%), 11-20 = 20 (9,6%), mais de 20 = 25 (12%). (Vide Figura 16).

Sexo: Feminino = 39 (18,7%), masculino = 170 (81,3%).

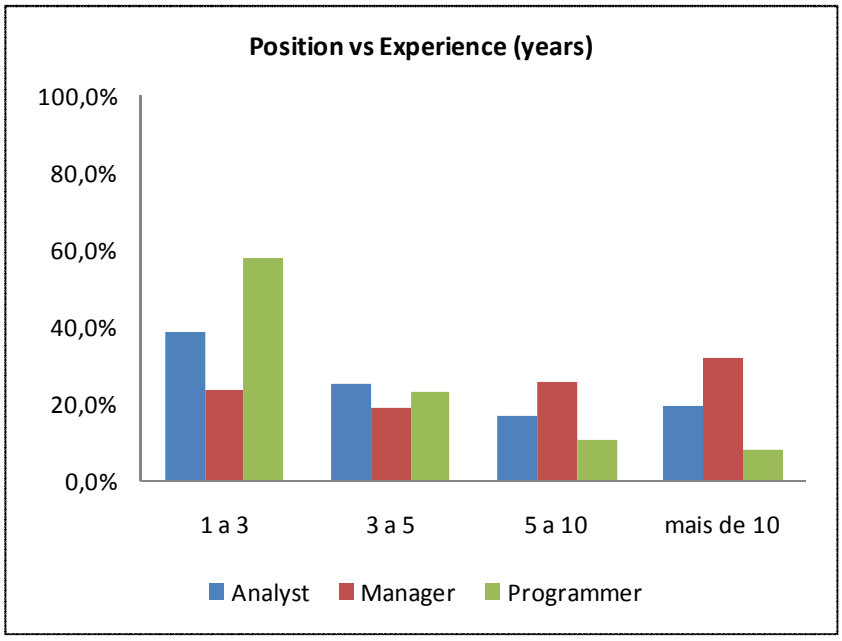

Figura 15. Gráfico de Posição x Experiência em anos

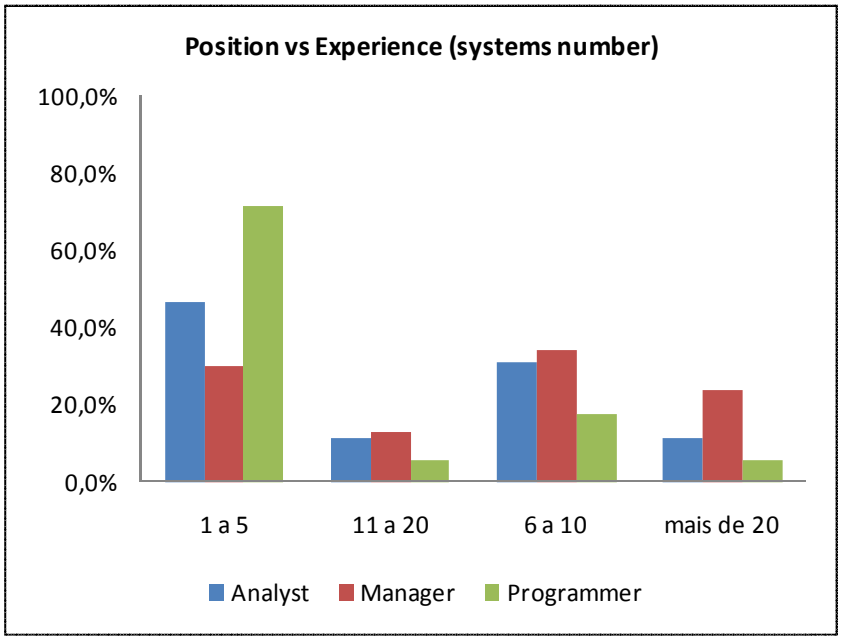

Figura 16. Gráfico de Posição x Experiência expressa em número de sistemas mantidos

4.1.3.1 TRI

A estimação dos parâmetros dos itens (questões) foi feita por meio do Software R (http://www.R-project.org), usando o pacote LTM (*Latent Trait Model* - modelo de variáveis latentes para dados dicotômicos), o qual implementa o modelo logístico de dois parâmetros (Rizopoulous, 2006).

O modelo dicotômico de dois parâmetros foi utilizado pelo fato de termos três variáveis latentes diferentes, as quais não podem estar numa mesma escala de aptidão. Em outras palavras, não são três alternativas de respostas que medem o mesmo traço latente. Cada questão possui três alternativas, as quais podem ser escolhidas em ordem de preferência e cada uma representa uma variável latente (Visual, Cinestésica ou Auditiva). Assim, para efeitos estatísticos, consideramos três questionários iguais para cada variável, tornando as respostas dicotômicas.

A entrada de dados para o Software R foi feita em três etapas: (1) atribuição de valor 1 (um) para as respostas referentes ao sistema visual e 0 (zero) para demais respostas; (2) atribuição de valor 1 (um) para as respostas referentes ao sistema aural e 0 (zero) para demais respostas; e (3) atribuição de valor 1 (um) para as respostas referentes ao sistema cinestésico e 0 (zero) para demais respostas. Esses três arquivos foram analisados individualmente no Software R, com o qual obtivemos os resultados.

As curvas características dos itens foram computadas para cada item (questão) dentro do intervalo de -4 a +4 desvios padrões. Este intervalo geralmente é usado para visualização do máximo de informação. Nas 16 questões, observamos o comportamento do parâmetro de discriminação. Houve uma variação considerável em todos os três sistemas (Visual: 0,17 - 1,49; Aural: -0,08 - 1,27; e Cinestésico: 0,21 - 2,16). Para este parâmetro, esperam-se valores entre 0 e +2, sendo mais apropriados valores acima de 1.

Os gráficos ilustrados nas Figuras 17, 18 e 19 mostram as CCIs para as três questões mais e menos discriminatórias, para cada um dos três sistemas. As questões 6, 13 e 16 apresentaram maior discriminação para o sistema visual. Questões 6, 8 e 13 apresentaram maior discriminação para o sistema cinestésico. Questões 6, 16 e 2 apresentaram maior

discriminação para o sistema aural. É possível observar curvas mais acentuadas para todas estas questões.

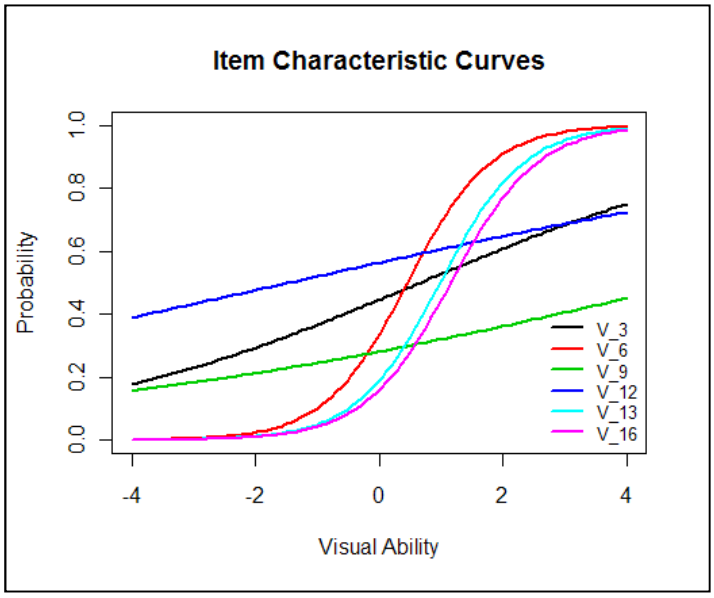

Figura 17. CCI dos 3 itens mais discriminantes e dos 3 menos discriminantes do sistema visual.

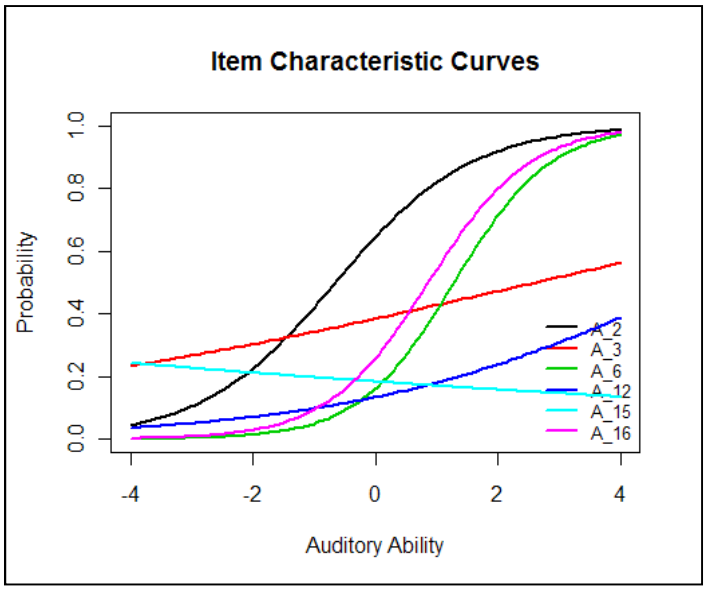

Figura 18. CCI dos 3 itens mais discriminantes e dos 3 menos discriminantes do sistema auditivo.

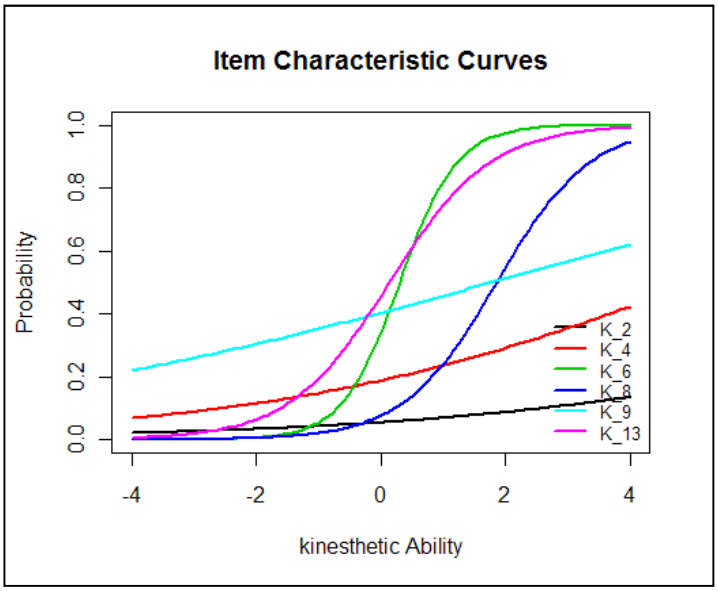

Figura 19. CCI dos 3 itens mais discriminantes e dos 3 menos discriminantes do sistema cinestésico.

Quanto ao parâmetro de dificuldade (b_i), o qual indica a região, na escala proposta, em que o item possui maior informação, houve muita variação para os 3 sistemas.

Tabela 8 – Parâmetro de dificuldade

Parâmetro de Dificuldade			
Questão	**Visual**	**Aural**	**Cinestésico**
1	0,646337	1,174751	0,9298117
2	0,902932	-0,65451	11,3411584
3	0,666455	2,620564	2,4384595
4	1,158724	0,123942	5,0827783
5	3,428154	3,675097	-1,2966636
6	0,456912	1,285889	0,3037288
7	1,263104	0,101664	3,0356221
8	0,238094	0,455426	1,8603075
9	5,067427	1,574619	1,7733581
10	1,340983	0,911794	2,0398499
11	2,96256	0,200122	0,3844435
12	-1,45069	5,285809	1,7353657
13	0,983909	1,515933	0,1381952
14	0,984837	2,522925	0,136623
15	0,874704	-16,7235	0,3147457
16	1,159074	0,863275	0,1625408

Esse parâmetro não representa a qualidade do item, pois observamos que os itens muito difíceis e os muito fáceis não apresentam boa discriminação. Os três itens com maior dificuldade foram, em ordem decrescente, os itens 9, 5 e 11 (visual); 12, 5 e 3 (aural) e 2, 4 e 7 (cinestésico). Já os itens com menor grau de dificuldade, em ordem crescente, foram: 12, 8 e 6 (visual); 15, 2 e 7 (aural) e 5, 14 e 13 (cinestésico).

Exemplificando: para dar uma resposta referente ao sistema visual, no item 9 (vide Tabela 8), deve-se ter um maior traço latente, ou seja, a habilidade visual deve ultrapassar a dificuldade do item. Já o item com menor dificuldade, item 12 (vide Tabela 8), é respondido por pessoas que tem baixa habilidade visual, ou seja, este item é respondido pela maioria dos Engenheiros de Software. Se o sujeito não endossou essa resposta, sua habilidade visual foi inferior à dificuldade do item.

Após a obtenção dos parâmetros das questões, procederam-se os cálculos dos escores de cada participante. O escore de cada participante foi obtido com o software R, através do cálculo algébrico do modelo de dois parâmetros.

Como seria inviável fazer um gráfico com as freqüências da pontuação, já que a mesma é contínua, intervalos de classes foram criados para que fosse possível ver o comportamento da população (distribuição) nos três sistemas. Os escores foram divididos em nove grupos de intervalos, variando de -1,541 a 2,636.

Os escores estimados têm pouca variação entre si. A pontuação, como alega a TRI, apresenta uma distribuição normal, o que implica que a maioria da população tem nível de habilidade médio para os 3 sistemas (vide Figura 20).

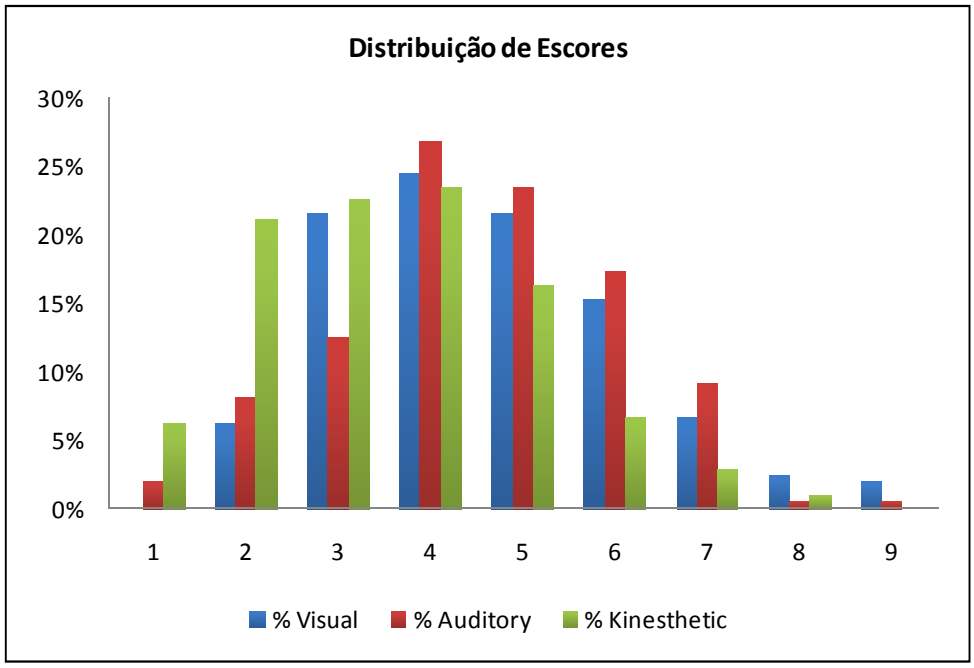

Figura 20 – Distribuição dos escores

Na etapa seguinte, procederam-se as classificações. Assim, quanto maior a pontuação obtida para um sistema, maior a prioridade do indivíduo para o mesmo. A seqüência das letras indica esta ordem, por exemplo, KAV indica Cinestésico (*Kianesthetic*) como primeira preferência e Visual como terceira. As classificações encontradas foram: AVK – 18.7%, KAV – 18.2%, VKA – 18.2%, AKV- 15.8%,VAK – 14.4%, KVA – 12.9%, VK – 0.5%, VA – 0.5%, VK- 0.5% e V – 0.5% . A Figura 21 mostra a representação gráfica dessas classificações. A TRI calcula um escore mesmo para indivíduos que respondem incorretamente todas as questões, contudo, identificamos os examinados que não endossaram nenhum item para algum sistema, permitindo as combinações: VK, VA, V e KV.

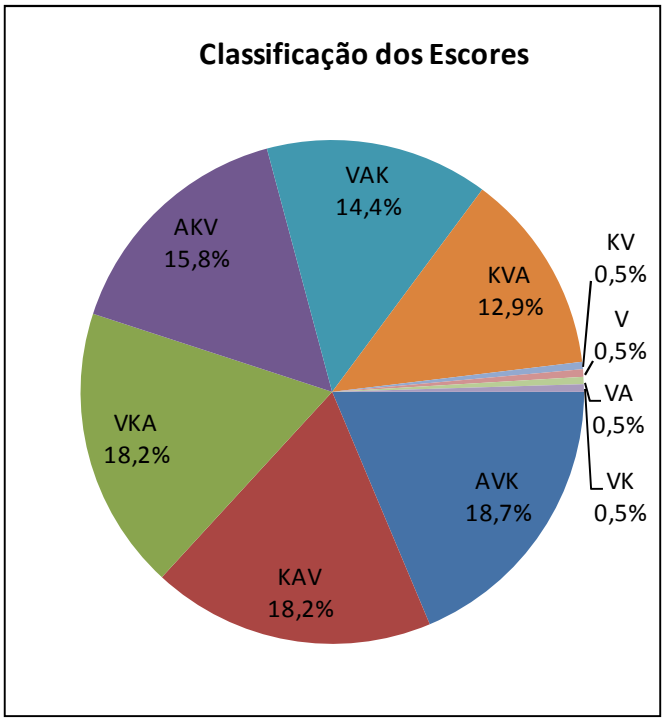

Figura 21. Classificação dos escores

Apesar de a TRI garantir classificações invariantes, verificamos se houve diferenças nas classificações, dependendo das posições e experiência. Através do teste χ^2, observamos as freqüências esperadas e observadas, concluindo se houve ou não diferença significativa entre as mesmas. Para Analista, Gerente e Programador não houve diferença nas classificações, aos níveis de significância de 5% e 10%. Considerando a experiência, também não houve diferença nas classificações, aos níveis de significância de 5% e 10%. Assim, independente da posição ou experiência, a estratégia cognitiva mais usada entre os entrevistados é a AVK (18.7% do total de indivíduos).

Por fim, testamos a hipótese anteriormente formulada. O objetivo foi provar que a primeira classificação apresenta diferença significativa entre a segunda e terceira, ou seja, a primeira classificação indica o canal preferido pelos Engenheiros de Software.

Para realizar essa análise, usamos o teste ANOVA e *Tukey* HSD. A ANOVA indica a probabilidade de que a hipótese nula seja verdadeira, ou seja, a probabilidade de que nenhuma diferença existe entre quaisquer das classificações. Se a hipótese nula (H0) é rejeitada, há indício de que há diferença de potência em alguma das classificações testadas.

Para isto, utilizamos os escores dos três sistemas associados com a ordem que eles representam. Neste momento, a pontuação do individuo foi o foco. Observamos todas as pontuações referentes às preferências (*data points*), independente do sistema, perfazendo 209 pontuações referentes à primeira preferência, 209 pontuações referentes à segunda preferência e 209 pontuações referentes à terceira preferência.

A partir desse principio, analisamos se as pontuações são diferentes, dependendo de qual ordem pertença: primeira, segunda ou terceira. A conclusão do teste é obtida com os valores de significância. Valores baixos indicam diferenças entre as médias dos grupos (valor de $p<0,05$), e que há, pelo menos, uma diferença entre os grupos analisados, permitindo rejeitar a hipótese nula. Então, se isso acontece, o teste Tukey é utilizado para identificar entre quais grupos esta diferença se evidencia.

A Tabela 9 apresenta os resultados para o ANOVA. A hipótese nula é rejeitada com uma significância muito elevada ($p << 0,05$ – vide coluna Sig.), ou seja, engenheiros de software têm um SRP. Há pelo menos uma diferença entre os grupos analisados. Além disso, o teste Tukey identificou que todos os grupos (1,2 e 3) são diferentes uns dos outros ($p << 0,05$ - vide Tabela 10, coluna Sig.).

Tabela 9 – ANOVA para as classificações

ANOVA

Score

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	254,761	2	127,380	821,897	,000
Within Groups	96,710	624	,155		
Total	351,471	626			

Tabela 10 – Teste de Tukey para as classificações

Multiple Comparisons

Score Tukey HSD

(I) Opition	(J) Opition	Mean Difference (I-J)	Std. Error	Sig.	Interval	
					Lower Bound	Upper Bound
1	2	,91902	,03851	,000	,8285	1,0095
	3	1,55266	,03851	,000	1,4622	1,6431
2	1	-,91902	,03851	,000	-1,0095	-,8285
	3	,63364	,03851	,000	,5432	,7241
3	1	-1,55266	,03851	,000	-1,6431	-1,4622
	2	-,63364	,03851	,000	-,7241	-,5432

4.1.3.2 Análise e interpretação

Os resultados indicaram que, de acordo com o questionário, os engenheiros de software têm diferentes canais preferenciais. Os testes ANOVA e Tukey mostraram que há uma diferença significativa que nos permite rejeitar a hipótese nula. No entanto, existem muitas outras informações interessantes presentes nos dados.

A ordem de preferência para os sistemas representacionais é muito equilibrada entre os participantes. O maior grupo foi o AVK (18,4%) e o menor foi o KVA (12,9%). Esta é uma diferença pequena, considerando que temos seis ordens possíveis de preferências (AVK, AKV, VAK, AVK, KAV, KVA). Isso indica que não existe apenas uma diferença significativa na ordem de preferências, mas também que a própria população é muito diversificada em termos de SRP.

Os resultados também mostraram que um número extraordinário de engenheiros de software é multimodal (98% dos participantes), ou seja, utiliza os 3 canais. Existem grupos consideráveis que preferem os sistemas auditivo, visual ou cinestésico. Então, a área de Engenharia de Software não deve negligenciar nenhum sistema

O canal auditivo é a preferência de 34,5% dos participantes. Isto é interessante, considerando que a maioria das ferramentas da área ainda tem suporte limitado para

comunicação e gravação integradas de áudio. Além disso, com exceção da engenharia de requisitos, poucos métodos de engenharia de software enfatizam a comunicação sonora. Não seria viável associar explicações em áudio para arquitetura do software e artefatos de código, por exemplo? Por que não gravar as entrevistas com os usuários e disponibilizá-las para programadores compreenderem melhor uma classe? Os resultados aqui apresentados são estimulantes para consideração dessas opções.

A Teoria de Resposta ao Item mostrou-se robusta para ajudar a medir as características subjetivas de seres humanos e conseqüentemente potencialmente muito útil para a comunidade de Engenharia de Software Experimental. A técnica pode ser empregada para medir habilidades e preferências de engenheiros de software de populações, domínios e contextos de projetos heterogêneos, contribuindo para dirimir problemas de meta-análise na área.

O autor do questionário VARK publicou em seu site (http://www.vark-learn.com) dados estatísticos sobre as classificações de seus respondentes e mostrou que as distribuições das preferências tendem à normalidade. Isso é usado como um parâmetro de ajustamento dos dados ao modelo utilizado para classificação da população usuária. Porém, além da normalidade ser esperada quando se analisa grandes amostras, a mesma não pode ser justificativa para inferir o desempenho de um modelo. A forma como é obtida a pontuação que determina a preferência do individuo ainda necessita de mais validações, pois o modelo utilizado não pondera as questões, não há discriminação entre as mesmas e todas são analisadas sem distinção, fornecendo indícios de fragilidade da análise.

O VARK também tem seus resultados validados pelos respondentes, todavia, uma variável latente não pode ser medida diretamente (Wu e Adams, 2007), ou seja, não é interessante uma estatística em que o examinado é quem valida. O individuo não é capaz de precisar, por sua percepção, o grau como certas variáveis estão distribuídas (Leite e Svinicki, 2010). Portanto, para medir o grau com que estas variáveis estão distribuídas, faz-se necessário o uso de um modelo como a TRI.

Por fim, é importante ressaltar que a análise TRI pode ajudar a melhorar questionários como o nosso. A técnica evidenciou que as questões 15, 2 e 9 são respectivamente pouco discriminatórias para os sistemas auditivo, cinestésico e visual. Na questão 15, provavelmente a palavra *ponto-chave*, na opção auditiva, influencia a escolha do

respondente. Consideramos essa palavra apelativa para o contexto da questão. A questão 2 também pode indicar uma descrição ruim para o item cinestésico, uma vez que o mesmo trata de uma atitude incondicional, inibindo a resposta. Este foi o item (questão 2) cinestésico mais difícil, como pode ser observado na Tabela 8. Na questão 9, observando o parâmetro de dificuldade (vide Tabela 8), denota-se que a situação não coube para o contexto de Engenheiros de Software. Isso quando se trata de imaginar uma situação de ambiente de trabalho diferente. A resposta visual dessa questão foi a mais difícil de ser respondida.

4.1.3.3 Ameaças à validade

Diferentes problemas podem ser ocasionados durante a participação dos indivíduos no questionário:

- Instrumentação adequadamente preparada (**validade interna**): Os participantes responderam ao questionário sem nenhuma supervisão, assim, há a probabilidade dos mesmos não terem entendido uma questão específica. Para mitigar esse tipo de problema, o questionário foi aplicado a 9 profissionais de Engenharia de Software (entre eles, dois com inglês nativo), os quais não participaram do questionário, mas contribuiram para modificações, tornando-o mais claro e objetivo.
- População representativa (**validade externa**): A variedade dos participantes que responderam ao questionário foi signitifcativa, contudo, será interessante aumentar o número de participantes, aumentando também a representatividade junto à população geral de engenheiros de software.
- Distribuição do conjunto de participantes (**validade de conclusão**): A experiência dos engenheiros de software ou as funções podem afetar os resultados, entretanto, tanto a experiência quanto os cargos ocupados pelos participantes do questionário estão distribuídos (vide seção resultados deste capítulo).

4.2 RESUMO

Neste capítulo, foi apresentada uma abordagem experimental de pesquisa de campo com engenheiros de software, a qual tenta caracterizar o uso preferencial dos sistemas de representação no processo de compreensão do software.

Foi produzido e publicado (www.neurominer.com/survey) um questionário específico para tentar capturar as formas preferidas de compreender software utilizadas por analistas e programadores. Ao total, 233 engenheiros de software responderam ao questionário, dentro de um período de 6 meses (março a agosto de 2010). Como todas as submissões foram feitas pela Internet, dos 233, apenas 209 foram considerados válidos para análise.

As pontuações foram analisadas e validadas pela TRI, a qual se mostrou extremamente eficiente na identificação do poder de discriminação dos itens, permitindo a indicação do uso de TRI para pesquisas de campo na área de engenharia de software.

O cálculo TRI para estimação dos parâmetros das questões foi feito por meio do Software R (http://www.R-project.org), usando o pacote LTM (*Latent Trait Model*- modelo de variáveis latentes para dados dicotômicos), o qual implementa o modelo logístico de dois parâmetros.

Os resultados encontrados para as ordens preferidas dos engenheiros de software na compreensão e priorização de artefatos foram: AVK – 18.7%; KAV – 18.2%; VKA – 18.2%; AKV – 15.8%; VAK – 14.4%; KVA – 12.9%; VK – 0.5%; VA – 0.5%; VK – 0.5% e V – 0.5%.

Vale ressaltar que essas classificações não são estanques, ou seja, elas representam apenas uma estratégia preferida de uso dos 3 sistemas. A grande pontuação recebida pelo sistema auditivo pode ser um indicativo interessante de que a área precisa investir mais em estímulos multimídia para compreensão de software. De qualquer forma, o uso dos 3 sistemas apresentou-se extremamente balanceado, evidenciando que nenhum dos 3 sistemas deve ser negligenciado pela área de engenharia de software.

Capítulo

5

Neste capítulo, é detalhado um experimento que objetivou reunir as três abordagens para avaliação final deste trabalho, envolvendo: (1) classificações do NEUROMINER; (2) questionário neuro-lingüístico; e (3) utilização de uma ferramenta de visualização de software (SourceMiner), observando o comportamento dos programadores ao usar a mesma. O experimento foi realizado em um ambiente industrial controlado.

5 EXPERIMENTO FINAL: COMBINANDO AS ABORDAGENS

5.1 DEFINIÇÃO DO OBJETIVO

O principal objetivo do estudo é avaliar se os SRPs classificados pelo NEUROMINER e pelo questionário neuro-lingüístico são evidenciados, quando programadores utilizam uma ferramenta de compreensão de software que explora profundamente um dos sistemas representacionais. Esse objetivo é formalizado utilizando o modelo GQM proposto por Basili (Basili e Weiss, 1984) e apresentado em (Solingen e Berghout, 1999):

Analisar programadores da indústria
com a finalidade de avaliação
em relação aos SRPs classificados pelo NEUROMINER e pelo *Survey*
do ponto de vista de pesquisadores de engenharia de software
no contexto de utilização da ferramenta SOURCEMINER.

5.2 PLANEJAMENTO

O experimento tem como alvo programadores de projetos de código fechado. Como parte do experimento, faz-se necessário replicar, para programadores da indústria, o experimento e a pesquisa de campo, descritos nos capítulos 3 e 4, respectivamente. Desta forma, obtêm-se as classificações do NEUROMINER e do *Survey* para a referida população. Após esta classificação, os programadores serão convidados a identificar *Code Smells* com a utilização de um ambiente de desenvolvimento enriquecido com recursos de visualização software. Os dados de utilização destes recursos serão então analisados e comparados com os perfis previamente identificados com o NEUROMINER e o *Survey*.

5.2.1 Formulação da hipótese

Os assuntos que estamos tentando explorar são os seguintes:

(1) Estamos interessados em verificar se programadores da indústria terão um SRP.
(2) Queremos analisar se programadores que não possuem um SRP visual sentem mais necessidade de acessar o código fonte na realização de tarefas de compreensão de software, especialmente auxiliadas pela ferramenta visual *SourceMiner*.
(3) Finalmente, verificaremos a Precisão (P) e Cobertura (R) (vide seção 2.1.1.1) das classificações feitas pelo NEUROMINER em relação ao *Survey*. Também será calculada a média harmônica dessas duas medidas.

Calculando a média harmônica, há a combinação dos benefícios das duas medidas (Grossman e Katz, 1986). A média harmônica, também chamada de *F-measure*, ou medida F, assegura o equilíbrio da caracterização do modelo.

A medida F é dada por:

$$F - measure = \frac{2 \times P \times R}{P + R}$$

Considerando a dificuldade de se obter a liberação de programadores da indústria para realização de experimentos, bem como a liberação de e-mails enviados relativos a projetos, a amostra é pequena. Conseqüentemente, devido ao baixo número de programadores liberados pela empresa parceira, **um teste estatístico formal não foi executado para as segunda e terceira questões**. Estas questões serão analisadas qualitativamente.

Formalmente, vamos confirmar então somente a hipótese relativa à primeira questão, verificando se programadores da indústria terão um SRP. Esta hipótese é a mesma do experimento apresentado no capítulo 3:

> **Hipótese nula H_0:** Programadores da indústria têm a mesma média mensal para os 3 e*scores* dos sistemas representacionais.
>
> H_0^{SRP}: μ(*Visual Final Weight*) = μ(*Auditive Final Weight*) = μ(*Kinesthetic Final Weight*).
>
> **Hipótese alternativa H_1**: no mínimo uma das médias é diferente das outras.

5.2.2 Seleção de participantes e objetos

A escolha dos programadores foi por conveniência. O autor desta tese conseguiu a liberação de cinco programadores de uma empresa da qual o mesmo é consultor, já apresentada no capítulo 3.

Por questões legais, não utilizaremos os nomes dos participantes neste trabalho. Letras serão utilizadas para identificar cada desenvolvedor. A Tabela 11 lista estes programadores junto com duas medidas de experiência em manutenção de software.

A execução foi, em parte, não-intrusiva, pois os dados foram retirados diretamente das listas de discussão dos programadores, sem nem mesmo o desenvolvedor saber que os e-mails postados seriam analisados um dia. Todavia, por outro lado, foi necessário requisitar um tempo extra dos desenvolvedores para realização de tarefas experimentais de compreensão com o *SourceMiner* e para o preenchimento do questionário neuro-lingüístico (*Survey*).

Tabela 11 – Experiência dos programadores disponibilizados

Programador	Anos de experiência em manutenção	Número de sistemas já mantidos
I	3	6
J	2	3
L	3	5
M	2	2
N	3	6

Para cada programador, o NEUROMINER foi executado para analisar o corpo das mensagens de e-mail por eles postadas na lista de discussão de um projeto de software desenvolvido pela empresa. Esta amostra envolve 4.604 mensagens postadas pelos cinco programadores entre 2008 e 2010.

5.2.3 Instrumentação

5.2.3.1 SourceMiner

O *SourceMiner* (Carneiro et al., 2009) é um *plug-in* para o ambiente de desenvolvimento Eclipse que auxilia programadores no processo de visualização e compreensão de software. Este *plug-in* fornece várias formas visuais de inspecionar o código fonte de um software, provendo visualizações em árvore, por dependência de classes e pacotes, filtro para seleção de recursos a serem analisados, entre outras funcionalidades.

A versão utilizada do *SourceMiner* integrava os seguintes paradigmas de visualização:

1. *TreeMap:* representa a estrutura hierárquica dos pacotes, classes e métodos de um projeto de software através de retângulos recursivamente aninhados. Os retângulos mais internos representam os métodos e os mais externos os pacotes. Retângulos maiores e com cores mais escuras representam os métodos maiores e mais complexos do projeto.
2. *Polymetric:* é uma representação bidimensional que usa retângulos arrumados em forma de árvore para representar herança entre entidades do software, tais como classes e interfaces. As dimensões dos retângulos são usadas para representar propriedades das entidades. A largura corresponde ao número de métodos, enquanto a altura corresponde ao número de linhas de código. A cor representa o tipo do elemento, por exemplo, se o mesmo é uma classe abstrata, uma enumeração ou uma interface.
3. *Dependency:* representa dependências entre classes ou pacotes através de grafos radiais. O nó central é a classe de interesse (inicialmente aquela com maior acoplamento). As cores indicam classes que pertencem a um mesmo pacote. Vários tipos de dependências podem ser retratadas pelas arestas do grafo.
4. *GridCoupling*: possui dois objetivos: (1) dar uma visão geral sobre o grau de acoplamento dos módulos do software. Para isto, é exibida uma grade em forma de tabuleiro de xadrez onde as células representam os módulos ordenados por grau de acoplamento com

outros módulos do sistema. (2) Detalhar o grau de acoplamento de um nó selecionado com outros. O acoplamento é representado por um grafo egocêntrico espiral, onde o nó central é o módulo sob análise e os outros nós são posicionados em espiral de acordo com a força da sua dependência do nó central.

Os recursos visuais do SourceMiner se combinam com os recursos visuais já fornecidos pelo ambiente Eclipse, o ambiente de desenvolvimento de software no qual o *plug-in* está integrado. Isto inclui o editor sensitivo a contexto (*Editor*), o explorador de pacotes (Package Explorer), o delineador de entidades (*Outliner*), entre outras visões do Eclipse.

As entidades representadas nas visões do *SouceMiner* podem ser dinamicamente filtradas de acordo com suas propriedades. Para isto, o *plug-in* utiliza duas outras visões: *FilterView* e *ConcernFilterView*. A Figura 22 apresenta uma tela exemplo do *SourceMiner*, na qual é possível ver três das visões do *plug-in*: *Polymetric* (B) e *TreeMap* (C), para visualização; e *FilterView* (D), para filtrar a informação apresentada pelas visões. No exemplo, as visões são organizadas lado a lado, possibilitando o compartilhamento do ambiente com o *Editor* (E) e o *Package Explorer* (A) do Eclipse.

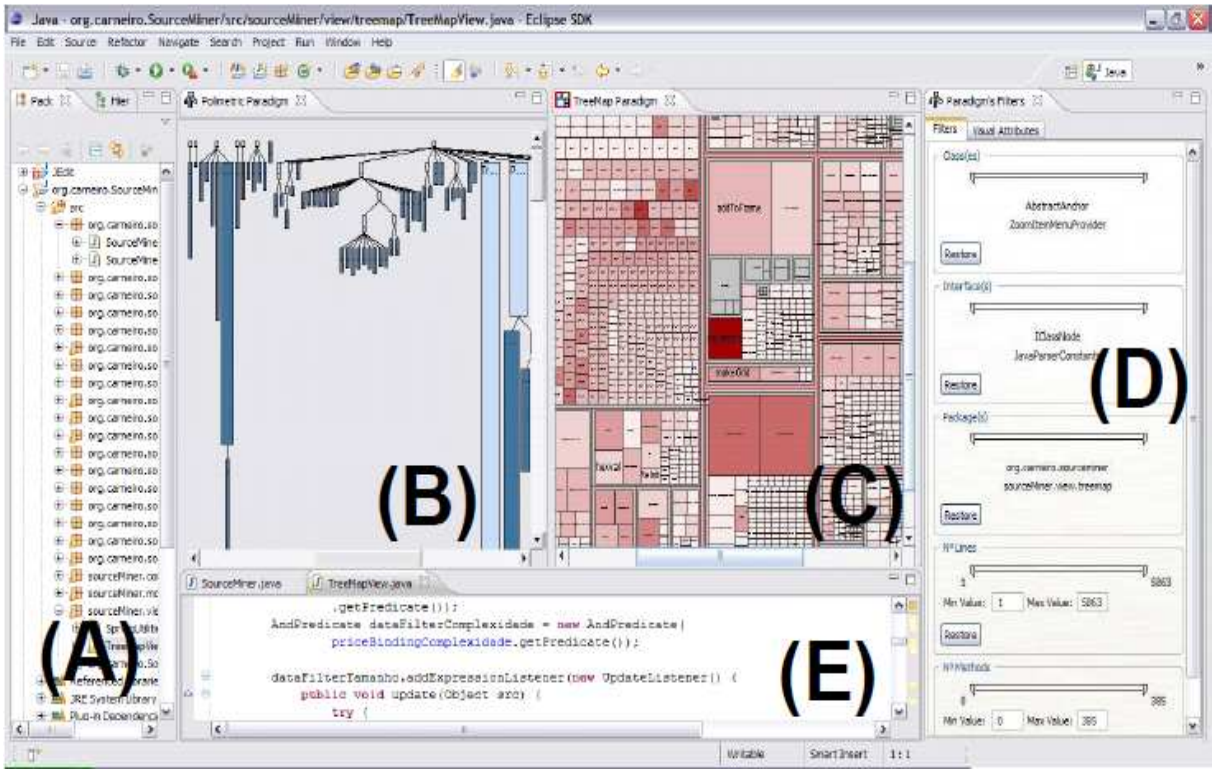

Figura 22. Um tela exemplo do *SourceMiner*

5.2.3.2 Logs

O *SourceMiner* monitora o ADS e provê *logs* contendo as ações realizadas pelos programadores durante o processo de compreensão e manutenção de um software. Ele registra cada ação feita pelo usuário no ADS e as armazena em um arquivo texto com os seguintes atributos: data-hora, a *feature* utilizada (recurso usado, o qual pode ser uma visão ou outra janela do ADS) e a ação que foi aplica a esta *feature*. O arquivo forma um histórico estruturado que registra seqüencialmente todas as ações tomadas pelo usuário no ambiente. Este histórico reflete os passos tomados pelo programador durante atividades de engenharia de software suportadas pelo ADS.

Abaixo, na Figura 23, pode ser visto um exemplo do *log* gerado pelo *SourceMiner* durante uma tarefa no Eclipse.

```
(01/26/2010 04:36:03) Package Explorer DEACTIVATED
(01/26/2010 04:36:10) Filters ACTIVATED
(01/26/2010 04:36:22) ConcernFilterView CLOSED
(01/26/2010 04:37:02) ConcernFilterView OPENED
(01/26/2010 04:37:31) Filters DEACTIVATED
(01/26/2010 04:37:34) ConcernFilterView ACTIVATED
(01/26/2010 04:39:04) ConcernFilterView DEACTIVATED
(01/26/2010 04:39:15) GridCoupling ACTIVATED
(01/26/2010 04:39:55) GridCoupling DEACTIVATED
(01/26/2010 04:40:12) TreeMap ACTIVATED
```

Figura 23. *Log* gerado pelo *SourceMiner*

5.2.3.3 ETL para o SourceMiner

Para que os *logs* gerados pelo *SourceMiner* pudessem ser efetivamente utilizados, seguindo a mesma arquitetura de *Software Data Warehousing* apresentada no capítulo 3, foi criado um novo ETL em C#. Esta ferramenta limpa, transforma e carrega os dados dos *logs* para um *data mart* passível de ser explorado por algoritmos de mineração de dados e por operações OLAP. Este módulo ETL também pode gerar um arquivo para a ferramenta WEKA de mineração de dados (Weka, 2009). Esta última funcionalidade não foi utilizada neste experimento.

O ETL recebe como entrada um arquivo texto com os *logs*, transformando-os para criação e carga do *data mart* em um banco de dados SQL SERVER. A Figura 24 apresenta a interface para configuração básica do processo. Na área A, devem ser introduzidas informações referentes à conexão com o servidor de banco de dados. O banco de dados é criado automaticamente para receber os dados do *log*.

Na área B, os atributos de entrada para o processamento são:

1. *Log file*: caminho do arquivo texto contendo o *log.*

2. *Transaction Time:* refere-se ao tempo que será considerado para uma transação. Como o *SourceMiner* não organiza o *log* em transações, o ETL particiona os dados do *log* em blocos contendo operações que aconteceram a cada X minutos. X é o tempo definido no campo *Transaction Time*. Este parâmetro foi definido baseando-se na clássica abordagem da janela deslizante (Fogel e O'Neill, 2009). Nela, dois eventos subsequentes E_i e E_{i+1} fazem parte de uma transação Δ se os mesmos foram executados em um determinado intervalo de tempo.

3. *Assigning to a programmer*: deve ser informado o nome do programador ao qual o log pertence. Isso permite o acúmulo identificado de dados de vários programadores no banco de dados.

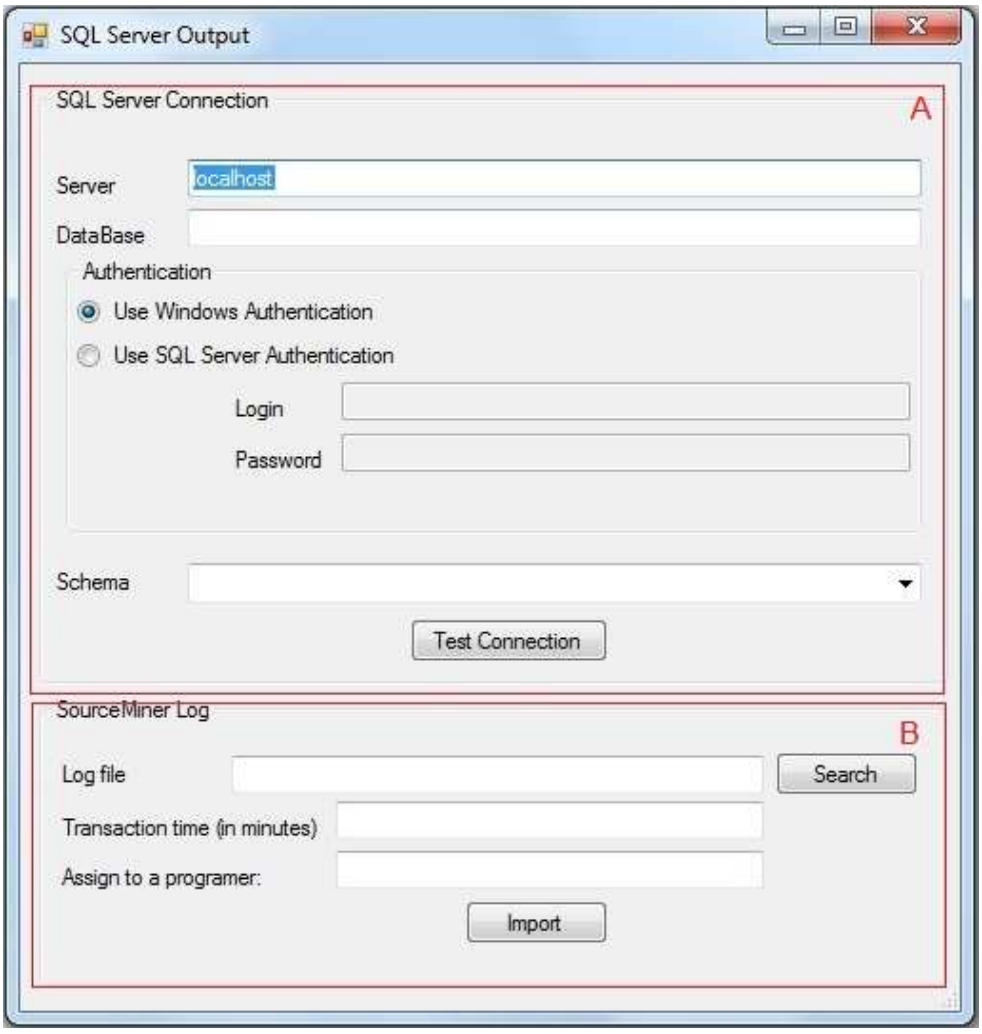

Figura 24. Interface de saída para o SQL SERVER

Para a nossa análise dos dados é necessário selecionar somente as linhas do *log* que têm relação com a utilização das funcionalidades de visualização de software proporcionadas pelo *SourceMiner* e pelo ADS, bem como as linhas que representam acesso ao Editor de código.

Para o *SourceMiner,* são consideradas todas as visões e interfaces para alterações de parâmetros: *Dependency, Polymetric, GridCoupling, TreeMap, FilterView* e *ConcernFilterView*. Para o ADS propriamente dito: *Package Explorer*, *Hierarchy* e *Editor*.

A palavra WORKBENCH aparece ao longo do *log*, toda vez que o programador retira o ADS de foco no *desktop*. Isto ocorre quando o ADS é minimizado e outro programa ou a área de trabalho entra em foco. Contabilizamos isso, para averiguar o tempo possivelmente ocioso do programador.

Como pode ser observado na Figura 23, após o nome da funcionalidade, eventualmente aparecem as palavras ACTIVATED e DEACTIVATED. A presença dessas palavras em uma das linhas do *log* representa, respectivamente, a postura ou retirada de foco de uma funcionalidade, ou seja, o momento exato em que o usuário do ADS começou ou parou de utilizar determinada funcionalidade. Desse modo, através de uma operação de transformação de dados, é possível saber o tempo que foi gasto pelo programador em cada uma dessas ações, bem como o número de acessos às funcionalidades.

Para quebra dos dados em transações, após a entrada do tempo a ser considerado para as mesmas, o algoritmo particiona as linhas em grupos, os quais contêm as ações que aconteceram dentro do intervalo de tempo que foi estabelecido. Para cada grupo, é associado um número inteiro, único e seqüencial (o número da transação).

Por fim, o *data mart* é carregado, disponibilizando as informações especificadas na Tabela 12.

Tabela 12 – Informações disponibilizadas pelo *data mart SourceMiner*

Atributo	Descrição
Feature	Nome da funcionalidade
Seq	Número seqüencial equivalente à transação gerada na transformação dos dados
SpentTime	Tempo gasto na funcionalidade
Programer	Nome do programador
File	Número do arquivo de *log*. Esse campo existe para diferenciar os dados que vêm de diferentes arquivos de *log* associados a uma pessoa
Transaction	Representa a transação que o registro está contido. É formado pela concatenação da coluna *File* com a coluna *Seq*.

A análise previamente descrita cria a possibilidade de detecção de heurísticas de uso do ADS pelo programador (Maletic e Kagdi, 2008)(El-Ramly e Stroulia, 2004), evidenciando os caminhos que são mais comumente percorridos para realização de uma tarefa.

5.2.3.4 Modelo de mineração de dados

Uma vez construído o DW, o próximo passo é analisar os seus dados. Além dos resultados de consultas OLAP básicas, este experimento também analisará as seqüências de uso das funcionalidades pelos programadores. Para isto, foi utilizado o algoritmo *Microsoft Sequence Clustering* (MSDN, 2008), disponibilizado pelo pacote de *Business Intelligence* do SQL Server. Este algoritmo pode ser usado para explorar dados com fatos que podem ser ligados por caminhos ou seqüências.

Esta abordagem localiza as seqüências mais comuns de ações, agrupando (em grupos ou *clusters*) as seqüências que são similares. Os dados devem ser disponibilizados em forma seqüencial. Em outras palavras, devem representar uma série de eventos ou transições de estados, como os que são gerados pelo ETL desenvolvido para o *SourceMiner*.

O algoritmo examina todas as probabilidades de transições, medindo a diferença, ou distância, entre as possíveis seqüências no conjunto de dados. De posse dessas diferenças, determina quais seqüências são as melhores ou mais freqüentes.

5.3 OPERAÇÃO

5.3.1 Execução

Em primeiro lugar, foram replicados, para os programadores selecionados, os experimentos apresentados nos capítulos 3 e 4. Após isso, para sistematização de uso do *SourceMiner* por parte dos programadores e conseqüente geração dos *logs*, foi feita a replicação de um terceiro experimento apresentado em (Carneiro et al., 2010).

Nesse experimento, o objetivo é medir a precisão da identificação de *Code Smells* com a utilização do *SourceMiner* (Carneiro et al., 2010). *Code Smells* são anomalias de modularidade de software geralmente causadas pela maneira que interesses são implantados no código fonte (Fowler, 1999). A sua identificação pode depender das propriedades que regem a estrutura de interesses individuais e suas interdependências na implementação do sistema.

Os participantes foram solicitados a analisar cinco versões de um sistema de código aberto, para identificar o seguinte conjunto de *Code Smells* (Fowler, 1999)(Riel, 1996):

1. *Feature Envy (FE)*: anomalia que ocorre quando um trecho de código parece estar mais interessado em uma classe diferente da que ele reside.
2. *God Class (GC)*: anomalia caracterizada pelo comportamento não coeso e a tendência de uma classe em atrair mais e mais funcionalidades.

3. *Divergent Change (DC)*: anomalia que ocorre quando uma classe muda constantemente de diferentes maneiras, por diferentes razões. Esta classe apresenta interesses mistos e é suscetível a ser alterada por diferentes razões.

4. *Shotgun Surgery (SS)*: anomalia similar ao *Divergent Change*, porém tem comportamento oposto. Ocorre sempre que uma alteração feita em uma classe desencadeia uma série de pequenas alterações em várias outras classes. Quando mudanças estão em vários lugares, as mesmas são difíceis de encontrar e é fácil deixar que uma alteração muito importante passe despercebida.

Como orientação básica, foi dito aos programadores que os mesmos deveriam tentar descobrir os *Code Smells* sem acessar o código, ou seja, utilizando apenas as visões do *SourceMiner* e do ADS (sem utilizar o seu Editor). Em último caso, se não fosse possível identificar apenas com as visões, o programador poderia acessar o código da aplicação.

Após a realização desta parte do estudo, foram coletados os *logs,* os quais foram submetidos ao ETL desenvolvido para o *SourceMiner*. Carregado o *data mart*, foram executadas consultas OLAP de tempo e número de acessos às *features* do ADS.

A fim de evitar a utilização de um tempo arbitrário de transação, após as primeiras consultas OLAP, fizemos uma média geral do tempo gasto por acesso às *features* do *SourceMiner* e do Eclipse. De posse desse tempo, multiplicamos o mesmo pelo número de *features* existentes no ambiente (nove no nosso caso). Nossa intenção foi considerar a possibilidade de uso de todas as *features* por parte de um programador, para resolver uma tarefa. O valor final aproximado foi de 4 minutos, o qual serviu como parâmetro de entrada para um novo processo de ETL (vide Figura 24). Este tempo foi endossado pelos programadores.

Após a determinação do tempo médio de transação, a base foi limpa e o ETL executado novamente. Com uma base de dados subdividida em transações compostas por seqüências de visões usadas, foi criado um modelo de mineração de dados baseado em seqüência (vide seção 5.2.3.4).

Finalmente, cada programador foi individualmente entrevistado sobre o seu perfil de trabalho na empresa, e os dados da execução de todos os experimentos foram integrados, validados, analisados e interpretados. Estes dados são apresentados e discutidos na seção 5.4.

5.3.2 Validação dos dados

Além das validações feitas nos experimentos replicados, bem como no processo ETL, as seguintes validações também foram efetuadas sobre os dados coletados:

1. Foi efetuada sistematicamente a execução de várias seqüências de ações com o *SourceMiner*, objetivando checar a ordem das ações dentro do *log*. Também foi analisada a presença das palavras ACTIVATED e DEACTIVATED durante a mudança de funcionalidade, comprovando assim que estas demarcam exatamente o momento em que ações são iniciadas e terminadas. Desse modo, o tempo total gasto em uma funcionalidade pôde ser capturado.
2. De posse dos dados já transformados, foi possível uma sumarização dos tempos gastos em cada funcionalidade. Esses tempos foram comparados com a diferença entre a hora inicial e final do experimento, registrados individualmente em cada *log*. Houve uma pequena margem de erro, devido ao fato de que nem todas as *features* do Eclipse são consideradas pelo ETL. A preocupação foi assegurar a consistência dos dados processados em relação aos dados brutos que estavam presentes nos *logs*.

5.4 RESULTADOS

Análise com o NEUROMINER

A Tabela 13 resume os resultados obtidos pelo NEUROMINER relativos à hipótese formalizada a partir da primeira questão de pesquisa listada na seção 5.2.1. A coluna *Total* apresenta o número de meses (*data points* para cada participante), de dias e de e-mails explorados. Para cada sistema representacional, o peso final é mostrado com base em todos os predicados sensoriais encontrados, bem como as médias mensais desses pesos. A coluna ANOVA retrata os *p-values* calculados para o teste da hipótese nula.

Tabela 13 – Resultados para os programadores da indústria

Participant	Totals (2008 - 2010)			Visual		Auditive		Kinesthetic		ANOVA p-value
	Months	Days	Emails	Final Weight	Monthly Mean	Final Weight	Monthly Mean	Final Weight	Monthly Mean	
I	30	357	674	0,750382512	0,790360072	0,737687699	0,783063282	0,898610331	0,836974247	0.105
J	20	387	442	0,498330721	0,476327848	0,546430665	0,539162195	0,986144411	0,926387548	.000
L	29	523	1342	0,751891345	0,759316714	0,451976343	0,444731946	0,697414428	0,696729164	.000
M	18	374	1192	0,926541967	0,976824693	0,634382161	0,622630894	0,576707234	0,561374571	.000
N	32	621	954	0,644618737	0,642536173	0,774446612	0,769016124	0,962536713	0,944168916	.000

Para os testes estatísticos, foi estabelecido um nível de significância (α) de 0.05. A Tabela 13 mostra que a hipótese nula é rejeitada para 4 desenvolvedores, os quais obtiveram um *p-value* 0.000. O único desenvolvedor não classificado foi o I, com *p-value* de 0.105, maior que 0.05.

Em resumo, os resultados para os desenvolvedores J, L, M e N são significativamente menores do que 0.05, permitindo a forte rejeição da hipótese nula e confirmação de que os mesmos possuem um SRP (destacado em amarelo).

Análise com o *Survey*

Uma vez realizada a análise de mineração dos e-mails com o NEUROMINER, o próximo passo foi analisar os programadores de acordo com as suas respostas ao *Survey*. Como discutido no capítulo 4, esta análise foi baseada na TRI.

A Tabela 14 mostra um comparativo das classificações geradas pelo NEUROMINER versus as classificações geradas pelo *Survey*. Considerando que as classificações englobam a ordem de preferência, e são feitas através de abordagens completamente diferentes, vemos que a mineração e os questionários apresentam boa consistência entre si.

Tabela 14 – Classificações NEUROMINER X Classificações do *Survey*

Participant	Survey	Neurominer
I	KAV	Não classificado
J	KAV	KAV
L	KVA	VKA
M	VAK	VAK
N	KAV	KAV

A Tabela 15 apresenta os resultados de Precisão, Cobertura e *F-measure*, considerando-se as classificações do *Survey* como gabarito. A Precisão foi calculada para cada Sistema Representacional e sua ordem relativa a cada programador. Desta forma, como temos 5 programadores e 3 sistemas para cada um, para o NEUROMINER ser 100% preciso, por exemplo, ele teria que acertar a classificação de 15 posições (três para cada programador). Considerando o questionário como oráculo, o NEUROMINER alcançou então uma média harmônica (*F-measure*) de Precisão e Cobertura de 74%.

Tabela 15 – Precisão e Cobertura do NEUROMINER em relação ao *Survey*

(a) Posições	15
(b) Posições classificadas	12
(c) Posições cobertas	10
(d) Posições classificadas corretamente	10
Precisão: (d) / (b)	83%
Cobertura: (c) / (a)	67%
F-measure	74%

Análise dos *logs* obtidos com o uso do Sourceminer-Eclipse

O próximo passo foi analisar os dados dos *logs* gerados a partir da replicação do experimento de detecção de *code smells*. Esta análise se iniciou com consultas OLAP feitas ao *data mart* montado. As Tabelas 16, 17, 18, 19 e 20 apresentam os tempos totais de acesso dos programadores às *features* do SourceMiner e do ADS, bem como as médias por acesso dos programadores a estas *features*.

As Figuras 25, 26, 27, 28 e 29 apresentam gráficos de Pareto em relação ao tempo total de acesso a cada *feature* pelo programador. O gráfico de Pareto é baseado na regra 80/20 (Pareto, 1972), sendo utilizado como ferramenta de qualidade que dispõe a informação de forma a tornar evidente e visual a priorização de temas (*features* no nosso caso). Em todos os gráficos, pode-se observar que todos os programadores passaram 80% do tempo em até aproximadamente três das nove *features* analisadas.

Essas Tabelas e gráficos são importantes para evidenciar os programadores que tiveram uma maior dependência de uso do EDITOR, caracterizando uma necessidade cinestésica maior.

Tabela 16 – Tempos e acessos do Programador I

Programador I			
Feature	**Tempo (minutos)**	**Acessos**	**Média**
GridCoupling	24,22	68	0,35618
Dependency	21,98	39	0,56359
PACKAGE EXPLORER	10,82	48	0,22542
TreeMap	7,87	35	0,22486
Polymetric	5,08	32	0,15875
ConcernFilterView	4,12	41	0,10049
EDITOR	3,43	13	0,26385
Filters	2,82	21	0,13429
Hierarchy	0,00	0	0
Total	80,34		

Tabela 17 – Tempos e acessos do Programador J

Programador J			
Feature	**Tempo (minutos)**	**Acessos**	**Média**
Dependency	40,45	52	0,77788
EDITOR	29,93	45	0,66511
Polymetric	19,57	11	1,77909
GridCoupling	12,40	18	0,68889
TreeMap	9,37	25	0,37480
PACKAGE EXPLORER	8,20	43	0,19070
ConcernFilterView	1,00	4	0,25000
Filters	0,05	1	0,05000
Hierarchy	0,00	0	0,00000
Total	120,97		

Tabela 18 – Tempos e acessos do Programador L

Programador L			
Feature	**Tempo (minutos)**	**Acessos**	**Média**
TreeMap	46,03	41	1,12268
GridCoupling	32,67	28	1,16679
Polymetric	25,17	38	0,66237
Dependency	12,55	18	0,69722
PACKAGE EXPLORER	10,85	49	0,22143
ConcernFilterView	2,60	24	0,10833
Filters	1,77	11	0,16091
EDITOR	1,43	3	0,47667
Hierarchy	0,00	0	0,00000
Total	133,07		

Tabela 19 – Tempos e acessos do Programador M

Programador M			
Feature	**Tempo (minutos)**	**Acessos**	**Média**
Dependency	35,00	32	1,09375
PACKAGE EXPLORER	26,38	67	0,39373
Polymetric	25,48	29	0,87862
TreeMap	23,47	20	1,17350
EDITOR	14,95	16	0,93438
GridCoupling	2,18	10	0,21800
Filters	0,90	8	0,11250
ConcernFilterView	0,25	5	0,05000
Hierarchy	0,00	0	0
Total	128,61		

Tabela 20 – Tempos e acessos do Programador N

Programador N			
Feature	**Tempo (minutos)**	**Acessos**	**Média**
EDITOR	30,38	54	0,56259
Dependency	21,55	32	0,67344
GridCoupling	18,60	32	0,58125
TreeMap	15,32	15	1,02133
PACKAGE EXPLORER	5,48	30	0,18267
Polymetric	4,60	12	0,38333
ConcernFilterView	0,13	3	0,04333
Filters	0,03	1	0,03000
Hierarchy	0,10	1	0,10000
Total	96,09		

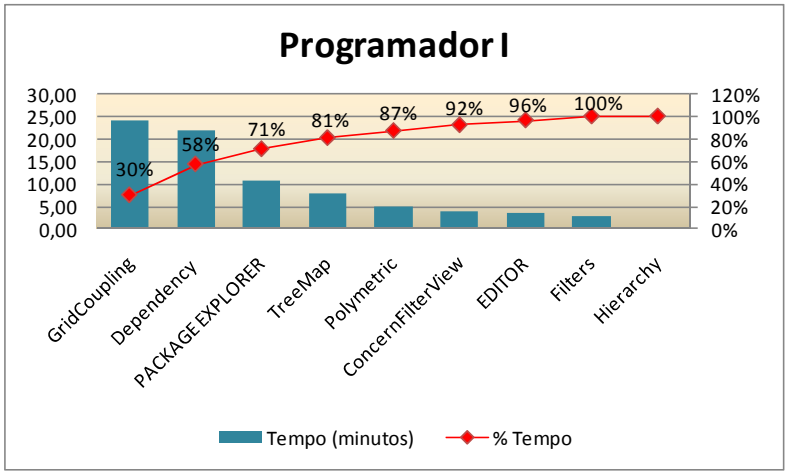

Figura 25. Tempos totais do Programador I

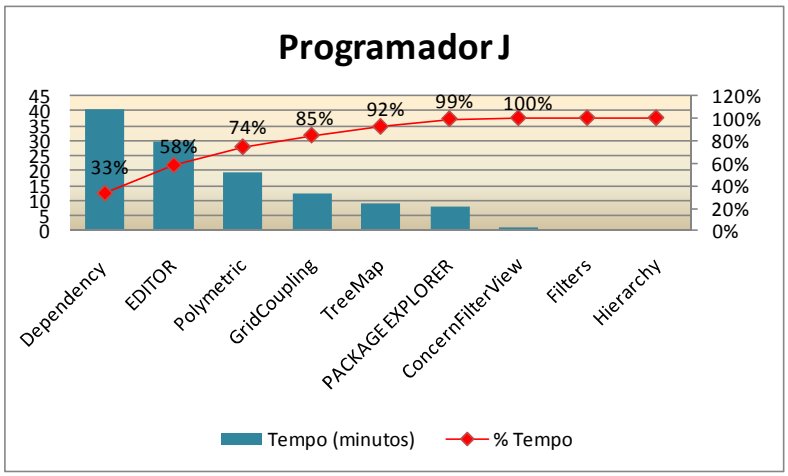

Figura 26. Tempos totais do Programador J

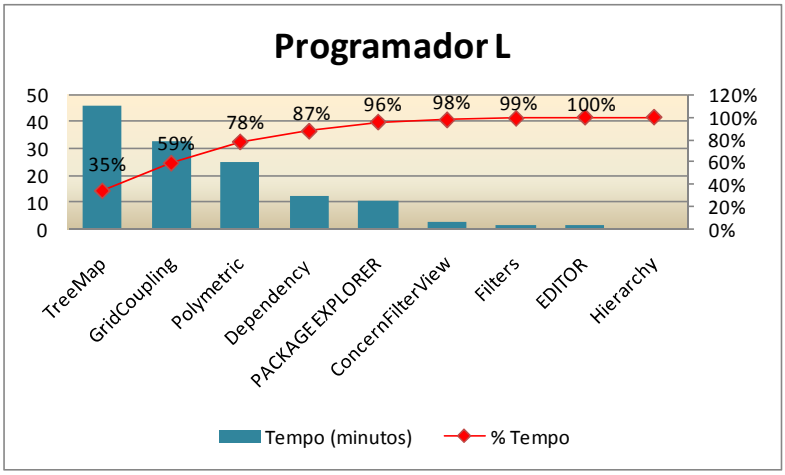

Figura 27. Tempos totais do Programador L

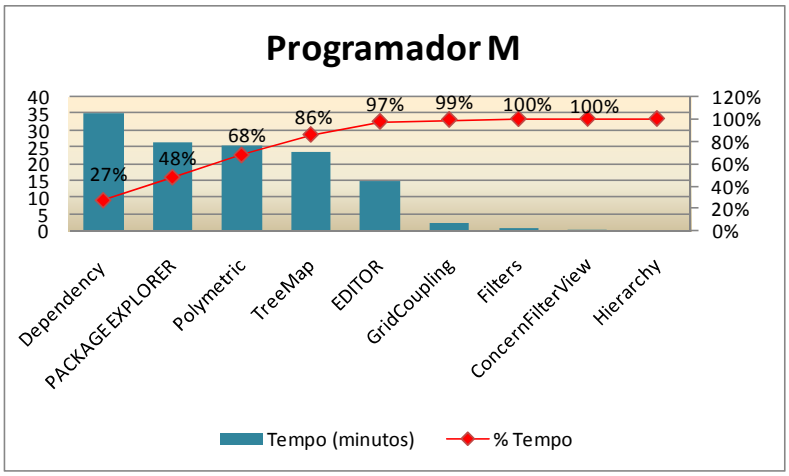

Figura 28. Tempos totais do Programador M

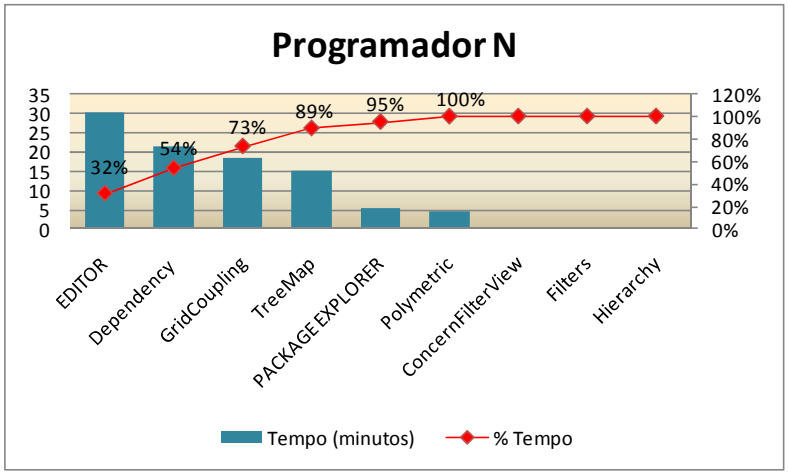

Figura 29. Tempos totais do Programador N

Com relação à aferição do desempenho de cada programador nas tarefas de compreensão, as Tabelas 21, 22, 23, 24 e 25 apresentam as quantidades de *Code Smells* que deveriam ser encontradas e as quantidades que foram efetivamente encontradas pelos programadores. Com estas medidas, foram calculadas as Precisão, Cobertura e F-*measure* do desempenho de cada programador na detecção dos *code smells*. Além disto, para finalidade de comparação, a primeira linha das tabelas também mostra o SRP calculado pelo NEUROMINER para cada programador.

Tabela 21 – Desempenho do Programador I

Programador I - Não classificado					
Medidas	**Code Smells**				
	FE	**GC**	**DC**	**SS**	**Geral**
Existentes	11	9	15	7	42
Acertos	2	4	5	0	11
Erros	3	1	0	5	9
Precisão	40%	80%	100%	0%	55%
Cobertura	18%	44%	33%	0%	26%
F-Measure	25%	57%	50%	0%	35%

Tabela 22 – Desempenho do Programador J

Programador J - KAV					
Medidas	Code Smells				
	FE	GC	DC	SS	Geral
Existentes	11	9	15	7	42
Acertos	1	4	1	0	6
Erros	4	1	5	9	19
Precisão	20%	80%	17%	0%	24%
Cobertura	9%	44%	7%	0%	14%
F-Measure	13%	57%	10%	0%	18%

Tabela 23 – Desempenho do Programador L

Programador L - VKA					
Medidas	Code Smells				
	FE	GC	DC	SS	Geral
Existentes	11	9	15	7	42
Acertos	0	7	7	0	14
Erros	16	15	14	0	45
Precisão	0%	32%	33%	0%	24%
Cobertura	0%	78%	47%	0%	40%
F-Measure	0%	45%	39%	0%	30%

Tabela 24 – Desempenho do Programador M

Programador M - VAK					
Medidas	Code Smells				
	FE	GC	DC	SS	Geral
Existentes	11	9	15	7	42
Acertos	0	3	0	0	3
Erros	5	2	0	5	12
Precisão	0%	60%	0%	0%	20%
Cobertura	0%	33%	0%	0%	11%
F-Measure	0%	43%	0%	0%	14%

Tabela 25 – Desempenho do Programador N

Programador N - KAV					
Medidas	Code Smells				
	FE	GC	DC	SS	Geral
Existentes	11	9	15	7	42
Acertos	2	4	3	0	9
Erros	3	1	2	5	11
Precisão	40%	80%	60%	0%	45%
Cobertura	18%	44%	20%	0%	21%
F-Measure	25%	57%	30%	0%	29%

Finalizando, as Figuras 30, 31, 32, 33 e 34 apresentam os resultados de aglomeração de seqüências minerados dos *logs*. Estas figuras representam respectivamente as seqüências de passos mais utilizadas pelos programadores I, J, L, M e N durante as tarefas de compreensão.

Cada retângulo das figuras representa um estado da seqüência, ou seja, uma *feature* acessada. Os retângulos que possuem sobre eles uma ponta de seta representam as *features* que têm uma probabilidade maior de iniciar uma seqüência. Esta probabilidade pode ser vista no número (cor vermelha) localizado ao lado de cada um deles. As setas que partem de um retângulo para outro representam a transição de estados de uma seqüência; o número ao lado da ponta de cada seta (com a cor preta) representa a porcentagem de vezes que há uma transição entre os estados ligados pela seta. Por exemplo, na Figura 30, 40% das seqüências são iniciadas pela *feature Dependency*. Partindo do estado *Dependency*, em 23% das vezes, há uma transição para o estado *Gridcoupling*, e assim sucessivamente.

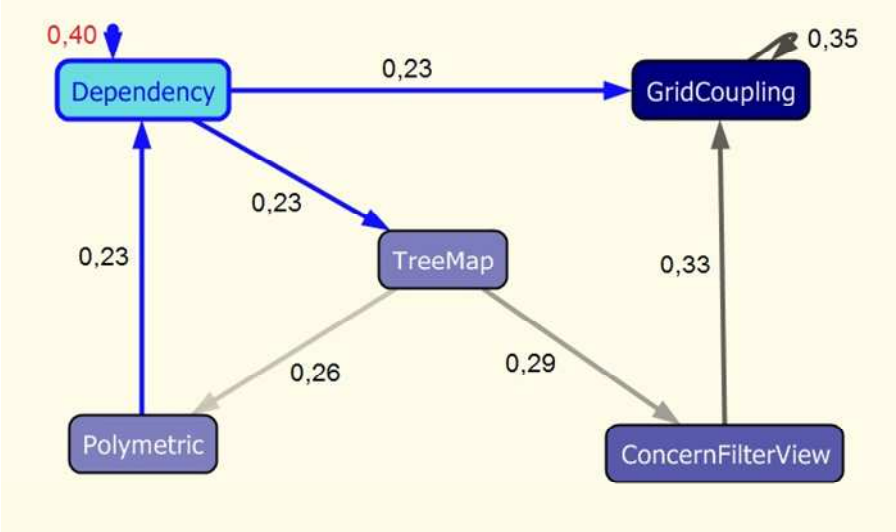

Figura 30. Padrão de Seqüência - Programador I

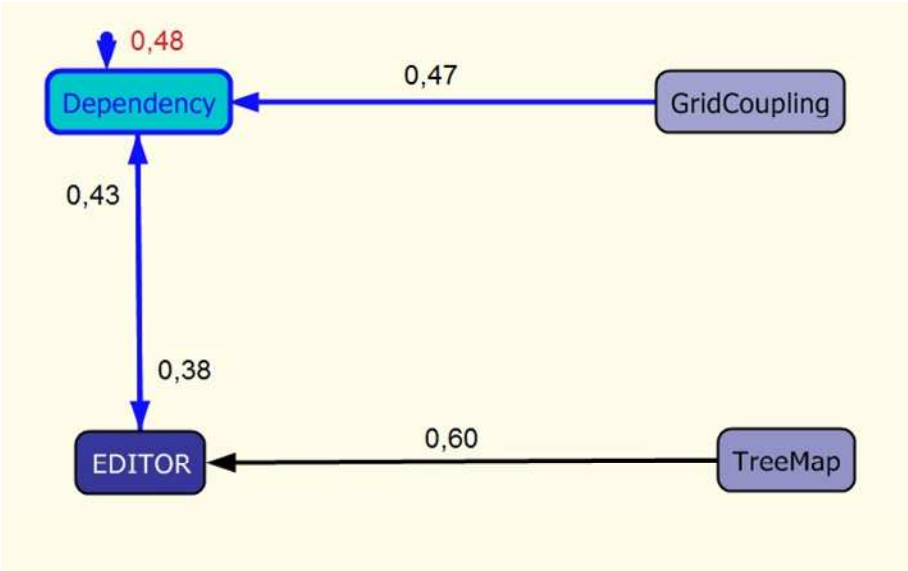

Figura 31. Padrão de Seqüência - Programador J

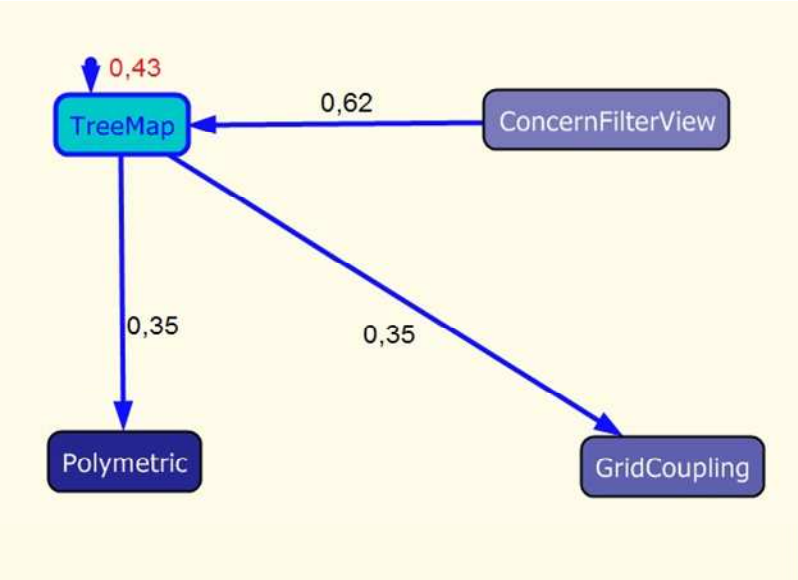

Figura 32. Padrão de Seqüência - Programador L

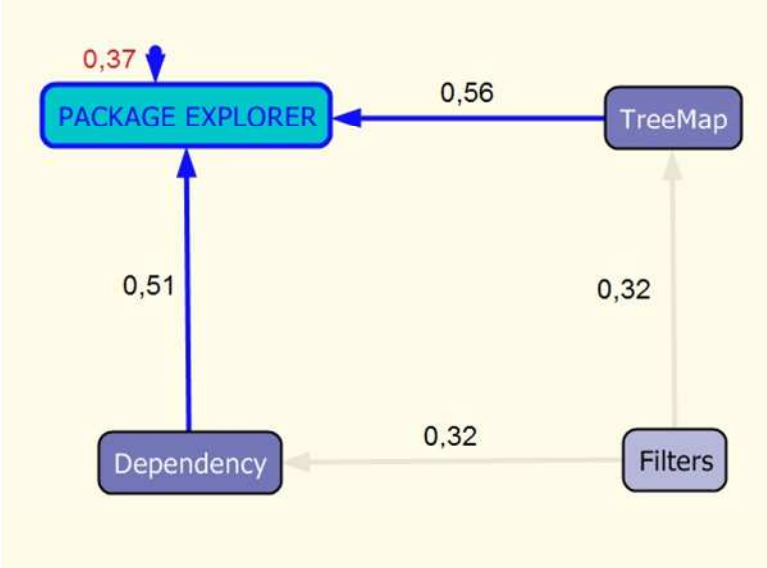

Figura 33. Padrão de Seqüência - Programador M

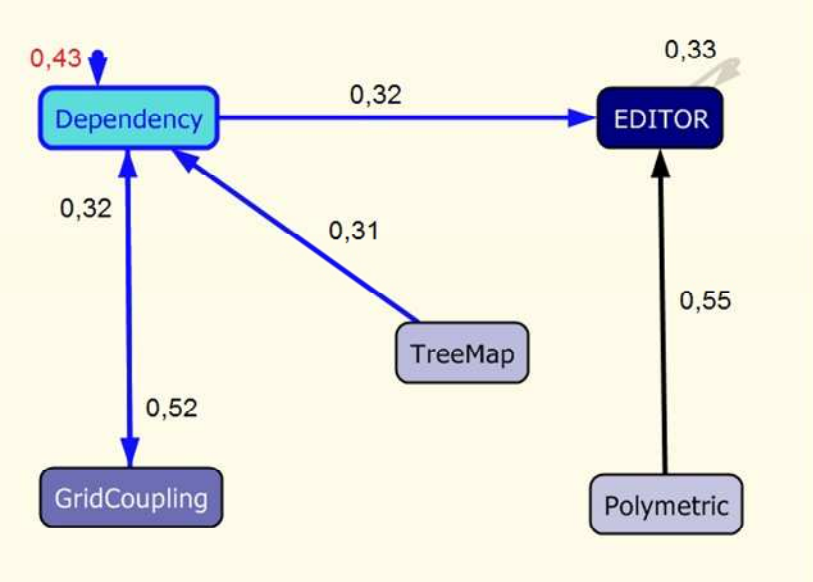

Figura 34. Padrão de Seqüência - Programador N

5.4.1 Análise e interpretação

Como mostrado na Tabela 14, as classificações geradas pelo NEUROMINER e pelo *Survey* apresentam boa consistência entre si.

Os *escores* (calculados com TRI) atribuídos pelo *Survey* aos programadores também foram baseados nas respostas dadas pelos 209 engenheiros de software que participaram do experimento apresentado no capítulo 4.

De acordo com a Tabela 15, considerando-se que o questionário aplicado e validado com a TRI apresenta um indicativo das preferências dos programadores, a média harmônica de Precisão e Cobertura alcançada entre o NEUROMINER e o *Survey* (74%) é um indicativo de que os programadores realmente escrevem sobre as tarefas que fazem e artefatos que usam. Por exemplo, se um programador usa muitos diagramas, ele tende a usar este predicado sensorial (diagrama), bem como outros predicados sensoriais visuais que normalmente são usados juntos, tais como *ver* e *figura*. Isto também já foi evidenciado em (Pattison et al., 2008), o qual encontrou uma relação entre as diversas entidades de um

software que são mencionadas em e-mails e o número de vezes que essas entidades estão incluídas nas tarefas realizadas.

Determinado o alinhamento entre o NEUROMINER e o *Survey*, a questão principal passa a ser então se há consistência entre eles e o comportamento dos programadores nas atividades de detecção de *code smells*.

Para uma análise geral deste ponto de vista, a Tabela 26 resume os resultados encontrados. Considerando a necessidade de uso do EDITOR, os resultados apresentam alguma evidência de que os programadores mais cinestésicos dependem um pouco mais do acesso ao código. Isto pode ser visto nos percentuais de uso do EDITOR para os programadores J e N, respectivamente 25% e 32%, em detrimento aos visuais L e M, com percentuais mais baixos. Em se tratando de uma análise de Pareto, o EDITOR só se apresenta como participante do percentual de 80% de *features* utilizadas, para os programadores mais cinestésicos. Além disso, para estes mesmos programadores (J e N), o EDITOR aparece nas suas seqüências padrão (vide Figuras 31 e 34).

Por outro lado, não encontramos nenhuma evidência de que os visuais tendem a ter um melhor desempenho ao usar uma ferramenta também visual. Isso fica claro ao notarmos que o programador de melhor desempenho não foi classificado como visual, bem como que o segundo colocado foi classificado como KAV. Além disso, o programador de pior desempenho, programador M, foi classificado como visual.

Tabela 26 – Resumo geral dos dados

Programador	Classificação Neurominer	Percentual de uso do EDITOR	F-*measure* alcançada	Número de sistemas já mantidos
I	Não classificado	4%	35%	6
J	KAV	25%	18%	3
L	VKA	1%	30%	5
M	VAK	11%	14%	2
N	KAV	32%	29%	6

A influência maior, neste experimento, parece ter sido a experiência de cada programador, pois como também pode ser visto na Tabela 26, os menos experientes, programadores J e M, obtiveram um menor desempenho de Precisão e Cobertura, perfazendo médias harmônicas (F-*measure*) de 18% e 14%, respectivamente.

Com relação aos perfis, como no experimento realizado em ambiente FOSS, houve um resultado estimulante. Todos os 5 programadores trabalham direta e profundamente com o código, contudo, na sua entrevista, o programador L, classificado como visual, foi o único que declarou fazer uso constante de Diagramas Visuais de Entidade e Relacionamento (Martin e Carma, 1985), criando, revisando e otimizando modelos do sistema. O programador M não relatou nenhuma atividade constante de documentação ou uso de diagramas. Sendo assim, pela sua classificação VAK, recomendamos à empresa parceira testar as suas potencialidades em tarefas que exijam o uso de artefatos visuais, bem como em levantamento e validação de requisitos com entrevistas. Sua segunda preferência, auditiva, pode indicar afinidade em ouvir e captar melhor necessidades dos usuários.

5.4.2 Ameaças à validade

A dificuldade em conseguir cooperação por parte da indústria na liberação dos seus programadores resultou em um tamanho pequeno da amostra. Desta forma, as evidências de uso maior do EDITOR por parte dos cinestésicos pode ter sofrido influência desse viés, possivelmente prejudicando a validade externa dos resultados.

Faz-se necessária a replicação deste experimento com o maior número possível de programadores.

5.5 RESUMO

Este capítulo apresentou um experimento realizado na indústria para tentar encontrar evidências de que programadores visuais têm melhor desempenho ao usar uma ferramenta de visualização. Além disso, o experimento também averiguou, em paralelo, o uso

direto do código por não-visuais e a paridade entre as classificações do NEUROMINER e as classificações do *Survey*.

Após a classificação dos programadores pelo NEUROMINER e pelo *Survey*, os mesmos foram convidados a identificar *Code Smells* com a utilização do *SourceMiner*.

Como orientação básica, foi dito aos programadores que os mesmos deveriam tentar descobrir os *Code Smells* sem acessar o código, ou seja, utilizando apenas as visões do *SourceMiner* e do ADS. Em último caso, se não fosse possível identificar apenas com as visões, o programador poderia acessar o código. Cada programador também foi individualmente entrevistado sobre o seu perfil de trabalho na empresa.

Foi criado um ETL específico para o *Sourceminer*, o qual foi utilizado para validar, integrar e carregar os *logs* gerados. Depois disso, os dados de uso dos programadores foram analisados e interpretados em relação às classificações geradas.

Os resultados apresentaram alguma evidência de que os programadores mais cinestésicos dependem um pouco mais do acesso ao código para realizar tarefas de compreensão. Com relação aos perfis, para 3 programadores, dos 4 classificados, há um alinhamento com as classificações do NEUROMINER, confirmando os resultados que já tinham sido encontrados no experimento apresentado no capítulo 3.

Finalmente, o NEUROMINER alcançou uma média harmônica de Precisão e Cobertura de 74% em relação às classificações do *Survey*. Este é um indicativo de que os programadores realmente escrevem sobre as tarefas que fazem e artefatos que usam. Mais ainda, estes resultados mostram um claro alinhamento entre as duas abordagens. Apesar de diferentes na sua natureza, ambas produzem classificações similares do SRP de engenheiros de software. Isto é uma forte evidência de que os SRP existem e podem de fato ser identificados.

Capítulo 6

Este capítulo apresenta a conclusão desta tese de doutorado, juntamente com as suas contribuições. Também são apresentadas as oportunidades de pesquisas futuras.

6 CONCLUSÃO

O desenvolvimento de software resulta em uma grande quantidade de dados: as alterações feitas no código-fonte são registradas nas ferramentas de controle de versão, os erros são reportados para sistemas de rastreamento e comunicações são arquivadas em e-mails. Como todos esses dados são úteis, a área de Mineração de Repositórios de Software tenta utilizar técnicas de análise de dados para prover um melhor entendimento e apoio ao desenvolvimento de sistemas. Considerando essa referida área, na sua introdução, esta tese se propôs a atacar as seguintes questões de pesquisa:

1. É viável manter um repositório central para mineração de dados da Engenharia de Software?
2. Listas de discussão permitem uma classificação Neuro-Lingüística?
3. Existem evidências de que a classificação Neuro-Lingüística identifica os sistemas representacionais preferenciais de um desenvolvedor de software?
4. Quais os sistemas representacionais usados pelos desenvolvedores mais importantes de projetos FOSS? Quais os dos menos importantes?
5. Desenvolvedores FOSS que mais alteram código são menos dependentes dos sistemas de representação visual e auditivo?
6. Desenvolvedores mais cinestésicos da indústria estudada dependem mais do acesso ao código para realizar tarefas de compreensão?

7. Programadores visuais da indústria estudada alcançam um desempenho melhor ao usar uma ferramenta de visualização?

6.1 RESULTADOS E CONTRIBUIÇÕES

No contexto das questões de pesquisa postas, esta tese fez as seguintes contribuições:

1. ***Software Data Warehousing***. Os estudos realizados propuseram uma arquitetura para sistemas de análise de dados de engenharia de software. O trabalho mostrou a viabilidade da construção e manutenção de um repositório central para mineração de dados de Engenharia de Software (primeira questão da pesquisa). Inicialmente, foram desenvolvidas duas ferramentas para camada de análise: (a) um minerador do histórico de alterações e (b) um *Dash Board* sobre manutenções corretivas (Capítulo 3). O diferencial destes trabalhos foi o armazenamento de dados da fase de construção de software em um DW. A maioria dos repositórios de engenharia de software da época não era estruturada como um armazém de dados, e os únicos DWs identificados não contemplavam o armazenamento de dados da fase de construção do software.
2. **Processo experimental.** Nós adaptamos o processo experimental para engenharia de software descrito por Wohlin et al. (Wohlin et al., 2000) para experimentos em mineração de repositórios de software (Capítulos 3 e 5). Acreditamos que os estudos, aplicações e ferramentas para mineração de repositórios de software podem se beneficiar deste tipo de abordagem, pois uma descrição experimental rigorosa facilita a replicação e a execução de revisões sistemáticas, bem como outros tipos de análises secundárias na área (Dyba et al., 2005).
3. **Automatização de recursos de classificação Neuro-Lingüística e combinação destes com uma taxonomia formal de Engenharia de Software (ES).** Antes do NEUROMINER, não existia nenhuma ferramenta que identificasse Sistemas Representacionais em Textos. Além disto, as pesquisas que identificamos envolvendo

mineração de textos da Engenharia de Software sempre utilizavam uma ferramenta oriunda única e exclusivamente da área de psicologia, ou uma ferramenta para análise de um assunto específico da Engenharia de Software. Esta tese combinou e contextualizou as duas áreas (Capítulos 2 e 3).

4. **Adaptação do processo tradicional de mineração de texto para classificação Neuro-Lingüística**. O NEUROMINER considera todos os e-mails escritos pelo programador como um grande texto a ser classificado. Esta nova abordagem implicou na necessidade de inovação nos cálculos dos escores dos termos encontrados nos textos (Capítulos 2 e 3).

5. **Criação e aplicação de questionário neuro-lingüístico voltado para ES e validado pela TRI.** Como especulado por nossa hipótese, os resultados da pesquisa de campo realizada indicaram que os engenheiros de software têm SRP diferentes. Mais importante que isso, os resultados também mostraram que os mesmos geralmente usam os três sistemas de representação, sendo todos eles muito importantes para os engenheiros de software (Capítulo 4). Neste estudo, obtivemos os seguintes resultados:

   5.1. A alta pontuação do sistema auditivo é um importante indício de que a área deve investir mais em ferramentas multimídia para a compreensão do software. Nós consideramos esta uma importante descoberta deste trabalho.

   5.2. O uso de TRI mostrou que a técnica tem muito a oferecer para a nossa área. A mesma pode ajudar aos pesquisadores de ES na medição de traços latentes em populações heterogêneas, facilitando o processo de meta-análise sobre organizações e culturas diferentes. Acreditamos que podemos usar com sucesso as medidas do poder de discriminação e dificuldade dos itens (questões) para melhorar outros questionários como o nosso. Neste momento, estamos analisando uma forma de melhorar as questões que tiveram baixo poder de discriminação no nosso questionário.

   5.3. O uso de questionários e de mineração de textos é um caminho promissor para a compreensão de traços latentes de engenheiros de software, baseada em técnicas de psicometria. Acreditamos que o uso de questionários com TRI tem muitas outras aplicações para a área.

   5.4. Quando comparamos o questionário com as classificações do NEUROMINER (Capítulo 5), o NEUROMINER alcançou uma média harmônica de Precisão e

Cobertura de 74% em relação às classificações do mesmo. Este é um indicativo de que os programadores realmente escrevem sobre as tarefas que fazem e artefatos que usam. Mais ainda, estes resultados mostram um claro alinhamento entre as duas abordagens. Apesar de diferentes na sua natureza, ambas produzem classificações similares do SRP de engenheiros de software. Isto é uma forte evidência de que os SRP existem e podem de fato ser identificados (segunda questão da pesquisa).

6. **Geração de evidências experimentais sobre Neuro-Lingüística na área de Engenharia de Software (terceira questão da pesquisa).** Os resultados encontrados nos experimentos FOSS e Industrial foram estimulantes (Capítulos 3 e 5).

6.1. Apesar de serem contrários às nossas expectativas, os escores SRP claramente diferenciaram os *top-committers* da população em geral dos projetos FOSS (quarta questão da pesquisa).

6.2. Nos dois ambientes (FOSS e industrial), os perfis neuro-lingüísticos dos participantes alinharam-se com os seus perfis de trabalho.

6.3. Não houve evidências de que os *top committers* FOSS são menos dependentes de artefatos visuais e auditivos (quinta questão da pesquisa). Foram classificados os 3 tipos de SRP para os mesmos (Capítulo 3).

6.4. No experimento na indústria, os resultados apresentaram alguma evidência de que os programadores mais cinestésicos dependem um pouco mais do acesso ao código para realizar tarefas de compreensão (sexta questão da pesquisa).

6.5. Não houve evidências de que programadores visuais alcançam um desempenho melhor ao usar uma ferramenta de visualização (sétima questão da pesquisa - Capítulo 5).

6.2 DISCUSSÃO DOS RESULTADOS OBTIDOS

Os estudos da teoria da Neuro-Lingüística deixaram claro que esta não lida com novos conceitos científicos. Na verdade, ela busca a junção e modelagem de outras teorias psicológicas, como, por exemplo, a *Gestalt*. Freqüentemente associada a estratégias de vendas e marketing, a PNL constantemente utiliza termos como "comportamento" e "re-programação", dando a impressão de que é possível entender e mudar o perfil de pessoas, de uma forma mecanicista. Contrariamente, o entendimento amplo das teorias psicológicas envolvidas nas diferentes maneiras com as quais indivíduos aprendem, leva a uma visão cibernética e sistêmica (Bertalanffy, 1969), ou seja, quaisquer respostas, experiências e comportamentos só são significativos dentro do contexto no qual acontecem.

A psicologia cognitivo-comportamental, baseada em uma filosofia construtivista sobre a realidade e uma filosofia fenomenológica da natureza humana, parte do pressuposto de que há uma diferença entre os nossos mapas mentais sobre o mundo e o mundo propriamente dito. A expressão o "mapa não é o território", introduzida por Korzybski (1933), foi uma das primeiras premissas que evidenciaram a realidade como algo diferente da experiência que um indivíduo tem dessa realidade.

Desta forma, as pesquisas psicológicas têm analisado como a comunicação evidencia a realidade (nossas experiências) e como a língua pode afetar o sistema corpo-mente. Em outras palavras, a expressão corporal, a velocidade e tom utilizados para falar, bem como o que é propriamente dito pelo indivíduo, em um determinado contexto, pode ajudar a identificar sua realidade.

A partir destes princípios, a área de Engenharia de Software pode obter muitos ganhos ao compreender que não é a "realidade" que limita as pessoas, mas as escolhas disponíveis percebidas através da interpretação desta realidade. Desenvolvedores precisam ser estimulados de diferentes modos na atribuição de tarefas, no alinhamento de diretrizes e na compreensão e manutenção do software. A comunicação deve ser eficiente entre gerente e colaboradores, bem como entre artefatos e programadores. Indivíduos com tendências visuais devem ser mais estimulados graficamente, bem como podem ser especificamente alocados em tarefas de documentação.

Neste contexto, esta tese apresentou evidências de que há realmente uma relação entre o que é dito pelos programadores e parte de seus mapas de mundo. Nossa abordagem para classificação Neurolinguística, através da mineração de textos, apresentou consistência com os papéis que as pessoas desempenham em projetos FOSS e com as opiniões que as pessoas expressaram no questionário desenvolvido. Isto indica que a classificação Neuro-Lingüística referente aos SRPs pode ajudar a traçar o perfil de pessoas para tarefas de engenharia de software e, possivelmente, melhorar a comunicação em organizações de desenvolvimento e manutenção de software. O trabalho descrito aqui não vai além. Ele não mostra, por exemplo, como utilizar estes perfis ou como melhorar a comunicação nestas organizações. Estes são temas para pesquisas futuras.

6.3 TRABALHOS FUTUROS

Esta tese deu os primeiros passos em um caminho promissor para compreensão de características neuro-lingüísticas de engenheiros de software, baseada em técnicas de psicometria. O conjunto de estudos aqui desenvolvidos aponta para os seguintes trabalhos futuros:

1. Definição de estratégias de uso de classificações do SRP para melhoria de comunicação em organização de desenvolvimento de software.
2. Uso da arquitetura de *Data Warehouse* para testar outras técnicas de mineração sobre os dados já coletados e pré-processados.
3. Exame da empatia de mensagens trocadas por desenvolvedores, avaliando se há uma relação entre a eficiência da comunicação e o alinhamento do SRP no diálogo.
4. Concepção de novas maneiras de se medir o SRP.
5. Aprimoramento da interface gráfica do NEUROMINER, melhorando usabilidade e disponibilidade do mesmo para testes por qualquer interessado que utilize a internet.

6. Aprimoramento do módulo OLAP, disponibilizando operações *drill-through*. Em particular, deve ser possível ir do termo mais pontuado às mensagens escritas pelo programador.

7. Análise temporal da evolução dos desenvolvedores, cruzando mudanças de SRP com suas atividades no projeto, já armazenadas no *Data Warehouse*.

8. Mineração de padrões de uso gerais de bons programadores, averiguando a seqüência de passos, independente do *SourceMiner*.

9. Expansão do NEUROMINER para atuação em diversas áreas do conhecimento, elevando expressivamente o potencial de mineração e análise da ferramenta. Isto será possível através da criação de técnicas para utilização e combinação de diversas taxonomias e ontologias (outras áreas tais como marketing e educação) e termos do dicionário neuro-linguístico, aumentando assim o poder contextual das análises.

10. Utilização de outras teorias da psicologia a fim de incrementar o NEUROMINER para também detectar crenças e valores limitantes de indivíduos. Detectar essas crenças pode nortear a política de motivação de um projeto, bem como melhorar o processo de seleção de colaboradores.

11. Aplicação e avaliação do NEUROMINER em organizações intensivas em comunicação, tais como *Call Centers*.

12. Criação e validação de módulos NEUROMINER para ferramentas como MSN, ORKUT e TWITTER.

13. Criação e validação de módulo NEUROMINER para ferramentas de educação à distância.

REFERÊNCIAS

ACHARYA, MITHUN, XIE, TAO, PEI, JIAN & XU, JUN. Mining API Patterns as Partial Orders from Source Code: From Usage Scenarios to Specifications. In Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2007), Dubrovnik, Croatia, pp. 25-34, 2007.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. In Proceedings of the 20th Very Large Data Bases Conference, p. 487–499, 1994.

ALLPORT, G. W. Personality: A psychlogical interpretation. NewYork: Holt and Co, 1937.

APACHE CONTRIBUTORS. http://httpd.apache.org/contributors/, accessed in January, 2010.

APACHE. www.apache.org, accessed in January, 2010.

BADENOCH, S. N. Personality type, learning style preference, and strategies for delivering training to a select group of store managers. Doctoral dissertation, University of Minnesota, 1986.

BAILENSON & YEE. Digital chameleons: automatic assimilation of nonverbal gestures in immersive virtual environments. Psychological Science. v16. 814-819, 2005.

BAKER, FRANK B. The Basics of Item Response Theory. ERIC Clearinghouse on Assessment and Evaluation, 2001.

BANDLER, R. & GRINDER, J. Frogs into Princes: Neuro-linguistic Programming, Moab, Utah: Real People Press, 1979.

BANDLER, R. & GRINDER, J. The Structure of Magic I: A Book about Language and Therapy, Palo Alto, California: Science and Behaviour Books, 1975.

BASILI, R. M. LINDVALL, P. COSTA. Implementing the Experience Factory concepts as a set of Experience Bases, 13th Int. Conf. on Software Engineering & Knowledge Engineering, 2001.

BASILI, V. & WEISS, D. “A Methodology for Collecting Valid Software Engineering Data,” IEEE Transactions on Software Engineering, vol.10(3): 728-738, November 1984.

BECKER, K. et al. “SPDW: a Software Development Process Performance Data Warehousing Environment”. In: Annual NASA Software Engineering Workshop, 30th, 2006.

BERTALANFFY, L. von. General Theory of Systems. N. York, George Braziller, 1969.

BIRD, C., GOURLEY, A., P. DEVANBU, M. GERTZ, & A. SWAMINATHAN. Mining e-mail social networks. Proceedings of the 2006 international workshop on Mining software repositories, pages 137–143, 2006.

BOGAROSH, NICHOLE. Collaborative Communication: An Integral Part Of Servant-Leadership. Organizational Communication, 2008.

BOSTIC ST.CLAIR, C. & GRINDER, J. Whispering in the Wind J & C Enterprises, Scotts Valley, CA, 2001.

CALERO, C., RUIZ, F., & PIATTINI, M. Ontologies for software engineering and software technology. Springer-Verlag, Heidelberg, 2006.

CARNEIRO, G DE F., MAGNAVITA, R., MENDONÇA, M. Proposing a Visual Approach to Support the Characterization of Software Comprehension Activities. In 17th IEEE International Conference on Program Comprehension, USA, Miami University, 2009.

CARNEIRO, GLAUCO DE F.; SILVA, M., MARA, L., FIGUEIREDO, E., SANT´ANNA, C. N.; GARCIA, A. F.; MENDONÇA, M. G. Identifying Code Smells with Multiple Concern Views. Proceedings of the CBSOFT-SBES, 2010.

CASSIDY, S. Learning styles: an overview of theories, models and measures. Educational Psychology. Vol 24, No 4, pp. 419-444, 2004.

CLEMENTS, PAUL. et al. Documenting Software Architectures: Views and Beyond. Addison-Wesley Professional; 1st edition, 2002.

COLAÇO JR. , METHANIAS, MENDONÇA, M. G. & RODRIGUES, F. Data Warehousing in an Industrial Software Development Environment. In: 33rd Annual IEEE/NASA Software Engineering Workshop, 2009.

COLAÇO JR., METHANIAS, MENDONÇA NETO, M. G., FARIAS, M. A. & HENRIQUE, PAULO. OSS Developers Context-Specific Preferred Representational Systems: A Initial Neurolinguistic Text Analysis of the Apache Mailing List. In: 7th IEEE Working Conference on Mining Software Repositories, Cape Town, 2010.

COLAÇO JR., METHANIAS. Implementing Decision Support Systems and Data Warehouses. Rio de Janeiro, Brazil: Axel Books, 2004.

COLAÇO JR., METHANIAS; MENDONÇA, M. G.; RODRIGUES, F. Mining Software Change History in an Industrial Environment. In: XXIII Brazilian Symposium on Software Engineering, Fortaleza, Brazil, 2009B.

CROWSTON, K. & HOWISON. J. The Social Structure of Free and Open Source Software Development. First Monday, 10(2), 2005.

CUNNINGHAM, H., MAYNARD, D., BONTCHEVA, K., & TABLAN, V. GATE: a framework and graphical development environment for robust NLP tools and applications. In Proceedings of the 40th Meeting ACL, 2002.

DENT, K.A. Cognitive Styles: Essence and Origins: Herman A. Witkin and Donald R. Goodenough. Journal of the American Academy of Psychoanalysis, 11:635-636, 1983.

DIEHL, S. Software Visualization -- Visualizing the Structure, Behavior and Evolution of Software. New York: Springer Verlag, 2007.

DILTS, R., GRINDER, J., BANDLER, R. & DELOZIER, J. Neuro-linguistic Programming: Volume 1, the study of the structure of subjective experience. California: Meta Publications, 1980.

DYBA, T., KITCHENHAM, B. & JORGENSEN, M. Evidence-based software engineering for practitioners. Software, IEEE, 22(1):58–65, 2005.

EINSPRUCH, ERIC L. & FORMAN, BRUCE D. Observations concerning Research Literature on Neuro-Linguistic Programming. Journal of Counseling Psychology, vol. 32, no. 4, 589-596, 1985.

ELICH, M., THOMPSON, R. W., & MILLER, L. Mental imagery as revealed by eye movements and spoken predicates: A test of neurolinguistic programming. Journal of Counseling Psychology, 32(4), 622-625, 1985.

EL-RAMLY, M. & STROULIA, E. Mining Software Usage Data. Proceedings 1st International Workshop on Mining Software Repositories, 2004.

ESTES, W. K. Learning theory and intelligence. American Psychologist, 29, 740-749, 1974.

FAYYAD, USAMA; PIATETSKI-SHAPIRO, GREGORY; SMYTH, PADHRAIC. The KDD Process for Extracting Useful Knowledge from Volumes of Data. In: Communications of the ACM, pp.27-34, 1996.

FINHOLT, T. & SPROULL, L. S. Electronic groups at work. Organization Science (1,1), 41-64, 1990.

FLEMING, N. D. I'm different; not dumb. Modes of presentation (VARK) in the tertiary classroom. Proceedings of the Annual Conference of the Higher Education and Research Development Society of Australasia, Vol 18, pp. 308 – 313, 1995.

FLEMING, N.D. & MILLS, C. Not Another Inventory, Rather a Catalyst for Reflection. To Improve the Academy, Vol 11, pp. 137-155, 1992.

FLEMING, N.D. VARK. Available in http://www.vark-learn.com/english/page.asp?p=questionnaire, 2008, last access in Jun/2009.

FOGEL, K. AND O'NEILL, M. cvs2cl.pl: CVS-log-message-to-ChangeLog conversion script. Available at: http://www.redbean.com/cvs2cl/, accessed in January, 2009.

FOWLER, M. Improving the Design of Existing Code. Addison-Wesley Professional, 1999.

GARITY, J. Learning styles: Basis for creative teaching and learning. Nurse Educator, Vol 10, Issue 2, 12-16, 1985.

GRINDER, J. & BANDLER, R. The Structure of Magic 2: a book about communication and change Science and Behaviour Books, Palo alto, 1976.

GROSSMAN, MICHAEL & KATZ, ROBERT. A new approach to means of two positive numbers. International Journal of Mathematical Education in Science and Technology, Volume 17, Number 2, pages 205 – 208, 1986.

HARTLEY, J. Learning and studying: A research perspective. London: Routledge, 1998.

HAYTHORNTHWAITE, C. & GRUZD, A. A Noun Phrase Analysis Tool for Mining Online Community Conversations. In: Proc. of the 3rd International Conference Communities and Technologies, 2007.

HEAP, M. "Neurolinguistic programming - an interim verdict," in Hypnosis: current clinical, experimental and forensic practices, M. Heap, ed., Croom Helm, London, pp. 268-280, 1988.

HIEMSTRA, D. E JONG, F. Statistical Language Models and Information Retrieval: Natural language processing really meets retrieval, Glot International 5(8), Blackwell Publishers, 2001.

HILL, M. M. & HILL, A. B. A construção de um questionário. Dinâmia, Lisboa, 1998.

HOWARD, A. Software engineering project management. Communications of the ACM, 44(5), 2001.

HUNT, E.B., FROST, N. & LUNNEBORG, C. Individual differences in cognition: A new approach to intelligence. In G. Bower (Ed.), The psychology of learning and motivation (Vol. 7, pp. 87-122). New York: Academic Press, 1973.

KARIM, JAHANVASH. An item response theory analysis of Wong and Law emotional intelligence scale. Procedia - Social and Behavioral Sciences, Volume 2, Issue 2, Pages 4038-4047, 2010.

KIMBALL, R. et al. The Data Warehouse Lifecycle Toolkit. New York: John Wiley & Sons, 1998.

KLEMOLA, T. & RILLING, J. Modeling Comprehension Processes in Software Development. Proceedings: First IEEE International Conference on Cognitive Informatics, 329-336, Aug 2002.

KOCHE, JOSÉ CARLOS. Fundamentos de Metodologia Científica. Porto Alegre: Vozes, 1982.

KORZYBSKI, A. Science and Sanity: An introduction to non-Aristotelian systems and general semantics, Englewood, New Jersey: The International Non-Aristotelian Library Publishing Company. Available online at http://www.esgs.org/uk/art/sands.htm, accessed 14.01.2009.

LAVE, J. & WENGER, E. Situated Learning: Legitimate peripheral participation, Cambridge University Press, 1991.

LAYMAN, LUCAS, NACHIAPPAN, NAGAPPAN, GUCKENHEIMER, SAM, JEFF BEEHLER, ANDREW BEGEL. Mining software effort data: preliminary analysis of visual studio team system data. In Proceedings of the International Working Conference on Mining Software Repositories, pages 43-46, 2008.

LEITE, W. L. & SVINICKI, M. Attempted validation of the scores of the VARK learning styles inventory with Multitrait-Multimethod Confirmatory Factor Analysis Models. In Educational and Psychological Measurement, 2010.

LEWIS, LAURIE K. Collaborative Interaction: Review of Communication Scholarship and Research Agenda. Communication Yearbook 30, pp. 197–247, 2006.

LI, Z., TAN, L., WANG, X., LU, S., ZHOU, Y. & ZHAI, C. Have things changed now? an empirical study of bug characteristics in modern open source software. In proceedings of ASID 2006, pages 1–9. ACM, 2006.

LOHMAN, DAVID F. & BOSMA, A. Using Cognitive Measurement Models in the Assessment of Cognitive Styles. The role of constructs in psychological and educational measurement (pp. 127-146). Hillsdale, NJ: Erlbaum, 2002.

LOHMAN, DAVID F. & IPPEL, M. J. Cognitive diagnosis: From statistically based assessment toward theory-based assessment. Test theory for a new generation of tests (pp. 41-71). Hillsdale, NJ: Lawrence Erlbaum, 1993.

MALDONADO, JOSÉ CARLOS, CARVER, JEFFREY, SHULL, FORREST, FABBRI, SANDRA CAMARGO PINTO FERRAZ, EMERSON DÓRIA, LUCIANA MARTIMIANO, MANOEL G. MENDONÇA, VICTOR R. BASILI: Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness. Empirical Software Engineering, 11(1): 119-142, 2006.

MALETIC, J. I. & KAGDI, H. Expressiveness and Effectiveness of Program Comprehension: Thoughts on Future Research Directions. In: Proceedings of the Frontiers of Software Maintenance, CHI, IEEE, p. 31-37, 2008.

MANNING, C. D., RAGHAVAN, P., & SCHUTZE, H. Introduction to Information Retrieval. Cambridge University Press, 2007.

MARON, M. E.; KUHNS, J. L. On relevance, probabilistic indexing, and information retrieval. Journal of the ACM, 7(3):216-244, 1960.

MARTIN, JAMES, & CARMA MCCLURE. Diagramming techniques for Analysts and Programmers. Englewood Cliffs, NJ: Prentice Hall, 1985.

MATTHEWS, D. B. Learning Styles Research: Implications for Increasing Students in Teacher Education Programs. Journal of Instructional Psychology, 18 (December) pp. 228-236, 1991.

MAZIERO, E.G. et al. A Base de Dados Lexical e a Interface Web do TeP 2.0 -Thesaurus Eletrônico para o Português do Brasil. VI Workshop em Tecnologia da Informação e da Linguagem Humana, pp. 390-392, 2008.

MENDONÇA, M. G. & SUNDERHAFT, NANCY L. Mining Software Engineering Data: A Survey. A DACS State-of-the-Art Report Available at: http://www.dacs.dtic.mil/techs/datamining/datamining.pdf., 1999.

MOCKUS, A. & VOTTA, L.. Identifying reasons for software changes using historic databases. International Conference on Software Maintenance, pages 120–130, 2000.

MOCKUS, A., R. FIELDING, T. & J. HERBSLEB. Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11(3):1–38, July 2002.

MOODY, L. D. The “Physics” of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering, v. 35, 2009.

MSDN. Disponivel em: http://msdn.microsoft.com/en-us/library/ms175462.aspx, 2008. Acesso em outubro de 2010.

NEUROMINER SURVEY. www.neurominer.com/survey, accessed in October, 2010.

PAICE, C. P. Soft evaluation of boolean search queries in information retrieval systems. Information Technology: Research and Development 3 (1), 33–42. 1984.

PALZA, E., C. FUHRMAN, A. ABRAN. “Establishing a Generic and Multidimensional Measurement Repository in CMMI context”. 28th Annual NASA Soft. Eng. Workshop (SEW’03), pp.12-20, 2003.

PANG-NING, TAN et al. Introduction to Data Mining. Addison-Wesley, 2006.

PAOLO, A. G. SIVILOTTI, SCOTT, M. PIKE. A collection of kinesthetic learning activities for a course on distributed computing. ACM SIGACT news distributed computing column 26. SIGACT News 38(2): 56-74, 2007.

PARETO, VILFREDO. Manual of Political Economy. Macmillan, 1972.

PARRA FILHO, Domingos. Metodologia Cientifica. São Paulo: Futura, 1998.

PATTISON, D., CHRISTIAN A. BIRD , PREMKUMAR T. DEVANBU. Talk and work: a preliminary report. Proceedings of the 2008 international working conference on Mining software repositories, May 10-11, 2008.

PENNEBAKER, J., M. FRANCIS, & R. BOOTH. Linguistic inquiry and word count: Liwc 2001. Mahwah, NJ: Erlbaum, 2001.

PETERS, DEREK et al. Preferred 'learning styles' in students studying sports-related programme in higher education in the United Kingdom. Studies in Higher Education. Vol 33, pages 155 – 166, 2008.

PORTER, M. F. An algorithm for suffix stripping. Program, [S.l.], v.14, n. 3, 1980. http://telemat.die.unifi.it/book/2001/wchange/download/stem_porter.html, último acesso em Jul/2009.

POSTGRESQL CONTRIBUTORS. http://www.postgresql.org/community/contributors/, accessed in July, 2010.

POSTGRESQL. www.postgresql.org, accessed in July, 2010.

RAYMOND, E. S. A Catedral e o Bazar, Disponível em: http://www.geocities.com/CollegePark/Union/3590/pt-cathedral-bazaar.html, 1998, último acesso em Jul/2010.

REEVE, B.B. & FAYERS, P., Applying item response theory modelling for evaluating questionnaire item and scale properties. In Oxford, p. 55–76, 2005.

REISE, S. P., AINSWORTH, A. T., & HAVILAND, M. G. Item response theory: Fundamentals, applications, and promise in psychological research. Current Directions in Psychological Science, 14, 95-101, 2005.

RICHARDSON, ROBERTO JARRY. Pesquisa social: métodos e técnicas. São Paulo: Atlas, 1999.

RIDING, R. J. & CHEEMA, I. Cognitive styles: An overview and integration. Educational Psychology, 11, Issue 3 & 4, 193-215, 1991.

RIDING, R. J. Cognitive style: a review. International perspectives on individual diferences: cognitive styles. Vol. 1 (pp. 315-346}). Stamford, Conn: Ablex, 2000.

RIDING, R. J. Cognitive styles analysis. Birmingham: Learning and training performance, 1991.

RIDING, R. J. Personal styles awareness. Birmingham: Learning and Training Technology, 1994.

RIEL, A. J. Object-Oriented Design Heuristics. Addison-Wesley Professional, 1996.

RIGBY, P. & HASSAN, A. What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List. Proceedings of the Fourth International Workshop on Mining Software Repositories, 2007.

RIJSBERGEN, C. J. V. Information Retrieval, 2nd edition. Butterworths, London, 1979.

RIZOPOULOUS, D. Ltm: An R package for latent variable modeling and item response theory analyses. Journal of Statistical Software, 17 (5), 1-25, 2006.

ROBLES, G. & GONZALEZ-BARAHONA, J. M.. Contributor Turnover in Libre Software Projects. Open Source Systems, 273–286, 2006.

ROBLES, G.; GONZALEZ-BARAHONA, J. M. & HERRAIZ. I. Evolution of the core team of developers in libre software projects. In 6th IEEE International Working Conference on Mining Software Repositories, pages 167–170, May 2009.

RUDIO, FRANZ V.. Introdução ao Projeto de Pesquisa Científica. Petrópolis: Vozes, 1980.

RUIZ, JOÃO ÁLVARO. Metodologia Científica: Guia para eficiência nos estudos. São Paulo: Atlas, 1982.

SALLNAS, E.L., RASSMUS-GROHN, K., & SJOSTROM, C. Supporting presence in collaborative environments by haptic force feedback. ACM Transactions on Computer-Human Interaction, 7(4), 461–476, 2000.

SALTON, G.; WONG, A.;YANG C.S. A Vector Space Model for Automatic Indexing. Communications of the ACM, New York, v.18, 1975.

SANTOS, FRANCISCO R. ; COLAÇO JR., METHANIAS; MENDONÇA, M. G. . Uma extensão do Microsoft Visual Studio para predição de modificações de software. In: Simpósio Brasileiro de Banco de Dados, V WAAMD, Fortaleza, 2009.

SCIALDONE, MICHAEL J. et al. Group Maintenance Behaviors of Core and Peripherial Members of Free/Libre Open Source Software Teams. In Proceedings of the OSS 2009, pp. 298-309, 2009.

SEI. Capability Maturity Model Integration, Version 1.1 – CMMI for Software Engineering (CMMI-SW, V1.1) Continuous Representation (CMU/SEI-2002-TR-028). Software Engineering Institute, Carnegie Mellon University, August 2002.

SHARPLEY, C. F. Research findings on neurolinguistic programming: nonsupportive data or an untestable theory?, Journal of Counseling Psychology, vol. 34, no. 1, pp. 103-107, 1987.

SHOLOM, M. WEISS et al. Text Mining: Predictive Methods for Analyzing Unstructured Information. Springer, 2005.

SINGH, JAGDIP. Tackling Measurement Problems with Item Response Theory: Principles, Characteristics, and Assessment, with an Illustrative Example. Journal of Business Research, 57 (2), 184–208, 2004.

STEIN, D. et al. Creating shared understanding through chats in a community of inquity. The Internet and Higher Education 10(2), pp. 103-115, 2007.

TAURION, C. Software Livre: potencialidades e modelos de negócios. 1ª. ed, Brasport, 2004.

THOMPSON, J. E., COURTNEY, L., & DICKSON, D. The effect of neurolinguistic programming on organizational and individual performance: a case study, Journal of European Industrial Training, vol. 26, no. 6, pp. 292-298, 2002.

TOSEY, P. & MATHISON, J. Neuro-linguistic Programming and Learning Theory: a response. The Curriculum Journal Vol. 14 no.3 pp. 361 – 378, 2003.

TOSEY, PAUL & MATHISON, JANE. Fabulous Creatures of HRD: A Critical Natural History Of Neuro-Linguistic Programming. 8th International Conference on Human Resource Development Research & Practice across Europe, Oxford Brookes Business School, 2007. Available in http://www.nlpresearch.org/, last access in Dez/2009.

TRAVASSOS, G. H., F. SHULL, M. FREDERICKS, & V.R. BASILI. Detecting Defects in Object Oriented Designs: Using Reading Techniques to Increase Software Quality. Proc. OOPSLA, pp. 47-56, 1999.

TURAN, BULENT, STEMBERGER, RUTH M. The effectiveness of matching language to enhance perceived empathy. Communication & Cognition, Vol 33(3-4), 287-300, 2000.

UNDERWOOD, B. J. Individual differences as a crucible in theory construction. American Psychologist, 30, 128-140, 1975.

VAN BAAREN, R. B., HOLLAND, R. W., STEENAERTS, B., & VAN KNIPPENBERG, A. Mimicry for money: Behavioral consequences of imitation. Journal of Experimental Social Psychology, 39, 393-398, 2003.

VAN SOLINGEN, R. & BERGHOUT, E. The Goal/Question/Metric Method: A practical guide for quality improvement of software development. McGraw-Hill, 1999.

VARK-LEARN. Disponível em: http://www.vark-learn.com/. Acesso em: 25 jul. 2010.

VERGARA, SYLVIA CONSTANT. Projetos e Relatórios de Pesquisa. São Paulo: Atlas, 2006.

VISA, A. Technology of Text Mining. Proceedings of Machine Learning and Data Mining in Pattern Recognition, Second International Workshop, MLDM 2001, Leipzig, 2001.

WEKA. Data Mining with Open Source Machine Learning Software in Java, available in http://www.cs.waikato.ac.nz/ml/weka, accessed in December, 2009.

WITTE, R., LI, Q., ZHANG, Y., & RILLING, J. Text mining and software engineering: An integrated source code and document analysis approach. IET Softw – Special Section on Natural Language in Software Engineering, 2, (1), pp. 3–16, 2008.

WOHLIN, P. RUNESON, M. HOST, M. C. OHLSSON, B. REGNELL, & A. WESSLÉN. Experimentation in software engineering: an introduction. Kluwer Academic Publishers, 2000.

WORDNET. A lexical database for the English language, Princeton University, 2006.

WU, M. & ADAMS, R. Applying the Rasch model to psycho-social measurement: A practical approach. Educational Measurement Solutions, Melbourne, 2007.

YOUNG, S. M., EDWARDS, H. M., MCDONALD, S. & THOMPSON, J. B. Personality characteristics in an XP team: A repertory grid study. In: Workshop on Human and Social Factors of Software Engineering (HSSE), May 16, 2005, St. Louis, Missouri, USA, ACM Press, 2005.

ZHANG, W., & STORCK, J. Peripheral members in online communities. In: AMCIS 2001, Boston, MA. – USA, 2001.

ZIMMERMANN, T.; KIM, S.; WHITEHEAD ,E. J. & ZELLER, A. Predicting Faults from Cached History. In Proc. of the 29th International Conference on Software Engineering (ICSE 2007), pages 489–498, 2007.

ZIMMERMANN, T.; WEISSGERBER, P. Preprocessing CVS data for fine-grained analysis. In: International Workshop on Mining Software Repositories, 2004.

ANEXOS

ANEXO A – Questionário em inglês

	Questions	visual	auditory	kinesthetic
1	When you are introduced to a new system, do you prefer...	seeing a UML diagram?	talking to yourself and to other people?	using it first to feel how it could evolve?
2	When you have a maintenance task to execute, do you prefer...	imagining different perspectives?	listening and talking about options to solve the task?	exposing your ideas no matter what?
3	When you implement a class, which is tested successfully, do you prefer...	designing a clear diagram for everybody to see?	telling everyone in the team the news?	greeting and patting everyone on the back?
4	When the possibility of a tool or methodology change is being discussed, do you prefer...	imagining the possibilities?	discussing the options?	having a flexible attitude?
5	In the training courses, do you prefer...	summarizing the meaning?	listening to the message, word by word?	taking the main point of the message?
6	When you need to understand a new class, do you initially prefer...	seeing an image which presents the respective class?	listening to someone talking about it?	looking through it and executing a test?
7	In the requirements analysis, do you prefer...	seeing a general schema of the users views?	listening carefully to users comments?	feeling the pressure of the needs?
8	In the definition of a software architecture, do you prefer...	having a general view of the situation?	listening and discussing ideas?	giving lots of suggestions?
9	If you need to program in a non-familiar place, will you prefer...	thinking and visualizing the unusual day you'll have?	talking about your programming (on that day)?	just living the experience of how you'll spend the day?
10	When you need to use a new library, do you prefer...	searching for a specialist's vision?	talking to someone who is able to discuss such use?	using another programmer's experience?
11	To acquire a new bibliography on Software Engineering you rely on:	an attractive and useful appearance (inside and outside the book).	a professional who has talked about it and recommended it.	the presence of real case studies and examples.
12	When you interview a less experienced programmer to work with you, do you prefer...	examining all aspects of his/her potential?	asking questions about specific comments in his/her resume?	mastering the tools and knowledge experienced by the interviewed?
13	To learn a new programming language, paradigm or design pattern, do you prefer:	observing the documentation diagrams.	talking to professionals who master the technique.	starting using the technique to learn by practicing.
14	When you use a website of a Software Engineering tool, what do you like to have:	screenshots showing what each module implements.	the opportunity to ask questions and talk about the tool and its features.	lots of examples of good and bad uses of the tool.
15	You were suddenly invited to present your latest implementation. How would you prepare your presentation? Think of what you would like to see first if you were in public. Remember also that there is little time to prepare it.	you would make diagrams and graphics to show.	you would write down the key points of the code and practice the speech.	you would join real and practical examples to explain.
16	Now try to remember a moment which you could totally comprehend a new code. How did you better comprehend it?	by using diagrams and graphics.	by listening to the programmer's explanations and asking questions.	by executing and writing the code for some type of maintenance.

ANEXO B – Publicações e dissertações de mestrado produzidas

Artigos publicados:

- Data Warehousing in an Industrial Software Development Environment. In: 33th IEEE/NASA SEW, 2009.
- Mining Software Change History in an Industrial Environment. In: XXIII Brazilian Symposium on Software Engineering, 2009.
- Uma extensão do Microsoft Visual Studio para predição de modificações de software. In: Brazilian Symposium on Databases, WAAMD 2009.
- OSS Developers Context-Specific Preferred Representational Systems: A Initial Neurolinguistic Text Analysis of the Apache Mailing List. In: 7th IEEE Working Conference on Mining Software Repositories, 2010.

Palestra:

Mineração de Repositórios de Software para Melhoria do Processo de Desenvolvimento, ERBASE 2010.

Artigos Submetidos para Congressos Nacionais:

- Uma Ferramenta para Mineração Neuro-Lingüística de Listas de Discussão de Projetos de Software. In: Brazilian Symposium on DataBases, 2011.

Artigos Submetidos para Periódicos Internacionais:

- A Neurolinguistic-based Methodology for Identifying FOSS Developers Context-Specific Preferred Representational Systems. In: Information and Software Technology, 2011.

- Do Software Engineers Have Preferred Representational Systems? In: Innovations in Systems and Software Engineering Journal, 2011.

Publicações Planejadas

- IDE Usage Mining: Extracting Visual Interaction-Patterns from Eclipse;
- Um Experimento Realizado na Indústria para Classificação e Validação dos Sistemas Representacionais Preferidos de Programadores;
- Um classificador Neuro-linguístico para melhoria da Gestão de Comunicação em Projetos de Software (Congresso de Administração);
- Uma metodologia Neuro-Lingüística para medição da empatia das mensagens trocadas em listas de discussão de projetos de software.

Dissertações de Mestrado

- Geyson Amaral. Classificação de Estilos de Aprendizagem através da Mineração de Ambientes de Educação à Distância (EAD). Início: 2011. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Sergipe. (Orientador).
- Alex Paulo Alves de Oliveira. Identificação da Maturidade CMMI em Projetos de Software. Início: 2010. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Pernambuco. (Co-orientador).
- Arquimedes Sidney Lima de Medeiros. Análise de Sentimentos de Mensagens Postadas na Internet. Início: 2010. Dissertação (Mestrado em Ciências da Computação) - UFPE. (Co-orientador).
- Mário André de Freitas Farias. Ontologia aplicada a um Dicionário Neurolinguístico para Mineração de Artefatos de Software. Início: 2009. Dissertação (Mestrado em Modelagem Computacional de Conhecimento) - Universidade Federal de Alagoas. (Co-orientador).

- Francisco Rodrigues. Aplicação e Validação da Mineração de Modificações de Software em Ambiente Industrial. Concluída. Dissertação (Mestrado em Sistemas e Computação) - Universidade Salvador. (Co-orientador).