

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Darlan Pedro de Campos

**SEI: SISTEMA ESPECIALISTA PARA O AUTOATENDIMENTO NO SUPORTE DE  
TECNOLOGIA DA INFORMAÇÃO**

Florianópolis  
2013/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA  
CURSO DE SISTEMAS DE INFORMAÇÃO

Darlan Pedro de Campos

**SEI: SISTEMA ESPECIALISTA PARA O AUTOATENDIMENTO NO SUPORTE DE  
TECNOLOGIA DA INFORMAÇÃO**

Trabalho de conclusão de curso  
apresentado como parte dos requisitos  
para obtenção do grau de Bacharel em  
Sistemas de Informação.

Orientadora:

Prof.<sup>a</sup> Dra. Luciana de Oliveira Rech

2013/1

Darlan Pedro de Campos

**SEI: SISTEMA ESPECIALISTA PARA O AUTOATENDIMENTO NO SUPORTE DE  
TECNOLOGIA DA INFORMAÇÃO**

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

**Orientadora:**

---

Prof.<sup>a</sup> Dra. Luciana de Oliveira Rech

**Banca examinadora:**

---

Prof. Dr. José Eduardo De Lucca

---

Prof. Dr. Lau Cheuk Lung

*"Cada pessoa é um gênio. Mas se você julgar um peixe por sua capacidade de subir em uma árvore, ele passará toda a sua vida acreditando que é estúpido. "*

*(Albert Einstein)*

## RESUMO

Com o aumento do uso da informática para a realização das mais diversas atividades, cresce também a demanda por uma assistência técnica especializada de alta disponibilidade. Frequentemente, essa demanda não é suprida completamente pela contratação e treinamento de novos técnicos de suporte. Surge, então, a necessidade de complementar o arcabouço de tecnologias de assistência com uma ferramenta capaz de fornecer soluções para incidentes básicos de informática diretamente ao usuário final, por meio de sessões interativas, nas quais os dados informados deflagram regras pré-definidas capazes de inferir conclusões a respeito da causa do incidente e, por conseguinte, da solução a ser aplicada. Este trabalho tem como tema o desenvolvimento de um sistema especialista (SE) que atue no atendimento especificamente da área de suporte interno de um órgão público do Estado de Santa Catarina. O sistema desenvolvido pretende, com a implantação do autoatendimento inteligente, possibilitar a diminuição do tempo decorrido entre a constatação e a efetiva resolução de um problema, a redução dos custos com contratação de pessoal e com transporte, e a difusão do conhecimento.

**Palavras-chave:** sistema especialista, raciocínio baseado em regras, *help desk*, autoatendimento, *software* livre.

## SUMÁRIO

<b>1 INTRODUÇÃO.....</b>	<b>7</b>
1.1 Motivação.....	7
1.2 Escopo do trabalho.....	8
1.3 Objetivos.....	9
1.3.1 Objetivo Geral.....	9
1.3.2 Objetivos Específicos.....	9
1.4 Organização do texto.....	10
<b>2 FUNDAMENTAÇÃO TEÓRICA.....</b>	<b>11</b>
2.1 Sistemas especialistas.....	11
2.1.1 Conceitos.....	11
2.1.2 Arquitetura dos SEs.....	12
2.1.3 Fases do desenvolvimento de um SE.....	14
2.1.3.1 Fase da identificação.....	14
2.1.3.2 Fase da conceituação.....	15
2.1.3.3 Fase da formalização .....	15
2.1.3.4 Fase da implementação.....	15
2.1.3.5 Fase de teste e avaliação.....	16
2.1.3.6 Fase da revisão.....	16
2.2 Estratégias de busca em espaço de estados.....	16
2.2.1 Busca guiada por dados.....	16
2.2.1.1 O algoritmo RETE.....	17
2.2.2 Busca guiada por objetivo.....	20
2.3 Ferramentas para criação de SEs em Java.....	21
2.3.1 Jess.....	21
2.3.2 Drools.....	24

2.4 Conclusões.....	26
<b>3 ESTADO DA ARTE.....</b>	<b>27</b>
3.1 SEs recentes.....	27
3.2 Trabalhos de IA na área de suporte.....	28
3.3 Comparação deste trabalho com os trabalhos estudados.....	30
3.4 Conclusões.....	31
<b>4 DESENVOLVIMENTO.....</b>	<b>32</b>
4.1 Fase da identificação.....	32
4.1.1 Identificação das características do problema.....	33
4.2 Fase da conceituação.....	34
4.2.1 Classificação dos atendimentos.....	34
4.2.2 Níveis de dificuldade.....	38
4.2.3 Formação do esboço das regras.....	40
4.3 Fase de formalização.....	40
4.4 Fase da implementação.....	42
4.4.1 Especificação de requisitos.....	43
4.4.1.1 Requisitos funcionais:.....	43
4.4.1.2 Requisitos não funcionais:.....	44
4.4.2 Modelagem.....	45
4.4.2.1 Diagrama de classes.....	46
4.4.2.2 Casos de uso.....	47
4.4.3 Desenvolvimento do shell.....	48
4.4.3.1 Máquina de inferência.....	49
4.4.3.2 Memória de trabalho.....	53
4.4.3.3 Base de conhecimento.....	54
4.4.4 Desenvolvimento da interface.....	55
4.4.4.1 Interface de consulta.....	57

4.4.4.2 Interface de edição da base de conhecimento.....	61
4.4.4.3 Implantação do banco de dados.....	65
4.5 Fase do teste e avaliação.....	65
4.6 Fase da revisão.....	69
<b>5 CONCLUSÕES E TRABALHOS FUTUROS.....</b>	<b>70</b>
5.1 Conclusões.....	70
5.2 Principais contribuições.....	71
5.3 Trabalhos futuros.....	71
<b>REFERÊNCIAS.....</b>	<b>73</b>



## LISTA DE FIGURAS

Figura 2.1: Estrutura de um SE baseado em regras.....	13
Figura 2.2: Exemplo de regra de produção.....	18
Figura 2.3: Rede construída para duas condições (C1 e C2).....	19
Figura 2.4: Declaração de um template e inserção de um fato em Jess.....	23
Figura 2.5: Exemplo de regra em Jess.....	23
Figura 2.6: Definição de uma regra (a) e declaração de um fato (b) na linguagem de regras Drools.....	25
Figura 4.1: Classificação dos atendimentos realizados em 2012, desde o primeiro (a) até o quarto (d) trimestre.....	37
Figura 4.2: Nivelamento por dificuldade dos atendimentos prestados no ano de 2012, desde o primeiro (a) até o quarto (d) trimestre.....	40
Figura 4.3: Demonstração de um par problema/solução identificado no processo de esboço das regras.....	41
Figura 4.4: Estrutura de regra de produção adotada.....	43
Figura 4.5: Especificação de uma regra de produção.....	43
Figura 4.6: Diagrama de classes simplificado do shell SEI.....	48
Figura 4.7: Diagrama simplificado de casos de uso do projeto SEI.....	49
Figura 4.8: Diagrama de atividades para o método execute.....	51
Figura 4.9: Diagrama de atividades para o método processVariable.....	52
Figura 4.10: Diagrama de atividades para o método processRule.....	53
Figura 4.11: Exemplo de regra definida em XML.....	55
Figura 4.12: Diagrama de classes das ações da interface.....	57
Figura 4.13: Tela de boas-vindas do sistema SEI.....	58
Figura 4.14: Tela inicial de consulta por uma solução.....	59
Figura 4.15: Tela de consulta com ajuda ativada.....	60
Figura 4.16: Tela de resultado, com uma solução encontrada e passo a passo para execução.....	61
Figura 4.17: Tela que é exibida quando não há nenhum resultado.....	62
Figura 4.18: Tela inicial de edição.....	63
Figura 4.19: Tela de edição de variáveis.....	64
Figura 4.20: Diálogo para edição e criação de variáveis.....	65

Figura 4.21: Tela de edição de regras.....	66
Figura 4.22: Diálogo de edição e criação de regras.....	67
Figura 4.23: Tela de edição e criação dos textos de ajuda.....	68

## LISTA DE QUADROS

Quadro 4.5.1: Resultados da primeira etapa de testes.....	70
Quadro 4.5.2: Resultados da segunda etapa de teste e avaliação.....	71

## LISTA DE REDUÇÕES

BRMS	Business Rule Management System
EMT	Elemento Da Memória De Trabalho
GETIG	Gerência de Tecnologia de Informação e Governança Eletrônica
GNU	GNU is Not Unix
GPL	General Public License
HTML	HyperText Markup Language
IA	Inteligência Artificial
IDE	Integrated Development Environment
IPREV	Instituto de Previdência do Estado de Santa Catarina
JESS	Java Expert System Shell
LHS	Left Hand Side
LIFO	Last In First Out
MP	Memória de Produção
MT	Memória de Trabalho
RHS	Right Hand Side
SADPC	Sistema de Auxílio ao Diagnóstico de Problemas em Computadores
SBC	Sistema Baseado em Conhecimento
SE	Sistema Especialista
SEI	Suporte Emergencial Inteligente
TI	Tecnologia da Informação
UML	Unified Modeling Language
URL	Uniform Resource Location
XML	eXtensible Markup Language

## 1 INTRODUÇÃO

Neste capítulo serão apresentados os aspectos introdutórios do trabalho, desde a motivação para o seu desenvolvimento até os objetivos gerais e específicos.

### 1.1 Motivação

A situação da gerência de tecnologia da informação (TI) do Instituto de Previdência do Estado de Santa Catarina (IPREV), uma autarquia do governo do Estado de Santa Catarina<sup>1</sup>, é o cenário que motivou o presente trabalho. Essa gerência é responsável pelo atendimento dos chamados abertos pelos usuários dos equipamentos de informática e programas de computador de todo o órgão, que conta com um edifício-sede, onde a maior parte dos recursos estão alocados, e uma rede de agências geograficamente separadas.

Atualmente há uma carência nos setores de TI em geral, e nos órgãos públicos da esfera estadual em particular, por pessoal qualificado na área (VALLE, 2011). As instituições de ensino e capacitação simplesmente não qualificam um número suficiente de profissionais para preencher todas as vagas disponíveis no mercado de trabalho, o que gera um déficit crescente. Ao mesmo tempo, o processo de contratar e capacitar um novo funcionário por conta do próprio empregador pode ser demorado e dispendioso.

Os setores de TI da administração pública estadual sofrem com alguns entraves adicionais para a contratação de novos funcionários. O primeiro deles é que todo ingresso deve acontecer necessariamente por meio de concurso público<sup>2</sup>, um processo que em geral ocorre num intervalo de dois anos ou mais e abrange vagas para todos os setores, não somente as de TI. Por isso, uma vaga que surge pode demorar anos a ser preenchida, aguardando a realização de um concurso.

---

<sup>1</sup> Disponível em: <<http://www.iprev.sc.gov.br/>>. Acesso em: 26 nov. 2012.

<sup>2</sup> BRASIL. Constituição da República Federativa do Brasil. Disponível em: <[http://www.planalto.gov.br/ccivil\\_03/constituicao/constituicao.htm](http://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm)>. Acesso em: 29 nov. 2012.

Outro entrave é a política de contenção de despesas que por vezes é estabelecida por algumas administrações, tendo por objetivo evitar que os gastos públicos ultrapassem a receita alcançada (SEF, 2012). Tal medida constantemente inibe a contratação de funcionários e a aquisição de novos equipamentos, concomitantemente aumentando o número de problemas pela obsolescência do maquinário e diminuindo o pessoal disponível para diagnosticar e resolver esses mesmos problemas.

Existem ainda sinais de que muitos incidentes que ocasionam a abertura de chamado pelos usuários poderiam ser resolvidos pelos próprios usuários, desde que dispusessem de uma ferramenta que os auxiliasse na tarefa de identificar o problema e escolher a solução. Como o suporte técnico de TI do IPREV é centralizado em Florianópolis e as agências do órgão se localizam na maioria a mais de 100 quilômetros do edifício-sede, a disponibilidade de uma ferramenta com essas características evitaria o atraso na resolução dos incidentes e os gastos com transporte, decorrentes do deslocamento de um técnico de suporte até a agência.

Por fim, a formação de uma base de conhecimento propiciaria a preservação e a difusão do conhecimento técnico disponível. A concentração de conhecimento, que dificulta a disponibilidade e a eficiência do suporte, poderia ser evitada, e novos técnicos poderiam utilizar a ferramenta para aprender a resolver novos incidentes, diminuindo o tempo de treinamento e facilitando o aprendizado.

## **1.2 Escopo do trabalho**

A fim de garantir a celeridade no processo de atendimento dos incidentes na área de TI, assim como tornar o conhecimento obtido acessível a quem dele necessitar, em tempo hábil e onde for necessário, pretende-se implantar um SE que seja capaz, por um processo interativo compreensível ao usuário, de auxiliar na identificação de uma possível solução para um problema da área de TI.

O SE será implantado inicialmente no IPREV, pela Gerência de Tecnologia de Informação e Governança Eletrônica (GETIG) do órgão. Posteriormente, tem-se o objetivo de tornar possível adaptá-lo à realidade de outros ambientes e

organizações. Para isso, seria necessária a criação de uma base de conhecimento diferente, específica para o ambiente de trabalho de destino.

Não serão resolvidos pelo SE incidentes que exijam, para a sua resolução, a presença de um especialista ou técnico de suporte. Nesses casos, o SE poderá encaminhar o usuário para um nível superior dentro da hierarquia de atendimento.

Para possibilitar a adoção do sistema pelo maior número possível de organizações sem a intervenção de entraves legais, apenas componentes e bibliotecas disponíveis sob licenças livres serão utilizados no desenvolvimento do sistema. Como licenças livres serão considerados aquelas compatíveis com a licença GNU GPL, versão 2 ou posterior<sup>3</sup>.

De modo semelhante, as ferramentas utilizadas no processo de modelagem, codificação e testes do código-fonte deverão preferencialmente ser de acesso livre e gratuito.

## **1.3 Objetivos**

### **1.3.1 Objetivo Geral**

Este trabalho é relativo ao desenvolvimento um SE capaz de, por meio de interação com o usuário, diagnosticar a causa de um determinado incidente técnico de informática relatado e apresentar uma solução adequada, a partir de uma base de conhecimento formada como resultado do histórico de atendimentos realizados.

### **1.3.2 Objetivos Específicos**

Para que sejam atingidos os resultados esperados, foi necessário alcançar os seguintes objetivos específicos:

- Especificar os requisitos funcionais e não-funcionais para o desenvolvimento do SE;

---

<sup>3</sup> Disponível em: <<http://www.gnu.org/licenses/gpl-2.0.html>>. Acesso em: 30 nov. 2012.

- Analisar o histórico de ordens de serviço, a partir das quais o sistema fornecerá as soluções aos problemas apresentados pelos usuários;
- Modelar e criar a base de conhecimento;
- Modelar os componentes do sistema (*software*), tanto as classes de controle e de acesso a dados como a interface com o usuário;
- Desenvolvimento das classes de controle e de acesso a dados;
- Desenvolvimento da interface com o usuário;
- Implantação do sistema e efetuação de testes no ambiente de trabalho.

#### **1.4 Organização do texto**

O presente trabalho está organizado em quatro capítulos, dos quais este primeiro capítulo apresenta os aspectos introdutórios, incluindo a motivação e os objetivos a serem alcançados no decorrer do projeto.

O capítulo 2 contém os temas fundamentais levantados durante a fase de revisão bibliográfica, necessários para nortear todo o processo de desenvolvimento e implantação.

No capítulo 3 serão apresentados os mais recentes SEs desenvolvidos em diversas áreas, assim como os trabalhos correlatos na área de sistemas inteligentes aplicados ao suporte técnico de TI.

O capítulo 4 traz a descrição das etapas do desenvolvimento do projeto propriamente dito, desde a fase da identificação até a fase da revisão.

Finalmente, o último capítulo apresenta as conclusões do trabalho.



## **2 FUNDAMENTAÇÃO TEÓRICA**

Neste capítulo serão delineados os principais fundamentos teóricos sobre os quais o projeto será elaborado.

### **2.1 Sistemas especialistas**

Com o objetivo de fundamentar a implantação de um SE, faz-se necessário o estudo das características dos SEs, das fases necessárias para o desenvolvimento de um SE, dos tipos de SE existentes e também a análise dos mais recentes SEs desenvolvidos.

#### **2.1.1 Conceitos**

De acordo com Giarratano e Riley (1993), os SEs são um ramo da Inteligência Artificial que faz uso intenso de conhecimento especializado para resolver problemas que de outra forma necessitariam de um especialista humano (*expert*), o qual detém conhecimento numa determinada área. Um SE, portanto, emula a habilidade de um perito no processo de tomada de decisão.

SEs são bastante eficientes quando aplicados a um domínio de aplicação suficientemente restrito e, para cumprirem sua função, necessitam dispor de um conhecimento vasto e aprofundado sobre sua área de atuação (RABUSKE, 1995). Isso torna a base de conhecimento um componente crucial de um SE, do qual depende todo o sistema. De fato, Feigenbaum afirmou: “A potência de um sistema especialista deriva do conhecimento que ele possui e não dos formalismos e esquemas específicos que ele emprega” (citado por RABUSKE, 1995, p. 74).

A aquisição do conhecimento dá-se principalmente com a ajuda de peritos humanos, mas poderia ser obtido também de outras fontes, como livros e revistas, desde que sejam viabilizados meios automáticos que permitam selecionar, extrair, reunir e organizar o conhecimento, considerados os aspectos de tempo e espaço de armazenamento necessários (RABUSKE, 1995).

O conhecimento adquirido pode ser representado por meio de diferentes técnicas, como regras de produção, redes semânticas, quadros, *scripts*, linguagens de representação de conhecimento diversas, grafos conceituais, entre outras (GIARRATANO; RILEY, 1993). Uma forma bastante comum de representação é a que utiliza regras de produção.

### 2.1.2 Arquitetura dos SEs

Segundo Rabuske (1995, p. 73), “[n]o contexto de IA, o termo arquitetura é referente à ciência de projetar algo, determinando-lhe a estrutura”. Existem variações significantes quanto à definição dos componentes de um SE, notando-se divergência entre os autores quanto à nomenclatura e até mesmo quanto ao número de componentes. No presente trabalho, será utilizado o modelo de Giarratano e Riley (1993) como base (figura 2.1), e as variantes serão apresentadas quando relevante ou necessário.

Os componentes de um SE são os seguintes:

- **Base de conhecimento:** é responsável por armazenar todo o conhecimento disponível, sob a forma de representação escolhida pelo projetista do sistema. Contém um somatório de fatos, heurísticas e crenças (RABUSKE, 1995).
- **Interface com o usuário:** é o meio pelo qual o SE interage com o usuário.
- **Mecanismo de explicação:** justifica ao usuário o raciocínio seguido pelo sistema. Também chamado de “sistema de justificação”, deve poder responder a perguntas tais como “como chegou a esta conclusão?”, “por que chegou a esta conclusão?”, “por que não chegou a outra conclusão?” (RABUSKE, 1995).

- **Memória de trabalho:** um banco de dados global de fatos usados pelas regras. Rabuske (1995) a chama de “quadro-negro” e lhe dá a atribuição de armazenar, além de fatos, informações e estruturas de suporte ao funcionamento do sistema.

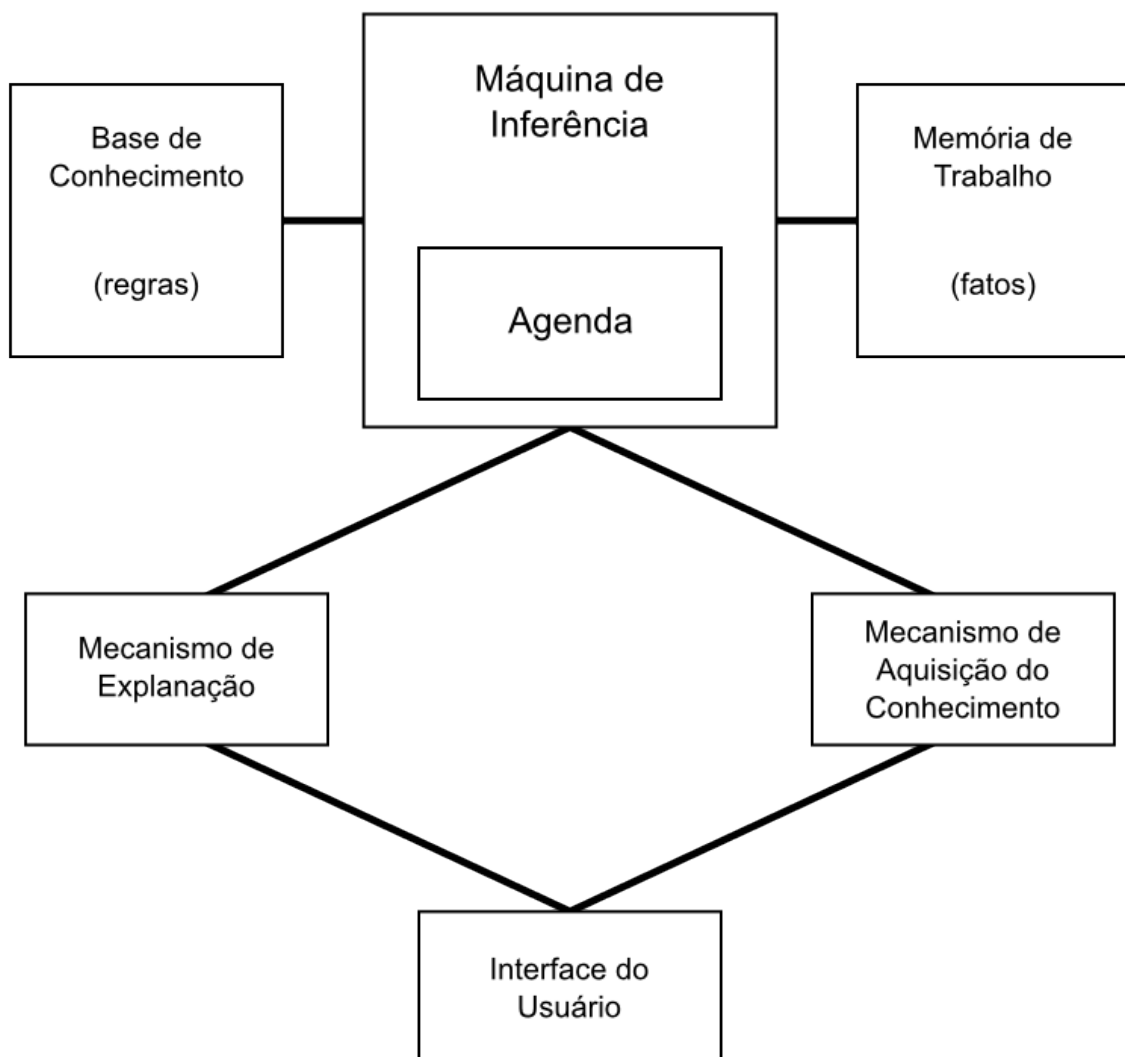


Figura 2.1: Estrutura de um SE baseado em regras.

Fonte: Giarratano e Riley (1993).

- **Máquina de inferência:** quando o sistema envolve regras de produção, é a responsável por decidir quais regras são satisfeitas por fatos ou objetos, dar prioridade às regras satisfeitas e executar a regra com a maior prioridade.
- **Agenda:** consiste na lista de regras cujos antecedentes foram satisfeitos pelos fatos contidos na memória de trabalho. Em Rabuske (1995), faz parte do quadro-negro.
- **Mecanismo de aquisição do conhecimento:** é um meio automático para que o conhecimento seja obtido pelo SE a partir da interação com o usuário, em vez de diretamente inserido pelo engenheiro do conhecimento. Para Rabuske (1995), é a parte mais crítica da construção de um SE.

### 2.1.3 Fases do desenvolvimento de um SE

Diferentemente dos sistemas computacionais tradicionais, que têm fases mais sequenciais e distintas, os SEs apresentam uma distinção de etapas bastante minimizada (RABUSKE, 1995). O processo é mais iterativo e evolucionário do que sequencial, de modo que se pode retornar a uma fase anterior sem ter chegado à fase seguinte (REZENDE, 2003). A divisão em fases apresentada por este trabalho baseia-se em Rabuske (1995), e as variações relevantes são também listadas.

#### 2.1.3.1 Fase da identificação

Esta etapa se subdivide em identificação dos participantes do projeto, dos recursos envolvidos, das características do problema e dos objetivos a alcançar. Em primeiro lugar é identificado o “dono” do sistema, que será o responsável por dar a última palavra quanto aos aspectos do sistema. Depois se procede à escolha do engenheiro do conhecimento e do especialista, que inicialmente pode ser apenas um.

A identificação dos recursos compreende a identificação das fontes de conhecimento, a delimitação do tempo, a definição dos recursos computacionais,

tanto os equipamentos como os sistemas, além da identificação dos recursos financeiros envolvidos.

#### **2.1.3.2 Fase da conceituação**

A etapa de conceituação consiste em definir os recursos básicos a serem utilizados para descrever o problema e estabelecer o grau de refinamento da representação do conhecimento. Rezende (2003) enumera as seguintes tarefas a serem desempenhadas pelo engenheiro do conhecimento nessa fase:

- Fazer entrevistas estruturadas seguindo um conjunto de perguntas formuladas a partir da análise do material coletado na fase anterior;
- Coletar casos para serem usados na modelagem e teste do SBC;
- Observar o especialista no trabalho.

#### **2.1.3.3 Fase da formalização**

Na fase da formalização, os conceitos e as relações chaves são expressas de uma maneira formal, com a identificação de estruturas de suporte para sua representação e armazenamento. O engenheiro do conhecimento deve escolher o formalismo que melhor se adapte à representação do problema/solução, como a lógica de primeira ordem, quadros e regras de produção (REZENDE, 2003).

#### **2.1.3.4 Fase da implementação**

A etapa da implementação consiste na escolha de uma linguagem de representação adequada ao formalismo adotado e da codificação dos programas necessários para o processamento, quando não for feita opção por uma ferramenta já disponível.

### **2.1.3.5 Fase de teste e avaliação**

A fase de teste e avaliação é um processo contínuo, desde a implementação do projeto inicial. Falhas de representação poderão ser descobertas nessa etapa, que também deve avaliar o SE em si e a sua razão de existência. Rezende (2003) afirma que os sistemas baseados em conhecimento possuem como característica peculiar o fato de que respostas incorretas em pequena escala podem ser admissíveis, o que traz como dificuldade adicional estabelecer o limite aceitável de erros.

### **2.1.3.6 Fase da revisão**

A fase da revisão, omitida por Rezende (2003), “consiste em revisar o sistema, especialmente para alterar e melhorar aspectos observados na fase de avaliação” (RABUSKE, 1995, p. 84).

## **2.2 Estratégias de busca em espaço de estados**

O processo de resolução de um problema pode acontecer de dois modos, dependendo de onde se inicia a busca: pelos estados iniciais ou pelos estados finais, aos quais se deseja chegar. A primeira estratégia é conhecida como “busca guiada por dados”, ou “encadeamento progressivo”, ao passo que a segunda é chamada de “busca guiada por objetivos”, ou “encadeamento regressivo” (LUGER, 2004).

### **2.2.1 Busca guiada por dados**

Na busca guiada por dados, como o próprio nome diz, a busca se inicia a partir dos fatos conhecidos em direção às conclusões inferidas a partir desses fatos. Com efeito, nessa forma de encadeamento, para se chegar a uma conclusão, é

como se nada fosse conhecido sobre a base de dados e fosse necessário descobrir ou solicitar explicitamente os dados (KELLER, 1991). A cada iteração, a parte esquerda de cada regra é comparada com o estado atual da memória de trabalho e, se as condições da regra tiverem sido satisfeitas, a sua parte direita, que contém as ações, é executada. A execução, por sua vez, pode levar à inserção de novos fatos na memória, e assim por diante (PY, 2000).

De acordo com Luger (2004), a busca guiada por dados é apropriada a problemas nos quais:

1. Todos os (ou a maioria dos) dados são fornecidos na formulação inicial do problema;
2. Existe um grande número de objetivos em potencial, mas há apenas poucas maneiras diferentes de usar os fatos e a informação fornecida de uma ocorrência particular do problema;
3. É difícil formular um objetivo ou hipótese.

### **2.2.1.1 O algoritmo RETE**

O algoritmo Rete (do lat. *rete*, "rede") foi inicialmente desenvolvido por Charles L. Forgy, como solução para o problema de casamento de muitos padrões para muitos objetos (FORGY, 1979). Posteriormente, diversas otimizações foram identificadas e descritas, a fim de remediar alguns contratempos encontrados no algoritmo original, algumas das quais vieram a se formar algoritmos diferentes, com destaque para o Rete/UL (DOOREMBOS, 1995). Algoritmos alternativos também foram desenvolvidos, como o TREAT (MIRANKER, 1987), e o LEAPS (BATORY, 1994).

Em uma máquina de inferência de implementação óbvia, cada fato inserido na memória de trabalho é comparado com todas as regras existentes na base de conhecimento. A cada ciclo as ações de uma regra modificam apenas alguns fatos na lista de fatos, o que é conhecido como "redundância temporal". Apesar disso, todas as regras têm que ser comparadas com todos os fatos, o que pode gerar uma

grande ineficiência. Para evitar essa consequência, indesejável principalmente em sistemas cujo número de regras é expressivo, o algoritmo Rete memoriza todos os casamentos efetuados a cada ciclo, mesmo aqueles que são parciais (GIARRATANO; RILEY, 1993).

Rete trabalha com uma memória de produção (MP) e uma memória de trabalho (MT), ambas as quais podem ser modificadas ao longo do tempo. A memória de trabalho contém itens que representam os fatos sobre a situação atual do sistema. Cada item na memória de trabalho é um elemento da memória de trabalho (EMT), também conhecido como fato. A memória de produção armazena as regras de produção (ou produções) com as quais se deseja fazer o casamento dos fatos (DOOREMBOS, 1995).

Basicamente, o algoritmo é composto de uma “rede” dividida em duas partes: a parte *alfa* e a parte *beta*. A parte alfa contém um nó inicial, nós que executam testes sobre cada fato, e memórias alfas, as quais cada uma corresponde a uma ou mais condições de uma regra. O nó inicial, por onde todos os fatos são inseridos na rede, possui a particularidade de não efetuar nenhum teste. Quando um fato passa por todos os testes de uma condição, é armazenada na memória alfa correspondente àquela condição. Regras diferentes podem compartilhar uma mesma memória alfa, contanto que tenham uma condição em comum. Por exemplo, dada a seguinte regra de produção:

```
(bloco-sobre-bloco-azul
  (<x> ^cor azul)
  (<y> ^sobre <x>)
  -->
  RHS
)
```

Figura 2.2: Exemplo de regra de produção.

Quando o elemento E1: (B1 ^cor azul) é inserido na memória de trabalho, o atributo constante “cor” será testado, em seguida o valor constante “azul”,



e finalmente o elemento será armazenado na memória alfa da primeira condição (C1).

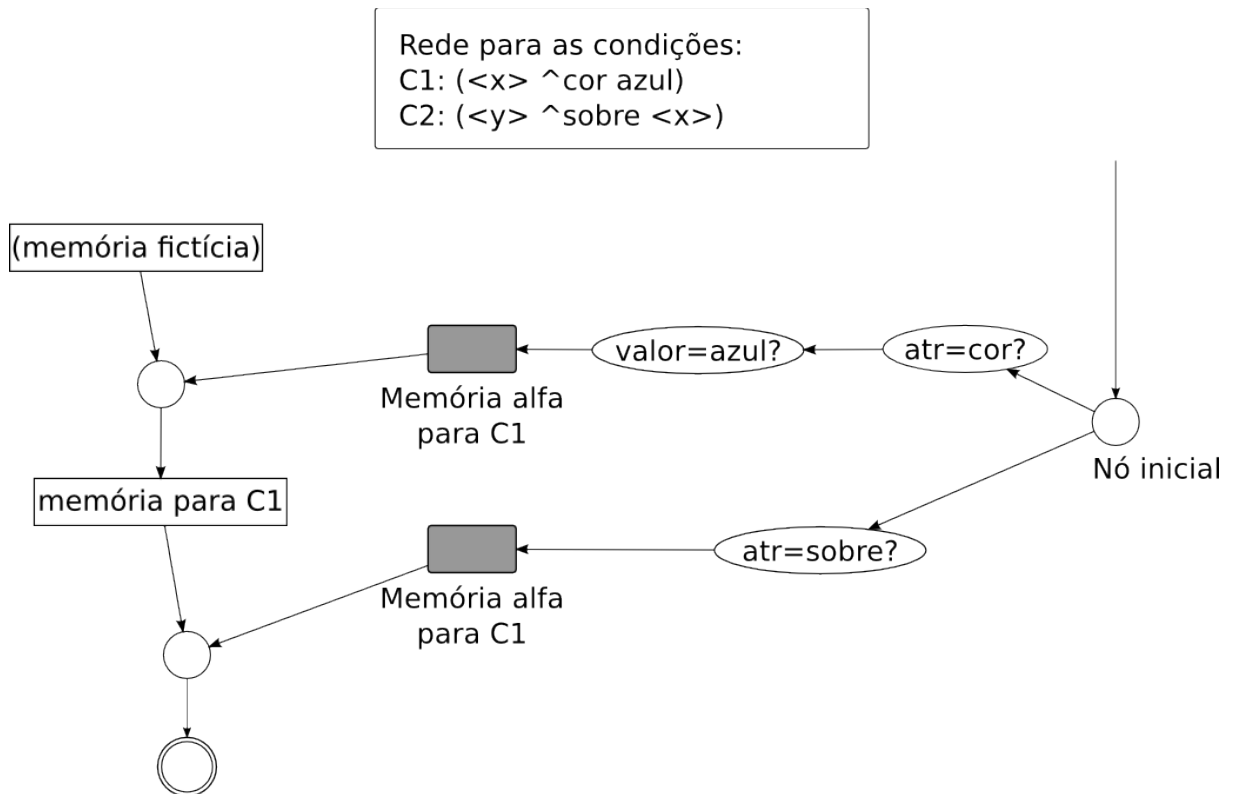


Figura 2.3: Rede construída para duas condições (C1 e C2).

Fonte: adaptado de Doorembo (1995).

A parte beta da rede contém essencialmente nós que cuidam da junção e do armazenamento dos elementos de memória entre condições. Os principais nós que fazem parte da rede beta são o nó de junção (*join node*) e a memória beta. O primeiro é responsável por efetuar testes de consistência entre atribuições de variáveis entre uma condição e as anteriores, enquanto a segunda tem a função de armazenar os padrões parcialmente satisfeitos. Se um nó de junção corresponder à primeira condição de uma regra, uma memória beta fictícia será acoplada como seu nó antecessor à esquerda. Tanto o nó de junção como a memória beta podem ser ativados pelos seus nós antecessores (“pais”), com a diferença de que os nós de junção podem ser ativados tanto à esquerda quanto à direita, ao passo que as

memórias betas podem receber ativação somente à direita. O processo de ativação consiste em receber um elemento de memória, efetuar os testes necessários e passá-lo adiante aos sucessores.

Dado um elemento E2:  $(B2 \text{ sobre } B1)$ , que passa por todos os testes da rede alfa e é armazenado na memória alfa correspondente à C2, o nó de junção imediatamente abaixo dessa memória alfa será ativado à direita, recebendo E2. Ao receber o elemento, efetuará testes de junção entre o elemento recebido e os casamentos parciais armazenados na memória beta acima, a fim de verificar a consistência entre as atribuições das variáveis. No exemplo, o nó de junção testaria se o valor de E2 (B1) corresponde ao identificador de E1 (B1). Caso o resultado seja positivo, o casamento se completaria e o nó de produção correspondente à regra seria ativado.

O algoritmo de casamento de padrões Rete tem como desvantagem principal o elevado consumo de memória, devido ao armazenamento dos casamentos completos e parciais entre as regras e os fatos, um problema que se torna especialmente relevante quando o número de regras é elevado. No entanto, para grande parte dos sistemas baseados em regras a troca de memória por velocidade de processamento é proveitosa, conquanto regras escritas de forma errônea possam diminuir consideravelmente tanto a rapidez de processamento como o consumo de memória (GIARRATANO; RILEY, 1993).

### **2.2.2 Busca guiada por objetivo**

A busca guiada por objetivo faz o caminho inverso ao seguido pela busca guiada por dados. Ao invés de iniciar pelos dados disponíveis, começa pelos objetivos que é preciso alcançar, isto é, pelo lado direito das regras. Se para que um determinado objetivo seja alcançado é necessário satisfazer condições de uma regra, essas condições são adicionadas à lista de objetivos, e assim sucessivamente até que a busca chegue aos fatos. Segundo Luger (2004), a busca guiada por objetivo é apropriada quando:

1. Um objetivo ou hipótese é dado na definição do problema, ou pode ser facilmente formulado;
2. Existe um grande número de regras que se aplicam aos fatos do problema, produzindo, assim um número crescente de conclusões ou objetivos;
3. Os dados do problema não são fornecidos, mas devem ser adquiridos pelo sistema para resolução de problemas.

## **2.3 Ferramentas para criação de SEs em Java**

Considerando que os SEs possuem características e componentes em comum, ao longo do tempo foram desenvolvidas diversas ferramentas que fornecem os elementos básicos para a criação de um SE, com a exceção da base de conhecimento, que é específica de cada sistema. Essas ferramentas são conhecidas como “shells”, ou “esqueletos de SEs” (LUGER, 2004).

### **2.3.1 Jess**

Jess (Java Expert System Shell) é uma ferramenta para criação de SEs baseada na linguagem de programação Java, que funciona como um interpretador para a linguagem baseada em regras Jess. O projeto foi criado por Ernest Friedman-Hill como uma reimplementação do *shell* CLIPS (STRAUSS, 2007), mas aos poucos foi se tornando uma linguagem independente, embora ainda conserve a aparência e a sintaxe básica de CLIPS. Assim como CLIPS, Jess utiliza o algoritmo Rete para efetuar o casamento das regras com os fatos inseridos em sua memória de trabalho de forma eficiente.

Por ser escrito em Java, é necessário ter instalada uma máquina virtual Java (JVM) antes de começar a trabalhar com a ferramenta. A versão 7.1 é compatível com todas as versões lançadas de Java a partir da 1.4. Também está disponível para trabalhar com Jess uma extensão para o IDE Eclipse, denominada JessDE, que inclui um editor, um depurador e um visualizador de redes Rete. Para utilizar a

biblioteca Jess é preciso obter uma licença, seja por meio de pagamento, seja por meio de autorização para fins acadêmicos. Uma licença para testes por trinta dias está disponível.

A linguagem de regras Jess tem uma estrutura semelhante à da linguagem Lisp. Cada componente (fatos, regras, chamadas de função etc.) toma a forma de uma lista. Espaços em branco são ignorados (STRAUSS, 2007). Os símbolos, que correspondem essencialmente aos identificadores em outras linguagens, são um conceito central. Símbolos são sensíveis à caixa e não podem começar com dígito. Jess também dá suporte a números de diversos formatos, cadeias de caracteres (*strings*), listas e variáveis (FRIEDMAN-HILL, 2008).

Chamadas de funções usam a notação de prefixo. Por exemplo, para efetuar a soma entre os algarismos 2 e 3 seria utilizada a expressão (+ 2 3), embora o resultado da execução dessa expressão seria o número 5, e não uma lista contendo o número 5. Um grande número de funções estão disponíveis por padrão, e novas funções podem ser definidas tanto na linguagem Jess como em Java. O seguinte exemplo imprime texto na saída padrão: (`printout t "Olá, mundo!" crlf`).

Variáveis são identificadores que iniciam com um ponto de interrogação, que faz parte do nome da variável. Podem se referir a um símbolo, número, cadeia de caracteres ou mesmo a uma lista. A atribuição de um valor a uma variável é possível por meio da função `bind`: (`bind ?x 1`).

Funções são também usadas para fluxo de controle. São incluídas por padrão as funções `if`, `while`, `for`, `try`, que têm papéis semelhantes aos das estruturas de controle homônimas da linguagem Java.

A memória de trabalho de um motor de regras baseado em Jess contém fatos com os quais é possível fazer reagir as regras. Cada fato possui um *template* (modelo), que por sua vez possui uma série de *slots*. Um template funciona como uma tabela de um banco de dados relacional, e os slots seriam como as colunas da tabela. Fatos podem ser adicionados, removidos e modificados, mas no caso de fatos desordenados ou de sombra (*shadow facts*, vinculados a classes Java) é obrigatório definir antes um template. Fatos ordenados não precisam ser declarados com antecedência.

A seguir, um exemplo de declaração de um *template* em Jess, bem como a inserção de um fato:

```
(deftemplate automovel "Um carro qualquer."
  (slot marca)
  (slot modelo)
  (slot ano (type INTEGER))
  (slot cor (default prata)))
(assert (automovel (marca fiat) (modelo palio) (ano 2005)))
```

Figura 2.4: Declaração de um *template* e inserção de um fato em Jess.

Para executar ações baseadas nos fatos da memória de trabalho, Jess permite a criação de regras por meio da função *defrule*. Regras possuem um lado esquerdo (LHS), composto de condições que precisam ser satisfeitas por fatos da memória de trabalho, e um lado direito (RHS), que contém chamadas de funções a serem executadas quando todas as condições estiverem satisfeitas.

Exemplo de regra:

```
(defrule carro-antigo
  "Diz se um carro é antigo"
  (automovel {ano < 1990})
  =>
  (printout t "Esse carro é bem antigo!" crlf))
```

Figura 2.5: Exemplo de regra em Jess.

A implementação do algoritmo Rete pela biblioteca Jess é bastante literal, mas apresenta algumas otimizações. Os vários tipos de nós de rede são representados por subclasses da classe Java `jess.Node`, algumas delas com especializações e variações. A execução da rede é tratada pela classe `Rete`. Jess permite a visualização gráfica da rede por meio de uma função especial, `view`.

Há duas otimizações do algoritmo Rete implementadas por Jess para melhorar o desempenho aumentar a eficiência da execução das regras. A primeira é compartilhar os nós de teste de fatos (nós de entrada única) entre regras que

compartilham condições. A segunda otimização consiste em compartilhar os nós de junção (nós de entrada dupla) que desempenham o mesmo papel para regras diferentes.

O que dificulta a adoção do Jess no projeto é a incompatibilidade da licença utilizada pelo shell, que permite apenas autorização para uso acadêmico, ou comercial mediante pagamento<sup>4</sup>. Como se pretende utilizar apenas bibliotecas de uso livre, preferencialmente de código fonte aberto, tendo em vista a sua implantação em um ambiente operacional real, o qual tem uma política de controle de despesas, a aquisição de uma licença comercial está praticamente descartada.

Ainda assim, a compatibilidade com a linguagem Java e a maturidade tornam o *shell* Jess uma excelente plataforma na área de SEs.

### 2.3.2 Drools

Drools é um sistema gerenciador de regras de negócio (BRMS), desenvolvido em Java e de código fonte aberto, que utiliza uma versão aprimorada do algoritmo Rete (DROOLS, 2006). Mais apropriadamente chamado de sistema de regras de produção, é atualmente desenvolvido pela empresa Red Hat<sup>5</sup> e implementa por completo a API de motores de regras JSR94 (JBOSS, 2011).

A plataforma Drools inclui diversos componentes, dos quais apenas o Drools Expert (motor de regras) será estudado aqui. O Drools Expert utiliza como mecanismo de inferência o algoritmo Rete, mais precisamente o Rete Orientado a Objetos (ReteOO), que é uma adaptação feita por Bob McWhirter, autor original do Drools, para possibilitar o melhor aproveitamento dos recursos disponíveis em linguagens orientadas a objetos (DROOLS, 2006).

O Drools Expert utiliza como método de execução de regras o encadeamento progressivo, ou raciocínio guiado por dados, e como estratégia de resolução de conflitos, saliência e LIFO.

---

<sup>4</sup> Disponível em: <<http://www.jessrules.com/download.shtml>>. Acesso em: 25 set. 2012.

<sup>5</sup> Disponível em: <<http://www.redhat.com>>. Acesso em: 29 nov. 2012.

Fatos podem ser definidos em Java ou na própria linguagem de regras. Quando escritos em Java, precisam ter métodos de acesso definidos para cada atributo, a fim de que estes possam ser referenciados pela máquina de inferência. Os exemplos exibidos na figura 2.6 demonstram como declarar um fato diretamente no arquivo de regras, usando a própria linguagem de regras, e como declarar uma regra propriamente dita.

<pre> <b>rule</b> "Carro antigo" when     Automovel( ano &lt; 1990 ) then     System.out.println( "Carro antigo!" ) end </pre>	<pre> <b>declare</b> Automovel     marca : String     modelo : String     ano : <b>int</b>     cor : String end </pre>
(a)	(b)

Figura 2.6: Definição de uma regra (a) e declaração de um fato (b) na linguagem de regras Drools.

Drools dispõe de uma Memória de Trabalho (*working memory*), que armazena os fatos e permite executar consultas (*queries*) e acessar pontos de entrada (*entry points*), os quais possibilitam a inserção, retração e atualização de fatos na própria memória de trabalho. A ordem de execução das regras é controlada pela Agenda (DROOLS, 2006).

Por ser escrito em Java e ter uma linguagem de regras com sintaxe orientada a objetos e integrada com Java, o uso da plataforma Drools seria bastante indicado para um projeto de SE desenvolvido em Java. No entanto, por tratar-se de um motor de regras com mecanismo de inferência que faz uso de raciocínio guiado a dados, ocorre a dificuldade de se identificar quais dados deverão ser obtidos por questionamento ao usuário. Tal dificuldade não existiria, ou seria muito menor, se fosse utilizado raciocínio guiado a objetivo. Existe um esforço em andamento para tornar o Drools Expert compatível com raciocínio guiado a objetivo, mas a

implementação ainda não alcançou aparentemente um nível de maturidade adequado (PROCTOR, 2011).

## **2.4 Conclusões**

SEs podem substituir eficazmente o especialista humano, desde que o domínio de aplicação seja adequadamente restrito e que a base de conhecimento seja ampla e aprofundada. Entre os seus componentes, são de fundamental importância a base de conhecimento, sem a qual o SE nada sabe, e o mecanismo de inferência, que torna possível determinar quais regras podem ser aplicadas a cada situação.

No desenvolvimento de um SE, as etapas não são claramente definidas, como o são as fases de um sistema computacional tradicional, mas essencialmente evolucionárias. Durante todo o processo é necessário realizar contínuos testes e avaliações, a fim de descobrir falhas e aprimorar o sistema.

A escolha entre as estratégias de busca guiada por dados e guiada por objetivos depende das características de cada problema, levando em consideração condições tais como a quantidade de dados disponível na formulação inicial do problema e a dificuldade de formular um objetivo ou hipótese. Na busca guiada por dados é fundamental a utilização de um algoritmo eficiente como o Rete.

Das ferramentas disponíveis para desenvolvimento de SEs baseados em regras, Jess apresenta uma grande flexibilidade para o desenvolvedor, possuindo inclusive suporte a busca guiada por objetivos, mas a sua licença restritiva dificulta a adoção no presente trabalho. Drools, por sua vez, embora tenha uma licença livre e uma sintaxe mais semelhante à da linguagem Java, permite apenas a estratégia de busca guiada a dados.



### 3 ESTADO DA ARTE

Neste capítulo serão apresentadas as mais recentes contribuições nas áreas de pesquisa de SEs e de uso de sistemas inteligentes no apoio ao suporte técnico.

#### 3.1 SEs recentes

SEs têm diversas áreas de aplicação, o que os torna um tema bastante produtivo para pesquisa e desenvolvimento de soluções.

Brotto (2009) apresenta um SE para avaliação da probabilidade de sucesso de empreendedores na abertura de novos negócios, com base em características pessoais e profissionais do empreendedor, do negócio e do mercado. O desenvolvimento iniciou-se com uma consulta a cinco especialistas em empreendedorismo e pequenas empresas, que listaram as características que indicassem sucesso ou fracasso de um negócio, as quais foram, a partir de então, colhidas junto a um grupo de empreendedores da cidade de Toledo (PR). Por fim, foi efetuada ainda uma pesquisa para testar a ferramenta criada. O sistema adota uma abordagem de rede Bayesiana, que é um modo de representação de um SBC por meio de estruturas gráficas formalizado pela teoria da probabilidade; tem a forma de grafo acíclico direcionado, com nós que representam as variáveis e suas medidas de incerteza e arcos que denotam a existência causal entre os nós conectados; e gera resultados satisfatórios quando o domínio apresenta características aleatórias (BROTTO, 2009). O *shell Spirit*, essencialmente uma ferramenta de protótipo, foi escolhido para a implementação do trabalho, devido à facilidade oferecida pela interface gráfica e à portabilidade da linguagem de programação Java, na qual o Spirit foi implementado.

Também com base na utilização de redes Bayesianas, Lange (2011) apresenta uma proposta de SE aplicado à gestão de manutenção da distribuição de energia elétrica. Como as perturbações no sistema de transmissão e distribuição são a maior causa de interrupção no fornecimento de energia, buscou-se tornar eficiente a gestão da manutenção, preterindo as manutenções corretivas em favor da

manutenção preventiva. Para isso, foi elaborado um *ranking* das zonas críticas quanto à confiabilidade, o qual oferece à gestão da manutenção subsídios no momento de decidir quais zonas deverão passar por manutenção preventiva. A solução proposta divide-se em duas partes: o Sistema de Suporte à Decisão (SSD) e o Sistema Especialista Probabilístico (SEP). O primeiro oferece suporte computacional às atividades necessárias para estabelecer o ranking das zonas de confiabilidade, ao passo que o segundo, fazendo uso de redes Bayesianas, julga/avalia as características físicas, técnicas e estratégicas associadas aos equipamentos de cada zona do sistema de distribuição.

Mossin (2012) propõe a utilização de sistemas inteligentes para diagnosticar e localizar falhas em redes de comunicação que utilizam o protocolo Profibus DP. O protocolo de rede industrial Profibus, que conta com três perfis, entre os quais o Profibus DP, permite a integração de equipamentos de diferentes fabricantes e a padronização das redes industriais. Nesse protocolo, os dispositivos dividem-se em mestres, que determinam a comunicação de dados em um barramento; e escravos, que não têm direito de acesso ao barramento, enviando ou recebendo informações apenas quando solicitado pelo mestre. O trabalho apresenta uma análise da camada física com o uso de redes neurais artificiais, para fins de classificação dos sinais elétricos de acordo com a sua forma de onda. Em caso de deformação nos sinais, são apresentadas possíveis soluções. SEs são usados para analisar o nível médio de tensão e os telegramas transmitidos pela camada de enlace do protocolo. É proposto ainda um sistema nebuloso (*fuzzy*), cuja função é calcular a *Target Rotation Time* ( $T_{TR}$ ), que se refere ao intervalo entre o início e o término da comunicação dos mestres com todos os dispositivos escravos da rede.

### **3.2 Trabalhos de IA na área de suporte**

Diversas soluções têm sido apresentadas na área de auxílio inteligente à resolução de problemas semelhantes na área de suporte de TI e assistência técnica.

Grossmann (2002) propõe o SADPC (Sistema de Auxílio ao Diagnóstico de Problemas em Computadores), um SE com uso de Raciocínio Baseado em Casos

(RBC) para diagnosticar e encontrar soluções para problemas de microcomputadores. O sistema dispõe de uma base de casos inicial formada a partir de mais de 10.000 ordens de serviço de problemas reais resolvidos, os quais foram filtrados, organizados e transformados manualmente. Num segundo momento, novos casos gerados pela interação com os usuários do sistema foram acrescentados à base de casos. Desenvolvido em Delphi, a interface do aplicativo é convencional (*desktop*) e permite o acesso a três tipos de usuários: especialistas, responsáveis por resolver problemas não cobertos pela base de conhecimento; engenheiros do conhecimento, que detêm o conhecimento sobre o processo de aquisição do conhecimento, agindo como elos de ligação entre os técnicos e o SE; e usuários leigos, principais beneficiários do uso do SE, que fazem a consulta ao sistema em busca de solução. Quando um novo caso é inserido por meio de uma consulta, as possíveis soluções são apresentadas, ordenadas pelo número de palavras descritivas em comum com o novo caso.

Zielinski e Bortoleto (2007) apresentam um estudo sobre a implantação de um SBC, denominado *e-HelpDesk*, como canal direto entre a empresa que presta suporte e o cliente final de um *software*, com o intuito de tornar possível atender com maior rapidez e eficácia os pedidos de assistência técnica, aproveitando a repetitividade das ocorrências. Faz-se uso de RBC para classificar as soluções apresentadas anteriormente, e novas ocorrências são assimiladas pela base de casos assim que são indicadas como “resolvidas”. O usuário, com acesso via internet ao sistema, pode iniciar o registro de uma solicitação de suporte técnico e preencher questões pré-definidas. O SBC tenta, então, responder a pergunta feita pelo cliente; caso não seja possível, a solicitação é automaticamente encaminhada para o técnico responsável.

Wang et al. (2010) estudam soluções para dois dos principais desafios que sistemas de suporte técnico inteligentes, especificamente os baseados em casos, enfrentam: a dificuldade inerente a tais sistemas quanto à classificação dos casos recuperados com base em casamento de palavras-chaves; e a representação dos resultados, que leva os usuários a analisarem os casos um a um para encontrar as soluções adequadas. O primeiro problema expõe as limitações da ordenação pelo

casamento de palavras-chaves, que frequentemente não conseguem capturar a relação semântica entre a solicitação e os casos passados; por exemplo, dada uma questão em inglês: “*can you switch the computers?*”, a maioria dos sistemas baseados em RBC retornaria casos relacionados a *switches* de rede. É proposto como solução para esse problema um método para calcular a similaridade semântica entre as sentenças dos casos anteriores e a solicitação atual, utilizando uma análise do papel semântico. Quanto ao segundo problema, representação dos resultados, foi proposta como possível solução organizar os casos mais relevantes em grupos, de acordo com o contexto ou cenário, e sumarizar cada grupo de casos gerado, a fim de aprimorar a usabilidade do sistema.

### **3.3 Comparação deste trabalho com os trabalhos estudados**

Brotto (2009) e Lange (2011) fazem uso da abordagem Bayesiana para cálculo de probabilidades, devido à necessidade de representar a natureza incerta do conhecimento. Neste trabalho, tal necessidade não foi identificada até o presente momento, tornando dispensável ou desnecessária a aplicação de tal técnica.

O foco do presente trabalho será resolver incidentes de TI, aplicando a manutenção corretiva, diferentemente de Lange (2011), cujo objetivo é fornecer subsídios para a aplicação de manutenção preventiva, antes que os incidentes aconteçam, evitando a interrupção do fornecimento de energia.

Mossin (2012) utiliza a lógica nebulosa (*fuzzy*), que possibilita lidar com a incerteza do conhecimento, além de ser capaz de modelar o raciocínio de senso comum (GIARRATANO; RILEY, 1993). Porém, para o presente trabalho, a lógica clássica foi considerada suficiente.

De acordo com Wangengeim (2003), sistemas de suporte técnico (*help desk*) ao cliente baseados em RBC vêm adquirindo importância crescente. Esses sistemas podem servir de auxílio aos funcionários no processo de encontrar uma solução para os problemas dos clientes, homogeneizando a qualidade do atendimento e possibilitando que até mesmo funcionários menos experientes forneçam soluções de

qualidade a perguntas complexas. Também podem atender diretamente o cliente, sem a mediação de um operador humano.

Em Grossman (2002), Zielinski e Bortoleto (2007) e Wang et al. (2010) são apresentados sistemas nesse contexto de RBC aplicado ao atendimento direto ao usuário final (ou cliente). Neste trabalho, no entanto, julgou-se mais apropriado, devido à natureza pouco consistente e incompleta do histórico de atendimentos, a utilização do Raciocínio Baseado em Regras (RBR) num primeiro momento, permanecendo aberta a possibilidade de adição de técnicas de RBC como complemento no futuro.

### **3.4 Conclusões**

Atualmente são várias as técnicas para o desenvolvimento de SEs, e cada qual é apropriada a determinados cenários ou domínios de conhecimento mais do que a outros. São também diversas as ferramentas de desenvolvimento (*shells*), aplicações cuja conveniência se encontra principalmente na facilidade que conferem ao rápido desenvolvimento de protótipos.

SEs encontram uso na prevenção de fracassos de negócios, como em Brotto (2009), e de interrupções no fornecimento de energia, de acordo com Lange (2011). Ao mesmo tempo, são úteis para identificar ou diagnosticar falhas ou problemas existentes, como em Mossin (2012) e nos sistemas de suporte técnico baseados em RBC.

De fato, a eficácia dos SEs tanto na prevenção como na resolução de incidentes e problemas demonstra-se claramente nas aplicações bem-sucedidas apresentadas, corroborando e solidificando a proposta do presente trabalho.

## **4 DESENVOLVIMENTO**

Neste capítulo serão apresentadas as etapas executadas durante o projeto e a implementação do SE desenvolvido, assim como os resultados obtidos com a implantação.

### **4.1 Fase da identificação**

Como primeira etapa no desenvolvimento do projeto, a fase de identificação foi utilizada para determinar os aspectos básicos do SE e as estruturas necessárias de suporte ao seu desenvolvimento.

Nessa etapa, em primeiro lugar foram indicados os participantes envolvidos no projeto. Ao autor do projeto foi atribuída a função de engenheiro do conhecimento e também a de programador do sistema, tornando-se responsável tanto pelos aspectos conceituais e formais como pelos aspectos técnicos de implementação. Para auxiliá-lo no processo de aquisição e validação do conhecimento, foram identificados como especialistas os técnicos responsáveis pelo atendimento dos incidentes de informática no IPREV, tendo em vista o aprofundado conhecimento que estes detêm, formado ao longo dos anos em alguns casos. Ao IPREV, na pessoa de seu gerente de TI, foi atribuído o papel de dono do SE.

A identificação das fontes de conhecimento veio em seguida. A principal fonte para a aquisição do conhecimento foram as ordens de serviço registradas pela GETIG, visto que elas contêm as soluções apresentadas no passado a problemas que podem vir a acontecer no futuro. Uma vez que esses registros nem sempre são completos e podem estar até mesmo incorretos ou obsoletos, os especialistas, como supracitado, tiveram atribuição fundamental no processo de transformação desses dados em conhecimento reutilizável, validando as regras geradas e acrescentando sempre que necessário o conhecimento faltante. Livros técnicos, manuais e até informações relevantes e confiáveis disponíveis na internet foram eventualmente empregados como fontes complementares.

O cronograma de desenvolvimento do SE apresentou duas limitações de grande significância. Por um lado, o processo de construção de um SE real apresenta uma grande demanda de tempo, tanto do engenheiro do conhecimento quanto do especialista (RABUSKE, 1995). Por outro lado, os prazos estabelecidos para conclusão do projeto foram limitados, de tal maneira que foi preciso fazer adaptações no cronograma e estender as horas trabalhadas, na medida do possível.

Quanto aos recursos computacionais e financeiros, vinculados entre si, decidiu-se não adquirir equipamentos especificamente para o desenvolvimento do projeto, tendo em vista que os recursos computacionais já disponíveis são suficientes. Na eventualidade de surgir a necessidade de instalar a aplicação em um servidor dedicado, foram aproveitados servidores de rede sem uso ou pouco usados, cortando custos, e, com o uso de ferramentas e bibliotecas de licença livre, foram evitados também gastos com licenças de *software*.

#### **4.1.1 Identificação das características do problema**

Antes de proceder à identificação das características do problema, convém fazer alguns esclarecimentos quanto à terminologia adotada neste trabalho. De acordo com o Information Technology Infrastructure Library (ITIL), “incidente” é qualquer interrupção ou redução da qualidade de um serviço de TI, ou até uma falha de um item de configuração que não tenha ainda causado impacto nos serviços, ao passo que “problema” seria a causa ou raiz de mais de um incidente (OGC, 2007).

O SE desenvolvido por este projeto tentará resolver os incidentes da área de TI, oferecendo uma solução a partir do conhecimento da causa desses incidentes. Aqui, portanto, os problemas sem causa conhecida não serão tratados. Os incidentes, porém, poderão ser considerados os “problemas” a serem resolvidos, numa definição menos restrita, fora do âmbito do ITIL.

A ocorrência de um incidente deve sempre estar ligada a um usuário, que geralmente é aquele que o relata ao sistema e será o responsável por interagir com o SE e informar os dados necessários, isto é, os “sintomas” ou efeitos observados

por ele durante a ocorrência do incidente. A correta descrição desses dados é essencial para que o diagnóstico efetuado pelo SE seja preciso.

Também podem estar ligados a um incidente um equipamento, um local ou setor, a data e hora da ocorrência, entre outros campos, que podem auxiliar no processo de descoberta da causa do problema.

Em se tratando de solução, é necessário que a sua eficácia seja comprovada e que o usuário incumbido de aplicá-la tenha a habilidade mínima para executá-la sem prejudicar o funcionamento dos serviços. Propôs-se que apenas soluções já aplicadas ao problema no passado sejam acionadas, e que os critérios para inclusão de uma solução na base de conhecimento sejam restritos. Ademais, cada solução deve exibir todos os passos necessários para sua implantação correta.

## **4.2 Fase da conceituação**

Na etapa de conceituação, foi realizada uma classificação geral e um nivelamento por nível de dificuldade dos atendimentos do ano de 2012, a fim de verificar quais incidentes poderiam ser atendidos pelo SE.

Efetou-se também um trabalho abrangente e detalhado de análise de todas as ordens de serviço existentes no histórico da GETIG, com o propósito de extração do conhecimento.

### **4.2.1 Classificação dos atendimentos**

Os atendimentos realizados pela gerência vão desde ligar um estabilizador até a criação de uma conta de e-mail, passando pela configuração de um aplicativo de processamento de texto. Tal variabilidade dificulta, se não impede, a análise da demanda por tipo de serviço, que seria necessária para determinar a eficácia de uso de um SE que auxilie na tarefa de encontrar e resolver problemas. Portanto, os atendimentos foram organizados em categorias ou tipos de serviços, que são listadas a seguir:



- **Administração de e-mail:** Criação e manutenção de contas de e-mail.
- **Administração de rede:** Criação e manutenção de contas de acesso à rede, administração dos servidores de rede, manutenção do acesso à Internet.
- **Configuração de impressora:** Configuração e manutenção de impressoras e multifuncionais, incluindo a instalação e configuração da impressora no sistema operacional.
- **Configuração de rede:** Configuração dos dispositivos de rede do sistema operacional (IP, DNS etc.), configuração de domínio e grupo de trabalho, mapeamento de pastas compartilhadas.
- **Configuração de software:** Configuração de aplicativos que possibilitam a execução das mais variadas atividades, como processadores de texto, planilhas eletrônicas, navegadores de internet e de arquivos.
- **Instalação de equipamento:** Instalação, mudança e troca de equipamentos e periféricos.
- **Instalação de software:** Instalação ou reinstalação de um aplicativo para execução de uma determinada tarefa.
- **Manutenção de hardware:** Dar manutenção a computadores, periféricos e demais equipamentos físicos, efetuando a troca de peças e equipamentos quando necessário.
- **Manutenção de rede:** Dar manutenção aos equipamentos de rede e internet, efetuando a troca e manutenção de cabos e equipamentos quando necessário.
- **Manutenção de software:** Serviço que visa a garantir o funcionamento correto de um aplicativo, por meio da correção de um problema existente ou que possa vir a acontecer no futuro.

- **Operação de equipamento:** Executar as atividades inerentes às funções dos funcionários, por inabilidade ou falta de capacitação destes para executar tais atividades.
- **Outros:** Serviços não categorizados.

Após definidas as categorias, procedeu-se à classificação dos atendimentos realizados pela gerência no ano de 2012. No total foram analisados 1353 atendimentos. Os resultados foram então tabelados e estão sumarizados em gráfico (Figura 4.1).

Os gráficos indicam que quase metade de todos os atendimentos estão relacionados a software (instalação, manutenção e configuração) em todos os trimestres. As tarefas de manutenção e configuração de rede aumentaram consideravelmente entre o primeiro e o último semestre, passando de menos de 5% para mais de 20%, tendência que não se repetiu em nenhuma das outras áreas, as quais se mantiveram relativamente estáveis. Tal variação poderia indicar um problema na infraestrutura de rede, por exemplo, embora uma investigação mais detalhada seria necessária. É notável ainda que as atividades de configuração de impressora conservaram um percentual praticamente igual entre o primeiro e o terceiro trimestres.

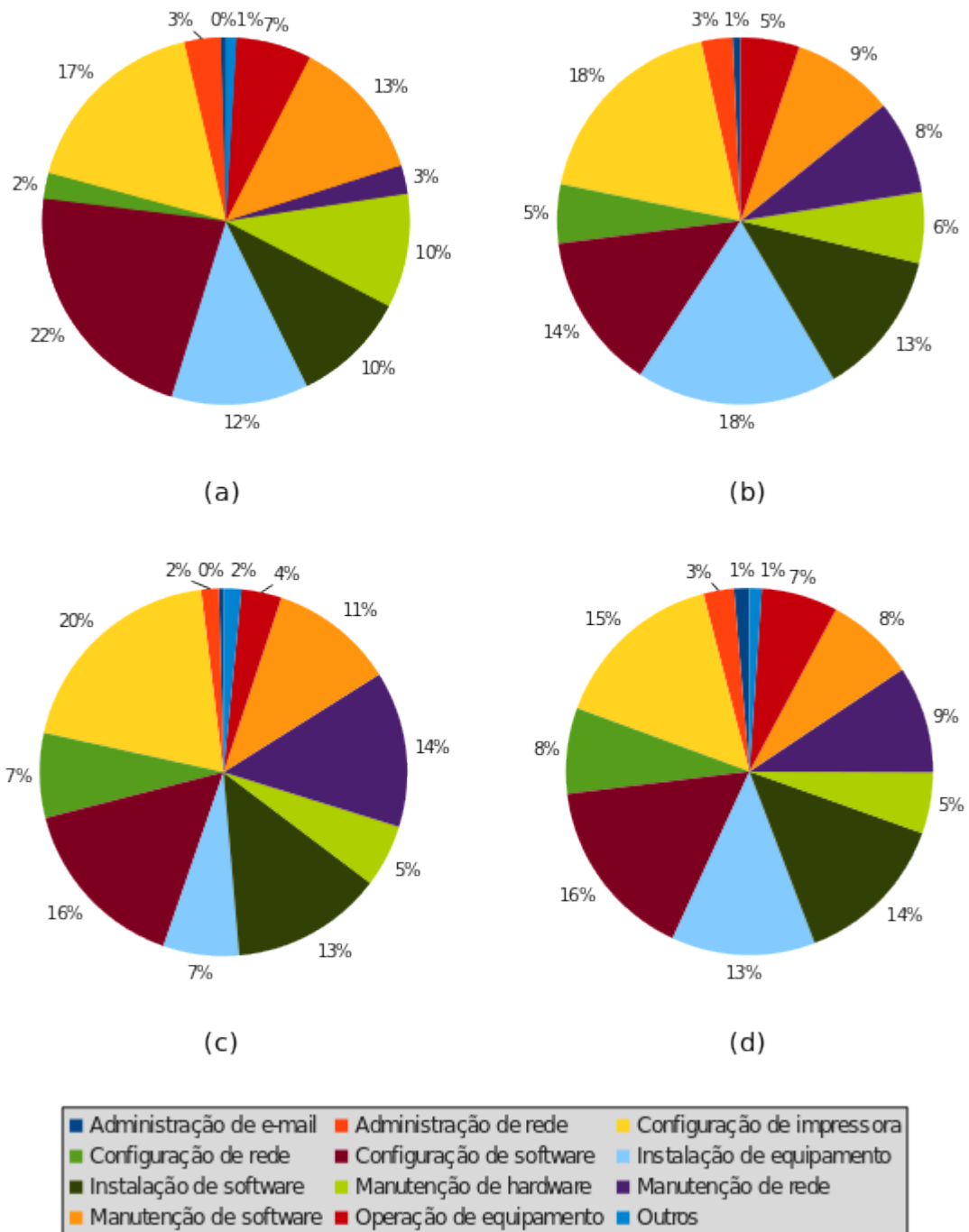


Figura 4.1: Classificação dos atendimentos realizados em 2012, desde o primeiro (a) até o quarto (d) trimestre.

#### 4.2.2 Níveis de dificuldade

Para medir a eficácia da adoção de uma tecnologia baseada em SEs, não seria suficiente categorizar os atendimentos somente, mas também identificar o grau de dificuldade encontrado na realização dos atendimentos. Haja vista que os dados disponíveis não incluem essa informação, foi necessário proceder a um nivelamento *a posteriori* dos atendimentos, tendo por base a descrição dos serviços prestados. Foram estabelecidos três níveis:

- **Fácil:** o usuário padrão teria condições de executar os procedimentos necessários para a realização do serviço, desde que recebesse a orientação necessária. Exemplos: conectar um cabo de rede desconectado, reiniciar o computador.
- **Médio:** o usuário padrão poderia ou não realizar a atividade, ou a atividade necessita da presença de um técnico. Exemplos: instalação de um aplicativo, configuração de um escâner.
- **Difícil:** o serviço só poderia ser realizado por um especialista e/ou responsável. Exemplo: instalação de um equipamento, criação de uma conta de e-mail oficial.

Definidos os níveis de dificuldade, procedeu-se ao nivelamento dos atendimentos realizados no ano de 2012 pela gerência. Os resultados foram então tabelados e estão sumarizados em gráfico (Figura 4.2).

Percebe-se que nos quatro trimestres, embora os percentuais sejam relativamente bem divididos entre os três níveis, há um predomínio dos atendimentos com nível de dificuldade “médio”. Isso demonstra que para grande parte das atividades existe incerteza quanto à possibilidade de realização por não especialistas. Por outro lado, os graus de dificuldade “fácil” e “difícil” mantêm equilíbrio entre si, com cerca de 30% dos atendimentos cada.

Considerando que todo atendimento que poderia ser realizado pelo próprio usuário é passível de ser resolvido com o auxílio de um SE, tem-se que todas as

atividades com nível de dificuldade “fácil” e algumas com nível de dificuldade “médio” poderiam ser realizadas dessa forma, de tal modo que a demanda por atendimentos diretos poderia diminuir em mais de um terço do total.

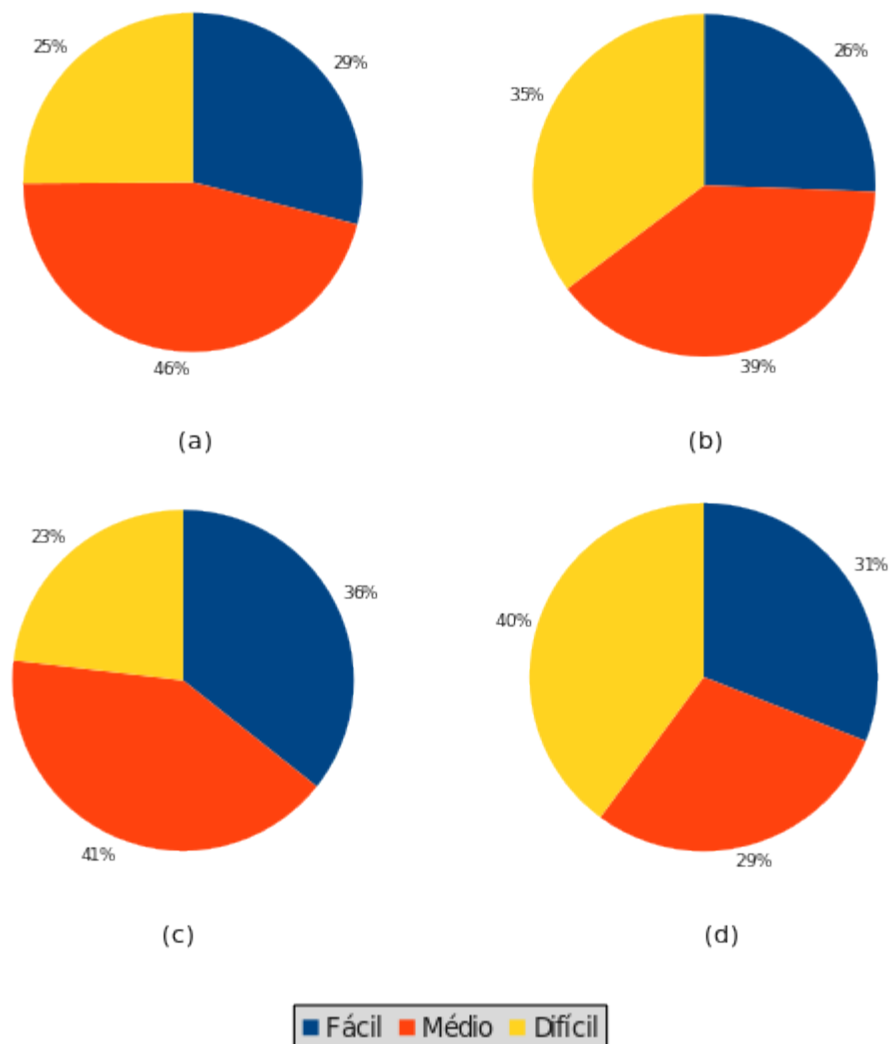


Figura 4.2: Nivelamento por dificuldade dos atendimentos prestados no ano de 2012, desde o primeiro (a) até o quarto (d) trimestre.

### 4.2.3 Formação do esboço das regras

Como este trabalho tem como principal fonte do conhecimento o histórico de ordens de serviço, e tendo por base o princípio segundo o qual um SE necessita de uma base de conhecimento abrangente e aprofundada, deu-se início a um processo longo e detalhado de análise dos atendimentos registrados desde o ano de 2011.

Para auxiliar na tarefa de encontrar os problemas e soluções semelhantes existentes no histórico, foi utilizado o cálculo da distância de Liechtenstein, que possibilita identificar a distância entre duas cadeias de caracteres (LEVENSHTAIN, 1966). Com base nesse cálculo, as ordens de serviço foram agrupadas pela descrição do problema, permitindo o cruzamento entre os problemas e as soluções.

O cruzamento possibilitou a construção de um esboço da base de conhecimento, ainda de maneira informal, que foi validado e completado pelos especialistas. Para cada par problema/solução foram definidas as condições para que a solução seja aplicável ao problema, e foi estabelecido o grau de dificuldade para implantar a solução, como visível na figura 4.3.

<p><b>Problema geral:</b> Computador não liga</p> <p><b>Problema:</b> Cabo de energia desconectado</p> <p><b>Condições:</b> Computador não liga, não há sinal de vídeo</p> <p><b>Solução:</b> Conectar o cabo de energia do PC</p> <p><b>Grau de dificuldade:</b> Fácil</p>
---

Figura 4.3: Demonstração de um par problema/solução identificado no processo de esboço das regras.

### 4.3 Fase de formalização

A identificação de estruturas de suporte à representação e armazenamento já existentes foi um dos alvos desde o início do projeto, com o propósito claro de reutilizar ao máximo as soluções já existentes, permitindo que o foco do desenvolvimento se mantivesse no processo de aquisição e aplicação do

conhecimento. Porém, foram antes estabelecidos alguns requisitos básicos para a verificação da adequação de uma ferramenta:

- Compatibilidade com a linguagem Java, possibilitando o acesso a partir de código escrito nessa linguagem de programação.
- Distribuição sob uma licença compatível com a GNU GPL, versão 2 ou superior.
- Possibilidade de uso do raciocínio guiado a objetivos (encadeamento regressivo).

Durante todo o processo de revisão da literatura e verificação do estado da arte, assim como nas etapas iniciais de desenvolvimento do SE, observou-se uma grande dificuldade de se optar por uma das ferramentas experimentadas. Ora por ter uma licença incompatível, ora por não prover os mecanismos necessários para a implementação, nenhum *shell* demonstrou atender a todos os requisitos básicos do trabalho.

Em decorrência de tal situação, decidiu-se implementar os componentes essenciais do SE, descartando a adoção de qualquer ferramenta para desenvolvimento de SE existente. Entretanto, foi definido que se faria uso de regras de produção na representação do conhecimento, por sua modularidade, naturalidade e uniformidade (RABUSKE, 1995). Luger (2004) cita ainda como vantagem dos sistemas de produção baseados em regras a separação entre conhecimento e controle, que torna possível que a base de conhecimento seja alterada sem requerer modificação das estruturas de controle do programa, e vice-versa.

Teve início, então, a transposição do conhecimento esboçado na etapa de conceituação para as regras de produção propriamente ditas, escritas de maneira formal. A forma específica adotada foi a do *shell* EXSYS, demonstrada na figura 4.4, que utiliza vocábulos da língua portuguesa para expressar as diferentes partes da regra (RABUSKE, 1995).

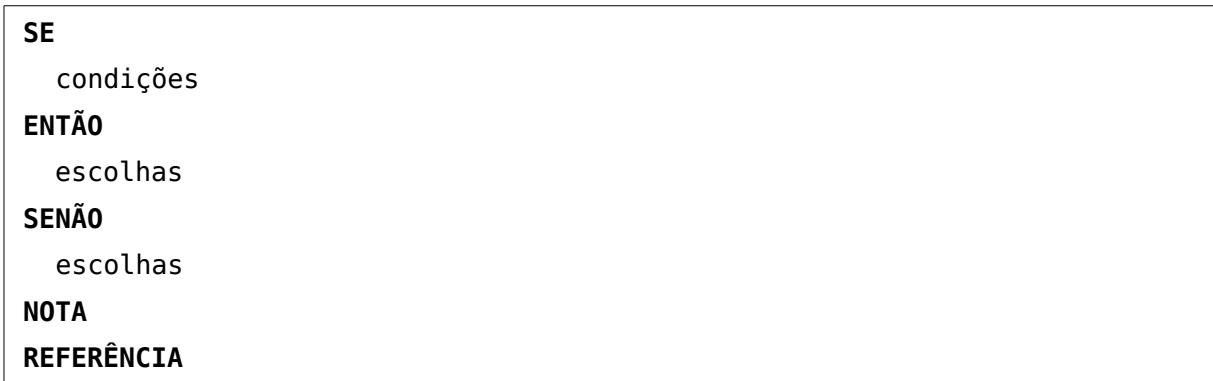


Figura 4.4: Estrutura de regra de produção adotada.  
Fonte: Rabuske (1995, p. 110).

Apenas os componentes obrigatórios (SE e ENTÃO) serão utilizados neste trabalho, por motivo de simplicidade. Por exemplo, o conhecimento contido na figura 4.3 pode ser representado formalmente pela regra de produção a seguir:

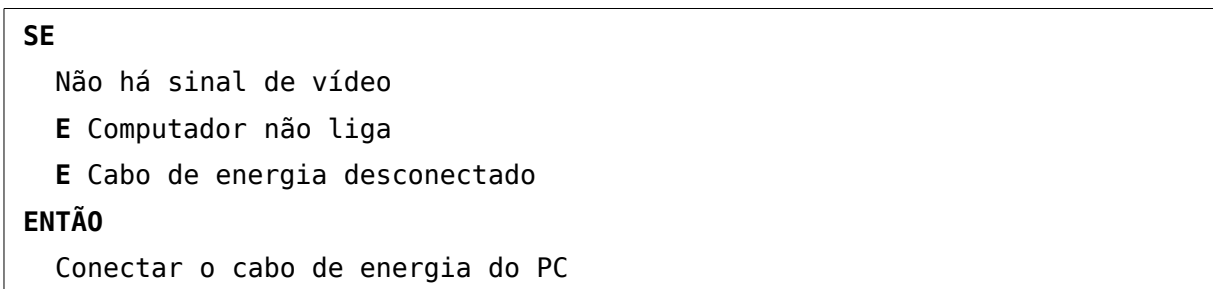


Figura 4.5: Especificação de uma regra de produção.

#### 4.4 Fase da implementação

Na fase de implementação foram definitivamente escolhidos os mecanismos de representação do conhecimento, e teve início a codificação dos programas necessários para processar as estruturas de controle do SE.

Uma das medidas iniciais dessa etapa foi atribuir um nome ao SE, com a intenção de construir uma identidade própria, que o tornasse mais facilmente reconhecido e referenciado. O sistema passou então a ser denominado Suporte Emergencial Inteligente (SEI), cuja sigla evidencia intencionalmente um dos efeitos esperados com a sua implantação: a difusão do conhecimento.



#### 4.4.1 Especificação de requisitos

A especificação de requisitos consiste em um documento, elaborado antes do início do desenvolvimento, que contém uma descrição detalhada das características de um projeto (PRESSMAN, 2011). Dividem-se normalmente em requisitos funcionais e não funcionais.

##### 4.4.1.1 Requisitos funcionais:

De acordo com Ramos (2006), os requisitos funcionais especificam as atividades ou funções que um sistema deve contemplar. Os seguintes requisitos funcionais foram estabelecidos para o SEI:

1. **Iniciar consulta:** o usuário deve poder iniciar uma sessão de consulta ao sistema, tendo em vista encontrar uma solução para um incidente de TI.
1. **Responder pergunta:** ao ser questionado sobre uma determinada informação, o usuário poderá responder a pergunta.
2. **Desfazer resposta:** ao usuário é facultado, depois de responder uma pergunta, descartar a resposta fornecida e dar uma nova resposta.
3. **Reiniciar consulta:** a qualquer momento deve ser possível reiniciar a consulta, sendo descartadas todas as respostas fornecidas até então.
4. **Avaliar o sistema:** o usuário deve poder enviar avaliações e comentários sobre o sistema e sua utilização, auxiliando no aprimoramento do mesmo.
5. **Iniciar sessão:** o administrador deve poder autenticar-se no sistema, a fim de obter permissão para executar as tarefas administrativas.
6. **Alterar informações:** o administrador, uma vez autenticado, pode alterar informações a respeito da base de conhecimento, como o nome, a descrição ou o autor.

7. **Adicionar variável:** o administrador pode adicionar uma variável à base de conhecimento.
8. **Editar variável:** o administrador pode editar uma variável já incluída na base de conhecimento.
9. **Remover variável:** o administrador pode remover uma variável.
10. **Adicionar regra:** o administrador pode adicionar uma regra à base de conhecimento.
11. **Editar regra:** o administrador pode editar uma regra já incluída na base de conhecimento.
12. **Remover regra:** o administrador pode remover uma regra.

#### 4.4.1.2 Requisitos não funcionais:

Os requisitos não funcionais estabelecem condições que o sistema deve satisfazer (RAMOS, 2006). Abaixo, são listados os requisitos não funcionais do sistema proposto:

1. **Especificação de projeto:** deve ser produzida especificação de projeto baseada em Unified Modeling Language (UML), segunda versão, contendo diagramas de classes e de casos de uso.
2. **Plataforma de desenvolvimento:** tanto o *shell* como a interface Web devem ser desenvolvidos na linguagem de programação Java, versão 6 ou posterior, utilizando como ambiente integrado de desenvolvimento o Eclipse<sup>6</sup>.
3. **Desenvolvimento do shell:** deve ser desenvolvida uma ferramenta para desenvolvimento de SEs (*shell*) independente, que permita o fácil acoplamento de diferentes interfaces.

---

<sup>6</sup> Disponível em: <<http://www.eclipse.org/>>. Acesso em: 10 maio 2013.

4. **Plataforma web:** como servidor Web e *container* de *servlets*, deve ser empregado o Apache Tomcat 7<sup>7</sup>, e o *framework* Stripes<sup>8</sup>, por sua simplicidade de configuração e facilidade de aprendizagem, será utilizado na construção da interface Web.
5. **Interface com os usuários:** deve ser desenvolvida uma interface Web para interação dos usuários, que, para não restringir o número de usuários do sistema, deve necessariamente ser compatível com o navegador Internet Explore 7 ou superior e com os mais modernos navegadores.
6. **Interface de administração:** deve ser desenvolvida uma interface Web para administração do sistema e alteração da base de conhecimento, compatível com os mais modernos navegadores.
7. **Persistência de dados:** o sistema deve utilizar como banco de dados relacional para armazenamento dos dados o servidor PostgreSQL<sup>9</sup>, versão 9 ou superior.

#### 4.4.2 Modelagem

Antes de proceder à programação propriamente dita, foi elaborada a modelagem gráfica do sistema de *software*, com o uso da Unified Modeling Language (UML). Segundo Folerca (2005, p. 25), a UML “é uma família de notações gráficas, apoiada por um metamodelo único, que ajuda na descrição e no projeto de sistemas de software, particularmente daqueles construídos utilizando o estilo orientado a objetos.

Existem diferentes maneiras de se usar a UML em um projeto de *software*: como esboço, com foco na especificidade, para transmitir alguns aspectos, ideias e alternativas de implementação importantes antes do início da programação; como projeto, com foco na completeza, em que os diagramas são desenvolvidos por um projetista e contêm todas as decisões, de todo o projeto ou de uma área em

<sup>7</sup> Disponível em: <<http://tomcat.apache.org/>>. Acesso em: 10 maio 2013.

<sup>8</sup> Disponível em: <<http://stripesframework.org/>>. Acesso em: 10 maio 2013.

<sup>9</sup> Disponível em: <<http://www.postgresql.org/>>. Acesso em: 10 maio 2013.

particular; e como linguagem de programação, que gera o código compilado a partir dos próprios diagramas (FOWLER, 2005).

Para este trabalho, adotou-se uma abordagem mista; a UML foi usada tanto como esboço, para clarificar os detalhes mais importantes do sistema a ser desenvolvido, quanto como projeto, especificando detalhes das decisões de implementação tomadas.

#### 4.4.2.1 Diagrama de classes

Diagramas de classes têm o papel de representar os tipos de objetos existentes no sistema, incluindo os relacionamentos estáticos existentes entre eles. Exibem também, as propriedades das classes, em forma de atributos e associações, e as suas operações (FOWLER, 2005). A figura 4.6 contém o diagrama de classes do *shell* SEI.

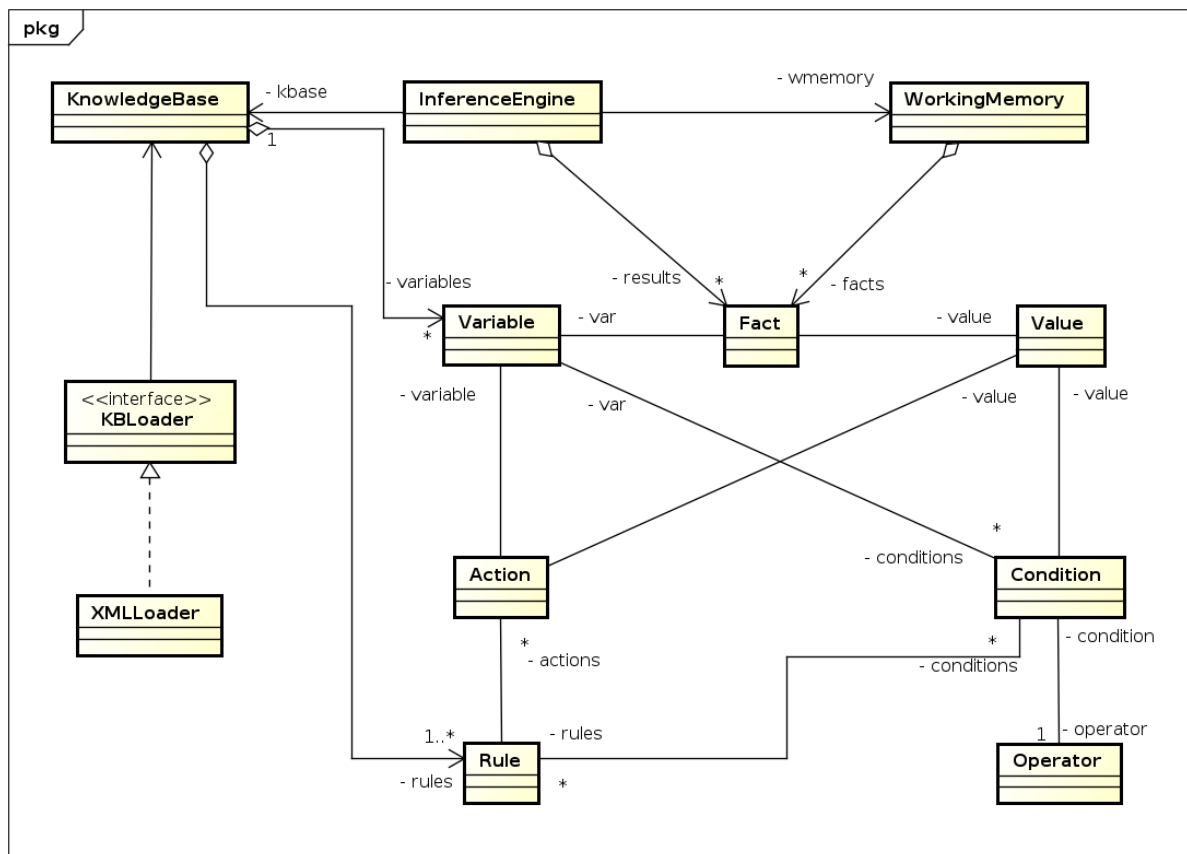


Figura 4.6: Diagrama de classes simplificado do *shell* SEI.

#### **4.4.2.2 Casos de uso**

Casos de uso são narrativas textuais que descrevem as interações dos usuários com um sistema na realização de uma tarefa, a fim de alcançar um determinado objetivo (LARMAN, 2007). Fornecendo uma narrativa sobre a utilização do sistema, os casos de uso são importantes para detectar os requisitos funcionais do sistema (FOWLER, 2005).

Na figura 4.7 são apresentados os casos de uso do SE desenvolvido neste projeto.

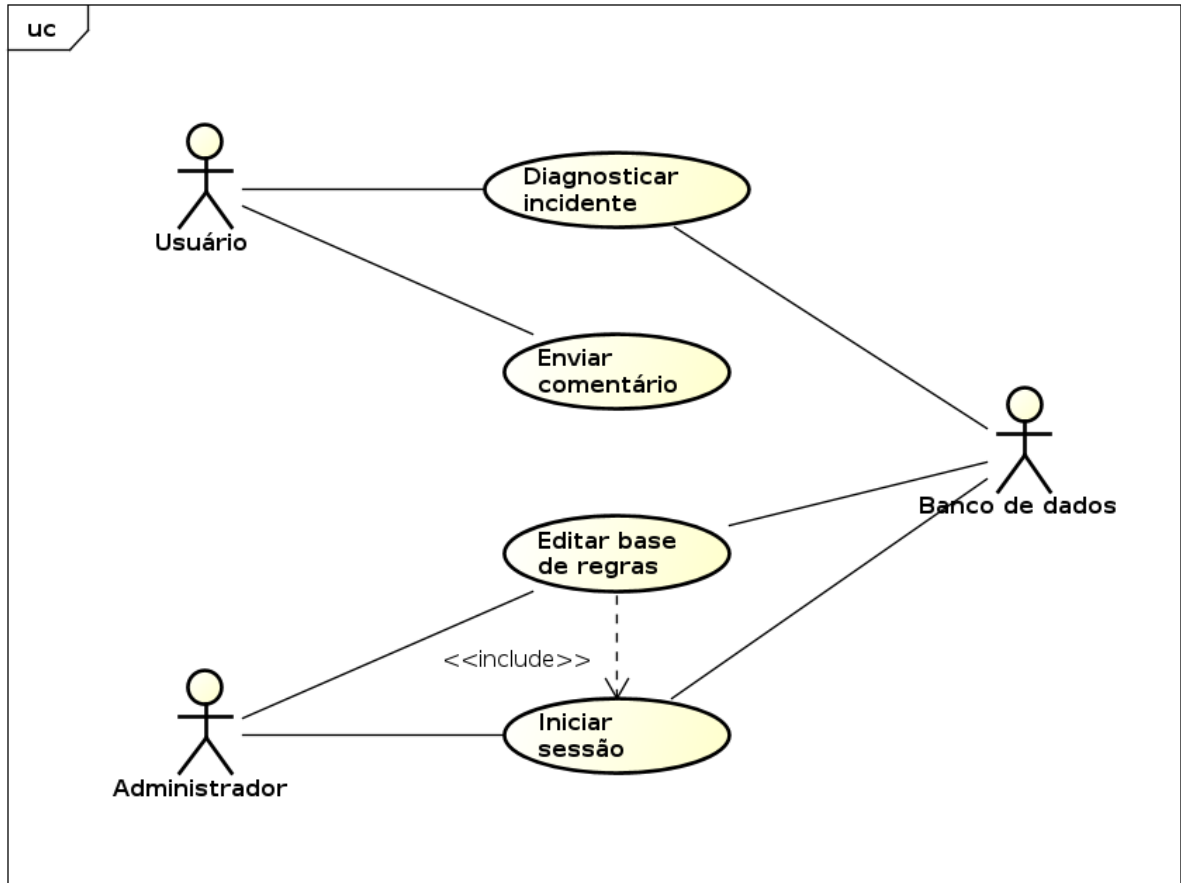


Figura 4.7: Diagrama simplificado de casos de uso do projeto SEI.

#### 4.4.3 Desenvolvimento do *shell*

Devido à indisponibilidade de uma ferramenta para desenvolvimento de SEs que satisfizesse os requisitos mínimos preestabelecidos no presente trabalho, situação descrita na seção 4.3, foi necessário construir desde o início uma nova ferramenta, capaz de fornecer a estrutura de componentes necessária para criação do SE.

A construção de um *shell* é justificável quando é planejada desde o início para ser reusada facilmente por diferentes SEs no futuro, possibilitando até mesmo o

acoplamento de novas interfaces com o usuário de modo transparente. Partindo desse fundamento, o desenvolvimento do SEI ocorreu em duas etapas: primeiramente foi desenvolvido o *shell*, com uso de uma interface básica de terminal para os testes; em seguida, deu-se a construção da interface com o usuário.

Para fins didáticos, e para simplificar ao máximo a implementação, a arquitetura do *shell* SEI seguiu os padrões clássicos tanto quanto possível; procurou-se seguir uma correspondência entre cada componente do SE e uma classe do projeto Java, de tal maneira que, por exemplo, o componente “máquina de inferência” corresponde diretamente à classe `InferenceMachine`, e assim por diante. Nas próximas três seções, será feita uma apresentação dos principais componentes do *shell*.

#### **4.4.3.1 Máquina de inferência**

A máquina de inferência do *shell* SEI tem um papel preponderante entre os demais componentes do SE, por ser a principal responsável por intermediar os relacionamentos entre eles. É ela que executa o processamento das variáveis e regras contidas na base de conhecimento, definindo a ordem em que deve ocorrer o processamento e efetuando as ativações necessárias a cada inserção de um fato na memória de trabalho.

A máquina de inferência é também responsável por: fazer a verificação necessária para determinar se a consulta deve continuar ou terminar, com base em parâmetros que definem até que ponto a busca deve ir; resolver qual a próxima variável a ser perguntada ao usuário; ao final da consulta, deve fornecer à interface os resultados obtidos.

Neste trabalho, foi decidido utilizar como estratégia de busca o encadeamento regressivo, pois é facilmente possível estabelecer uma hipótese ou objetivo, e os dados referentes ao problema devem ser obtidos pelo sistema.

A classe `InferenceMachine.java` concentra todos os métodos responsáveis pelo funcionamento da máquina de inferência, entre os quais os mais

importantes são descritos a seguir, acompanhados de diagramas de atividades UML para facilitar a compreensão.

- **execute:** é o método que inicia a operação da máquina de inferência. Seu primeiro passo é verificar se há uma variável objetivo sendo procurada e se essa variável já está na memória de trabalho; caso esteja, um novo fato é adicionado à lista de resultados. O segundo teste efetuado é se o objetivo atual é nulo (nenhum objetivo foi definido) ou se a pilha de variáveis está vazia; para ambas as condições, a ação é definir uma nova variável objetivo e chamar o método `processVariable` para processar a próxima variável da pilha. A terceira verificação, que também leva ao processamento da próxima variável, consiste em descobrir se a pilha de variáveis a perguntar está vazia e se ainda há variáveis a processar. Um diagrama de atividades para o método `execute` é apresentado na figura 4.8.



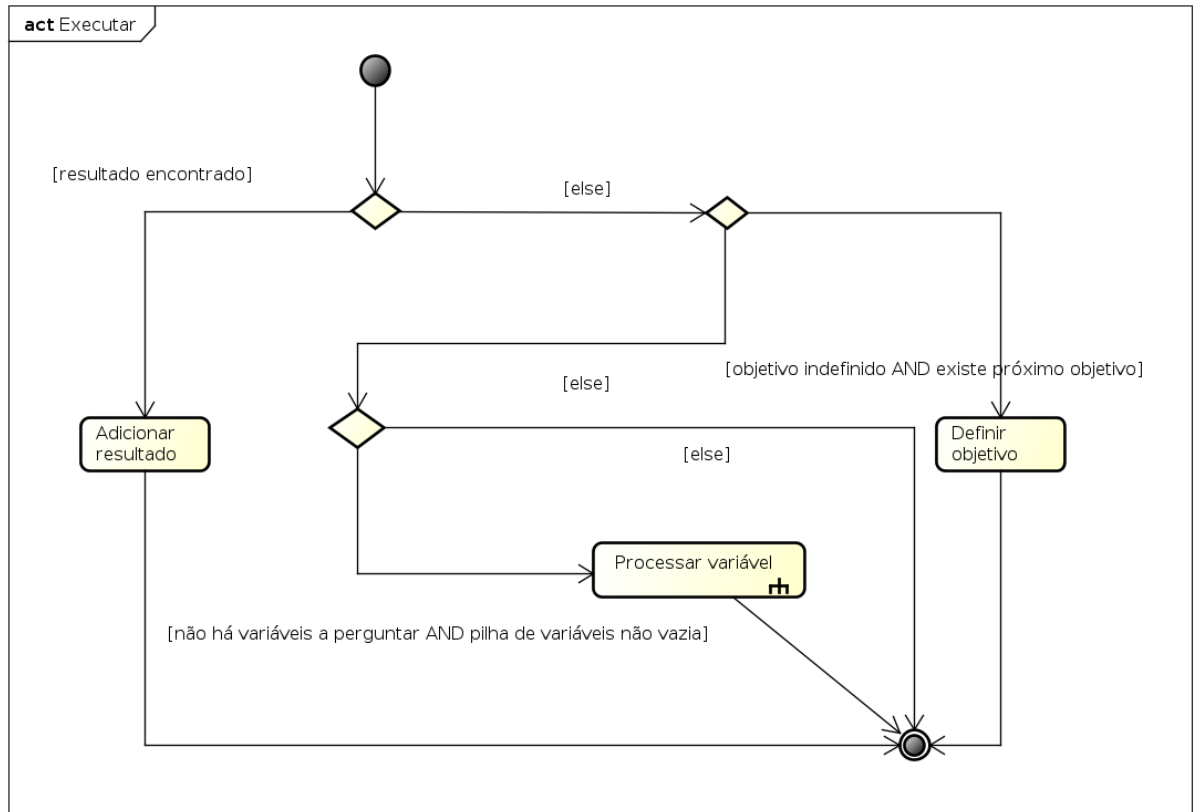


Figura 4.8: Diagrama de atividades para o método execute.

- `processVariable`: método que faz o processamento de uma variável da base de conhecimento, geralmente aquela que está no topo da pilha de variáveis. Em primeiro lugar, é feito um teste para identificar se a variável já foi processada ou não. Se já o tiver sido, e se a fila de regras cujas ações levam a ela estiver vazia, o processamento da variável é interrompido, a sua pilha de regras é abandonada e o processamento é direcionado para a próxima variável na pilha de variáveis; se o seu processamento já tiver sido iniciado e se a sua fila de regras não estiver vazia, a primeira regra da fila é processada; caso contrário, uma nova fila de regras é criada para a variável, efetuando-se em seguida o processamento da primeira regra da fila. Um

diagrama de atividades para o método `processVariable` é apresentado na figura 4.9.

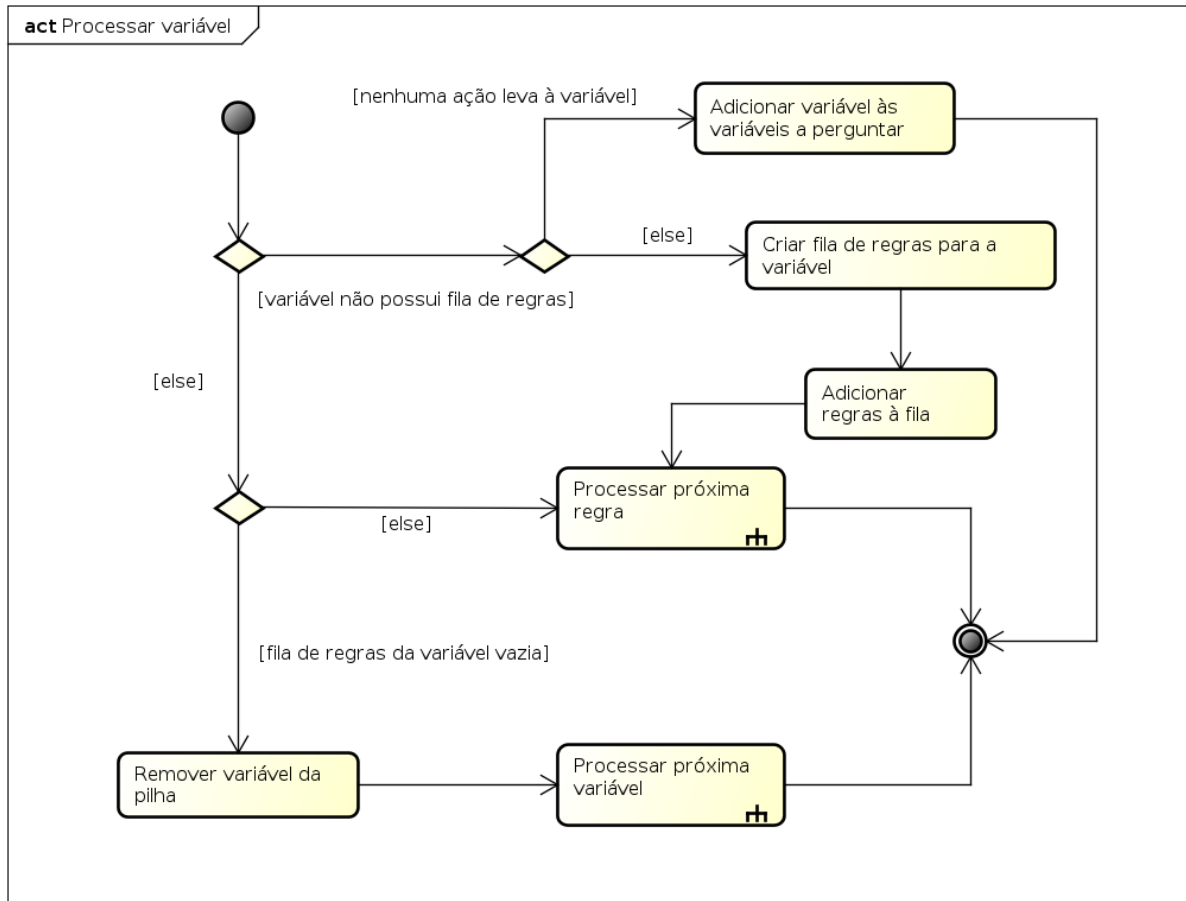


Figura 4.9: Diagrama de atividades para o método `processVariable`.

- `processRule`: esse método, como o nome indica, efetua o processamento de uma regra. O processamento se inicia com a seleção da primeira condição da regra que está sendo processada; se a variável da condição já está na memória de trabalho, é sinal de que a condição já foi processada, e, se a condição for falsa, toda a regra deve ser abandonada e o processamento deve ter continuidade com a próxima variável da pilha; se a condição for verdadeira, deve-se processar a próxima condição da regra. Uma situação

diferente ocorre quando a variável da condição ainda não está na memória de trabalho, ocasião em que essa variável deve ser colocada no topo da pilha de variáveis e processada imediatamente. Na figura 4.10 o método `processRule` tem o seu algoritmo apresentando em um diagrama de atividades.

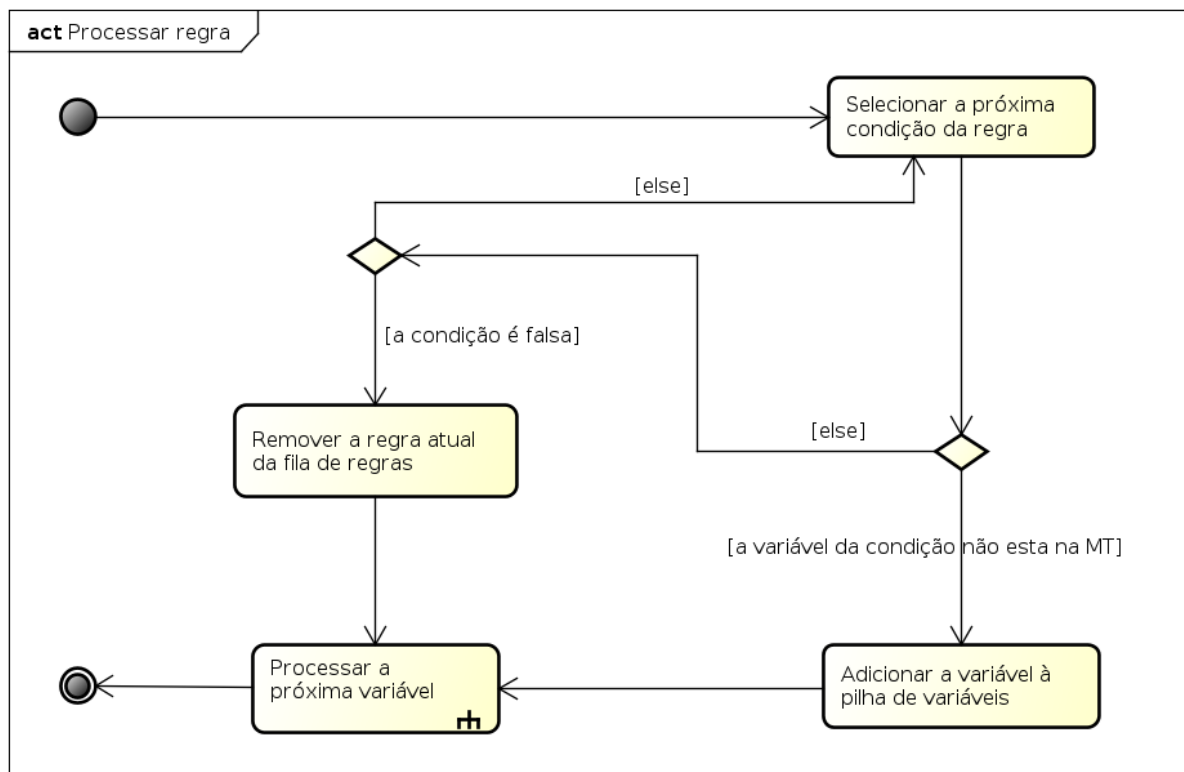


Figura 4.10: Diagrama de atividades para o método `processRule`.

#### 4.4.3.2 Memória de trabalho

A memória de trabalho do SEI cumpre o papel de armazenar, em tempo de execução, os fatos referentes à consulta atual. Os fatos são simples associações de um valor a uma variável e podem ser inseridos e removidos da memória de trabalho quando necessário. Além dos fatos, a memória de trabalho contém uma fila de

regras para cada variável em processamento, a pilha de variáveis propriamente dita a pilha de variáveis que precisam ser perguntadas ao usuário e uma pilha de variáveis que já foram perguntadas. A função de memorização dos elementos que dizem respeito à consulta atual ao SE, portanto, foi delegada na maior parte à memória de trabalho, atuando esta como importante auxiliar da máquina de inferência.

#### 4.4.3.3 Base de conhecimento

Na implementação da base de conhecimento do *shell* SEI, dedicou-se particular atenção tanto à flexibilidade da estrutura, para permitir a reutilização posterior, como à simplicidade, evitando que funcionalidades desnecessárias desviassem o foco do desenvolvimento. Conseqüentemente, a classe que representa a base de conhecimento, KnowledgeBase, tem apenas atributos e métodos de acesso a eles; ao mesmo tempo, a interface KBLoader permite flexibilidade e transparência quanto à origem dos dados. Pode-se carregar a base de conhecimento a partir de um arquivo escrito na linguagem XML, por exemplo, o que já conta com uma implementação básica distribuída com o próprio *shell*.

A base de conhecimento do SEI contém os seguintes elementos:

- **Informações:** são informações gerais sobre a base, como nome, descrição, autor. Uma informação também relevante incluída é o número de revisão da base de conhecimento; o propósito é que esse número seja incrementado a cada alteração na base.
- **Variáveis:** são elementos aos quais podem ser associados valores e que compõem, quando associados aos valores, as condições e ações de uma regra. Cada variável pode estar associada a apenas um tipo de valor: texto, booleano ou numérico.
- **Regras:** as regras formalizam o conhecimento contido na base de conhecimento. Seu lado esquerdo (LHS) é composto de *condições*, que por

sua vez se compõem de uma variável, um operador (por exemplo, “igual” ou “diferente”) e um valor. No seu lado direito, encontram-se *ações*, que na implementação atual podem ser apenas atribuições de um valor a uma variável. A figura 4.11 é um exemplo de uma regra criada para o *shell* SEI na linguagem XML.

```
<regra nome="Computador trava - reiniciar">
  <condicao variavel="problema.tipol" valor="outro" />
  <condicao variavel="computador.trava" valor="sim" />
  <condicao variavel="computador.reiniciado" valor="não" />
  <acao variavel="solucao" valor="computador.reiniciar" />
</regra>
```

Figura 4.11: Exemplo de regra definida em XML.

#### 4.4.4 Desenvolvimento da interface

Devido à centralização da equipe de suporte do IPREV em Florianópolis, decidiu-se que a interface com o usuário seria desenvolvida em plataforma Web, para permitir o acesso a partir de qualquer instalação física do IPREV, sem a necessidade de instalação de um novo aplicativo em cada computador; os usuários precisam apenas dispor da URL para acessar o sistema.

Durante as etapas iniciais de construção do *shell*, foi utilizada para interação com o sistema uma interface simples, baseada em linha de comando, cuja existência foi necessária para testar as funcionalidades do *shell* enquanto a interface Web, prevista para ser criada após o *shell*, ainda não estava pronta. Tão logo o *shell* foi considerado funcional, teve início o desenvolvimento da interface Web, que seguiu três etapas incrementais:

1. Desenvolvimento da interface de consulta ao SE.
2. Desenvolvimento da interface de edição da base de conhecimento.
3. Criação do banco de dados relacional e codificação das classes de acesso aos dados.

Enquanto o banco de dados relacional não estava disponível, a base de conhecimento permaneceu armazenada em um arquivo XML, sendo carregada por meio da classe XMLLoader e editada manualmente. Porém, a partir do momento em que o documento XML se aproximava de 1.000 linhas de código, tornou-se claro que seria conveniente dispor de uma interface própria para edição; isso e algumas experiências obtidas durante a fase de teste e avaliação do sistema motivaram a migração da base de conhecimento para um banco de dados relacional.

No desenvolvimento da interface foi usado o *framework* de apresentação Stripes, cuja arquitetura é baseada em ações (ActionBeans), que são classes Java que implementam uma interface em comum (ActionBean) e que recebem e processam os dados enviados pelo usuário (FENNEL, 2013). A figura 4.12 apresenta o diagrama de classes com as ações da interface Web; resalte-se a classe SeiActionBean, que implementa a interface ActionBean e serve de base para todas as demais.

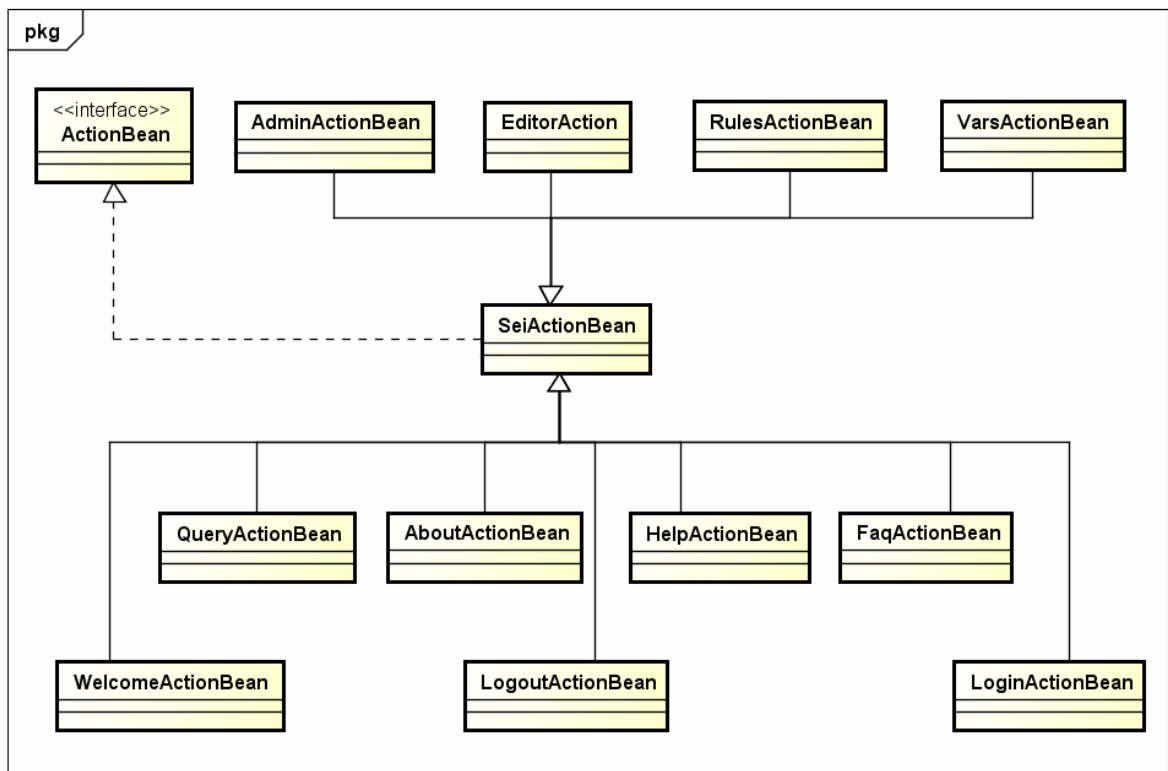


Figura 4.12: Diagrama de classes das ações da interface.

#### 4.4.4.1 Interface de consulta

A interface de consulta é o meio de interação entre o usuário e o SEI, permitindo que o SE obtenha do usuário as informações necessárias com respeito ao incidente atual e informe a ele as soluções encontradas. Conseqüentemente, os princípios que nortearam o seu desenvolvimento foram:

- **Facilidade de uso:** os usuários estão em busca de uma solução para um problema; nenhuma complicação desnecessária deve interpor-se nesse processo.
- **Clareza:** as mensagens devem ser escritas em linguagem compreensível, e os itens de controle e exibição devem ter características que permitam ao usuário identificar claramente a função de cada um.
- **Responsividade:** para garantir a rapidez no processo de consulta, a interface deve responder rapidamente aos comandos do usuário.

Uma consulta ao SEI começa com a página de boas-vindas do sistema, exibida na figura 4.13, a qual contém informações básicas e atalhos para acesso a informações sobre o sistema, ajuda e, principalmente, para iniciar uma consulta. Para iniciar uma consulta, o usuário deve clicar no botão “Procurar solução”, que recebe destaque pelo tamanho e pelas cores, evitando que o usuário se confunda.

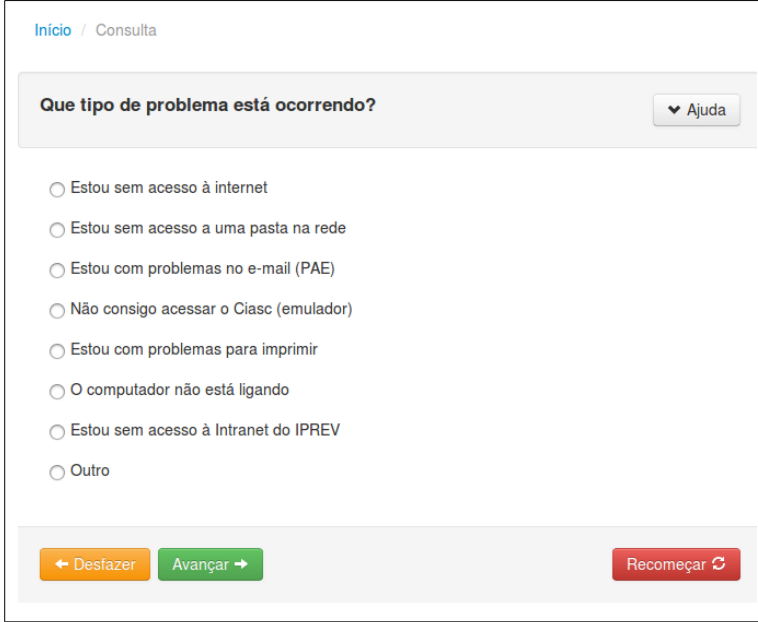
Deve-se perceber também dois elementos da interface que aparecem em quase todas as páginas do sistema: o cabeçalho, que contém os logotipos do IPREV e do SEI; e o rodapé, com os direitos autorais e uma pequena lista de atalhos.



Figura 4.13: Tela de boas-vindas do sistema SEI.

Na figura 4.14 tem-se a tela de abertura de uma nova consulta ao sistema. Nota-se em primeiro lugar o texto da pergunta, tendo ao lado o botão “Ajuda”, que mostra um pequeno texto com informações adicionais e esclarecimentos quanto à pergunta sendo feita. Logo abaixo da pergunta estão as opções de resposta; o usuário deve selecionar uma resposta e clicar no botão “Avançar”. Há ainda dois outros botões de controle: “Desfazer” volta à tela anterior, removendo a última resposta fornecida; “Recomeçar” apaga todas as respostas dadas até então e reinicia a consulta.





Início / Consulta

Que tipo de problema está ocorrendo? Ajuda

Estou sem acesso à internet

Estou sem acesso a uma pasta na rede

Estou com problemas no e-mail (PAE)

Não consigo acessar o Ciasc (emulador)

Estou com problemas para imprimir

O computador não está ligando

Estou sem acesso à Intranet do IPREV

Outro

Desfazer Avançar Recomeçar

Figura 4.14: Tela inicial de consulta por uma solução.

A figura 4.15 exibe uma outra pergunta do sistema ao usuário, desta vez sobre uma variável booleana. Note-se que o botão “Ajuda” foi acionado, ativando a exibição do texto de ajuda, que contém uma imagem de um cabo de rede; essa funcionalidade foi resultado da primeira fase de testes do sistema, a ser considerada na seção 4.5.

Quando a consulta chega ao fim, as soluções, se encontradas, são exibidas na página de resultados (figura 4.16). Cada solução contém uma descrição sucinta e um passo a passo, que orienta a execução da solução. O passo a passo, assim como o texto de ajuda das perguntas, pode conter imagens e outros elementos HTML.

Se o sistema não encontrar nenhum resultado, será exibida uma tela informando que o usuário pode tentar uma nova consulta ou entrar em contato com o suporte técnico por telefone ou e-mail (figura 4.17).

Início / Consulta

**O cabo de rede está conectado?** Ajuda

Uma imagem de um cabo de rede:



Sim  
 Não

← Desfazer Avançar → Recomeçar ↻

Figura 4.15: Tela de consulta com ajuda ativada.

Consulta / Resultado

**Solução encontrada!**  
O sistema encontrou uma ou mais possíveis soluções! Veja abaixo.

**Solução 1**

Defina a Intranet do Iprev como sua página inicial.

1. Abra o seu navegador.
2. Clique no menu Ferramentas > Opções da internet.
3. Insira o endereço <http://intranet.iprev.sc.gov.br>.
4. Clique em Ok para salvar as configurações.

← Voltar Recomeçar ↻

Figura 4.16: Tela de resultado, com uma solução encontrada e passo a passo para execução.

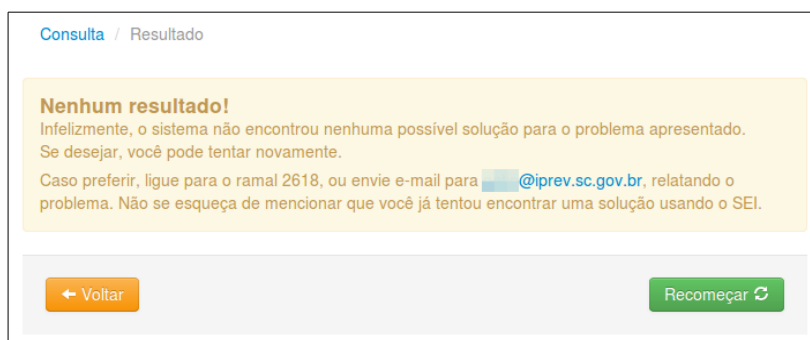


Figura 4.17: Tela que é exibida quando não há nenhum resultado.

#### 4.4.4.2 Interface de edição da base de conhecimento

A interface para edição da base de conhecimento foi criada para facilitar o gerenciamento das variáveis e regras que compõem o conhecimento disponível ao sistema SEI. Possibilita ainda a alteração das informações a respeito da base e a criação e manutenção dos textos de ajuda das perguntas e das soluções.

Para acessar a interface de edição é preciso antes iniciar uma sessão como administrador do sistema. Então, se tem acesso às abas de edição, das quais a primeira (figura 4.18) é a que permite alterar as informações gerais do sistema. Todos os campos dessa aba são obrigatórios, com exceção do campo “Revisão”, que é gerado automaticamente pelo banco de dados e não pode ser alterado.

Figura 4.18: Tela inicial de edição.

A segunda aba possibilita a manutenção das variáveis da base de conhecimento (figura 4.19). As variáveis são ordenadas em uma lista, que exibe uma caixa de seleção para cada variável, o nome da variável, o tipo e se ela é um objetivo; por fim, alguns ícones para alteração da variável estão disponíveis em cada linha. No topo da lista de variáveis há uma barra de ferramentas fixa, com botões para adicionar uma variável, salvar a ordem das variáveis e remover as variáveis selecionadas.

Ao se clicar no botão “Nova variável” ou no ícone para editar uma variável, será exibido diálogo de edição e alteração de variável. Esse diálogo possui um campo obrigatório, “Nome da variável”, uma pergunta, que é necessária apenas para as variáveis que serão perguntadas ao usuário, o tipo do valor que a variável requer, e se é um objetivo. Para variáveis do tipo “texto”, a lista de opções que podem ser escolhidas aparecerá logo abaixo da caixa de seleção; a lista de opções funciona tanto para que o usuário escolha uma durante a consulta, como para que o sistema selecione uma resposta, caso seja uma variável objetivo.

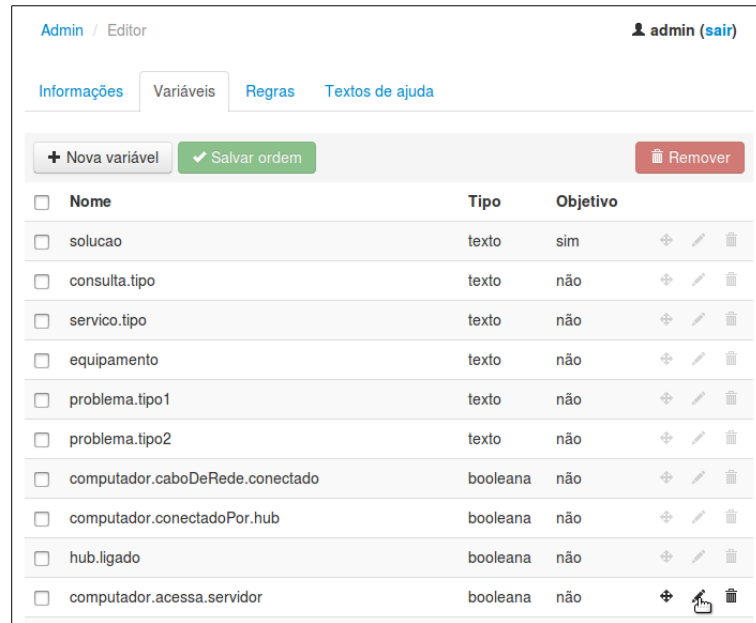


Figura 4.19: Tela de edição de variáveis.

Já a terceira aba possibilita a manutenção das regras da base de conhecimento (figura 4.20). As regras são ordenadas em uma lista, que exibe uma caixa de seleção para cada regra, o nome da regra e alguns ícones para alteração da regra. No topo da lista de regras há uma barra de ferramentas fixa, com botões para adicionar uma regra, salvar a ordem das regras e remover as regras selecionadas.

Ao se clicar no botão “Nova regra” ou no ícone para editar uma regra, será exibido diálogo de edição e alteração de regra. Esse diálogo possui um campo obrigatório, “Nome da regra”, e duas listas: lista de condições e lista de ações. Cada condição possui três campos: “variável”, “operador” e “valor”, dos quais o primeiro e o último têm o recurso de preenchimento automático; cada ação possui os mesmos campos, com exceção de “operador”.

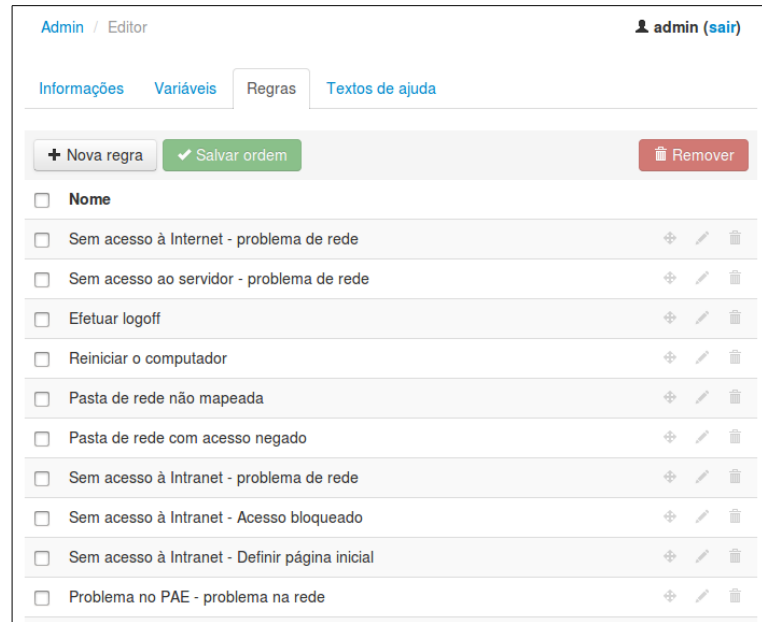


Figura 4.20: Tela de edição de regras.

A quarta e última aba exibe a edição dos textos de ajuda (figura 4.21). Esses são os textos que são mostrados, na interface de consulta, quando se clica no botão “Ajuda” ao lado de uma pergunta, ou quando uma resposta é encontrada pelo sistema. Para alterar um texto de ajuda, basta preencher o primeiro campo com o nome da variável, no caso da ajuda das perguntas; se o objetivo for alterar um texto de resposta, deve-se preencher tanto o campo “Variável” como o campo “Valor”. Os textos de ajuda podem ser formatados com o uso de HTML.

Admin / Editor admin (sair)

Informações Variáveis Regras **Textos de ajuda**

Variável\*

Valor

Conteúdo\*

```
<ol>
<li>Acesse o endereço <a href="http://meuip.ciasc.gov.br">http://meuip.ciasc.gov.br</a>.
<li>Anotar a informação "seu IP".
<li>Ligue para o ramal 2618, ou envie e-mail para getin@iprev.sc.gov.br, informando o endereço IP que você anotou.
</ol>
```

Figura 4.21: Tela de edição e criação dos textos de ajuda.

#### 4.4.4.3 Implantação do banco de dados

A implantação do banco de dados relacional foi a última etapa do desenvolvimento da interface. Com essa etapa, o editor de regras ganhou maior flexibilidade, tornando-se possível a adição dos textos de ajuda, que são uma funcionalidade exclusiva da interface Web. Além disso, o editor recebeu autonomia em relação à base de conhecimento do *shell*, podendo alterar as informações diretamente pela interface de dados.

#### 4.5 Fase do teste e avaliação

Assim que o primeiro protótipo foi considerado pronto, foi aplicada a primeira fase de teste e avaliação ao SE, com o objetivo de avaliar a eficiência e detectar possíveis falhas na representação do conhecimento e na implementação. Essa primeira fase foi restrita a poucos usuários, devido ao estágio ainda incipiente do

desenvolvimento, o que, se houvesse ampla abrangência, poderia causar rejeição precipitada por parte dos futuros usuários.

O modo de avaliação no primeiro momento foi o de observação. O engenheiro do conhecimento instruiu cada usuário a acessar a URL do sistema e efetuar alguns passos preestabelecidos; especificamente, ele deveria iniciar uma consulta para encontrar uma solução para um problema definido pelo engenheiro do conhecimento. Ao mesmo tempo, cada dúvida ou dificuldade encontrada em cada passo era anotada. Completado cada teste, o usuário era estimulado a sugerir mudanças no sistema. Os resultados sumarizados da primeira etapa de testes estão no quadro 4.5.1.

De modo geral, os usuários tiveram dificuldades quanto a termos técnicos e passos para obter informações e aplicar soluções. Quando o funcionamento do sistema era diferente do esperado, houve indução a erro; o botão avançar, por exemplo, não precisava ser acionado após a seleção da opção, mas os usuários tenderam mesmo assim a clicar nele, por ser esse o comportamento esperado. Cumpre ressaltar que o nível de conhecimento das pessoas que realizaram os testes pode ser definido entre o básico e o médio, justamente o público para o qual o SEI foi desenvolvido, ocasionando essas dificuldades. Na seção 4.6 são relatadas as mudanças feitas no sistema para lidar com os pontos levantados nessa etapa de testes.



Quadro 4.5.1: Resultados da primeira etapa de testes.

<b>Indivíduo</b>	<b>Testes</b>	<b>Dificuldades</b>	<b>Sugestões</b>
A	3	Encontrou dificuldade em responder algumas perguntas por não saber como conseguir a informação.	Sugeriu um detalhamento ou o uso de imagens para ajudar na identificação de termos desconhecidos.
B	2	Encontrou dificuldade em responder algumas perguntas por não saber como conseguir a informação. Também observou que não conhecia alguns itens pelo nome.	Sugeriu o uso de imagens para ajudar na identificação de termos desconhecidos.
C	2	Encontrou dificuldade em responder algumas perguntas por não saber como conseguir a informação. Informou que não sabia aplicar a solução proposta. Clicou no botão “Avançar” para enviar a resposta, mesmo não sendo necessário.	Sugeriu um passo a passo, se possível gráfico, para ajudar na aplicação da solução.

Corrigidos os problemas observados no primeiro protótipo, foi lançada a segunda etapa de teste e avaliação, desta vez com um número de usuários muito maior. Optou-se, para garantir maior abrangência, por permitir aos próprios usuários do sistema, isto é, aos servidores do IPREV, testarem o SEI e relatarem as suas experiências por meio de um questionário de avaliação. O convite para os testes foi enviado por e-mail para todos os servidores do IPREV.

Essa avaliação incluiu questões sobre a facilidade de uso do sistema, a clareza das mensagens, a rapidez do carregamento das páginas e, especialmente, se o respondente usaria o sistema para procurar uma solução para um incidente de informática que encontrasse. Além dessas questões, foi questionado também o local de trabalho do servidor, se prédio-sede (Florianópolis) ou agências e coordenadorias. Os resultados obtidos nessa etapa estão sumarizados no quadro 4.5.2.

Quadro 4.5.2: Resultados da segunda etapa de teste e avaliação.

<b>Local de trabalho</b>	<b>Sede (Florianópolis)</b>	<b>Agências e coordenadorias</b>
<b>Usaria o sistema?</b>	100%	100%
<b>Considera que o sistema é fácil de usar?</b>	100%	92%
<b>As mensagens e elementos visuais são claros?</b>	100%	100%
<b>O carregamento das páginas é rápido?</b>	100%	100%
<b>Participantes</b>	11 (65%)	6 (35%)

Nota-se que o número total de respondentes foi 24, o que pode ser considerado abrangente, haja vista que o IPREV tem entre 200 e 300 servidores ativos e que o período de testes foi de apenas dois dias.

A avaliação dos servidores foi totalmente positiva, validando o sistema apresentado, embora algumas considerações e esclarecimentos precisem ser feitos. Primeiro, quanto à porcentagem dos usuários que consideraram as mensagens e os elementos visuais claros, 13% informaram que encontraram alguma dificuldade de compreensão; isso se deve, provavelmente, ao fato de que não havia textos de ajuda para todas as perguntas e soluções no momento dos testes.

Já quanto à rapidez do carregamento das páginas, houve uma diferença entre as respostas dos servidores das agências e coordenadorias, e as dos servidores lotados na sede. Nas coordenadorias e agências, 67% dos avaliadores consideraram o carregamento “rápido”, e 33% “muito rápido”; na capital, 75% consideraram “muito rápido” o carregamento, ao passo que os restantes 25% consideraram apenas rápido. Os motivos para tal diferença podem estar no fato de o servidor que hospeda o SEI estar localizado no prédio-sede, e na menor banda de rede disponível atualmente para as agências e coordenadorias.

Na segunda etapa de teste, também foram colhidas sugestões dos usuários para melhorias do sistema, das quais algumas estão listadas abaixo:

- Enriquecimento do sistema com mais opções de problemas e soluções.
- Em caso de problemas mais complexos, agregar ao sistema SEI a possibilidade de o técnico resolver o problema através de "acesso remoto".
- Incluir a possibilidade de o usuário responder que não sabe a resposta.

#### **4.6 Fase da revisão**

Após cada etapa de teste e avaliação, o sistema foi revisado, a fim de corrigir as falhas e implantar as alterações julgadas necessárias.

Na primeira etapa de avaliação, a principal dificuldade encontrada foi a falta de conhecimento técnico avançado para conseguir responder as perguntas feitas pelo sistema. Foi sugerido então o uso de imagens para os termos desconhecidos pelos usuários, e guias passo a passo para obter as respostas e efetuar as soluções sugeridas pelo sistema. Ambas as sugestões foram acatadas por meio da funcionalidade dos “textos de ajuda”, disponível no segundo protótipo.

Também ocorreram problemas de usabilidade quanto ao comportamento do sistema, especialmente quanto à presença inútil do botão “Avançar”, quando a simples seleção de uma opção submetia a resposta. No segundo protótipo, decidiu-se requerer que o botão fosse acionado, evitando surpresas para o usuário.

Quanto às sugestões apresentadas na segunda etapa de avaliação, é perfeitamente viável implementá-las nos futuros protótipos do SEI. O enriquecimento do sistema com mais problemas e soluções é um processo contínuo, que está em andamento.

## 5 CONCLUSÕES E TRABALHOS FUTUROS

### 5.1 Conclusões

O desenvolvimento de um SE de autoatendimento para diagnóstico de incidentes de TI e identificação de soluções é, de acordo com este trabalho, plenamente viável com as tecnologias existentes na atualidade, encontrando inclusive ampla aceitação por parte dos usuários. Os requisitos estabelecidos para o desenvolvimento foram de fundamental importância para que isso fosse possível.

O estabelecimento do requisito de que todas as ferramentas utilizadas no desenvolvimento do sistema SEI fossem *software* livre, ainda que tenha trazido alguma dificuldade na escolha de um *shell* para SEs, demonstrou ser uma escolha acertada, pelo grande número de bibliotecas livres que foram utilizadas no projeto, permitindo que os custos com licenciamento permanecessem nulos.

Quando se dispõe de um histórico de problemas e soluções adequado, o trabalho torna-se menos dificultoso; porém, essa nem sempre é a realidade na maioria das organizações, especialmente nos órgãos públicos. Neste trabalho, a análise do histórico de atendimentos teve de adicionar informações faltantes, o que, se a base de soluções fosse mais detalhada, poderia ter sido evitado. Não obstante, a análise permitiu que fosse levada a efeito a modelagem da base de conhecimento do sistema.

No processo de desenvolvimento de um SE, o uso de uma ferramenta apropriada preexistente (*shell*) é conveniente, mas não se deve atrelar o SE de tal forma à ferramenta que a sua adequação ao problema seja comprometida. É necessário, portanto, escolher a ferramenta de acordo com o problema, e não o contrário; a modelagem e o desenvolvimento de uma ferramenta própria foi considerada a melhor solução no presente trabalho.

A escolha da plataforma Web durante o processo de desenvolvimento para a criação da interface permitiu uma grande consistência entre os diferentes computadores e sistemas operacionais do ambiente de implantação, haja vista que

todos acessam a mesma base de conhecimento, sem necessidade de instalação de um novo aplicativo.

Por fim, os testes demonstraram que não é possível desenvolver um sistema, especialmente um SE, sem ter em grande consideração o usuário final. Foi por meio da observação das suas dificuldades e necessidades com o primeiro protótipo que se obteve o aperfeiçoamento do SEI, o que foi validado pela ampla aceitação do sistema na avaliação do segundo protótipo feita pelos usuários.

## 5.2 Principais contribuições

Neste trabalho, foi desenvolvido um SE baseado em regras para encontrar soluções para incidentes de informática por meio de autoatendimento. Demonstrou-se que é possível extrair a maior parte do conhecimento necessário a partir de um histórico de incidentes, desde que seja feita a validação por um especialista.

Como base desse SE, foi construído um *shell* reutilizável, que emprega o raciocínio guiado a objetivos como meio de inferência. Acima do *shell*, desenvolveu-se uma interface Web de consulta que pode ser acessada de qualquer computador com acesso à rede da organização, além de um editor da base de conhecimento plenamente funcional, que não depende do aprendizado de uma nova linguagem para utilização.

## 5.3 Trabalhos futuros

Propõe-se como continuação do presente trabalho que se estude a viabilidade da agregação do raciocínio baseado em casos como complemento ao raciocínio baseado em regras utilizado atualmente.

Seria relevante ainda que fosse implantada a possibilidade de salvar a consulta atual, para que seja viável, por exemplo, reiniciar o computador e retomar a consulta em seguida no ponto em que parou.

A propósito das sugestões apresentadas pelos usuários na segunda etapa de avaliação, é especialmente relevante que se estude não só a possibilidade de o usuário informar que não sabe responder a pergunta, como também a inclusão de graus de confiança para cada resposta.

## REFERÊNCIAS

VALLE, James Della. **Faltam profissionais e sobram oportunidades em TI.**

Disponível em:

<<http://veja.abril.com.br/noticia/vida-digital/faltam-profissionais-e-sobram-oportunidades-em-ti>>. Acesso em: 26 nov. 2012.

SEF. **Atos que aumentem gasto com folha no Estado estão suspensos até o fim do ano.** Disponível em:

<<http://www.sef.sc.gov.br/noticias/atos-que-aumentem-gasto-com-folha-no-estado-es-estao-suspensos-ate-o-fim-do-ano>>. Acesso em: 26 nov. 2012.

RABUSKE, Renato A. **Inteligência artificial.** Florianópolis: Ed. da UFSC, 1995.

LUGER, George F. **Inteligência artificial: estruturas e estratégias para a solução de problemas complexos.** Porto Alegre: Bookmann, 2004.

REZENDE, Solange O. **Sistemas inteligentes: Fundamentos e aplicações.** São Paulo: Manole, 2003.

GIARRATANO, Joseph C.; RILEY, Gary. **Expert Systems.** Boston: PWS Publishing Company, 1993. 644 p.

KELLER, Robert. **Tecnologia de sistemas especialistas: desenvolvimento e aplicação.** São Paulo: Makron, 1991.

PY, Mônica X. **Sistemas Especialistas: Uma introdução.** 1 ed. Porto Alegre: 2000. 10p. Disponível em:  
<<http://www.inf.ufrgs.br/gppd/disc/cmp135/trabs/mpy/sistemasespecialistas.pdf>>.  
Acesso em: 29 nov. 2012.

FORGY, Charles L. **On the Efficient Implementation of Production Systems.** 178 p. Tese (Doutorado) – Department of Computer Science, Carnegie-Mellon University, Pittsburgh, 1979.

DOOREMBOS, Robert B. **Production Matching for Large Learning Systems.** 194 p. Tese (Doutorado) – Department of Computer Science, Carnegie-Mellon University, 1995.

MIRANKER, Daniel P. **TREAT: A Better Match Algorithm for AI Production Systems**. Austin: AAI-87 Proceedings, 1987. Disponível em: <<http://www.aaai.org/Papers/AAAI/1987/AAAI87-008.pdf>>. Acesso em: 06 dez. 2012.

BATORY, Don. **The LEAPS Algorithms**. Austin: University of Texas, 1994.

FRIEDMAN-HILL, Ernest J. **Jess: the Rule Engine for the Java platform**. 2008. Disponível em: <<http://www.jessrules.com/jess/docs/71/index.html>> Acesso em: 21 nov. 2008.

STRAUSS, Martin. **Jess: the Java Expert System Shell**. Disponível em: <[http://www.dfki.de/~kipp/seminar\\_ws0607/reports/martin\\_strauss.pdf](http://www.dfki.de/~kipp/seminar_ws0607/reports/martin_strauss.pdf)>. Acesso em: 29 nov. 2012.

BUDKE, Gerson F.; MAYER, Daniel. **Jess: The Rule Engine for the Java Platform**. Disponível em: <[http://www.das.ufsc.br/~gb/pg-ia/Jess08/Artigo\\_Jess\\_Daniel\\_Gerson.pdf](http://www.das.ufsc.br/~gb/pg-ia/Jess08/Artigo_Jess_Daniel_Gerson.pdf)>. Acesso em: 29 de nov. 2012.

DROOLS. **Drools**. Disponível em: <<http://legacy.drools.codehaus.org/>>. Acesso em: 29 out. 2012.

JBOSS. **Drools Expert User Guide**. Disponível em: <<http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html/index.html>>. Acesso em: 25 out. 2012.

PROCTOR, Mark. **Backward Chaining emerges in Drools**. Disponível em: <[http://planet.jboss.org/post/backward\\_chaining\\_emerges\\_in\\_drools](http://planet.jboss.org/post/backward_chaining_emerges_in_drools)>. Acesso em: 29 out. 2012.

LANGE, Thales. **Sistema especialista probabilístico aplicado à gestão da manutenção da distribuição de energia elétrica**. 2001. 162 p. Dissertação (Mestrado) - Universidade do Vale do Itajaí, São José, 2011.

BROTTO, Osvaldo C. **Estruturação de um sistema especialista probabilístico de avaliação de sucesso ou insucesso na abertura de novos negócios**. 126 p. Dissertação (Mestrado) - Universidade Federal de Santa Maria, Santa Maria, 2009.

MOSSIN, E. A. **Diagnóstico Automático de Redes Profibus**. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2012.



GROSSMANN JÚNIOR, Helmuth. **Um sistema especialista para auxílio ao diagnóstico de problemas em computadores utilizando raciocínio baseado em casos**. 109 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Florianópolis, 2002.

ZIELINSKI, Fabio; BORTOLETO, Silvio. **Aplicação de RBC em sistema de Help Desk: estudo de caso Radsystem**. Florianópolis: SEGeT – Simpósio de Excelência em Gestão e Tecnologia, 2007.

WANG, Dingding et al. **IHelp: An Intelligent Online Helpdesk System**. [S.I.]: IEEE Transactions on Systems, Man, And Cybernetics Society, 2010.

WANGENHEIM, Christiane G. von; WANGENHEIM, Aldo von. **Raciocínio baseado em casos**. Barueri: Manole, 2003.

OGC. **ITIL V3: Service operation**. [S.I.]: The Stationery Office, 2007.

LEVENSHTEIN, Vladimir I. **Binary codes capable of correcting deletions, insertions, and reversals**. [S.I.]: Soviet Physics Doklady, 1966.

PRESSMAN, Roger S. **Engenharia de Software**. Porto Alegre: AMGH, 2011.

RAMOS, Ricardo Argenton. **Treinamento prático em UML**. São Paulo : Digerati Books, 2006.

FOWLER, Martin. **UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos**. Porto Alegre: Bookman, 2005.

LARMAN, Craig. **Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo**. Porto Alegre: Bookman, 2007.

FENNEL, Tim. **Stripes Framework**. Disponível em:  
<<http://www.stripesframework.org/display/stripes/Home>>. Acesso em: 20 mai. 2013.