

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO**

**PUC/SP**

**Leila Souto de Assis**

**Uma Aproximação Prática no Ambiente de Trabalho: Resolução  
de Problemas em Matemática e Processo de Manutenção de  
Sistemas Computacionais**

**DOUTORADO EM EDUCAÇÃO MATEMÁTICA**

**São Paulo**

**2011**

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO  
PUC/SP**

**Leila Souto de Assis**

**Uma Aproximação Prática no Ambiente de Trabalho: Resolução  
de Problemas em Matemática e Processo de Manutenção de  
Sistemas Computacionais**

*Tese apresentada à Banca Examinadora da  
Pontifícia Universidade Católica de São Paulo,  
como exigência parcial para a obtenção do título  
de **DOUTOR EM EDUCAÇÃO MATEMÁTICA**, sob  
a orientação da **Professora Doutora Celina  
Aparecida Almeida Pereira Abar**.*

**São Paulo**

**2011**

**Banca Examinadora**

---

---

---

---

---

Autorizo, exclusivamente para fins acadêmicos e científicos, a reprodução total ou parcial desta Tese por processos de fotocopiadoras ou eletrônicos.

**Assinatura:** \_\_\_\_\_ **Local e Data:** \_\_\_\_\_

## DEDICATÓRIA

À minha mãe, irmãs Nanci e Tica e ao amor da minha vida: meu pai (*in  
memorian*).

## **AGRADECIMENTOS**

À minha orientadora, Prof<sup>a</sup> Dr<sup>a</sup> Celina Aparecida Almeida Pereira Abar, pela compreensão nos momentos de maiores crises pessoais e profissionais e também pela constante confiança, apoio e sugestões na elaboração deste projeto.

Ao Programa de Estudos Pós-Graduados da Pontifícia Universidade Católica de São Paulo, que me proporcionou a oportunidade de estudar e concluir este Doutorado.

À banca examinadora que, com muita compreensão, sugeriu valiosas sugestões e críticas.

Aos profissionais participantes pela contribuição imensurável a este trabalho assim como pela colaboração com sugestões enriquecedoras.

Aos colegas de doutorado e amigos que contribuíram para meu crescimento pessoal e profissional, durante o tempo.

E, finalmente, à família pela presença constante nesta caminhada e por representar a melhor parte de minha vida!

## RESUMO

Esta tese trata do uso da Matemática nas práticas profissionais, mais especificamente da existência de aproximação entre as estratégias de resolução de problemas em Matemática e o processo de manutenção de *softwares* no ambiente de trabalho de programadores de computador. A metodologia adotada foi a *Grounded Theory* (GT), na visão de Glaser, Strauss, Corbin e Charmaz. Foi utilizado, como técnica de coleta de dados, entrevistas semi-estruturadas bem como o aplicativo GRUMPS para registro das interações dos programadores durante a execução cotidiana de seu trabalho. Foram analisadas as práticas de trinta e quatro programadores observados e entrevistados durante o exercício de sua profissão em seu ambiente corporativo. A pesquisa exploratória indicou características do pensamento matemático na resolução de problemas do cotidiano dos profissionais participantes deste estudo. Como resultado, concluímos que as estratégias de resolução de problemas na Matemática, aproximadas à realidade destes especialistas podem potencialmente contribuir para o processo de manutenção de sistemas computacionais. Isto porque na Computação a compreensão dos programas pré-existentes, e o cuidado em não se inserir novos problemas em função de alterações realizadas, são características fundamentais para garantir o sucesso desse tipo de perfil profissional no mercado de trabalho.

**Palavras-Chave:** Educação Matemática, Ciências da Computação, manutenção de sistemas, resolução de problemas, ambiente de trabalho.

## **ABSTRACT**

This thesis handles the use of Mathematics in the workplace, more specifically about the existence of relationship between Mathematics problem solving strategies and the software maintenance process in the computer developers' workplace. The adopted methodology was Grounded Theory (GT), under Glaser, Strauss, Corbin and Charmaz' perspectives. It has been used, as the data collecting technique, semi-structured interviews as well as GRUMPS tool to record all developers' interactions during their daily duty. We analyzed the current practices of thirty four developers observed and interviewed during the exercise of their profession in their corporate environment. The exploratory research indicated characteristics of mathematical thought during the developers' professional problem solving process. As an outcome, we concluded that problem solving strategies in Mathematics, connected to the reality of this type of specialists may potentially contribute to the computational systems maintenance process. This would happen because, in Computer Science, the comprehension of pre-existing programs, and the attention to not insert new issues based on the changes performed, are crucial skills to ensure the success of this type of professional in the market place.

**Keywords:** Mathematics Education, Computer Science, systems maintenance, problem solving, workplace.

## LISTA DE FIGURAS

Figura 1: Uma visão geral do referencial teórico-metodológico da presente pesquisa.....	55
Figura 2: Adaptação da figura de estratégias e métodos de pesquisa de De Villiers (2005) (tradução nossa).....	57
Figura 3: Exemplo de janela do aplicativo GRUMPS para coleta de opiniões parciais dos programadores.....	88
Figura 4: Exemplo registros de Sessão (primeira tabela) e Evento/Ação (segunda tabela) extraídos do GRUMPS.....	89
Figura 5: Exemplo de um relatório detalhado mostrando as ações de uma sessão, aplicativos acessados e tempo gasto.....	91
Figura 6: Síntese da categorização conceitual resultante da etapa de Codificação Axial da GT.....	100



## LISTA DE QUADROS

Quadro 1: Definições, segundo IEEE (1998), para as categorias de manutenção de software.....	25
Quadro 2: Esquema das sessões conduzidas para este estudo.....	65
Quadro 3: Exemplo de relatório consolidado pela pesquisadora a partir dos dados extraídos do aplicativo GRUMPS para algumas sessões e ações de 3 programadores participantes.....	95
Quadro 4: Exemplo de registros na fase de Codificação Aberta.....	97
Quadro 5: Lista de Categorias oriundas dos principais Códigos In Vivo identificados nesta pesquisa.....	98



## LISTA DE GRÁFICOS

Gráfico 1: Distribuição da frequência de idade dos programadores participantes da segunda fase de coleta de dados.....	80
Gráfico 2: Distribuição dos cursos de graduação dos programadores participantes da segunda fase de coleta de dados.....	80
Gráfico 3: Distribuição do ano de conclusão da graduação dos programadores da segunda fase de coleta de dados.....	81
Gráfico 4: Distribuição das instituições de graduação dos programadores participantes da segunda fase de coleta de dados.....	81
Gráfico 5: Distribuição dos cursos de pós-graduação dos programadores participantes da segunda fase de coleta de dados.....	82
Gráfico 6: Distribuição da experiência profissional (em anos) dos programadores da segunda fase de coleta de dados.....	82
Gráfico 7: Distribuição dos cargos dos programadores participantes da segunda fase de coleta de dados.....	84
Gráfico 8: Desempenho escolar dos programadores participantes da segunda fase de coleta de dados.....	85



# SUMÁRIO

LISTA DE FIGURAS.....	i
LISTA DE QUADROS.....	iii
LISTA DE GRÁFICOS.....	v
INTRODUÇÃO .....	1
ESTRUTURA DE EXPOSIÇÃO .....	5
<b>1 APRESENTAÇÃO DA PESQUISA.....</b>	<b>7</b>
1.1 OBJETIVOS.....	7
1.2 JUSTIFICATIVAS .....	9
1.3 QUESTÕES NORTEADORAS .....	17
1.4 CONSIDERAÇÕES PRELIMINARES .....	18
<b>2 MANUTENÇÃO DE SOFTWARE: PROCESSO, DEFINIÇÕES E DESAFIOS.....</b>	<b>19</b>
2.1 PROCESSO DE MANUTENÇÃO DE SOFTWARE.....	19
2.2 CONCEITOS .....	21
2.3 TIPOS DE MANUTENÇÃO DE SOFTWARES .....	23
2.4 DESAFIOS NO PROCESSO DE MANUTENÇÃO DE SOFTWARES .....	25
2.5 CONEXÕES ENTRE O PROCESSO DE MANUTENÇÃO DE SOFTWARE E OS OBJETIVOS DESTA PESQUISA.....	26
<b>3 SOBRE RESOLUÇÃO DE PROBLEMAS .....</b>	<b>29</b>
3.1 RESOLUÇÃO DE PROBLEMAS EM MATEMÁTICA.....	29
3.2 APORTES TEÓRICOS SOBRE HEURÍSTICA EM UM CONTEXTO DE RESOLUÇÃO DE PROBLEMAS.....	34
3.3 RESOLUÇÃO DE PROBLEMAS: PRIMEIRAS APROXIMAÇÕES .....	36
<b>4 METODOLOGIA .....</b>	<b>55</b>
4.1 ABORDAGEM QUALITATIVA .....	56

4.2	PESQUISA INTERPRETATIVA.....	57
4.3	<i>GROUND</i> ED THEORY (GT) .....	60
4.4	ANÁLISE DOS DADOS POR MEIO DA GT.....	62
5	PROCEDIMENTOS METODOLÓGICOS.....	67
5.1	ETAPA 1 .....	67
5.2	ETAPA 2.....	78
5.3	CRITÉRIOS DE ANÁLISE .....	93
6	ANÁLISE DOS DADOS .....	101
6.1	BUSCA SISTEMÁTICA DE PADRÕES .....	102
6.2	INDEPENDÊNCIA.....	109
6.3	JULGAMENTO .....	114
6.4	ORIGINALIDADE .....	119
7	CONSIDERAÇÕES FINAIS .....	125
8	REFERÊNCIAS BIBLIOGRÁFICAS .....	131
9	ANEXOS .....	141
9.1	ANEXO 1 – ROTEIRO PARA ABORDAGEM INICIAL (ETAPA 1) .....	141
9.2	ANEXO 2 – ESPECIFICAÇÃO TÉCNICA-FUNCIONAL DO COMPONENTE DE <i>SOFTWARE</i> (ETAPA 1) .....	142
9.3	ANEXO 3 – QUESTIONÁRIO PÓS-IMPLEMENTAÇÃO (ETAPA 1).....	145
9.4	ANEXO 4 – ROTEIRO PARA ABORDAGEM INICIAL (ETAPA 2) .....	147
9.5	ANEXO 5 – QUESTIONÁRIO PÓS-IMPLEMENTAÇÃO (ETAPA 2).....	148
9.6	ANEXO 6 – TABULAÇÃO DOS DADOS DO ROTEIRO INICIAL (ETAPA 2) .....	150

## INTRODUÇÃO

O presente estudo marca um momento pessoal importante de uma trajetória, que se iniciou ainda no curso de graduação em Tecnologia de Processamentos de Dados. Neste curso mantivemos os primeiros contatos com pesquisas voltadas ao estudo do uso de recursos computacionais como soluções potencialmente facilitadoras para trabalhos desenvolvidos em instituições particulares ou situações em ensino-aprendizagem.

Concluído o curso, e após um ano de estudos voltados ao entendimento de dificuldades de aplicação do conceito de variância por negociadores de mesas de operações bancárias em um curso de pós-graduação, decidimos aprofundar esses estudos, e ingressamos no Mestrado em Educação Matemática na Pontifícia Universidade Católica de São Paulo (PUC-SP), ano de 2003.

Após um semestre como aluna especial em tal Programa, ingressamos efetivamente como aluna regular participando do grupo de pesquisa G3 - Tecnologias e Meios de Expressão em Matemática (TecMEM), no qual as investigações são dirigidas à compreensão e debate sobre formas de incorporação do uso de tecnologias no ensino da Matemática.

Nossa participação nos estudos e pesquisas sobre ambientes informatizados e as relações entre práticas matemáticas, aprendizagem e tecnologias digitais, instigaram-nos a desenvolver nosso mestrado sobre concepções de professores de Matemática sobre a utilização de objetos de aprendizagem, focando-nos especificamente em alguns casos selecionados do projeto RIVED-Brasil.

Participando nos anos seguintes de consultorias sobre a exploração do uso de redes informáticas para instituições privadas, nosso enfoque consistiu na análise e levantamento de requisitos para a construção de sistemas. Para isso, foram utilizados os mais diversos tipos de tecnologia disponíveis no mercado aplicados à implementação de portais de *Internet* corporativos para desenvolvimento e formação de funcionários e colaboradores, divulgação de marca e arquitetura de sistemas em ambientes informáticos.

Como resultado do constante contato com profissionais da Ciência da Computação e afins, essa experiência nos permitiu um primeiro delineamento de algumas dificuldades de entendimento e uso de alguns recursos computacionais ligados às práticas destes especialistas, especialmente àquelas ligadas à Matemática presente na compreensão de certos processos para elaboração dos artefatos desejados (neste caso, programas para um sistema maior).

Diante disso, esta pesquisa visa identificar e descrever as aproximações entre as estratégias de resolução de problemas da Matemática com relação às aplicadas pelos programadores no processo de manutenção de *softwares*, a fim de estabelecermos potenciais contribuições da Matemática às práticas de tais profissionais.

Entre algumas das justificativas que alavancaram a iniciativa desta pesquisa podemos destacar que, nesta época de intensa dinâmica concorrencial e profundas mudanças nas organizações, os profissionais da Computação estão engajados na concepção de sistemas de informação destinados a servir os objetivos estratégicos, táticos e operacionais das organizações. Necessitam, portanto, para esse efeito, possuir certas competências-chave nos domínios da Matemática, além de suas especialidades dentro da Computação, que permitirão a eles conceber, gerir e avaliar de forma sistemática e integradora as melhores soluções.

Consideramos relevantes para este estudo que a Matemática também deva ter como fim preparar os estudantes para exercerem com sucesso suas profissões. Acreditamos que as competências no campo da Matemática que se adquirem na escola, irão possibilitar a tais alunos atuarem como interlocutores de sucesso ao desempenhar suas funções-chave na gestão de informação nas empresas.

Esses elementos motivaram a nossa opção pelo Curso de Doutorado em Educação Matemática da PUC-SP, ainda na linha de pesquisa do grupo G3 - TecMEM, visando identificar aproximações sobre o pensamento da Matemática no ambiente de trabalho de um profissional da Ciência da Computação e afins.

Neste contexto, este trabalho tem por objetivo investigar quais são as aproximações entre as estratégias de resolução de problemas em Matemática e a prática de programadores. Buscamos identificar algumas potenciais contribuições das estratégias da Matemática como agente influenciador em um meio onde a compreensão dos programas pré-existentes é fundamental para acelerar o processo de entendimento e resolução dos problemas de seu cotidiano profissional.



## ESTRUTURA DE EXPOSIÇÃO

Para estruturação deste trabalho, ele foi dividido em sete capítulos, conforme descrito a seguir.

O capítulo 1 é destinado à apresentação do problema de pesquisa, dos objetivos, justificativas, questões norteadoras e primeiros resultados. Devido à escolha da *Grounded Theory* (GT) como metodologia de pesquisa deste estudo, propositalmente não levantamos hipóteses nas primeiras reflexões desta investigação. Segundo tal metodologia, as mesmas devem ser criadas a partir do processo da coleta e análise dos dados e não antes do pesquisador entrar em campo.

O capítulo 2 apresenta um panorama geral sobre a evolução dos sistemas computacionais e o processo de manutenção de *softwares* contextualizando os conceitos envolvidos, principais características e desafios futuros.

No capítulo 3, são apresentados aportes teóricos da Heurística<sup>1</sup> da resolução de problemas em Matemática, e também um breve histórico sobre a Resolução de Problemas. Entre estes aportes estão os apresentados aqueles de Arquimedes, Pappus, Descartes e Polya.

No capítulo 4 tratamos da metodologia de pesquisa desta tese que é a *Grounded Theory* (GT). Em linhas gerais, segundo Glaser e Strauss (1967), a GT consiste numa “abordagem de pesquisa qualitativa com o objetivo de descobrir teorias, conceitos e hipóteses, baseados nos dados coletados, ao invés de utilizar aqueles predeterminados” (tradução nossa). Esta metodologia possui raízes no Interacionismo Simbólico (IS) e compreende a realidade a partir do conhecimento da percepção ou significado que certo contexto ou objeto tem para a pessoa.

Utilizamos tal metodologia por acreditarmos que, por meio de seus procedimentos de análise, ela fornece métodos para construção de teoria com base nos dados investigados de determinada realidade (neste caso, o processo profissional de manutenção de programas de computador), de maneira

---

<sup>1</sup> O vocábulo heurístico deriva do grego *heuritiko* e significa que serve para descobrir. Esse método é também chamado método da redescoberta, método interrogatório ou método socrático.

indutiva ou dedutiva que, mediante a organização em categorias conceituais, possibilita a explicação do fenômeno investigado (que para este estudo, consiste em refletir e elucidar aproximações com as estratégias de resolução de problemas na Matemática).

O capítulo 5 apresenta os sujeitos, os procedimentos (divididos em duas fases: estudo inicial de três profissionais e posteriormente de outros trinta e quatro) bem como os critérios de análise e os instrumentos de coleta dos dados utilizados. A fim de justificarmos o porquê dos procedimentos terem sido divididos em duas grandes fases, baseamo-nos novamente na GT por possibilitar um método de pesquisa circular, permitindo-nos mudar o foco de atenção e buscar outras direções, reveladas pelos dados que entraram em cena. Encontramos neste método investigativo circular uma oportunidade de saturar a coleta de dados para sua posterior codificação e análise.

O capítulo 6 apresenta a análise dos resultados à luz dos conceitos e dos marcos teóricos face a face com as observações *in loco* e entrevistas realizadas, expondo o entendimento obtido a respeito dos argumentos não somente dos programadores, como também dos autores em que este estudo está baseado.

No capítulo 7 são apresentadas as considerações finais desta pesquisa, assim como considerações sobre a contribuição da para análises futuras e mais aprofundadas neste campo de estudo da Educação Matemática.

Estão presentes ao final os anexos, entre eles o roteiro de perguntas da primeira seção de introdução das Etapas 1 e 2, a especificação técnica-funcional do componente desenvolvido na Etapa 1, o questionário da entrevista realizada após a fase de conclusão da manutenção do sistema pelos programadores nas Etapas 1 e 2 bem como a tabulação dos dados do Roteiro Inicial da Etapa 2 deste trabalho.

# 1 APRESENTAÇÃO DA PESQUISA

## 1.1 Objetivos

Esta pesquisa busca contribuir com as discussões sobre a Matemática colocada em prática nos locais de trabalho. Segundo Kolikant e Pollack (2002), a Matemática é indispensável para o conhecimento ligado à Computação. Para estes pesquisadores, os programas são expressões matemáticas, as linguagens de programação podem ser interpretadas como teorias matemáticas, e a própria programação seria uma atividade Matemática.

Estes autores, por um lado, categorizam os programadores como engenheiros que devem colocar a Matemática em prática, e por outro, constataam que poucos estudos sobre esta categorização foram feitos. Estes estudos teriam como finalidade verificar as habilidades matemáticas envolvidas no entendimento e execução de atividades no trabalho.

Sobre estas habilidades citamos também os estudos de Tall (2001) que defendem que ao invés do uso do convencimento deve-se utilizar a prova como uma forma lógica de transicionar da Matemática elementar para a Matemática avançada (*Advanced Mathematical Thinking ou AMT*). Isso significa que para Tall (2001), assim como para Kolikant e Pollack (2002), evidenciar o raciocínio lógico e colocá-lo em prática são ações essenciais na formação de profissionais da Computação. Afinal, sintetizando o papel dos programadores de *software*, podemos dizer que por meio da manipulação de entidades abstratas e definições formais das linguagens de computador, estes indivíduos devem construir sistemas funcionais completos.

Embora o desenvolvimento de novos aplicativos computacionais seja uma área em constante expansão devido às necessidades de mercado, a manutenção de *softwares*, que envolve melhorias, adaptações e correções aos sistemas computacionais existentes, é responsável por mais da metade do tempo gasto pelos programadores (Layzell et al, 1993; Holgeid et al, 2000). Do ponto de vista técnico, segundo Corritore e Wiedenbeck (1999, p.12), a manutenção dos programas se constitui em um desafio ainda maior do que sistemas novos visto

que “os desenvolvedores primeiramente deverão compreender a situação atual, analisar o impacto das mudanças requeridas, planejar as modificações no código para somente então passarem a programar” (tradução nossa).

É nosso pressuposto que a manutenção é representativa no processo computacional e por isso, requer uma extensa compreensão e poder de resolução de problemas por parte dos programadores. Basicamente, o programador deve ter um bom entendimento do objetivo do programa bem como de que forma ele foi construído, a fim de fazer modificações estruturais e estender funcionalidades, sem introduzir novos erros.

Além disso, focando na coleta de dados para esta pesquisa, a manutenção de *softwares* se tornou uma alternativa bastante interessante no sentido de desafiar, de forma mais próxima da realidade, os profissionais participantes deste estudo. Dividimos a coleta de dados deste estudo em dois momentos: um primeiro grupo de três profissionais desafiados para solucionar um problema pré-determinado por esta pesquisadora (relacionado à criação de um componente de calendário) e posteriormente, um segundo grupo de trinta e quatro programadores observados diretamente em seu dia-a-dia (resolvendo problemas reais na manutenção dos programas de computador).

Optamos por realizar um segundo ciclo investigativo devido ao fato de que todos os programadores selecionados possuem experiência mínima de cinco anos na área, e portanto, a abordagem inicial de uma tarefa pré-determinada para observação do processo de resolução e posterior análise, não se mostrou muito eficiente e de fato, pouco provocativa. Esses motivos trouxeram à tona que a observação *in loco* dos programadores realizando seu trabalho durante o processo de manutenção dos *softwares* forneceria elementos interessantes para a fundamentação deste trabalho.

Embora já disponíveis estudos experimentais sobre programadores de computador realizando tarefas de manutenção (Littman et al, 1986; Pennington, 1987; Koenemann e Robertson, 1991; Von Mayrhauser e Vans, 1997), nosso entendimento é que o processo de manutenção de sistemas potencialmente

alinhado às estratégias da Matemática para resolução de problemas, ainda é um campo inexplorado.

Visamos investigar se a resolução de problemas em Matemática poderá auxiliar os programadores nas tarefas de manutenção de sistemas no processo de compreensão dos programas existentes e implementação prática das atividades.

A importância da resolução de problemas para a Ciência tem inspirado pesquisadores a pensar em estratégias para a mesma. Estas estratégias são importantes no caso computacional porque neste campo os problemas são mais estruturados e exigem uma linguagem “Matemática” formal, entendida pelo computador.

Esta tese tem por objetivo também contribuir com a identificação da Matemática presente no ambiente profissional e corporativo ao aproximar as estratégias de resolução de problemas da Matemática com relação às aplicadas pelos programadores no processo de manutenção de *softwares*.

Buscamos também identificar como estratégias aplicadas à resolução de problemas conectam-se às decisões e julgamentos realizados pelos desenvolvedores em sua prática profissional, abrindo caminhos para um melhor direcionamento estratégico-pedagógico dos conteúdos de alguns tópicos do ensino-aprendizagem da Matemática no âmbito da Ciência da Computação e afins.

## **1.2 Justificativas**

De acordo com Hartmanis (1995), os profissionais de sucesso na área da Ciência da Computação e afins possuem a habilidade lógico-matemática como uma de suas principais características para lidar com uma imensidade de fenômenos e informações, e assim conseguem abstrair o detalhe e a visão geral, e simultaneamente caminhar entre estas duas dimensões.

Forbellone e Eberspacher (2000, p. 71) consideram os especialistas na área de desenvolvimento de programas como uma profissão que:

[...] busca utilizar corretamente as leis de pensamento, da ordem da razão e processos de raciocínio e simbolização formais na programação de computadores, objetivando racionalidade e o desenvolvimento de técnicas que cooperam para a produção de soluções logicamente válidas e coerentes, que resolvam com qualidade os problemas que se deseja programar.

Para a análise da prática de programadores de computador nos referenciamos em Hoare e Jones (1989), quando eles destacam três aspectos do pensamento matemático ligados ao trabalho de profissionais da área da Computação:

- A contribuição direta para o conhecimento do domínio dos problemas (aplicação dos conceitos matemáticos para solucionar a situação em questão)
- As ferramentas essenciais utilizadas no processo de resolução da situação proposta
- A ligação entre a utilização de conceitos da Ciência da Computação e estratégias de resolução do problema em Matemática

Para que o computador execute as tarefas que desejamos, devemos estabelecer qual a lógica de programação que ele deve utilizar. Considerando que o raciocínio é algo abstrato, intangível, os seres humanos têm a capacidade de expressá-lo por meio da palavra falada ou escrita, baseada em um determinado idioma, que segue uma série de padrões definidos por uma gramática. Um mesmo raciocínio pode ser expresso em qualquer um dos inúmeros idiomas existentes, mas continuará representando o mesmo raciocínio, usando apenas outra convenção.

O mesmo ocorre com a lógica de programação, que pode ser concebida pela mente e implementada por meio de algoritmos, que na Computação, são representados por programas computacionais.

Farrer (1999, p. 52) descreve um algoritmo da seguinte forma:

Ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações.

Podemos encontrar diversas definições para o termo algoritmo na literatura atual. Em geral, que um algoritmo é um instrumento para realizar um conjunto de tarefas na resolução de um problema que esteja bem definido. Na medida em que precisamos especificar uma sequência de passos, precisamos ordenar o pensamento; portanto, precisamos elaborar um raciocínio lógico (FORBELLONE e EBERSPACHER, 2000).

Se pensarmos em atividades rotineiras, podemos constatar que, muitas vezes, estamos intuitivamente executando um algoritmo, ainda que não estruturado, por exemplo, quando utilizamos uma receita de bolo. Nela está descrita uma série de ingredientes necessários e uma sequência de passos (ações) que devem ser fielmente cumpridos para que se consiga fazer o bolo desejado.

Quando elaboramos um algoritmo, devemos especificar ações claras e precisas, que, a partir de um estado inicial, após um período de tempo finito, produzem um estado final previsível e bem definido. Isso significa que o algoritmo fixa um padrão de comportamento a ser seguido, uma norma de execução a ser trilhada, com vistas a alcançar, como resultado final, a solução de um problema, garantindo que, sempre que executado sob as mesmas condições, produza o mesmo resultado.

Um algoritmo tem por objetivo descrever o raciocínio lógico envolvido na solução de um problema. Dessa forma, viabiliza a abstração de uma série de detalhes computacionais via a lógica de programação.

Um fator relevante na construção de algoritmos é que, uma vez concebida uma solução algorítmica para um problema, esta pode ser traduzida para qualquer linguagem de programação e ser agregada às funcionalidades disponíveis nos diversos ambientes existentes.

Há muitas formas de se resolver um problema, pois cada pessoa pensa e age de maneira diferente. Cada indivíduo tem uma lógica própria, associada a padrões conhecidos por ele, e não a um objeto concreto. Isso significa que, para cada problema proposto, pode haver diversas estratégias para encontrar a solução. A prática e o bom senso indicarão qual é a mais adequada, que, com menor esforço e maior objetividade, produz o resultado almejado.

O homem é um agente computacional capaz de vivenciar as potencialidades e limitações desses dois mundos, o “real” e o “abstrato”. Isso não ocorre com os computadores, que, apesar de ser fruto de uma concepção matemática abstrata, são meios para resolver problemas no mundo real.

Os algoritmos computacionais devem seguir regras rígidas, organizadas e formalizadas a fim de que essas máquinas trabalhem de forma coerente. A Matemática pode ser útil nesta organização, pois viabiliza a formalização adequada do conteúdo dos algoritmos.

Como parte dos entregáveis finais mais significativos de sua profissão, os programadores são responsáveis por implementar algoritmos computacionais e por construir sistemas que, devido às novas necessidades de negócio, se tornam cada vez maiores e, possivelmente, mais complexos.

Friedmann et al (2005, p. 20) salientam que:

Resgatar parte das contribuições da Matemática à Ciência da Computação é um processo interessante sob o ponto de vista do ensino, pois possibilita a valorização de alguns conhecimentos matemáticos e como eles evoluíram a ponto de ganharem vida própria em outra ciência.

Esse processo serve também para apontar aquelas deficiências que dificultam o entendimento em fazer conexões entre conceitos matemáticos e como eles são usados no estudo dos algoritmos, pois existem semelhanças e diferenças ao se transladar esses conceitos para a Algorítmica.

O fato dessas deficiências no ensino da Matemática não serem abordadas, é um obstáculo para o entendimento do aluno sobre o que é pensar de forma algorítmica e sobre o que é um algoritmo.

Esses questionamentos são relevantes em um mundo informatizado, pois estão relacionados com a resolução de problemas e com as limitações impostas pela realidade física dos computadores, os quais utilizam procedimentos algorítmicos para executar tarefas, sejam elas simples ou complexas.

O trabalho de um profissional da Computação pode ser visto como incessantes transições entre a comunicação formal (humano-computador) e a informal (humano-humano). Tal especialista deve ser capaz de traduzir intuições e anseios dos usuários para uma linguagem formal, por meio do uso, por exemplo, de objetos matemáticos tais como funções, estruturas ou classes. Ele também deve ser apto a escrever sistemas de forma sucinta e compreensível a outros colegas de trabalho, principalmente quando falamos de complexos programas inter-relacionados e que possam necessitar de manutenção e expansão funcional a serem executada por outros especialistas concorrentemente.

Inúmeros estudos já foram realizados na tentativa de compreender-se um pouco mais acerca da manutenção de programas ao longo das últimas décadas. Détienne, por exemplo, aborda modelos cognitivos e experimentos conduzidos nesta área (Détienne, 2002) e provê uma pesquisa de estudos empíricos sobre as dificuldades que os novatos em programação possuem em relação ao modelo de programação orientado a objetos (Détienne, 1997). Storey (2005) ressalta algumas das teorias-chave para a compreensão de programas pré-existentes e discute como estas teorias são relacionadas às ferramentas que as suportam. Upchurch (2002) também fornece uma bibliografia extensa deste campo de estudo.

Outras pesquisas também foram conduzidas no contexto de estratégias gerais empregadas por programadores na compreensão de programas (Burkhardt et al, 1998; Wiedenbeck e Ramalingam, 1999; Mosemann e Wiedenbeck, 2001; Von Mayrhauser e Vans, 1996) bem como manutenção de *softwares* e melhorias (VON MAYRHAUSER e VANS, 1997; CORRITORE e WIEDENBECK, 1999; PARKIN, 2004).

É nosso pressuposto nesta tese ser relevante o estímulo existente na natureza da manutenção de sistemas em problematizar a observação prática, incentivando os programadores à procura de novos elementos que enderecem as lacunas ou novos requisitos da realidade.

Neste percurso, capturamos alguns dos indícios da Heurística em Descartes, Arquimedes, Pappus e em especial Polya (1945, 1981 e 1995), por ter sido responsável por estruturar didaticamente o processo de resolução de problemas, resgatando o senso lógico das respostas dos problemas em Matemática.

Destacamos também Wielewski (2005) que menciona que no âmbito da Matemática, a capacidade humana de perceber e resolver problemas é fortemente influenciada não somente pelo pensamento matemático, mas também por experiências pessoais, história cultural e social e representações que desencadeiam distintos processos de resolução de problemas. Em outras palavras, para o contexto desta pesquisa, as estratégias de pensamento matemático aliadas às atividades de manutenção de *software*, podem vir a desencadear e contribuir mais significativamente para o processo de resolução de problemas práticos dos programadores.

Durante a nossa formação profissional e no decorrer das leituras que fizemos, percebemos que a preocupação com a forma como são conduzidos os conteúdos de Matemática, no tocante a Resolução de Problemas, aparecem enfaticamente nos trabalhos da pesquisadora Onuchic (2004), do NCTM (1994), do pesquisador D'Ambrosio (1989), dos pesquisadores Lester (1993), Schoenfeld (1992 e 1996) e do precursor de um modo de conceber a resolução de problemas na Matemática, Polya (1945, 1981 e 1995), o qual apoiaremos este trabalho.

Para os pesquisadores anteriormente citados, cabe ao professor propiciar aos seus alunos oportunidades para deixar de ser apenas memorizadores e reprodutores de conhecimentos matemáticos. Há necessidade de que sejam transformadores e críticos, capazes de elaborar de forma concreta de construir seu conhecimento e desenvolver suas condições como cidadãos. Acreditam também, que o aluno se sente mais motivado em aprender Matemática quando é

incitado a resolver problemas de variados tipos e é capaz de resolvê-los de diferentes maneiras.

Onuchic (2004) resgata que no início do século XX o ensino de Matemática estava profundamente ligado à concepção de que Matemática de qualidade era aquela caracterizada pelo trabalho de repetição e memorização de tabuadas, bem como de algoritmos pré-definidos pelo livro didático e/ou professor. Anos depois, numa outra concepção, percebeu-se que os alunos deveriam aprender compreendendo, ou seja, além da técnica; deveriam entender o que faziam e por que faziam.

Dentre os autores de diferentes áreas do conhecimento que versam sobre a Resolução de Problemas, ressaltamos o que Saviani (2002, p. 87) afirma:

A essência do problema é a necessidade, uma questão em si não caracteriza um problema, nem mesmo aquele cuja resposta é desconhecida, mas uma questão cuja resposta se desconhece e se necessita conhecer. Eis aí um problema.

Segundo Saviani (2002), durante as aulas de Matemática, os problemas que favorecem os processos heurísticos contribuem para uma melhora do potencial criativo dos alunos. Além disso, proporcionam a aprendizagem mais significativa de conceitos, algoritmos e determinados formalismos em Matemática. O raciocínio e os processos heurísticos incluem estratégias de resolução de problemas. A capacidade de comunicar-se matematicamente, e o desenvolvimento de métodos de raciocínio para a resolução de problemas, são elementos essenciais da Matemática, e são esses elementos desencadeadores que buscaremos identificar nas aproximações com as práticas profissionais dos desenvolvedores de programa em suas estratégias para resolução de problemas de seu ambiente de trabalho.

Outro aspecto importante que justifica este estudo é acerca da investigação da Matemática no ambiente de trabalho, campo de pesquisa ainda não muito explorado. Segundo Noss e Hoyles (1996), somente algumas áreas tais como a Enfermagem, Engenharia Civil e Contabilidade possuem estudos significativos do uso da Matemática no ambiente de trabalho, e mesmo nestas, os trabalhadores

têm dificuldade em identificar a Matemática em sua rotina, especialmente àquelas “escondidas” (tradução nossa) pelo uso de diferentes tecnologias, incluindo o uso do computador.

Nesta mesma linha, Noss et al (2002) também reforçam a falta de mais pesquisas em áreas práticas do uso da Matemática, e ainda apresentam o conceito de cristalização de conceitos matemáticos por meio de processos cada vez mais automatizados, o que torna a Matemática cada vez mais invisível aos trabalhadores.

Noss et al (2006) também argumentam que as pesquisas realizadas em campos distintos de trabalho nos quinze anos anteriores apontam para uma conclusão muito similar: a investigação revela que os julgamentos rotineiros ou intuitivos, advindos da prática dos profissionais, embora não reconhecidos necessariamente como a Matemática visível (ou escolar), são verdadeiras evidências práticas da necessidade real do uso da Matemática nos ambientes de trabalho, e por isso, ilustram a relação intrínseca e fundamental de várias profissões e o ensino da Matemática.

Considerando como relevantes todas as justificativas anteriormente citadas, pretendemos analisar algumas das potenciais relações existentes entre os especialistas programando em seu dia-a-dia e a resolução das situações-problema da Matemática.

Intencionamos fazê-lo por meio do uso de conhecimentos matemáticos envolvidos no processo de programação de computadores, que segundo Ross e Howe (1981) fornecem rigor formal matemático, encorajamento à exploração das atividades e contexto para a solução de problemas em uma linguagem familiar aos especialistas. Desta forma, para estes autores, os especialistas podem, por meio da programação, descrever seu próprio pensamento de resolução para um problema dado.

### 1.3 Questões Norteadoras

Neste item, indicamos as questões que norteiam esta pesquisa:

Questão principal:

- *Que aproximações podemos estabelecer entre as estratégias de resolução de problemas da Matemática e o processo de manutenção de programas?*

Questões auxiliares:

- Tais aproximações com a resolução de problemas em Matemática podem ser consideradas como um ponto de partida para as atividades de manutenção de sistemas computacionais?

- Quais potenciais contribuições as estratégias de resolução de problemas da Matemática podem trazer aos desenvolvedores de programas durante o processo de manutenção dos *softwares*?

- Nas estratégias de resolução de problemas utilizadas pelos programadores, que aspectos seriam indícios da importância do pensamento matemático por parte dos desenvolvedores em sua prática profissional?

Pelo fato de termos nos apoiados na *Grounded Theory* (GT), não criamos hipóteses iniciais, visto que a metodologia ressalta o intuito de especificar e explorar a realidade investigada primeiramente, evitando pré-conceitos ou pressupostos iniciais por parte do pesquisador.

De fato, levantamos algumas considerações preliminares destacadas como indicativos das propriedades e as dimensões que identificamos nos dados durante o primeiro ciclo de coleta dos dados (estudo inicial de três profissionais). Usaremos estas considerações como ferramentas para nos ajudar a entender melhor os dados à nossa frente, emergidos no segundo ciclo de coleta dos dados (posteriormente de outros trinta e quatro profissionais).

## 1.4 Considerações Preliminares

Nesta pesquisa as considerações preliminares destacadas como resultados do primeiro ciclo de coleta dos dados são:

- A partir da observação das práticas do trabalho envolvidas nas atividades de programação dos profissionais da Ciência da Computação e afins, emergirão elementos que poderão ser encarados como oportunidades para conexões e criações de situações de ensino-aprendizagem de certos conceitos da Matemática essenciais para este perfil de especialistas.
- O ato de realizar a manutenção de um programa de computador favorece o desencadeamento da reflexão e dos processos heurísticos, provocando uma consciência maior de métodos de raciocínio e provas, elementos essenciais da Matemática.
- A aplicação de estratégias de resolução de problemas da Matemática no processo de manutenção de programas é um ponto de destaque e eixo de sustentação de atividades heurísticas, que podem auxiliar na preparação deste tipo de profissionais para o mercado de trabalho.

No próximo capítulo apresentaremos um esboço histórico, do ponto de vista de alguns autores, a respeito da evolução dos sistemas computacionais, destacando alguns dos desafios envolvidos no processo de manutenção de *softwares*.

## 2 MANUTENÇÃO DE SOFTWARE: PROCESSO, DEFINIÇÕES E DESAFIOS

Este capítulo tem como objetivo apresentar informações sobre o processo de manutenção de sistemas computacionais, bem como descrever algumas de suas definições e desafios no campo da Computação.

Ao final deste capítulo, apresentaremos também as conexões entre alguns dos desafios do processo de manutenção de *software* e os objetivos desta presente investigação.

### 2.1 Processo de Manutenção de Software

A Engenharia de *Software*, como aceita por muitos, iniciou-se em 1968, quando aproximadamente cinquenta especialistas em Computação de onze países se reuniram em Garmisch na Alemanha, para discutir os problemas de desenvolvimento de *software* da época (Naur e Randell, 1976). Durante a conferência, houve debates sobre o tema que os participantes escolheram chamar de “crise dos sistemas”. Entre os principais problemas relacionados a este tema estavam:

- Projetos realizados acima do orçamento
- Projetos finalizados acima do tempo esperado
- Produtos de *software* de baixa qualidade
- Produtos de *software* sem atender aos requisitos do cliente
- Projetos ingerenciáveis e com código de difícil manutenção

Segundo Naur e Randell (1976), a crise se concentrava no problema principal de como implementar e manter grandes, complexos e confiáveis sistemas computacionais por meio de uma forma controlada e efetiva, do ponto de vista de custo, tempo e qualidade.

Após mais de quatro décadas, muitos métodos, técnicas e teorias no âmbito da Engenharia de *Software* surgiram exatamente com o objetivo de resolver ou amenizar os problemas reportados durante a conferência. Grande parte destas evoluções veio como resultado do aprendizado produzido em projetos reais, e fizeram com que a Engenharia de *Software* amadurecesse como área da Ciência da Computação. O interesse em Engenharia de *Software* foi mantido ano após ano, e no atual estágio de evolução, este campo de conhecimento passou a dedicar também atenção à atividade de manutenção de *softwares*.

Essa postura decorre, em parte, da crescente quantidade de sistemas computacionais em funcionamento globalmente nas organizações, que por representarem investimentos significativos, precisam continuar em funcionamento por anos, momento no qual surge a necessidade de sua manutenção. Essa necessidade traz consigo problemas originados de diversas fontes, como a própria administração da organização, o perfil dos clientes ou as deficientes técnicas utilizadas na construção do *software* original.

O processo de manutenção constitui-se como uma parte fundamental do ciclo de vida do *software*, e a partir dele, é possível realizar correções, aperfeiçoamentos, adaptações e prevenções de erro em um sistema, garantindo sua continuidade. Como mudanças são inevitáveis ao longo da sua vida, mecanismos devem ser previstos para avaliar, controlar e fazer essas modificações de forma bem sucedida.

Segundo Robillard et al (2007), entender as atividades de manutenção permite identificar pontos críticos e tomar ações apropriadas para aumentar sua eficiência. De acordo com Swebok (2004), a manutenção de *software* é considerada a fase mais dispendiosa, do ponto de vista de custo, do ciclo de vida de um sistema, e muitas vezes devido à baixa qualidade dos reparos e atualizações dos programas, o desempenho total do aplicativo pode ser comprometido.

Embora não exista um consenso sobre o valor exato do custo atrelado à atividade de manutenção, as pesquisas na área apontam, na totalidade dos

casos, sempre mais de 50% dos investimentos realizados no *software*. De fato, para Bhatt et al (2006), esse percentual corresponde a algo entre 67% a 75% do investimento total, enquanto para Polo et al (1999) corresponde a um valor entre 67% a 90%. Ainda de acordo com esses últimos autores, a razão do custo elevado deve-se, em parte, à própria natureza da atividade de manutenção, caracterizada principalmente pela imprevisibilidade. Além dos custos financeiros, essa é também a atividade que exige maior esforço dentre as atividades de Engenharia de *Software*, conforme apontado por Sneed (2003). Pressman (2005) ainda completa que o grande esforço necessário na manutenção se justifica pela abrangência do significado desse termo no contexto de *software*.

Acreditamos que entender o que significa manutenção de *software*, e principalmente a abrangência do significado do termo, constitui passo fundamental para o melhor entendimento da relevância desta pesquisa. Trataremos de descrever alguns conceitos utilizados neste campo de estudo na seção a seguir.

## **2.2 Conceitos**

A atividade de manutenção de *software* é caracterizada pela modificação de um produto de *software* já entregue ao cliente final, para a correção de eventuais erros, melhora em seu desempenho, ou qualquer outro atributo, ou ainda para adaptação desse produto a um ambiente modificado (IEEE, 1998).

Embora a definição trate genericamente qualquer produto de *software*, existem diferenças entre a manutenção de *softwares* com propósitos distintos. Essa distinção é explicada por Pfleeger (2001), que estabelece três categorias de sistemas:

- 1) Na primeira classificação estão representados aqueles *softwares* construídos com base em uma especificação rígida e bem definida, cujos resultados esperados são bem conhecidos. Por exemplo, um sistema computacional construído para realizar operações com matrizes (adição, multiplicação e inversão). Nesse tipo de *software*, uma vez que

tenha sido construído considerando a correta implementação do método, dificilmente haverá a necessidade de manutenções.

- 2) Na segunda classificação são agrupados os *softwares* que constituem implementações de soluções aproximadas para problemas do mundo real. Como exemplo desta categoria, podemos citar um jogo de xadrez, versão para computadores pessoais. Embora suas regras sejam bem definidas, não é possível construir um *software* que calcule, a cada passo, todos os possíveis movimentos de peças do tabuleiro, de forma a determinar o melhor movimento. Isso porque o número de movimentos possíveis é muito grande para ser calculado em um intervalo de tempo relativamente curto para um computador pessoal. A técnica utilizada para desenvolver esse tipo de solução, baseia-se em descrever o problema de forma abstrata e então definir os requisitos de *software* a partir dessa abstração. Percebe-se que esse tipo de sistema já abre espaço para diferentes interpretações por parte do programador, o que tende a produzir *software* com maior necessidade de manutenção do que quando comparado aos da classificação anterior. Por considerar uma abstração para especificação de requisitos, a necessidade de mudança pode aparecer caso a abstração mude, na medida em que um maior entendimento do problema seja alcançado.
- 3) Finalmente, na terceira classificação são consideradas mudanças no ambiente em que o *software*, criado com base em um modelo dos processos abstratos envolvidos no sistema, precisará mudar sempre que ocorrerem alterações nesses modelos, sendo, portanto, parte do ambiente que ele modela. Um exemplo desse tipo de *software* seria aquele que apresenta informações da economia de um país. À medida que a economia passa a ser mais bem compreendida, o modelo muda e com ele a abstração do problema, causando uma necessidade inevitável de manutenção no sistema.

Atividades de manutenção de *software* são caracterizadas por intervenções no produto de *software* de forma a evitar sua deterioração. Um sistema não se desgasta como peças de um equipamento, mas se deteriora no sentido de os

objetivos de suas funcionalidades cada vez menos se adequarem ao ambiente externo.

Do ponto de vista de novos desenvolvimentos de *software*, Pfleeger (2001) explica que o foco das atenções está em produzir programas que atendam aos requisitos e funcionem corretamente. Isso inclui dizer que, a cada estágio do desenvolvimento, haverá uma referência contínua a elementos produzidos em estágios anteriores.

O desenvolvimento de um novo sistema levará a uma integração dos componentes desenvolvidos, passando por etapas de revisão e testes para certificação de seu funcionamento e adequação aos requisitos e ao projeto. Resumindo, o desenvolvimento inicial do sistema terá foco em considerar estágios anteriores do processo de maneira controlada.

A manutenção de *software* incluirá não apenas considerar resultados de etapas anteriores, mas atividades do presente, de forma a conseguir compreender o nível de satisfação dos usuários em relação ao sistema. Uma característica importante é também considerar o futuro durante a manutenção, buscando antever necessidades de mudanças nos negócios, o que causaria mudanças nessa solução.

Outra característica fundamental está relacionada à imprevisibilidade da manutenção. Esse fato está relacionado à influência que o sistema sofre de fatores externos, naturalmente imprevisíveis (Bhatt et al, 2006). Fica claro, portanto, que o sistema é sensível ao contexto no qual está inserido, estando sujeito a mudanças na medida em que seu ambiente muda.

### **2.3 Tipos de Manutenção de Softwares**

Na visão de Lientz e Swanson (1980), as ações ligadas à atividade de manutenção de *software* foram classificadas de acordo com sua natureza em três categorias: corretivas, adaptativas e perfectivas.

Manutenções do tipo *corretivas* visam corrigir defeitos de funcionalidade, o que inclui acertos emergenciais de programa. Pfleeger (2001) expõe um exemplo desse tipo de manutenção, que consiste em um usuário apresentando um problema de impressão em um relatório. O número de linhas impresso por folha é muito grande, o que causa sobreposição de informações. O problema foi identificado como uma necessidade de se alterar o programa do relatório para aceitar um parâmetro adicional, que determina o número máximo de linhas impressas por folha.

Manutenções do tipo *adaptativas* referem-se a adequar o *software* ao seu ambiente externo. O exemplo apontado por Pfleeger (2001) ilustra bem essa categoria. Suponha um sistema que gerencia a rede de computadores de uma empresa. Em uma atualização do gerenciador, os programadores perceberam que as rotinas existentes de acesso a estes computadores precisavam de mais um parâmetro adicional para tratar um novo fabricante de *hardware*. Essa manutenção corresponde a uma adaptação, uma vez que teve por finalidade adequação do *software* ao seu ambiente e não a correção de um defeito.

Manutenções do tipo *perfectivas* têm por objetivo acrescentar novos recursos de funcionalidade à solução, normalmente em razão de solicitações dos usuários. Significam ainda re-projetar partes de um *software*, de forma a tornar mais simples a compreensão e utilização do mesmo. Como exemplo, pode-se citar o pedido do usuário por um novo relatório com informações que até então não podiam ser obtidas do banco de dados.

Pigoski (1996) sugere a união das categorias *adaptativa* e *perfectiva* de uma forma única denominada *aprimoramentos*. Essa classificação estaria de acordo com organizações que geralmente utilizam o termo manutenção para se referir à execução de pequenas mudanças no *software*, enquanto o termo desenvolvimento é usado para as demais modificações.

Uma quarta categoria de manutenção é apresentada por alguns autores, como Pressman (2005). Essa categoria se refere às manutenções do tipo *preventivas* que buscam identificar previamente possíveis fontes de problemas no *software* e corrigi-las antecipadamente.

A IEEE (1998) modifica a definição apresentada inicialmente por Lientz e Swanson (1980), definindo uma categoria a mais chamada *emergencial*. Essa categoria é caracterizada pela execução de uma manutenção corretiva não planejada, com o intuito de manter o sistema operacional. Tal classificação insere a ideia de manutenção planejada e não-planejada, bem como manutenção reativa e proativa, como é ilustrado no Quadro 1.

	Planejada	Não-Planejada
Reativa	Corretiva Adaptativa	Emergencial
Proativa	Preventiva	-

Quadro 1: Definições, segundo IEEE (1998), para as categorias de manutenção de *software*

Dentro dessa classificação, um estudo com empresas de *software* realizado por Souza et al (2004), constatou que entre as organizações pesquisadas, as manutenções do tipo corretivas prevaleceram com 50% dos resultados, seguido pelas preventivas (30%) e adaptativas (20%). Nesta mesma pesquisa não foi identificada nenhuma organização realizando manutenção do tipo preventiva.

## 2.4 Desafios no Processo de Manutenção de Softwares

Nos desafios descritos por Paduelli e Sanches (2006), podemos listar as dez principais como os seguintes:

- 1) Registro inexistente ou superficial de manutenções anteriores
- 2) Documentação insuficiente ou superficial
- 3) Falhas de comunicação com o usuário a respeito de suas necessidades
- 4) Ausência de manutenção preventiva

- 5) Ausência de um ambiente computacional específico para manutenção
- 6) Estimativa de prazo não condizente com a complexidade do sistema, e por consequência, atrasos na entrega
- 7) Validação e teste insuficiente de manutenções efetuadas
- 8) Falta de compreensão do *software* e suas estruturas
- 9) Elevada rotatividade de membros e funções dentro da equipe
- 10) Manutenção de *software* é vista como uma tarefa não prestigiosa e consequentemente, não planejada

Frente a todos estes desafios e seus respectivos impactos financeiros, podemos ainda reforçar que existe também um risco de perda de oportunidades de negócio pela falta de gerenciamento e compreensão total da dinâmica do processo de manutenção do *software*. De acordo com a pesquisa realizada por Dart et al (2001), esse gerenciamento deve considerar três fatores: ferramentas, pessoas e processos, revelando-se, pois, uma atividade gerencial complexa.

Se por um lado a atividade de manutenção é dispendiosa, por outro ela é um desafio para as organizações que precisam considerá-la em seu dia-a-dia. Não é de se esperar que uma empresa troque todos seus sistemas a todo o momento: esses sistemas representam ativos importantes da organização, e ela estará disposta a investir de maneira a manter seus valores vivos (SOMMERVILLE, 2003).

## **2.5 Conexões entre o Processo de Manutenção de *Software* e os Objetivos desta Pesquisa**

O crescimento do número de sistemas computacionais legados e a dificuldade para mantê-los, baseado nos autores citados, contribuem também para justificar a relevância desta investigação.

Também partindo dos desafios existentes, podemos inferir que a manutenção de sistemas computacionais envolve certa incerteza (porque nem todas as situações as quais os programas serão submetidos são inesgotavelmente mapeadas), certo grau de auto-regulação (com constante julgamento e interpretação pelos programadores do que é considerado um comportamento esperado do sistema), bem como formalização lógica de critérios múltiplos para que o desempenho dos programas reflita seus requisitos originais.

Estamos assim perante um tipo de pensamento que pode ser encontrado na aprendizagem de vários conceitos matemáticos, presente em diversos níveis de escolaridade. Tal pensamento não pode ser visto apenas como um assunto específico ou mesmo de algo abordado só durante a graduação dos profissionais de Ciência da Computação e afins: trata-se de um conjunto de conceitos, representações e experiências potencialmente proporcionados por meio da interação com a Matemática.

Acreditamos que é o processo de como estes pensamentos são processados e julgados durante a resolução de um problema via programação, que determina a destreza dos profissionais desta área, possibilitando que a manutenção dos sistemas computacionais ocorra de forma mais precisa e por consequência, diferenciado-lhes como melhores especialistas no setor.

Partindo desta premissa, conforme já mencionado anteriormente, buscaremos identificar e descrever as potenciais aproximações das estratégias de resolução de problemas aplicadas pelos programadores no processo de manutenção de *software* àquelas aplicadas na Matemática, contribuindo para que as decisões tomadas por tais especialistas os conduzam, de forma mais eficiente, à investigação e resolução de problemas em seu âmbito profissional.

No próximo capítulo, trataremos de descrever e situar os conceitos de problema, resolução de problemas e a fundamentação teórica a qual esta pesquisa se baseia, apoiada principalmente em George Polya.



### 3 SOBRE RESOLUÇÃO DE PROBLEMAS

Este capítulo apresenta algumas definições sobre definição de problema, de forma mais genérica, bem como no escopo da Matemática. Resgatamos também alguns aportes teóricos sobre a heurística segundo Arquimedes, Pappus, Descartes e Polya.

Temos por objetivo descrever sinteticamente os trabalhos destes autores para esclarecer aspectos históricos da heurística na resolução de problemas, e principalmente elucidar alguns elementos estudados por Polya (1945, 1981 e 1995), os quais utilizaremos na fase de análise dos dados coletados para este estudo.

#### 3.1 Resolução de Problemas em Matemática

Primeiramente, gostaríamos de estabelecer a definição de problema utilizada nesta pesquisa, que seguirá os conceitos assim definidos por Kantowski (1997) e Charles e Lester (1982), descritos a seguir.

Segundo Kantowski (1997, p. 47), “uma pessoa está diante de um problema quando confronta uma questão que não pode dar a resposta, ou uma situação que não sabe resolver, usando os conhecimentos imediatamente disponíveis” (tradução nossa).

Já para Charles e Lester (1982, p. 51):

[...] um problema é uma tarefa para a qual: (1) o indivíduo, que com ela se confronta, quer e precisa encontrar uma solução; (2) o indivíduo não tem procedimento prontamente disponível para achar a solução e (3) o indivíduo deve fazer uma tentativa para encontrar a solução. (tradução nossa)

A respeito de problemas no âmbito da Matemática, concordamos com Vila e Callejo (2006, p. 37) que explicam:

[...] o termo problema para designar uma situação, proposta com finalidade educativa, que propõe uma questão matemática cujo método de solução não

é imediatamente acessível ao aluno/resolvedor ou ao grupo de alunos que tenta resolvê-la, porque não dispõe de um algoritmo que relaciona os dados e a incógnita ou de um processo que identifique automaticamente os dados com a conclusão e, portanto, deverá buscar, investigar, estabelecer relações e envolver suas emoções para enfrentar uma situação nova.

Analisando cada ponto, percebe-se que não há desejo de resolver um problema se este não gera no indivíduo algum tipo de provocação. Todos têm problemas, o tempo todo. Cada novo problema remete a algum tipo de estratégia que possa ser criada ou fazer uso de algum conhecimento já utilizado em alguma situação análoga.

Quem quer resolver o problema precisa escolher uma estratégia (inventada ou pré-concebida) para iniciar um plano de tentativas. Aquelas que não resolvem o problema são deixadas de lado, e são eliminadas do processo de resolução do problema; aquelas que resolvem o problema podem ser aperfeiçoadas, podendo ser úteis a outros problemas mais complexos.

Quando o professor afirma: “Ok, terminamos a explicação, agora vamos resolver alguns problemas para fixar o assunto...”, o que o professor está entendendo por problema? O que é problema para ele é problema para o aluno?

Para que se possa pensar em uma situação como situação-problema é preciso ter consciência dela, é preciso existir a necessidade de responder às questões provocadas por esta situação.

Para tentar ilustrar, tomemos este exemplo adaptado em Vianna (2002): durante a aula, um professor de Matemática está elaborando problemas com a turma, preocupado em contextualizá-los para que tenham significado junto aos alunos. Ele escolhe uma menina e menciona um mercado do bairro, elaborando o seguinte enunciado: “Melissa foi ao mercado para comprar uma dúzia de ovos. Na volta, encontrou-se com sua prima, com a qual ficou brincando. Durante a brincadeira, quebraram-se quatro ovos. Com quantos ovos inteiros Melissa chegou em casa?”. A turma permanece em completo silêncio... Até que timidamente uma garotinha do fundo da sala perguntou: “Professora, a Melissa apanhou quando chegou em casa?”.

Parece bastante evidente que o problema da aluna não é o problema do professor. O que é problema para um matemático pode não ser do interesse de um aluno de Matemática. Quando se pensa em contextos, em “Matemática contextualizada”, está-se muito mais inclinado a conceber que um problema só é problema para o aluno quando provoca, instiga, desafia, motiva.

Em sala de aula, cabe ao professor planejar e deflagrar as ações, de modo que situações como esta se tornem problemáticas para seus alunos. Nesse sentido, um problema que apresente apenas algoritmos pode vir a ser interessante para os alunos, desde que as circunstâncias sejam planejadas de modo a levar em conta sua relação com diferentes contextos.

Tendo em vista essas questões iniciais, pode-se tentar algumas aproximações para a questão: o que é problema em aulas de Matemática?

Delinear-se-á algumas aproximações acerca do que se entende ser e não ser “problema” a ser resolvido em aulas de Matemática. O que se entende como “não-problema” será chamado daqui em diante de exercício.

Problemas não são chamados de “problemas” se o resolvidor não necessita identificar situações matemáticas, ou seja, se ele pode resolver o “problema” utilizando um simples modelo de resolução de outro já resolvido. Tais problemas não passam de meros exercícios, já que podem ser numerosos, não necessitam da interpretação do enunciado, e envolvem um único conteúdo e uma única metodologia. Esses proliferam em muitos livros didáticos.

Pode-se entender a Resolução de Problemas como uma importante ferramenta para o aluno enfrentar problemas dos mais diversos, em que o não conhecimento de determinadas formalidades matemáticas pode atrapalhar suas ações cotidianas. Pode-se ainda pensar em ensinar Matemática a partir da resolução dos mais diversos problemas nas mais diferentes situações, encarando a resolução de problemas como objetivo no processo ensino-aprendizagem.

Segundo Vianna (2002), apresentar ideias matemáticas com significado é uma maneira de responder à pergunta: “para que serve isso?”. Na verdade, com as novas ideias sendo apresentadas “em ação”, dificilmente ocorrerá aos alunos

essa pergunta; ou seja, os problemas já são uma situação de “aplicação” do conteúdo matemático e mostram, de forma a não deixar dúvidas, “para que ele serve” (VIANNA, 2002).

Para Polya (1945), dentro da perspectiva de Resolução de Problemas, o que se exige é que, além de formular e pedir que se resolva determinado problema, deve-se:

- Questionar as respostas obtidas
- Questionar a própria questão original
- Questionar a estratégia (plano de resolução)
- Aproximar os resultados aos contextos em questão
- Verificar o sentido matemático da resposta

Daqui, captura-se a noção de que resolver um problema em aulas de Matemática não pressupõe apenas cumprir com a exigência de simples aplicação de técnicas ou fórmulas pré-estabelecidas e a consequente obtenção da resposta correta. Além disso, pressupõe desenvolver uma atitude investigativa em relação àquilo que está pronto, discutindo a solução do problema, os dados do problema e o próprio problema dado.

A postura investigativa faz diminuir o valor dado à simples obtenção da resposta correta. A ênfase será dada ao processo de resolução, permitindo o aparecimento de diferentes soluções, instigando a criatividade e possibilitando vários momentos de avaliação.

Vianna (2002, p. 402) diz que:

O aspecto subjetivo é muito forte na determinação do que venha a ser um problema, mas há outro lado: cada problema é colocado em uma situação determinada, há um lado objetivo que consiste exatamente nessa circunstância. Em sala de aula, cabe ao professor planejar e deflagrar as ações de modo que essas circunstâncias se tornem problemáticas para seus alunos. Nesse sentido, um problema de ‘prestações’ pode vir a ser

interessante para os mesmos alunos desde que as circunstâncias sejam planejadas de modo a levar em conta sua subjetividade.

Tendo em vista essas questões iniciais, tentam-se algumas aproximações para uma possível significação de problema em aulas de Matemática.

Problema em aulas de Matemática ocorre quando o estudante necessita identificar quais situações matemáticas podem dar solução ao mesmo, levando em consideração a ausência de palavras-chave na identificação de tais situações, como por exemplo: “juntar, repartir, diminuir, etc”.

Um problema em aulas de Matemática pode ter enunciado longo (várias linhas) ou enunciado curto (poucas linhas); o cerne do mesmo é o questionamento necessário às etapas resolutivas do problema acerca das situações matemáticas.

Um aspecto essencial a ser observado para que um enunciado de problema seja um problema para o aluno, é a necessidade de se conhecer alguns conceitos para iniciá-lo. Por exemplo, pedir para uma criança de 1ª série calcular a área e o perímetro de um retângulo não é um problema, porque ela não tem a menor ideia do que seja área ou do que seja perímetro.

Vianna (2002, p. 407) diz que problema corresponde a:

[...] tudo o que, de uma maneira ou de outra, implica da parte do aluno a construção de uma resposta ou de uma ação que produza certo efeito. A noção de problema não tem sentido se o aluno não puder aplicar um sistema de respostas inteiramente constituído.

Um bom problema depende de muitos fatores. O fato de o resolvidor já conhecer os procedimentos para encontrar a resposta torna-o um simples exercício.

Questões rotineiras (consideradas como àquelas integradas à rotina de exercícios de repetição e imitação, geralmente de aplicação direta de algum algoritmo ou regra pré-estabelecida) não podem ser consideradas como problemas. Tais questões são meros exercícios. Além disso, existe um aspecto muito importante que é comum a todas as aproximações citadas, e que nada tem

a ver com o conteúdo de uma determinada disciplina. Trata-se do desejo: o aluno precisa ter interesse, precisa estar seduzido pela questão, precisa ter necessidade de chegar a uma resposta. De alguma forma, o problema deve lhe parecer familiar ou ao menos o desafiar.

### **3.2 Aportes Teóricos sobre Heurística em um contexto de Resolução de Problemas**

Conforme nos mostra a História, o homem buscou de todas as maneiras modos de facilitar seus processos de contagem e de produção. Até que ponto os problemas e o progresso da humanidade podem ser atribuídos a essa capacidade do homem de resolver problemas das mais diversas situações diárias? Muitas vezes, em tais situações, a solução não é imediata. O que faz o homem resolver problemas é a necessidade.

Existe um senso comum de que todos têm problemas. Deste modo, todos os seres vivos têm problemas, pois problemas surgem como um caminho à evolução. Todos os seres vivos enfrentam um problema comum: a necessidade de encontrar, a todo e qualquer custo, uma maneira de sobreviver. Na tentativa natural de solucionar esses problemas, por exemplo, movimentam-se as folhas de uma planta pela necessidade de receber os raios solares. Já na tentativa estratégica para resolver um problema, alguém pode mudar a posição de um guarda-sol na praia, para evitar que os raios causem queimaduras e danifiquem a pele.

Problemas, sejam eles práticos ou teóricos, mesmo tendo muita experiência, pode-se chegar a soluções mal sucedidas. Então, num processo de descobrir pelos seus próprios meios, o homem se sente provocado. Por necessidade, cria, elabora estratégias e planos de ação. Para resolver seus problemas, descobre um método que até então era desconhecido, útil à situação-problema.

A esse esquema de descobrir pelos seus próprios meios dá-se o nome de *atividade heurística*, conforme salienta Puchkin (1969, p. 72):

Acontece que, na vida cotidiana, (...), frequentemente surgem diante do homem situações que geram conflitos entre as circunstâncias e as exigências do exercício de uma atividade. Precisa o homem executar uma série de ações e solucionar este ou aquele problema. Contudo, as condições reinantes não lhe propiciam meios para solucionar esses problemas. E mesmo todo o seu arsenal de experiências passadas não lhe apresenta qualquer esquema completo adequado às condições emergentes. A fim de descobrir uma saída para a situação, deve o homem criar uma nova estratégia de ação, isto é, concretizar um ato de criação. Contingência como esta é, normalmente, denominada um problema ou uma situação problemática, ao passo que o processo psíquico que, ao auxiliar sua solução elabora uma nova estratégia que se mostra como algo inédito é designado como pensamento criador ou, para usarmos terminologia que nos vem de Arquimedes, atividade heurística.

Lexicalmente, o termo heurística, em um contexto pedagógico, é definido como: “método educacional que consiste em fazer descobrir pelo aluno o que se lhe quer ensinar” (Houaiss, 2001, p.141). Percebe-se, portanto, que falar em heurística na resolução de problemas é falar sobre “métodos e regras que conduzem à descoberta, inovação, investigação e resolução de problemas” (VILANOVA, 2000, p. 27, tradução nossa).

De acordo com as considerações feitas em relação ao processo da atividade heurística, justifica-se a possível inserção desta relacionada à Resolução de Problemas e as contribuições favoráveis para o ensino da Matemática, pois, como salienta Polya (1945, p. 38):

A Heurística moderna esforça-se por compreender o processo de resolução de problemas, especialmente as operações mentais, tipicamente úteis nesse processo. Dispõe de várias fontes de informação, nenhuma das quais deve ser desprezada. Um estudo sério da heurística deve levar em conta tanto as suas bases lógicas quanto as psicológicas, não deveria negligenciar aquilo que autores antigos como Pappus, Descartes, Leibnitz e Bolzano disseram sobre o assunto, mas muito menos deveria negligenciar a experiência imparcial. A experiência na resolução de problemas e a experiência na observação dessa atividade por parte de outros devem ser a base em que a heurística é construída. Nesse estudo, não deveríamos descurar de nenhum tipo de problema, e deveríamos buscar os aspectos comuns na maneira de tratar de problemas de toda a sorte: deveríamos visar aos aspectos gerais, independentemente do assunto do problema. O estudo da heurística tem

objetivos “práticos”: uma melhor compreensão das operações mentais tipicamente úteis na resolução de problemas poderia exercer uma influência benéfica sobre o ensino, especialmente sobre o ensino da Matemática. (tradução nossa)

Conforme indicou Polya, muitos matemáticos se propuseram a refletir sobre a Resolução de Problemas com o enfoque na Heurística. Pappus, matemático grego que viveu por volta do ano 300, escreveu um livro cujo título pode ser traduzido como *O Tesouro da Análise* (Arte de Resolver Problemas) ou *Heurística*, onde procurava sistematizar um método para resolver problemas.

As mais famosas tentativas de sistematização da Heurística foram feitas pelos filósofos e matemáticos Descartes, Leibnitz e Bolzano. Não se pode deixar de mencionar também os trabalhos de Poincaré, notável matemático francês. Segundo Puchkin (1969), Poincaré apresentou, em suas Memórias sobre as *Funções de Fuchs*, uma das mais expressivas descrições da atividade heurística.

### **3.3 Resolução de Problemas: Primeiras Aproximações**

#### **3.3.1 Situando historicamente a Resolução de Problemas em um contexto escolar**

Apresenta-se a seguir “historicamente” as modificações mais significativas sobre a representatividade da Resolução de Problemas no contexto escolar, partindo da década de 70, passando pelas décadas de 80 e 90. Temos que salientar que breve sumário não representa toda a complexidade dos fatos ocorridos nestas décadas, mas apenas constitui como uma visão geral segundo Medeiros Júnior (2007).

O objetivo deste breve descritivo é conectar alguns possíveis desafios enfrentados pelos sujeitos participantes do segundo ciclo de coleta de dados desta pesquisa durante sua formação escolar (na maioria das vezes, ocorrida entre estas três décadas), auxiliando-nos a ilustrar mais à frente, os argumentos utilizados na fase de análise dos dados.

## Década de 70

Conforme descrito por Medeiros Júnior (2007), os educadores matemáticos iniciam uma mudança de direção em suas pesquisas, no sentido de dar mais ênfase aos processos de resolução utilizados por seus alunos na solução de um problema. Esse movimento ficou conhecido como *back to basics*, tendo, no entanto, pouca influência na prática de ensino da Educação Matemática.

## Década de 80

O *National Council of Teachers of Mathematics* (NCTM) (Conselho Nacional de Professores de Matemática dos Estados Unidos – tradução nossa) elabora o documento *An Agenda for Action*, com diretrizes para o progresso da Matemática nos anos 80, e mais tarde o *Professional Standards for Teaching Mathematics*, com normas diretivas para o ensino de Matemática, enfatizando "a Matemática como resolução de problemas, raciocínio e comunicação".

O NCTM, assim como fez os Parâmetros Curriculares Nacionais de Matemática no Brasil (PCN), concebe que se deve ensinar Matemática por meio de resolução de problemas, enfatizando essa estratégia como metodologia de ensino, como modo de se ensinar Matemática de forma criativa. Contudo, ensinar Matemática vai além desta concepção.

Quando o NCTM iniciou a publicação, em três volumes, dos chamados Padrões para a Educação Matemática (ou *Standards*), houve modificações no que se entendia por ensinar Matemática para além do que o Movimento da Matemática Moderna (M.M.M.) propunha. O M.M.M. surgiu em 1959, na Conferência Internacional em Royalmont, com características de forte ligação com a teoria dos conjuntos, o alto nível de generalidade e o maior rigor lógico. Como se era de esperar, concepções e crenças foram colocadas em debate, mudanças radicais ocorreram, em duplo sentido.

Segundo Medeiros Júnior (2007), no sentido favorável à Educação Matemática, destaque aos inúmeros exemplos práticos de como aplicar as teorias descritas nos Padrões e estudos de casos presentes no material como um todo. Os episódios relatados sobre aulas de Matemática por professores em diferentes níveis contribuem para uma leitura mais informativa. Existem ainda protocolos sobre provas com materiais manipuláveis, calculadoras gráficas e jogos, que, de certa forma, contribuem para uma reflexão sobre a própria prática.

Em sentido contrário, chama a atenção a pouca Matemática presente nos Padrões. O aspecto mais notável dos Padrões é a ausência da Matemática como um sistema. Encontram-se vários episódios matemáticos bem conhecidos e problemas úteis, nada cotidianos, mas todos fora do contexto natural. Por exemplo, o teorema de Pitágoras é mencionado nos Padrões junto com uma figura bem conhecida que pode ser usada para demonstrá-lo; no entanto, propõe-se o uso da figura apenas para “descobrir a relação por meio de exploração” e ainda “o professor ajuda os alunos a ampliar a compreensão do teorema” (NCTM, 1994, tradução nossa).

Para Medeiros Júnior (2007) há uma abordagem que indica o que deve ser feito sem mostrar matematicamente como fazer. Isto também acontece no PCN de Matemática (Brasil, 1999). A possibilidade de se demonstrar e discutir as diferentes maneiras de se obter esse importante teorema não são mencionados em lugar algum do documento.

Fica a impressão de que o professor ao trabalhar com a repetição de padrões e analogias estará trabalhando o fundamental da Matemática. Na Rússia, por exemplo, cálculos mentais e com papel e lápis foram sempre recomendados para todas as idades e considerados essenciais para a compreensão das operações.

Os episódios relatados nos Padrões parecem excluir problemas verbais (aqueles com um predomínio de termos da língua materna nos enunciados e podem, além disso, referir-se ou não a contextos reais) em prol dos “problemas do mundo real”.

Os livros de problemas russos (por exemplo, a coleção russa da editora *MIR: Lecciones populares de matemáticas*) estão repletos, principalmente, de problemas verbais. A característica desses problemas é o uso de palavras que não são termos matemáticos, como carros e trens; distância, tempo e velocidade; barcos e correntezas; aviões e vento; caixas, latas e bolas; canos, bombas e piscinas; massa e misturas; horas, minutos e hora do dia; ponteiros do relógio, anos e idade; dinheiro, preço, juros e descontos etc.

Nos problemas de Matemática russos, a insistência de problemas verbais na Educação Matemática sempre foi normal, mas nos Estados Unidos foi muito diferente. Apesar de educadores americanos referenciarem George Polya na maioria de seus trabalhos, muitas vezes ignoram suas opiniões. Polya (1981, p. 123) escreveu:

Por que problemas verbais? Espero chocar algumas pessoas ao afirmar que, por si só, a tarefa mais importante da instrução nas escolas médias é o ensino da montagem de equações para resolver problemas verbais. Existe um argumento forte a favor dessa opinião. Ao resolver problemas verbais, armando equações, o estudante traduz uma situação real em termos matemáticos; ele tem uma oportunidade de vivenciar que conceitos matemáticos podem estar relacionados com realidades, mas que tais relações precisam ser trabalhadas cuidadosamente. (tradução nossa)

Outra consideração a ser feita, e que está muito presente nos PCN de Matemática (Brasil, 1999) e que, por conseguinte, está presente na década de 90, é o clichê da contextualização. Observa-se que nestes documentos as citações de um problema do mundo real são atreladas a nomes de marcas registradas.

Alguns livros texto incluíam problemas como este: “O biscoito Oreo é o mais vendido dos biscoitos em embalagens... o diâmetro de um biscoito Oreo é 1,75 polegadas. Expresse o diâmetro do biscoito Oreo como fração na sua forma mais simples”. Um típico exemplo de problema que pode ser encontrado em livros didáticos brasileiros e que não contribui com a elevação do potencial criativo do aluno.

## Década de 90

De acordo com Medeiros Júnior (2007), em 1989, o NCTM publicou os primeiros Padrões para o currículo de Matemática, o que, mais tarde, em 1997, levou à criação dos PCN de Matemática, no Brasil, para as turmas de 1ª a 4ª e de 5ª a 8ª séries do Ensino Fundamental e para os 1º, 2º e 3º anos do Ensino Médio.

Os PCN de Matemática (Brasil, 1999) apontam para algumas reflexões no campo da Educação Matemática, por exemplo, a formação do cidadão e o fato de que o professor não pode mais restringir a transmissão do conhecimento sem relacionar os fatos de sua prática escolar com os acontecimentos globais, uma vez que “nas sociedades modernas, uma boa parte da informação é veiculada em linguagem matemática”, e porque “vivemos num mundo de taxas percentuais, coeficientes multiplicativos, diagramas, gráficos e verdades estatísticas” (IMENES e LELLIS, 1994).

Segundo os PCN de Matemática (Brasil, 1999) a resolução de problemas na Matemática escolar deve ser entendida como um “recurso” ou “ponto de partida” para a atividade matemática, mas o que se tem praticado nos diferentes níveis de ensino é uma Matemática “formalista”, axiomática (euclidiana), sintética, que privilegia excessivamente os processos de demonstrações e a repetição de conceitos definidos *a priori*, onde a necessidade do pré-requisito está fortemente ligada a axiomas e signos de um mundo distante da realidade escolar (BATISTI, 1999).

### 3.3.2 A Heurística em Arquimedes, Pappus, Descartes e Polya

Nesta subseção, apresentamos um relato dos indícios heurísticos de quatro fundadores da chamada *heurística* na Resolução de Problemas: Arquimedes, Pappus, Descartes e Polya. Este último é responsável por organizar didaticamente os princípios da heurística e terá, nesta tese, maior aprofundamento.

Pretende-se apresentar alguns aportes sobre a heurística, presentes nas obras *Regras para a Direção do Espírito e Discurso do Método*, de Descartes. Procurou-se estabelecer relação com a sistematização da atividade heurística, intuição e dedução propostas por George Polya nas obras *A arte de Resolver Problemas e Matemática e Raciocínio Plausível*. Especialmente, devido ao seu método de análise em processos heurísticos, algumas ideias de Descartes serão elucidadas.

### 3.3.2.1 Heurística em Arquimedes

Arquimedes (287 a.C. - 212 a.C.) foi um matemático e inventor grego, nascido na cidade-estado grega de Siracusa, na ilha da Sicília. Foi um dos mais importantes matemáticos da Antiguidade. Criou notáveis aparatos bélicos para a 2ª. Guerra Púnica, contra o poderoso exército e marinha romanos, comandados pelo Cônsul Marcelo, além de demonstrar um método para calcular, com aproximação tão grande quanto se queira, do número  $\pi$  (3,1415926535...; razão entre o perímetro de uma circunferência e seu diâmetro).

Em Física, no seu *Tratado dos Corpos Flutuantes*, estabeleceu as leis fundamentais da estática e da hidrostática. Um dos princípios fundamentais da hidrostática é assim enunciado: "todo corpo mergulhado total ou parcialmente em um fluido sofre uma impulsão vertical, dirigido de baixo para cima, igual ao peso do volume do fluido deslocado, e aplicado no centro de impulsão." Isto quer dizer que, para o objeto flutuar, o peso da água deslocada pelo objeto tem de ser maior que o próprio peso do objeto.

Rege a lenda que certa vez, Hierão, rei de Siracusa, no século III a.C., havia encomendado uma coroa de ouro para homenagear uma divindade que supostamente o protegera em suas conquistas, mas foi levantada a acusação de que o ourives o enganara, misturando o ouro maciço com prata em sua confecção. Para descobrir, sem danificar o objeto, se o seu interior continha uma parte feita de prata, Hierão pediu a ajuda de Arquimedes.

Arquimedes se deparou com este problema e buscou uma solução coerente e de simples demonstração, a qual, dizem, lhe ocorreu durante um

banho. A lenda afirma que Arquimedes teria notado que uma quantidade de água correspondente ao seu próprio volume transbordava da banheira quando ele entrava nela e que, utilizando um método semelhante, poderia comparar o volume da coroa com os volumes de iguais pesos de prata e ouro: bastava colocá-los em um recipiente cheio de água, e medir a quantidade de líquido derramado. Feliz com essa fantástica descoberta, Arquimedes teria saído à rua despido, gritando: *Eureka! Eureka!* (Descobri! Descobri!). Assim, podemos aproximar o termo Heurística de *Eureka*, ou seja, uma grande descoberta.

Aportes da Heurística são encontrados no trabalho *O Método*, de Arquimedes. *O Método* encontra-se na forma de uma carta endereçada a Erastóstenes e é importante devido às informações que fornece sobre o método que Arquimedes usava para descobrir muitos de seus teoremas. O notável matemático o usava de maneira experimental para descobrir resultados que ele então tratava de colocar em termos rigorosos mediante o Método de Exaustão.

Balieiro Filho (2004, p. 40) nos relata que:

Com relação ao método de exaustão, convém salientar, segundo Babini, que não se trata de um método para se realizar descobertas, mas de um método para se fazer demonstrações, isto é, é necessário ter-se um conhecimento prévio do resultado que se quer demonstrar para que seja possível realizar uma demonstração rigorosa.

Partindo do pressuposto de que Arquimedes utilizava o Método da Exaustão ao apresentar demonstrações elegantes e rigorosas para suas descobertas matemáticas e não para ensinar um método de descobertas que conduzisse a atividade heurística, considera-se o modo como Balieiro Filho (2004, p. 40) define o Método de Arquimedes:

A primeira observação importante que se formula é que não se trata de um método de descobrimento, mas de demonstração, isto é, que supõe conhecido de alguma maneira o resultado, e oferece um procedimento rigoroso para demonstrá-lo. Além disso, observamos como, já na época de Eudoxo, a Matemática refletia sua característica fundamental de ter por acento o processo dedutivo, a demonstração, e não o resultado. Conhecido, pois, de antemão, o resultado, a demonstração pelo método de Eudoxo de que, por exemplo, uma certa figura A é equivalente a uma figura conhecida B,

consiste numa dupla redução ao absurdo provando que os supostos de A maior ou menor que B conduzem a contradições, de maneira que não fica outra alternativa senão a de que A seja equivalente a B. E é nessa demonstração que joga seu papel o postulado, já que a demonstração exige que se possa decompor a figura em partes tais que uma delas seja inferior a uma figura dada, e isso se obtém precisamente em virtude do postulado. Essa decomposição da figura em partes cada vez menores foi a causa pela qual um matemático renascentista deu ao método o nome de ‘método de exaustão’, embora na verdade tal decomposição não ‘esgote’ a figura, mas que só chega ao ponto em que certa figura é menor que uma figura dada.

O método heurístico de Arquimedes é relevante para a presente pesquisa porque pretendemos observar a heurística na resolução de problemas em Matemática, e potencialmente, no ambiente de trabalho dos programadores.

### 3.3.2.2 Heurística em Pappus

Pappus, grande matemático grego (300 d.C), organizou uma obra importante, composta originalmente por oito livros, chamada *A Coleção Matemática*, onde procurava sistematizar um método para resolver problemas. No livro VII de sua Coleção, Pappus descreve um ramo de estudo que ele chamou de *Analyomenos*, que pode ser traduzido como *Tesouro da Análise*, ou *Arte de Resolver Problemas*, ou mesmo, *Heurística*.

Considerando o intuito deste item, dentre os livros de *A Coleção Matemática*, foca-se a análise no livro VII, por seu valor do ponto de vista histórico, e, em particular, por abordar e conceituar os aspectos referentes à *análise e síntese*, que fornecem subsídios à atividade heurística.

Dentre os trabalhos a que se recorre para o estudo da Heurística, George Polya atribui um sentido moderno a ela, chamando-a de “heurística moderna” e se refere a esta obra de Pappus.

Segundo Balieiro Filho (2004, p. 67):

Pappus descreve em detalhes o método analítico dos antigos geômetras gregos na demonstração de teoremas ou na construção de figuras

geométricas. Esse procedimento consistia em um duplo movimento: análise, na qual se buscavam os antecedentes das proposições a serem provadas ou as condições que tornassem possíveis a construção de figuras geométricas, e a síntese, na qual, a partir das condições descobertas na análise, apresentava-se ou a prova do teorema na sequência lógica usual ou a construção efetiva da figura geométrica. A análise se subdividia em transformação (busca das condições para a solução do problema) e resolução (legitimação das condições descobertas). A síntese se subdividia, por sua vez, em construção (dos dados do problema) e prova.

Com relação à solução de um problema para Pappus, é bastante comum poder-se “*adivinhá-la*”, utilizando intuições, inferências, induções e baseando-se em analogias com outros problemas resolvidos (Polya, 1995), e não se pode negar que, talvez por esses procedimentos, foram alcançados resultados importantes, como se verifica no desenvolvimento das ideias matemáticas; mas, apesar de parecer uma prática comum em problemas de Matemática, tal “adivinhação” não é um método científico propriamente dito.

Entrando no âmbito da demonstração e da análise é interessante observar o que dizem Pappus e Polya, segundo Balieiro Filho (2004, p. 68):

Duplo é o gênero da análise, um a pesquisa do verdadeiro, o qual é chamado teórico, o outro capaz de dizer o que foi proposto, o qual é chamado problemático. Enquanto que, no gênero teórico, tendo estabelecido o que é procurado como existente e verdadeiro, em seguida, por meio das consequências sucessivas como verdadeiras, e como existem segundo a hipótese, tendo avançado até algo admitido, caso, por um lado, fosse verdadeiro aquilo admitido, será verdadeiro também o procurado, e a demonstração é uma inversão da análise; caso, por outro lado, encontramos falso o admitido, falso será também o procurado. No gênero problemático, tendo estabelecido o que foi proposto como conhecido, em seguida, por meio das consequências sucessivas, como verdadeiras, tendo avançado até algo admitido, caso, por um lado, o admitido seja possível é obtível, o que os matemáticos chamam dados, possível também será o proposto, e, de novo, a demonstração é uma inversão à análise; caso, por outro lado, encontramos impossível o admitido, impossível será também o problema.

A análise, proposta por Pappus, consiste em resolver um problema admitindo o resultado que se quer demonstrar como verdadeiro, buscando, em

seguida, um antecedente do qual seja possível deduzir o resultado que se quer demonstrar e que foi admitido como verdadeiro. Repetindo esse processo de regressão (ou raciocínio regressivo) sucessivamente busca-se chegar a algum resultado que já se conhece ou admite-se como válido.

Polya (1995, p. 98), partindo dos ensinamentos de Pappus, afirmou:

Na análise, começamos por aquilo de que se precisa e que admitimos como certo e extraímos consequências disso e consequência das consequências até chegarmos a um ponto que podemos usar como de partida da síntese. Porque na análise admitimos que o que precisa ser feito já o foi (o que se procura já foi encontrado, o que se tem a demonstrar é verdadeiro). Indagamos de qual antecedente poderá ser deduzido o resultado desejado; em seguida, indagamos de novo qual poderá ser o antecedente desse antecedente e assim por diante, até chegarmos finalmente a algo que já conhecemos ou que admitimos como verdadeiro. A este procedimento chamamos análise, ou regressão ou raciocínio regressivo.

A etapa conclusiva é a síntese e consiste em chegar a um resultado, isto é um raciocínio progressivo; em seguida, é necessário mostrar que as condições primitivamente postas são também satisfeitas.

Pappus utilizava os procedimentos heurísticos para solucionar seus problemas matemáticos, criando modelos matemáticos que utilizava a *análise* para encontrar a solução de um problema ou a demonstração de um teorema e, em seguida, a *síntese* para expor o que se encontrou para solucionar o problema ou a demonstração de um teorema.

Em relação à *síntese*, Polya (1995, p. 113) concluiu que:

...na síntese, invertendo o processo, partimos do último ponto a que chegamos à análise, daquilo que já sabemos ou admitimos como verdadeiro. Disso deduzimos o que o procedeu na análise e continuamos a fazer deduções até que, percorrendo o mesmo caminho no outro sentido, conseguimos finalmente chegar aonde queríamos. A este procedimento chamamos síntese, ou resolução construtiva ou raciocínio progressivo.

A *análise* e a *síntese* foram os procedimentos utilizados por Pappus para a solução de problemas geométricos. Esses procedimentos fazem uso da atividade

heurística no raciocínio regressivo, porém, ao partir de soluções prontas e verdadeiras *a priori*, perde-se o potencial criativo próprio dos programadores quando resolvem problemas da forma que se pretende observar nesta pesquisa.

### 3.3.2.3 Heurística em Descartes

René Descartes nasceu na França em 31 de março de 1596, na pequena cidade de La Haye (agora “Descartes”). Foi criado por sua avó materna e, quando tinha oito anos, mandaram-no para o recém fundado colégio jesuíta de La Flèche, em Anjou, onde permaneceu como aluno interno por nove anos.

Deixou a escola dos jesuítas aos dezesseis anos rumo a Paris. Era especialmente feliz no jogo, uma vez que baseava seus palpites mais em princípios matemáticos do que em leis do azar.

Em linhas gerais, o sistema definido por Descartes está exposto em uma obra inacabada, escrita no final da segunda década do século XVII, as *Regulae ad directionem ingenii* (Regras para a direção da inteligência natural). Descartes define ali o “conhecimento” (*scientia*).

A regra II das *Regulae* define *scientia* como a “cognição certa e evidente”, baseada na apreensão mental direta das verdades imediatamente evidentes (que Descartes denomina “intuição”), aconselha-nos a rejeitar toda a crença que é apenas provável e a decidirmo-nos “a crer somente no que é perfeitamente conhecido e do que não se pode duvidar” (ADAM e TANNERY, 1964-76 apud COTTINGHAM, 1995).

Esboça, além disso, segundo Adam e Tannery (1964-76) apud Cottingham (1995, p. 52), o plano para uma “ciência universal”, que envolveria todos os ramos do conhecimento humano:

Percebi que a Matemática interessa-se exclusivamente por questões de ordem ou medida. O que é irrelevante se a medida em questão envolve números, forma, estrelas, sons ou qualquer outro objeto; isso me fez entender que deve haver, necessariamente, uma ciência geral que explique todos os

pontos passíveis de serem levantados em relação à ordem e à medida. Qualquer que seja o assunto.

Neste mesmo período Descartes, segundo Adam e Tannery (1964-76) apud Cottingham (1995, p. 37), munido de intenso ceticismo, afirma em suas *Meditações*:

Acima de tudo, devemos duvidar de tudo. Como eu desejasse entregar-me inteiramente à procura da verdade, cuidei que me fosse necessário (...) rejeitar como absolutamente falso o que quer que possa conter, ao meu juízo, a menor parcela de incerteza (...) E já que todos os pensamentos e imaginações que nos acodem quando acordados são os mesmos que podem acudir-nos enquanto dormimos, sem que nenhum deles seja, ao mesmo tempo, verdadeiro, determinei estabelecer que tudo quanto jamais entrará em meu espírito não era mais verdadeiro que as ilusões dos meus sonhos.

O “sonhar”, descrito por Descartes, leva-o à sua primeira realidade: Cogito, ergo sum (Penso, logo existo).

Em meados de 1620, Descartes começou a escrever um tratado sobre regras para a direção do espírito; para descobrir a ciência universal. Segundo ele, teríamos inicialmente de adotar um método adequado de reflexão, que consistia na adoção de duas regras de operação mental: intuição e dedução.

Definia intuição como “a concepção inequívoca de um espírito claro e formado exclusivamente pela luz da razão”, e dedução como a “necessária inferência a partir de outros fatos tidos como certos” (STRATHERN, 1997).

O celebrado método de Descartes - que veio a ser conhecido como método cartesiano - baseava-se na aplicação correta dessas duas regras de pensamento.

De acordo com as biografias lidas no decorrer desta pesquisa (Strathern, 1997 e Descartes, 2005) as coordenadas cartesianas foram denominadas por Leibniz, retiradas do tratado escrito por Descartes, *Tratado sobre o Universo*, onde lançou os fundamentos da Geometria Analítica.

Em 1628, Descartes trabalhou em definitivo na obra *Règles pour la Direction de L'Esprit (Regras para a Direção do Espírito)*, em que pretendia

apresentar um método universal para a resolução de problemas. Esta obra ficou incompleta. Fragmentos dela apareceram depois, entre 1633 e 1637, no *Discours de la Méthode de Bien Conduire la Raison et Chercher la Vérité dans les Sciences* (*Discurso sobre o método para raciocinar bem e procurar a verdade nas ciências*).

De acordo com Boyer (1996, p. 75), Descartes define seu método geral lançando mão das operações algébricas e a resolução de equações quadráticas por meio de interpretações geométricas. Para tal, enuncia:

Se, pois, queremos resolver qualquer problema, primeiro supomos a solução efetuada e damos nomes a todos os segmentos que parecem necessários à construção – aos que são desconhecidos e aos que são conhecidos. Então, sem fazer distinção entre segmentos conhecidos e desconhecidos, devemos esclarecer a dificuldade de modo que mostre mais naturalmente as relações entre esses segmentos, até conseguirmos exprimir uma mesma quantidade de dois modos. Isso constituirá uma equação (numa única incógnita), pois os termos de uma dessas expressões são juntas iguais aos termos da outra.

Descartes vê o processo de resolução de problemas em três fases:

1. Reduzir todo problema algébrico a um problema contendo apenas equações
2. Reduzir todo problema matemático a um problema algébrico
3. Reduzir qualquer problema a um problema matemático

Observa-se que Descartes objetiva reduzir todo problema que existe no mundo a um problema matemático; mais que isso, a ideia de Descartes era completar o projeto de resolver problemas citado anteriormente e ainda usufruir de seus benefícios.

Segundo Balieiro Filho (2004, p. 79):

Descartes filósofo racionalista do século XVII, revelou em suas obras que a intuição específica da percepção criativa não tem base lógica no raciocínio, mas numa peculiar e súbita (insight) visão intelectual (...) considerava a concepção intuitiva do real uma forma superior de criação. Nela, a mente

raciocina e, simultaneamente, medita nas três dimensões conhecidas do conhecimento: profundidade, abrangência e atualidade

Não obstante, Descartes (2005) destaca algumas ideias de valor e relevância relacionadas ao ensino e que podem ser aplicadas à resolução de problemas. Como exemplo, podemos citar as regras III, IV, V, VI e VII, que enfoca:

- Regra III: “No que tange aos objetos considerados, não é o que pensa outrem ou o que nós mesmos conjecturamos que se deve investigar, mas o que podemos ver por intuição com clareza e evidência, ou o que podemos deduzir com certeza: não é de outro modo, de fato, que se adquire a ciência”. *Revela-se a importância da argumentação, intuição e da dedução ao invés do uso da autoridade.*
- Regra IV: “O método é necessário para a busca da verdade”. *Ressalta-se a importância da sistematização e do formalismo.*
- Regra V: “O método todo consiste na ordem e na organização dos objetos sobre os quais se deve fazer incidir a penetração da inteligência para descobrir alguma verdade. Nós lhe ficaremos fiéis se reduzirmos gradualmente as proposições complicadas e obscuras a proposições mais simples e, em seguida, se, partindo da intuição daquelas que são as mais simples de todas, procurarmos elevar-nos pelas mesmas etapas ao conhecimento de todas as outras”. *Descartes critica veementemente aqueles que examinam os problemas com tal falta de ordem que lhe parecem querer atingir com um salto, da parte de baixo de um edifício, o topo, desprezando os degraus da escada.*
- Regra VI: “Para distinguir as coisas mais simples daquelas que são complicadas e pôr ordem em sua investigação, cumpre, em cada série de coisas em que deduzimos diretamente algumas verdades umas das outras, observar o que é mais simples e como se distancia, mais ou menos, ou igualmente, o resto”. *Alerta-se que, para se constituir uma definição formal, deve-se partir do problema mais simples para o mais elaborado.*

Descartes (2005) usa como exemplo uma relação entre números, “o número 6 é o dobro do número 3, procuraria em seguida o dobro do número 6, ou seja, 12; em seguida procuraria, igualmente, o dobro do último número, ou seja, 24, e também o dobro deste, ou seja, 48, etc. Daí, deduziria que a relação entre 3 e 6 é igual àquela entre 6 e 12, assim como entre 12 e 24, etc, e que, por conseguinte, os números 3, 6, 12, 24, 48, ... são continuamente proporcionais.” A etapa final seria definir a sequência anterior como sendo uma progressão geométrica de razão 2.

- Regra VII: “Para o aperfeiçoamento da ciência, é preciso passar em revista, uma por uma, todas as coisas que se relacionam com a nossa meta por um movimento de pensamento contínuo e sem nenhuma interrupção, e é preciso organizar tais coisas numa enumeração suficiente e metódica”, mostrando que é importante mantermos controle sobre o que estamos fazendo, sob pena de se perder em um trabalho infrutífero.

É importante observar Descartes, pois suas sugestões para o ensino e a resolução de problemas antecipam ideias de um importante pesquisador da Resolução de Problemas, George Polya, o qual nos basearemos neste estudo.

#### **3.3.2.4 Heurística em Polya**

George Polya (1897-1985) nasceu em Budapeste, capital da Hungria. Viveu boa parte de sua vida nos Estados Unidos, onde fez seus estudos e pesquisas. Inicialmente, Polya ingressou na faculdade de Direito, provavelmente seguindo os caminhos de seu pai, mas abandonou o curso, passando para o estudo de Línguas e Literatura. Mais tarde focou seus estudos em Latim, Filosofia, Física e, finalmente, optou pela Matemática, em meados de 1912. Foi professor em Zurique, de 1914 a 1940, e depois em Stanford, Estados Unidos, onde se aposentou em 1953.

Alguns de seus trabalhos, como a classificação dos 17 grupos de simetria bidimensional, acabaram por inspirar o pintor M. S. Escher. Em 1925, Polya escreveu juntamente com seu compatriota, Gabor Szegő, um trabalho intitulado

*Aufgaben und Lehrsätze aus der Analysis*, depois em 1972 traduzido para o inglês com o título *Problems and Theorems in Analysis*. Neste trabalho, apresentado em dois volumes, os autores mostram como o ensino da Análise Matemática pode ser gradativamente desenvolvido, dos fundamentos até algumas fronteiras do conhecimento, por meio de uma acertada sequência de exercícios e problemas, alguns dotados de apurada estética do conhecimento.

Polya tratou de apresentar problemas matemáticos de forma intuitiva, fazendo uso da arte/técnica com que os conceitos matemáticos eram formados, sendo responsável por organizar didaticamente os princípios da Heurística. Para isso, elaborou um pequeno dicionário de Heurística, presente na parte três do livro *How to Solve It* (traduzido para o português como *A Arte de Resolver Problemas*).

O modo como é difundido o trabalho de Polya, em seu livro *A Arte de Resolver Problemas*, parece um tanto simplista. Entende-se que os passos (quais sejam: primeiro, é preciso compreender o problema; segundo, procure encontrar a conexão entre os dados e a incógnita – é preciso chegar afinal a um plano para a resolução; terceiro, execute seu plano; quarto, examine a solução obtida), aplicados à resolução de problemas da Matemática, podem ser suficientes, reduzindo toda a sua teoria a uma “receita” para que se aprenda a ser um bom “resolvedor de problemas”. Para Polya (1995, p. 5):

Uma grande descoberta resolve um grande problema, mas há sempre uma pitada de descoberta na resolução de qualquer problema. O problema pode ser modesto, mas se ele desafiar a curiosidade e puser em jogo as faculdades inventivas, quem o resolver, por seus próprios meios, experimentará a tensão e gozará o triunfo da descoberta.

É de se esperar que o modo com que Polya abordava, de forma intuitiva e analítica, os problemas de Matemática, por meio de uma regra prática que fosse capaz de resolver qualquer tipo de problema, fosse bem antigo.

Polya dizia que o ensino da Matemática deve ser ativo e que não se deve suprimir as atividades informais de produzir e extrair conceitos matemáticos do

mundo que nos rodeia, o que de certa forma está conectado intimamente aos objetivos deste estudo.

Polya obteve destaque com seus trabalhos ao circunstanciar Matemática como Resolução de Problemas, colocando-a como o foco principal do saber matemático. Para ele, a gênese dos conceitos matemáticos está na ação didática proveniente da resolução de problemas. Mas o que é ser bom “resolvedor de problemas”? Como se adquire o que Polya chamou de *know-how* em Matemática?

*Know-how* é aqui entendido como a habilidade para resolver problemas, não apenas os que são rotineiros, mas, também, aqueles que exigem algum grau de independência, julgamento, originalidade e criatividade.

Polya percebe a Matemática como uma disciplina dependente da intuição, da imaginação e da descoberta, defendendo que se deve imaginar a ideia da prova de um teorema antes de prová-lo.

Há muito tempo, as propostas de ensino da Matemática que envolvem resolução de problemas vêm sendo discutidas e avaliadas. As perguntas são as mais diversas: até que ponto a resolução de problemas, da forma como é entendida e aplicada, pode vir a contribuir para a solução dos problemas no ensino de Matemática? A resolução de problemas deve ser entendida como recurso ou ponto de partida? Ou ainda, deve-se pensar a resolução de problemas como um objetivo ou um processo?

Polya diz que o ensino da Matemática deve ser ativo e que não se deve suprimir as atividades informais de produzir e extrair conceitos matemáticos do mundo que nos rodeia. Ele obteve sucesso e destaque junto à comunidade matemática, com seus trabalhos, ao conceitualizar Matemática como Resolução de Problemas, colocando-a como o foco principal da instrução matemática. Para ele, a Epistemologia Matemática e a Pedagogia Matemática estão profundamente conectadas.

Para Polya, a abstração de conceitos matemáticos, a partir de situações matemáticas, no ensino superior, por exemplo, podem ser o centro do ensino de Matemática.

Partamos do estudo do que Polya chamou de Heurística, Heurética ou *ars inveniendi* como o nome de certo ramo de estudo pertencente à Lógica, Filosofia ou Psicologia. O objetivo da Heurística, segundo Polya, é o estudo dos métodos e das regras da descoberta e da invenção. Tais métodos foram usados por matemáticos como: Bolzano, Lakatos, Descartes, Leibnitz e Poincaré na resolução de problemas, Polya elaborou o que chamou de “as quatro fases da Resolução de Problemas” como no livro *How to solve it?*, cuja primeira edição data de 1944, fundamentando praticamente todos os estudos e pesquisas neste campo da Educação Matemática.

Polya utiliza quatro fases para a resolução de problemas. Como primeira fase, deve-se compreender o enunciado, buscar e organizar dados e incógnitas, conhecer a pergunta do problema. A segunda fase consiste em estabelecer planos para solucioná-lo. Tais planos devem ser procurados em problemas semelhantes (para Polya, correlatos). Se não há nada semelhante, deve-se procurar reformular o problema. Reformular pode ser entendido como fazer uma nova interpretação do problema, uma atitude que enriquece o aspecto intuitivo do problema.

Após isso, vem a aplicação dos planos com a verificação passo a passo e, por fim, o retrospecto com a validação do resultado enquanto problema de solução compatível com a pergunta do problema. Nesse ponto acredita-se ocorrer a grande contribuição de Polya no âmbito da Educação Matemática.

Polya concebe a Matemática não como uma disciplina formal, mas enfatiza a sua correlação com a intuição, a imaginação e a descoberta, defendendo que se deve imaginar a ideia da prova de um teorema antes de prová-lo. Pode-se, dessa maneira, perceber que muitas vezes erra-se e tem-se que descobrir outras saídas, o que acaba contribuindo para melhorar nossa capacidade de imaginar soluções: “O resultado do trabalho criativo do matemático é o raciocínio

demonstrativo, a prova, mas a prova é descoberta por raciocínio plausível, pela imaginação” (POLYA, 1945 apud SCHOENFELD, 1992, tradução nossa).

Sendo assim, Polya estabelece em seu trabalho não apenas o hábito de resolver problemas, chamando-nos a atenção para o potencial das descobertas em Matemática, do trabalho criativo dos alunos e da Heurística na resolução de problemas.

Tivemos por objetivo descrever sucintamente os trabalhos de Arquimedes, Pappus, Descartes e Polya para esclarecer aspectos históricos da heurística na resolução de problemas e porque nos basearemos em Polya (1945, 1981 e 1995) para a futura análise dos dados deste estudo.

No próximo capítulo, apresentaremos a metodologia de pesquisa utilizada neste estudo.

## 4 METODOLOGIA

A presente seção tem por objetivo apresentar os elementos metodológicos constituintes do processo de investigação, como forma de garantir a confiabilidade e o rigor científico do trabalho, com vistas a construir ou refinar o processo de análise dos resultados dos levantamentos realizados.

Descreveremos os sujeitos, instrumentos e procedimentos metodológicos necessários para responder ao problema desta pesquisa. Além disso, também apresentaremos a fundamentação metodológica e as fases da *Grounded Theory* (Glaser e Strauss, 1967; Strauss e Corbin, 1998; Charmaz, 2000) utilizadas na coleta e análise dos dados.

A Figura 1 indica o caminho percorrido em relação à metodologia de trabalho escolhida para este estudo:

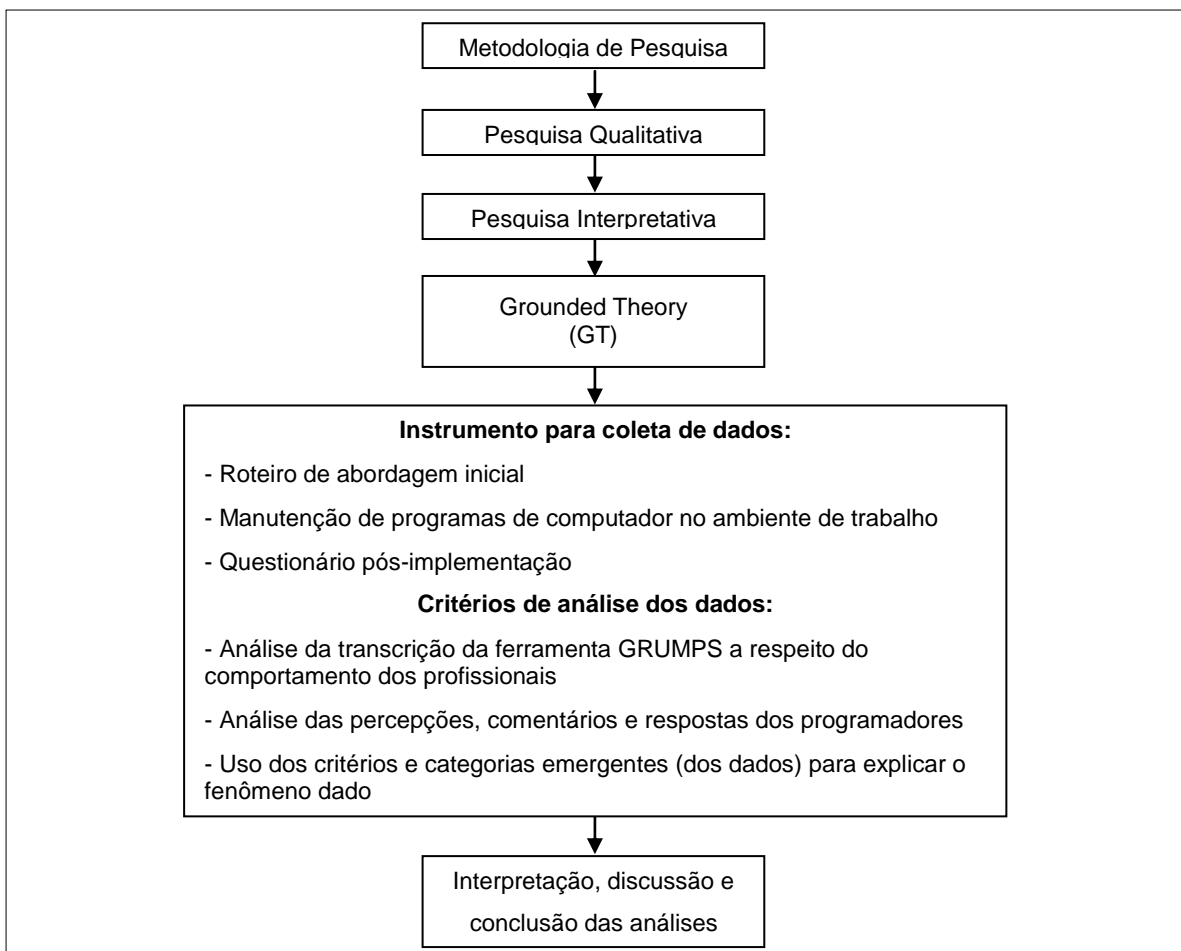


Figura 1: Uma visão geral do referencial teórico-metodológico da presente pesquisa

## 4.1 Abordagem Qualitativa

Em relação à forma de abordagem do problema a ser estudado, esta pesquisa está classificada como qualitativa, uma vez que considera a relação dinâmica existente entre o mundo real e o sujeito como um vínculo indissociável entre o mundo objetivo e a subjetividade de alguns profissionais da área de Ciência da Computação e afins, que não pode ser traduzido em números.

Uma vez que o presente estudo objetiva descrever e interpretar as atitudes e significados produzidos pelos profissionais pesquisados dentro de uma “referência significativa que merece investigação” (Chizzotti, 2003), acreditamos que as particularidades dos componentes de análise nesta pesquisa e suas contextualizações, somadas às ideias da fundamentação teórica apontam para a adoção deste método.

Para tanto, apoiamos este estudo em Godoy (1995, p. 65), que descreve:

De maneira diversa, a pesquisa qualitativa não procura enumerar e/ou medir os eventos estudados, nem emprega instrumental estatístico na análise dos dados. Parte de questões ou focos de interesses amplos, que vão se definindo à medida que o estudo se desenvolve. Envolve obtenção de dados descritivos sobre pessoas, lugares e processos interativos pelo contato direto do pesquisador com a situação estudada, procurando compreender os fenômenos segundo a perspectiva dos sujeitos, ou seja, dos participantes da situação em estudo.

Assumimos o tipo de pesquisa qualitativa por encontrar nela uma autonomia e flexibilidade que acreditamos que o contato com os desenvolvedores exigiu ao longo deste trabalho assim como também proporcionou avaliar a situação estudada com mais criatividade ao tentar buscar, nas interações com os profissionais, um tipo de revelação que só pode emergir quando estamos frente a frente com o objeto estudado, avaliando as expectativas, os valores e as expressões esboçadas nos momentos analisados.

Associada à abordagem qualitativa, utilizaremos a Pesquisa Interpretativa e a *Grounded Theory* (GT) por possibilitarem uma investigação de forma mais

ampla das questões colocadas nesta pesquisa, permitindo explicar as experiências dos programadores em seu ambiente natural.

Com o intuito de fornecer um panorama sobre a abordagem metodológica adotada neste estudo, a Figura 2, extraída de De Villiers (2005), ilustra os métodos de pesquisa situados entre o Positivismo e Pesquisa Interpretativa, bem como explicita as tendências de cada um deles no que tange os estudos quantitativos, qualitativos ou mesmo a intersecção deles. Esta pesquisa situa-se no eixo em direção à Pesquisa Interpretativa, com uma mescla de observação *in loco*, entrevistas e estudo de caso, destacados na base da Figura 2 com pontilhados.

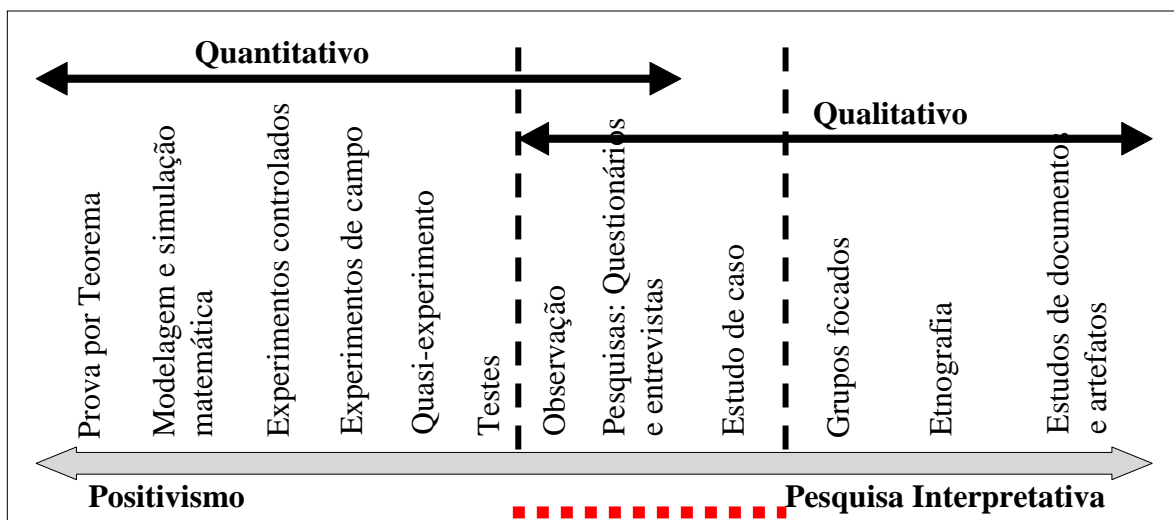


Figura 2: Adaptação da figura de estratégias e métodos de pesquisa de De Villiers (2005) (tradução nossa)

## 4.2 Pesquisa Interpretativa

A Pesquisa Interpretativa emergiu na década de 70 quando Boland (1979) primeiramente enfocou a relevância da Hermenêutica (ramo da Filosofia que se debate a compreensão humana e a interpretação de textos escritos) e Fenomenologia (estudo da consciência e dos objetos da consciência) nas pesquisas do campo de Sistemas da Informação, Ciência da Computação e afins. Desde então, o estilo interpretativo de pesquisa se tornou popular nos trabalhos destas áreas, principalmente por encorajar os pesquisadores a serem mais

interpretativos e indutivos, do que somente procurar por confirmações ou refutas de hipóteses.

A fim de reforçar isso, Orlikowski e Baroudi (1991) afirmam que a Pesquisa Interpretativa é uma abordagem bastante valiosa nos estudos dos campos de Sistemas de Informação nas organizações, sendo na visão deles, um método melhor do que o Positivismo para este propósito. De forma similar, Walsham (1995) também concorda com esta afirmação, embora o Positivismo ainda permaneça como a abordagem mais popularmente publicada nos principais trabalhos de Sistema de Informação, com o Positivismo sendo a perspectiva dominante em pesquisas do início da década de 90 com o percentual de 96,8% (Pesquisa Interpretativa com apenas 3,2%). Entretanto, como mencionado por Walsham (2006), há um aumento no número significativo de pesquisas e artigos da área de Sistemas de Informação entre 1993 e 2000 que envolvem a Pesquisa Interpretativa, totalizando cerca de 17% do total.

Estudos no campo da Pesquisa Interpretativa envolvem o entendimento do fenômeno de forma subjetiva. De acordo com Orlikowski e Baroudi (1991), o critério adotado em classificar estudos interpretativos é evidenciado por uma perspectiva não-determinística. A intenção da pesquisa consiste em aprofundar o entendimento do fenômeno dentro do contexto cultural, ou seja, onde o fenômeno de interesse está na sua configuração natural e da perspectiva dos participantes; e onde os pesquisadores não imponham seu entendimento *a priori* na situação.

Em outras palavras, a pesquisa interpretativa não pré-define variáveis dependentes e independentes dado um contexto cultural: ela elenca-as por meio da observação dos fatos e das situações que emergem a partir deles (KAPLAN e DUCHON, 1988).

Na visão de Johari (2009), a Pesquisa Interpretativa é ainda mais relevante quando envolve estudos de sistemas em contextos culturais diferentes e que também consideram diferentes perspectivas de profissionais no âmbito organizacional. Por tais motivos, o estudo interpretativo forneceria um excelente guia de como as entrevistas deveriam ser conduzidas ou de mais importante que isso, como os estudos de caso seriam interpretados. Isto ocorre porque sendo a

Pesquisa Interpretativa visto como um termo “guarda-chuva”, ele ajudaria a filtrar as afirmações, ações e comportamentos dos participantes por meio da lente da subjetividade dos pesquisadores, e então produzir uma “história” sobre os eventos que teriam ocorrido assim como as razões para eles (WALSHAM, 1995).

Walsham (2006) ainda argumenta que realizar o trabalho de campo é base fundamental para o estudo interpretativo, sendo a entrevista e observação presencial as principais formas de acessar as interpretações dos informantes em seu contexto. Ele ainda reforça que os dados qualitativos são a maior preocupação do estudo interpretativo. Isto porque a intenção da Pesquisa Interpretativa é entender o fenômeno por meio dos significados que as pessoas atribuem a ele, tentativas de aumentar o entendimento do fenômeno dentro das situações culturais e contextuais, e também onde o fenômeno de interesse será examinado na sua configuração natural e da perspectiva dos participantes.

Adicionalmente, as técnicas da Pesquisa Interpretativa permitem aos participantes usar suas próprias palavras e imagens, e delinear seus próprios conceitos e experiências. Os experimentos interpretativos diferem dos estudos positivistas porque neste segundo, o pesquisador assume que ele pode aprender sobre o fenômeno sob investigação por meio da construção de uma série de modelos das relações entre variáveis dependentes e independentes. Em cada ciclo experimental, as variáveis independentes são alteradas para valores diferentes e a saída é medida e comparada ao comportamento previsto das variáveis dependentes. Se a previsão falha, o modelo é modificado.

Nos experimentos interpretativos, o pesquisador assume que os comportamentos são melhores explicados pelos significados atribuídos às situações do experimento. O experimento tenta entender as razões do porquê as pessoas agem de determinadas formas, focando-se nos significados que eles constroem em uma situação experimental. Não há modelos *a priori* e não é assumido que o fenômeno sob investigação possa ser separado em variáveis dependentes e independentes.

Vários ramos podem ser seguidos nos estudos interpretativos, como por exemplo, os estudos de caso, Etnografia, narrativas e *Grounded Theory*. Esta

última pode ser usada para coletar dados importantes de diversas fontes, refinar conceitos, definir propriedades de categorias e identificar contextos relevantes (Charmaz, 2000). Na abordagem da *Grounded Theory*, a análise objetiva refletir as perspectivas até que a saturação dos dados tenha ocorrido (GLASER e STRAUSS, 1967; DE VILLIERS, 2005).

Utilizaremos a Pesquisa Interpretativa neste trabalho por concordarmos que, por meio de métodos qualitativos, é possível investigar de uma forma mais ampla às questões colocadas nesta pesquisa, possibilitando explicar as experiências dos programadores em seu ambiente natural e o entendimento mais detalhado da potencial influência das estratégias de resolução de problemas da Matemática. Iremos fornecer maiores detalhes sobre esta teoria na próxima seção.

### **4.3 *Grounded Theory* (GT)**

Várias práticas de pesquisa podem ser usadas como metodologia para guiar estudos interpretativos e prover consistência na análise de dados (De Villiers, 2005). Uma destas é a *Grounded Theory* (ou GT), que foi usada neste estudo como forma de coletar e organizar os dados.

A GT foi inicialmente proposta em 1967 por Glaser e Strauss com o propósito de prover um modelo de pesquisa. Em 1998, Strauss e Corbin desenvolveram uma descrição mais detalhada da GT, assim como forneceram orientações para a organização dos dados usando um conjunto de categorias bem definidas.

A GT é definida por Glaser e Strauss (1967) como o processo para “... a descoberta da teoria dos dados sistematicamente obtidos pela pesquisa social ou de campo” (tradução nossa), podendo ser empregada de duas formas.

A primeira como filosofia de pesquisa e, portanto, o pesquisador aborda a questão de pesquisa sem nenhum modelo ou contexto teórico *a priori*. Uma questão de pesquisa considerada interessante é colocada e os dados são

agrupados em torno dela. A análise de dados subsequente é empregada para assistir ao pesquisador em como os dados devem ser usados para responder as questões de pesquisa.

Na segunda forma, correspondendo à utilizada neste trabalho, a GT pode ser usada como técnica para a análise dos dados, que envolve o processo de constante comparação. A teoria sugere que as categorias e propriedades são conceitos identificados pelo pesquisador e envolvem a constante comparação dos dados. Uma categoria emerge dos dados e pode se consolidar por si própria como um elemento conceitual. Uma propriedade é um atributo da categoria. Por exemplo, a categoria “Comunicação” pode ter propriedade de “escrita” ou “verbal”. O constante processo de comparação pode suportar as categorias existentes ou gerar novas. Segundo Glaser e Strauss (1967, p. 56) colocam:

Por meio da comparação de onde os fatos são similares ou diferentes, podemos gerar as propriedades de categorias que aumentam a generalidade delas e conseqüentemente o poder de explicação das coisas (tradução nossa)

O principal propósito da GT é coletar os dados e usá-los para desenvolver uma teoria que provê interpretações relevantes, aplicações, previsões e explicações (Leedy e Ormrod, 2001; Glaser e Strauss, 1967). A GT envolve o processo de coleta dos dados para cobrir lacunas conceituais, aplicando-se análises comparativas constantemente, refinando-se conceitos, definindo-se propriedades das categorias e identificando seus contextos relevantes (CHARMAZ, 2000).

Para Strauss e Corbin (1998), as teorias fundamentadas, por serem baseadas em dados, tendem a melhorar o entendimento de um contexto e fornecer um guia importante para ação. Segundo Seaman (2008), um aspecto importante da GT é que ela intercala as fases de coleta e análise/validação dos dados para fornecer um entendimento sobre o que ocorre na prática e as razões que explicam tais fatos. Portanto, durante as entrevistas e observações da prática, hipóteses são geradas, testadas e modificadas conforme os dados são coletados.

O processo de análise dos dados envolve três tipos de codificação. Primeiro, o “aberto” que envolve atribuir categorias aos dados identificados diretamente pelo pesquisador. Segundo, “axial” ou “teórica” que envolve a identificação das relações entre as categorias. Estas relações suportam a identificação de um modelo teórico geral. Terceiro, “seletiva” que envolve garantir que todos os dados disponíveis sejam associados com uma categoria emergente e que as categorias principais sejam identificadas para suportar a conceituação teórica do modelo. Finalmente, a situação de saturação teórica é alcançada quando nenhuma nova categoria ou propriedade emerge dos dados restantes.

#### **4.4 Análise dos dados por meio da GT**

Um dos mais importantes passos na GT é a codificação dos dados, que se refere ao processo analítico por meio do qual os dados são divididos, conceitualizados e integrados à teoria (Strauss e Corbin, 1998). Ainda segundo os mesmos autores, as categorias referem-se a um conceito que representa um fenômeno; um tema representa uma “parte da realidade” que pode ser usado como base para um argumento, e a teoria denota um conjunto de categorias e/ou temas que são sistematicamente integrados para explicar um fenômeno.

O propósito dos “códigos” (resultados da etapa de codificação da GT) é capturar o significado dos dados e classificá-los, fornecendo novas perspectivas a eles a fim de guiar o pesquisador durante o processo de análise. Códigos relacionados podem ser agrupados em categorias guiados pelos dados. O processo de refinamento de categoria é contínuo até que as principais categorias sejam identificadas e integradas à unidade maior. A coleção e análise dos dados é, portanto, um processo recursivo até que a saturação ocorra. A GT busca constante verificação por meio do processo de saturação, que é alcançado quando nenhuma outra evidência emerge, e quando comportamentos múltiplos indicam propriedades e padrões similares (DE VILLIERS, 2005).

Na GT, de acordo com Charmaz (2000), a análise dos dados acontece por meio dos procedimentos de:

- **Codificação Aberta:** Este é o primeiro processo analítico de codificação. Seu propósito é destacar elementos dos dados e associá-los a códigos. Os dados são decompostos em partes (conceitos) e as partes são comparadas e agrupadas (por similaridades ou diferenças) em uma nova categoria. Este processo de agrupar os conceitos em níveis maiores de abstração é chamado de categorização e ele estabelece a base da construção da teoria (De Villiers, 2005). A seleção de grupos para comparação de várias similaridades e diferenças é vital para definir as diferentes categorias (GLASER e STRAUSS, 1967).

- **Codificação Axial:** Este processo é definido como “o processo de relacionar as categorias a suas subcategorias” (Strauss e Corbin, 1998, tradução nossa). O propósito é refinar a informação sobre cada categoria para detalhar suas condições, contexto, estratégia e consequências. “Códigos” são relacionados e o foco passa a ser as relações entre várias categorias. Tais relações podem não ser necessariamente explícitas, mas também implícitas ou conceituais apenas.

- **Codificação Seletiva:** É o processo em que uma categoria principal é selecionada para que outras sejam relacionadas a ela. É o processo pelo qual as categorias são organizadas em torno de um conceito central e finalmente cada uma delas pode se tornar um tema. Esse processo de integrar e refinar a teoria pode facilitar a diagramação, identificação de eventos críticos e análise detalhada.

- **Interpretação e desenvolvimento da teoria:** A interpretação qualitativa é construída a partir de “achados”. De acordo com Denzil e Lincoln (2000), o processo de interpretação é crítico e essencial para o entendimento do que se constitui a teoria. Strauss e Corbin (1998) enfatizam que a teoria denota um conjunto de categorias bem definidas que são sistematicamente integradas em uma teoria que explica um fenômeno. A teoria deve ser utilizável em aplicações práticas, deve prover uma perspectiva no comportamento, e deve guiar e fornecer um estilo de pesquisa. Por isso, a GT pode ser aplicada em trabalhos ligados à Computação, pois também pode conduzir a modelos e/ou representações para explicar um fenômeno específico no domínio em questão (DE VILLIERS, 2005).

Estes procedimentos serão adotados nesta pesquisa, seguindo a dinâmica descrita de coleta e observação dos dados, agrupamento (codificação aberta), refinamento e definição de relacionamentos (codificação axial), organização em torno de um conceito central (codificação seletiva) a fim de possibilitar a análise dos dados (interpretação) após confirmação dos padrões e propriedades sob investigação (saturação). Maiores detalhes das etapas de codificação estão descritos na seção de Critérios de Análise, do capítulo de Procedimentos Metodológicos.

Este estudo usa a GT como uma metodologia para estudar os dados de uma forma exploratória e como sugerido por Benbasat et al (1987), também por ser consistente com as três razões a seguir:

- 1) A pesquisa é um estudo no ambiente natural, em que a teoria e análise dos dados irão emergir da prática
- 2) O pesquisador pode responder questões que conduzam a um entendimento melhor da natureza e complexidade dos processos que estão sob análise
- 3) É apropriada para pesquisar uma área de estudo pouco explorada

Adicionalmente a isso, devido à experiência profissional da pesquisadora na área de estudo, a GT foi também bastante apropriada porque forneceu um método para controlar o risco de introduzir desvios no estudo. Este controle aconteceu devido às constantes comparações exigidas pelo método, que forçam os pesquisadores a olharem criticamente para suas premissas e seu próprio conhecimento como dados e compará-los com outros dados do estudo.

Esta constante comparação dos eventos então valida, modifica ou rejeita as observações dos pesquisadores experientes. Por isso, para pesquisadores com boa experiência profissional no campo de pesquisa, a constante comparação é uma característica valiosa do método GT, pois reduz (mas não elimina completamente), o risco de distorções ou desvios induzidos.

É importante ressaltar que embora os procedimentos adotados neste estudo se baseiem nas etapas da GT, o processo metodológico não é linear, permitindo idas e vindas em cada uma das etapas.

As sessões de interação conduzidas para esta pesquisa podem ser sumarizadas da seguinte forma:

	<b>Objetivo</b>	<b>Duração</b>	<b>Artefatos utilizados</b>
<b>Sessão 1</b>	<ul style="list-style-type: none"> <li>- Introduzir objetivo desta pesquisa</li> <li>- Levantar perfil acadêmico e formação escolar</li> <li>- Levantar experiência e reconhecimento profissionais</li> </ul>	30 minutos por programador	Anexo 4 – Roteiro para Abordagem Inicial
<b>Sessões 2, 3, 4, 5 e 6</b>	- Captura dos dados, por meio da ferramenta GRUMPS instalada no computador de cada participante, do processo de manutenção de programas computacionais no ambiente de trabalho	Aproximadamente 8 horas e 30 minutos por programador	Interações Capturadas pelo Aplicativo GRUMPS
<b>Sessão 7</b>	- Aplicar questionário pós-implementação	Aproximadamente 1 hora por programador	Anexo 5 – Questionário Pós-Implementação

Quadro 2: Esquema das sessões conduzidas para este estudo

No próximo capítulo, apresentaremos detalhes de tais interações bem como os sujeitos participantes das entrevistas e os procedimentos metodológicos utilizados neste trabalho.



## 5 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo pretendemos apresentar os sujeitos participantes desta pesquisa bem como os instrumentos utilizados para a coleta dos dados.

Para apresentar os resultados da análise realizada, os dados obtidos foram sistematizados de modo a facilitar a leitura e a interpretação dos mesmos via a aplicação das etapas da GT, conforme descrito na seção de critérios de análise a seguir.

Dividimos a coleta de dados deste estudo em dois momentos: um primeiro grupo de três profissionais desafiados a solucionar um problema pré-determinado por esta pesquisadora (relacionado a criação de um componente de calendário) e posteriormente, um segundo grupo de trinta e quatro programadores observados diretamente em seu dia-a-dia (resolvendo problemas reais na manutenção dos programas de computador).

Optamos por realizar um segundo ciclo de coleta de dados devido ao fato de que todos os programadores selecionados possuem experiência mínima de cinco anos na área, e portanto, a abordagem inicial de uma tarefa pré-determinada para observação do processo de resolução e posterior análise, não se mostrou muito eficiente e de fato, pouco provocativa. Esses motivos trouxeram à tona que a observação *in loco* dos programadores realizando suas atividades durante o processo de manutenção dos *softwares* forneceriam elementos interessantes para a fundamentação deste trabalho.

Nas sessões a seguir descreveremos os sujeitos e os instrumentos de coleta de dados utilizados no primeiro e segundo ciclos de coleta dos dados.

### 5.1 Etapa 1

#### 5.1.1 Sujeitos

Pretendemos nesta seção descrever a trajetória profissional dos três participantes do primeiro ciclo desta pesquisa bem como suas formações

acadêmicas e experiências anteriores em programação, sempre focando a perspectiva qualitativa.

Os participantes escolhidos para a primeira etapa desta pesquisa foram três profissionais da área de Tecnologia da Informação de uma mesma empresa multinacional americana prestadora de serviços na área de consultoria, desenvolvimento e manutenção de *software*. Os três participantes escolhidos foram selecionados para este estudo primeiramente por terem despertado interesse em contribuir para este trabalho. Além disso, eles também trabalham na mesma empresa a qual a pesquisadora fornece serviços, em um diferente setor. E por último, mas não menos importante, por serem reconhecidos em seu ambiente de trabalho como bons profissionais, com alto poder de entrega, cumprimento de prazo e boa qualidade em seus resultados.

Todos trabalham no mesmo departamento de projetos e prestação de serviços para organizações do ramo farmacêutico. De forma geral, os três participantes possuem nível similar de experiência na área de programação, com cerca de nove a onze anos de vivência neste setor.

A fim de resguardar a identidade de cada um deles e por questões de confidencialidade, seus nomes não serão divulgados e estaremos identificando-os como Prog1, Prog2 e Prog3.

As informações sobre o perfil de cada profissional participante foram capturadas na primeira sessão conduzida para coleta dos dados, seguindo o roteiro para abordagem inicial, presente no Anexo 1 desta pesquisa.

### **5.1.2 Prog1**

O profissional Prog1 tem trinta e dois anos, é graduado em Ciência da Computação há dez anos, e possui uma especialização em Programação Orientada a Objetos, formações obtidas na mesma universidade.

Quando questionado a respeito de seu desempenho universitário nas disciplinas técnicas de seu curso tais como Introdução à Lógica de Programação,

Linguagens de Programação e afins, o Prog1 ressaltou seu bom e constante desempenho acima da média da sala. Ainda adicionou que isso foi consequência de uma forte influência familiar para sua formação técnica, visto que seu pai e mãe (agora aposentados) exerceram atividades de programação nos anos setenta e oitenta.

A respeito de seu histórico escolar em disciplinas ligadas diretamente à Matemática (tais como Cálculo, Estatística, Álgebra Linear, etc), o Prog1 comentou que também tinha relativo sucesso, constantemente superando a expectativa de média para aprovação no curso.

Prog1 também destacou que tinha facilidade em conectar elementos aprendidos da Matemática com sua aplicação prática nas técnicas de programação. Para exemplificar isso, ele comentou sobre como aplicou elementos da Matemática tais como matrizes e vetores e suas melhores práticas para definir melhor as estruturas de dados para armazenamento das informações nos programas que desenvolvia.

Questionado sobre sua experiência e desenvoltura em resolver problemas da Matemática em sala de aula, Prog1 comentou que normalmente para obter uma boa nota na disciplina, ele não só decorava as “fórmulas prontas” dadas, mas também buscava indagar aos professores a respeito de quais situações exatas ele deveria se utilizar daquele “passo-a-passo”. Prog1 disse ainda que nunca foi muito bom em demonstrações e sempre as achou extremamente teóricas.

Uma das únicas vezes que se recorda a respeito de conseguir traduzir um problema da Matemática para algo mais perto de sua realidade foi quando os professores de Matemática e Lógica de Programação se “uniram” para apresentar um conteúdo em comum: teoria dada na aula de Matemática sobre senos/cosenos/tangente e demonstração prática de programas construídos para mostrar os gráficos correspondentes.

Sobre sua carreira profissional, Prog1 já trabalhou em três empresas diferentes ao longo dos seus onze anos de experiência profissional na área de

desenvolvimento de programas. Na primeira, uma instituição bancária, estagiou por um ano na área de Tecnologia da Informação, focando-se na linha de desenvolvimento de páginas para *Intranet* corporativa.

Já na segunda empresa do ramo de varejo nacional, onde permaneceu por oito anos, Prog1 iniciou como programador júnior na área de Arquitetura de Sistemas, participando da equipe responsável pela programação da loja de compras virtual da companhia via Internet. Em função de seu bom desempenho, recebeu duas promoções após três e sete anos de companhia, sempre como parte da mesma equipe de desenvolvimento.

Finalmente na última e atual companhia onde trabalha há cerca de dois anos, Prog1 atua como analista-programador sênior, especializado em desenvolvimento de componentes customizados para plataforma *Web*. É reconhecidamente respeitado como um forte detentor de conhecimento técnico (publica artigos em revistas técnicas e é sempre envolvido em problemas complexos) e tido como referência no departamento.

### **5.1.3 Prog2**

Com vinte e cinco anos de idade, o programador Prog2 é formado em Sistemas de Informação há três anos e atua a cerca de nove anos no mercado de trabalho.

Sobre seu histórico de desempenho nas disciplinas técnicas, Prog2 explanou que obteve boas notas e, no geral, um bom histórico. Para exemplificar isso, Prog2 comentou que sempre gostava de participar ativamente das atividades de “desafio”, que segundo ele, correspondiam a situações mais elaboradas, propostas pelos professores, com o intuito de provocar uma reflexão maior para os alunos acerca de um tópico ou técnica de programação ensinados.

Acerca de seu desempenho nas disciplinas ligadas à Matemática, o profissional Prog2 comentou que embora visualizasse aplicação direta de alguns conceitos matemáticos nas disciplinas técnicas, normalmente não tinha grande

interesse nas matérias não-técnicas e suas notas eram normalmente em torno da média mínima exigida.

Quando questionado sobre sua experiência e desenvoltura em resolver problemas da Matemática em sala de aula, Prog2 respondeu que sempre achou a Matemática extremamente “formalista” (palavras do entrevistado) na resolução de problemas e que não entendia o porquê precisava fazer tantas demonstrações de “coisas que jamais utilizaria” (palavras do entrevistado). Prog2 também comentou que sempre se queixava da excessiva repetição na realização de cansativos “exercícios” (palavras do entrevistado).

Com relação a sua vivência profissional, Prog2 ingressou em 2001 no programa de estágio na mesma companhia que ainda trabalha atualmente. Após um ano de participação no programa e tendo realizado diversos treinamentos técnicos de preparação para posterior efetivação como funcionário, Prog2 tornou-se analista-programador júnior na área de Manutenção de Redes de Computadores.

Após oito meses de trabalho nesta área de manutenção de equipamentos, Prog2 decidiu mudar o foco de sua carreira e investir em seu desenvolvimento individual no setor de Tecnologia da Informação, mais especificamente na área de implementação de sistemas *Web*.

Em termos de reconhecimento profissional, já recebeu duas premiações corporativas por sugerir inovações e mudanças nos modelos de especificação técnicas que chegam até aos analistas-programadores de *software* da área. É reconhecidamente considerado uma referência jovem no time de desenvolvimento pela desenvoltura em lidar com usuários e traduzir os requisitos funcionais em boas e rápidas soluções de *software*.

#### **5.1.4 Prog3**

O profissional Prog3 graduou-se em Tecnologia de Processamento de Dados em 2002 e cursou um ano de complementação para o grau de

bacharelado em Sistemas de Informação em 2003. Finalizou também uma especialização em Gerenciamento de Projetos em 2008.

Em relação a seu histórico escolar nas disciplinas técnicas, o Prog3 mencionou ter tido um bom desempenho, embora considere como um ponto de melhoria sua dedicação aos estudos. De acordo com o depoimento de Prog3, seu diferencial em relação aos outros colegas de classe era principalmente pelo fato de não se dedicar exaustivamente aos “estudos teóricos” (palavras do entrevistado), mas conseguir “captar a essência” (palavras do entrevistado) do que era solicitado e conseguir desempenhar, sem grandes complicações, as tarefas técnicas necessárias.

De forma similar, Prog3 comentou que seu desempenho nas disciplinas relacionadas à Matemática também obedecia ao mesmo padrão: boas notas, sem grande dedicação à “teoria”, mas quando desafiado, conseguia imprimir de forma clara e lógica seus pensamentos. Prog3 ainda comentou que esta característica de flexibilidade o ajudou muito em sua carreira profissional, visto que ao se deparar com determinadas situações adversas, ele conseguia sugerir ou recomendar soluções satisfatórias do ponto de vistas dos requisitantes.

Acerca de sua experiência e desenvoltura em resolver problemas da Matemática em sala de aula, Prog3 destacou que teve uma experiência bastante válida com relação a isso, pois segundo ele, seus professores de Matemática durante o Ensino Médio sempre apresentavam problemas extraídos de diversas edições das Olimpíadas de Matemática, o que sempre o cativava. Sobre a resolução de problemas ligados à demonstração de teoremas, Prog3 comentou que “até compreendia o raciocínio por trás dos passos necessários, e até conseguiu corrigir uma demonstração de um teorema por um professor de Cálculo 1, durante a faculdade. Porém meu argumento foi lógico e não formal: era impossível aquela condição estar ali...”.

Sua experiência profissional é composta de diversas passagens em consultorias nos primeiros cinco anos de carreira, sempre focando no desenvolvimento de sistemas para plataforma *Web*. Nos demais cinco anos de experiência, o profissional Prog3 passou na companhia que atualmente presta

serviços, sempre no setor de suporte e manutenção a componentes reutilizados por outras áreas de desenvolvimento de sistemas. Nunca recebeu nenhuma premiação, mas seu desempenho é constante e suas entregas com alta qualidade (Prog3 comentou deter o menor grau do departamento em relação a retrabalho nos componentes desenvolvidos, marca atingida durante sete meses seguidos, considerado um recorde na área).

### **5.1.5 Instrumentos para Coleta de Dados**

Como instrumentos para a coleta de dados no primeiro ciclo desta pesquisa, utilizamos uma especificação funcional (ou detalhamento técnico) de um programa de computador combinado com observação dos participantes executando as tarefas solicitadas, bem como entrevista via um questionário após a implementação da atividade proposta. Tais elementos são descritos mais detalhadamente a seguir.

#### **5.1.5.1 Programa de Computador**

Os três participantes foram requisitados a projetar um ou mais programas de computador que atendessem os mínimos requisitos detalhados na especificação técnica-funcional presente no Anexo 2.

Uma vez que os três programadores trabalham em um setor de desenvolvimento de sistemas para áreas usuárias localizadas na América do Norte (Canadá e Estados Unidos), e a fim de tornar a atividade proposta nesta pesquisa o mais próximo possível do cotidiano deles, decidimos optar por criar a especificação técnica-funcional em Inglês. Acreditamos que tal atitude poderia tornar o trabalho proposto neste estudo mais familiar e real aos três profissionais.

Basicamente, o objetivo principal da tarefa proposta consistia em construir um componente de *software* (um ou mais programas de computador) para cálculos, conversões e manipulação de datas, elementos muito comuns na prática destes profissionais. Uma vez que buscamos descrever o desenrolar do processo

de implementação da solução para o problema dado, elegemos um assunto de simples entendimento, com o intuito de fazer com que os profissionais se concentrassem mais no desenvolvimento e raciocínio exigidos, do que nas dificuldades técnicas que poderiam surgir.

O componente requerido por esta pesquisa não possui um escopo pré-determinado de uso, ou seja, os programadores foram desafiados a construir um objeto sem saberem de antemão, quais outros sistemas estariam reutilizando suas funcionalidades.

Além das características mínimas requeridas no documento de especificação, e com o intuito de deixarmos o problema em questão mais aberto, os participantes puderam decidir se incorporariam mais funcionalidades ou melhorias ao componente.

Em termos técnicos, os programas poderiam ser feitos nas linguagens de programação JavaScript, Java ou VB.NET<sup>2</sup>, a critério de cada profissional participante. Permitimos esta flexibilidade pelo fato de também dominarmos as três linguagens e acreditarmos que isso não se tornaria obstáculo para a coleta e análise dos dados para esta pesquisa.

A especificação técnica-funcional foi dividida em três seções principais. O intuito da primeira seção é introduzir o objetivo do componente em questão a ser implementado, bem como situar e contextualizar o programador sobre a necessidade do mesmo. A segunda seção tem por objetivo descrever as especificações técnicas de programação, tais como as linguagens e ferramentas permitidas, assim como impor limites importantes e que influenciarão o processo de pesquisa do algoritmo de conversão de datas. Por fim, a última seção busca delinear os requisitos de validação, elementos de tela, arquitetura e funcionalidades básicas que o componente deverá conter.

---

<sup>2</sup> JavaScript, Java e Visual Basic.NET correspondem a linguagens técnicas de programação de alto nível utilizadas para construção de componentes e aplicativos computacionais, cujos comandos e sintaxe se assemelham à linguagem natural.

### **5.1.5.2 Observação Atenta dos Sujeitos no Processo de Desenvolvimento**

A observação dos participantes durante o primeiro ciclo de investigação foi conduzida de 14 de Setembro de 2009 a 26 de Novembro de 2009 nas instalações de seus ambientes de trabalho. Organizamos três sessões de uma hora cada (exceto para a primeira sessão, acrescida de quinze minutos para introduções) a fim de acompanhar os participantes durante o processo de desenvolvimento do componente proposto, bem como capturar as reflexões e o progresso das atividades. Estabelecemos, aproximadamente, três horas para completar esta atividade em função de nossa experiência na área, baseado no histórico de esforço gasto para componentes de complexidade similar no mercado.

Ao final de cada sessão, os programadores eram solicitados a gravar a última versão de seus programas e arquivos de trabalho em diretórios pessoais presentes no dispositivo *pen-drive* da pesquisadora. A retomada do processo de desenvolvimento só acontecia durante as sessões agendadas individualmente com cada profissional.

Na primeira sessão foi tomado o cuidado de explicar aos entrevistados que as observações e perguntas seriam pertinentes somente a este estudo e por isso não haveria motivos para constrangimentos ou omissões. Ainda salientamos que o objetivo do estudo não era apresentar aos locais de trabalhos dos programadores as respostas ou compartilhar as notas recolhidas por meio das interações ali gravadas: todo o material coletado foi interpretado como confidencial pela pesquisadora.

Ainda na primeira sessão foi explanado e reforçado o objetivo do programa de computador a ser criado e questionado a respeito de eventuais dúvidas a serem esclarecidas. Também reforçamos o fato de não estarmos nos limitando somente a uma análise do produto final (programa criado) e sim, por buscarmos enriquecer este estudo por meio da observação do comportamento, e questionamentos que viriam a surgir durante o processo de programação. Ou seja, eles estariam livres para interagir e teriam abertura suficiente para questionar e expor suas ideias durante o desenrolar da criação do componente.

Após toda essa explicação na primeira sessão, seguimos o roteiro para abordagem inicial, presente no Anexo 1 desta pesquisa com o intuito de coletar algumas informações históricas sobre cada profissional participante. Tais informações são importantes não somente do ponto de vista de desconstruir o ambiente para início da observação do processo de programação do componente, mas também pelo fato de serem utilizadas mais à frente no processo de análise dos dados.

Como próximo passo, foi iniciado o desenvolvimento dos programas. Posicionamo-nos ao lado de cada profissional com o intuito de não atrapalharmos a linha de raciocínio, porém ao mesmo tempo, estarmos próximos dos eventos que ali aconteciam. Aproximadamente a cada quinze minutos, a fim de esclarecer eventuais ambiguidades e ainda instigar o processo que ali ocorria, intervínhamos com questões abertas tais como sobre o percentual de progresso conquistado até então, eventual necessidade de acesso à Internet para validar o algoritmo de conversão de datas, etc.

A respeito da dinâmica da segunda e terceira sessões, ambas foram bastante similares no sentido de fornecer ao programador acesso aos seus arquivos de trabalho e então observarmos, interagirmos e questionarmos seu progresso durante a implementação do programa.

Durante as três sessões da primeira etapa de coleta de dados, a pesquisadora anotava atentamente as principais falas dos participantes bem como questionava mais profundamente certos momentos considerados significativos por estarem diretamente relacionados às questões desta pesquisa.

### **5.1.5.3 Questionário Pós-Implementação**

Ao final da terceira sessão observatória do processo de desenvolvimento do programa de computador solicitado, realizamos uma entrevista semi-estruturada, segundo a visão de Triviños (1987).

Partimos do pressuposto que existia uma real necessidade desse tipo de entrevista, pois buscávamos um direcionamento mais aberto com os entrevistados. Em outras palavras, seguimos um roteiro pré-definido de perguntas a fim de nos orientarmos, mas sem perdermos a liberdade dos participantes responderem com flexibilidade, permitindo uma conversa mais harmônica com os profissionais pesquisados. Por tais motivos, a entrevista semi-estruturada segundo Triviños (1987) foi o formato utilizado.

O questionário, presente no Anexo 3 deste estudo, foi dividido em três partes:

- *Especificação Técnica-Funcional:* Nesta seção buscamos coletar a opinião dos entrevistados em relação à qualidade dos requisitos fornecidos e sua influência no resultado final criado;
- *Reusabilidade de Componentes de Software:* Procuramos capturar aqui a concepção dos profissionais acerca do conceito de reusabilidade, sua aplicação prática no componente desenvolvido bem como as inter-relações existentes entre o processo de programação e o raciocínio lógico, habilidades e estratégias da Matemática;
- *Programação e sua Relação com a Matemática:* Nesta última parte nossa intenção foi a de identificar, do ponto de vista dos entrevistados, de que forma os elementos da Matemática influenciam suas decisões e julgamentos na prática profissional, assim como que elementos da Matemática eles consideram importante na preparação de profissionais da área de desenvolvimento de sistemas para o mercado de trabalho.

O objetivo de aplicarmos um questionário ao término das três sessões não foi no sentido de avaliarmos a qualidade do produto final gerado, mas sim com a finalidade de termos outra oportunidade para detalharmos juntos (pesquisadora e pesquisados) os conhecimentos mobilizados e o raciocínio envolvido para expressar a lógica do programador na solução concebida para o problema dado.

## 5.2 Etapa 2

Apoiando-nos na GT como metodologia de pesquisa, buscamos neste trabalho explicar um fenômeno.

Segundo Seaman (2008), um aspecto importante da GT é que ela intercala as fases de coleta e análise/validação dos dados para fornecer um entendimento sobre o que ocorre na prática e as razões que explicam tais fatos. Portanto, durante as entrevistas e observações da prática, hipóteses são geradas, testadas e modificadas conforme os dados são coletados.

Esse foi o principal motivo para a realização de uma segunda etapa de coleta dos dados: o primeiro ciclo de questionamentos não se mostrou suficiente para a saturação das categorias emergentes a fim de suportarem a conceituação teórica do modelo proposto. Para tanto, esta pesquisadora se lançou novamente à etapa de coleta de dados, de uma maneira mais provocativa e natural aos participantes deste estudo: observação *in loco* de trinta e quatro programadores realizando seu trabalho durante o processo de manutenção dos *softwares*.

### 5.2.1 Sujeitos

Pretendemos nesta seção descrever, de forma consolidada, a trajetória profissional e acadêmica dos trinta e quatro participantes do segundo ciclo de coleta de dados desta pesquisa.

Tais participantes escolhidos para esta segunda etapa de nossa pesquisa são profissionais da área de Tecnologia da Informação de uma mesma empresa multinacional americana prestadora de serviços na área de consultoria, desenvolvimento e manutenção de *software*. Todos eles pertencem ao mesmo departamento de Gerenciamento de Serviços e Operações, focado na prestação de serviços para organizações do ramo farmacêutico na região do Vale do Paraíba, em São Paulo.

O principal motivo da seleção de um departamento inteiro para a coleta de dados na Etapa 2 deste estudo foi baseado nas lições aprendidas da Etapa 1 de coleta de dados, visto que ao decidirmos focar em um grupo maior em número e com maior diversidade de perfis, estaríamos estimulando naturalmente a proporção de mais dados para que alcançássemos a saturação teórica das categorias emergentes via a metodologia GT e por consequência, posterior suporte da conceituação teórica do modelo proposto por esta pesquisa.

De forma geral, todos participantes possuem nível similar de experiência na área de programação, com cerca de nove a onze anos de vivência neste setor.

A fim de resguardar a identidade de cada um deles e por questões de confidencialidade, seus nomes não serão divulgados e utilizamos a identificação Prog1 a Prog34.

As informações seguintes sobre o perfil consolidado dos profissionais participantes foram capturadas na primeira sessão da Etapa 2 conduzida para coleta dos dados, seguindo o roteiro para abordagem inicial, presente no Anexo 4 desta pesquisa.

Os gráficos a seguir visam ilustrar e descrever, de forma geral, o perfil dos trinta e quatro participantes deste estudo, elencando algumas das características capturadas durante a aplicação do roteiro para abordagem inicial (Anexo 4) realizada na primeira sessão da Etapa 2 de coleta de dados.

Iniciando a consolidação dos dados coletados, foi possível constatar que a idade média dos trinta e quatro especialistas observados durante a segunda fase de coleta de dados é de trinta e três anos. Conforme mencionado anteriormente, todos pertencem ao mesmo departamento de Gerenciamento de Serviços e Operações, focado na prestação de serviços para organizações do ramo farmacêutico na região do Vale do Paraíba, em São Paulo. Esta área é responsável pela manutenção de cerca de dezoito sistemas computacionais críticos para o negócio, incluindo websites, aplicações de banco de dados e sistemas de apoio à decisão gerencial.

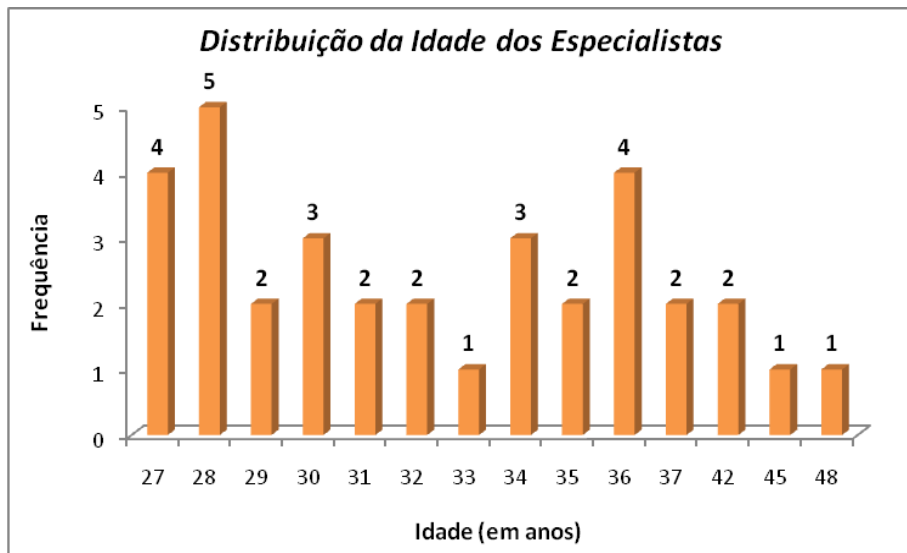


Gráfico 1: Distribuição da frequência de idade dos programadores participantes da segunda fase de coleta de dados.

Com relação à formação acadêmica dos programadores selecionados, todos possuem graduação em áreas da Computação, conforme descrito no Gráfico 2, exceto o especialista Prog10, formado em Administração de Empresas, porém pós-graduado (mestrado acadêmico) em Engenharia de *Software* pelo Instituto de Matemática e Estatística da Universidade São Paulo (USP).

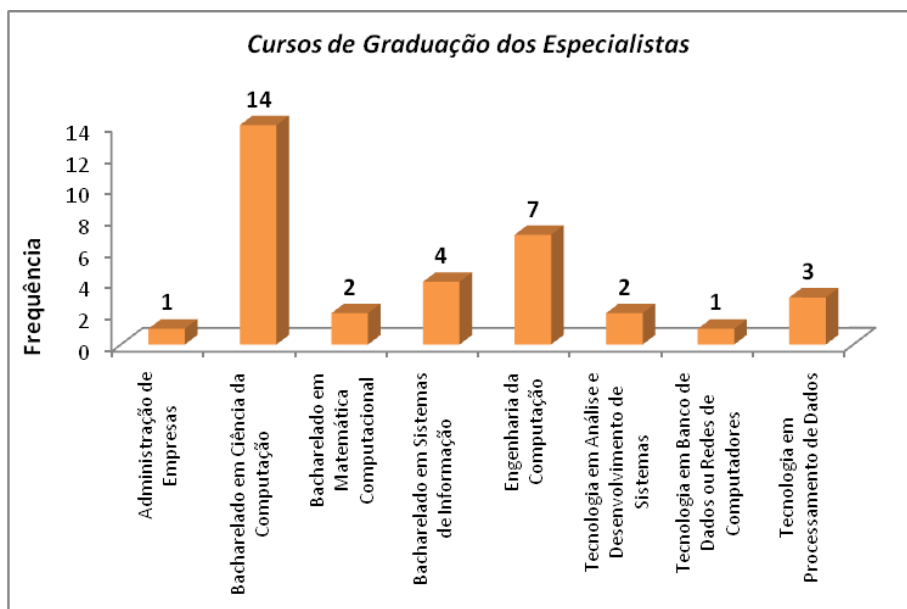


Gráfico 2: Distribuição dos cursos de graduação dos programadores participantes da segunda fase de coleta de dados.

Do ponto de vista da formação acadêmica dos participantes, vinte e dois deles se graduaram entre os anos de 1996 e 2005, conforme distribuição presente no Gráfico 3.

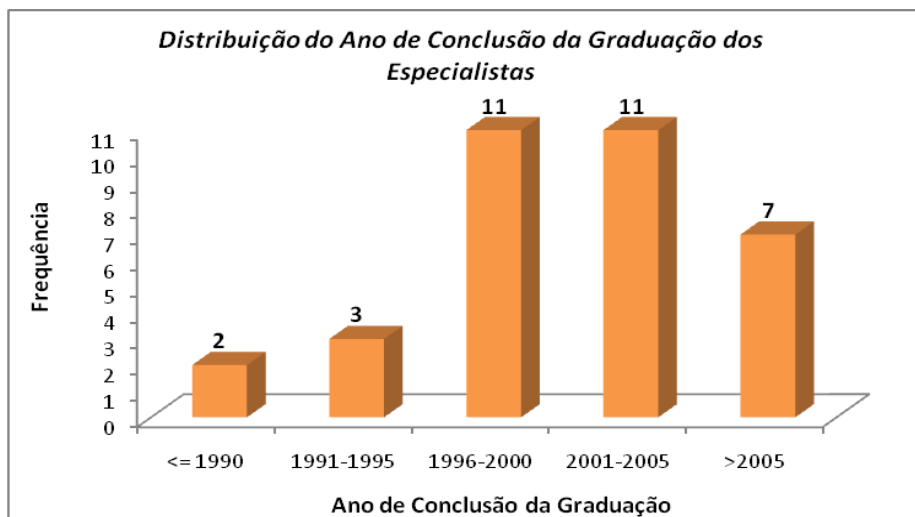


Gráfico 3: Distribuição do ano de conclusão da graduação dos programadores da segunda fase de coleta de dados.

Uma vez que a maioria dos sujeitos participantes nasceram, moram ou trabalham na região do Vale do Paraíba-SP, podemos entender o porquê, olhando para as instituições onde cursaram seus respectivos cursos de graduação, a grande parte das universidades se encontram em São José dos Campos ou Itajubá (região Sul de Minas Gerais, localidade próxima ao Vale do Paraíba também), dois dos principais pólos educacionais da região.

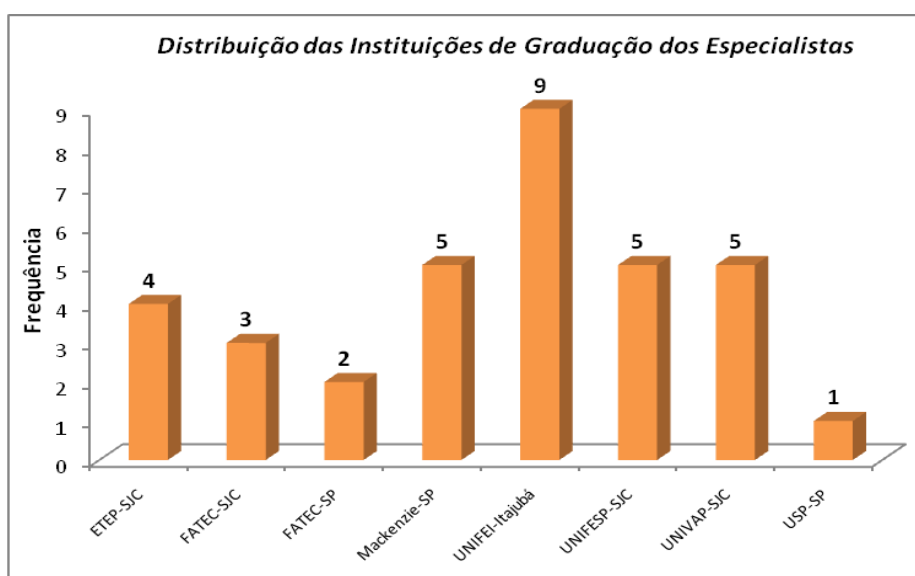


Gráfico 4: Distribuição das instituições de graduação dos programadores participantes da segunda fase de coleta de dados.

Em relação aos cursos de pós-graduação, vinte e sete programadores observados não possuem nenhum tipo de especialização, quatro já concluíram o curso de MBA em Gerenciamento de Projetos pela Faculdade Getúlio Vargas – São José dos Campos bem como outros dois indivíduos já finalizaram sua especialização em Sistemas Computacionais Orientados à Objeto, pela Universidade Mackenzie em São Paulo. Somente Prog10 possui mestrado acadêmico em Engenharia de *Software*, conforma citado anteriormente.

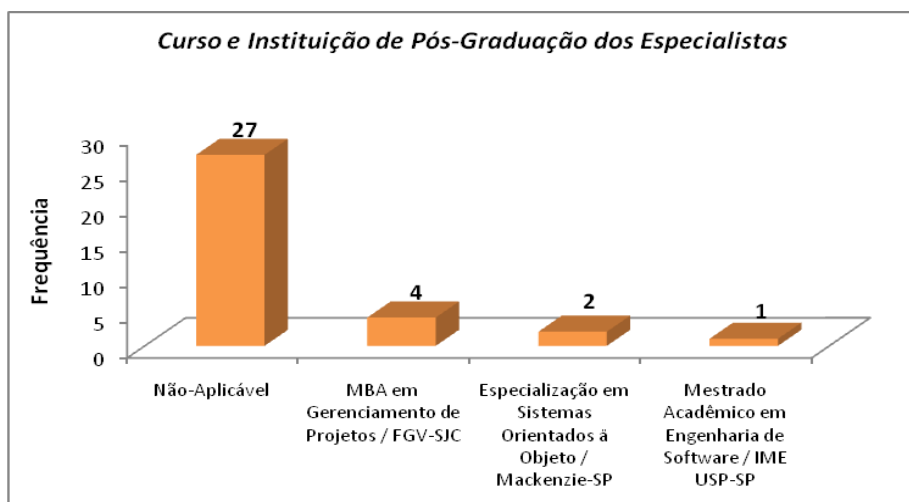


Gráfico 5: Distribuição dos cursos de pós-graduação dos programadores participantes da segunda fase de coleta de dados.

Já no campo profissional foi possível constatar que, em média, os especialistas possuem onze anos de experiência no mercado de trabalho e que mais da metade deles possui até dez anos de experiência em Computação, ou seja, trata-se de um grupo de especialistas experientes.



Gráfico 6: Distribuição da experiência profissional (em anos) dos programadores da segunda fase de coleta de dados.

Acerca dos cargos e perfis dos especialistas, apropriamo-nos das seguintes descrições resumidas, conforme no organograma da empresa na qual os programadores selecionados para este estudo trabalham:

- Analista Júnior: Profissional com cerca de cinco anos de vivência no mercado de trabalho, responsável por levantamento de requisitos, análise, programação e testes dos sistemas solicitados por seus usuários.

- Analista Pleno: Profissional com cerca de sete anos de vivência no mercado de trabalho, responsável por levantamento de requisitos, análise, especificação, projeto do sistema, programação, testes, homologação, implantação e acompanhamento dos sistemas solicitados por seus usuários.

- Analista Sênior: Profissional com cerca de dez anos de vivência no mercado de trabalho, responsável por levantamento de requisitos, análise, especificação, projeto do sistema, programação, testes, homologação, implantação e acompanhamento dos sistemas solicitados por seus usuários. Também é responsável por interagir e suportar tanto aos usuários e infraestrutura tecnológica, quanto outros analistas com menor tempo de experiência.

- Arquiteto: Profissional com cerca de dez a quinze anos de vivência no mercado de trabalho, responsável por projetar novas soluções sistêmicas, realizar análises de impactos em mudanças nas aplicações, bem como zelar pela estabilidade dos sistemas solicitados por seus usuários. Também é responsável por definir, verificar e orientar as boas-práticas de programação a serem seguidas pelo grupo de analistas ao qual está vinculado.

- Especialista em Banco de Dados: Profissional com cerca de dez anos de vivência no mercado de trabalho, responsável por planejar, organizar e programar o processamento, armazenamento, recuperação e disponibilidade das informações guardadas em banco de dados dos sistemas solicitados por seus usuários bem como definir, verificar e orientar as boas-práticas de acesso à dados a serem seguidas pelo grupo de analistas ao qual está vinculado.

- Líder Técnico: Profissional com cerca de dez a quinze anos de vivência no mercado de trabalho, responsável pela supervisão técnica de um grupo de

analistas, monitoramento da qualidade dos programas de computador por eles criados/alterados, desenvolvimento de programas cuja complexidade englobe conhecimento profundo da área de negócio, bem como gerenciamento dos requisitos dos sistemas solicitados por seus usuários.

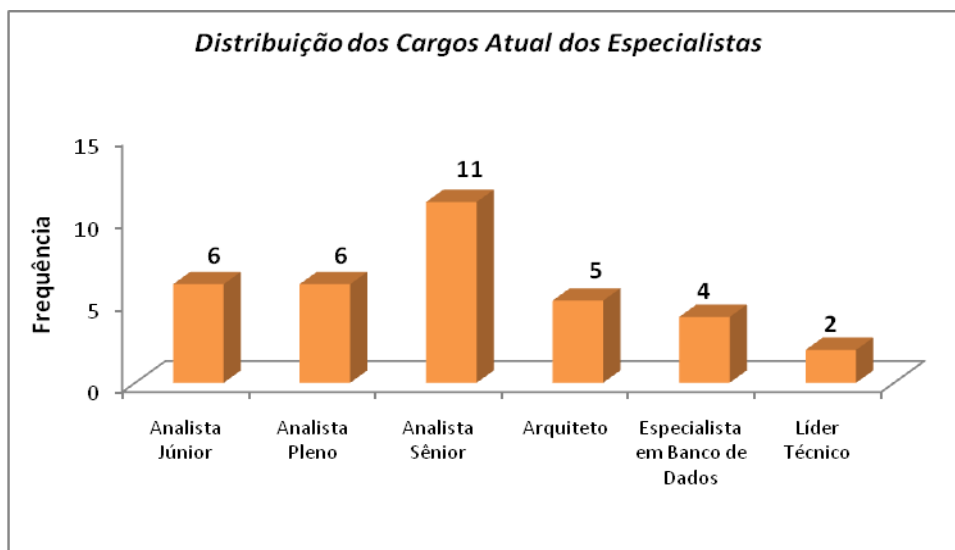


Gráfico 7: Distribuição dos cargos dos programadores participantes da segunda fase de coleta de dados.

Classificamos o desempenho dado pelos próprios programadores com relação às disciplinas relacionadas à Matemática, resolução de problemas da Matemática em sala de aula e desempenho nas disciplinas técnicas da Computação conforme a seguir:

- Insatisfatório: O sujeito expunha, durante sua fala, claras evidências de desempenho abaixo da média exigida na respectiva disciplina em seu curso de graduação ou pós-graduação, ocasionando, na maioria das vezes, necessidade de se refazer a matéria devido à reprovação da mesma

- Satisfatório: O sujeito apresentava, durante sua fala, certa familiaridade com a disciplina e desempenho na média ou pouco significativamente maior que a exigida em seu curso de graduação ou pós-graduação, nunca ocasionando necessidade de se refazer a matéria devido à reprovação da mesma

- Muito Satisfatório: O sujeito demonstrava, durante sua fala, grande familiaridade com a disciplina e desempenho significativamente acima da exigida

em seu curso de graduação ou pós-graduação, nunca ocasionando necessidade de se refazer a matéria devido à reprovação da mesma

Quando questionados a respeito de seu desempenho escolar nas disciplinas ligadas à Matemática, quatro programadores assumiram ter seu desempenho escolar classificado como Insatisfatório, sendo que três deles foram reprovados na disciplina em questão e tiveram que refazê-la mais uma vez para cumprimento do respectivo crédito.

Uma observação interessante é que, embora vinte e cinco programadores tenham assumido um certo grau de satisfação nas disciplinas relacionadas à Matemática, o percentual cai para dezesseis quando questionados a respeito de seu desempenho em resolver problemas da Matemática em sala de aula. Entretanto, o número de profissionais que consideraram muito satisfatório seu desempenho nas disciplinas relacionadas à Matemática subiu de cinco para doze.

Em relação aos créditos técnicos, como já esperado para um time maduro em sua operação, não obtivemos nenhum grau de insatisfação, sendo que um pouco mais da metade confirmou ter tido um desempenho satisfatório em relação a seu desempenho em disciplinas técnicas dos cursos de graduação ou pós.

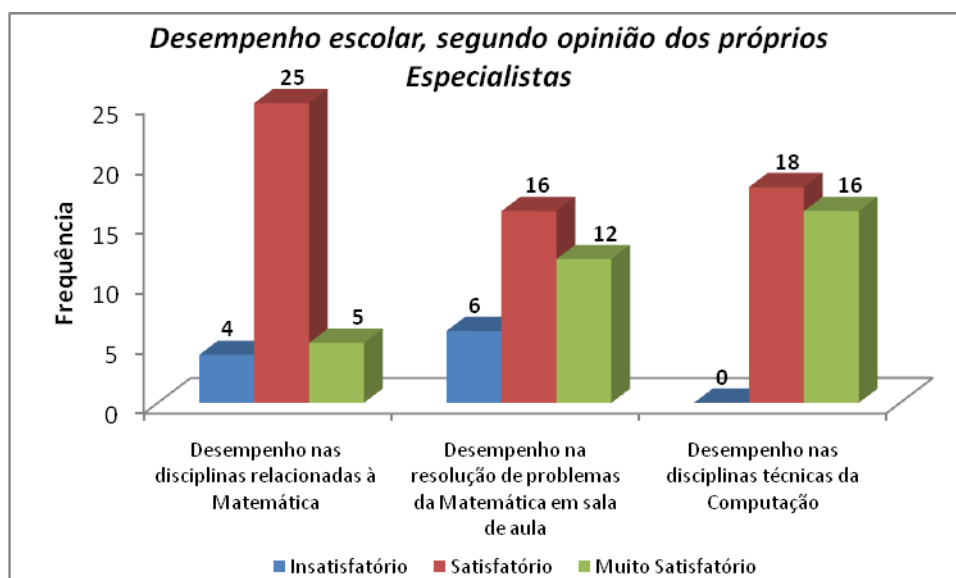


Gráfico 8: Desempenho escolar dos programadores participantes da segunda fase de coleta de dados.

Na próxima seção descreveremos os instrumentos de coleta de dados utilizado durante este segundo ciclo investigativo deste estudo.

## **5.2.2 Instrumentos para Coleta de Dados**

Como instrumentos para a coleta de dados no segundo ciclo desta pesquisa, utilizamos um roteiro para abordagem inicial, as interações capturadas pelo aplicativo GRUMPS, bem como entrevista via um questionário após a implementação da atividade proposta. Tais elementos são descritos mais detalhadamente a seguir.

### **5.2.2.1 Roteiro para Abordagem Inicial**

De forma similar à primeira etapa, também utilizamos como instrumento para a coleta de dados na primeira sessão do segundo ciclo desta pesquisa, um roteiro para abordagem inicial.

O objetivo deste roteiro, presente no Anexo 4 desta pesquisa, foi o de capturar algumas informações sobre o perfil de cada profissional participante.

Essa primeira sessão durou em média de trinta minutos por participante, e foi tomado o cuidado de explicar aos entrevistados que as observações e perguntas seriam pertinentes somente a este estudo e por isso não haveria motivos para constrangimentos ou omissões. Também combinamos que preservaríamos a identidade de cada pessoa, sem usarmos nomes específicos nesta pesquisa. Ainda salientamos que o objetivo do estudo não é apresentar aos locais de trabalhos dos programadores as respostas ou compartilhar as notas recolhidas por meio das interações ali gravadas: todo o material coletado será interpretado como confidencial pela pesquisadora.

Após toda essa explicação na primeira sessão, seguimos o roteiro para abordagem inicial, presente no Anexo 4 desta pesquisa com o intuito de coletar algumas informações históricas sobre cada profissional participante. Tais

informações são importantes não somente do ponto de vista de desconstruir o ambiente para início da observação do processo de programação do componente, mas também pelo fato de serem utilizadas mais à frente no processo de análise dos dados.

### **5.2.2.2 Interações Capturadas pelo Aplicativo GRUMPS**

As interações registradas pelo aplicativo GRUMPS foram utilizadas como principal fonte de dados para análise nesta segunda etapa.

Elegemos estes apontamentos como fonte principal de exploração dos dados porque, não somente possibilitavam registro contínuo das interações dos programadores em seu ambiente natural, mas também se mostrava um método bastante eficiente e consistente ao salvar, por longos períodos de tempo, todas as ações tomadas pelos programadores de forma não-invasiva.

O *Generic Remote Usage Measurement Production System* (GRUMPS, 2001) foi desenvolvido na universidade de Glasgow (Evans et al, 2003), e tem como objetivo principal possibilitar a coleta mais precisa dos dados, do ponto de vista quantitativo, e potencialmente fornecer melhores subsídios para a análise qualitativa dos dados. Segundo seus criadores, ao se utilizar este aplicativo, uma coleção de arquivos, vídeos ou observações registradas em um período de tempo limitado, pode ser substituída por detalhes mais minuciosamente coletados do computador do usuário investigado, como por exemplo, por meio de chamadas a programas, visitas às páginas da Internet, etc.

Evans et al (2003) descrevem que este tipo de aplicativo é chamado de REDDI (*Rapidly Evolving Digitally-Derived Investigations*). Para eles, “digitalmente derivado” (tradução nossa) corresponde ao fato do uso de computadores e equipamentos similares para a coleta automatizada dos dados. Entretanto, e em contraste com os arquivos de registro precusores, agora é possível fazer mudanças rápidas frente aos dados de fato coletados, respondendo de forma mais rápida a novos interessantes, opiniões e hipóteses do investigador. Por isso tais investigações potencialmente “evoluem rapidamente” (tradução nossa): elas

não dependem de dados estáticos coletados antes de a análise começar, mas sim caminham junto com eles.

Utilizamos a versão *Windows* do aplicativo GRUMPS nesta pesquisa para a segunda etapa e coleta dos dados a fim de não somente monitorarmos todas as ativações de diferentes janelas/programas, *clicks* de *mouse* e eventos de teclado, mas também capturamos os comentários feitos sobre o que os programadores pensavam (num intervalo de quinze minutos para captura de cada resposta) no decorrer de tais eventos. Esta versão do GRUMPS foi instalada nos computadores de cada um dos trinta e quatro participantes da segunda etapa e ficou visível na barra de tarefas do *Windows* de todos eles.

A cada quinze minutos uma janela surgia na frente do programador, questionando-o a respeito do que ele estava pensando naquele momento (“What are you thinking now?”, conforme exemplo na Figura 3). Foi dada ao programador a opção de escrever até trezentos caracteres no campo aberto de comentário, e em seguida, clicar na opção Salvar (*Save*) a interação em questão ou simplesmente escolher a opção Sair (*Quit*). Neste segundo cenário, um registro em branco era gravado como parte da sessão do especialista em banco de dados.

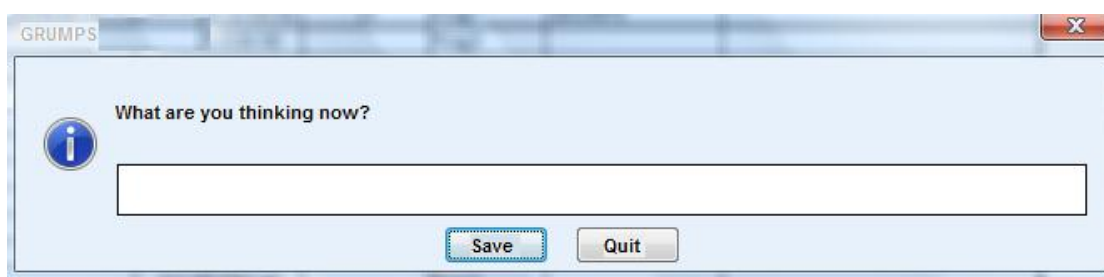


Figura 3: Exemplo de janela do aplicativo GRUMPS para coleta de opiniões parciais dos programadores

Do ponto de vista técnico, o repositório onde as informações são armazenadas foi projetado para conter duas principais tabelas: uma de sessões e outra para eventos/ações, conforme exemplo dado na Figura 4 a seguir. Uma sessão se iniciava no momento em que o programador ligava seu computador (normalmente somente ao início do dia de trabalho) e era finalizada ao término do horário comercial (fim do expediente). Em alguns casos, os programadores precisaram reiniciar suas máquinas e por isso, uma nova sessão era iniciada

também. Cada sessão possui um ou mais eventos/ações, que correspondem efetivamente aos aplicativos, sistemas ou páginas de Internet acessados.

SessionID	StartTime	EndTime	UserID	MachineID	UARExitReason
5253	1045142859063	1045144730173	87858268	bo715-11-02	User Logged Out

ActionID	Session	Time	XML	Type
1251079	5253	1045143002268	<p>adagide.exe</p> <wl>58</wl> <wt>62</wt> <wr>806</wr> <wb>568</wb> <ws>nor</ws>	9

Figura 4: Exemplo registros de Sessão (primeira tabela) e Evento/Ação (segunda tabela) extraídos do GRUMPS

Partindo dos eventos/ações realizados dentro de uma sessão, conseguimos extrair qual foi o programa acessado, alguns outros atributos deste programa (tais como coordenadas, tamanho da janela, comentários e opiniões capturados dos programadores), tempo gasto em cada um deles assim como o tipo de ação, que corresponde a um dos dez listados a seguir:

- Tipo 1: Solicitação de início de sessão (ativamento do GRUMPS ao ligar computador)
- Tipo 2: Tempo sem nenhuma interação via *mouse* ou teclado
- Tipo 3: Interação via teclado com o aplicativo (interação com campos da tela)
- Tipo 4: Interação via *clicks* de *mouse* com o aplicativo (interação com campos da tela)
- Tipo 5: *Tipo Configurável:* Ativamento do aplicativo para abertura dos diagramas e especificações técnicas-funcionais dos requisitos para mudanças do(s) programa(s) de computador pelos especialistas
- Tipo 6: *Tipo Configurável:* Ativamento do aplicativo para alteração do(s) programa(s) de computador pelos especialistas (interface de programação)

- Tipo 7: Tipo Configurável: Ativamento da execução do sistema ao qual o(s) programa(s) analisado(s)/alterado(s) pelos desenvolvedores pertence(m) para fins de testes
- Tipo 8: Tipo Configurável: Ativamento do aplicativo para empacotamento da versão final do(s) programa(s) alterado(s) pelos desenvolvedores (esta atividade marca o fim da manutenção do(s) programa(s) pelos especialistas)
- Tipo 9: Mudança de foco da janela (ativamento de um aplicativo)
- Tipo 10: Solicitação de término de sessão (fechamento do GRUMPS ao reiniciar ou desligar computador)

No exemplo dado na Figura 4, o programa *adagide.exe* foi ativado, com as coordenadas de janela especificadas no campo “XML” e tamanho da janela normal (não minimizado ou maximizado).

A coleta de dados dos participantes durante o segundo ciclo de investigação via o aplicativo GRUMPS foi conduzida de 23 de Novembro de 2010 a 29 de Novembro de 2010 nas instalações de seus ambientes de trabalho. Organizamos cinco sessões durante o expediente semanal de trabalho dos profissionais participantes desta pesquisa.

Apoiando-nos na GT como metodologia de trabalho, a razão principal para termos realizado somente cinco sessões práticas foi devido a este número de encontros ter se mostrado suficiente na saturação teórica das categorias emergidas a partir dos dados coletados. Foi possível constatar que a partir da quarta sessão, a contribuição dos dados coletados na imersão de novas categorias foi praticamente nula. Obtivemos a confirmação disso, quando na quinta sessão, praticamente todos os dados coletados puderam ser classificados a partir das categorias já definidas nas quatro sessões anteriores.

Com o objetivo de viabilizar, do ponto de vista operacional, a coleta de dados de aproximadamente oito horas e trinta minutos por programador por dia, a utilização do aplicativo GRUMPS foi fundamental no intercalonamento entre a

fase de coleta e a validação dos dados a fim de auxiliar no processo de decisão do ponto de parada de coleta dos dados (devido à saturação teórica das categorias), embora automaticamente capturados.

Ao final de cada dia de trabalho, os programadores eram solicitados a gravar uma cópia do arquivo de saída do aplicativo GRUMPS em diretórios pessoais presentes no dispositivo *pen-drive* da pesquisadora. A retomada do processo de manutenção dos programas acontecia normalmente como parte integrante do expediente diário de trabalho dos programadores.

A fase de preparação de dados envolveu conectar os devidos eventos/ações às sessões correspondentes, calcular a duração de cada um dos eventos/ações, encontrar o tipo de cada evento/ação realizado, bem como consolidar as considerações dadas a cada quinze minutos pelos programadores.

Action	What	TimeOffset	What are you thinking comments
1250975	UAR.exe	2	Teste integrado do portal de Internet para validação da implementação
1250978	Explorer.EXE	8	Busca dos arquivos de teste no laptop
1250983	iexplore.exe	16	Execução do plano de testes
1251031	Explorer.EXE	87	Simulação forçada dos valores para testes
1251032	eudora.exe	138	Checagem dos emails de teste recebidos
...			

Figura 5: Exemplo de um relatório detalhado mostrando as ações de uma sessão, aplicativos acessados e tempo gasto

Gostaríamos de reforçar que o aplicativo GRUMPS não é capaz de analisar a qualidade das ações tomadas pelos usuários, mas sim descreve, em detalhes, os eventos e ações realizados por eles por meio de registro contínuo das interações com o computador. Cabe ao investigador aproximar os resultados do contexto em questão e analisar qualitativamente os dados coletados.

Do ponto de vista da GT, concordamos com Seaman (2008), que afirma que um aspecto importante desta metodologia é que ela intercala as fases de coleta e análise/validação dos dados para fornecer um entendimento sobre o que ocorre na prática e as razões que explicam tais fatos.

Acreditamos nesta pesquisa que, baseado na dinamicidade e detalhamento dos dados coletados por meio do aplicativo GRUMPS e fundamentados na metodologia GT, obtivemos elementos suficientes para a explicação dos fenômenos propostos neste estudo.

### 5.2.2.3 Questionário Pós-Implementação

Ao final da sétima sessão do processo de coleta de dados deste trabalho, realizamos uma entrevista semi-estruturada, segundo a visão de Triviños (1987).

Seguimos um roteiro pré-definido de perguntas a fim de nos orientarmos, mas sem perdermos a liberdade dos participantes responderem com flexibilidade, permitindo uma conversa mais harmônica com os profissionais pesquisados. Por tais motivos, a entrevista semi-estruturada segundo Triviños (1987) foi o formato utilizado.

O questionário, presente no Anexo 5 desta pesquisa, foi dividido em três partes:

- Compreensão do Problema Dado: Nesta seção buscamos coletar a opinião dos entrevistados em relação à qualidade dos requisitos fornecidos e entendimento do problema inicial;
- Estabelecimento do Plano de Resolução: Procuramos capturar algumas das estratégias utilizadas para a resolução do problema inicialmente dado, incluindo julgamentos realizados na tentativa de estabelecimento de um plano de ação final, implementado nos programas de computador;
- Validação dos Resultados: Nesta última parte buscamos identificar de quais maneiras os participantes fizeram um retrospecto para a validação dos resultados obtidos, bem como descrever, do ponto de vista dos entrevistados, como os elementos da Matemática influenciam suas decisões e julgamentos na prática profissional.

Similarmente à primeira etapa, o objetivo de aplicarmos um questionário ao término das sete sessões não foi no sentido de avaliarmos a qualidade do produto final gerado, mas sim com a finalidade de termos outra oportunidade para detalharmos juntos (pesquisadora e pesquisados) os conhecimentos mobilizados

e o raciocínio envolvido para expressar a lógica do programador na solução concebida para o problema dado.

### **5.3 Critérios de Análise**

A fim de identificarmos as categorias relevantes para o processo de análise dos dados, passamos para a etapa de identificação dos significados das interações e respostas dos programadores-participantes com vistas à formulação de classes, sempre nos atentando ao quadro teórico de referência que adotamos neste estudo.

A fim de ilustrarmos o material inicialmente resgatado do aplicativo GRUMPS e organizado em ordem cronológica pela pesquisadora, tomemos como exemplo a amostragem a seguir no Quadro 3.

Conforme mencionado anteriormente, as janelas de questionamento sobre o que o programador estava pensando apareciam a cada quinze minutos, e estas reflexões foram compiladas na última coluna do relatório do Quadro 3. Para entendermos melhor a frequência das respostas dada à pergunta “O que você está pensando?”, tomemos como exemplo o participante Prog3.

Neste trecho a seguir extraído do aplicativo GRUMPS, a duração total da sessão do Prog3 foi de cerca de cinquenta e sete minutos (do início no dia 24/11/2010 às 13:01:28 até 24/11/2010 às 13:58:51) e ocorreram três interações de resposta à pergunta “O que você está pensando?”. No caso da ação 2229865 de “Acesso ao programa a ser alterado” foram acumuladas duas interações (uma aos quinze e outra aos trinta minutos) por ter sido iniciada no momento treze minutos e oito segundos (soma dos tempos gastos nas ações anteriores) e finalizada em trinta e oito minutos e dez segundos (duração acumulada até o final da ação).

As respostas e comentários à pergunta “O que você está pensando?” forneceram uma conexão interessante entre a ação (por exemplo, na sessão 158, ação 3253763, do Prog33 no horário entre 12:37pm até 01:35pm como sendo o

horário de almoço da pessoa) e o discurso (como por exemplo, na sessão 158, ação 2229868, no horário entre 1:39pm até 01:57pm, que evidencia que a partir de determinados testes, Prog3 se deparou com algo considerado complexo por ele, decidindo voltar certos passos para trás na tentativa de definir uma melhor solução ao problema dado).

Id da Sessão	Id da Ação	Tipo da Ação	Tipo da Ação (Comentário atribuído pela pesquisadora)	Data/Hora Início	Data/Hora Fim	Duração (seg)	Programador	Aplicativo Acessado	"O que você está pensando?"
17	1949001	Tipo 1	Início sessão	24/11/10 10:58:59	24/11/10 11:00:25	00:01:26	Prog7	GRUMPS	
17	1949002	Tipo 3	Teclado	24/11/10 11:00:25	24/11/10 11:00:28	00:00:03	Prog7	Windows Explorer	
17	1949003	Tipo 4	Mouse	24/11/10 11:00:28	24/11/10 11:00:29	00:00:01	Prog7	Windows Explorer	
17	1949004	Tipo 9	Janela Ativa	24/11/10 11:00:29	24/11/10 11:00:30	00:00:01	Prog7	MS Word	
17	1949005	Tipo 5	Acesso à Especificação Técnica	24/11/10 11:00:30	24/11/10 11:14:54	00:14:24	Prog7	MS Word	Resposta 1: Esse programa eu já mexi...é simples de alterar e já tenho o ambiente montado na minha máquina...tã tranquilo...
17	1949006	Tipo 4	Mouse	24/11/10 11:14:54	24/11/10 11:16:20	00:01:26	Prog7	Windows Explorer	
17	1949007	Tipo 10	Término sessão	24/11/10 11:16:20	24/11/10 11:17:38	00:01:18	Prog7	GRUMPS	
158	2229858	Tipo 1	Início sessão	24/11/10 13:01:28	24/11/10 13:02:03	00:00:35	Prog3	GRUMPS	
158	2229859	Tipo 3	Teclado	24/11/10 13:02:03	24/11/10 13:02:11	00:00:09	Prog3	Windows Explorer	
158	2229860	Tipo 4	Mouse	24/11/10 13:02:11	24/11/10 13:02:12	00:00:01	Prog3	Windows Explorer	
158	2229861	Tipo 9	Janela Ativa	24/11/10 13:02:12	24/11/10 13:02:21	00:00:09	Prog3	MS Word	
158	2229862	Tipo 5	Acesso à Especificação Técnica	24/11/10 13:02:21	24/11/10 13:13:09	00:10:48	Prog3	MS Word	
158	2229863	Tipo 4	Mouse	24/11/10 13:13:09	24/11/10 13:13:10	00:00:01	Prog3	Windows Explorer	
158	2229864	Tipo 9	Janela Ativa	24/11/10 13:13:10	24/11/10 13:14:36	00:01:26	Prog3	Windows Explorer	
158	2229865	Tipo 6	Acesso ao programa a ser alterado	24/11/10 13:14:36	24/11/10 13:39:38	00:25:02	Prog3	Visual Studio	Resposta 1: Nossa esse é brabo... Resposta 2: Hmm...o q eu pensei antes não vai dar...vou começar de novo...
158	2229866	Tipo 4	Mouse	24/11/10 13:39:38	24/11/10 13:39:39	00:00:01	Prog3	Internet Explorer	
158	2229867	Tipo 9	Janela Ativa	24/11/10 13:39:39	24/11/10 13:39:39	00:00:00	Prog3	Internet Explorer	
158	2229868	Tipo 7	Acesso ao sistema para teste	24/11/10 13:39:39	24/11/10 13:57:12	00:17:33	Prog3	Internet Explorer	Resposta 1: É...esquece...pensei besteira...vou "baixar" o sistema na minha máquina local de novo...
158	2229869	Tipo 4	Mouse	24/11/10 13:57:12	24/11/10 13:57:13	00:00:01	Prog3	Internet Explorer	
158	2229870	Tipo 3	Teclado	24/11/10 13:57:13	24/11/10 13:57:14	00:00:01	Prog3	Windows Explorer	
158	2229871	Tipo 4	Mouse	24/11/10 13:57:14	24/11/10 13:57:25	00:00:11	Prog3	Windows Explorer	
158	2229872	Tipo 10	Término sessão	24/11/10 13:57:25	24/11/10 13:58:51	00:01:26	Prog3	GRUMPS	
158	3253762	Tipo 1	Início sessão	26/11/10 12:37:01	26/11/10 12:37:36	00:00:35	Prog33	GRUMPS	
158	3253763	Tipo 2	Tempo Ocioso	26/11/10 12:37:36	26/11/10 13:35:12	00:57:36	Prog33	Nenhum	Resposta 1: <Branco> Resposta 2: <Branco> Resposta 3: <Branco> Resposta 4: "Rango"
158	3253764	Tipo 4	Mouse	26/11/10 13:35:12	26/11/10 13:36:29	00:01:18	Prog33	GRUMPS	
158	3253765	Tipo 10	Término sessão	26/11/10 13:36:29	26/11/10 13:37:43	00:01:13	Prog33	GRUMPS	

**Legenda:**

**Tipo 1:** Solicitação de início de sessão

**Tipo 2:** Tempo ocioso

**Tipo 3:** Interação via teclado

**Tipo 4:** Interação via mouse

**Tipo 5:** Ativamento do aplicativo para abertura dos diagramas e especificações técnicas-funcionais

**Tipo 6:** Ativamento do aplicativo para alteração do(s) programa(s) de computador

**Tipo 7:** Ativamento do sistema a qual o(s) programa(s) pertence(m) para fins de teste

**Tipo 8:** Ativamento do aplicativo para empacotamento do(s) programa(s) alterado(s)

**Tipo 9:** Mudança de foco da janela (ativamento de um aplicativo)

**Tipo 10:** Solicitação de término de sessão

Quadro 3: Exemplo de relatório consolidado pela pesquisadora a partir dos dados extraídos do aplicativo GRUMPS para algumas sessões e ações de 3 programadores participantes

Com base na primeira etapa da GT de Codificação Aberta e apoiados em Charmaz (2000, p.39) que ressalta que “codificar significa categorizar segmentos de dados com pequenos nomes que simultaneamente resumem e descrevem cada parte dos dados” (tradução nossa), iniciamos o exercício de reunir os códigos por suas similaridades ou diferenças conceituais. Após a delimitação das categorias, algumas subcategorias também surgiram.

A dinâmica utilizada em tal processo de delimitação, baseado na metodologia GT, constitui-se primeiramente em analisar os dados das transcrições do aplicativo GRUMPS (principalmente os campos de comentários ou “O que você está pensando?”), iniciando o processo de criação de códigos relacionados a trechos destas interações. Não utilizamos “categorias-semente” (*seed categories* – um conjunto inicial de códigos para começar a codificação), mas de fato, criamos códigos *in vivo* (oriundos da análise da própria citação) a partir das transcrições desta fonte de dados. A seguir ilustramos alguns exemplos de trechos extraídos do aplicativo GRUMPS onde identificamos os códigos *in vivo* entre colchetes ao longo dos comentários:

Programador	Data	Tempo Total da Sessão no dia (em horas)	Síntese das principais Ações capturadas no GRUMPS x Comentários "O que você está pensando?"
Prog4	23/11/2010	8,31	Tipo 5 (Acesso à Especificação Técnica) - Duração Total: 1h 13m 58s Tipo 6 (Acesso ao programa a ser alterado) - Duração Total: 3h 02m 17s Tipo 7 (Acesso ao sistema para teste) - Duração Total: 1h 41m 23s Outros Tipos - Duração Total: 2h 20m 58s  "Este sistema é bem parecido com os padrões (de programação e boas práticas) que definimos a pouco tempo atrás [ <i>Analogias</i> ] ...dá pra aplicar aqui o mesmo tipo de integração com o sistema de RH que fizemos na semana passada [ <i>Analogias</i> ][ <i>Organização do Pensamento</i> ] ..."
Prog23	24/11/2010	7,90	Tipo 5 (Acesso à Especificação Técnica) - Duração Total: 48m 19s Tipo 6 (Acesso ao programa a ser alterado) - Duração Total: 2h 43m 20s Tipo 7 (Acesso ao sistema para teste) - Duração Total: 2h 34m 42s Outros Tipos - Duração Total: 1h 47m 39s  "Eu to pensando em tirar toda essa parte de validação (dos campos obrigatórios na tela) desta camada (parte do programa) e usar só as 3 primeiras (rotinas de ) consistências (dos dados digitados) [ <i>Capacidade de decompor e recombinar</i> ] ..."
Prog17	26/11/2010	7,33	Tipo 5 (Acesso à Especificação Técnica) - Duração Total: 27m 13s Tipo 6 (Acesso ao programa a ser alterado) - Duração Total: 1h 31m 42s Tipo 7 (Acesso ao sistema para teste) - Duração Total: 3h 40m 49s Tipo 9 (Ativamento de um aplicativo) - Duração Total: 31m 13s Outros Tipos - Duração Total: 2h 08m 51s  "O teste q falta agora é o mais difícil (teste de integração) porque o outro sistema só me mandou uns registros de teste e vou ter q criar meus dados...isso pode não virar (dar certo)...porque to viciando os dados [ <i>Percepção das implicações e impacto na solução dada</i> ]"
Prog6	29/11/2010	9,58	Tipo 5 (Acesso à Especificação Técnica) - Duração Total: 05m 37s Tipo 6 (Acesso ao programa a ser alterado) - Duração Total: 2h 19m 24s Tipo 7 (Acesso ao sistema para teste) - Duração Total: 3h 35m 10s Tipo 9 (Ativamento de um aplicativo) - Duração Total: 53m 46s Outros Tipos - Duração Total: 2h 40m 51s  "Ao invés de dar o refresh na página (recarregar a página de Internet), to tentando fazer um esquema mais ninja pro usuário não precisar ficar esperando [ <i>Pensar diferente, "fora da caixa"</i> ] ..."

Quadro 4: Exemplo de registros na fase de Codificação Aberta

Esse conjunto inicial de códigos era revisado diariamente por esta pesquisadora, sempre após as dezoito horas, que correspondia ao final do expediente dos programadores e a entrega dos artefatos por eles produzidos durante o dia. Os códigos encontrados nas transcrições das interações capturadas pelo aplicativo GRUMPS (comentários e ações) foram agrupados de acordo com as suas propriedades, formando assim conceitos que representam categorias. A este processo, conforme a metodologia GT, é dado o nome de Codificação Aberta.

Os procedimentos da Codificação Aberta estimulam a constante criação de novos códigos e a fusão com os códigos existentes quando novos dados de evidência e interpretação emergem.

As categorias criadas foram analisadas novamente, e subcategorias foram identificadas com o objetivo de proporcionar uma maior clareza sobre o fenômeno em questão. Finalmente, as categorias e subcategorias foram relacionadas entre si, na etapa de Codificação Axial.

Na prática, os passos da Codificação Aberta e Axial se sobrepõem e se unem, devido à interatividade do processo. Os códigos e categorias identificados passaram por sucessivas revisões entre os códigos e as categorias levantadas, sendo que, ao final da presente versão da análise, foram produzidos doze principais códigos associados a quatro categorias:

<b>Categorias</b>	<b>Principais Códigos <i>In Vivo</i> Identificados</b>
<b>Busca Sistemática de Padrões</b>	Organização do Pensamento
	Analogias
	Base de conhecimento procedural/rotineiro
<b>Independência</b>	Capacidade de decompor e recombinar
	Familiaridade com as estratégias de solução aplicadas
	Questionamentos sobre os resultados a serem obtidos
<b>Julgamento</b>	Conexões com boas práticas de programação
	Decisões baseadas nas experiências profissionais anteriores
	Percepção das implicações e impacto na solução dada
	Auto-Regulação
<b>Originalidade</b>	Pensar diferente, "fora da caixa"
	Ser proativamente oportunista

Quadro 5: Lista de Categorias oriundas dos principais Códigos In Vivo identificados nesta pesquisa

A regra geral em GT é continuar o processo de coletar e analisar sistematicamente os dados até a saturação teórica ser atingida. Segundo Bandeira-de-Mello e Cunha (2006), esse estágio final ocorre quando ganhos marginais no poder explicativo da teoria para mais evidências coletadas é aproximadamente nulo.

Para o escopo desta pesquisa, conforme mencionado anteriormente, foi possível constatar que a partir da quarta sessão prática de coleta de dados, a

contribuição dos mesmos na imersão de novas categorias foi praticamente nula. Obtivemos a confirmação disso, quando na quinta sessão, praticamente todos os dados coletados puderam ser classificados a partir das categorias já definidas nas quatro sessões anteriores. Com o auxílio operacional do aplicativo GRUMPS, o intercalonamento entre a fase de coleta e a validação dos dados no processo de decisão do ponto de parada devido à saturação teórica das categorias foi atingido.

Então, uma vez realizadas novas coletas de dados, poderá ser executada a validação das proposições feitas com os resultados das Codificações Aberta e Axial, assim como poderão ser propostos e verificados novos conceitos, categorias e relacionamentos, até o momento de integração de uma teoria substantiva na Codificação Seletiva.

Finalmente, a partir desta terceira etapa aplicada da GT (a Codificação Seletiva), obtivemos o refinamento de todo o processo identificando a categoria central da teoria, a qual todas as outras estão relacionadas. Segundo Charmaz (2000), tal categoria central deve ser capaz de integrar todas as outras e expressar a essência do processo social que ocorre entre os envolvidos.

No caso desta pesquisa, a categoria central identificada foi a “Aproximações do Pensamento Matemático na Resolução de Problemas aplicado ao Processo de Manutenção de Software”, por tangenciar todas as demais, fornecer um poder explicativo e integrar todas as outras classes. Como resultado desta etapa, obteve-se o diagrama da Figura 6 (categorias conceituais ordenadas alfabeticamente):

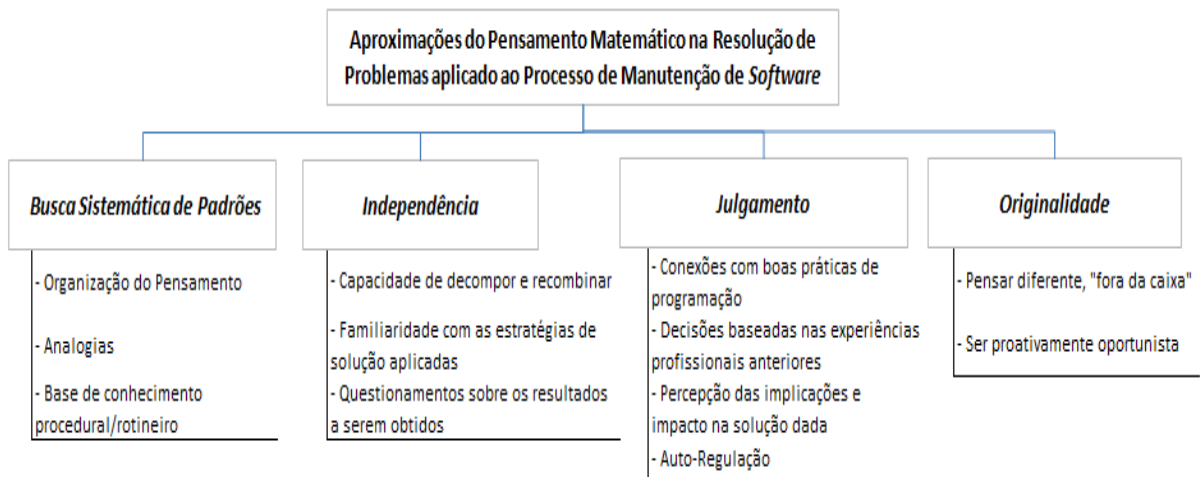


Figura 6: Síntese da categorização conceitual resultante da etapa de Codificação Axial da GT

Junto com as demais identificadas, tal categoria central foi importante e decisiva no ciclo de saturação teórica, que na visão de Charmaz (2000), corresponde a uma meta que o pesquisador deve perseguir para que, no limite de seus esforços e dos dados disponíveis, nenhum novo dado gere novas descobertas, sendo sempre associada a alguma outra categoria já existente.

Pelo fato de termos alcançado significativo grau de saturação em função das categorias central e conceituais elencadas anteriormente, no próximo capítulo, apresentaremos a análise dos resultados frente à fundamentação teórica adotada nesta pesquisa, dando continuidade à etapa final proposta pela GT: a interpretação dos dados.

## 6 ANÁLISE DOS DADOS

Nesta seção analisaremos as informações coletadas dos trinta e quatro programadores participantes da segunda etapa de coleta de dados desta pesquisa, a fim de tentarmos estabelecer uma aproximação entre as estratégias de resolução de problemas da Matemática, segundo Polya (1945, 1981 e 1995) e as estratégias de resolução de problemas utilizados por eles em sua prática profissional cotidiana.

Neste estudo, a *Grounded Theory* foi utilizada como uma ferramenta para auxiliar a realização da análise qualitativa. Baseado nos princípios de um experimento interpretativo, gostaríamos de reforçar que na GT o pesquisador assume que os comportamentos são melhores explicados pelos significados atribuídos às situações do experimento. Como parte integrante da GT, após a coleta e codificação, temos a fase de análise dos dados, a fim de apontar os eventos que sejam indicativos das categorias conceituais anteriormente estabelecidas visando explicar o fenômeno sob investigação.

Conforme descrito na seção anterior na Figura 6 - Síntese da categorização conceitual resultante da etapa de Codificação Axial da GT, adotaremos como critérios de estudo as quatro categorias conceituais de análise que correspondem à: busca sistemática de padrões, independência, julgamento e originalidade, conectadas à categoria central deste estudo, identificada como “Aproximações do Pensamento Matemático na Resolução de Problemas aplicado ao Processo de Manutenção de *Software*”.

A fim de minimizarmos problemas de ordem tendenciosa neste estudo, buscamos utilizar diferentes pontos de vistas de distintos programadores ao longo das citações capturadas e analisadas nesta seção. Portanto, no decorrer desta análise, não repetimos os mesmos programadores em nenhuma das quatro categorias conceituais, tentando fornecer diversas perspectivas de diferentes profissionais participantes deste estudo.

Uma vez que o quantitativo não foi o foco deste estudo, nem todas as interações de todos os programadores foram analisadas nesta seção. Entretanto, vale lembrar que devido à natureza qualitativa deste trabalho e também visto que

acreditamos ter alcançado a saturação dos dados ao longo das sete sessões realizadas na Etapa 2, cremos ter conseguido enxergar, via a categorização conceitual dos dados, que embora não analisadas no detalhe descrito neste capítulo, todas as colocações dos trinta e quatro programadores fornecem indícios de consistência, na mesma direção, dos comentários analisados.

Enfatizamos que a análise realizada neste estudo corresponde a uma aproximação teórica, visto que os profissionais observados e entrevistados não necessariamente conhecem os autores em que esta pesquisa se fundamenta e, do mesmo modo, sua formação acadêmica e profissional não necessariamente se apóiam nos mesmos princípios pedagógicos com os quais os autores que estamos nos baseando construíram seus modelos.

Nas citações, a transcrição dos dados corresponde exatamente à forma com que os participantes desta pesquisa interagiram com o aplicativo GRUMPS ou mesmo responderam ao Questionário Pós-Implementação. As notas entre parênteses correspondem às explicações dadas por esta pesquisadora para melhor contextualização de alguns momentos e entendimento de alguns termos técnicos da Computação.

## **6.1 Busca Sistemática de Padrões**

O objetivo principal desta categoria é levantar alguns dos elementos ligados à busca sistemática de padrões que os programadores lançaram mão ao investigar, analisar e implementar suas soluções. Pretendemos investigar alguns dos elementos imprescindíveis que, do ponto de vista dos programadores, são essenciais no momento de se pensar, comparar, propor e resolver um problema de seu cotidiano profissional. Em paralelo, indicaremos as potenciais similaridades ou diferenças com as estratégias de resolução de problemas na Matemática, segundo Polya (1945, 1981 e 1995).

Iniciaremos a análise dos dados exemplificando alguns dos primeiros comentários dados por Prog4. Após as devidas introduções presentes no roteiro para abordagem inicial da primeira sessão da Etapa 2, demos início ao processo

de análise dos comentários do programador Prog4 capturados por meio do aplicativo GRUMPS a partir da segunda sessão da Etapa 2, e observamos algumas das seguintes colocações dadas por este profissional:

Este sistema é bem parecido com os padrões (de programação e boas práticas) que definimos há pouco tempo atrás...dá pra aplicar aqui o mesmo tipo de integração com o sistema de RH que fizemos na semana passada [...].

[...] olhando a especificação (técnica-funcional) o que precisa ser feito aqui é basicamente mudar alguns parâmetros de chamada de um dos componentes (programas) já disponíveis e testar [...] ele (o componente) já está estabilizado e funcionando muito bem lá (no outro sistema).

Na tentativa de explicarmos e discutirmos parte do processo heurístico mencionado por Prog4 e os elementos que dele fazem parte, iniciaremos uma aproximação com Polya (1995), que no primeiro capítulo desta sua obra, criou um pequeno dicionário de Heurística, com sessenta e sete artigos, dando o significado e fundamentos de cada um deles.

O primeiro verbete que consta no dicionário é *Analogia*. Analogia é definida por Polya (1995) como uma relação de semelhança entre objetos distintos, ou seja, dois objetos são análogos se relações entre suas partes são coincidentes. Um exemplo da Matemática utilizado por ele descreve que dois ângulos opostos pelo vértice são iguais; dois diedros opostos pela aresta são iguais. Assim, ângulos e diedros são análogos, pois possuem relações semelhantes.

Para Polya (1995, p. 29):

Analogia é uma espécie de semelhança. Objetos semelhantes coincidem uns com os outros em algum aspecto; objetos análogos coincidem em certas relações das suas respectivas partes.

Concordamos com Polya (1995) e podemos inferir, baseado nas observações anteriormente destacadas nas falas e ações do programador Prog4, que perceber a semelhança de relações entre os objetos é uma experiência fundamental no caminho de se levar à descoberta da resolução de um problema Matemático ou não. Novamente citando Polya (1995, p. 29), também reforçamos a ideia de que:

A analogia permeia todo o nosso pensamento, a nossa fala cotidiana e as nossas conclusões triviais, assim como os modos de expressão artística e as mais elevadas conclusões científicas. Ela é empregada nos mais diferentes níveis. É comum o uso de analogias vagas, incompletas ou obscuras, porém a analogia pode alcançar-se ao nível do rigor matemático. Todos os tipos de analogia podem desempenhar uma função na descoberta da solução e, por isso, não devemos desprezar nenhum deles.

Ao conceber um plano de resolução para a situação dada, o Prog21 demonstrou familiaridade com problemas similares anteriormente já resolvidos e que não mereceriam aproveitamento total da solução aplicada. Ele explica:

Eu to vendo aqui que o visual (disposição dos campos) desse relatório é bem parecido com o da versão anterior do website de e-Business (site de Internet para comércio eletrônico) [...] só q não são de tecnologias completamente diferentes e não dá pra aproveitar quase nada...vou pegar só a ideia mesmo [...]

Dessa forma, a primeira aproximação das estratégias matemáticas que poderíamos efetivamente identificar como elemento fundamental no processo de resolução de problemas na manutenção de sistemas computacionais é, de fato, a habilidade de se criar analogias efetivas e utilizá-las no momento certo, a fim de atribuir significado à solução que se pretende adotar ao problema dado. Como um dos elementos essenciais, percebemos que a constante indagação e senso crítico sobre as analogias encontradas é fundamental, e esse processo pode conduzir a simplesmente descartar tais analogias em busca de novos caminhos, como no caso descrito por Prog21.

A resolução de um problema parece também depender das experiências e dos conhecimentos prévios de cada indivíduo, e dos objetivos que ele estabelece enquanto implementa a solução. Para ilustrar isso, tomemos como exemplo Prog10 e Prog19.

Somente para lembrarmos, Prog10 possui cerca de quinze anos de experiência na área de Computação, formado em Tecnologia em Processamento de Dados em 1996, e classificou como Muito Satisfatório seu desempenho nas disciplinas relacionadas à Matemática, na resolução de problemas da Matemática em sala de aula e também nas disciplinas técnicas da Computação.

Já Prog19, Analista Júnior, possui cerca de quatro anos de experiência na área de Computação, formado em Tecnologia em Engenharia da Computação em 2008 e classificou como Insatisfatório seu desempenho nas disciplinas relacionadas à Matemática e na resolução de problemas da Matemática em sala de aula, porém Satisfatório nas disciplinas técnicas da Computação.

Lembrando que independentemente da complexidade ou natureza dos problemas dados a estes dois profissionais (visto que se encontram em níveis de maturidade profissional distintos) foi possível evidenciar que, ao longo das cinco sessões que capturamos as observações e comentários destes programadores por meio do aplicativo GRUMPS, eles praticamente se comportaram de forma oposta: enquanto Prog10 gastava a maioria de seu tempo, cerca de 70%, focado nas ações do tipo 5 (especificação técnica-funcional), 6 (alteração do programa de computador) e 7 (testes do sistema alterado), Prog19 gastou praticamente o mesmo percentual de seu expediente de trabalho focado nas ações do tipo 2 (ocioso – computador sem execução de nenhuma atividade), 7 (testes do sistema alterado) e 9 (mudança constante de foco de janela), evidenciando uma certa desorientação em relação aos passos a serem seguidos para resolver o problema dado.

Outra evidência comparativa entre estes dois perfis é em relação ao número de vezes que a ação do tipo 8 (empacotamento dos programas alterados – potencial marco de fim da atividade de manutenção) foi executada: para Prog10 ao longo de uma semana de trabalho verificamos duas vezes, enquanto para Prog19 contamos nove vezes. Somente para fins de esclarecimentos, todo o momento que existia uma ação de empacotamento dos programas alterados, outro grupo de pessoas responsáveis por testes dos programas alterados era acionado, e por consequência, caso algum erro ou defeito no funcionamento do sistema fosse identificado, o(s) programa(s) alterado(s) era(m) enviado(s) de volta ao programador que originalmente o(s) modificou.

De certa forma, apoiando-se em Polya (1981), nota-se que na Matemática, a busca de estabelecimento de relações e argumentos lógicos, expostos de forma explícita e de um modo bastante preciso, poderia fazer emergir nos alunos de Ciência da Computação e afins questões relacionadas ao como eles seriam

capazes de retomar experiências passadas, e efetivamente utilizá-las em outras situações. Assim, apoiados na prática de rever outros casos do passado via constantes comparações, eles poderiam minimizar os efeitos negativos que a falta de controle e entendimento do problema pode trazer ao sistema em manutenção. Contudo, essa relação não foi percebida como explícita por parte de todos os programadores. Vamos explicar o porquê dessa nossa percepção.

O próximo verbete, *Conhece um problema correlato?*, está relacionado com o procedimento de estabelecer analogias, pois, ao procurar um problema que seja correlato ao que pretende-se resolver, tem-se que buscar relações semelhantes entre eles. Polya (1995, p. 26) relata:

Conhece um problema correlato? É difícil imaginar um problema absolutamente novo, sem qualquer semelhança ou relação com qualquer outro que já haja sido resolvido; se um tal problema pudesse existir, ele seria insolúvel. De fato, ao resolver um problema, sempre aproveitamos algum problema anteriormente resolvido, usando o seu resultado, ou o seu método, ou a experiência adquirida ao resolvê-lo. Além do que, naturalmente, o problema de que nos aproveitamos deve ser, de alguma maneira, relacionado com o nosso problema atual. Daí a pergunta: Conhece um problema correlato?

Ao estar diante de uma proposição que deve ser demonstrada ou refutada, supondo que se tenha uma compreensão global dessa, Polya propõe que se deve compreender, e atentamente se considerar, a sequência de ideias e os diversos aspectos do problema dado. À luz desta colocação de Polya (1995), vamos analisar a situação colocada por Prog13:

Acho q o q estou fazendo aqui não vai impactar muitas outras coisas nas outras funcionalidades do sistema [...] pra dizer a verdade, não conheço muito dessa base (banco de dados onde as informações do sistema são armazenados) e me jogaram um negócio pra fazer pra “ontem” aqui [...]

Ou seja, concordando com Polya (1995), podemos considerar que Prog13 carecia de uma formulação mais firme do que efetivamente sua experiência na manutenção de outros sistemas poderia contribuir na solução do problema que tinha em mãos, que provavelmente só foi totalmente alcançado depois, conforme evidenciado e capturado pelo aplicativo GRUMPS via ações (atividades do tipo 6

e 7 – alteração do(s) programa(s) e testes foram predominantemente as principais ações realizadas por Prog13 durante o dia 26/11/2011) e comentários inseridos pelo próprio especialista:

Nossa, demorou, mas foi [...] (essa foi a) primeira vez q altero esse sistema e apesar da modificação não ser super complexa, eu não imaginei em gastar tanto tempo assim [...]

Inclusive, ao confrontarmos Prog13 durante a aplicação do Questionário de Pós-Implementação (presente no Anexo 5 deste trabalho) sobre sua opinião a respeito dos requisitos fornecidos e de que forma ele julgou ter compreendido o problema dado, ele respondeu:

Estava claro que eu precisava “passar” (transportar) alguns novos dados de uma interface para outra e claramente isso estava descrito na especificação [...] eu vi que estava certo no momento que imaginei e testei por minha conta as alterações que eu fiz na parte da base (banco de dados onde as informações do sistema são armazenados) junto com as modificações que seriam feitas por outro programador [...] inclusive deixei documentado dentro da função do banco (programa pertencente ao sistema) como ele precisará passar os dados para que dê tudo certo.

Gostaríamos de salientar que em nenhum momento estamos enfatizando que a Matemática para resolução de problemas práticos é a única solução fundamentalmente disponível para reforçar o pensamento análogo nos profissionais da Computação. Pelo contrário, o que estamos enfatizando nesta análise é que expor tais profissionais cada vez mais aos conhecimentos matemáticos e científicos durante sua formação, pode contribuir substancialmente para este tipo de profissional em sua carreira.

Alinhado a este pensamento e baseado nas informações dadas pelos próprios programadores durante a fase de abordagem inicial da Etapa 2, podemos dizer que há indícios que os que relataram ter tido desempenho Insatisfatório na resolução de problemas da Matemática em sala de aula, também apresentavam o mesmo desempenho ruim nas disciplinas relacionadas à Matemática (Prog4, Prog11, Prog19 e Prog21). Certamente novos trabalhos neste campo devem contribuir na investigação e ampliação dessa relação. O que buscamos nesta pesquisa não é afirmar que o desempenho em compreender o

problema dado, qualquer que seja a complexidade ou natureza do mesmo, bem como propor soluções para os mesmos, está ligado somente à formação acadêmica dos profissionais, seu desempenho escolar ou mesmo em função apenas de sua experiência profissional. Buscamos sim contribuir para tal ampliação com os elementos emergidos da investigação realizada neste trabalho, e que parecem relevantes para este perfil de profissional.

Além disso, a fim de investigarmos mais profundamente tais indícios, os mesmos quatro programadores, por meio de evidência coletadas nas ações do aplicativo GRUMPS, também tiveram dificuldades em encontrar soluções completas aos problemas dados a cada um deles.

Para ilustrar essa afirmação, tomando como base o tempo total gasto pelos quatro especialistas na semana de trabalho observada, podemos dizer que o tempo gasto em ações do tipo 5 e 6 (especificação técnica-funcional e alteração do(s) programa(s), respectivamente) não foi predominante durante tais dias. De fato, tais especialistas gastaram grande parte de seu expediente de trabalho em ações do tipo 7 (testes), a qual podemos pressupor uma tentativa de aplicar alguma solução ao problema inicialmente dado, e posterior técnica de tentativa-e-erro ou mesmo ações do tipo 2 (ocioso – computador sem execução de nenhuma atividade), a qual podemos assumir falta de interesse, naquele instante, em compreender/buscar alternativas de resolução para o problema dado ou momentos pensativos dos pesquisados, não interagindo com o computador.

Buscamos por meio desta pesquisa destacar que a Matemática, como ciência dedutiva e sistemática, pode contribuir da mesma forma que a Matemática como ciência indutiva e experimental. Para nós, as possibilidades e limitações de ambos os lados podem contribuir para um maior aprofundamento na compreensão dos problemas dados aos programadores, sua capacidade de questionar, fazer comparações e estabelecer um plano para resolução do problema dado, quer seja na Matemática ou na prática profissional de alguma área da Computação.

A abstração da essência de um problema, matemático ou prático-computacional, para a identificação de suas variações análogas em busca de

padrões por meio de argumentos lógicos parece fornecer razões suficientes para comprovar a conexão existente entre o papel das estratégias de resolução de problemas da Matemática, segundo Polya (1995), e a ordem/organização do pensamento dos profissionais da Computação na busca de similaridades em sua base de conhecimento, especialmente para aqueles que executam manutenção de programas.

Enfatizamos especialmente os programadores que realizam alterações de programas pré-existentes, porque a complexidade do ambiente que os cercam, cria condições ainda mais explícitas para se conectar ao pensamento matemático preciso, no sentido de evitar inserções de novos problemas aos cenários já existentes no sistema computacional.

## **6.2 Independência**

Nossa intenção nesta categoria é capturar alguns elementos oriundos das observações e respostas dos especialistas, por meio das ações e comentários a partir do aplicativo GRUMPS e Questionário Pós-Implementação. Pretendemos investigar a capacidade de independência de tais programadores em analisar, compreender e propor soluções às situações dadas a eles em seu cotidiano profissional.

O critério de independência está relacionado às demonstrações de autonomia e senso crítico, no sentido de familiaridade com as estratégias de solução aplicadas, bem como o questionamento sobre os resultados a serem obtidos na implementação das mesmas. Buscamos evidenciar como tais manifestações potencialmente se aproximam ou não das estratégias de resolução de problemas na Matemática, segundo Polya (1945, 1981 e 1995).

Em relação aos questionamentos levantados por Prog2 para os possíveis resultados a serem obtidos pelas soluções propostas por ele, por exemplo, extraímos os seguintes comentários do aplicativo GRUMPS:

O resultado da query (pesquisa no banco de dados) que eu fiz pra mostrar os registros nesta busca (funcionalidade do site para pesquisa de registros

financeiros de ativos da companhia) traz um total de 10372,33 (reais) pra a soma de todos os lançamentos de ativos em Maio para 576 registros [...] preciso checar se isso ta certo [...] só que dá muito trabalho (checar) um a um (dos registros) [...] bom, se estivesse errado esta soma dos ativos (que corresponde à funcionalidade alterada que precisa ser testada), o balanço de fechamento de Maio (outra funcionalidade do site para reconciliação) mostraria uma diferença, já que o total é ativo + passivo [...] então, como no relatório de discrepância (do fechamento de Maio) não aparece nada e tem somente 10 registros de passivo, se eu verificar todos estes 10 e estiverem certos, entao a parte de ativo estará certa tb [...] vai me economizar um bom tempo ☺ [...]

Aproximando de Polya (1981), podemos estabelecer certa relação com os conceitos de demonstração por absurdo visto que, segundo este pesquisador, temos que:

A demonstração por absurdo mostra a falsidade de uma suposição derivando dela um absurdo flagrante. É um procedimento matemático que é exagerado e repetido até conduzir a um manifesto absurdo. (tradução nossa)

Ou seja, por meio dos comentários de Prog2 foi possível evidenciarmos que o mesmo apresentou em sua atividade heurística, alguns aspectos oriundos do método de demonstração por absurdo como instrumento de descoberta e questionamento sobre suas recomendações de solução para o problema dado.

Ao provar por contradição a existência de um elemento com determinada característica (erro no total dos registros de ativos da companhia), sem no entanto mostrar tal elemento (conferir se o total estava correto por meio da verificação um a um dos registros), Prog2 acabou por utilizar uma estratégia Matemática de prova por demonstração indireta ao absurdo, onde, de forma bastante sucinta, na tentativa de se provar algo (total dos registros de ativos está correto?), assume-se como verdade o contrário do que queremos provar (reconciliação de ativo e passivo estaria errada), para então se chegar a uma contradição (como reconciliação está certa e se a soma dos dez registros de passivo estiver correta, então a soma dos 576 registros de ativo estará correta também, baseado nos conceitos de contrapartida de lançamentos de ativo e passivo da Contabilidade).

Já Prog18, quando perguntado, por meio do Questionário Pós-Implementação, sobre seus critérios para selecionar determinado caminho, eliminando outras alternativas de alteração do programas no processo de manutenção do sistema, comentou:

Estou estendendo o cadastro de aplicações (incluindo nova funcionalidade no sistema sob manutenção) para relacionar um contrato financeiro a uma ou mais aplicações do inventário [...] mas parece que está faltando algo aqui na especificação [...] assumindo que um contrato financeiro não se liga a somente uma área de negócio, mas sim sempre a várias, e uma aplicação também pode ser usada por várias áreas de negócio ao mesmo tempo, então uma aplicação não pode mais estar ligada às áreas de negócio diretamente, e sim via os contratos financeiros a qual pertence. [...] então eu vou acrescentar um controle de lista de múltiplas seleções (campo na tela para selecionar mais de um valor) e também retirar a referência direta da área de negócio com a aplicação, já que esta conexão será via os contratos a qual tal aplicação pertence.

Analisando as ações e sequência descritas por Prog18, podemos conectá-las às seguintes observações de Polya (1995, p.58):

A demonstração indireta estabelece a verdade de uma afirmação por revelar a falsidade da suposição oposta. Envolve o princípio do terceiro excluído, um axioma fundamental da Lógica que afirma que, ou  $A$  é verdadeiro, ou  $A$  é falso, onde  $A$  é qualquer proposição passível de análise. Em essência, o axioma exclui qualquer estado intermediário entre a veracidade e a falsidade de  $A$  [...] na demonstração indireta, tem-se que provar que se  $B$  for falso, então  $A$  será falso, isto é, tem-se que supor  $B$  falso e deduzir que  $A$  será falso.

Em nossa análise, embora Prog18 não tenha encontrado na especificação técnica-funcional uma referência explícita para retirada da conexão do campo de aplicação com a área de negócio e conseqüentemente, nova conexão da aplicação com um ou mais contratos financeiros, a partir de seus questionamentos em relação ao que foi funcionalmente requerido (objetivo da alteração do programa), ele conseguiu examinar a essência do problema dado, e chegar a uma conclusão lógica das ações necessárias para se obter o resultado esperado.

Conseguimos também perceber em Prog23 que, a partir de sua capacidade de decompor os elementos da situação atual a qual o programa se encontra, ele parece ter conseguido recombina os diversos aspectos do problema dado para compor uma nova solução:

Estou pensando em tirar toda essa parte de validação (dos campos obrigatórios na tela) desta camada (parte do programa) e usar só as 3 primeiras (rotinas de) consistências (dos dados digitados) [...]

Tanto Prog18 quanto Prog23 parecem ter se utilizado de argumentos indiretos para trazer à tona uma nova realidade requisitada ao sistema (“retirar a referência direta da área de negócio com a aplicação, já que esta conexão será via os contratos a qual tal aplicação pertence” ou “estou pensando em tirar toda essa parte de validação e usar só as 3 primeiras consistências”). Essas situações parecem indicar uma crescente necessidade desse tipo de profissional em saber trabalhar com recomposições a partir de uma situação originalmente dada, ou para ser capaz de criar novas sequências das proposições inicialmente disponíveis nos programas pré-existentes.

Para Polya o pensamento matemático não está relacionado apenas com axiomas, definições e demonstrações rigorosas, mas também com analogias, induções, conjecturas, relações, generalizações e outros processos mentais. Nessa perspectiva, ele destaca estes como verdadeiros representantes dos fundamentos básicos das operações mentais envolvidas na resolução de problemas, principalmente os de Matemática.

De forma similar a esta ideia de extensão do pensamento matemático à resolução de problemas e analisando os comentários de Prog18 e Prog23, concordamos com Polya (1981), que descreve que embora a tarefa de estruturar o pensamento humano em um número finito de passos pareça quase mecânica, ela é de fato, uma forma matemática de pensar, e deve ser levada em consideração no ensino de Matemática de qualquer profissional.

Baseado em alguns indícios coletados neste trabalho, acreditamos que tal forma matemática de pensar, também parece bem adequada ao tipo de especialista em Computação que focamos nossa pesquisa.

Outro elemento oriundo dos dados coletados, o qual foi possível aproximarmos dos pensamentos expostos por Polya (1981), foi em relação à habilidade dos especialistas em desmembrar e reconstruir, em que novas soluções são dadas a problemas similares, anteriormente conhecidos. Foi possível de constatarmos isso também nos comentários do arquiteto de sistemas Prog3, capturado por meio do aplicativo GRUMPS:

Existe um módulo (programa pré-existente a ser reutilizado) de usuários disponível no *framework* (repositório de padrões, programas ou módulos) para tratamento de *login* (processo de entrada do usuário no sistema) [...] só que ele serve para sites *Intranet* (sites acessados somente nos limites da própria organização) e não para sites *Internet* (sites externos, acessados por toda a comunidade na *web*) [...] Estou construindo um novo (módulo de *login*), exclusivo para sites Internet, mas me baseando no de *Intranet*, reforçando principalmente a parte de segurança, que é bem diferente e mais rigorosa [...]

Polya (1981) ainda reforça que a capacidade de decomposição e recomposição das partes de uma solução a fim de aplicá-la em outro cenário, nem sempre é elementar, e deve ser objeto de constante investigação pelos alunos.

O processo de reconstituição e recombinação das partes frente a uma nova realidade segundo relações já conhecidas, constitui-se “num movimento matemático contínuo de encadeamento lógico de idéias precisas” (Polya, 1981, tradução nossa), o que parece evidenciar também o caminho crítico trilhado por Prog26, conforme seguinte trecho de resposta ao Questionário Pós-Implementação – Etapa 2:

Quando eu mudei a chamada de uma das rotinas (programa pré-existente), e começou a mostrar um erro na execução (do sistema), quase apavorei [...] porque é um sistema que ninguém quer mexer muito, toda vez que alteram dá problema e sempre impacta a performance de entrega do analista (programador) que trabalha nele [...] mas fui persistente e consegui finalizar com sucesso [...] vamos ver da próxima vez (risos) [...]

Acreditamos que existe uma aproximação entre as estratégias de resolução de problemas da Matemática, segundo Polya (1995), e as estratégias utilizadas pelos programadores em relação a sua proposta de solução ao

problema dado. Esta aproximação ocorre no que Polya (1995) explica como a fase de estabelecimento de um plano, um conjunto de indagações e sugestões que, após contínuas discussões na mente do próprio indivíduo, tendem a provocar o surgimento de uma ideia pertinente ao escopo e potencialmente solucionadora do problema dado.

As indagações e argumentos expostos pelos especialistas analisados nesta subseção também parecem refletir alguns aspectos evidenciados em Tall (2001) na investigação sobre o pensamento matemático avançado (AMT), acerca de deduzir elementos de forma mais abstrata, ou seja, pensar mais genericamente (real necessidade do usuário do sistema, muitas vezes implícita nas especificações técnicas-funcionais) do que somente localmente (retendo-se somente à documentação disponibilizada). Acreditamos que isso, alinhado ao pensamento de Polya (1995), também faz parte das dimensões de estabelecimento de um plano para resolução do problema dado.

### **6.3 Julgamento**

O objetivo desta categoria conceitual é trazer à tona alguns dos aspectos procedentes das observações e respostas dos especialistas, por meio das ações e comentários a partir do aplicativo GRUMPS e Questionário Pós-Implementação, frente ao julgamento que exerceram ao perceber e entender o problema dado, bem como melhor decidiram, entre várias, a mais adequada solução que deveria ser aplicada às situações dadas a eles em seu cotidiano profissional.

O critério de julgamento está relacionado às conexões com boas práticas de programação, percepção proativa de implicações diversas da solução dada ao sistema maior, assim como aspectos de auto-regulação utilizados pelo programador durante o processo de busca da solução, que acreditava ser a mais adequada para o cenário em questão. Pretendemos investigar como tais manifestações potencialmente se aproximam ou não das estratégias de resolução de problemas na Matemática, segundo Polya (1945, 1981 e 1995).

A respeito de conexões com boas práticas de programação, e como elas influenciam na determinação de um caminho ou outro a ser seguido, foi possível verificarmos, que a percepção de qualidade da alteração inserida no programa foi decisiva ao julgar uma solução como a mais adequada, especialmente a longo prazo. De acordo com o que Prog7 e Prog24 explicaram, uma solução sustentável reflete não somente a necessidade do momento presente, satisfazendo os requisitos solicitados pelos usuários hoje, mas principalmente olhando a facilidade e posterior manutenção do mesmo programa por outros especialistas no futuro.

O analista Prog7 menciona em uma de suas interações por meio do GRUMPS:

Não adianta eu colocar fixo no código (programa) este parâmetro, se mais pra frente eu mesmo posso ter q dar manutenção pra ele [...] fora isso, olhando pro q eu alterei na semana passada, qdo o usuário trocou de ideia, foi bem mais tranquilo de alterar o programa pra expandir o escopo do relatório.

Já o arquiteto Prog24 contemplou em uma de suas interações via GRUMPS:

Não sei bem se pros outros parece frescura, mas pra mim é questão de organização [...] não vou deixar isso aqui assim [...] no final sobra pra gente mesmo porque os padrões de qualidade não foram seguidos e acaba impactando, por exemplo, na performance do sistema.

Posteriormente, durante a fase de aplicação do Questionário Pós-Implantação, foi possível esclarecer com Prog24 que sua referência ao comentar “não vou deixar isso aqui assim” correspondia a sua indignação em função de como uma parte de programa tinha sido tão mal desenvolvida. Basicamente, ele destacou como a principal razão para esta indignação os elementos fixos (não configuráveis) diretamente definidos no programa, embora a boa prática definida no departamento zele por manter o máximo de elementos em forma de parâmetros configuráveis via interface do sistema, possibilitando maior flexibilidade ao *software*.

A respeito do caráter auto-regulador, primeiramente gostaríamos de ressaltar que nesta pesquisa utilizaremos a definição de Perrenoud (1999), que argumenta que este consiste “nas capacidades do sujeito para gerir ele próprio

seus projetos e progressos diante das tarefas e obstáculos”. Este autor ainda destaca que todas as pessoas possuem certo grau de auto-regulação e, que com a aprendizagem escolar e profissional, as pessoas devem buscar elevá-lo, o que certamente favoreceria uma autonomia progressiva no aprender e praticar na vida. Perrenoud (1999, p. 96) ainda complementa que:

[...] para aprender, o indivíduo não deixa de operar regulações intelectuais. Na mente humana, toda regulação em última instância, só pode ser uma auto-regulação: nenhuma intervenção externa age se não for percebida, interpretada e assimilada por um sujeito.

A partir desta definição do termo auto-regulação, foi possível identificarmos algumas reflexões de Prog12, Prog28 e Prog31 acerca de seus processos cognitivos ao adquirirem, organizarem e transformarem as informações intencionalmente, com vistas ao objetivo previamente estabelecido de resolver o desafio profissional que a eles foi dado. Prog12 fez o seguinte comentário via GRUMPS a respeito de suas táticas para enfrentar o imprevisto, o inesperado e a incerteza do problema dado:

É a primeira vez que vou alterar este programa e vou começar olhando o que ele faz primeiro, porque não quero impactar outra parte q está funcionando direito [...] tem umas coisas aqui que nunca vi, mas acho que entendi como funcionam [...] antes de modificar, vou salvar alguns *backups* (cópias) primeiro e ficar comparando com a versão anterior pra ter certeza de q não causei outros problemas na aplicação.

Sem dúvida, a autonomia e o agir de forma crítica em relação ao problema disponibilizado são virtudes que possibilitam ao profissional modificar seu pensamento, ajustar-se para re-elaborar e redirecionar seus esforços quando julgarem necessário. E a respeito disso, Prog28 comentou nas suas interações capturadas via aplicativo GRUMPS:

É bem verdade que já gastei um tempo nisso, mas parece que vou abortar (desistir) esta linha de pensamento porque o arquivo tá sendo criado sim [...] (anteriormente Prog28 achava q o erro no programa era devido a um arquivo de configuração que faltava em seu computador)

Já Prog31 explanou também sobre a influência do clima organizacional e do time de trabalho que também incentivam e encorajam os próprios especialistas

a privilegiar conscientemente o comportamento de abandono de determinadas rotas de resolução e, a partir da reflexão, perceber que a iniciativa em enfrentar os desafios profissionais também depende da persistência e autoconsciência de cada um deles, frente a algo que não está saindo conforme o esperado:

Gastei um tempo conversando com outros colegas para entender o porquê a adição de um campo na tela de cadastro de contatos afetou a parte de contratos financeiros [...] entendi que isso aconteceu devido a uma ligação na base de dados (relacionamento entre contratos financeiros que possuem um ou mais contatos) [...] agora estou no caminho certo e inclusive já testei localmente a solução que vou fazer o teste integrado mais tarde.

Concordamos com Schunk (2000), que em nossa visão também se encontra alinhado à Polya (1995), ao afirmar que as estratégias matemáticas aplicadas à programação podem constituir fonte de constante regulação no sentido de continuamente provocar os programadores a modificar, melhorar e agir eficientemente, até que o problema seja resolvido com sucesso. Conseguimos evidenciar isso na seguinte ponderação de Prog1 a respeito de sua perseverança e tempo de reação na identificação de um erro no programa alterado (posteriormente identificado como a passagem de parâmetros incorretos para outro programa):

Mas se eu to passando os dados (enviando os parâmetros requeridos) pra essa função (outro programa) não sei porque ela me retorna um erro desse [...] (na próxima interação após quinze minutos, Prog1 complementa) [...] Achei o q era: o tipo das variáveis (formato dos campos enviados) estavam truncando (limitando o tamanho) o valor passado [...]

Acreditamos que existe uma aproximação entre as estratégias de resolução de problemas da Matemática, segundo Polya (1995), e as estratégias utilizadas pelos programadores em relação a seu julgamento no momento de propor uma solução ao problema dado. Esta aproximação ocorre no que Polya (1995) explica como a fase de retrospecto da resolução completa, reconsiderando e re-examinando o resultado matemático final e o caminho que levou até este.

Gostaríamos de destacar uma diferença no que Polya (1995) menciona sobre o retrospecto acontecer sempre após “chegarmos a uma solução do problema e escrita a demonstração”. A aproximação em termos de objetivo final

entre as estratégias de resolução de problemas na Matemática, segundo Polya (1995) e as utilizadas na prática pelos programadores, indicam como fim a consolidação e aperfeiçoamento da capacidade de resolver problemas. Entretanto, baseado nas informações coletadas durante a Etapa 2, foi possível verificarmos que nem sempre a “solução completa e demonstrada” precisa estar totalmente pronta para desencadear o processo de retrospecto.

Para ilustrar essa situação, tomemos como exemplo o Prog5:

Acabei grande parte do dev (acrônimo de *development*, traduzido por esta pesquisadora como desenvolvimento) e vou testar esta parte antes porque se ela já não der certo, tenho que pensar em outra (solução) [...] (na interação após quarenta e cinco minutos, Prog5 complementa) [...] é, não dá pra aplicar isso aqui [...] melhor ir por outro caminho antes que eu gaste muito mais tempo tentando consertar esse.

Quando falamos a respeito de analogias na categoria conceitual de “Busca Sistemática de Padrões” deste trabalho (subseção 6.2), enfatizamos e concordamos com Polya (1995), que ao nos depararmos com um problema a ser resolvido, a pergunta *Conhecemos um problema correlato?* deve ser feita. Entretanto, olhando do ponto de vista desta nova categoria de “Julgamento”, é nosso entendimento que Polya (1995) não levou em consideração o fato de que as próprias soluções ou respostas dadas podem estimular a busca de analogias, e não somente os problemas em si.

Em outras palavras, se quisermos aplicar suas recomendações precisamos ter uma perspectiva do assunto e olhá-lo de um ângulo diferente para revelar uma provável analogia. Portanto, não basta simplesmente afirmar que existe analogia entre problemas, e sim, devemos buscar elementos que a confirmem. Embora Polya (1995) relate sobre o retrospecto de uma solução para verificação de sua adequação ao problema dado e reforço para posteriores analogias, isso ainda não completa um ciclo de julgamento proativo (antes de se aplicar uma solução) que busca restringir aspectos superficiais que a aparência sugere.

A exemplo disso, tomemos a colocação do Prog22 ao ser questionado, ao final das sessões de coleta de dados, sobre de que maneira ele acreditava ter

examinado a solução obtida, e qual critério usou para estabelecer que ela era suficiente para endereçar todos os requisitos iniciais:

Eu sabia que para esta alteração (de programa) muitos outros membros da equipe já tinham tentado resolver o problema de busca no site. E daí até para não afetar mais a moral do time, eu decidi pegar para resolver [...] O mais intrigante sempre foi que embora esse site seja bem semelhante (em termos de funcionalidade) aos demais utilizados na mesma área (de usuários), pela complexidade da forma com que foi desenvolvido, era bem provável que não conseguiríamos aplicar a mesma solução. Então comecei praticamente do zero [...]

Portanto, embora em constante busca de similaridades com problemas anteriores baseados em sua experiência profissional ou destreza em combinar e recompor elementos de problemas anteriores a fim de solucionar o atual, o critério de julgamento deve ir além da estrutura do problema, aliando outros elementos e percepções de potenciais impactos quando determinadas opções são selecionadas.

Acreditamos que esta ampliação do estudo da obra de Polya (1995) reforça ainda mais a importância do pensamento na atividade de Matemática e seu potencial caráter influenciador no processo de programação e manutenção de *software* na busca decisiva da mais adequada solução a um problema em particular.

#### **6.4 Originalidade**

Temos por intenção nesta categoria descrever alguns elementos procedentes das observações e respostas dos especialistas, por meio das ações e comentários a partir do aplicativo GRUMPS e Questionário Pós-Implementação, a respeito da categoria conceitual originalidade.

O critério de originalidade está relacionado a aspectos de inovação ou identificação de oportunidades proativamente pelos programadores durante o processo de busca da solução, que acreditavam ser a mais adequada para o cenário em questão na manutenção de um sistema computacional. Pretendemos

investigar como tais manifestações potencialmente se aproximam ou não das estratégias de resolução de problemas na Matemática, segundo Polya (1945, 1981 e 1995).

Para Prog6, por exemplo, conforme capturado na última sessão por meio do Questionário Pós-Implementação, ele acredita que a criatividade é papel fundamental do processo de desenvolvimento de programas pelas seguintes razões:

[...] sendo um bom programador, como você pode se conformar a limitar sua solução a somente o que foi pedido, sabendo que você pode entregar mais?  
[...] claro que tem outros aspectos aí no meio em relação ao prazo de entrega, principalmente, mas ter liberdade para criar significa fazer mais, no mesmo tempo e com maior qualidade [...]

Analisando as falas do Prog6 na perspectiva de inovação, podemos identificar que ele valoriza o “[...] criar significa fazer mais [...]” e parece colocar em prática predominantemente elementos que refletem as ações de recordar, aplicar e criar, pois estima estas como atitudes nobres e imprescindíveis para especialistas desta área.

Do ponto de vista mais abrangente, a manutenção de *software* deve considerar não apenas resultados do presente de forma a conseguir compreender o nível de satisfação dos usuários. Uma característica importante seria também considerar o futuro durante a manutenção do sistema, buscando antever necessidades e mudanças nos negócios, o que fatalmente ocasionaria mudanças nestes *softwares*. Do ponto de vista deste caráter proativo, Prog9 comentou em suas interações via aplicativo GRUMPS:

Do jeito que estou criando esse módulo (programa) de cadastro de clientes para produtos médico-hospitalar, usando uma nova tecnologia de integração com outros sistemas da companhia, eu vejo uma grande oportunidade de estender seu uso para todos os produtos médicos [...]

A qualidade, arquitetura e estrutura dos programas pode se tornar um problema quando não está muito bem especificado o processo de mudanças ou análise de impacto na alteração a qual o programa está sofrendo. Ao passo que esse desafio poderia tornar os programas muito mais difíceis de ser manipulados,

ele também abre um leque de possibilidades para constante melhorias. A exemplo disso, podemos elencar algumas das interações de Prog16 e Prog34 capturadas via o aplicativo GRUMPS e Questionário de Pós-Implementação.

Para Prog16, ele indicou o seguinte quanto à relevância do processo criativo e originalidade na manutenção de sistemas computacionais:

Claro que não é a mesma coisa que começar um desenvolvimento de um sistema do zero, mas no suporte (manutenção de *softwares*) tem também bastante espaço para inovar [...] principalmente quando os usuários chegam com uns pedidos meio absurdos pra gente [...] daí, nestas situações, é que teremos um desafio maior que começar um sistema do nada: teremos que implementar algo novo, e ainda por cima, não quebrar (ocasionar erros não propositalmente em) nada já existente.

Segundo Prog34, a originalidade no processo de manutenção de *software* está intimamente ligada ao pensar “fora da caixa”, que significa para ele:

[...] se a todo momento considerarmos somente o que já se encontra disponível no *framework* (repositório de padrões, programas ou módulos), sem questionar o que mais pode ser trazido do mercado para dentro das bibliotecas-padrão, nunca evolveremos [...] pra mim, essa característica questionadora de pensar é parte integrante do perfil de um bom programador [...]

Nem mesmo a falta de padrões nas atividades de manutenção é um problema para Prog30, que vê tal situação como uma área de oportunidade para expansão de seus conhecimentos bem como exercício constante de criatividade na alteração dos programas:

Toda vez que eu encontro alguma coisa fora do padrão, eu vejo aquilo como um grande desafio de melhoria [...] claro que depende primeiro de uma comunicação efetiva com meu supervisor primeiro para alinhar expectativas, principalmente com relação a tempo e esforço, mas depois disso, é só dar asas à imaginação para achar as soluções que cabem melhor ao problema do usuário.

Thurston (1994), matemático cuja produção se pode destacar a formulação da conjectura da geometrização e a demonstração do teorema de geometrização para as variedades de Haken, discute o avanço do campo da Matemática em um

artigo chamado *On the Proof and Progress in Mathematics*, publicado no *Bulletin of the American Mathematical Society*. Partindo da questão “como os matemáticos demonstram teoremas?”, Thurston (1994, p.20, tradução nossa) relata que, devido à sua experiência e em respostas às pressões sociais, passou a escrever sobre o desenvolvimento da infra-estrutura dos resultados que havia publicado e pode perceber que:

[...] o que os matemáticos mais queriam e necessitavam de mim era aprender minhas maneiras de raciocinar, e não aprender minha prova da conjectura da geometrização para as variedades de Haken. (tradução nossa)

Assim, os procedimentos utilizados na resolução de problemas e os raciocínios que conduzem à descoberta são tão importantes quanto à resposta obtida. Como afirma Polya (1945, p. 87):

O resultado do trabalho criador do matemático é o raciocínio demonstrativo, uma prova, mas a prova se descobre por raciocínio plausível, isto é, por intuição. Se isso é assim, e eu o creio, haverá um lugar para a intuição no ensino da Matemática. A educação deve preparar-nos para a invenção, ou, ao menos, para o gosto por ela [...] Meu conselho aos professores de Matemática de todos os graus pode resumir-se nesta exclamação: Ensinemos a intuir! (tradução nossa)

Entretanto, Polya (1945) considera que “ensinar a intuir” (tradução nossa) não é tarefa fácil, visto que não há um método geral de intuição e, conseqüentemente, um método para ensinar os alunos a intuir. Ainda assim, o autor sugere, para professores que tenham experiência com atividades de resolução de problemas, exercícios que podem ser utilizados em sala de aula e levem os alunos a fazer analogias, inferências, induções, deduções e conjecturas. Assim, considera-se a aprendizagem da Matemática semelhante, de alguma forma, a invenção dessa ciência, a intuição torna-se indispensável à sua aprendizagem.

Ainda que a intuição não seja um dos verbetes que constam no dicionário de Heurística de Polya, ela está presente no processo heurístico, é tratada por Descartes e aparece em alguns trechos das obras de Polya. Polya (1945) apropria-se da definição de intuição ou método indutivo de Descartes, que a

estabelece como uma ação da mente pela qual se obtém um conhecimento imediato.

Acreditamos que existe uma aproximação entre as estratégias de resolução de problemas da Matemática, segundo Polya (1995), e as estratégias utilizadas pelos programadores em relação ao critério de originalidade ao propor soluções inovadoras ao problema dado. Esta aproximação ocorre parcialmente no que Polya (1995) destaca como a intuição e o que foi possível identificarmos como inovação no processo de programação.

Afirmamos ser uma aproximação parcial porque a intuição, segundo Polya (1945) propicia uma oportunidade de aprender por meio da invenção, do exercício da criatividade, da habilidade do raciocínio e potencialmente toca o âmbito da formulação de algo novo, da consciência de algo originalmente não pensado antes.

Nessa perspectiva, a resolução de problemas é vista como uma metodologia de ensino que possibilita ao aluno a manipulação dos objetos matemáticos que podem levá-lo a desenvolver sua capacidade mental, exercitar sua criatividade, refletir sobre seu processo de pensamento, adquirir confiança em si mesmo e então, poder melhor estar preparado para o mercado de trabalho.



## 7 CONSIDERAÇÕES FINAIS

Nosso objetivo neste trabalho é estudar as potenciais aproximações entre as estratégias de resolução de problemas da Matemática e o processo de manutenção de programas. Este estudo foi analisado sob alguns aspectos da Heurística segundo a perspectiva de Polya (1945, 1981 e 1995), principalmente com relação aos conceitos de analogia, retrospecto, habilidade para decompor e recombinar, intuição, entre outros elementos propostos por este pesquisador.

Neste enfoque, analisamos os dados primeiramente coletados de três programadores e em uma segunda etapa, de outros trinta e quatro durante cinco sessões práticas diretamente no ambiente de trabalho dos participantes, sempre buscando estabelecer quais as possibilidades de conexão entre o pensamento na resolução de problemas da Matemática e a resolução de problemas na vida profissional cotidiana dos programadores selecionados.

Nossa intenção, com este trabalho, é fornecer a análise de algumas unidades, com o intuito de identificar possíveis tendências e abrir caminhos para uma posterior compreensão da generalidade das mesmas ou, pelo menos, estabelecer bases para uma próxima investigação, mais aprofundada.

Sob esta ótica, acreditamos que os procedimentos metodológicos utilizados a partir da *Grounded Theory* (GT) associados ao uso do aplicativo GRUMPS foram determinantes para entender o comportamento dos participantes. Apoiados na GT segundo Glaser e Strauss (1967), Strauss e Corbin (1998) bem como Charmaz (2000), após o desenrolar das fases de codificação aberta, codificação axial e codificação seletiva, conseguimos conectar um modelo conceitual (por meio das categorias elencadas) às explicações do fenômeno investigado, possibilitando a esta investigadora, desenvolver e relacionar conceitos. Como resultado final, não construímos um modelo teórico completo, mas estabelecemos algumas reflexões teóricas a partir de tal fenômeno estudado.

Recuperando a questão principal deste trabalho sobre as potenciais aproximações entre as estratégias de resolução de problemas da Matemática e o processo de manutenção de programas de computador, por meio da aplicação da metodologia *Grounded Theory* (GT), quatro principais categorias conceituais

emergiram da codificação dos dados. A partir de tais categorias foi possível identificarmos algumas conexões ao aproximarmos a resolução de problemas em Matemática às atividades de manutenção de sistemas computacionais.

Para ilustrarmos isso, e começando a partir da categoria conceitual de “Busca Sistemática de Padrões”, conseguimos perceber que, baseado nas observações destacadas nas falas e ações de alguns programadores, parece ser experiência fundamental e necessária para eles saberem como abstrair a essência de um problema para a identificação de suas variações análogas, em busca de padrões por meio de argumentos lógicos.

De certa forma, apoiando-se em Polya (1981), nota-se que na Matemática, a busca de estabelecimento de relações e argumentos lógicos, expostos de forma explícita e de um modo bastante preciso, poderia fazer emergir nos alunos de Ciência da Computação e afins questões relacionadas ao como eles seriam capazes de retomar experiências passadas e efetivamente utilizá-las em outras situações. Assim, apoiados na prática de rever outros casos do passado via constantes comparações, eles poderiam minimizar os efeitos negativos que a falta de controle e entendimento do problema pode trazer ao sistema em manutenção.

A respeito da categoria “Independência”, conseguimos identificar que autonomia e senso crítico correspondem a uma crescente necessidade para este tipo de profissional. Segundo nossa análise, isso significa, por exemplo, que a partir da capacidade de decompor os elementos da situação atual a qual o programa se encontra, eles deveriam estar preparados para recombinar os diversos aspectos do problema dado para compor uma nova solução.

Alinhados à Polya, concordamos que o pensamento matemático não está relacionado apenas com axiomas, definições e demonstrações rigorosas, mas também com analogias, induções, conjecturas, relações, generalizações e outros processos mentais. Nessa perspectiva, o processo de reconstituição e recombinação das partes frente a uma nova realidade segundo relações já conhecidas constitui-se “num movimento matemático contínuo de encadeamento lógico de idéias precisas” (POLYA, 1981, tradução nossa).

Já na categoria “Julgamento”, foi possível observar que, alinhado à Polya (1995), as estratégias matemáticas aplicadas à programação podem constituir fonte de constante regulação no sentido de continuamente provocar os programadores a modificar, melhorar e agir eficientemente até que o problema seja resolvido com sucesso. Aproximamos o julgamento dos programadores no momento de propor uma solução ao problema dado ao que Polya (1995) explica como a fase de retrospecto da resolução completa, reconsiderando e re-examinando o resultado matemático final e o caminho que levou até este.

Uma ampliação ao trabalho de Polya (1995) que esta pesquisa busca contribuir frente aos dados emergidos nesta categoria “Julgamento”, está relacionado a um aspecto específico que este pesquisador explica sobre o retrospecto: ele deve acontecer após “chegarmos a uma solução do problema e escrita a demonstração”. Entretanto, baseado nas informações coletadas durante a Etapa 2, foi possível verificarmos que nem sempre a “solução completa e demonstrada” precisa estar totalmente pronta para desencadear o processo de retrospecto. Este pode acontecer ao longo do percurso de análise e planejamento da solução a ser dada ao problema em questão.

Outra contribuição de nosso trabalho está relacionada às múltiplas perspectivas em busca de analogias, não somente do ponto de vista do enunciado dos problemas. Embora Polya (1995) relate sobre o retrospecto de uma solução para verificação de sua adequação ao problema dado e reforço para posteriores analogias, isso ainda não completa um ciclo de julgamento proativo (antes de se aplicar uma solução) que busca restringir aspectos superficiais que a aparência sugere.

Portanto, na busca por problemas similares anteriormente resolvidos, os critérios de julgamento dos programadores deve ir além da estrutura do problema, avaliando outros elementos e percepções de potenciais impactos quando determinadas opções são selecionadas. Acreditamos que a presença cada vez mais marcante do exercício do pensamento matemático na resolução de problemas, pode influenciar as decisões dos profissionais com este perfil na busca da mais adequada solução a um problema prático de seu cotidiano.

A respeito da última categoria conceitual “Originalidade”, relacionada a aspectos de inovação ou identificação de oportunidades proativamente pelos programadores durante a busca da solução para o cenário em questão, estamos alinhados à Polya (1945), ao destacar a importância do raciocínio plausível ao se intuir, ação que identificamos parcialmente aproximada à inovação no processo de manutenção dos sistemas computacionais.

Afirmamos ser uma aproximação parcial porque a intuição, segundo Polya (1945) propicia uma oportunidade de aprender por meio da invenção, do exercício da criatividade, da habilidade do raciocínio e potencialmente toca o âmbito da formulação de algo novo, da consciência de algo originalmente não pensado antes.

Entretanto, concordando com Polya (1945), também consideramos que “ensinar a intuir” não é tarefa fácil, visto que não há um método geral de intuição e, conseqüentemente, um método para ensinar os alunos a intuir. Acreditamos novamente que as estratégias de resolução de problemas da Matemática podem auxiliar no exercício dessa prática de dedução e raciocínio lógico, que se mostraram importantes habilidades para os programadores selecionados para esta pesquisa.

Analisando algumas pesquisas ao longo da última década (Wazlawick, 2002; Sanches, 2002; Araújo et al, 2004; Santos e Costa, 2005; França et al, 2010), percebe-se que o foco dos cursos de Ciência da Computação e afins ainda concentra-se demasiadamente em ensinar tecnologias e linguagens de programação. Como resultado deste trabalho, estamos propondo que deveria existir concomitantemente um esforço das instituições escolares em desenvolver e estimular formas algorítmicas de pensar, principalmente àquelas associadas à resolução de problemas matemáticos, as quais pressupõem constante exercício de investigação, questionamento, raciocínio e lógica.

Sabemos que muitas teorias matemáticas levam à prática, porém a motivação primordial do matemático está longe de provocar ou construir estudos que sejam imediatamente colocados em exercício, demonstrando uma intenção explícita de modificar a realidade social, política e econômica. A exemplo disso, a

Teoria dos Jogos Não-Cooperativos de John Forbes Nash Jr não obteve aplicação prática instantânea, porém em 1994 rendeu-lhe o prêmio Nobel de Economia, juntamente com o húngaro John Harsanyi e o alemão Reinhard Selten. Esta lacuna temporal entre a teoria e a descoberta de uma prática que a tornasse socialmente importante foi de quase meio século.

Portanto, torna-se simplório o pensamento de que uma prática investigativa do aluno na resolução de problemas dentro da Matemática, o levaria a descobertas da função social ou profissional da própria Matemática. O que acreditamos ser essencial neste processo de reforçar a preparação dos profissionais da Computação é conscientizá-los que as estratégias da Matemática podem e devem ser extrapoladas para outros campos de conhecimento, por meio de diferentes métodos, e entre eles, a resolução de problemas.

Concluindo, reconhecemos a importância da atitude de investigação na resolução de problemas, de forma geral. Contudo, devemos enfatizar que, baseado nesta pesquisa, acreditamos que a Matemática a ser focada para os profissionais da Computação para melhor direcioná-los a sua realidade profissional e social deve aprofundar (1) a investigação matemática como apuração, crítica, lógica e focada na elaboração de alternativas e resolução de problemas em oposição à (2) investigação matemática como elaboração de conjecturas com posterior verificação e demonstração de sua validade, dirigindo o foco somente para a Matemática, como uma averiguação sistemática de seus padrões, implicações e inter-relações. Não que a alternativa (2) seja inválida ou menos importante, mas baseado neste trabalho, parecem haver indícios na direção da opção (1).

As possibilidades de abordagem do pensamento matemático ligado à manutenção de sistemas computacionais não foram esgotadas neste trabalho. Assim, outras pesquisas podem ser desenvolvidas como forma de dar continuidade à exploração deste tema, tais como, de que forma estabelecer as diretrizes para um curso da Computação focando-se nos aspectos das estratégias de resolução de problemas que apresentamos aqui; em que medida os programadores de computador enxergam o real valor da Matemática em sua prática; de que maneira a manutenção de sistemas pode ser encarada como uma

oportunidade prática de conexão com outros elementos da Matemática, etc. Nossa questão é se teríamos as mesmas tendências evidenciadas nesta pesquisa.

## 8 REFERÊNCIAS BIBLIOGRÁFICAS

ARAÚJO, F. V.; FALKEMBACH, G. A. M. **Ambiente de Aprendizagem Adaptado para Algoritmos (A4) Utilizando Módulos Interativos de Auxílio ao Aprendizado.** Anais SIIE. Cáceres – Espanha, 2004.

BALIEIRO FILHO, I. F. **Arquimedes, Pappus, Descartes e Polya – Quatro episódios da história da heurística.** Tese de Doutorado em Educação Matemática – Instituto de Geociências e Ciências Exatas. Rio Claro: Universidade Estadual Paulista, 2004.

BANDEIRA-DE-MELLO, R.; CUNHA, C. Grounded Theory In: Godoi, C. K., Bandeira-de-Mello, R., Silva, A. B. d. (eds), **Pesquisa Qualitativa em Estudos Organizacionais: Paradigmas, Estratégias e Métodos**, São Paulo, Saraiva, 2006.

BENBASAT, I.; GOLDSTEIN D. K.; MEAD, M. **The Case Research Strategy in Studies of Information Systems.** *MIS Quarterly*, v.11, 1987.

BHATT, P.; WILLIAMS, K.; SHROFF, G.; MISRA, K. **Influencing Factors in Outsourced Software Maintenance.** ACM SIGSOFT: Software Engineering Notes, 2006.

BOLAND, R. **Control, causality and information system requirements.** Londres: Accounting, Organizations and Society, 1979.

BOYER, C. B. **História da Matemática.** Tradução: Elza F. Gomide – 2ª ed. – São Paulo: Edgard Blücher, 1996.

BRASIL. Ministério da Educação – MEC. Secretaria de Educação Média e Tecnológica. **Parâmetros Curriculares para o Ensino Médio. Parte III – Ciências da Natureza, Matemática e suas Tecnologias.** Brasília: MEC/Semtec, 1999.

BURKHARDT, J.-M.; DED TIENNE, F.; WIEDENBECK, S. **The effect of object oriented programming expertise in several dimensions of comprehension**

**strategies.** IWPC'98: Proceedings of the Sixth International Workshop on Program Comprehension. New York: IEEE, 1998.

CHARLES, R.; LESTER, F. K. **Teaching Problem Solving: What, Why and How**, Palo Alto, Dale Seymor Publications, 1982.

CHARMAZ, K. **Grounded Theory. Objectivist and Constructivist Methods.** Londres: Handbook of Qualitative Research - Sage Publications, 2ª ed., 2000.

CHIZZOTTI, A. **Pesquisa em ciências humanas e sociais.** São Paulo: Cortez, 6ª ed., 2003.

CORRITORE, C. L.; WIEDENBECK, S. **Mental representations of expert procedural and object-oriented programmers in a software maintenance task.** International Journal of Human-Computer Studies, v50, 1999.

COTTINGHAM, J. **Dicionário de Descartes.** Tradução, Helena Martins. – Rio de Janeiro: Jorge Zahar Ed., 1995.

D'AMBROSIO, B. S. **Como ensinar Matemática hoje?** Temas & Debates. Nº. 2, 1989.

DART, S.; CHRISTIE, A. M.; BROWN, A. W. **A Case Study in Software Maintenance.** Carnegie Mellon University, 2001.

DE VILLIERS, M. R. **Three approaches as pillars for interpretative information systems research: development research, action research and grounded theory.** New York-EUA: ACM Press, 2005.

DENZIL, N. K.; LINCOLN, Y. S. **Competing paradigms in qualitative research.** Londres: Handbook of Qualitative Research - Sage Publications, 2ª ed., 2000.

DESCARTES, R. **Discurso do método: regras para a direção do espírito.** Tradução: Pietro Nasseti. São Paulo: Martin Claret, 2005.

DÉTIENNE, F. **Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams.** Interacting with Computers, n.9, 1997.

- DÉTIENNE, F. **Software Design - Cognitive Aspects**. London: Springer, 2002.
- EVANS, H.; ATKINSON, M.; BROWN, M.; CARGILL, J.; CREASE, M.; DRAPER, S.; GRAY, P.; THOMAS, R. **Pervasiveness of Evolution in GRUMPS Software**. *Software: Practice & Experience*, 33 (2), 2003.
- FARRER, H. **Algoritmos Estruturados**. Rio de Janeiro: LTC, 3ª Edição, 1999.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: a construção de algoritmos e estrutura de dados**. São Paulo: Makron Books, 2ª ed., 2000.
- FRANÇA, V. F.; PIMENTEL, E. P.; NORONHA, R. V.; OMAR, N. **Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores**. Disponível na Internet em < <http://ceie-sbc.educacao.ws/pub/index.php/wie/article/view/819/805>>. Acessado em 4 de Abril de 2010 às 07:58:00.
- FRIEDMANN, C. V. P.; JURKIEWIZ, S.; LOZANO, A. **Algumas questões sobre algoritmos, modelagem discreta e ensino**. Feira de Santana: Anais da IV Conferência Nacional sobre Modelagem e Educação Matemática, 2005.
- GLASER, B. G.; STRAUSS, A. L. **The discovery of grounded theory: strategies for qualitative research**. Londres: Weidenfeld and Nicolson, 1967.
- GODOY, A. S. **Introdução à pesquisa qualitativa e suas possibilidades**. São Paulo: *Revista de Administração de Empresas*, v. 35, n. 2, 1995.
- GRUMPS, 2001. **The GRUMPS Research Project**. Disponível na Internet em <<http://grumps.dcs.gla.ac.uk>>. Acessado em 18 de Outubro de 2010 às 12:17:00.
- HARTMANIS, J. **On Computational Complexity and the Nature of Computer Science**. New York-EUA: ACM Press, 1995.
- HOARE, C. A. R.; JONES, C. B. **Essays in Computing Science**. Londres: Practice Hall International, 1989.

HOLGEID, K. K.; KROGSTIE, J.; SJØBERG, D. I. K. **A Study of Development and Maintenance in Norway: Assessing the Efficiency of Information Systems Support Using Functional Maintenance.** Information and Software Technology, n.10, 2000.

HOUAISS, A. **Dicionário da Língua Portuguesa.** Rio de Janeiro Instituto. Ed. Objetiva, 2001.

IEEE. **Std 1219 – IEEE Standard for Software Maintenance.** New York-EUA: Institute of Electrical and Electronic Engineers, 1998.

IMENES, L. M. P.; LELLIS, M. **O Ensino de Matemática e a Formação do Cidadão.** Temas e Debates, Blumenau, n.5, 1994.

JOHARI, J. **Interpretivism in Information Systems (IS) Research.** Putra: University Putra Malaysia Press, 2009.

KANTOWSKI, M. G. **Processes involved in mathematical problem solving,** Journal for Research in Mathematics Education, vol. 8, nº. 3, 1997.

KAPLAN B.; DUCHON D. **Combining Qualitative and Quantitative Methods in Information Systems Research: A Case Study.** Londres: MIS Quarterly, v. 12, 1988.

KOENEMANN, J.; ROBERTSON, S. **Expert problem solving strategies for program comprehension.** CHI'91 Proceedings. New York: ACM, 1991.

KOLIKANT, Y. B.; POLLACK, S. **Improving mathematically oriented programming skills in Computer Science studies.** Boston: 32ª Conferência ASEE/IEEE de Fronteiras em Educação, 2002.

LAYZELL, P.; CHAMPION, R.; FREEMAN, M. **DOCKET: program comprehension-in-the-large.** Proceedings of the IEEE 2nd Workshop on Program Comprehension. Los Alamitos, CA: IEEE Computer Society, 1993.

LEEDY, P. D.; ORMROD, J. E. **Practical Research, Planning and Design.** Nova Jersey: Prentice Hall, 7ª ed., 2001.

LESTER, F. **O que aconteceu à investigação em resolução de problemas de Matemática? A situação nos Estados Unidos.** In FERNANDES, D.; Borralho, A.; Amaro, G. Resolução de problemas: Processos cognitivos, concepções de professores e desenvolvimento curricular. Lisboa: IIE, 1993.

LIENTZ, B. P.; SWANSON, E. B. **Software Maintenance Management.** Reading MA: Addison-Wesley, 1980.

LITTMAN, D. C.; PINTO, J.; LETOVSKY, S.; SOLOWAY, E. **Mental models and software maintenance.** In SOLOWAY, E.; IYENGAR, S. Eds. Empirical Studies of Programmers. Norwood, NJ: Ablex, 1986.

MEDEIROS JÚNIOR, R. J. **Resolução de problemas e ação didática em Matemática no Ensino Fundamental.** Dissertação de Mestrado em Educação Matemática – Setor de Educação, Curitiba: Universidade Federal do Paraná, 2007.

MOSEMANN, R.; WIEDENBECK, S. **Navigation and Comprehension of Programs by Novice Programmers.** IEEE 9th Int. Workshop on Program Comprehension (IWPC 2001), 2001.

NAUR, P.; RANDALL, B. **Software Engineering Concepts and Techniques.** Garmisch: NATO Conferência em Engenharia de Software, 1976.

NCTM. **Professional Standards for Teaching Mathematics.** National Council of Teachers of Mathematics, Virginia, 1991. Normas Profissionais para o Ensino de Matemática. Tradução da Associação de professores de Matemática, Portugal, 1994.

NOSS, R.; HOYLES, C. **Windows on Mathematical Meanings: Learning cultures and computers.** Dordrecht: Academia Kluwer, 1996.

NOSS, R.; HOYLES, C.; POZZI S. **Abstraction in Expertise: A study of nurses conceptions of concentration.** Journal for Research in Mathematics Education, n.33, 2002.

NOSS, R.; HOYLES, C.; POZZI, S. **Working Knowledge: Mathematics in Use Mathematical Sciences Group**. Institute of Education, University of London, 2006.

ONUCHIC, L. R. Novas reflexões sobre o ensino-aprendizagem de Matemática através da Resolução de Problemas. In: BICUDO, A. V. & BORBA, M. C. (Orgs.). **Educação Matemática: pesquisa em movimento**. São Paulo: Ed. Cortez, 2004.

ORLIKOWSKI, W. J.; BAROUDI, J. J. **Studying Information Technology in Organizations: Research Approaches and Assumptions**. Massachusetts: MIT Press, 1991.

PADUELLI, M. M.; SANCHES, R. **Problemas em manutenção de software: caracterização e evolução**. Vila Velha-ES: III Workshop de Manutenção de Software Moderna, 2006.

PARKIN, P. **An exploratory study of code and document interaction during task-directed program comprehension**. Australian Software Engineering Conference, 2004.

PENNINGTON, N. **Comprehension strategies in programming**. In OLSON, G.; SHEPPARD, S; SOLOWAY, E. Eds. Empirical Studies of Programmers: 2nd Workshop. Norwood, NJ: Ablex, 1987.

PERRENOUD, P. **Avaliação: da excelência à regulação das aprendizagens: entre duas lógicas**. Porto Alegre: Artmed, 1999.

PFLEEGER, S. L. **Software Engineering: theory and practice**. New Jersey: Prentice Hall, 2<sup>a</sup> ed., 2001.

PIGOSKI, T. M. **Practical Software Maintenance: Best Practices for Managing Your Software Investment**. Willey Computer Publishing, 1996.

POLO, M.; PIATTINI, M.; RUIZ, F.; CALERO, C. **Roles in maintenance process**. ACM SigSoft Software Engineering Notes, v.24, 1999.

POLYA, G. **How to solve it - a new aspect of mathematical method**. New Jersey: Princeton University Press, 2<sup>a</sup> ed., 1945.

\_\_\_\_\_. **Mathematical Discovery: On Understanding, learning, and teaching**

**Problem Solving.** New York: John Wiley and Sons, 1981.

\_\_\_\_\_. **A Arte de resolver problemas.** Rio de Janeiro: Ed. Interciência, 1995.

PRESSMAN, R. S. **Software Engineering: a practitioner's approach.** McGrawHill Higher Education, 6<sup>a</sup> ed., 2005.

PUCHKIN, V. N. **Heurística: a ciência do pensamento criador.** Rio de Janeiro, Zahar Ed., 1969.

ROBILLARD, P. N.; KERZAZI, N.; TAPP, M.; HMIMA, H. **Outsourcing Software Maintenance: Processes, Standards & Critical Practices.** Electrical and Computer Engineering, 2007.

ROSS, P.; HOWE, J. **Teaching mathematics through programming: Ten years on.** Amsterdam: North-Holland Press, 1981.

SANCHES, J. A. **How to teach algorithmic reasoning.** Anais do IV Curso de Qualidade – Metodologia de ensino para cursos de graduação das áreas de Computação e Informática - Congresso da Sociedade Brasileira de Computação, Florianópolis, 2002.

SANTOS, R. P.; COSTA, H. A. X. **TBC-AED e TBC-AED/WEB: Um Desafio no Ensino de Algoritmos, Estruturas de Dados e Programação.** Em: Anais do IV WEIMIG, Varginha-MG, 2005.

SAVIANI, D. **Educação do senso comum à consciência filosófica – 14<sup>a</sup> ed.-** Campinas, SP: Ed. Autores Associados, 2002. – (Coleção educação contemporânea), 2002.

SCHOENFELD, H. A. **Learning to think mathematically: problem solving, metacognition, and sense making in mathematics,** Handbook for Research on Mathematics Teaching and Learning (D. Grouws, Ed.), New York: MacMillan, 1992.

SCHOENFELD, H. A. **Por quê toda essa agitação acerca da Resolução de Problemas?** In: ABRANTES, P. et al (Org.). *Investigar para Aprender Matemática*. Lisboa: APM, 1996.

SEAMAN, C. B. **Qualitative Methods. Guide to Advanced Empirical Software Engineering**. Londres: Springer-Verlag, 2008.

SNEED, H. M. **Critical Success Factors in Software Maintenance**. Amsterdam: International Conference on Software Maintenance, 2003.

SOMMERVILLE, I. **Engenharia de Software**. São Paulo: Addison-Wesley, 6ª ed., 2003.

SOUZA, S. C.; NEVES, W. C. G.; ANQUETIL, N.; OLIVERIA, K. M. **Documentação Essencial para Manutenção de Software II**. Brasília-DF: I Workshop de Manutenção de Software Moderna, 2004.

STOREY, M. A. **Theories, Models and Tools in Program Comprehension: Past, Present and Future**. 13th International Workshop on Program Comprehension, 2005.

STRATHERN, P. **Descartes em 90 minutos**. Rio de Janeiro: Jorge Zahar Ed., 1997. Tradução: Maria Helena Geordane.

STRAUSS, A.; CORBIN, J. **Basics of Qualitative Research. Techniques and Procedures for Developing Grounded Theory**. Londres: Sage Publications, 1998.

SWEBOK. **Guide to the Software Engineering Body of Knowledge**. The Institute of Electrical and Electronics Engineers, Inc – IEE, 2004.

TALL, D. **Cognitive Development in Advanced Mathematics Using Technology**. *Mathematics Education Research Journal*, v. 12, 2001.

THURSTON, W. P. **Three-dimensional manifolds, Kleinian groups and hyperbolic geometry**. *American Mathematical Society Bulletin* v. 6, 1994.

TRIVIÑOS, A. N. S. **Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação.** São Paulo: Atlas, 1987.

UPCHURCH, R., 2002. **Code reading and program comprehension: annotated bibliography.** Disponível na Internet em <http://www2.umassd.edu/SWPI/ProcessBibliography/bib-coderading2.html>.

Acessado em 10 de Março de 2011 às 03:15:00.

VIANNA, C. R. **Resolução de Problemas. In: Futuro Congressos e Eventos. (Org.). Temas em Educação I - Livro das Jornadas 2002.** Curitiba: Futuro Congressos e Eventos, 2002.

VILA, A.; CALLEJO, M. L. **Matemática para aprender a pensar: o papel das crenças na resolução de problemas.** Tradução: Erani Porto. Alegre: Editora Artmed, 2006.

VILANOVA, S. **Concepciones y creencias sobre la matemática. Una experiencia con docentes de 3er. Ciclo de la Educación General Básica.** Revista Iberoamericana de Educación, Argentina, 2000.

VON MAYRHAUSER, A.; VANS, A. M. **Identification of dynamic comprehension processes during large scale maintenance.** IEEE Transactions on Software Engineering, v22, 1996.

VON MAYRHAUSER, A.; VANS, A. M. **Program understanding behavior during debugging of large scale software.** In S. WIEDENBECK, J.; SCHOLTZ, C. Eds. Empirical Studies of Programmers: Seventh Workshop, Norwood, NJ: Ablex, 1997.

WALSHAM, G. **The Emergence of Interpretivism in IS Research.** Lancaster: European Journals of Information Systems, 1995.

WALSHAM, G. **Doing Interpretive Research.** Lancaster: European Journals of Information Systems, 2006.

WAZLAWICK, R. S. **Diretrizes metodológicas para ensino de programação.** Anais do IV Curso de Qualidade – Metodologia de ensino para cursos de

graduação das áreas de Computação e Informática - Congresso da Sociedade Brasileira de Computação, Florianópolis, 2002.

WIEDENBECK, S.; RAMALINGAM, V. **Novice Comprehension of small programs written in the procedural and object-oriented styles.** Int. J. Human-Computer Studies, n.51, 1999.

WIELEWSKI, G. D. **Aspectos do pensamento matemático na resolução de problemas: uma apresentação contextualizada da obra de Krutetskii.** Tese de Doutorado em Educação Matemática. São Paulo: PUC-SP, 2005.

## **9 ANEXOS**

### **9.1 Anexo 1 – Roteiro para Abordagem Inicial (Etapa 1)**

1. Identificar idade, formação acadêmica na graduação e pós-graduação bem como respectivos anos de conclusão.
2. Levantar histórico de desempenho (visão pessoal sobre o próprio desempenho) nas disciplinas técnicas durante cursos de graduação ou pós-graduação. Capturar percepções (exemplos de situações, comentários) que evidenciem o comportamento e atuação descritos.
3. Levantar histórico de desempenho (visão pessoal sobre o próprio desempenho) nas disciplinas ligadas à Matemática durante cursos de graduação ou pós-graduação. Capturar percepções (exemplos de situações, comentários) que evidenciem o comportamento e atuação descritos.
4. Questionar anos de experiência profissional na área de programação de computadores. Discutir sobre carreira (ingresso na área de desenvolvimento de sistemas de informação) até o presente momento.
5. Solicitar elementos que evidenciem reconhecimento profissional no atual cargo/empresa/área que trabalha.

## 9.2 Anexo 2 – Especificação Técnica-Funcional do Componente de Software (Etapa 1)

### Introduction

#### Objective

The objective of this component is to convert from Gregorian/Julian calendar to different date formats. This specification is to set out the technical details of the DateConversion component. It explains what are the minimum functionalities required as well as how it will be built and deployed. It excludes non-technical matters such as project management, business cases, human resources, or training.

#### System Overview

Various calendar systems have been in use at different times and places around the world. The aim of this component is to deal with, at least, two: the Gregorian calendar, now used universally for civil purposes, and the Julian calendar, its predecessor in the western world.

The Julian calendar has a leap year every fourth year, while the Gregorian calendar has a leap year every fourth year except century years not exactly divisible by 400. This component should assume that the changeover from the Julian calendar to the Gregorian calendar occurred in October of 1582, according to the scheme instituted by Pope Gregory XIII. Specifically, for dates on or before 4 October 1582, the Julian calendar is used; for dates on or after 15 October 1582, the Gregorian calendar is used.

Just to provide a better context to the required component, the sixteenth century was a time of severe religious division right across Europe. States still obedient to the Papacy adopted the Gregorian calendar at once, in October 1582.

In Britain, the Gregorian calendar was not adopted until 1752, and the day following 31st December 1751 was decreed to be 1st January 1752, and 2nd September 1752 was followed by 14th September. As England had taken the year 1700 to be a leap year, the difference between the Julian and Gregorian calendars now amounted to eleven days. The changes were to apply to all the dominions of the British Crown, including the North American colonies.

---

### Programming Specifications

#### Programming Languages

Programming languages that are allowed to be used for the project:

- JavaScript
- Java
- VB.NET

#### Programming Tools

List the programming tools, if appropriate, that are allowed to be used for the project:

- Microsoft Visual Studio
  - Any testing tools
  - Limited Internet access to consult Julian-Gregorian algorithm are allowed
-

## System Requirements

### Client Side Requirements

In terms of how the screen will be composed, the following items need to be available:

- Text field to type Gregorian or Julian date
- Calculate button
- Read-only label to present the Gregoria/Julian corresponding date

Text field need to be initialized with the current date (today).

For validation purposes, text field should contain a valid date

No database access or different data sources will be used.

### System Architecture

Platform: Web based

Database Access: None

Performance Requirements: The response should not delay more than 4 seconds to come up.

### Basic Functionalities

- 1) **Julian Data Conversion:** This feature corresponds to the core objective of this component. More conversions can be added, but at least, the Gregorian to Julian must be part of this. The calculation should follow the concepts as follow:

Julian Date Conversion =

$$367 * \text{Year} - \text{Integer Part}((7 * (\text{Year} + \text{Integer Part}((\text{Month} + 9) / 12))) / 4) + \text{Integer Part}((275 * \text{Month}) / 9) + \text{Day} + 1721013.5 + \text{Universal Time in Hours} / 24 - 0.5 * \text{Sign}(100 * \text{Year} + \text{Month} - 190002.5) + 0.5$$

- 2) **Gregorian Data Conversion:** This feature corresponds to the core objective of this component. More conversions can be added, but at least, the Julian to Gregorian must be part of this. All letters below are being used for reference only. The calculation should follow the concepts as follow:

Gregorian Date Conversion =

$$X = \text{Integer Part}(\text{Julian Date}) + 0.5$$
$$Z = \text{Integer Part}(X)$$
$$F = X - Z$$
$$Y = \text{Integer Part}((Z - 1867216.25) / 36524.25)$$
$$A = Z + 1 + Y - \text{Integer Part}(Y / 4)$$
$$B = A + 1524$$
$$C = \text{Integer Part}((B - 122.1) / 365.25)$$
$$D = \text{Integer Part}(365.25 * C)$$
$$G = \text{Integer Part}((B - D) / 30.6001)$$
$$\text{month} = (G < 13.5) ? (G - 1) : (G - 13)$$
$$\text{year} = (\text{month} < 2.5) ? (C - 4715) : (C - 4716)$$
$$\text{UT} = B - D - \text{Integer Part}(30.6001 * G) + F$$
$$\text{day} = \text{Integer Part}(\text{UT})$$
$$\text{UT} -= \text{Integer Part}(\text{UT})$$
$$\text{UT} *= 24;$$

hour = Integer Part(UT)  
UT -= Integer Part(UT)  
UT \*= 60  
minute = Integer Part(UT)  
UT -= Integer Part(UT)  
UT \*= 60  
second = Integer Part(UT)

- 3) **Validation:** This feature corresponds to the form validation in order to check if the inputted texts are really valid dates and times. A message should be presented to user to allow them to make the appropriated changes.

### 9.3 Anexo 3 – Questionário Pós-Implementação (Etapa 1)

#### Parte 1: Especificação Técnica-Funcional

1. Com relação à qualidade dos requisitos fornecidos, eles foram suficientemente claros para a criação do componente de *software*? O que você acrescentaria, caso necessário?
2. Na sua opinião, qual foi a influência da especificação fornecida na resultado final do seu trabalho? Você se ateu somente ao documento no momento de avaliar o que efetivamente seria implementado?
3. Você atribui algum grau de flexibilidade menor ou maior no seu componente devido às sentenças descritas na especificação técnica-funcional?

---

#### Parte 2: Reusabilidade de Componentes de *Software*

1. De que critérios você se utiliza para classificar um componente como altamente reutilizável? E de escopo restrito?
  2. Quão reutilizável você classifica o componente desenvolvido por você para esta pesquisa?
  3. Quando implementando o componente proposto para este estudo, o quanto e de que formas você visualizou a solução dada em relação ao seu reuso por outros sistemas futuramente?
  4. Em sua opinião, caso exista, há relação entre o grau de reutilização de um componente, a experiência profissional do programador e sua preparação acadêmica nas disciplinas da Matemática?
  5. Assumindo um componente de *software* com alto grau de reutilização, que tipo de estratégias, habilidades ou raciocínio lógico-matemático você acredita serem fundamentais na construção destes programas?
-

### **Parte 3: Programação e sua Relação com a Matemática**

6. Que elementos da Matemática você acredita ter empregado durante o desenvolvimento do componente proposto para esta pesquisa?
7. Você enxerga alguma conexão sobre suas decisões e julgamentos realizados em sua prática profissional em relação à aplicação de conceitos da Matemática?
8. Caso exista, de que forma você enxerga que elementos da Matemática influenciam o processo de desenvolvimento de um componente de *software*?
9. O que para você pode ser considerado essencial na preparação de profissionais programadores da Ciência da Computação e afins para o mercado de trabalho? Que tópicos da Matemática você considera importante para este perfil de especialistas?

#### **9.4 Anexo 4 – Roteiro para Abordagem Inicial (Etapa 2)**

1. Identificar idade, formação acadêmica na graduação e pós-graduação bem como respectivos anos de conclusão.
2. Levantar histórico de desempenho (visão pessoal sobre o próprio desempenho) nas disciplinas técnicas durante cursos de graduação ou pós-graduação. Capturar percepções (exemplos de situações, comentários) que evidenciem o comportamento e atuação descritos.
3. Levantar histórico de desempenho (visão pessoal sobre o próprio desempenho) nas disciplinas ligadas à Matemática durante cursos de graduação ou pós-graduação. Capturar percepções (exemplos de situações, comentários) que evidenciem o comportamento e atuação descritos.
4. Perguntar sobre sua experiência com resolução de problemas nas disciplinas ligadas à Matemática durante cursos de graduação ou pós-graduação. Capturar percepções (exemplos de situações, comentários) que evidenciem o comportamento e atuação descritos.
5. Questionar anos de experiência profissional na área de programação de computadores. Discutir sobre carreira (ingresso na área de desenvolvimento de sistemas de informação) até o presente momento.

## **9.5 Anexo 5 – Questionário Pós-Implementação (Etapa 2)**

### **Parte 1: Compreensão do Problema Dado (Especificação Técnica-Funcional)**

1. Com relação aos requisitos fornecidos, em sua opinião, eles foram suficientemente claros para que você pudesse seguir em frente com as modificações nos programas em questão? O que você acrescentaria, caso necessário?
  2. Em sua opinião, qual foi a influência da especificação fornecida no resultado final do seu trabalho? Você se ateu somente ao documento no momento de avaliar o que efetivamente seria implementado?
  3. Você julga ter compreendido bem o requisito dados? Como você poderia evidenciar isso?
  4. De quais formas você se lembra de ter analisado e organizado os dados, inicialmente disponibilizados na especificação técnica-funcional, para a criação de um plano de ação na tentativa de resolução do problema dado?
  5. Você atribui algum grau de flexibilidade menor ou maior das alterações realizadas nos programas devido às sentenças descritas na especificação técnica-funcional?
- 

### **Parte 2: Estabelecimento do Plano de Resolução**

6. Você pensou em alternativas para a solução do problema dados?
7. Quais foram os critérios utilizados para selecionar um determinado caminho e eliminar outras alternativas de implementação?
8. Baseado em sua experiência profissional, você buscou levantar problemas semelhantes em seu repertório? Quais? De que forma eles influenciaram na solução final dada?
9. Em relação ao caminho traçado originalmente, você acredita que o plano de resolução do problema dado foi satisfatoriamente realizado? Porque?

---

### Parte 3: Validação dos Resultados

10. De que maneira você acredita ter examinado a solução obtida? Qual foi seu critério para estabelecer que a solução dada era suficiente para endereçar todos os requisitos iniciais?
11. Você se considera um bom resolvidor de problemas? Que características, na sua opinião, evidenciam ou não este perfil?
12. Você enxerga alguma conexão sobre suas decisões e julgamentos realizados em sua prática profissional em relação à aplicação de estratégias de resolução de problemas na Matemática?
13. Caso exista, de que forma você enxerga que elementos da Matemática influenciam o processo de desenvolvimento de um componente de *software*?
14. O que para você pode ser considerado essencial na preparação de profissionais programadores da Ciência da Computação e afins para o mercado de trabalho? Que tópicos da Matemática você considera importante para este perfil de especialistas?

## 9.6 Anexo 6 – Tabulação dos Dados do Roteiro Inicial (Etapa 2)

#	Nome	Idade	Ensino Fundamental Finalizado em	Ensino Médio Finalizado em	Curso Graduação	Graduação Finalizada em	Universidade Graduação	Curso Pós-Graduação	Pós-Graduação Finalizada em	Universidade Pós-Graduação	Anos de Experiência Profissional	Cargo Atual	Desempenho nas disciplinas relacionadas à Matemática	Desempenho na resolução de problemas de Matemática em sala de aula	Desempenho nas disciplinas técnicas da Computação	Comentários
1	Prog1	34	1992	1995	Bacharelado em Sistemas de Informação	1998	Mackenzie-SP	Especialização em Sistemas Orientados à Objeto	2002	Mackenzie-SP	13	Líder Técnico	Satisfatório	Satisfatório	Satisfatório	
2	Prog2	37	1989	1992	Engenharia da Computação	1997	ETEP-SJC	N/A	N/A	N/A	15	Analista Sênior	Muito Satisfatório	Muito Satisfatório	Satisfatório	
3	Prog3	42	1984	1987	Engenharia da Computação	1992	UNIFEL-Itajubá	N/A	N/A	N/A	20	Arquiteto	Satisfatório	Muito Satisfatório	Muito Satisfatório	
4	Prog4	28	1998	2002	Bacharelado em Ciência da Computação	2007	UNIVAP-SJC	N/A	N/A	N/A	5	Analista Júnior	Insatisfatório	Insatisfatório	Satisfatório	Reprovado uma vez em alguma das disciplinas ligadas à Matemática
5	Prog5	35	1991	1994	Bacharelado em Ciência da Computação	1998	UNIVAP-SJC	N/A	N/A	N/A	14	Analista Sênior	Satisfatório	Muito Satisfatório	Muito Satisfatório	
6	Prog6	37	1989	1992	Tecnologia em Processamento de Dados	1996	FATEC-SP	N/A	N/A	N/A	15	Especialista em Banco de Dados	Muito Satisfatório	Muito Satisfatório	Muito Satisfatório	
7	Prog7	33	1993	1996	Bacharelado em Sistemas de Informação	1999	Mackenzie-SP	N/A	N/A	N/A	11	Analista Sênior	Satisfatório	Satisfatório	Satisfatório	

#	Nome	Idade	Ensino Fundamental Finalizado em	Ensino Médio Finalizado em	Curso Graduação	Graduação Finalizada em	Universidade Graduação	Curso Pós-Graduação	Pós-Graduação Finalizada em	Universidade Pós-Graduação	Anos de Experiência Profissional	Cargo Atual	Desempenho nas disciplinas relacionadas à Matemática	Desempenho na resolução de problemas da Matemática em sala de aula	Desempenho nas disciplinas técnicas da Computação	Comentários
8	Prog8	36	1990	1993	Tecnologia em Processamento de Dados	1997	FATEC-SP	N/A	N/A	N/A	16	Analista Sênior	Satisfatório	Muito Satisfatório	Muito Satisfatório	
9	Prog9	28	1999	2002	Bacharelado em Ciência da Computação	2008	UNIFEI-Itajubá	N/A	N/A	N/A	9	Analista Sênior	Satisfatório	Satisfatório	Satisfatório	
10	Prog10	48	1978	1982	Administração de Empresas	1986	USP-SP	Mestrado Acadêmico em Engenharia de Software	2001	IME USP-SP	17	Arquiteto	Muito Satisfatório	Muito Satisfatório	Muito Satisfatório	
11	Prog11	32	1994	1997	Bacharelado em Ciência da Computação	2003	UNIVAP-SJC	N/A	N/A	N/A	9	Analista Sênior	Insatisfatório	Insatisfatório	Satisfatório	Reprovado uma vez em alguma das disciplinas ligadas à Matemática
12	Prog12	28	1998	2001	Bacharelado em Ciência da Computação	2005	UNIVAP-SJC	N/A	N/A	N/A	6	Analista Pleno	Satisfatório	Satisfatório	Satisfatório	
13	Prog13	27	1999	2002	Tecnologia em Análise e Desenvolvimento de Sistemas	2005	FATEC-SJC	N/A	N/A	N/A	8	Especialista em Banco de Dados	Satisfatório	Satisfatório	Muito Satisfatório	

#	Nome	Idade	Ensino Fundamental Finalizado em	Ensino Médio Finalizado em	Curso Graduação	Graduação Finalizada em	Universidade Graduação	Curso Pós-Graduação	Pós-Graduação Finalizada em	Universidade Pós-Graduação	Anos de Experiência Profissional	Cargo Atual	Desempenho nas disciplinas relacionadas à Matemática	Desempenho na resolução de problemas de Matemática em sala de aula	Desempenho nas disciplinas técnicas da Computação	Comentários
14	Prog14	30	1996	1999	Bacharelado em Ciência da Computação	2004	UNIFESP-SJC	N/A	N/A	N/A	7	Analista Pleno	Satisfatório	Satisfatório	Satisfatório	
15	Prog15	30	1996	1999	Bacharelado em Ciência da Computação	2005	UNIFEI-Itajubá	N/A	N/A	N/A	6	Analista Pleno	Satisfatório	Insatisfatório	Satisfatório	
16	Prog16	36	1990	1993	Bacharelado em Sistemas de Informação	1995	Mackenzie-SP	N/A	N/A	N/A	15	Arquiteto	Satisfatório	Muito Satisfatório	Muito Satisfatório	
17	Prog17	31	1995	1998	Bacharelado em Ciência da Computação	2004	UNIFEI-Itajubá	MBA em Gerenciamento de Projetos	2004	FGV-SJC	7	Analista Pleno	Satisfatório	Satisfatório	Satisfatório	
18	Prog18	34	1992	1996	Bacharelado em Ciência da Computação	2000	UNIVAP-SJC	N/A	N/A	N/A	12	Analista Sênior	Satisfatório	Satisfatório	Muito Satisfatório	
19	Prog19	27	1999	2002	Engenharia da Computação	2008	UNIFEI-Itajubá	N/A	N/A	N/A	4	Analista Júnior	Insatisfatório	Insatisfatório	Satisfatório	
20	Prog20	28	1998	2001	Engenharia da Computação	2007	ETEP-SJC	N/A	N/A	N/A	5	Analista Júnior	Satisfatório	Satisfatório	Satisfatório	
21	Prog21	29	1997	2000	Tecnologia em Análise e Desenvolvimento de Sistemas	2004	FATEC-SJC	N/A	N/A	N/A	8	Analista Pleno	Insatisfatório	Insatisfatório	Satisfatório	Reprovado uma vez em alguma das disciplinas ligadas à Matemática

#	Nome	Idade	Ensino Fundamental Finalizado em	Ensino Médio Finalizado em	Curso Graduação	Graduação Finalizada em	Universidade Graduação	Curso Pós-Graduação	Pós-Graduação Finalizada em	Universidade Pós-Graduação	Anos de Experiência Profissional	Cargo Atual	Desempenho nas disciplinas relacionadas à Matemática	Desempenho na resolução de problemas de Matemática em sala de aula	Desempenho nas disciplinas técnicas da Computação	Comentários
22	Prog22	45	1981	1984	Bacharelado em Ciência da Computação	1988	UNIFEI-Itajubá	MBA em Gerenciamento de Projetos	2007	FGV-SJC	20	Líder Técnico	Muito Satisfatório	Muito Satisfatório	Muito Satisfatório	
23	Prog23	27	1999	2002	Engenharia da Computação	2008	ETEP-SJC	N/A	N/A	N/A	4	Analista Júnior	Satisfatório	Insatisfatório	Satisfatório	
24	Prog24	42	1984	1987	Bacharelado em Ciência da Computação	1991	UNIFEI-Itajubá	MBA em Gerenciamento de Projetos	2010	FGV-SJC	19	Arquiteto	Satisfatório	Muito Satisfatório	Muito Satisfatório	
25	Prog25	36	1990	1994	Bacharelado em Ciência da Computação	1999	UNIFEI-Itajubá	MBA em Gerenciamento de Projetos	2005	FGV-SJC	12	Arquiteto	Satisfatório	Satisfatório	Satisfatório	
26	Prog26	29	1998	2001	Bacharelado em Ciência da Computação	2006	UNIFESP-SJC	N/A	N/A	N/A	5	Analista Júnior	Satisfatório	Satisfatório	Satisfatório	
27	Prog27	34	1992	1995	Tecnologia em Processamento de Dados	1998	Mackenzie-SP	Especialização em Sistemas Orientados à Objeto	2001	Mackenzie-SP	15	Especialista em Banco de Dados	Satisfatório	Satisfatório	Muito Satisfatório	
28	Prog28	27	1992	1996	Bacharelado em Matemática Computacional	2001	UNIFESP-SJC	N/A	N/A	N/A	7	Analista Pleno	Satisfatório	Satisfatório	Satisfatório	

#	Nome	Idade	Ensino Fundamental Finalizado em	Ensino Médio Finalizado em	Curso Graduação	Graduação Finalizada em	Universidade Graduação	Curso Pós-Graduação	Pós-Graduação Finalizada em	Universidade Pós-Graduação	Anos de Experiência Profissional	Cargo Atual	Desempenho nas disciplinas relacionadas à Matemática	Desempenho na resolução de problemas da Matemática em sala de aula	Desempenho nas disciplinas técnicas da Computação	Comentários
29	Prog29	35	1992	1995	Bacharelado em Sistemas de Informação	1998	Mackenzie-SP	N/A	N/A	N/A	12	Analista Sênior	Satisfatório	Satisfatório	Satisfatório	
30	Prog30	32	1995	1998	Bacharelado em Matemática Computacional	2002	UNIFESP-SJC	N/A	N/A	N/A	9	Analista Sênior	Muito Satisfatório	Muito Satisfatório	Muito Satisfatório	
31	Prog31	28	1998	2001	Engenharia da Computação	2007	ETEP-SJC	N/A	N/A	N/A	4	Analista Júnior	Satisfatório	Muito Satisfatório	Muito Satisfatório	
32	Prog32	36	1990	1994	Bacharelado em Ciência da Computação	1998	UNIFESP-SJC	N/A	N/A	N/A	13	Analista Sênior	Satisfatório	Satisfatório	Muito Satisfatório	
33	Prog33	30	1996	1999	Tecnologia em Banco de Dados ou Redes de Computadores	2002	FATEC-SJC	N/A	N/A	N/A	10	Especialista em Banco de Dados	Satisfatório	Satisfatório	Muito Satisfatório	
34	Prog34	31	1995	1998	Engenharia da Computação	2003	UNIFEI-Itajubá	N/A	N/A	N/A	7	Analista Sênior	Satisfatório	Muito Satisfatório	Muito Satisfatório	